Zur Analyse der Optimierungszeit randomisierter Suchheuristiken für kombinatorische Probleme

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik
von

Christian Thyssen

Dortmund
2010

Tag der mündlichen Prüfung:	31. August 2010
Dekan:	Prof. Dr. Peter Buchholz
Gutachter:	Dr. Thomas Jansen und Prof. Dr. Günter Rudolph

Danksagung

Diese Dissertation wäre ohne die Unterstützung der nachfolgend genannten Kollegen nicht möglich gewesen. Ich bin Ingo Wegener (04.12.1950 – 26.11.2008) zu Dank verpflichtet, der mir die Möglichkeit gegeben hat, meine Arbeit an seinem Lehrstuhl für effiziente Algorithmen und Komplexitätstheorie anzufangen. Weiterhin möchte ich mich bei Thomas Jansen bedanken, der mir (trotz der räumlichen Distanz) in den letzten Monaten auf mannigfaltige Weise geholfen hat. Viele der Veröffentlichungen, auf denen diese Dissertation beruht, sind gemeinsam mit anderen entstanden. Daher möchte ich an dieser Stelle ausdrücklich Benjamin Doerr, Anton Eremeev, Tobias Friedrich, Thomas Jansen, Frank Neumann, Dirk Sudholt, Madeleine Theile und Christine Zarges für die fruchtbare Zusammenarbeit danken. Besonders hervorheben möchte ich Frank Neumann, der mich besonders in der schwierigen Anfangsphase unterstützt hat. Auch die Förderung durch die Deutsche Forschungsgemeinschaft (DFG), von der ich als Mitarbeiter des Sonderforschungsbereich 531 „Design und Management komplexer technischer Prozesse und Systeme mit Methoden der Computational Intelligence“ profitiert habe, möchte ich hier würdigen.

Nicht zuletzt danke ich meiner Frau Nadine für ihre stete Unterstützung.

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Die Fähigkeit, Probleme lösen zu können, macht intelligente Wesen aus. Es drängen sich in diesem Zusammenhang zwei Fragen auf: Was ist ein Problem und wie löst man es? Die theoretische Informatik liefert auf beide Fragen präzise Antworten, die natürlich nicht allen Problemen gerecht werden können. Ein algorithmisches Problem Π ist eine Relation $\Pi \subseteq X \times Y$, wobei die Mengen X und Y alle zulässigen Eingaben bzw. Ausgaben umfassen. Dabei ist ein geordnetes Paar $(x, y) \in X \times Y$ genau dann in Π enthalten, wenn die Eingabe x und die Ausgabe y zueinander passen. Ein Algorithmus A löst ein Problem Π genau dann, wenn A für jede Eingabe $x \in X$ nach endlich vielen Rechenschritten eine Ausgabe $y \in Y$ mit $(x, y) \in \Pi$ präsentiert. Der Entwurf und die Analyse solcher Algorithmen stehen im Zentrum der (theoretischen) Informatik.

Wir verdeutlichen die Konzepte anhand eines anschaulichen Beispiels. Nehmen wir an, wir möchten n Städte durch Telefonleitungen miteinander verbinden, sodass jeder Bürger mit jedem anderen Bürger telefonieren kann. Außerdem möchten wir den Ressourcenverbrauch minimieren, den das Verlegen der Telefonleitungen erfordert. Der erste Schritt besteht darin, die Eingabe zu formalisieren. Alle zulässigen Eingaben sind durch gewichtete, zusammenhängende, ungerichtete Graphen $G = (V, E, w)$ gegeben, wobei $V \subseteq \{v_1, v_2, \dots\}$ die Knotenmenge, $E \subseteq \{V' \subseteq V \mid |V'| = 2\}$ die Kantenmenge und $w\colon E \to \mathbb{R}^+$ die Gewichtsfunktion bezeichnen. Die Menge V enthalte die Knoten $v_1, \dots, v_n$, die die Städte repräsentieren, und die Menge E enthalte eine Kante $\{v_i, v_j\}$ genau dann, wenn die i-te und die j-te Stadt durch eine Telefonleitung miteinander verbunden werden können. Außerdem weise die Gewichtsfunktion w einer Kante $\{v_i, v_j\}$ die Kosten der im letzten Satz genannten Telefonleitung zu. Alle zulässigen Ausgaben sind durch Kantenmengen $T \subseteq \{V' \subseteq \{v_1, v_2, \dots\} \mid |V'| = 2\}$ gegeben. Eine Ausgabe $T = \{e_1, \dots, e_\ell\}$ passt genau dann zu einer Eingabe $G = (V, E, w)$, wenn $T \subseteq E$ gilt und $G = (V, T, w)$ zusammenhängend ist. Außerdem darf $w(e'_1) + \dots + w(e'_{\ell'}) < w(e_1) + \dots + w(e_\ell)$ für keine solche Teilmenge $T' = \{e'_1, \dots, e'_{\ell'}\}$ gelten. Insgesamt sind wir auf das aus vielen Lehrbüchern bekannte Problem, einen minimalen aufspannenden Baum zu finden, gestoßen.

Beispielsweise der Algorithmus von Kruskal kann das im letzten Absatz vorgestellte Problem lösen (siehe [Kruskal, 1956]). Der Algorithmus beginnt mit der leeren Kantenmenge $T = \emptyset$ und ergänzt diese schrittweise. Solange es möglich ist, wählt er aus den noch nicht gewählten Kanten $E \setminus T$ eine kürzeste Kante, die mit den bereits gewählten Kanten T keinen Kreis bildet. Man kann beweisen, dass die gegebene Eingabe und die so berechnete Ausgabe zueinander passen, d. h., dass der Algorithmus

das Problem löst. Außerdem kann man beweisen, dass die Berechnung der Ausgabe nicht mehr als $\mathcal{O}(|E| \cdot \log |E|)$ Rechenschritte benötigt, da das erforderliche Sortieren der Kantengewichte die Laufzeit dominiert, wenn eine geeignete Implementierung zugrunde gelegt wird.

In den letzten beiden Absätzen haben wir beispielhaft demonstriert, wie man ein tatsächliches Problem auf klassische Weise mit Methoden aus der Informatik lösen kann. Wir haben gezeigt, wie das Ausgangsproblem auf ein algorithmisches Problem reduziert und wie ein Algorithmus für das entstandene Problem entworfen und analysiert werden kann. Um eine passende Ausgabe effizient zu berechnen, konnte der Algorithmus dabei uneingeschränkt auf die Eingabe zugreifen. Diese komfortable Situation entspricht häufig leider nicht der Realität. Nehmen wir beispielsweise an, dass eine zu optimierende Funktion implizit durch ein Simulationsmodell gegeben ist. In diesem Fall kann die Güte einer Konfiguration einzig durch einen Simulationslauf bestimmt werden. Ein Algorithmus kann also nicht auf die Funktionsvorschrift der zu optimierenden Funktion zugreifen; die Funktion ist anschaulich durch einen schwarzen Kasten (auf Englisch „black box“) gegeben, in dem die Funktionsvorschrift verborgen liegt. Um Informationen zu gewinnen, ist der Algorithmus darauf angewiesen, die Funktion Punkt für Punkt auszuwerten. Die geschilderte Situation wird häufig „Black-Box-Szenario“ genannt. Ein Algorithmus, der für dieses Szenario geeignet ist, heißt „Black-Box-Algorithmus“.

Wir diskutieren zunächst, dass *alle* algorithmischen Probleme auf diese Weise formuliert werden können. Sei $\Pi \subseteq X \times Y$ ein beliebiges algorithmisches Problem. Dann können wir das Problem Π beispielsweise durch eine Menge $\{f_x \mid x \in X\}$ von zu maximierenden Funktionen $f_x \colon Y \to \mathbb{R}$ mit $f_x(y) = 1$ falls $(x, y) \in \Pi$ und $f_x(y) = 0$ falls $(x, y) \notin \Pi$ darstellen. Eine solche Modellierung stellt für Black-Box-Algorithmen eine schwierige Aufgabe dar, da die Algorithmen nach der sprichwörtlichen Nadel im Heuhaufen suchen müssen. Natürlich können algorithmische Probleme auch differenzierter modelliert werden. Es muss einzig sichergestellt sein, dass $(x, y) \in \Pi$ für alle globalen Optima $y \in Y$ von f_x gilt. In den meisten Fällen reicht es aus, pseudoboolesche Funktionen, d. h., Funktionen, die Bitfolgen einer festen Länge in die reellen Zahlen abbilden, zu betrachten. Das eingangs vorgestellte Problem, einen minimalen aufspannenden Baum in einem Graphen $G = (V, E, w)$ mit $E = \{e_1, \ldots, e_m\}$ zu finden, könnte beispielsweise durch die zu minimierende Funktion $f_G \colon \mathbb{B}^m \to \mathbb{R}$ mit $f_G(x = (x_1, \ldots, x_m)) = c(x) \cdot W + w(x)$ dargestellt werden, wobei $c(x)$ die Anzahl der Zusammenhangskomponenten von $(V, \{e_i \mid x_i = 1\})$, W die Summe der Gewichte der Kanten aus E und $w(x)$ die Summe der Gewichte der Kanten aus $\{e_i \mid x_i = 1\}$ bezeichnen (siehe [Neumann und Wegener, 2007]).

Wir haben motiviert, dass ein Algorithmus, der beliebige pseudoboolesche Funktionen $f \colon \mathbb{B}^n \to \mathbb{R}$ optimieren kann, prinzipiell auch beliebige algorithmische Probleme lösen kann. In diesem Bereich werden häufig sogenannte Suchheuristiken eingesetzt. Das sind Verfahren, die die Suchpunkte in einer bestimmten – oft von Zufallsentscheidungen beeinflussten – Reihenfolge anfragen. Die Zeit, bis ein globales Optimum gefunden wird, hängt von der eingesetzten Suchheuristik und von der zu optimierenden Funktion ab.

Die meisten gebräuchlichen Suchheuristiken treffen zufällige Entscheidungen. Viele randomisierte Suchheuristiken basieren zudem auf natürlichen Phänomenen. Das sehr bekannte Optimierungsverfahren „Simulierte Abkühlung“ (auf Englisch „Simulated Annealing“, SA) ahmt beispielsweise den langsamen Abkühlungsprozess von Metallen nach, der es den Atomen ermöglicht, stabile Kristalle zu bilden und somit einen energiearmen Zustand einzunehmen (siehe [Kirkpatrick et al., 1983]). Das genannte Verfahren stellt eine Verallgemeinerung des ebenfalls bekannten Metropolis-Algorithmus dar (siehe [Metropolis et al., 1953]). Wir werden uns in dieser Dissertation hauptsächlich auf evolutionäre Algorithmen und Ameisenalgorithmen konzentrieren. Im Allgemeinen werden zahlreiche recht unterschiedliche Verfahren als evolutionäre Algorithmen bezeichnet, denen einzig gemein ist, dass sie auf den Prinzipien der biologischen Evolution fußen. Ameisenalgorithmen versuchen, das Verhalten von Ameisenkolonien nachzuahmen, die auf erstaunliche Weise kürzeste Wege zwischen ihrem Nest und einer Futterquelle finden. Außerdem werden wir Algorithmen kennenlernen, deren Verhalten an das Immunsystem angelehnt ist, mit dessen Hilfe höhere Lebewesen Krankheiten widerstehen können.

Wenn ein Problem gegeben ist, das im Black-Box-Szenario optimiert werden muss bzw. soll, dann hat man die Wahl zwischen den im letzten Absatz angesprochenen (und vielen weiteren) Suchheuristiken. Das Verhalten der meisten Suchheuristiken kann außerdem mithilfe von Parametern konfiguriert werden. Die zu optimierenden Funktionen sind in den meisten Fällen nicht gänzlich unbekannt. In diesem Fall würde es die Auswahl eines geeigneten Verfahrens, das (hoffentlich) möglichst rasch eine gute Lösung findet, signifikant erleichtern, wenn Informationen über die Klassen von Funktionen vorhanden wären, die bestimmte Suchheuristiken besonders gut bzw. schlecht optimieren können. Außerdem kann ein fundiertes Verständnis des Verhaltens von Suchheuristiken den Entwurf von verbesserten Varianten befruchten. Die vorliegende Dissertation setzt an dieser Stelle an und liefert neue Ansätze, die die Arbeitsweisen von evolutionären Algorithmen, von Ameisenalgorithmen und von künstlichen Immunsystemen erklären, wobei die betrachteten evolutionären Algorithmen hauptsächlich auf multikriterielle Optimierungsprobleme zugeschnitten sind (siehe Kapitel 2).

Die meisten vorhandenen Studien über randomisierte Suchheuristiken basieren auf Experimenten. Wir ergänzen diese Arbeiten, indem wir Aussagen über die Effizienz von grundlegenden randomisierten Suchheuristiken auf grundlegenden Problemen *beweisen*. Indem wir randomisierte Suchheuristiken als randomisierte Algorithmen auffassen, können wir auf nützliche Methoden zurückgreifen, die das gut fundierte Teilgebiet der Informatik hervorgebracht hat, das sich der Analyse von randomisierten Algorithmen widmet. Wir werden diese Methoden hier einsetzen und ausbauen, um die Optimierungszeit der genannten randomisierten Suchheuristiken auf einzelnen Funktionen wie auch auf Funktionenklassen zu bestimmen. Der zentrale Begriff „Optimierungszeit“ bezeichnet hier die Anzahl von Funktionsauswertungen, bis zum ersten Mal ein globales Optimum angefragt wird. Diese Aufgabe ist auch für relativ einfach erscheinende Suchheuristiken anspruchsvoll, da diese im Gegensatz zu klassischen randomisierten Algorithmen nicht darauf ausgelegt sind, eine möglichst

einfache Analyse zu ermöglichen.

1.2. Überblick

Diese Dissertation umfasst im Wesentlichen drei Teile.

Der erste Teil befasst sich mit der Analyse evolutionärer Algorithmen für multikriterielle Optimierungsprobleme. Wir führen in Kapitel 2 zunächst in das Thema ein. Darin definieren wir multikriterielle Optimierungsprobleme. Anschließend erläutern wir, wie die Ausgabe eines Algorithmus bewertet werden kann. In diesem Zusammenhang kommen wir auch auf das wichtige Konzept eines Qualitätsindikators zu sprechen. Auf diesen Grundlagen aufbauend beschreiben wir, wie die Qualität von Algorithmen für multikriterielle Optimierungsprobleme bewertet werden kann. Anschließend geben wir einen kurze Einführung in das Gebiet „Evolutionäre Algorithmen“, wobei wir eine pragmatische Sicht einnehmen. Wir definieren insbesondere wichtige Sprechweisen wie auch den SEMO genannten Algorithmus, der hier häufig als Referenzalgorithmus dient.

Evolutionäre Algorithmen bestehen aus verschiedenen Modulen, die z. B. die Selektion und die Variation der Individuen steuern. Wir konzentrieren uns im ersten Teil der Dissertation auf die Selektion. In Kapitel 3 und 4 untersuchen wir die sogenannte Selektion zur Ersetzung, d. h., die Selektion, die festlegt, welche Individuen am Ende einer Iteration überleben. In Kapitel 3 vergleichen wir den Mechanismus, der für jeden nicht dominierten Zielpunkt ein Individuum speichert, mit einem Mechanismus, der den Zielraum zunächst in Boxen partitioniert und für jede nicht dominierte Box ein Individuum speichert. Den zweiten Algorithmus nennen wir DEMO. Wir konstruieren zwei Beispielfunktionen, die verdeutlichen, in welchen Situationen der eine Mechanismus dem anderen vorzuziehen ist. In Kapitel 4 betrachten wir einen evolutionären Algorithmus namens RADEMO, dessen Populationsgröße im Gegensatz zu den zuvor betrachteten Algorithmen fest ist. Die Individuen sollen mithilfe der Mechanismen, die aus den klassischen Algorithmen NSGA2 und SPEA2 bekannt sind, auf der Paretofront verteilt werden. Auch hier machen wir anhand von Beispielfunktionen deutlich, in welchen Fällen die genannten Mechanismen gut funktionieren und in welchen nicht.

In Kapitel 5 lenken wir unsere Aufmerksamkeit auf die sogenannte Selektion zur Mutation, die festlegt, welches Individuum als nächstes mutiert wird. Wir vergleichen den Algorithmus SEMO mit einem Algorithmus namens FEMO. Der Algorithmus SEMO zieht das nächste Individuum rein zufällig aus der aktuellen Population. Andererseits trifft der Algorithmus FEMO eine differenziertere Wahl, indem er speichert, wie oft die einzelnen in der Population befindlichen Individuen bereits zur Mutation ausgewählt wurden. Um eine gleichmäßige Exploration des Suchraumes zu forcieren, zieht er ein Individuum aus der aktuellen Population, dessen Zähler am kleinsten ist. Wir machen anhand von Beispielfunktionen deutlich, in welchen Situationen der Mechanismus die Optimierungszeit verkleinert und in welchen er sie vergrößert.

Im letzten Kapitel des ersten Teils – in Kapitel 6 – wenden wir uns einem der popu-

lärsten Qualitätsindikatoren zu, dem sogenannten Hypervolumen. Viele evolutionäre Algorithmen benutzen dieses Maß, um die Selektion zur Ersetzung zu realisieren. Wir untersuchen zwei Möglichkeiten, μ Punkten auf kontinuierlichen Fronten zu verteilen. Dabei vergleichen wir die Verteilung, die das Hypervolumen maximiert, mit der Verteilung, die die (multiplikative) Approximationsgüte minimiert. Diese Betrachtungen liefern neue Einsichten, wie die Approximationsgüten der beiden Verteilungen zueinander in Beziehung stehen.

Der zweite Teil der Dissertation behandelt evolutionäre Algorithmen für kombinatorische Optimierungsprobleme. In Kapitel 7 betrachten wir zunächst einen natürlichen evolutionären Algorithmus für ein multikriterielles Kürzeste-Wege-Problem. Diese Problemvariante zeichnet sich dadurch aus, dass jede Kante nicht auf ein Gewicht sondern auf $k \geq 1$ Gewichte abgebildet wird. Die Aufgabe besteht darin, eine Approximation der Paretofront zu finden. Wir beweisen, dass der evolutionäre Algorithmus diese Aufgabe erstaunlich gut meistert, wobei wir das Problem auf natürliche Weise in eine Fitnessfunktion gießen. Anschließend ergänzen wir die oberen Schranken für die Optimierungszeit durch untere Schranken.

In Kapitel 8 widmen wir uns einem Entwurfsmuster, das „Dynamische Programmierung“ genannt wird [Bellman und Dreyfus, 1962]. Diesem Ansatz folgend konnten in den letzten fast 50 Jahren zahlreiche erfolgreiche Algorithmen entworfen werden. In diesem Kapitel zeigen wir einen Zusammenhang zwischen evolutionären Algorithmen und dynamischen Programmen auf. Wir definieren einen Rahmen für dynamische Programme und einen Rahmen für evolutionäre Algorithmen. Anschließend zeigen wir, wie ein evolutionärer Algorithmus beschaffen sein muss, um ein Problem genauso erfolgreich wie ein dynamisches Programm lösen zu können. Diese neue Sichtweise plausibilisiert den Erfolg von bekannten evolutionären Algorithmen und vereinfacht in vielen Fällen die Analyse der Optimierungszeit. Auch für neue kombinatorische Probleme kann mit diesem Ansatz gezeigt werden, dass kanonische evolutionäre Algorithmen häufig sehr erfolgreich sind.

Im dritten Teil dieser Dissertation widmen wir uns anderen randomisierten Suchheuristiken. In Kapitel 9 und Kapitel 10 konzentrieren wir uns auf Ameisenalgorithmen für verschiedene Kürzeste-Wege-Probleme und in Kapitel 11 betrachten wir einen aus künstlichen Immunsystemen bekannten Alterungsoperator. Ameisenalgorithmen für die Probleme, in einem gewichteten, gerichteten Graphen kürzeste Wege von jedem Knoten zu einem festen Knoten bzw. zu jedem Knoten zu finden, untersuchen wir in Kapitel 9. Dabei setzen wir auf die Analyse eines n-ANT genannten Ameisensystems für die erste Problemvariante auf azyklischen Graphen von Attiratanasunthron und Fakcharoenphol [2008] auf. Wir betrachten Ameisenalgorithmen, die die Pheromone auf die gleiche Weise wie in dem populären Max-Min-Ameisensystem (auf Englisch „Max-Min Ant System“, MMAS) von Stützle und Hoos [2000] aktualisieren. Wir begrenzen die zugehörigen Optimierungszeiten sowohl nach oben als auch nach unten. Wir beweisen außerdem, dass das Ameisensystem für die zweite Problemvariante von einem einfachen Interaktionsmechanismus profitiert. In diesem Fall arbeitet das Ameisensystem interessanterweise schneller als alle bislang analysierten evolutionären Algorithmen mit und ohne Rekombination.

In Kapitel 10 untersuchen wir ein Ameisensystem für ein verrauschtes Kürzeste-Wege-Problem. Hier konzentrieren wir uns auf gerichtete azyklische Graphen mit positiven Gewichten. Das Rauschen auf den einzelnen Kanten e modellieren wir dabei durch Zufallsvariable $\eta(e)$ mit $\eta(e) \geq 0$. Das Ziel besteht darin, kürzeste Wege von jedem Knoten zu einem festen Knoten zu finden. Die Schwierigkeit für den Ameisenalgorithmus besteht darin, dass die Auswertung der Länge eines Pfades nicht die tatsächliche Länge sondern eine verrauschte Länge liefert. Wir beweisen eine allgemeine obere Schranke für die Optimierungszeit, die für beliebig verteiltes Rauschen gilt. Außerdem zeigen wir, dass es bei gamma-verteiltem Rauschen mit überwältigender Wahrscheinlichkeit sehr lange dauern kann, bis eine angemessene Approximation gefunden wird. Dieses Verhalten ändert sich überraschenderweise, wenn das Rauschen auf den einzelnen Kanten nicht wie zuvor unabhängig sondern korreliert ist.

Alterungsoperatoren, die unter anderem aus künstlichen Immunsystemen bekannt sind, basieren auf einem wichtigen Parameter, der sogenannten „Lebensspanne“. Sie verwalten für jedes Individuum, das sich in der aktuellen Population befindet, eine Größe, die in diesem Kontext „Alter“ genannt wird. Wenn das Alter die Lebensspanne übersteigt, dann wird das Individuum unabhängig von seiner Fitness aus der Population entfernt. In Kapitel 11 betten wir diesen aus künstlichen Immunsystemen bekannten Alterungsoperator in einen populationsbasierten evolutionären Algorithmus ein, um zu untersuchen, welche Auswirkungen die Wahl der Lebensspanne haben kann. Wir zeigen Beispielfunktionen auf, die nur dann erfolgreich optimiert werden können, wenn die Lebensspanne hinreichend klein ist, hinreichend groß ist oder sich in einem schmalen Intervall befindet. Dabei werden wir auch eine Methode kennenlernen, um Beispielfunktionen zu kombinieren, um deren Eigenschaften zu verstärken bzw. zu kombinieren.

Der vierte und letzte Teil liefert in Kapitel A eine Übersicht über die in dieser Dissertation benötigten mathematischen Grundlagen. Anschließend findet sich ein Verzeichnis, das Erklärungen zu den meisten vorkommenden mathematischen Symbolen enthält. Am Ende befinden sich das Abbildungs- und das Literaturverzeichnis.

1.3. Veröffentlichungen und Eigenanteil des Doktoranden

Die vorliegende Dissertation basiert auf den folgenden Arbeiten, die wir hier in der Reihenfolge ihrer Verwendung auflisten:

1. Christian Horoba und Frank Neumann. Benefits and drawbacks for the use of ϵ-dominance in evolutionary multi-objective optimization. In *10th Genetic and Evolutionary Computation Conference (GECCO '08)*, Seiten 641–648. ACM Press, 2008. Nominiert für einen *Best-Paper-Award*.

   Die Inhalte der folgenden, erweiterten Fassung sind teilweise in Kapitel 3 zu finden.

   Christian Horoba und Frank Neumann. *Advances in Multi-Objective Nature Inspired Computing*, Band 272 aus *Studies in Computational Intelligence*, Kapitel

Approximating Pareto-Optimal Sets Using Diversity Strategies in Evolutionary Multi-Objective Optimization, Seiten 23–44. Springer-Verlag, 2010.

2. Christian Horoba und Frank Neumann. Additive approximations of Pareto-optimal sets by evolutionary multi-objective algorithms. In *10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, Seiten 79–86. ACM Press, 2009.

   Die Inhalte der folgenden, erweiterten Fassung sind teilweise in Kapitel 4 zu finden.

   Christian Horoba und Frank Neumann. *Advances in Multi-Objective Nature Inspired Computing*, Band 272 aus *Studies in Computational Intelligence*, Kapitel Approximating Pareto-Optimal Sets Using Diversity Strategies in Evolutionary Multi-Objective Optimization, Seiten 23–44. Springer-Verlag, 2010.

3. Tobias Friedrich, Christian Horoba und Frank Neumann. Runtime analyses for using fairness in evolutionary multi-objective optimization. In *10th International Conference on Parallel Problem Solving from Nature (PPSN '08)*, Band 5199 aus *Lecture Notes in Computer Science*, Seiten 671–680. Springer-Verlag, 2008a.

   Die Inhalte der folgenden, erweiterten Fassung sind teilweise in Kapitel 5 zu finden.

   Tobias Friedrich, Christian Horoba und Frank Neumann. Showcases of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science*, 2010. Erscheint.

4. Tobias Friedrich, Christian Horoba und Frank Neumann. Multiplicative approximations and the hypervolume indicator. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 571–578. ACM Press, 2009. Ausgezeichnet mit einem *Best-Paper-Award*.

   Die Inhalte sind in Kapitel 6 zu finden.

5. Christian Horoba. Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In *10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, Seiten 113–120. ACM Press, 2009.

   Die Inhalte der folgenden, erweiterten Fassung sind in Kapitel 7 zu finden.

   Christian Horoba. Exploring the runtime of an evolutionary algorithm for the multiobjective shortest path problem. *Evolutionary Computation*, 18(3), 2010. Erscheint.

6. Benjamin Doerr, Anton Eremeev, Christian Horoba, Frank Neumann und Madeleine Theile. Evolutionary algorithms and dynamic programming. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 771–777. ACM Press, 2009.

   Die Inhalte sind in Kapitel 8 zu finden.

7. Christian Horoba und Dirk Sudholt. Running time analysis of ACO systems for shortest path problems. In *2nd International Workshop on Engineering Stochastic Local Search Algorithms (SLS '09)*, Band 5752 aus *Lecture Notes in Computer Science*, Seiten 76–91. Springer-Verlag, 2009.

   Die Inhalte sind in Kapitel 9 zu finden.

8. Christian Horoba und Dirk Sudholt. Ant colony optimization for stochastic shortest path problems. In *12th Genetic and Evolutionary Computation Conference (GECCO '10)*. ACM Press, 2010. Nominiert für einen *Best-Paper-Award*. Erscheint.

   Die Inhalte sind in Kapitel 10 zu finden.

9. Christian Horoba, Thomas Jansen und Christine Zarges. Maximal age in randomized search heuristics with aging. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 803–810. ACM Press, 2009. Nominiert für einen *Best-Paper-Award*.

   Die Inhalte sind in Kapitel 11 zu finden.

Bei Veröffentlichungen mit k Autoren beträgt der Eigenanteil des Verfassers dieser Dissertation mindestens $1/k$. Beachte, dass alle genannten Publikationen unter dem Geburtsnamen des Verfassers – Horoba – erschienen sind.

Teil I.

Evolutionäre Algorithmen für multikriterielle Optimierung

2. Grundlagen

2.1. Szenario

Unter einem multikriteriellen Optimierungsproblem verstehen wir die Aufgabe, optimale Alternativen in einer Menge von Alternativen zu finden, wobei die Qualität einer Alternative in zwei oder mehr Dimensionen gemessen wird. Die Herausforderung, mehrere Ziele zu optimieren, die potenziell unvereinbar sind, führt dazu, dass es im Allgemeinen nicht eine optimale Alternative sondern mehrere optimale Alternativen gibt, die paarweise unvergleichbar sind. Wenn es mehrere optimale Alternativen gibt, dann ist es sinnvoll, sich zunächst einen Überblick zu verschaffen. Eine Menge von Alternativen liefert die gewünschte Übersicht, wenn sie für jeden optimalen Qualitätsvektor genau eine zugehörige Alternative enthält. Anschließend kann aus dieser repräsentativen Menge eine Alternative ausgesucht werden. Viele Entscheidungsprobleme, mit denen wir im Laufe unseres Lebens konfrontiert werden, können als multikriterielle Optimierungsprobleme aufgefasst werden.

Stellen wir uns vor, wir befinden uns in Recklinghausen und möchten mit dem Auto nach Dortmund reisen. Um Dortmund zu erreichen, entscheiden wir uns zunächst für eine Route von Recklinghausen nach Dortmund. Wir sind einerseits an einer möglichst *schnellen* und andererseits an einer möglichst *kurzen* Route interessiert. Nehmen wir an, dass sieben mögliche Routen existieren. Abbildung 2.1 zeigt die relevanten Eigenschaften der Routen. Offensichtlich können wir die Routen B und D ausschließen, da es Routen gibt, die sowohl schneller als auch kürzer sind. Die Alternativen A und G stellen die schnellste bzw. kürzeste Route dar. Die verbliebenen Alternativen C, E und F sind ebenfalls interessant, da sie einen Kompromiss zwischen den extremen Routen A und G bieten. Weil die für uns wichtigen Eigenschaften – Zeit und Distanz – der optimalen Routen E und F übereinstimmen, genügt es in den meisten Fällen, im Folgenden nur eine dieser Routen zu betrachten. Wir halten fest, dass die Alternativen A, C, E, F und G optimale Routen darstellen. Weiterhin reicht es aus, wenn uns die Alternativen A, C, E und G oder die Alternativen A, C, F und G angeboten werden, da die Alternativen E und F im Wesentlichen gleich sind. Die Entscheidung für eine der optimalen Möglichkeiten hängt einzig von unseren persönlichen Präferenzen ab, die wir erst präzisieren können oder wollen, wenn uns alle optimalen Alternativen vorliegen.

Es lassen sich unzählige weitere Beispiele für multikriterielle Optimierungsprobleme finden. Betrachten wir beispielsweise das Problem, einen Verbrennungsmotor zu konstruieren, der einerseits möglichst wenig Kraftstoff verbraucht und andererseits möglichst leistungsstark ist. Oder betrachten wir das Problem, ein Programm zu schreiben, das eine Aufgabe möglichst schnell löst und dabei möglichst wenig Spei-

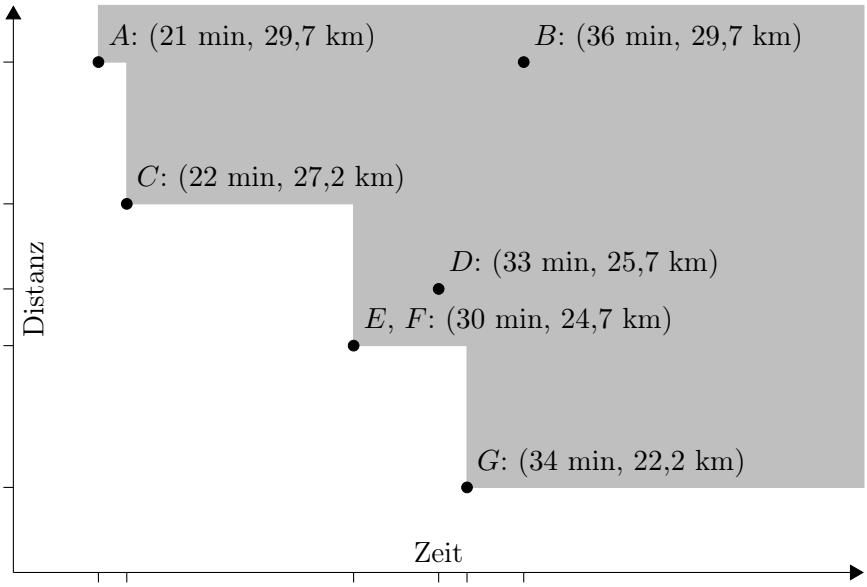


Abbildung 2.1.: Die Eigenschaften der möglichen Routen von Recklinghausen nach Dortmund.

cherplatz benötigt. Wir verzichten an dieser Stelle auf weitere Beispiele und halten fest, dass viele relevante Probleme auf natürliche Weise als multikriterielle Optimierungsprobleme aufgefasst werden können.

Wir werden nun formaler. Wir untersuchen Optimierungsprobleme, die durch einen Suchraum S und eine Zielfunktion $f = (f_1, \ldots, f_d) \colon S \to f(S) \subseteq \mathbb{R}^d$ gegeben sind. Der Suchraum umfasst die zulässigen Alternativen und die Zielfunktion misst die Qualität einer Alternative. Um die Qualität von zwei Alternativen miteinander vergleichen zu können, müssen wir eine Ordnungsrelation auf $f(S) \subseteq \mathbb{R}^d$ festlegen. Sinnvollerweise sollte es sich dabei um eine Quasiordnung handeln, d. h., eine reflexive und transitive Ordnungsrelation (siehe auch Definition A.8 und A.9). Wir nehmen im Folgenden ohne Beschränkung der Allgemeinheit an, dass alle Teilfunktionen f_i zu maximieren sind. Diese Annahme ist aufgrund von $\min_{x \in S} f_i(x) = -\max_{x \in S} -f_i(x)$ gerechtfertigt. Wenn wir die Präferenzen der Entscheidungsinstanz nicht kennen, dann können wir eine Alternative x als mindestens so gut wie eine Alternative x' ansehen genau dann, wenn x bezüglich jedes einzelnen Ziels f_i mindestens so gut wie x' ist. Es ist daher sinnvoll, festzulegen, dass ein Vektor $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ einen Vektor $x' = (x'_1, \ldots, x'_d) \in \mathbb{R}^d$ *schwach dominiert* genau dann, wenn $x_i \geq x'_i$ für alle $i \in \{1, \ldots, d\}$ gilt. In diesem Fall schreiben wir $x \succeq x'$. Die Ordnungsrelation $\succeq$ können wir wie folgt präzisieren. Dazu definieren wir weiter, dass ein Vektor x einen Vektor x' *dominiert* genau dann, wenn $x \succeq x'$ gilt und $x' \succeq x$ nicht gilt, und schreiben $x \succ x'$. Die letzte Definition können wir alternativ durch

$$x \succ x' \iff (\forall i \in \{1, \ldots, d\} \colon x_i \geq x'_i) \wedge (\exists i \in \{1, \ldots, d\} \colon x_i > x'_i)$$

ausdrücken. Genau dann, wenn weder $x \succeq x'$ noch $x' \succeq x$ gilt, dann nennen wir x und x' *unvergleichbar* und schreiben $x \parallel x'$. Außerdem nennen wir x und x' *indifferent* und notieren $x \equiv x'$ genau dann, wenn $x \succeq x'$ und $x' \succeq x$ gelten.

Wir weisen an dieser Stelle explizit darauf hin, dass multikriterielle Optimierungsprobleme in gewisser Weise unvollständig definiert sind. Die Entscheidungsinstanz legt *a priori* ausschließlich fest, dass sie an Alternativen interessiert ist, die in jeder Dimension eine möglichst hohe Qualität aufweisen. Diese unvollständige Festlegung auf ein Optimierungsziel führt im Allgemeinen dazu, dass mehrere optimale Qualitätsvektoren existieren. Die Entscheidungsinstanz artikuliert ihre genauen Präferenzen erst *a posteriori*, wenn sie alle optimalen Qualitätsvektoren kennt. Erst dann ist sie in der Lage, eine geeignete Alternative auszuwählen.

Wir können $f(S)$ und $\succeq$ als quasigeordnete Menge $(f(S), \succeq)$ auffassen. Wir interessieren uns für die maximalen Elemente $\max(f(S), \succeq) = \{x \in f(S) \mid \nexists x' \in f(S) \colon x' \succ x\}$ dieser Menge. Da wir in der vorliegenden Dissertation ausschließlich nicht leere, endliche, quasigeordnete Mengen betrachten, ist die Existenz eines maximalen Elementes gesichert. Definitionsgemäß enthält $\max(f(S), \succeq)$ alle optimalen Qualitätsvektoren, die mindestens einer zulässigen Alternative zugeordnet werden. Da wir auch an den Alternativen interessiert sind, die auf optimale Qualitätsvektoren abgebildet werden, betrachten wir zusätzlich die durch $\succeq$ und f induzierte Quasiordnung $\succeq_f$ auf S, die durch $x \succeq_f x' \iff f(x) \succeq f(x')$ definiert ist. Wir interessieren uns also ebenfalls für die maximalen Elemente $\max(S, \succeq_f)$ der quasigeordneten Menge $(S, \succeq_f)$. Um die Schreibweise zu vereinfachen, werden wir diese Ordnungsrelation im Folgenden auch mit $\succeq$ bezeichnen. Die Mengen $\max(f(S), \succeq)$ und $\max(S, \succeq)$ werden häufig nach dem italienischen Ingenieur, Ökonomen und Soziologen Vilfredo Frederico Pareto (1848–1923) *Paretofront* bzw. *Paretomenge* genannt; die Elemente der genannten Mengen nennt man häufig *Pareto-optimal*. Auch dem irischen Ökonomen Francis Ysidro Edgeworth (1845–1926) wird prägender Einfluss auf die hier behandelte Theorie zugeschrieben, wie z. B. die von der „International Society on Multiple Criteria Decision Making (MCDM)“ vergebene Auszeichnung „Edgeworth-Pareto Award“ belegt. Außerdem nennen wir den durch $n_i = \min_{x \in S} f_i(x)$ definierten Punkt $n = (n_1, \ldots, n_d) \in \mathbb{R}^d$ *Nadirpunkt* und den durch $u_i = \max_{x \in S} f_i(x)$ definierten Punkt $u = (u_1, \ldots, u_d) \in \mathbb{R}^d$ *Utopiapunkt*. Dann gilt $f(S) \subseteq [n_1, u_1] \times \cdots \times [n_d, u_d]$, wobei weder der Nadir- noch der Utopiapunkt in $f(S)$ enthalten sein müssen. Das Wort „Nadir“ hat seinen Ursprung in dem arabischen Wort für den Begriff „Gegenteil“ und bezeichnet beispielsweise in der Geometrie und in der astronomischen Navigation den Fußpunkt, der dem Zenit gegenüberliegt. Das Wort „Utopia“ geht auf ein 1516 veröffentlichtes Buch von Thomas More (1478–1535) zurück, in dem es eine imaginäre Insel bezeichnet, auf der eine „ideale“ Gesellschaft beheimatet ist; das Wort stammt aus dem Griechischen und wird heutzutage mit Vollkommenheit assoziiert.

Normalerweise werden die Paretofront und die Paretomenge gemeinsam und nicht getrennt voneinander betrachtet. Die Menge

$$\{(x, f(x)) \in S \times f(S) \mid \nexists x' \in S \colon x' \succ x\}$$

enthält alle Informationen, die die Entscheidungsinstanz benötigt, um sich für ei-

ne optimale Alternative entscheiden zu können. Wenn wir davon ausgehen, dass die Entscheidungsinstanz für jeden optimalen Qualitätsvektor nicht alle zugehörigen Alternativen sondern nur eine zugehörige Alternative benötigt, dann ist es sinnvoll, eine möglichst kleine Teilmenge der Paretomenge zu berechnen, die die Paretofront repräsentiert. Wir suchen also eine Menge $A \subseteq S$, die die Eigenschaften $f(A) = \max(f(S), \succeq)$ und $x \neq x' \implies x \parallel x'$ für alle $x, x' \in A$ besitzt.

Wenn man eine solche Menge effizient berechnen möchte, dann wird man potenziell mit zwei Schwierigkeiten konfrontiert. Zum einen kann es bereits schwierig sein, einzelne Elemente der Paretofront zu berechnen, zum anderen kann die Kardinalität der Paretofront zu groß sein, im Extremfall kann sie mit der Kardinalität des Suchraumes übereinstimmen. Einerseits können wir beispielsweise eine bikriterielle Variante des NP-schwierigen Rucksack-Problems betrachten, wobei das Gewicht einer Rucksackbepackung zu minimieren und der Nutzen einer Rucksackbepackung zu maximieren sind. Offensichtlich ist es NP-schwierig, das Element der Paretofront zu berechnen, dessen Gewicht nicht größer als eine gegebene Gewichtsschranke ist und dessen Nutzen so groß wie möglich ist. Andererseits können wir beispielsweise das durch $f(x) := (\mathrm{BinVal}(x), -\mathrm{BinVal}(x))$ definierte bikriterielle Optimierungsproblem $f\colon \mathbb{B}^n \to \mathbb{R}^2$ betrachten, wobei $\mathrm{BinVal}(x = (x_1, \ldots, x_n)) = \sum_{i=1}^{n} x_i \cdot 2^{n-i}$ ist. Offensichtlich umfassen sowohl die Paretofront als auch die Paretomenge jeweils 2^n Elemente. Die Kardinalität der Paretofront wächst also exponentiell mit der Dimension n des Suchraumes $\mathbb{B}^n$. In beiden Fällen besteht ein Ausweg darin, sich mit einer Approximation der Paretofront zu begnügen. Wir werden in Abschnitt 2.2 geeignete Approximationsbegriffe kennenlernen.

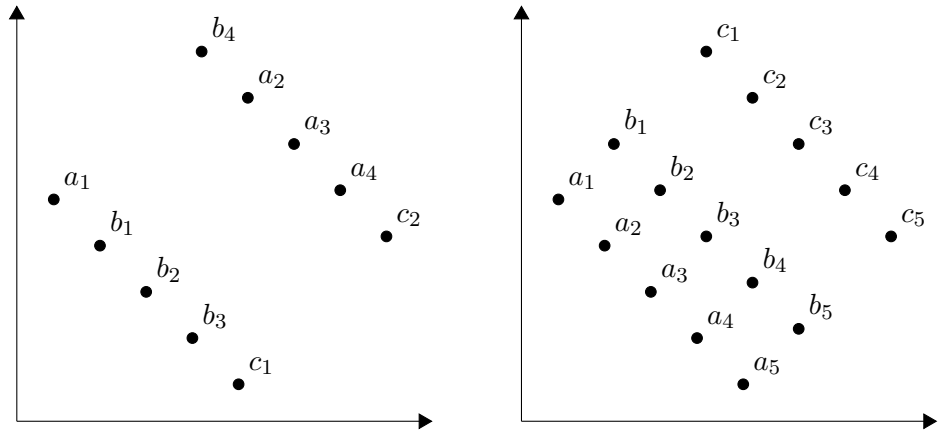
2.2. Bewertung von Alternativenmengen

Ein Algorithmus für ein multikriterielles Optimierungsproblem gibt letztlich eine Teilmenge des Suchraumes aus. Diese Alternativenmenge wird der Entscheidungsinstanz präsentiert, die daraus eine Alternative auswählt. Im Idealfall liefert der Algorithmus eine Menge, die die gesamte Paretofront repräsentiert. Die meisten Algorithmen liefern jedoch Alternativenmengen, die nicht die gesamte Paretofront umfassen. Um z. B. die Leistungsfähigkeit von Algorithmen miteinander vergleichen zu können, müssen wir festlegen, wann wir eine Alternativenmenge einer anderen Alternativenmenge vorziehen.

Das natürliche Vorgehen besteht darin, die in Abschnitt 2.1 definierte Ordnungsrelation $\succeq$ auf Mengen von Alternativen zu verallgemeinern. Wir definieren die Ordnungsrelation

$$A \succeq B \iff \forall b \in B\colon \exists a \in A\colon a \succeq b \tag{2.1}$$

auf $\mathrm{Pot}(f(S))$ bzw. $\mathrm{Pot}(S)$ und sprechen in diesem Fall davon, dass die Alternativenmenge A die Alternativenmenge B schwach dominiert. Wir definieren außerdem, dass eine Alternativenmenge A eine Alternativenmenge B dominiert genau dann, wenn $A \succeq B$ gilt und $B \succeq A$ nicht gilt, und schreiben $A \succ B$. Wenn wir Formel 2.1



(a) Beispiel 1: $B \parallel A$, $C \parallel A$ und $C \parallel B$.

(b) Beispiel 2: $B \succ A$ und $C \succ A$.

Abbildung 2.2.: Beispiele für Alternativenmengen $A = \{a_i\}$, $B = \{b_i\}$ und $C = \{c_i\}$ für ein bikriterielles Optimierungsproblem.

einsetzen, dann erhalten wir die Beziehung

$$A \succ B \iff (\forall b \in B\colon \exists a \in A\colon a \succeq b) \wedge (\exists a \in A\colon \nexists b \in B\colon b \succeq a).$$

Wenn sowohl die Elemente in $A \subseteq \mathrm{Pot}(f(S))$ als auch die Elemente in $B \subseteq \mathrm{Pot}(f(S))$ paarweise unvergleichbar sind, dann gilt der einfachere Zusammenhang

$$A \succ B \iff (A \succeq B) \wedge (A \neq B).$$

Auf die gleiche Weise wie in Abschnitt 2.1 können wir auch für Alternativenmengen die Begriffe „unvergleichbar“ und „indifferent“ definieren.

Die auf $\succeq$ basierenden Vergleichsmethoden $\succeq$, $\succ$, $\parallel$ und $\equiv$ sind so allgemein wie möglich, da sie nicht auf die Präferenzen der Entscheidungsinstanz zugeschnitten sind. Diese allgemeine Sichtweise führt zu zwei unausweichlichen Konsequenzen. Zum einen sind die meisten Alternativenmengen unvergleichbar (siehe Abbildung 2.2(a)). Wenn wir die genannten Relationen heranziehen, um die Ergebnisse von zwei Algorithmen miteinander zu vergleichen, dann können wir in den meisten Fällen lediglich schließen, dass die Ergebnisse unvergleichbar sind, und demzufolge beide Algorithmen als gleichwertig einstufen. Zum anderen ermöglichen die genannten Relationen keine quantitativen Aussagen. Wenn $A \succ B$ gilt, dann wissen wir lediglich, dass die Alternativenmenge A besser als die Alternativenmenge B ist (siehe Abbildung 2.2(b)). Es ist für den systematischen Vergleich von Algorithmen hilfreich, wenn die Qualitätsunterschiede durch numerische Größen quantifiziert werden können.

Eine weitverbreitete Möglichkeit, die im letzten Absatz beschriebenen Schwierigkeiten zu umgehen, besteht darin, die Präferenzen der Entscheidungsinstanz durch zusätzliche Annahmen zu präzisieren. Die Definition von sogenannten *Qualitätsindikatoren* $I\colon \mathrm{Pot}(f(S)) \to \mathbb{R}$, die angeben, wie zufrieden die Entscheidungsinstanz mit

einer Alternativenmenge ist, stellt einen solchen Versuch dar. Es gibt sowohl Qualitätsindikatoren, die zu maximieren sind, als auch solche, die zu minimieren sind. Qualitätsindikatoren können beispielsweise benutzt werden, um die Ergebnisse von mehreren Algorithmen für ein Problem unter verschiedenen Gesichtspunkten miteinander zu vergleichen. Solche Untersuchungen können helfen, geeignete Algorithmen für gegebene Probleme auszuwählen. Außerdem können Qualitätsindikatoren in moderne Suchheuristiken eingebettet werden, um die persönlichen Präferenzen bereits in die Suche einfließen zu lassen.

Wir weisen an dieser Stelle darauf hin, dass Qualitätsindikatoren das ursprüngliche multikriterielle Optimierungsproblem vereinfachen. Durch die Abbildung von Alternativenmengen auf reelle Zahlen gehen unweigerlich Informationen verloren. Wir wissen beispielsweise, dass es keinen Qualitätsindikator I gibt, der die wünschenswerte Beziehung $I(f(A)) > I(f(B)) \iff A \succ B$ bzw. $I(f(A)) < I(f(B)) \iff A \succ B$ für alle Zielfunktionen $f\colon S \to \mathbb{R}^d$ und für alle Alternativenmengen $A, B \subseteq \mathrm{Pot}(S)$ garantiert [Zitzler et al., 2003].

Wir können einen Qualitätsindikator I verwenden, um durch $A \succeq_I B \iff I(f(A)) \geq I(f(B))$ bzw. $A \succeq_I B \iff I(f(A)) \leq I(f(B))$ eine totale Quasiordnung $\succeq_I$ auf S festzulegen. Außerdem definieren wir $A \succ_I B \iff I(f(A)) > I(f(B))$ bzw. $A \succ_I B \iff I(f(A)) < I(f(B))$. Ein Qualitätsindikator ermöglicht, zwei beliebige Alternativenmengen miteinander zu vergleichen und quantitative Aussagen wie beispielsweise „A ist doppelt so gut wie B“ zu treffen. Um einen sinnvollen Qualitätsindikator I zu definieren, müssen wir darauf achten, dass die induzierte Quasiordnung $\succeq_I$ die natürliche Quasiordnung $\succeq$ erweitert und nicht zu Widersprüchen führt. Wir nennen einen Qualitätsindikator I *monoton* genau dann, wenn $A \succeq B \implies A \succeq_I B$ für alle $A, B \subseteq S$ gilt. Betrachte einen nicht monotonen Qualitätsindikator I. Definitionsgemäß gibt es dann Alternativenmengen $A, B \subseteq S$ mit $A \succeq B$ und $B \succ_I A$. Der Hinweis des Qualitätsindikators, dass Alternativenmenge B besser als Alternativenmenge A ist, führt in diesem Fall offensichtlich zu einem Widerspruch. Es ist daher sinnvoll, sich auf monotone Qualitätsindikatoren zu beschränken.

Natürlich können sich Qualitätsindikatoren durch weitere wünschenswerte Eigenschaften auszeichnen. Wir nennen einen Qualitätsindikator I *streng monoton* genau dann, wenn zusätzlich $A \succ B \implies A \succ_I B$ für alle $A, B \subseteq S$ gilt. Wenn ein Qualitätsindikator I streng monoton ist, dann ist sichergestellt, dass eine Alternativenmenge, die auf einen optimalen Wert abgebildet wird, eine Teilmenge enthält, die die Paretofront repräsentiert.

Es ist außerdem wünschenswert, dass ein Qualitätsindikator nicht von der Werteskala der Zielfunktion beeinflusst wird. Wir können die Werteskala transformieren, indem wir die Zielfunktion f und eine Funktion $s = (s_1, \ldots, s_d)\colon \mathbb{R}^d \to \mathbb{R}^d$, die aus streng monotonen Funktionen $s_i\colon \mathbb{R} \to \mathbb{R}$ besteht, komponieren und fortan die skalierte Zielfunktion $s \circ f$ verwenden. Die skalierte Zielfunktion $s_i(f_i(x)) = (f_i(x) - n_i)/(u_i - n_i)$ wird beispielsweise gerne benutzt, um $f(S) \subseteq [0, 1]^d$ zu garantieren, wobei n und u den Nadir- bzw. Utopiapunkt bezeichnen. Wir nennen einen Qualitätsindikator I *skalierungsinvariant* genau dann, wenn $I(s \circ f(A)) = I(f(A))$

für alle $A \subseteq S$ und für alle genannten Funktionen s gilt.

Weiterhin ist es häufig entscheidend, dass die Berechnung eines Qualitätsindikators die Ressourcen Zeit und Speicher nicht übermäßig beansprucht. Dies gilt besonders, wenn der Qualitätsindikator innerhalb einer Suchheuristik verwendet werden soll. Eine schnelle Auswertung ist jedoch auch dann hilfreich, wenn die Ergebnisse verschiedener Suchheuristiken einem umfassenden Vergleich unterzogen werden sollen. Des Weiteren ist es in diesem Kontext von großer Wichtigkeit, ob die effiziente Auswertung eines Qualitätsindikators genaue Informationen über das zu optimierende Problem, beispielsweise die Kenntnis der Paretofront, voraussetzt oder nicht. Wenn wir einen Qualitätsindikator verwenden, um die Leistungsfähigkeit von Algorithmen auf Beispielproblemen zu bewerten, dann ist die Paretofront in der Regel bekannt. Wenn wir den Qualitätsindikator allerdings innerhalb einer Suchheuristik verwenden möchten, dann kommen natürlich nur Qualitätsindikatoren in Frage, die auch ohne die Kenntnis der Paretofront effizient ausgewertet werden können.

Da ein Qualitätsindikator die Präferenzen der Entscheidungsinstanz widerspiegelt, existieren sehr viele verschiedene Qualitätsindikatoren. Wir stellen eine Auswahl von Qualitätsindikatoren vor, die in unserem Kontext von Interesse sind.

Der zu maximierende Qualitätsindikator $\mathrm{I}_{\mathrm{fro}} \colon \mathrm{Pot}(S) \to \mathbb{R}$ misst den von A repräsentierten Anteil der Paretofront [Ulungu et al., 1999]. Er ist durch

$$\mathrm{I}_{\mathrm{fro}}(A) := \frac{|\{x \in \max(f(S), \succeq) \mid \exists a \in A \colon f(a) \succeq x\}|}{|\max(f(S), \succeq)|}$$

definiert. Der Qualitätsindikator ist monoton, nicht streng monoton und skalierungsinvariant. Er kann in der Zeit $\mathcal{O}(|\max(f(S), \succeq)| \cdot |A| \cdot d)$ berechnet werden, wenn die Paretofront bekannt ist. In diesem Fall genügt es, $f(a) \succeq x$ für alle $x \in \max(f(S), \succeq)$ und für alle $a \in A$ zu prüfen. Da in der Zeit $\mathcal{O}(d)$ ermittelt werden kann, ob $f(a) \succeq x$ gilt, ergibt sich die genannte Gesamtlaufzeit.

Die beiden zu minimierenden Qualitätsindikatoren $\mathrm{I}_{\mathrm{mul}} \colon \mathrm{Pot}(S) \to \mathbb{R}$ wie auch $\mathrm{I}_{\mathrm{add}} \colon \mathrm{Pot}(S) \to \mathbb{R}$ messen die multiplikative bzw. die additive Approximationsgüte von A [Laumanns et al., 2002a]. Der Qualitätsindikator $\mathrm{I}_{\mathrm{mul}}$ ist durch

$$\mathrm{I}_{\mathrm{mul}}(A) := \inf\{\epsilon \in \mathbb{R} \mid \forall x \in \max(f(S), \succeq) \colon \exists a \in A \colon f(a) \succeq_{\epsilon,\cdot} x\}$$

und

$$x \succeq_{\epsilon,\cdot} x' \iff \forall i \in \{1, \ldots, d\} \colon (1 + \epsilon) \cdot x_i \geq x'_i$$

definiert. Es ist zu beachten, dass der Qualitätsindikator $\mathrm{I}_{\mathrm{mul}}$ implizit auf der Annahme $f(S) \subseteq (\mathbb{R}^+)^d$ basiert. Der Qualitätsindikator $\mathrm{I}_{\mathrm{add}}$ ist durch

$$\mathrm{I}_{\mathrm{add}}(A) := \inf\{\epsilon \in \mathbb{R} \mid \forall x \in \max(f(S), \succeq) \colon \exists a \in A \colon f(a) \succeq_{\epsilon,+} x\}$$

und

$$x \succeq_{\epsilon,+} x' \iff \forall i \in \{1, \ldots, d\} \colon x_i + \epsilon \geq x'_i$$

definiert. Beide Qualitätsindikatoren sind monoton und nicht streng monoton. Der Qualitätsindikator $\mathrm{I}_{\mathrm{mul}}$ ist skalierungsinvariant und $\mathrm{I}_{\mathrm{add}}$ ist nicht skalierungsinvariant. Sie können in der Zeit $\mathcal{O}(|\max(f(S), \succeq)| \cdot |A| \cdot d)$ berechnet werden, wenn die

Qualitätsindikator	monoton	streng monoton	skalierungsinvariant
I_{fro}	ja	nein	ja
I_{mul}	ja	nein	ja
I_{add}	ja	nein	nein
I_{hyp}	ja	ja	nein

Tabelle 2.1.: Zusammenfassung der Eigenschaften der Qualitätsindikatoren I_{fro}, I_{mul}, I_{add} und I_{hyp}.

Paretofront bekannt ist. Um diese Laufzeit zu erzielen, wird $\inf\{\epsilon \in \mathbb{R} \mid f(a) \succeq_{\epsilon,.} x\}$ bzw. $\inf\{\epsilon \in \mathbb{R} \mid f(a) \succeq_{\epsilon,+} x\}$ in der Zeit $\mathcal{O}(d)$ für alle $x \in \max(f(S), \succeq)$ und für alle $a \in A$ bestimmt. Die Qualitätsindikatoren I_{mul} und I_{add} basieren wesentlich auf der Arbeit [Papadimitriou und Yannakakis, 2000].

Der zu maximierende Qualitätsindikator $I_{hyp}\colon \mathrm{Pot}(S) \to \mathbb{R}$ misst das Volumen des von $f(A)$ dominierten Anteils des Zielraumes [Zitzler und Thiele, 1998a, 1999]. Er ist durch

$$I_{hyp}(A) := \mathrm{Vol}\left(\bigcup_{(a_1,\ldots,a_d)\in f(A)} [r_1, a_1] \times \cdots \times [r_d, a_d]\right)$$

definiert, wobei Vol das Lebesguemaß und $r = (r_1, \ldots, r_d) \in \mathbb{R}^d$ einen sogenannten Referenzpunkt bezeichnen. Der Qualitätsindikator ist streng monoton, wenn $n_i > r_i$ für alle $i \in \{1, \ldots, d\}$ gilt, wobei $n = (n_1, \ldots, n_d)$ den Nadirpunkt von f bezeichnet. Momentan ist I_{hyp} der einzige bekannte Qualitätsindikator, der streng monoton ist [Zitzler et al., 2007]. Leider ist der Qualitätsindikator nicht skalierungsinvariant und die Zeit, die für seine Berechnung erforderlich ist, wächst exponentiell mit der Anzahl der Ziele. Die Komplexität des Problems wurde systematisch in [Bringmann und Friedrich, 2008] und [Bringmann und Friedrich, 2009] untersucht. Der erste Algorithmus für die Berechnung des Hypervolumens wurde unabhängig von Zitzler [2001] und Knowles [2002] vorgeschlagen; dieser Algorithmus wird kurz HSO (Hypervolume by Slicing Objectives) genannt. Der derzeit schnellste Algorithmus von Beume und Rudolph [2006] berechnet das Hypervolumen in der Laufzeit $\mathcal{O}(n^{d/2} + n \cdot \log n)$, wobei $n = |A|$ ist. Dieser Algorithmus nutzt aus, dass die Berechnung des Hypervolumens einen Spezialfall des prominenten Maßproblems von Klee darstellt, das die Suche nach dem Volumen der Vereinigung von n d-dimensionalen Rechtecken bezeichnet. Das letztgenannte Problem kann mithilfe des Algorithmus von Overmars und Yap [1991] in der Laufzeit $\mathcal{O}(n^{d/2} \cdot \log n)$ gelöst werden, falls $d \geq 3$ gilt. Die Definition des Hypervolumens kann noch etwas allgemeiner gefasst werden. Die naheliegende Verallgemeinerung

$$I(A) = \int_{(r_1,\ldots,r_d)}^{(\infty,\ldots,\infty)} w(z) \cdot \alpha_A(z)\,\mathrm{d}z$$

mit $\alpha_A\colon \mathbb{R}^d \to \mathbb{B}$ und $w\colon \mathbb{R}^d \to \mathbb{R}^+$ wird gewichtetes Hypervolumen genannt (siehe [Zitzler et al., 2007, Auger et al., 2009b,c]). Dabei bezeichnen α_A die durch $\alpha_A(z) =$

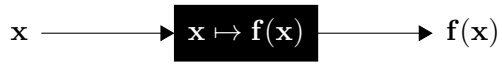


Abbildung 2.3.: Darstellung einer Zielfunktion f als Black-Box.

$1 \iff f(A) \succeq \{z\}$ definierte Indikatorfunktion und w eine beliebige Gewichtsfunktion. Das gewichtete Hypervolumen weist ähnliche Eigenschaften wie das Hypervolumen auf. Der entscheidende Vorteil besteht darin, dass zusätzlich Präferenzen der Entscheidungsinstanz mithilfe der Gewichtsfunktion in den Qualitätsindikator eingebettet werden können.

Tabelle 2.1 gibt einen Überblick über die Eigenschaften der genannten Qualitätsindikatoren. Die Charakteristika weiterer Qualitätsindikatoren werden beispielsweise in [Zitzler et al., 2003] beschrieben.

2.3. Bewertung von Algorithmen

Nachdem wir in Abschnitt 2.1 multikriterielle Optimierungsprobleme eingeführt haben und in Abschnitt 2.2 darauf eingegangen sind, wie die Qualität von Alternativenmengen bewertet werden kann, werden wir nun besprechen, wie die Qualität von Algorithmen beurteilt werden kann. Wir gehen in dieser Dissertation davon aus, dass ein Algorithmus die zu optimierende Zielfunktion a priori nicht kennt. Die einzige Möglichkeit, etwas über die Zielfunktion und damit die Paretofront zu erfahren, besteht darin, die Zielfunktion für einzelne Suchpunkte auszuwerten. Da die Abbildungsvorschrift der Zielfunktion verborgen ist, können wir die Zielfunktion anschaulich durch einen schwarzen Kasten repräsentieren, der in diesem Kontext auch Black-Box genannt wird (siehe Abbildung 2.3). Diese Sicht ist beispielsweise dann angebracht, wenn die Qualität eines Suchpunktes durch die Durchführung eines Experimentes oder die Berechnung eines Simulationsmodelles ermittelt werden muss. Heutzutage werden randomisierte Suchheuristiken jedoch auch dann eingesetzt, wenn Informationen über die zu optimierende Zielfunktion vorliegen. Häufig mangelt es an der Zeit oder an der Expertise, um das vorhandene Problemwissen auszunutzen und einen maßgeschneiderten Algorithmus zu entwerfen. In diesen Fällen wird häufig zu randomisierten Suchheuristiken gegriffen, mit deren Hilfe prinzipiell beliebige Probleme gelöst werden können. Da diese Heuristiken das vorhandene Problemwissen nicht ausnutzen und das Problem einzig durch das punktweise Auswerten der Zielfunktion „kennenlernen“, ist auch hier das sogenannte Black-Box-Szenario angebracht [Droste et al., 2006].

Wir betrachten nun die Optimierung einer Zielfunktion $f\colon S \to \mathbb{R}^d$. In diesem allgemeinen Szenario können wir die Sicht auf einen Algorithmus vereinfachen, indem wir nur die Reihenfolge betrachten, in der der Algorithmus die Qualität von Suchpunkten x_i anfragt. Im Allgemeinen kann die i-te Anfrage x_i eines solchen Algorithmus von den bisherigen Anfragen $x_1, \ldots, x_{i-1}$, den zugehörigen Antworten $f(x_1), \ldots, f(x_{i-1})$ und natürlich Zufallsentscheidungen abhängen.

Um die Optimierungszeit eines Algorithmus zu definieren, müssen wir uns zunächst auf ein Optimierungsziel einigen. Wir können ein Optimierungsziel mithilfe einer Funktion $z\colon \mathrm{Pot}(S) \to \mathbb{B}$ formalisieren. Der Algorithmus hat das Optimierungsziel nach der i-ten Anfrage eines Suchpunktes genau dann erreicht, wenn $z(\{x_1, \ldots, x_i\}) = 1$ gilt. Es ist sinnvoll, nur Funktionen z zuzulassen, für die

$$\forall A, B \subseteq S\colon A \subseteq B \implies z(A) \leq z(B)$$

gilt, da die Kenntnis der Qualitäten weiterer Suchpunkte ein erreichtes Optimierungsziel nicht zunichtemachen kann. Interessante Optimierungsziele können beispielsweise mithilfe der in Abschnitt 2.2 eingeführten Qualitätsindikatoren definiert werden. Wir können z. B. festlegen, dass das Optimierungsziel darin besteht, alle Pareto-optimalen Suchpunkte anzufragen. Dieses Optimierungsziel wird durch

$$z(A) = \lfloor \mathrm{I}_{\mathrm{fro}}(A) \rfloor$$

beschrieben. Wir wissen, dass das Erreichen dieses Ziels im Allgemeinen aus Effizienzgründen illusorisch ist. Ein bescheideneres Optimierungsziel könnte darin bestehen, die Paretofront zu approximieren. Wenn wir die multiplikative Approximationsgüte $1 + \epsilon$ anstreben, dann wird ein solches Ziel durch

$$z(A) = \begin{cases} 1 & \text{falls } \mathrm{I}_{\mathrm{mul}}(A) \leq 1 + \epsilon \\ 0 & \text{falls } \mathrm{I}_{\mathrm{mul}}(A) > 1 + \epsilon \end{cases}$$

formalisiert.

Wir können nun den für diese Dissertation zentralen Begriff „Optimierungszeit“ definieren. Betrachte einen Algorithmus A, eine Zielfunktion f und ein Optimierungsziel z. Dann definieren wir die *Optimierungszeit* $T(A, f, z)$ von A auf f bezüglich z durch

$$T(A, f, z) := \min\{i \in \mathbb{N} \mid z(\{X_1, \ldots, X_i\}) = 1\},$$

wobei $X_i \in S$ den Suchpunkt bezeichnet, der von A im i-ten Schritt angefragt wird. Die Optimierungszeit kann auf die naheliegende Weise auf Mengen von Zielfunktionen verallgemeinert werden. Betrachte eine Menge $F = \{f\colon S \to \mathbb{R}^d\}$. Wir definieren die Optimierungszeit $T(A, F, z)$ von A auf F bezüglich z durch ihre Verteilungsfunktion

$$\mathrm{W}(T(A, F, z) \leq x) := \sup\{\mathrm{W}(T(A, f, z) \leq x) \mid f \in F\},$$

wobei $x \in \mathbb{R}$ ist. Dann gilt $\mathrm{E}(T(A, F, z)) = \sup\{\mathrm{E}(T(A, f, z)) \mid f \in F\}$ für den Erwartungswert. Die Definition von $T(A, F, z)$ ist durch die folgende pessimistische Annahme motiviert: Wenn der Algorithmus A nach x Schritten terminiert, dann gibt $\mathrm{W}(T(A, F, z) \leq x)$ die Wahrscheinlichkeit, das Optimierungsziel z zu erreichen, gerade für die Zielfunktion $f \in F$ an, für die diese am kleinsten ist.

Wir werden in der vorliegenden Dissertation die Optimierungszeit von randomisierten Suchheuristiken sowohl auf einzelnen Zielfunktionen (siehe Kapitel 3, 4, 5 und 11) als auch auf Mengen von Zielfunktionen (siehe Kapitel 7, 8, 9 und 10) analysieren.

Algorithmus 1 Evolutionärer Algorithmus

Eingabe: Fitnessfunktion $f\colon S \to \mathbb{R}^d$ mit endlichem Suchraum S
Ausgabe: Suchpunkte $P = (p_i)$ mit $p_i \in S$

1: $t \leftarrow 0$
2: wähle $P^{(t)} = (p_i^{(t)})$ mit $p_i^{(t)} \in S$ ▷ Initialisierung
3: **repeat**
4: $t \leftarrow t + 1$
5: wähle (x_i) mit $x_i \in \{p_i^{(t-1)}\}$ ▷ Selektion zur Rekombination
6: erzeuge (x_i') mit $x_i' \in S$ aus (x_i) durch Rekombination ▷ Rekombination
7: erzeuge (x_i'') mit $x_i'' \in S$ aus (x_i') durch Mutation ▷ Mutation
8: wähle $P^{(t)} = (p_i^{(t)})$ mit $p_i^{(t)} \in \{p_i^{(t-1)}\} \cup \{x_i''\}$ ▷ Selektion zur Ersetzung
9: **until** Abbruchkriterium erfüllt
10: **return** $P^{(t)}$

Da wir es mit Suchheuristiken zu tun haben, die zufällige Entscheidungen treffen, ist die Optimierungszeit eine Zufallsvariable. Wir werden daher den Erwartungswert $\mathrm{E}(T)$ der Optimierungszeit T oder die Wahrscheinlichkeit $\mathrm{W}(T \leq t)$, dass das Optimierungsziel nach höchstens t Schritten erreicht wird, betrachten. Wir bezeichnen eine Heuristik als *effizient*, wenn die erwartete Optimierungszeit polynomiell groß ist. Anderenfalls nennen wir eine Heuristik *ineffizient*. Die Situation ist besonders aussichtslos, wenn die Wahrscheinlichkeit exponentiell klein ist, das Optimierungsziel in polynomiell vielen Schritten zu finden. In diesen Fällen ist sogar der Versuch, das Optimierungsziel durch parallele oder sequenzielle Läufe effizient zu erreichen, zum Scheitern verurteilt.

2.4. Evolutionäre Algorithmen

Wir werden uns in dieser Dissertation vornehmlich mit einer Art von randomisierten Suchheuristiken befassen, die *evolutionäre Algorithmen* genannt werden. Es gibt mittlerweile zahlreiche Bücher, die sich diesem Thema widmen (z. B. [Bäck et al., 1997]). Diese Algorithmen basieren auf den Prinzipien der natürlichen Evolution (Rekombination, Mutation, Selektion). Wir werden zunächst den schematischen Aufbau eines solchen Algorithmus beschreiben. Der Algorithmus beginnt die Suche mit einer Multimenge von Suchpunkten. Die Multimenge und die Elemente der Multimenge werden in diesem Kontext *Population* bzw. *Individuen* genannt. Anschließend wird die sogenannte *Fitness* der einzelnen Individuen bestimmt. Die Fitness eines Individuums x entspricht dem Zielfunktionswert $f(x)$, wobei die Zielfunktion hier *Fitnessfunktion* genannt wird. Anschließend wiederholt der Algorithmus die folgenden Schritte bis ein *Abbruchkriterium* erfüllt ist. Die einzelnen Iterationen werden hier auch *Generationen* genannt. Zunächst werden Individuen aus der aktuellen Population zur Rekombination ausgewählt (Selektion zur Rekombination). Die Auswahl dieser Individuen erfolgt häufig rein zufällig; eine relativ hohe Fitness eines Indivi-

duums kann jedoch auch die Wahrscheinlichkeit erhöhen, dieses auszuwählen. Dann werden die Bestandteile der gewählten Individuen mithilfe eines sogenannten Rekombinationsoperators neu zusammengefügt (Rekombination). Ein Rekombinationsoperator erzeugt aus zwei (oder mehr) Individuen ein neues Individuum. Ein hilfreicher Rekombinationsoperator zeichnet sich dadurch aus, dass er die guten Eigenschaften verschiedener Individuen auf geeignete Weise kombiniert (siehe z. B. [Radcliffe, 1991, Bäck et al., 1997]). Anschließend werden die so entstandenen Individuen zufälligen Veränderungen unterworfen (Mutation). Der Mutationsoperator sollte so beschaffen sein, dass er das zu mutierende Individuum mit hoher Wahrscheinlichkeit nur leicht modifiziert. Dieser Schritt erhöht die Diversität in der Population. Dann werden die neuen Individuen der aktuellen Population hinzugefügt und ihre Fitness bestimmt. Anschließend wird entschieden, welche Individuen aus der aktuellen Population in die nachfolgende Population übernommen werden (Selektion zur Ersetzung). In diesem Schritt spielt die Fitness der Individuen gewöhnlich eine entscheidende Rolle. Algorithmus 1 fasst unsere Beschreibung eines recht allgemeinen evolutionären Algorithmus zusammen.

Wir werden im Folgenden einfache Varianten von evolutionären Algorithmen betrachten. Die Algorithmen, die wir analysieren werden, erzeugen beispielsweise in jeder Generation genau ein neues Individuum. Außerdem werden wir uns auf Algorithmen beschränken, die neue Individuen einzig mithilfe eines Mutationsoperators erschaffen. Wenngleich Rekombinationsoperatoren in vielen evolutionären Algorithmen zu finden sind, wollen wir uns hier hauptsächlich auf den Einfluss unterschiedlicher Selektionsverfahren konzentrieren. Wir erwähnen an dieser Stelle, dass die Frage, ob der Rekombinations- oder der Mutationsoperator wichtiger ist, seit der Erfindung von evolutionären Algorithmen kontrovers diskutiert wird. Mittlerweile konnte bewiesen werden, dass der Einsatz eines Rekombinationsoperators essenziell ist, um bestimmte Beispielfunktionen effizient zu optimieren [Jansen und Wegener, 2002, 2005]. Dass ein Rekombinationsoperator die Optimierungszeit für ein kombinatorisches Optimierungsproblem leicht verbessern kann, wurde von Doerr et al. [2008a] gezeigt.

Wir werden nun einen einfachen evolutionären Algorithmus für multikriterielle Optimierungsprobleme beschreiben. Wir verweisen an dieser Stelle auf die Bücher [Deb, 2001] und [Coello Coello et al., 2007], die sich einzig dem Thema widmen, wie multikriterielle Probleme mithilfe von evolutionären Algorithmen gelöst werden können. Der Algorithmus wird in [Laumanns et al., 2002b] eingeführt und dort „Simple Evolutionary Multiobjective Optimizer (SEMO)“ genannt. Er kann auch als Instanz des Basisalgorithmus PR aus [Rudolph und Agapie, 2000] aufgefasst werden. Der Algorithmus arbeitet wie folgt. Er startet mit einer Population, die genau ein Individuum enthält, das rein zufällig aus dem Suchraum gezogen wird. In den einzelnen Iterationen wird zunächst ein Individuum rein zufällig aus der aktuellen Population ausgewählt. Anschließend wird dieses Individuum kopiert und das Duplikat einer Mutation unterzogen. Wenn das neue Individuum von einem Individuum aus der aktuellen Population dominiert wird, dann wird es verworfen. Anderenfalls werden zunächst alle Individuen aus der aktuellen Population entfernt, die von dem neuen

Algorithmus 2 Simple Evolutionary Multiobjective Optimizer (SEMO)

Eingabe: Fitnessfunktion $f\colon S \to \mathbb{R}^d$ mit endlichem Suchraum S
Ausgabe: Suchpunkte $P \subseteq S$

```
1: t ← 0
2: wähle x_t ∈ S rein zufällig
3: P_t ← {x_t}
4: repeat
5:     t ← t + 1
6:     wähle x ∈ P_{t−1} rein zufällig
7:     erzeuge x_t aus x durch Mutation
8:     if ∄x ∈ P_{t−1}: f(x) ≻ f(x_t) then
9:         P_t ← (P_{t−1} \ {x ∈ P_{t−1} | f(x_t) ⪰ f(x)}) ∪ {x_t}
10:    else
11:        P_t ← P_{t−1}
12:    end if
13: until Abbruchkriterium erfüllt
14: return P_t
```

Individuum schwach dominiert werden, und anschließend das neue Individuum der resultierenden Population hinzugefügt. Auf diese Weise wird sichergestellt, dass nach der t-ten Generation die Beziehung $f(P_t) \succeq f(\{x_0, \ldots, x_t\})$ gilt. Außerdem sind alle Individuen in P_t unvergleichbar. Algorithmus 2 fasst die Beschreibung zusammen.

Um mithilfe des Algorithmus SEMO eine konkrete Funktion $f\colon S \to \mathbb{R}^d$ optimieren zu können, müssen wir natürlich die Funktionsweise des Mutationsoperators konkretisieren. Wir werden zunächst Fitnessfunktionen untersuchen, deren Suchraum $\mathbb{B}^n$ entspricht. Daher stellen wir nun zwei Mutationsoperatoren vor, die für diesen Suchraum geeignet sind. Betrachte ein Individuum $x = (x_1, \ldots, x_n) \in \mathbb{B}^n$. Der erste Operator wird *k-Bit-Mutation* genannt. Der Mutationsoperator erzeugt das mutierte Individuum $x' = (x'_1, \ldots, x'_n)$ mit $x'_i = 1 - x_i$ falls $i \in M$ und $x'_i = x_i$ falls $i \notin M$, wobei $M \subseteq \{1, \ldots, n\}$, $|M| = k$, eine rein zufällig gewählte Menge ist. Der zweite Operator wird *Standard-Bit-Mutation mit Mutationswahrscheinlichkeit* k/n genannt. Der Mutationsoperator erzeugt das mutierte Individuum $x' = (x'_1, \ldots, x'_n)$ mit $x'_i = 1 - x_i$ falls $X_i = 1$ und $x'_i = x_i$ falls $X_i = 0$, wobei X_i, $1 \leq i \leq n$, unabhängige Bernoulliverteilte Zufallsvariable mit $\mathrm{W}(X_i = 1) = k/n$ und $\mathrm{W}(X_i = 0) = 1 - k/n$ sind. Der Parameter k wird in der Regel auf einen kleinen Wert gesetzt; die Werte $k = 1$ und $k = 2$ sind häufig anzutreffen. Natürlich kann die Mutationswahrscheinlichkeit im Allgemeinen auf einen beliebigen Wert zwischen 0 und 1 gesetzt werden. Die 1-Bit-Mutation kippt also genau 1 Bit und die Standard-Bit-Mutation mit Mutationswahrscheinlichkeit $1/n$ kippt im Erwartungswert genau 1 Bit. Die beiden Mutationsoperatoren werden in [Doerr et al., 2008b] miteinander verglichen. Wir geben in der vorliegenden Dissertation der Standard-Bit-Mutation den Vorzug, da dieser Mutationsoperator durch einen größeren Sprung das Entfliehen aus einem lokalen Optimum ermöglicht und daher häufiger als Komponente von evolutionären Algo-

rithmen eingesetzt wird.

Wir weisen an dieser Stelle explizit darauf hin, dass der Algorithmus SEMO, genau wie jeder andere multikriterielle evolutionäre Algorithmus, den wir in dieser Dissertation kennenlernen werden, natürlich auch auf monokriterielle Optimierungsprobleme angewandt werden kann. Wenn wir eine Fitnessfunktion $f\colon \mathbb{B}^n \to \mathbb{R}$ unterstellen, dann enthält die Population stets genau ein Individuum, da sichergestellt ist, dass zwei beliebige Individuen vergleichbar sind. Ein neues Individuum x_t wird genau dann akzeptiert, wenn es das in der Population befindliche Individuum x schwach dominiert, d. h., $x_t \succeq x$ bzw. $x_t \geq x$ gilt. In diesem Szenario arbeitet der Algorithmus SEMO folglich genau wie der bekannte evolutionäre Algorithmus (1+1)-EA (siehe z. B. [Droste et al., 2002]).

3. Vergleich der Algorithmen SEMO und DEMO

3.1. Einleitung

Wir untersuchen in diesem Kapitel die Selektion zur Ersetzung. Dazu vergleichen wir die Optimierungszeiten von zwei grundlegenden evolutionären Algorithmen. Auf der einen Seite betrachten wir den aus dem letzten Kapitel bekannten Algorithmus SEMO. Auf der anderen Seite diskutieren wir den Algorithmus DEMO, der durch einen einfachen Mechanismus sicherstellt, dass die Individuen in der Population eine gewisse Diversität aufweisen. Wir analysieren zwei Beispielfunktionen und das Optimierungsziel, eine feste additive Approximationsgüte zu erreichen. Wir werden beweisen, dass sich das Verhalten der beiden Algorithmen deutlich unterscheiden kann.

3.2. Der Algorithmus DEMO

Wir stellen in diesem Abschnitt den Algorithmus DEMO vor, der in dieser Form in [Horoba und Neumann, 2008] eingeführt wird. Wir haben in Abschnitt 2.4 den Algorithmus SEMO vorgestellt, der sich dadurch auszeichnet, dass er für alle nicht dominierten Fitnesswerte, die im Laufe der Optimierung gefunden wurden, einen Repräsentanten in der Population speichert. Dieses Vorgehen ist sicherlich sinnvoll, wenn wir die komplette Paretofront suchen. Wenn wir an einer Approximation der Paretofront interessiert sind, dann werden wir sehen, dass dieser Ansatz den Optimierungsprozess deutlich verlangsamen kann. Dieses Verhalten kann durch eine zu große Population hervorgerufen werden. Im Allgemeinen kann die Größe der Population nur durch die Größe des Suchraumes beschränkt werden, da alle Suchpunkte paarweise unvergleichbar sein können. Da die Zusammensetzung der aktuellen Population unmittelbar die Wahl des als nächstes zu mutierenden Individuums beeinflusst, kann es vorteilhaft sein, für eine Menge von Individuen, die auf ähnliche Fitnesswerte abgebildet werden, nur einen Repräsentanten in der Population zu verwahren. Damit ist die Idee beschrieben, die dem Algorithmus DEMO zugrunde liegt.

Das Verhalten des Algorithmus DEMO wird durch einen Parameter $\delta \in \mathbb{R}^+$ beeinflusst, den wir im Folgenden als *Boxgröße* bezeichnen werden. Wir gliedern den Zielraum der zu optimierenden Fitnessfunktion $f\colon S \to \mathbb{R}^d$ in disjunkte *Boxen* der Größe δ. Formal identifizieren wir jede Box mit einem eindeutigen Index, dem soge-

Algorithmus 3 Diversity-maintaining Evolutionary Multiobjective Optimizer (DEMO)

Eingabe: Fitnessfunktion $f\colon S \to \mathbb{R}^d$ mit endlichem Suchraum S
Ausgabe: Suchpunkte $P \subseteq S$
Parameter: Boxgröße $\delta \in \mathbb{R}^+$

```
1:  t ← 0
2:  wähle x_t ∈ S rein zufällig
3:  P_t ← {x_t}
4:  repeat
5:      t ← t + 1
6:      wähle x ∈ P_{t−1} rein zufällig
7:      erzeuge x_t aus x durch Mutation
8:      if ∄x ∈ P_{t−1}: (f(x) ≻ f(x_t)) ∨ (b(f(x)) ≻ b(f(x_t))) then
9:          P_t ← (P_{t−1} \ {x ∈ P_{t−1} | b(f(x_t)) ⪰ b(f(x))}) ∪ {x_t}
10:     else
11:         P_t ← P_{t−1}
12:     end if
13: until Abbruchkriterium erfüllt
14: return P_t
```

nannten *Boxindex*. Wir definieren $b = (b_1, \ldots, b_d)\colon \mathbb{R}^d \to \mathbb{Z}^d$ durch

$$b_i(y = (y_1, \ldots, y_d)) = \lfloor y_i/\delta \rfloor.$$

Mithilfe dieser Funktion können wir den Boxindex $b(f(x)) = (b_1(f(x)), \ldots, b_d(f(x)))$ der Box

$$[b_1(f(x)) \cdot \delta, (b_1(f(x)) + 1) \cdot \delta) \times \cdots \times [b_d(f(x)) \cdot \delta, (b_d(f(x)) + 1) \cdot \delta)$$

bestimmen, in die ein Suchpunkt $x \in S$ abgebildet wird. Die Algorithmen SEMO und DEMO unterscheiden sich einzig in der Selektion zur Ersetzung. Beide Algorithmen verwerfen ein neues Individuum, wenn die aktuelle Population ein Individuum enthält, dessen Fitness die Fitness des neuen Individuums dominiert. Außerdem verwirft der Algorithmus DEMO ein neues Individuum, wenn die aktuelle Population ein Individuum enthält, dessen Boxindex den Boxindex des neuen Individuums dominiert. Wenn beide Bedingungen nicht erfüllt sind, wird das neue Individuum der aktuellen Population hinzugefügt. Zuvor werden jedoch alle Individuen aus der Population entfernt, deren Boxindizes von dem Boxindex des neuen Individuums schwach dominiert werden. Algorithmus 3 fasst die Beschreibung zusammen.

Wir werden nun zwei charakteristische Eigenschaften des Algorithmus DEMO beschreiben. Die erste Aussage drückt aus, dass alle bisher untersuchten Suchpunkte von mindestens einem Individuum in der aktuellen Population δ-dominiert werden. Wir machen uns in diesem Kapitel die durch $x \succeq_{\delta,+} x'$ definierte Dominanzrelation $x \succeq_\delta x'$ zunutze, um die Notation kurz zu halten (siehe auch Abschnitt 2.2). Wir sprechen in diesem Fall von δ-Dominanz.

Lemma 3.1. *Wir wenden den Algorithmus DEMO mit Boxgröße $\delta \in \mathbb{R}^+$ auf die Fitnessfunktion $f = (f_1, \ldots, f_d)\colon S \to \mathbb{R}^d$ an. Dann gilt nach der Selektion zur Ersetzung*

$$P_t \succeq_\delta \{x_0, \ldots, x_t\}$$

für alle $t \in \mathbb{N}$.

Beweis. Wir beweisen zunächst die Aussage $b(P_t) \succeq b(\{x_0, \ldots, x_t\})$ durch eine vollständige Induktion. Vor Iteration 1 gilt $P_0 = \{x_0\}$ und folglich $b(P_0) \succeq b(\{x_0\})$. Wir nehmen an, dass $b(P_{t-1}) \succeq b(\{x_0, \ldots, x_{t-1}\})$ vor Iteration t gilt und unterscheiden zwei Fälle.

Wenn der Suchpunkt x_t akzeptiert wird, dann ergibt sich

$$P_t = (P_{t-1} \setminus \{x \in P_{t-1} \mid b(f(x_t)) \succeq b(f(x))\}) \cup \{x_t\}.$$

Für alle x_i, $0 \leq i \leq t-1$, gibt es ein $x \in P_{t-1}$ mit $b(f(x)) \succeq b(f(x_i))$. Wenn $b(f(x_t)) \succeq b(f(x))$ gilt, folgt $b(f(x_t)) \succeq b(f(x_i))$. Also gilt $b(P_t) \succeq b(\{x_0, \ldots, x_t\})$.

Wenn der Suchpunkt x_t nicht akzeptiert wird, dann ergibt sich

$$P_t = P_{t-1}.$$

Da es ein $x \in P_{t-1}$ mit $f(x) \succ f(x_t)$ oder $b(f(x)) \succ b(f(x_t))$ gibt, folgt $b(P_t) \succeq b(\{x_0, \ldots, x_t\})$.

Wir zeigen abschließend, dass $b(f(x_t)) \succeq b(f(x)) \implies f(x_t) \succeq_\delta f(x)$ gilt. Diese Behauptung wird durch die Rechnung

$$\begin{aligned} b_i(f(x_t)) \geq b_i(f(x)) &\iff \lfloor f_i(x_t)/\delta \rfloor \geq \lfloor f_i(x)/\delta \rfloor \\ &\implies f_i(x_t)/\delta \geq f_i(x)/\delta - 1 \\ &\iff f_i(x_t) + \delta \geq f_i(x) \end{aligned}$$

bewiesen. □

Wenn wir garantieren möchten, dass die Populationen zudem ausschließlich Individuen enthalten, die nicht von einem im Laufe der Optimierung gefundenen Individuum dominiert werden, dann müssen wir das Akzeptanzkriterium wie folgt adaptieren. Die genannte Anforderung können wir auch so formulieren, dass $y \succ x$ für alle Individuen $x \in P_t$ und $y \in \{x_0, \ldots, x_t\}$ *nicht* gilt. Wir können diese Eigenschaft sicherstellen, indem wir die in Zeile 8 stehende Bedingung in Algorithmus 3 durch

$$\nexists x \in P_{t-1}\colon [(b(f(x)) = b(f(x_t))) \wedge \neg(f(x_t) \succeq f(x))] \vee [b(f(x)) \succ b(f(x_t))]$$

substituieren (siehe auch [Laumanns et al., 2002a]). Wir werden diese Variante nicht weiter untersuchen, da das strengere Akzeptanzkriterium die Exploration wie folgt hemmt. Wenn ein Individuum x_t erzeugt wird, das in eine Box abgebildet wird, in die auch ein Individuum x aus der aktuellen Population abgebildet wird, dann wird das neue Individuum genau dann akzeptiert, wenn $x_t \succeq x$ gilt. In der gleichen Situation akzeptiert der Algorithmus DEMO das Individuum x_t genau dann, wenn $x \succ x_t$

nicht gilt (siehe Algorithmus 3). Der Algorithmus DEMO akzeptiert hier also auch unvergleichbare Individuen.

Die zweite Aussage begrenzt die Größe der Population.

Lemma 3.2. *Wir wenden den Algorithmus DEMO mit Boxgröße $\delta \in \mathbb{R}^+$ auf die Fitnessfunktion $f = (f_1, \ldots, f_d)\colon S \to \mathbb{R}^d$ an. Dann gilt nach der Selektion zur Ersetzung*

$$|P_t| \leq \min_{i \in \{1,\ldots,d\}} \prod_{j \in \{1,\ldots,d\} \setminus \{i\}} |b_j(f(S))|$$

für alle $t \in \mathbb{N}$.

Beweis. Betrachte S. Das Bild $g(S)$ umfasst Fitnessvektoren aus

$$|b(g(S))| \leq \prod_{j \in \{1,\ldots,d\}} |b_j(f(S))|$$

verschiedenen Boxen. Die Population enthält höchstens

$$\min_{i \in \{1,\ldots,d\}} \prod_{j \in \{1,\ldots,d\} \setminus \{i\}} |b_j(f(S))|$$

Individuen, da zwei Fitnessvektoren, die sich in höchstens einer Komponente unterscheiden, vergleichbar sind. Insgesamt folgt die Behauptung. □

Wenn $n = (n_1, \ldots, n_d)$ und $u = (u_1, \ldots, u_d)$ den Nadir- bzw. Utopiapunkt bezeichnen, dann gilt

$$|b_j(f(S))| \leq \left\lfloor \frac{u_j}{\delta} \right\rfloor - \left\lfloor \frac{n_j}{\delta} \right\rfloor + 1.$$

Folglich kann die Populationsgröße mithilfe von Lemma 3.2 durch

$$\left(\frac{\max\{u_1, \ldots, u_d\} - \min\{n_1, \ldots, n_d\}}{\delta} + 2 \right)^{d-1}$$

begrenzt werden.

Wenn wir nicht die additive Approximationsgüte $\delta \in \mathbb{R}^+$ sondern die multiplikative Approximationsgüte $1 + \delta$ anstreben, dann können wir den Algorithmus DEMO wie folgt anpassen. In diesem Fall bietet es sich an, den Boxindex $b = (b_1, \ldots, b_d)\colon \mathbb{R}^d \to \mathbb{Z}^d$ gemäß

$$b_i(y = (y_1, \ldots, y_d)) = \lfloor \log(y_i) / \log(1 + \delta) \rfloor$$

zu bestimmen, wobei wir hier voraussetzen, dass die Fitnessfunktion nur positive Werte annimmt. Die Ergebnisse, die wir im Folgenden nur für die erste Variante des Algorithmus DEMO beweisen werden, lassen sich natürlich auch auf die zweite Variante übertragen, wenn man die zugehörigen Beispielfunktionen entsprechend transformiert. Wenn eine beliebige Fitnessfunktion $f = (f_1, \ldots, f_d)\colon \mathbb{B}^n \to \mathbb{R}^d$ gegeben ist, dann verhält sich die erste Variante des Algorithmus auf f genau wie die zweite auf $g = (g_1, \ldots, g_d)\colon \mathbb{B}^n \to \mathbb{R}^d$ mit $g_i(x) = (1 + \delta)^{f_i(x)/\delta}$. Demzufolge gelten

analoge Ergebnisse auch für die auf die Minimierung der multiplikativen Approximationsgüte zugeschnittene Variante.

Wir betrachten abschließend den Fall, dass ein monokriterielles Optimierungsproblem vorliegt. Wenn wir eine Fitnessfunktion $f\colon \mathbb{B}^n \to \mathbb{R}$ zugrunde legen, dann unterscheidet sich das Verhalten der Algorithmen SEMO und DEMO nicht. Demzufolge verhält sich in diesem Szenario auch der Algorithmus DEMO genau wie der bekannte evolutionäre Algorithmus (1+1)-EA (siehe z. B. [Droste et al., 2002]).

3.3. Ein Problem, für das SEMO ineffizient und DEMO effizient ist

Wir führen in diesem Abschnitt eine Beispielfunktion ein, die verdeutlicht, wie sich eine zu große Population nachteilig auf die effiziente Berechnung einer guten Approximation der Paretofront auswirken kann. Wir benutzen hier den Qualitätsindikator $\mathrm{I}_{\mathrm{add}}$, um die Güte einer Approximation zu messen. Wir beweisen auf der einen Seite, dass der Algorithmus SEMO nicht in der Lage ist, eine beliebige nicht triviale Approximation effizient zu berechnen. Auf der anderen Seite zeigen wir, dass der Algorithmus DEMO eine gute Approximation effizient berechnen kann, wenn wir die Boxgröße richtig einstellen.

Wir definieren eine pseudoboolesche Beispielfunktion $f_\alpha\colon \mathbb{B}^n \to \mathbb{R}^2$, die die 2^n Suchpunkte auf 2^n unvergleichbare Zielpunkte abbildet. Dabei kann der Bereich $[0,\alpha)^2$, in den die Suchpunkte abgebildet werden, durch den Parameter $\alpha \in \mathbb{R}^+$ eingestellt werden. Der Funktionswert von $x \in \mathbb{B}^n$ wird durch $\mathrm{OneMax}(x)$ und $\mathrm{BinVal}(x)$ bestimmt, wobei $\mathrm{OneMax}(x)$ wesentlich größeren Einfluss als $\mathrm{BinVal}(x)$ ausübt. Im Wesentlichen besteht die Paretofront aus $n+1$ Teilabschnitten. Auf welchen Teilabschnitt ein Suchpunkt x abgebildet wird, hängt von $\mathrm{OneMax}(x)$ ab; seine genaue Position auf dem genannten Teilabschnitt hängt zudem von $\mathrm{BinVal}(x)$ ab. Aus den bisherigen Schilderungen folgt beispielsweise, dass der Teilabschnitt, auf den alle Suchpunkte x mit $\mathrm{OneMax}(x) = n/2$ abgebildet werden, aus genau

$$\binom{n}{n/2} = \frac{n \cdot \ldots \cdot (n/2+1)}{(n/2) \cdot \ldots \cdot 1}$$

verschiedenen Zielpunkten besteht, während die Teilabschnitte, auf die alle Suchpunkte x mit $\mathrm{OneMax}(x) = 0$ bzw. $\mathrm{OneMax}(x) = n$ abgebildet werden, aus genau einem Zielpunkt bestehen. Die Suchpunkte verteilen sich also *nicht* gleichmäßig auf die Teilabschnitte; es gibt exponentielle Unterschiede. Der Vollständigkeit halber wiederholen wir zunächst die Definitionen der bekannten Beispielfunktionen OneMax und BinVal.

Definition 3.3. *Sei $n \in \mathbb{N}^+$. Dann definieren wir die verbreiteten Beispielfunktionen* $\mathrm{OneMax}\colon \mathbb{B}^n \to \mathbb{R}$ *und* $\mathrm{BinVal}\colon \mathbb{B}^n \to \mathbb{R}$ *durch*

$$\mathrm{OneMax}(x = (x_1, \ldots, x_n)) := \sum_{i=1}^{n} x_i$$

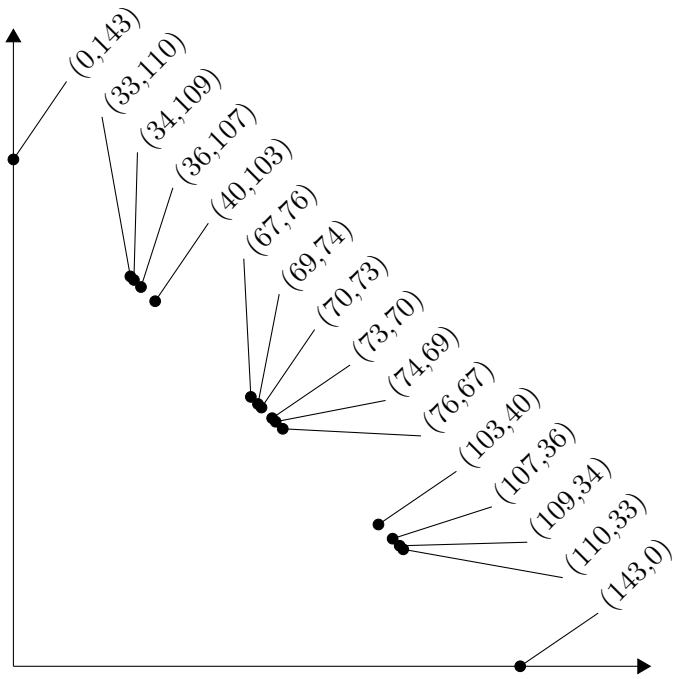


Abbildung 3.1.: Zielraum der Beispielfunktion f_α für $n = 4$ und $\alpha = 2^{n+1} \cdot (n+1) = 160$.

bzw.

$$\mathrm{BinVal}(x = (x_1, \ldots, x_n)) := \sum_{i=1}^{n} x_i \cdot 2^{n-i}.$$

Definition 3.4. *Seien $\alpha \in \mathbb{R}^+$ und $n \in \mathbb{N}^+$. Dann definieren wir die Beispielfunktion $f_\alpha = (f_{\alpha,1}, f_{\alpha,2}) \colon \mathbb{B}^n \to \mathbb{R}^2$ durch*

$$f_{\alpha,1}(x) := \frac{\mathrm{OneMax}(x) + \mathrm{BinVal}(x)/2^{n+1}}{n+1} \cdot \alpha$$

und

$$f_{\alpha,2}(x) := \frac{\mathrm{OneMax}(\overline{x}) + \mathrm{BinVal}(\overline{x})/2^{n+1}}{n+1} \cdot \alpha.$$

Das Komplement $(1 - x_1, \ldots, 1 - x_n)$ eines Bitvektors $x = (x_1, \ldots, x_n)$ wird in Definition 3.4 durch die Notation $\overline{x}$ symbolisiert. Die Beispielfunktion wird in Abbildung 3.1 für $n = 4$ und $\alpha = 160$ veranschaulicht. Um den Zusammenhang zwischen der Paretofront von f_α und der Eingabelänge n zu verdeutlichen, stellt Abbildung 3.2 die Front für $n \in \{2, 4, 8\}$ und $\alpha = 1$ dar. Im Folgenden werden wir darauf verzichten, den Parameter α explizit im Index des Funktionssymbols f zu nennen, d. h., wir werden anstelle von f_α einfach f schreiben.

Das nächste Theorem zeigt, dass der Algorithmus SEMO nicht geeignet ist, um die Paretofront der Beispielfunktion f effizient zu approximieren.

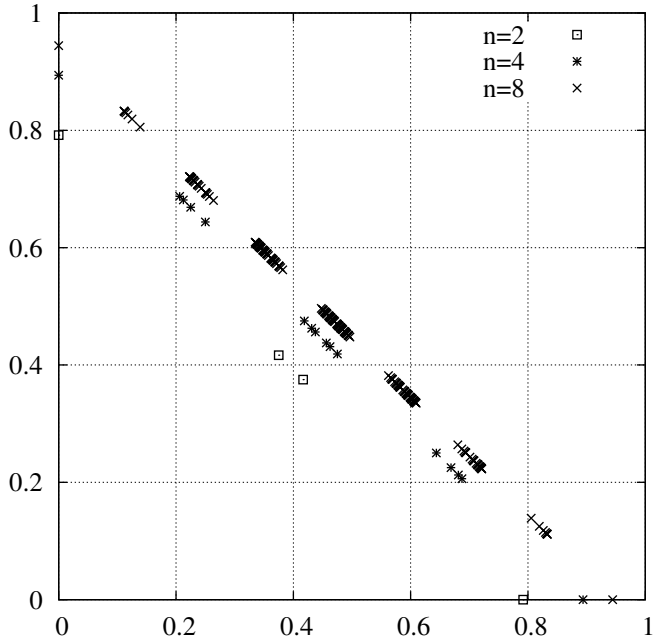


Abbildung 3.2.: Zielraum der Beispielfunktion f_α für $n \in \{2, 4, 8\}$ und $\alpha = 1$.

Theorem 3.5. *Wir wenden den Algorithmus SEMO auf die Fitnessfunktion f mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) > \alpha/3$ für die ersten $2^{\Omega(n)}$ Populationen P mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$.*

Beweis. Der Suchraum besteht aus 2^n Suchpunkten. Die zugehörigen Zielpunkte sind paarweise unvergleichbar. Wir betrachten die Reihenfolge, in der die Suchpunkte in die Population aufgenommen werden. Aus Symmetriegründen ist die Wahrscheinlichkeit, dass ein fester Suchpunkt als i-ter Suchpunkt aufgenommen wird, gleich $1/2^n$. Da in jeder Generation höchstens ein neuer Suchpunkt generiert wird, beträgt die Wahrscheinlichkeit, dass ein fester Suchpunkt in den ersten $2^{0{,}04 \cdot n}$ Generationen aufgenommen wird, höchstens $2^{-0{,}96 \cdot n}$. Offensichtlich gibt es $\binom{n}{i}$ Suchpunkte mit $\mathrm{OneMax}(x) = i$. Also schließen wir mithilfe von Lemma A.13, dass es

$$\sum_{i=0}^{n/3} \binom{n}{i} \leq 2^{H(1/3) \cdot n} = 2^{(-(1/3) \cdot \log(1/3) - (2/3) \cdot \log(2/3)) \cdot n} < 2^{0{,}92 \cdot n}$$

Suchpunkte mit $\mathrm{OneMax}(x) \leq n/3$ gibt, wobei $H(p) = -p \cdot \log p - (1-p) \cdot \log(1-p)$ die binäre Entropie von p bezeichnet. Die Wahrscheinlichkeit, dass einer dieser Suchpunkte in den ersten $2^{0{,}04 \cdot n}$ Generationen in die Population aufgenommen wird, beträgt gemäß der booleschen Ungleichung höchstens $2^{0{,}92 \cdot n} \cdot 2^{-0{,}96 \cdot n} = 2^{-0{,}04 \cdot n} = 2^{-\Omega(n)}$ (siehe Lemma A.4).

Wenn $\mathrm{OneMax}(x) > n/3$ für alle in der Population enthaltenen Suchpunkte x gilt, dann wird der Zielpunkt $f(0^n)$ relativ schlecht approximiert. Der Approximati-

onsfehler beträgt für diesen Punkt mindestens $f_1(x) - f_1(0^n)$, wobei $x \in \mathbb{B}^n$ einen Suchpunkt mit $\text{OneMax}(x) > n/3$ bezeichnet. Wenn wir die Definition von f einsetzen, dann können wir den Approximationsfehler durch

$$f(x) - f(0^n) \geq \frac{n/3+1}{n+1} \cdot \alpha - 0 = \frac{1+3/n}{3+3/n} \cdot \alpha > \frac{\alpha}{3}$$

nach unten abschätzen. Insgesamt folgt das Theorem. □

Der Beweis von Theorem 3.5 zeigt, dass der Algorithmus SEMO durch die immer größer werdende Population daran gehindert wird, eine gute Approximation zu erreichen. Es werden zwar nach und nach immer mehr Pareto-optimale Individuen in die Population aufgenommen, diese Individuen befinden sich jedoch ausschließlich auf dem mittleren Drittel der Paretofront. Folglich werden das obere und das untere Drittel der Paretofront schlecht approximiert. Wenn man eine Approximation der kompletten Paretofront anstrebt, dann kann es vorteilhaft sein, die Population auf wenige Individuen zu begrenzen, die den gefundenen Teil der Paretofront approximieren. Dem Algorithmus DEMO liegt genau diese Idee zugrunde. Das nächste Theorem zeigt, dass der Mechanismus sehr gut funktioniert, wenn wir die Boxgröße auf einen geeigneten Wert setzen.

Theorem 3.6. *Wir wenden den Algorithmus DEMO mit Boxgröße $\delta = \alpha/(n+1)$ auf die Fitnessfunktion f mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\mathrm{I}_{\text{add}}(P) < \alpha/(n+1)$ im Erwartungswert nach $\mathcal{O}(n^2 \cdot \log n)$ Generationen.*

Beweis. Indem wir die Funktionsdefinition aus Definition 3.4 einsetzen, erhalten wir $b(f(x)) = (\text{OneMax}(x) + \lfloor \text{BinVal}(x)/2^{n+1} \rfloor, \text{OneMax}(\overline{x}) + \lfloor \text{BinVal}(\overline{x})/2^{n+1} \rfloor)$ für jeden beliebigen Suchpunkt $x \in \mathbb{B}^n$. Da $\text{BinVal}(x)/2^{n+1} \leq (2^{n+1}-1)/2^{n+1} < 1$ gilt, folgt $b(f(x)) = (\text{OneMax}(x), \text{OneMax}(\overline{x}))$. Die Boxgröße ist also so gewählt, dass die Anzahl der 1-Bits eines Suchpunktes festlegt, in welche Box er abgebildet wird. Folglich besteht die Population aus höchstens $n+1$ Individuen. Wenn die Population für alle $i \in \{0, \ldots, n\}$ ein Individuum $x \in \mathbb{B}^n$ mit $\text{OneMax}(x) = i$ enthält, dann ist der Approximationsfehler gemäß Funktionsdefinition kleiner als $\alpha/(n+1)$. Nehmen wir an, die Population enthält ein Individuum mit $i > 0$ 1-Bits. Dann ist die Wahrscheinlichkeit, einen Suchpunkt mit $i-1$ 1-Bits zu erzeugen, durch $1/|P| \cdot i \cdot 1/n \cdot (1-1/n)^{n-1} > i/(|P| \cdot n \cdot e)$ nach unten beschränkt. Auf analoge Weise zeigt man, dass die Wahrscheinlichkeit, einen Suchpunkt mit $i+1$ 1-Bits zu erzeugen, durch $(n-i)/(|P| \cdot n \cdot e)$ nach unten beschränkt ist. Nehmen wir an, dass der initiale Suchpunkt genau m 1-Bits enthält. Dann beträgt die erwartete Anzahl von Generationen, bis die gewünschte Approximationsgüte erreicht ist, höchstens

$$\begin{aligned}
&\sum_{i=1}^{m} \frac{e \cdot n \cdot (n+1)}{i} + \sum_{i=m}^{n-1} \frac{e \cdot n \cdot (n+1)}{n-i} \\
&\leq 2 \cdot e \cdot n \cdot (n+1) \cdot H_{\lceil n/2 \rceil} \\
&< 2 \cdot e \cdot n \cdot (n+1) \cdot (\ln(n/2+1)+1)
\end{aligned}$$

$$
\begin{aligned}
&= (2 \cdot e \cdot n^2 + 2 \cdot e \cdot n) \cdot (\ln n + \ln(e/2 + e/n)) \\
&\leq \left(1 + \frac{\log(e/2 + e/n)}{\log n} + \frac{1}{n} + \frac{\log(e/2 + e/n)}{n \cdot \log n}\right) \cdot \frac{2 \cdot e}{\log e} \cdot n^2 \cdot \log n \\
&= \mathcal{O}(n^2 \cdot \log n),
\end{aligned}
$$

wobei H_n die n-te harmonische Zahl bezeichnet (siehe Lemma A.17). Insgesamt folgt das Theorem. □

Nutzen wir die Gelegenheit, um die Aussagen der letzten beiden Theoreme zu reflektieren. Theorem 3.5 sagt aus, dass der Algorithmus SEMO die Approximationsgüte $\alpha/3$ nach $2^{c \cdot n}$ Generationen höchstens mit Wahrscheinlichkeit $2^{-c' \cdot n}$ erreicht hat. Aus Theorem 3.6 folgt mithilfe der Markoff-Ungleichung (siehe Lemma A.5), dass der Algorithmus DEMO die Approximationsgüte $\alpha/(n+1)$ nach $t \cdot c'' \cdot n^2 \cdot \log n$ Generationen mindestens mit Wahrscheinlichkeit $1 - 1/t$ erreicht hat, wobei $t \geq 1$ ein Parameter ist, mit dessen Hilfe die Erfolgswahrscheinlichkeit eingestellt werden kann. Wir weisen darauf hin, dass es sich bei c, c' und c'' um positive Konstanten handelt, für die in den Beweisen der genannten Theoreme geeignete Werte zu finden sind. Wenn wir für n, α und t feste Werte, z. B. $n = 1024$, $\alpha = 1$ und $t = 4$, einsetzen, dann können die Ergebnisse wie folgt konkretisiert werden. In diesem Fall gelten die Aussagen für die Konstanten $c = 0{,}04$, $c' = 0{,}04$ und $c'' = 4$. Somit erreicht der Algorithmus SEMO höchstens mit Wahrscheinlichkeit $2^{-0{,}04 \cdot 1024} \approx 4{,}675 \cdot 10^{-13}$ die Approximationsgüte $1/3 \approx 3{,}333 \cdot 10^{-1}$ in $2^{0{,}04 \cdot 1024} \approx 2{,}139 \cdot 10^{12}$ Generationen und der Algorithmus DEMO erreicht mindestens mit Wahrscheinlichkeit $1 - 1/4 = 7{,}500 \cdot 10^{-1}$ die Approximationsgüte $1/(1024+1) \approx 9{,}756 \cdot 10^{-4}$ in $4 \cdot 4 \cdot 1024^2 \cdot \log 1024 \approx 1{,}678 \cdot 10^8$ Generationen. Das Beispiel macht deutlich, dass sich die Effizienz der Algorithmen schon für recht kleine Eingabelänge qualitativ unterscheidet.

In Theorem 3.6 wird die Boxgröße δ in Abhängigkeit von n und α auf einen bestimmten Wert – $\alpha/(n+1)$ – gesetzt. Die gleiche Approximationsgüte kann natürlich auch dann erreicht werden, wenn die Boxgröße auf einen kleineren Wert gesetzt wird. In diesem Fall kann die Populationsgröße jedoch nicht mehr durch $n + 1$ beschränkt werden. Wenn wir die gröbere Abschätzung für die Größe der Population aus Lemma 3.2 benutzen, dann erhalten wir für den allgemeineren Fall $\delta \leq \alpha/(n+1)$ die Laufzeitschranke $\mathcal{O}(\alpha/\delta \cdot n \cdot \log n)$.

Wir weisen an dieser Stelle darauf hin, dass der Algorithmus DEMO offensichtlich nur dann ein anderes Verhalten als der Algorithmus SEMO zeigen kann, wenn mehrere Suchpunkte in die gleiche Box abgebildet werden. Ist das nicht der Fall, arbeiten die Algorithmen SEMO und DEMO auf die gleiche Weise. Wir führen diese Anmerkung nun genauer aus. Sei ein beliebiges multikriterielles Optimierungsproblem $f = (f_1, \ldots, f_d) \colon S \to \mathbb{R}^d$ gegeben. Betrachte den minimalen Abstand

$$
m := \min\{|f_i(x') - f_i(x)| \mid x \in S,\, x' \in S \text{ und } 1 \leq i \leq d \text{ mit } |f_i(x') - f_i(x)| > 0\}
$$

zwischen zwei verschiedenen Zielpunkten, wobei wir den Abstand entlang der einzelnen Dimensionen betrachten. Wenn die Boxgröße δ nicht größer als m ist, dann ist aus

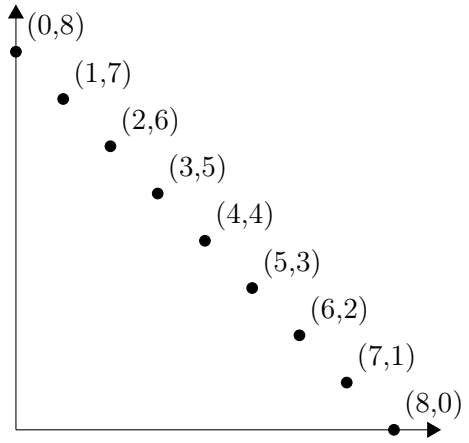


Abbildung 3.3.: Zielraum der Beispielfunktion g für $n = 8$ und $\alpha = n = 8$.

dem genannten Grund sichergestellt, dass sich das Verhalten der Algorithmen SEMO und DEMO nicht unterscheidet. Wenn wir die Beispielfunktion aus Definition 3.4 in Augenschein nehmen, dann gilt für den minimalen Abstand $m \geq \alpha/((n+1) \cdot 2^{n+1})$. Wenn die Boxgröße also nicht größer als $\alpha/((n+1) \cdot 2^{n+1})$ ist, dann zeigen die Algorithmen SEMO und DEMO auf der Beispielfunktion f mit Sicherheit das gleiche Verhalten. Der Algorithmus DEMO kann vor diesem Hintergrund als Verallgemeinerung des Algorithmus SEMO aufgefasst werden.

3.4. Ein Problem, für das DEMO ineffizient und SEMO effizient ist

Wir haben im letzten Abschnitt beispielhaft demonstriert, dass der Algorithmus DEMO wesentlich bessere Resultate als der Algorithmus SEMO liefern kann, wenn eine sinnvolle Boxgröße benutzt wird. Wir stellen in diesem Abschnitt eine einfache Beispielfunktion vor, die deutlich zeigt, dass die Wahl der Boxgröße entscheidend sein kann. Wir beweisen zunächst, dass der Algorithmus DEMO keine brauchbare Approximation liefert, wenn die Boxgröße auf einen von unendlich vielen „ungünstigen“ Werten gesetzt wird. Anschließend zeigen wir, dass der Algorithmus SEMO die Paretofront effizient findet.

Wir definieren eine pseudoboolesche Beispielfunktion $g\colon \mathbb{B}^n \to \mathbb{R}^2$, die die 2^n Suchpunkte auf $n+1$ unvergleichbare Zielpunkte abbildet, wobei der Parameter $\alpha \in \mathbb{R}^+$ den Bereich $[0, \alpha]^2$ festlegt, in den die Suchpunkte abgebildet werden. Die Funktion ist so definiert, dass der Funktionswert von $x \in \mathbb{B}^n$ ausschließlich von $\mathrm{OneMax}(x)$ abhängt. Beachte in diesem Zusammenhang, dass $\mathrm{OneMax}(\overline{x}) = n - \mathrm{OneMax}(x)$ gilt.

Definition 3.7. *Seien $\alpha \in \mathbb{R}^+$ und $n \in \mathbb{N}^+$. Dann definieren wir die Beispielfunktion*

$g_\alpha = (g_{\alpha,1}, g_{\alpha,2}) \colon \mathbb{B}^n \to \mathbb{R}^2$ *durch*

$$g_{\alpha,1}(x) = \frac{\mathrm{OneMax}(x)}{n} \cdot \alpha$$

und

$$g_{\alpha,2}(x) = \frac{\mathrm{OneMax}(\overline{x})}{n} \cdot \alpha.$$

Die Beispielfunktion wird in Abbildung 3.3 für $n = 8$ und $\alpha = 8$ veranschaulicht. Auch hier werden wir im Folgenden anstelle von g_α einfach g schreiben.

Der Algorithmus DEMO kann sich nur dann anders als der Algorithmus SEMO verhalten, wenn mehrere Suchpunkte in eine Box abgebildet werden. Wenn das nicht der Fall ist, dann arbeiten die Algorithmen SEMO und DEMO auf die gleiche Weise. Wir haben im letzten Abschnitt beleuchtet, wie der letztgenannte Algorithmus von einer gröberen Sicht auf den Zielraum profitieren kann. Das nächste Theorem demonstriert, dass eine gröbere Sicht den Optimierungsprozess auch nachteilig beeinflussen kann.

Um das Theorem möglichst elegant beweisen zu können, führen wir zunächst eine vielseitig einsetzbare Beweistechnik ein, die *Drift-Analyse* genannt wird. Mithilfe dieser Technik kann abgeschätzt werden, wie viele Schritte nötig sind, bis ein Markoff-Prozess einen Zustand aus einer Menge von Zuständen erreicht. Die Beweistechnik ist ganz wesentlich Hajek [1982] zu verdanken, auch wenn ihre Grundlagen schon länger bekannt sind. Sie wurde erstmalig von He und Yao [2001, 2002, 2004] für die Analyse von evolutionären Algorithmen eingesetzt. Die Autoren haben sich in den genannten Arbeiten darauf konzentriert, die erwartete Optimierungszeit zu beschränken. Giel und Wegener [2003] haben erkannt, dass nicht nur Aussagen über die erwartete Laufzeit sondern auch über die Erfolgswahrscheinlichkeit möglich sind. Das Drift-Theorem aus der letztgenannten Arbeit ist in zahlreichen Studien benutzt worden, um exponentielle untere Schranken für die Optimierungszeit zu beweisen, die mit überwältigender Wahrscheinlichkeit gelten. Sie kommt beispielsweise in [Giel und Wegener, 2003] für die Analyse eines evolutionären Algorithmus für das Problem, eine größte Paarung (auf Englisch „maximum matching“) zu finden, in [Oliveto et al., 2007b] für die Analyse eines evolutionären Algorithmus für das Knotenüberdeckungsproblem, in [Friedrich et al., 2008b] für die Analyse von populationsbasierten evolutionären Algorithmen mit Diversitätsmechanismen und in [Horoba und Neumann, 2008] für die Analyse eines multikriteriellen evolutionären Algorithmus zum Einsatz. Die Anwendungsbeispiele belegen, dass das genannte Drift-Theorem außerordentlich potent ist. Seine Anwendung ist jedoch häufig mit relativ komplizierten Rechnungen verbunden. In den letzten Jahren sind Versuche unternommen worden, speziellere Varianten des Theorems zu formulieren, die jedoch im Kontext von evolutionären Algorithmen leichter anzuwenden sind. Die vereinfachte Variante namens „Global Gambler's Ruin“ von Happ et al. [2008] kann als ein solcher Versuch gesehen werden. Wir greifen hier auf die Variante von Oliveto und Witt [2010] zurück, die auf Englisch „Simplified Drift Theorem“ genannt wird.

Lemma 3.8 (Vereinfachtes Drift-Theorem)**.** *Seien X_t, $t \geq 0$, die Zufallsvariablen, die einen Markoff-Prozess über einem endlichen Zustandsraum $S \subseteq \mathbb{R}_0^+$ beschreiben. Sei die Drift $\Delta_t(i) := (X_{t+1} - X_t \mid X_t = i)$ für alle $i \in S$ und $t \geq 0$ definiert. Angenommen es gibt ein Intervall $[a, b]$, zwei Konstanten $\delta, \epsilon > 0$ und eine Funktion $r(\ell := b - a)$ mit $1 \leq r(\ell) = \mathrm{o}(\ell / \log \ell)$, sodass für alle $t \geq 0$ die folgenden beiden Bedingungen erfüllt sind.*

1. $\mathrm{E}(\Delta_t(i)) \geq \epsilon$ *für alle* $a < i < b$.
2. $\mathrm{W}(\Delta_t(i) \leq -j) \leq r(\ell)/(1+\delta)^j$ *für alle* $i > a$ *und* $j \in \mathbb{N}$.

Dann gibt es eine Konstante $c^* > 0$, sodass $\mathrm{W}(T^* \leq 2^{c^* \cdot \ell / r(\ell)}) = 2^{-\Omega(\ell / r(\ell))}$ gilt, wobei $T^* := \min\{t \geq 0 \colon X_t \leq a \mid X_0 \geq b\}$ ist.

Wir werden das vereinfachte Drift-Theorem nun einsetzen, um das folgende Theorem zu beweisen.

Theorem 3.9. *Wir wenden den Algorithmus DEMO mit Boxgröße $\delta = \alpha/(2 \cdot m + 1)$, $m \in \mathbb{N}^+$ konstant, auf die Fitnessfunktion g mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) > (m/(2 \cdot m + 1)) \cdot \alpha \geq \alpha/3$ für die ersten $2^{\Omega(n)}$ Populationen P mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$.*

Beweis. Betrachte die Box mit dem Index (m, m). Indem wir die Funktionsdefinition von g benutzen, schließen wir, dass $b(g(x)) = (\lfloor (\mathrm{OneMax}(x)/n) \cdot (2 \cdot m + 1) \rfloor, \lfloor (\mathrm{OneMax}(\overline{x})/n) \cdot (2 \cdot m + 1) \rfloor)$ für alle Suchpunkte $x \in \mathbb{B}^n$ gilt. Eine einfache Umformung zeigt, dass $b(g(x)) = (m, m)$ genau dann gilt, wenn $(m/(2 \cdot m + 1)) \cdot n < \mathrm{OneMax}(x) < ((m+1)/(2 \cdot m + 1)) \cdot n$ gilt. Da der erste Suchpunkt rein zufällig aus dem Suchraum gezogen wird, fällt er mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$ in die fixierte Box.

Wir interpretieren einen Lauf des Algorithmus DEMO als einen Markoff-Prozess. Unser Ziel besteht zunächst darin, eine untere Schranke für die Anzahl von Generationen, bis ein Suchpunkt außerhalb der fixierten Box erschaffen wird, zu zeigen. Damit wir uns Lemma 3.8 zunutze machen können, bilden wir zufällige Populationen P_t mithilfe von

$$h(P_t) := \min\{\min\{\mathrm{OneMax}(x), \mathrm{OneMax}(\overline{x})\} \mid x \in P_t\}$$

auf nicht negative Zufallsvariable $X_t \in \{0, \ldots, \lfloor n/2 \rfloor\}$ ab. Wir wählen $a := (m/(2 \cdot m + 1)) \cdot n$ und $b := (a + (1/2) \cdot n)/2$. Somit gilt $b = ((m/(2 \cdot m + 1)) \cdot n + (1/2) \cdot n)/2 = ((4 \cdot m + 1)/(8 \cdot m + 4)) \cdot n$. Wir beweisen nun, dass die beiden Bedingungen aus Lemma 3.8 erfüllt sind.

Betrachte einen Zustand $X_t = i$ mit $a < i < b = ((4 \cdot m + 1)/(8 \cdot m + 4)) \cdot n$. Ein solcher Zustand entspricht einer Population, die aus genau einem Individuum mit i bzw. $n - i$ 1-Bits besteht. Wir nehmen ohne Beschränkung der Allgemeinheit an, dass das Individuum i 1-Bits umfasst. Die Wahrscheinlichkeit, dass sich die Anzahl der 1-Bits in einer Mutation um $j \in \mathbb{Z}$ verändert, beträgt genau

$$p_j := \sum_{k=0}^{n} \binom{i}{k-j} \cdot \binom{n-i}{k} \cdot \left(\frac{1}{n}\right)^{2 \cdot k - j} \cdot \left(1 - \frac{1}{n}\right)^{n - 2 \cdot k + j}.$$

Wenn $j \geq 0$ ist, dann gilt

$$p_j \leq \binom{n-i}{j} \cdot \left(\frac{1}{n}\right)^j \leq \frac{1}{j!} \leq 2 \cdot \left(\frac{1}{2}\right)^j.$$

Im Erwartungswert ändert sich die Anzahl der 1-Bits also um

$$\sum_{j=-i}^{n-i} p_j \cdot j = \frac{n-i}{n} - \frac{i}{n} = \frac{n-2\cdot i}{n}.$$

Sei $\Delta(i)$ die Drift. Bedingung 1 ist erfüllt, wenn $\mathrm{E}(\Delta(i)) \geq \epsilon$ für eine Konstante $\epsilon > 0$ gilt. Wir schließen

$$\begin{aligned}
\mathrm{E}(\Delta(i)) &= \sum_{j=-i}^{\lfloor n/2 \rfloor - i} p_j \cdot j + \sum_{j=\lfloor n/2 \rfloor - i + 1}^{n-i} p_j \cdot (n - 2\cdot i - j) \\
&= \sum_{j=-i}^{n-i} p_j \cdot j - \sum_{j=\lfloor n/2 \rfloor - i + 1}^{n-i} p_j \cdot (2\cdot j - n + 2\cdot i) \\
&\geq \frac{n-2\cdot i}{n} - 2 \cdot \left(\frac{1}{2}\right)^{\lfloor n/2 \rfloor - i + 1} \cdot n^2 \\
&\geq \frac{1}{4\cdot m + 2} - 2 \cdot \left(\frac{1}{2}\right)^{\lfloor n/2 \rfloor - i} \cdot n^2 \\
&\geq \frac{1}{4\cdot m + 2} - 2 \cdot \left(\frac{1}{2}\right)^{n/(4\cdot m + 2)} \cdot n^2 \\
&= \frac{1}{4\cdot m + 2} - \mathrm{o}(1).
\end{aligned}$$

Um sicherzustellen, dass Bedingung 1 für hinreichend große n erfüllt ist, wählen wir $\epsilon = 1/(8\cdot m + 4)$.

Wenn wir erneut $X_t = i$ annehmen, entspricht $\mathrm{W}(\Delta(i) \leq -j) \leq r(b-a)/(1+\delta)^j$ für $j \in \mathbb{N}$ gerade Bedingung 2. Damit ein Zustand X_{t+1} mit $X_{t+1} \leq i - j$ von X_t erreicht werden kann, müssen mindestens j Bits gekippt werden. Also folgt

$$\mathrm{W}(\Delta(i) \leq -j) \leq \binom{n}{j} \cdot \left(\frac{1}{n}\right)^j \leq \frac{1}{j!} \leq 2 \cdot \left(\frac{1}{2}\right)^j.$$

Um zu garantieren, dass auch Bedingung 2 erfüllt ist, wählen wir $\delta = 1$ und $r(b-a) = 2$.

Wir wenden nun Lemma 3.8 an und schließen, dass es eine Konstante $c^* > 0$ gibt, sodass die Wahrscheinlichkeit, in $2^{c^*\cdot(b-a)} = 2^{c\cdot n}$ Generationen einen Suchpunkt außerhalb der fixierten Box zu erschaffen, durch $2^{-\,\Omega(b-a)} = 2^{-\,\Omega(n)}$ nach oben beschränkt ist, wobei $c := c^* \cdot (b-a)/n > 0$ eine weitere Konstante ist.

Nun fehlt einzig eine Aussage über die Approximationsgüte, wenn die Population aus genau einem Individuum aus der fixierten Box besteht. Eine solche Aussage werden wir nun abschließend herleiten. Wenn die Population aus einem Suchpunkt aus der fixierten Box besteht, dann ist der Approximationsfehler für den Suchpunkt 0^n durch $g_1(x) - g_1(0^n)$ mit $x \in \mathbb{B}^n$ und $\text{OneMax}(x) \geq \lfloor (m/(2 \cdot m + 1)) \cdot n \rfloor + 1$ nach unten beschränkt. Also beträgt der Approximationsfehler mindestens $(m/(2 \cdot m + 1)) \cdot \alpha$. Insgesamt folgt das Theorem. □

Theorem 3.9 gilt für alle Boxgrößen δ, die den relevanten Bereich $[0, \alpha]^2$ des Zielraumes $\mathbb{R}^2$ gleichmäßig in m^2 Boxen aufteilen, wobei $m \geq 3$ eine ungerade Zahl ist. Der Algorithmus DEMO durchsucht den Suchraum, solange die Population aus genau einem Individuum aus der mittleren Box besteht, rein zufällig. Da durch die Standard-Bit-Mutation tendenziell Individuen erzeugt werden, die $n/2$ 0-Bits sowie $n/2$ 1-Bits aufweisen, gelingt es dem Algorithmus mit überwältigender Wahrscheinlichkeit nicht, in exponentiell vielen Generationen die mittlere Box zu verlassen. Selbstverständlich gelten analoge Ergebnisse auch für andere Boxgrößen. Wir ersparen uns an dieser Stelle umfangreiche Fallunterscheidungen, da diese nicht zu einem Erkenntnisgewinn führen würden.

Wir überzeugen uns abschließend davon, dass der Algorithmus SEMO die in diesem Abschnitt betrachtete Beispielfunktion sehr effizient optimieren kann.

Theorem 3.10. *Wir wenden den Algorithmus SEMO auf die Fitnessfunktion g mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\text{I}_{\text{add}}(P) = 0$ im Erwartungswert nach $\mathcal{O}(n^2 \cdot \log n)$ Generationen. Außerdem gilt $\text{I}_{\text{add}}(P) > 0$ im Erwartungswert für die ersten $\Omega(n^{2-c} \cdot \log n)$ Generationen, wobei $c > 0$ eine beliebige Konstante ist.*

Beweis. Die Gleichung $g(x) = ((\text{OneMax}(x)/n) \cdot \alpha, (\text{OneMax}(\overline{x})/n) \cdot \alpha)$ gilt gemäß der Funktionsdefinition von g für jeden beliebigen Suchpunkt $x \in \mathbb{B}^n$. Folglich wird der Funktionswert ausschließlich durch die Anzahl der 1-Bits eines Suchpunktes festlegt und die Population enthält demzufolge höchstens $n + 1$ Individuen. Nehmen wir an, die Population enthält ein Individuum mit $i > 0$ 1-Bits. Dann ist die Wahrscheinlichkeit, einen Suchpunkt mit $i - 1$ 1-Bits zu erzeugen, durch $1/|P| \cdot i \cdot 1/n \cdot (1 - 1/n)^{n-1} > i/(|P| \cdot n \cdot e)$ nach unten beschränkt. Auf analoge Weise zeigt man, dass die Wahrscheinlichkeit, einen Suchpunkt mit $i + 1$ 1-Bits zu erzeugen, durch $(n - i)/(|P| \cdot n \cdot e)$ nach unten beschränkt ist. Nehmen wir an, dass der initiale Suchpunkt genau m 1-Bits enthält. Dann beträgt die erwartete Anzahl von Generationen, bis die gewünschte Approximationsgüte erreicht ist, höchstens

$$\sum_{i=1}^{m} \frac{(n+1) \cdot n \cdot e}{i} + \sum_{i=m}^{n-1} \frac{(n+1) \cdot n \cdot e}{n-i} = \mathcal{O}(n^2 \cdot \log n).$$

Die Rechnung erfolgt ähnlich wie im Beweis von Theorem 3.6.

Wir beweisen nun die untere Schranke für die erwartete Optimierungszeit. Das erste Individuum enthält mit überwältigender Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$ höchstens $(2/3) \cdot n$ 1-Bits. Betrachte eine Phase der Länge $\mathcal{O}(n^{2-3 \cdot d} \cdot \log n)$, wobei $d := c/3$

ist. In dieser Phase kippen in jeder Mutation mit überwältigender Wahrscheinlichkeit $1 - 2^{-\Omega(n^d)}$ höchstens n^d Bits. Definiere das Intervall $I_j := [n - j \cdot n^d, n - (j-1) \cdot n^d)$ für alle $1 \leq j \leq n^{1-d}/6$. Wir warten bis das erste Individuum x mit $\text{OneMax}(x) \in I_{n^{1-d}/6}$ erzeugt wird. Zu diesem Zeitpunkt enthält die Population mindestens $n^{1-d}/6 = \Omega(n^{1-d})$ Individuen. Sei $\text{pot}(P) := \min\{1 \leq j \leq n^{1-d}/6 \mid \exists x \in P\colon \text{OneMax}(x) \in I_j\}$. Dann ist die Wahrscheinlichkeit, ein Individuum x mit $\text{OneMax}(x) \in I_{\text{pot}(P)-1}$ zu erschaffen, nicht größer als

$$\frac{n^d}{|P|} \cdot \frac{\text{pot}(P) \cdot n^d}{n} \leq \frac{n^d}{n^{1-d}/6} \cdot \frac{\text{pot}(P) \cdot n^d}{n} = \frac{6 \cdot \text{pot}(P)}{n^{2-3\cdot d}}.$$

Also dauert es im Erwartungswert mindestens

$$\sum_{i=2}^{n^{1-d}/6} \frac{n^{2-3\cdot d}}{6 \cdot i} = \Omega(n^{2-3\cdot d} \cdot \log n) = \Omega(n^{2-c} \cdot \log n)$$

Generationen bis ein Individuum x mit $\text{OneMax}(x) \in I_1$ erschaffen wird.

Insgesamt folgt das Theorem. □

3.5. Fazit

Selektionsmechanismen, die die Diversität in der Population erhalten, sind eine wichtige Ingredienz für den erfolgreichen Einsatz von evolutionärer Algorithmen für multikriterielle Optimierungsprobleme. Wir haben in diesem Kapitel zwei grundlegende Varianten miteinander verglichen. Auf der einen Seite haben wir den Algorithmus SEMO betrachtet, der für jeden gefundenen nicht dominierten Zielpunkt ein entsprechendes Individuum in der Population verwahrt. Auf der anderen Seite haben wir den Algorithmus DEMO betrachtet, der prinzipiell ähnlich vorgeht. Der entscheidende Unterschied besteht darin, dass der Zielraum zunächst in Boxen einer bestimmten Größe partitioniert wird und ein Individuum nicht mit seinem Zielpunkt sondern mit seinem Boxindex identifiziert wird. Der Algorithmus DEMO ist also eine Verallgemeinerung des Algorithmus SEMO. Wir haben in diesem Kapitel das Optimierungsziel, eine gewisse additive Approximationsgüte zu erreichen, untersucht und dabei bewiesen, dass die Gliederung des Zielraumes in Boxen sowohl ein Fluch als auch ein Segen sein kann. Die konstruierten Beispielfunktionen sind so gestaltet, dass sie die Schwierigkeit für den Algorithmus SEMO bzw. DEMO deutlich herausstellen. Die Erkenntnisse aus diesem Kapitel legen die Vor- und Nachteile der beiden untersuchten Selektionsmechanismen offen. Diese Informationen können hilfreich sein, um für ein bestimmtes Problem einen geeigneten evolutionären Algorithmus zusammenzustellen.

Im nächsten Kapitel – in Kapitel 4 – befassen wir uns mit populären Diversitätsmechanismen, die eingesetzt werden, um eine *feste* Anzahl von Individuen auf der Paretofront zu verteilen.

4. Vergleich der Algorithmen SEMO und RADEMO

4.1. Einleitung

In diesem Kapitel widmen wir uns erneut dem Einfluss der Selektion zur Ersetzung. Wir vergleichen die Optimierungszeiten von zwei grundlegenden evolutionären Algorithmen. Auf der einen Seite betrachten wir den bereits bekannten Algorithmus SEMO. Auf der anderen Seite nehmen wir einen neuen Algorithmus namens RADEMO in Augenschein. Dabei werden wir zwei verschiedene Ausprägungen des letztgenannten Algorithmus betrachten (siehe Abschnitt 4.2). Dieser Algorithmus basiert im Gegensatz zu den zuvor betrachteten Algorithmen SEMO und DEMO auf einer Population, die eine feste Anzahl μ von Individuen enthält. Mithilfe geeigneter Diversitätsmechanismen soll sichergestellt werden, dass sich die Individuen in der Population nach einigen Generationen auf der Paretofront verteilen. Die untersuchten Diversitätsmechanismen sind den populären Algorithmen NSGA2 [Deb et al., 2000] und SPEA2 [Zitzler et al., 2001] entnommen. Wir analysieren zwei Beispielfunktionen und das Optimierungsziel, eine feste additive Approximationsgüte zu erreichen.

4.2. Der Algorithmus RADEMO

Der Algorithmus SEMO, den wir in Abschnitt 2.4 eingeführt haben, und der Algorithmus DEMO, den wir in Abschnitt 3.2 kennengelernt haben, zeichnen sich durch eine Population aus, deren Größe veränderlich ist. Für die Populationsgröße können im Allgemeinen nur die triviale untere Schranke 1 und die triviale obere Schranke $|S|$ angegeben werden, wenn keine Informationen über die zu optimierende Fitnessfunktion $f\colon S \to \mathbb{R}^d$ vorliegen. Auf der anderen Seite ist es in vielen Anwendungen vorteilhaft, die Populationsgröße möglichst genau einstellen zu können. Die Population sollte beispielsweise möglichst klein sein, um eine effiziente Optimierung zu ermöglichen. Andererseits sollte die Population hinreichend groß sein, um die Paretofront angemessen approximieren zu können. Der Algorithmus DEMO stellt eine gute Wahl dar, wenn a priori Informationen über den Wertebereich der zu optimierenden Fitnessfunktion vorhanden sind, beispielsweise eine untere Schranke $F_{\min}$ und eine obere Schranke $F_{\max}$ für die Funktionswerte. In diesem Fall kann die Population gemäß Lemma 3.2 höchstens $P_{\max} = \mathcal{O}(((F_{\max} - F_{\min})/\delta)^{d-1})$ Individuen enthalten und die Populationsgröße kann durch eine geeignete Wahl der Boxgröße δ begrenzt werden. Wir müssen beachten, dass sich in Abhängigkeit von der zu optimierenden Fitnessfunktion und der Präzision der Grenzen $F_{\min}$ und $F_{\max}$ in der

Algorithmus 4 Rank- And Density-based Evolutionary Multiobjective Optimizer (RADEMO)

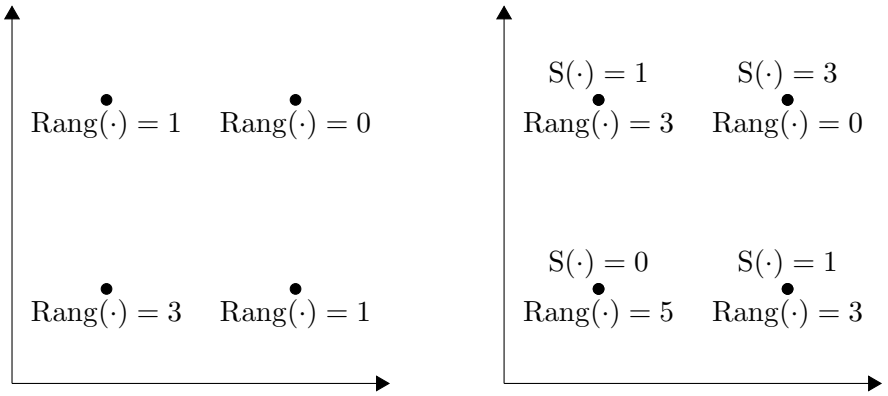
Eingabe: Fitnessfunktion $f\colon S \to \mathbb{R}^d$ mit endlichem Suchraum S
Ausgabe: Suchpunkte $P \subseteq S$
Parameter: Populationsgröße $\mu \in \mathbb{N}^+$

1: $t \leftarrow 0$
2: wähle $x_{t,1}, \ldots, x_{t,\mu} \in S$ unabhängig und rein zufällig
3: $P_t \leftarrow \{x_{t,1}, \ldots, x_{t,\mu}\}$
4: **repeat**
5: $\quad t \leftarrow t + 1$
6: $\quad$ wähle $x \in P_{t-1}$ rein zufällig
7: $\quad$ erzeuge x_t aus x durch Mutation
8: $\quad Q \leftarrow P_{t-1} \cup \{x_t\}$
9: $\quad Q' \leftarrow \operatorname{argmax}_{x \in Q} \mathrm{Rang}_Q(x)$
10: $\quad Q'' \leftarrow \operatorname{argmin}_{x \in Q'} \mathrm{Dichte}_{Q'}(x)$ bzw. $Q'' \leftarrow \operatorname{argmin}_{x \in Q'} \mathrm{Dichte}_Q(x)$
11: $\quad$ wähle $x \in Q''$ rein zufällig
12: $\quad P_t \leftarrow (P_{t-1} \cup \{x_t\}) \setminus \{x\}$
13: **until** Abbruchkriterium erfüllt
14: **return** P_t

letzten Population deutlich weniger als $P_{\max}$ Individuen befinden können. Wenn die gelieferten Alternativen die Entscheidungsinstanz nicht zufriedenstellen, dann muss die Optimierung in diesem Fall mit einer kleineren Boxgröße wiederholt werden.

Wenn über den Wertebereich der zu optimierenden Funktion nichts bekannt ist oder die Entscheidungsinstanz eine bestimmte Anzahl von Alternativen bevorzugt, dann greift man in vielen Anwendungen zu Algorithmen, die mit einer vor Beginn der Optimierung einzustellenden festen Populationsgröße μ arbeiten. Diese Algorithmen versuchen, die Individuen in der Population mithilfe unterschiedlicher Diversitätsmechanismen auf der Paretofront zu verteilen. Die in der Praxis beliebten Algorithmen „Nondominated Sorting Genetic Algorithm 2 (NSGA2)“ [Deb et al., 2000] und „Strength Pareto Evolutionary Algorithm 2 (SPEA2)“ [Zitzler et al., 2001] gehören zu den prominenten Vertretern dieser Klasse von Algorithmen. Da die Selektion zur Ersetzung in beiden Algorithmen auf ähnliche Weise funktioniert, werden wir im Folgenden einen Rahmen für zwei Algorithmen vorstellen, die den genannten Algorithmen nachempfunden sind. Sie spiegeln einerseits die Arbeitsweisen der genannten Algorithmen wider und sind andererseits einer theoretischen Analyse zugänglich, da sie auf komplizierte Variationsoperatoren verzichten.

Der zu entwerfende Rahmenalgorithmus arbeitet mit einer Population der Größe $\mu \in \mathbb{N}^+$. Anfangs besteht die Population aus μ Individuen, die unabhängig und rein zufällig aus dem Suchraum gezogen werden. In jeder Generation wird rein zufällig ein Individuum aus der Population ausgewählt. Anschließend wird dieses Individuum kopiert und die Kopie einer Mutation unterzogen. Das neue Individuum wird danach der aktuellen Population hinzugefügt. Da sich die Population nun aus $\mu + 1$ Indivi-



(a) Beispiel 1: Rang aus NSGA2.

(b) Beispiel 2: Rang aus SPEA2.

Abbildung 4.1.: Beispiele für die Rangbestimmung in den Algorithmen NSGA2 und SPEA2.

duen zusammensetzt, muss ein Individuum aus der Population entfernt werden, um ein Anwachsen der Population zu verhindern. Die Idee besteht darin, zunächst die Individuen zu suchen, die von möglichst vielen Individuen in der Population dominiert werden. In einem zweiten Schritt wird aus diesen Individuen ein Individuum ausgewählt, das eine möglichst kleine Dichte aufweist. Den Begriff „Dichte“ werden wir im Anschluss präzisieren. Das auf diese Weise bestimmte Individuum wird anschließend gelöscht. Dieser Prozess wird fortgesetzt bis ein Abbruchkriterium erfüllt ist.

Wir präzisieren nun, wie der erste Schritt der Selektion zur Ersetzung realisiert werden kann. Betrachte eine Menge P von Individuen. Der erste Selektionsschritt basiert einzig auf der Dominanzrelation. Wir berechnen für jedes Individuum den sogenannten *Rang*. Wir definieren die Abbildung $\mathrm{Rang}_P \colon P \to \mathbb{N}$ zunächst auf die naheliegende Weise

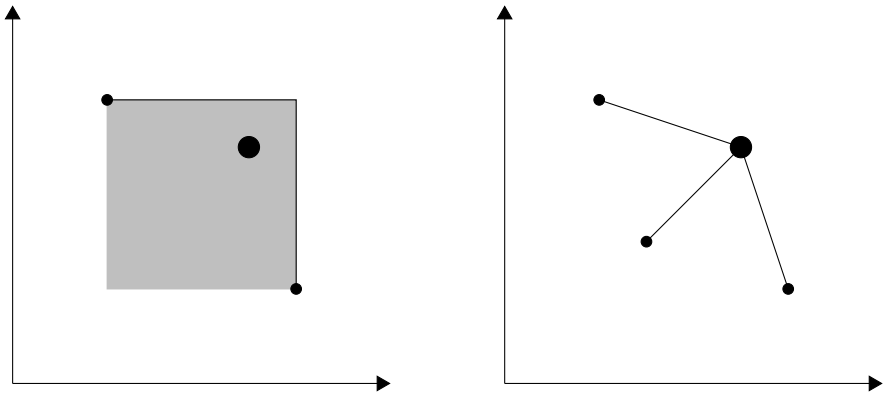
$$\mathrm{Rang}_P(x) := |\{y \in P \mid y \succ x\}| = \sum_{y \in P | y \succ x} 1.$$

Der Rang eines Suchpunktes $x \in P$ gibt also an, wie viele Suchpunkte $y \in P$ diesen dominieren. Demzufolge sind Individuen mit einem kleinen Rang besser als Individuen mit einem großen Rang. Auf genau diese Weise werden die Ränge in dem Algorithmus NSGA2 bestimmt. Die Rangbestimmung wird in Abbildung 4.1(a) verdeutlicht. In dem Algorithmus SPEA2 werden die Ränge gemäß

$$\mathrm{Rang}_P(x) := \sum_{y \in P | y \succ x} \mathrm{S}_P(y)$$

berechnet, wobei die dominierenden Individuen zusätzlich mit

$$\mathrm{S}_P(x) := |\{y \in P \mid x \succ y\}|$$



(a) Beispiel 1: Dichte aus NSGA2. (b) Beispiel 2: Dichte aus SPEA2.

Abbildung 4.2.: Beispiele für die Dichtebestimmung in den Algorithmen NSGA2 und SPEA2.

gewichtet werden. Das Gewicht eines Individuums entspricht also der Anzahl der dominierten Individuen. Zitzler et al. [2001] nennen $\mathrm{S}_P(x)$ die *Stärke* des Individuums $x \in P$. Die Summe der Stärken der Individuen, die ein Individuum dominieren, ergibt folglich dessen Rang. Diese Variante der Rangbestimmung wird in Abbildung 4.1(b) verdeutlicht.

Wir beschreiben nun, wie der zweite Schritt der Selektion zur Ersetzung implementiert werden kann. Betrachte erneut eine Menge P von Individuen. Um die Dichte des Bereiches, in dem sich ein Individuum befindet, schätzen zu können, müssen wir die Distanz zwischen zwei Individuen messen können. Wir benötigen also eine geeignete Metrik $d\colon S \times S \to \mathbb{R}_0^+$. Um die Vorgehensweise der Algorithmen NSGA2 und SPEA2 beschreiben zu können, definieren wir die von der euklidischen Vektornorm induzierte Metrik

$$d(x,y) := \left(\sum_{i=1}^{d} (d_i(x,y))^2 \right)^{1/2},$$

wobei $d_i(x,y) := |f_i(y) - f_i(x)|$ ist. Wir können eine Metrik $d\colon S \times S \to \mathbb{R}_0^+$ kanonisch auf Mengen von Suchpunkten verallgemeinern. Dazu definieren wir $d\colon \mathrm{Pot}(S) \times \mathrm{Pot}(S) \to \mathbb{R}_0^+ \cup \{\infty\}$ durch

$$d(X,Y) := \begin{cases} \min\{d(x,y) \mid x \in X \wedge y \in Y\} & \text{falls } X \neq \emptyset \text{ und } Y \neq \emptyset \\ \infty & \text{falls } X = \emptyset \text{ oder } Y = \emptyset. \end{cases}$$

Der Algorithmus NSGA2 schätzt die Dichte eines Individuums durch eine Auswertung der durch

$$\mathrm{Dichte}_P(x) := \sum_{i=1}^{d} d_i(\{y \in P \setminus \{x\} \mid f_i(y) \leq f_i(x)\}, \{y \in P \setminus \{x\} \mid f_i(y) \geq f_i(x)\})$$

definierten Abbildung $\text{Dichte}_P \colon P \to \mathbb{R}_0^+ \cup \{\infty\}$. Anschaulich kann man sich um jeden Zielpunkt aus $f(P)$ den größtmöglichen Hyperquader vorstellen, in dem sich keine anderen Zielpunkte aus $f(P)$ befinden (siehe Abbildung 4.2(a)). Die Dichte ist dann proportional zur Summe der Kantenlängen dieses Hyperquaders. Das bedeutet insbesondere, dass die Dichte eines einzelnen „Randpunktes“ aus P unendlich groß ist. Dieses Verhalten ist gewünscht, da es als sinnvoll angesehen wird, die Randpunkte zu erhalten. Dieses Verfahren stößt natürlich an seine Grenzen, wenn sehr viele Ziele zu optimieren sind, da dann in der Regel alle Individuen Randpunkte darstellen. Die Dichte wird in dem Algorithmus SPEA2 durch eine Auswertung der durch

$$\text{Dichte}_P(x) := (d_1, \ldots, d_{|P|})$$

definierten Abbildung $\text{Dichte}_P \colon P \to (\mathbb{R}_0^+)^{|P|}$ geschätzt, wobei die einzelnen Distanzen $d_1, \ldots, d_{|P|}$ durch $\{d_1, \ldots, d_{|P|}\} = \{d(x, y) \mid y \in P\}$ und $d_1 \leq \cdots \leq d_{|P|}$ auf wohldefinierte Weise gegeben sind. Anschaulich beschreibt dieser Vektor die Distanz zum nächsten Zielpunkt aus P, zum zweitnächsten Zielpunkt aus P usw. Hier verwenden wir die euklidische Distanz als Entfernungsmaß (siehe Abbildung 4.2(b)). Andere Distanzmaße sind natürlich denkbar. Wir weisen darauf hin, dass stets $d_1 = 0$ gilt. Um zwei Dichtevektoren miteinander zu vergleichen, wird die lexikographische Ordnung herangezogen.

Wir spezifizieren nun den Ablauf der Selektion zur Ersetzung. Der Algorithmus beginnt mit einer Menge Q von Individuen, die sich aus der aktuellen Population und dem neuen Individuum zusammensetzt. Dann wird die Menge, aus der das zu löschende Individuum gewählt wird, zunächst auf $Q' := \text{argmax}_{x \in Q} \text{Rang}_Q(x)$ eingeschränkt. Anschließend setzt der Algorithmus NSGA2 die Kandidatenmenge auf $Q'' := \text{argmin}_{x \in Q'} \text{Dichte}_{Q'}(x)$. Der Algorithmus SPEA2 setzt die Kandidatenmenge auf $Q'' := \text{argmin}_{x \in Q'} \text{Dichte}_Q(x)$, wobei die Dichtevektoren mithilfe der lexikographischen Ordnung sortiert werden. Beachte, dass die Dichte im ersten Fall bezüglich Q' und im zweiten Fall bezüglich Q bestimmt wird. Anschließend wird das zu löschende Individuum rein zufällig aus Q'' gezogen. Algorithmus 4 fasst die Beschreibung zusammen.

Zusammenfassend kann man sagen, dass der Rahmenalgorithmus RADEMO zunächst aus den Individuen, die den größten Rang aufweisen, die Individuen auswählt, die die kleinste Dichte aufweisen. Daraufhin wird aus diesen Individuen, das zu löschende Individuum rein zufällig gezogen. Wir weisen an dieser Stelle noch einmal auf die Bedeutung der hier verwendeten Begrifflichkeiten „Rang“ und „Dichte“ hin. Da der Rang eines Individuums prinzipiell angibt, von wie vielen Individuen es dominiert wird, werden Individuen mit einem niedrigen Rang bevorzugt. Außerdem werden Individuen mit einer hohen Dichte bevorzugt, da ein Individuum umso weiter von den anderen Individuen entfernt ist, je größer seine Dichte ist.

Wir thematisieren abschließend, was passiert, wenn wir den dargebotenen Algorithmus RADEMO einsetzen, um eine Fitnessfunktion $f \colon \mathbb{B}^n \to \mathbb{R}$ zu optimieren. Das Folgende gilt übrigens für beide Varianten des Algorithmus RADEMO. In diesem Fall entsprechen die Individuen mit dem größten Rang den Individuen mit der kleinsten

Fitness. Außerdem stimmen die Dichten von allen genannten Individuen überein. Der Algorithmus entfernt also rein zufällig ein Individuum mit der kleinsten Fitness aus der Population. Wir haben damit gezeigt, dass der Algorithmus RADEMO als Verallgemeinerung des bekannten evolutionären Algorithmus (μ+1)-EA aufgefasst werden kann (siehe z. B. [Witt, 2006, 2008] wie auch Kapitel 11).

4.3. Ein Problem, für das SEMO ineffizient und RADEMO effizient ist

Wir machen in diesem Abschnitt deutlich, in welchen Situationen der Algorithmus RADEMO dem Algorithmus SEMO vorzuziehen ist. Dazu greifen wir auf die aus Abschnitt 3.3 bekannte Beispielfunktion f_α zurück, die wir im Anschluss f nennen werden. Die Funktion ist wie folgt definiert.

Definition 3.4. *Seien $\alpha \in \mathbb{R}^+$ und $n \in \mathbb{N}^+$. Dann definieren wir die Beispielfunktion $f_\alpha = (f_{\alpha,1}, f_{\alpha,2})\colon \mathbb{B}^n \to \mathbb{R}^2$ durch*

$$f_{\alpha,1}(x) := \frac{\mathrm{OneMax}(x) + \mathrm{BinVal}(x)/2^{n+1}}{n+1} \cdot \alpha$$

und

$$f_{\alpha,2}(x) := \frac{\mathrm{OneMax}(\overline{x}) + \mathrm{BinVal}(\overline{x})/2^{n+1}}{n+1} \cdot \alpha.$$

Wir haben in Abschnitt 3.3 bewiesen, dass der Algorithmus SEMO mit überwältigender Wahrscheinlichkeit exponentiell viele Generationen benötigt, um eine angemessene additive Approximationsgüte zu erreichen. Das Theorem lautet wie folgt.

Theorem 3.5. *Wir wenden den Algorithmus SEMO auf die Fitnessfunktion f mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) > \alpha/3$ für die ersten $2^{\Omega(n)}$ Populationen P mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$.*

Wir beweisen nun, dass der Algorithmus RADEMO ähnlich gute Resultate wie der Algorithmus DEMO liefert, wenn die Population hinreichend viele Individuen enthält. Wir erinnern an dieser Stelle daran, dass wir in Abschnitt 3.3 das Verhalten des Algorithmus DEMO für die Beispielfunktion f untersucht haben. Im Erwartungswert erreicht dieser Algorithmus nach $\mathcal{O}(n^2 \cdot \log n)$ Generationen eine Population P mit $\mathrm{I}_{\mathrm{add}}(P) < \alpha/(n+1)$, wenn die Boxgröße $\delta = \alpha/(n+1)$ benutzt wird (siehe Theorem 3.6). Das nächste Theorem zeigt, dass auch die in diesem Kapitel thematisierten Diversitätsmechanismen ausreichen, um eine gute Approximation schnell zu finden.

Theorem 4.1. *Wir wenden den Algorithmus RADEMO (beide Varianten) mit Populationsgröße $\mu \geq n+1$ auf die Fitnessfunktion f mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) < \alpha/(n+1)$ im Erwartungswert nach $\mathcal{O}(\mu \cdot n \cdot \log n)$ Generationen.*

Beweis. Wir untersuchen zunächst, wie die Selektion zur Ersetzung ein Individuum auswählt. In diesem Beweis verwenden wir die Abkürzungen $|\cdot|_1 := \text{OneMax}(\cdot)$ und $|\cdot|_2 := \text{BinVal}(\cdot)$. Wir betrachten zwei Suchpunkte x und y mit i 1-Bits. Dann gilt

$$\begin{aligned}
&|f_1(y) - f_1(x)| \\
&= \left| \frac{|y|_1 + |y|_2/2^{n+1}}{n+1} \cdot \alpha - \frac{|x|_1 + |x|_2/2^{n+1}}{n+1} \cdot \alpha \right| \\
&\leq \frac{|1^i 0^{n-i}|_1 + |1^i 0^{n-i}|_2/2^{n+1}}{n+1} \cdot \alpha - \frac{|0^{n-i}1^i|_1 + |0^{n-i}1^i|_2/2^{n+1}}{n+1} \cdot \alpha \\
&= \frac{i + \sum_{j=n-i}^{n-1} 2^{j-n-1} - i - \sum_{j=0}^{i-1} 2^{j-n-1}}{n+1} \cdot \alpha \\
&< \frac{1}{2} \cdot \frac{1}{n+1} \cdot \alpha.
\end{aligned}$$

Wir betrachten nun zwei Suchpunkte x und y mit i bzw. $i+1$ 1-Bits. Dann gilt

$$\begin{aligned}
&|f_1(y) - f_1(x)| \\
&= \left| \frac{|y|_1 + |y|_2/2^{n+1}}{n+1} \cdot \alpha - \frac{|x|_1 + |x|_2/2^{n+1}}{n+1} \cdot \alpha \right| \\
&\geq \frac{|0^{n-i-1}1^{i+1}|_1 + |0^{n-i-1}1^{i+1}|_2/2^{n+1}}{n+1} \cdot \alpha - \frac{|1^i 0^{n-i}|_1 + |1^i 0^{n-i}|_2/2^{n+1}}{n+1} \cdot \alpha \\
&= \frac{i+1 + \sum_{j=0}^{i} 2^{j-n-1} - i - \sum_{j=n-i}^{n-1} 2^{j-n-1}}{n+1} \cdot \alpha \\
&> \frac{1}{2} \cdot \frac{1}{n+1} \cdot \alpha.
\end{aligned}$$

Außerdem gilt

$$\begin{aligned}
|f_2(y) - f_2(x)| &= \left| \left(\frac{n + (2^n - 1)/2^n}{n+1} \cdot \alpha - f_1(y) \right) - \left(\frac{n + (2^n - 1)/2^n}{n+1} \cdot \alpha - f_1(x) \right) \right| \\
&= |f_1(x) - f_1(y)|.
\end{aligned}$$

Da alle Individuen bezüglich $\succ$ paarweise unvergleichbar sind, wird jedem Individuum der Rang 0 zugewiesen. Die Selektion zur Ersetzung entfernt also ein Individuum mit der kleinsten Dichte aus der Population. Wir betrachten zunächst die SPEA2-artige Dichteberechnung. Dann gelten für alle x und y mit $\text{OneMax}(x) = \text{OneMax}(y)$

$$d(x,y) < (\sqrt{2} \cdot \alpha)/(2 \cdot (n+1))$$

und für alle x und y mit $\text{OneMax}(x) \neq \text{OneMax}(y)$

$$d(x,y) > (\sqrt{2} \cdot \alpha)/(2 \cdot (n+1)).$$

Folglich ist die Dichte eines Individuums $x \in P$, für das es ein weiteres Individuum $y \in P$ mit $\text{OneMax}(y) = \text{OneMax}(x)$ gibt, kleiner als die Dichte eines Individuums

$x' \in P$, für das es *kein* weiteres Individuum $y' \in P$ mit $\text{OneMax}(y') = \text{OneMax}(x')$ gibt. Wir betrachten nun die NSGA2-artige Dichteberechnung. Die Dichte eines Individuums, für das es *kein* weiteres Individuum mit der gleichen Anzahl von 1-Bits gibt, ist größer als $(2 \cdot \alpha)/(n+1)$. Wenn es ein solches Individuum gibt, dann gilt aufgrund der Populationsgröße, dass es ein weiteres Individuum $x \in P$ gibt, für das ein weiteres Individuum $y \in P$ mit $\text{OneMax}(y) = \text{OneMax}(x)$ und ein weiteres Individuum $z \in P$ mit $|\,\text{OneMax}(z) - \text{OneMax}(x)| \leq 1$ gibt. Demzufolge ist die Dichte von x oder die Dichte von y kleiner als $\alpha/(n+1)$.

Wir haben in beiden Fällen bewiesen, dass ein Individuum x nur dann aus der Population entfernt wird, wenn sichergestellt ist, dass sie ein weiteres Individuum y mit $\text{OneMax}(y) = \text{OneMax}(x)$ enthält, da wir die Anforderung $\mu > n$ an die Populationsgröße μ stellen. Der Algorithmus hat das Optimierungsziel also erreicht, sobald er für jedes $i \in \{0, \ldots, n\}$ ein Individuum x mit $\text{OneMax}(x) = i$ erzeugt hat.

Sei x ein Individuum in der Population. Dann beträgt die Wahrscheinlichkeit, ein Individuum x' mit $\text{OneMax}(x') = \text{OneMax}(x) + 1$ zu erzeugen, mindestens $\text{OneMax}(\overline{x})/(e \cdot \mu \cdot n)$. Außerdem beträgt die Wahrscheinlichkeit, ein Individuum x' mit $\text{OneMax}(x') = \text{OneMax}(x) - 1$ zu erzeugen, mindestens $\text{OneMax}(x)/(e \cdot \mu \cdot n)$. Wenn die anfängliche Population ein Individuum mit m 1-Bits enthält, dann wird die gewünschte Approximationsgüte im Erwartungswert nach

$$\sum_{i=1}^{m} \frac{e \cdot \mu \cdot n}{i} + \sum_{i=m}^{n-1} \frac{e \cdot \mu \cdot n}{n-i} = \mathcal{O}(\mu \cdot n \cdot \log n).$$

Generationen erreicht. □

Wir nutzen die Gelegenheit, um den letzten Beweis Revue passieren zu lassen. Im Beweis haben wir abgeschätzt, wie lange es im Erwartungswert dauert, bis ein Individuum x' mit $\text{OneMax}(x') = i - 1$ bzw. $\text{OneMax}(x') = i + 1$ erzeugt wird, wenn die Population ein Individuum x mit $\text{OneMax}(x) = i$ enthält. Wir haben die Wahrscheinlichkeit, ein solches Individuum in einer Generation zu erzeugen, durch die Wahrscheinlichkeit, das Individuum x zur Mutation auszuwählen und genau ein 1- bzw. 0-Bit zu kippen, abgeschätzt. Die letztgenannte Wahrscheinlichkeit beträgt $i/(e \cdot \mu \cdot n)$ bzw. $(n-i)/(e \cdot \mu \cdot n)$. Dann haben wir angenommen, dass die Wartezeit auf dieses Ereignis geometrisch verteilt ist, und daraus geschlossen, dass die erwartete Wartezeit höchstens $(e \cdot \mu \cdot n)/i$ bzw. $(e \cdot \mu \cdot n)/(n-i)$ beträgt. Ist diese Schranke zu pessimistisch? Wenn die Population nicht nur ein Individuum x mit $\text{OneMax}(x) = i$ sondern j solche Individuen enthält, dann kann die relevante Wahrscheinlichkeit offenbar durch $(j \cdot i)/(e \cdot \mu \cdot n)$ bzw. $(j \cdot (n-i))/(e \cdot \mu \cdot n)$ abgeschätzt werden. Wir betrachten im nächsten Absatz einige Argumente aus einem Beweis einer oberen Schranke für den evolutionären Algorithmus (μ+1)-EA, die sich – wie wir anschließend sehen werden – leider nicht verallgemeinern lassen. Auch wenn wir die obere Schranke aus Theorem 4.1 auf diese Weise nicht verkleinern können, sind die folgenden Betrachtungen dennoch interessant.

Wir skizzieren nun die zentralen Argumente aus dem Beweis der oberen Schranke $\mathcal{O}(\mu \cdot n + n \cdot \log n)$ für die Optimierungszeit des evolutionären Algorithmus (μ+1)-EA

für die Beispielfunktion OneMax aus [Witt, 2006]. Wir betrachten darin die Situation, dass die Population genau ein Individuum x mit $\text{OneMax}(x) = i$ enthält. Wir interessieren uns dafür, wie lange es dauert, bis ein Individuum x' mit $\text{OneMax}(x') > i$ erzeugt wird. Witt [2006] argumentiert wie folgt. Wenn die Population genau j Individuen mit i 1-Bits enthält, dann ist die Wahrscheinlichkeit, ein Individuum x' mit $\text{OneMax}(x') > i$ zu erzeugen, nicht kleiner als

$$\frac{j}{\mu} \cdot \frac{n-i}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{j \cdot (n-i)}{e \cdot \mu \cdot n}.$$

Außerdem ist die Wahrscheinlichkeit, ein Individuum x' mit $\text{OneMax}(x') = i$ zu erzeugen, nicht kleiner als $(j/\mu) \cdot (1 - 1/n)^n \geq j/(2 \cdot e \cdot \mu)$. Die erwartete Zeit, bis die Population mindestens $\min\{\lceil n/(n-i) \rceil, \mu\}$ Individuen y mit $\text{OneMax}(y) \geq i$ enthält, ist demzufolge nicht größer als

$$\sum_{i=1}^{\lceil n/(n-i) \rceil - 1} \frac{2 \cdot e \cdot \mu}{j} \leq 2 \cdot e \cdot \mu \cdot \ln\left(\frac{e \cdot n}{n-i}\right).$$

Anschließend ist die erwartete Zeit, bis ein Individuum x' mit $\text{OneMax}(x') > i$ erzeugt wird, nicht größer als

$$\frac{e \cdot \mu \cdot n}{\min\{\lceil n/(n-i) \rceil, \mu\} \cdot (n-i)} \leq \frac{e \cdot \mu \cdot n}{n} + \frac{e \cdot \mu \cdot n}{\mu \cdot (n-i)}.$$

Wir erhalten die Behauptung, indem wir die Summe

$$\sum_{i=0}^{n-1} 2 \cdot e \cdot \mu \cdot \ln\left(\frac{e \cdot n}{n-i}\right) + \frac{e \cdot \mu \cdot n}{n} + \frac{e \cdot \mu \cdot n}{\mu \cdot (n-i)}$$

ausrechnen. Auf die hier skizzierten Ideen werden wir noch in Kapitel 11 zu sprechen kommen, wenn wir den Algorithmus (μ+1)-EA (mit Älterwerden) untersuchen.

Hier können wir diese Ideen leider nicht benutzen, um die obere Schranke aus Theorem 4.1 zu verkleinern, da die Selektion zur Ersetzung nicht garantiert, dass ein Individuum aus einer vollsten Box entfernt wird. Wenn die Population beispielsweise $\mu - 1$ verschiedene Individuen $x_1, \ldots, x_{\mu-1}$ mit $\text{OneMax}(x_i) = 0$ und 2 identische Individuen $x_\mu, x_{\mu+1}$ mit $\text{OneMax}(x_i) = 1$ enthält, dann würde entweder x_μ oder $x_{\mu+1}$ gelöscht, wenn wir die SPEA2-artige Dichteberechnung unterstellen.

4.4. Ein Problem, für das RADEMO ineffizient und SEMO effizient ist

Wir definieren hier eine weitere Beispielfunktion, die verdeutlicht, in welchen Situationen der Algorithmus RADEMO nicht in der Lage ist, eine angemessene Approximationsgüte zu erreichen. Wir begrenzen unsere Betrachtungen hier auf die Variante, deren Dichteberechnung dem Algorithmus SPEA2 entnommen ist. Um die Funktion definieren zu können, greifen wir auf die bekannte Beispielfunktion LeadingOnes zurück, die folgendermaßen definiert ist.

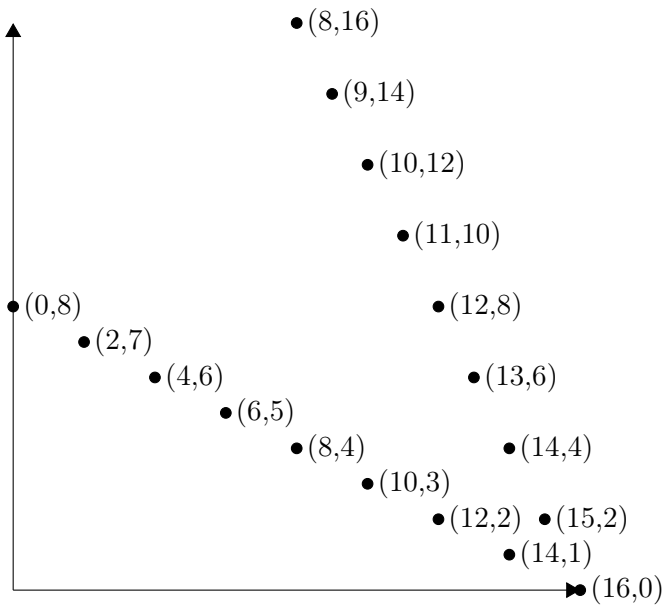


Abbildung 4.3.: Zielraum der Beispielfunktion g für $n = 16$ und $\alpha = n = 16$.

Definition 4.2. *Sei $n \in \mathbb{N}^+$. Dann definieren wir die verbreitete Beispielfunktion* $\text{LeadingOnes}\colon \mathbb{B}^n \to \mathbb{R}$ *durch*

$$\text{LeadingOnes}(x = (x_1, \ldots, x_n)) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

Wir beschreiben nun die Beispielfunktion g_α für $\alpha \in \mathbb{R}^+$. Sie bildet die Suchpunkte aus $\mathbb{B}^n$ auf $n+1$ verschiedene Zielpunkte aus $[0, \alpha] \subseteq \mathbb{R}^2$ ab. Auf welchen Zielpunkt ein Suchpunkt $x \in \mathbb{B}^n$ abgebildet wird, hängt einzig von $\text{LeadingOnes}(x) \in \{0, \ldots, n\}$ ab. Die Suchpunkte $x \in \mathbb{B}^n$ mit $\text{LeadingOnes}(x) < n/2$ werden auf eine lineare Front abgebildet. Diese Front wird von der linearen Paretofront dominiert, auf die die Suchpunkte $x \in \mathbb{B}^n$ mit $\text{LeadingOnes}(x) \geq n/2$ abgebildet werden. Die Suchpunkte $x \in \mathbb{B}^n$ mit $\text{LeadingOnes}(x) = n/2$ werden auf einen Zielpunkt abgebildet, der das Verbindungsglied zwischen den beiden genannten Fronten darstellt (siehe Abbildung 4.3). Wenn wir die Anzahl der führenden 1-Bits eines Suchpunktes erhöhen, durchlaufen wir im Zielraum die erste Front von links nach rechts und die zweite Front von rechts nach links. Wir definieren g_α nun formal.

Definition 4.3. *Seien $\alpha \in \mathbb{R}^+$ und $n \in \mathbb{N}^+$. Dann definieren wir die Beispielfunktion* $g_\alpha = (g_{\alpha,1}, g_{\alpha,2})\colon \mathbb{B}^n \to \mathbb{R}^2$ *durch*

$$g_{\alpha,1}(x) = \begin{cases} 2 \cdot \text{LeadingOnes}(x)/n \cdot \alpha & \textit{falls } \text{LeadingOnes}(x) < n/2 \\ (3/2 - \text{LeadingOnes}(x)/n) \cdot \alpha & \textit{falls } \text{LeadingOnes}(x) \geq n/2 \end{cases}$$

und

$$g_{\alpha,2}(x) = \begin{cases} (1/2 - \text{LeadingOnes}(x)/n) \cdot \alpha & \textit{falls } \text{LeadingOnes}(x) < n/2 \\ (2 \cdot \text{LeadingOnes}(x)/n - 1) \cdot \alpha & \textit{falls } \text{LeadingOnes}(x) \geq n/2. \end{cases}$$

Die Beispielfunktion g_α wird in Abbildung 4.3 für $n = 16$ und $\alpha = 16$ dargestellt. Wir werden die Funktion im Anschluss einfach g nennen.

Das folgende Theorem zeigt, dass der Algorithmus RADEMO nicht in der Lage ist, eine angemessene Approximation der Paretofront zu berechnen, wenn die Population nicht zu groß ist. Die wesentliche Beweisidee lässt sich wie folgt zusammenfassen. Die Wahrscheinlichkeit ist überwältigend, dass alle Individuen in der initialen Population eine relativ geringe Anzahl von führenden 1-Bits aufweisen. Im Anschluss verteilen sich die Individuen in der Population über die suboptimale Front. Während dieser Phase sorgt der Diversitätsmechanismus für einen gewissen Mindestabstand zwischen den einzelnen Individuen, den wir im Zielraum messen. Diese gleichmäßige Verteilung der Individuen über die suboptimale Front verhindert letztlich, dass Individuen, die auf die in Abbildung 4.3 am weitesten rechts unten befindlichen Zielpunkte der Paretofront abgebildet werden, akzeptiert werden. Diese Individuen weisen genau wie alle anderen Individuen den Rang 0 auf. Sie werden jedoch unmittelbar nach ihrer Erzeugung aus der Population entfernt, da sie eine geringere Dichte als die Individuen aufweisen, die sich über die suboptimale Front verteilt haben. Indem wir die skizzierten Ideen präzisieren und formalisieren, erhalten wir das folgende Theorem.

Theorem 4.4. *Wir wenden den Algorithmus RADEMO (nur SPEA2-artige Dichteberechnung) mit Populationsgröße $\mu = \mathrm{o}(n^{1/4})$ auf die Fitnessfunktion g mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) \geq \alpha/2$ für die ersten $2^{\Omega(\sqrt{n})}$ Populationen P mit Wahrscheinlichkeit $1 - 2^{-\Omega(\sqrt{n})}$.*

Beweis. Betrachte die Potenzialfunktion $\text{pot}(P) := \max\{\text{LeadingOnes}(x) \mid x \in P\}$. Wenn ein Suchpunkt $x \in \mathbb{B}^n$ rein zufällig gewählt wird, dann gilt $\text{LeadingOnes}(x) \geq n/6$ mit Wahrscheinlichkeit $2^{-n/6}$, da die führenden $n/6$ Bits auf 1 gesetzt werden müssen. Mithilfe der booleschen Ungleichung folgt, dass das Potenzial der anfänglichen Population mit Wahrscheinlichkeit mindestens $1 - \mu \cdot 2^{-n/6} = 1 - 2^{-\Omega(n)}$ kleiner als $n/6$ ist.

Sei m ein beliebiger Potenzialwert und $t \geq 0$ ein beliebiger Zeitpunkt. Betrachte eine beliebige Folge von Populationen P_i mit $0 \leq i \leq t$, sodass $\text{pot}(P_i) \leq m$ für alle P_i gilt. Sei $x = (x_1, \ldots, x_n) \in P_i$, $0 \leq i \leq t$, ein beliebiges Individuum. Dann beeinflussen die Bits $x_{m+2}, \ldots, x_n$ die Fitness $g(x)$ von x nicht. Wir beweisen, dass die genannten Bits rein zufällig verteilt sind. Das Individuum ist aus einem der rein zufällig gezogenen initialen Individuen entstanden. Ein Bit des initialen Individuums ist durch eine Bernoulli-verteilte Zufallsvariable X mit $\mathrm{W}(X = 0) = 1/2$ und $\mathrm{W}(X = 1) = 1/2$ gegeben. Anschließend können Standard-Bit-Mutationen das Bit gekippt haben. Nach einer Standard-Bit-Mutation ist das Bit durch eine Bernoulli-verteilte Zufallsvariable Y mit $\mathrm{W}(Y = b \mid X = 1 - b) = 1/n$ und

$\mathrm{W}(Y = b \mid X = b) = 1 - 1/n$ für alle $b \in \mathbb{B}$ gegeben. Eine Anwendung des Satzes von der totalen Wahrscheinlichkeit (siehe Lemma A.3) liefert

$$\begin{aligned} & \mathrm{W}(Y = b) \\ = \ & \mathrm{W}(Y = b \mid X = 1 - b) \cdot \mathrm{W}(X = 1 - b) + \mathrm{W}(Y = b \mid X = b) \cdot \mathrm{W}(X = b) \\ = \ & (1/n) \cdot (1/2) + (1 - 1/n) \cdot (1/2) \\ = \ & 1/2 \end{aligned}$$

für alle $b \in \mathbb{B}$. Wir schließen durch Induktion, dass die Bits $x_{m+2}, \ldots, x_n$ rein zufällig verteilt sind.

Bezeichne $m = \mathrm{pot}(P)$ das aktuelle Potenzial. Wir zeigen nun, dass es unwahrscheinlich ist, dass sich das Potenzial in einem Mutationsschritt um mehr als $\sqrt{n}$ erhöht. Wenn ein Individuum $x = (x_1, \ldots, x_n)$ zur Mutation ausgewählt wird, dann wissen wir, dass die Bits $x_{m+2}, \ldots, x_n$ rein zufällig verteilt sind. Folglich beträgt die Wahrscheinlichkeit, dass sich das Potenzial in einem Schritt um mehr als $\sqrt{n}$ erhöht, höchstens $2^{-\sqrt{n}}$. Wir betrachten eine Phase der Länge $2^{\sqrt{n}/2}$. Mit Wahrscheinlichkeit mindestens $1 - 2^{\sqrt{n}/2} \cdot 2^{-\sqrt{n}} = 1 - 2^{-\Omega(\sqrt{n})}$ enthält die Phase keinen Mutationsschritt, in dem das Potenzial um mehr als $\sqrt{n}$ erhöht wird.

Wir warten zunächst bis das erste Individuum x mit $\mathrm{LeadingOnes}(x) > n/3 - \sqrt{n}$ in der Population enthalten ist. Dann gilt $\mathrm{LeadingOnes}(x) \leq n/3$. Anschließend benötigt der Algorithmus noch mindestens $\sqrt{n}/6$ Potenzialerhöhungen bis das erste Individuum x mit $n/2 - \sqrt{n} < \mathrm{LeadingOnes}(x) \leq n/2$ in der Population enthalten ist. Die Wahrscheinlichkeit, das Potenzial zu erhöhen, beträgt höchstens $1/n$, da das am weitesten links befindliche 0-Bit gekippt werden muss. Wir betrachten eine Phase der Länge $n^{3/2}/9$ und schließen mithilfe einer Chernoff-Ungleichung, dass das Potenzial in der betrachteten Phase weniger als $(\sqrt{n}/6)$-mal mit Wahrscheinlichkeit mindestens $1 - e^{-\sqrt{n}/72} = 1 - 2^{-\Omega(\sqrt{n})}$ erhöht wird, da wir $\sqrt{n}/6$ durch $(1 + 1/2) \cdot n^{3/2}/9$ ausdrücken können. Also dauert es mit überwältigender Wahrscheinlichkeit mindestens $n^{3/2}/9$ Schritte bis das gewünschte Individuum erzeugt wird.

Wir benutzen den Begriff „Bit-Abstand“ im Folgenden als Abkürzung für den Term $|\,\mathrm{LeadingOnes}(y) - \mathrm{LeadingOnes}(x)|$, wobei $x \in \mathbb{B}^n$ und $y \in \mathbb{B}^n$ beliebige Individuen sind. Zu Anfang der im letzten Absatz betrachteten Phase beträgt der maximale Bit-Abstand in der Population mindestens $(n/3 - \sqrt{n} + 1) - (n/6 - 1) \geq n/12 =: d_{\max}$. Wir zeigen, dass $\mu - 2$ Ereignisse ausreichen, um den minimalen Bit-Abstand in der Population auf $d_{\max}/(2 \cdot (\mu - 1))$ zu erhöhen. Wir nennen zwei Individuen aus der aktuellen Population „benachbart“, wenn es in der Population kein Individuum gibt, dass sich im Zielraum zwischen den beiden Individuen befindet. Es gibt stets zwei benachbarte Individuen mit Bit-Abstand mindestens $d_{\max}/(\mu - 1)$, da der durchschnittliche Abstand zwischen benachbarten Individuen $d_{\max}/(\mu - 1)$ beträgt. Wenn $(\mu - 2)$-mal ein Individuum erzeugt wird, dass sich genau in der Mitte zwischen zwei benachbarten Individuen mit dem genannten Bit-Abstand befindet, dann beträgt der minimale Bit-Abstand in der Population mindestens $d_{\max}/(2 \cdot (\mu - 1))$. Um ein solches Individuum zu erzeugen, reicht es aus, ein Individuum x mit $\mathrm{LeadingOnes}(x) = \mathrm{pot}(P)$ zur Mutation auszuwählen und genau ein bestimmtes Bit zu kippen. Die Wahrscheinlichkeit

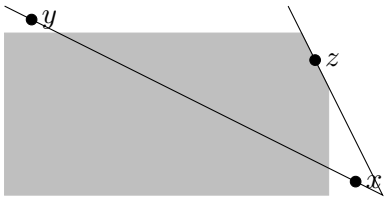


Abbildung 4.4.: Situation im Beweis von Theorem 4.4.

für ein solches Ereignis beträgt mindestens $(1/\mu) \cdot (1/(e \cdot n))$. Im Erwartungswert treten die $\mu - 2$ genannten Ereignisse nach höchstens $e \cdot \mu^2 \cdot n$ Schritten ein. Wir schließen mithilfe der Markoff-Ungleichung, dass die Ereignisse in einer Phase der Länge $2 \cdot e \cdot \mu^2 \cdot n$ mit Wahrscheinlichkeit mindestens $1/2$ eintreten. Wir weisen darauf hin, dass in der hier betrachteten Phase alle Individuen den Rang 0 aufweisen und die Entscheidung, ob ein Individuum gelöscht wird, einzig von seiner Dichte abhängt. Wenn wir die Phase der Länge $n^{3/2}/9$ in $\sqrt{n}/(18 \cdot e \cdot \mu^2)$ Subphasen der Länge $2 \cdot e \cdot \mu^2 \cdot n$ einteilen, dann können wir insgesamt schließen, dass der minimale Bit-Abstand in der Population, wenn die betrachtete Phase beendet wird, mit Wahrscheinlichkeit mindestens $1 - 2^{-\Omega(\sqrt{n})}$ mindestens $d_{\min} := d_{\max}/(2 \cdot (\mu - 1)) > 6 \cdot \sqrt{n}$ beträgt.

Wir betrachten nun die Situation nachdem das erste Individuum x mit $n/2 - \sqrt{n} < \text{LeadingOnes}(x) \leq n/2$ in die Population aufgenommen wurde und der Bit-Abstand zwischen zwei benachbarten Individuen mindestens $d_{\min}$ beträgt. Für das Individuum y mit dem geringsten Bit-Abstand zu x gilt also $\text{LeadingOnes}(y) \leq \text{LeadingOnes}(x) - d_{\min}$. Betrachte ein Individuum z mit $n/2 + 2 \cdot \sqrt{n} \leq \text{LeadingOnes}(z) < n/2 + 3 \cdot \sqrt{n}$. Die Situation wird in Abbildung 4.4 veranschaulicht. Dann gelten

$$\begin{aligned} g_1(z) &= (3/2 - \text{LeadingOnes}(x)/n) \cdot \alpha \\ &\leq (3/2 - (n/2 + 2 \cdot \sqrt{n})/n) \cdot \alpha \\ &= (1 - 2/\sqrt{n}) \cdot \alpha \\ &< 2 \cdot \text{LeadingOnes}(x)/n \cdot \alpha = g_1(x) \end{aligned}$$

und

$$\begin{aligned} g_2(z) &= (2 \cdot \text{LeadingOnes}(x)/n - 1) \cdot \alpha \\ &< (2 \cdot (n/2 + 3 \cdot \sqrt{n})/n - 1) \cdot \alpha \\ &= 6/\sqrt{n} \cdot \alpha \\ &\leq (1/2 - \text{LeadingOnes}(x)/n) \cdot \alpha = g_2(y). \end{aligned}$$

Wenn das Individuum z erzeugt wird, dann dominiert es folglich kein Individuum in der Population. Wir zeigen nun, dass das Individuum z eine geringere Dichte als alle anderen Individuen in der Population aufweist und demzufolge sofort wieder entfernt wird. Insgesamt folgt dann das Theorem, da die Approximationsgüte für den Zielpunkt $(\alpha/2, \alpha)$ mindestens $\alpha/2$ beträgt.

Wir betrachten nur die SPEA2-artige Dichteberechnung. Wir sehen uns die Abstände zwischen x, y und z an. Dann gilt

$$\begin{aligned} d(z,x) &= ((g_1(x) - g_1(z))^2 + (g_2(z) - g_2(x))^2)^{1/2} \\ &< (((1 - (1 - 3/\sqrt{n})) \cdot \alpha)^2 + ((6/\sqrt{n} - 0) \cdot \alpha)^2)^{1/2} \\ &= \sqrt{45}/\sqrt{n} \cdot \alpha. \end{aligned}$$

Auf analoge Weise zeigt man, dass $d(z,y) < d(x,y)$ gilt. Offensichtlich gilt $d(x',x'') \geq (((2 \cdot d_{\min}/n) \cdot \alpha)^2 + ((d_{\min}/n) \cdot \alpha)^2)^{1/2} = (\sqrt{180}/\sqrt{n}) \cdot \alpha$ für alle $x', x'' \in P \setminus \{x,z\}$. Insgesamt haben wir gezeigt, dass das Individuum z die kleinste Dichte aufweist und folglich aus der Population entfernt wird. □

Wir weisen an dieser Stelle darauf hin, dass die Approximationsgüte $\alpha/2$, die mit überwältigender Wahrscheinlichkeit in polynomiell vielen Generationen nicht übertroffen wird, weit von der optimalen Approximationsgüte entfernt ist, die mit einer Population, die aus $\mu = \mathrm{o}(n^{1/4})$ Individuen besteht, erreichbar ist. Wenn man die μ Individuen äquidistant auf der Paretofront von g verteilt, dann erreicht man offensichtlich die Approximationsgüte $\Theta(\alpha/\mu)$. Die erreichbare Approximationsgüte kann hier also durch $\omega(\alpha/n^{1/4})$ charakterisiert werden.

Wir beweisen abschließend, dass der Algorithmus SEMO im Erwartungswert in polynomiell vielen Generationen die komplette Paretofront findet. Die Idee, die dem Beweis unterliegt, lässt sich wie folgt zusammenfassen. Wir gehen davon aus, dass der Algorithmus seine Suche mit einem Individuum auf der suboptimalen Front beginnt. Dann breitet sich die Population über die suboptimale Front aus bis schließlich das erste Individuum auf der Paretofront gefunden wird. Anschließend breitet sich die Population über die Paretofront aus, wobei letztlich die Individuen auf der suboptimalen Front nach und nach verdrängt werden.

Theorem 4.5. *Wir wenden den Algorithmus SEMO auf die Fitnessfunktion g mit $\alpha \in \mathbb{R}^+$ an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) = 0$ im Erwartungswert nach $\mathcal{O}(n^3)$ Generationen.*

Beweis. Wir gliedern einen Lauf des Algorithmus SEMO in zwei Phasen. Die erste Phase endet, wenn ein Suchpunkt x mit $\mathrm{LeadingOnes}(x) = n$ in der Population enthalten ist. Die zweite Phase endet, wenn für alle $i \in \{n/2, \ldots, n\}$ ein Suchpunkte x mit $\mathrm{LeadingOnes}(x) = i$ in der Population enthalten ist.

Betrachte die Potenzialfunktion $\mathrm{pot}(P) := \max\{\mathrm{LeadingOnes}(x) \mid x \in P\}$. Die Größe der Population beträgt höchstens $n/2 + 1$. Daher beträgt die Wahrscheinlichkeit, ein Individuum $x \in P$ mit $\mathrm{LeadingOnes}(x) = \mathrm{pot}(P)$ zur Mutation auszuwählen, mindestens $1/|P| \geq 2/(n+2)$. Die Wahrscheinlichkeit, das Potenzial zu erhöhen, ist demzufolge durch $(2/(n+2)) \cdot (1/(e \cdot n))$ nach unten beschränkt. Die erste Phase wird daher im Erwartungswert nach höchstens

$$n \cdot \frac{n+2}{2} \cdot e \cdot n = \frac{e}{2} \cdot n^3 + e \cdot n^2 = \mathcal{O}(n^3)$$

Generationen beendet.

Die Wahrscheinlichkeit, einen Suchpunkt x' mit LeadingOnes$(x') = i$ zu erzeugen, ist anschließend durch $(2/(n+2)) \cdot (1/(e \cdot n))$ nach unten beschränkt. Die zweite Phase wird daher im Erwartungswert nach höchstens

$$\frac{n}{2} \cdot \frac{n+2}{2} \cdot e \cdot n = \frac{e}{4} \cdot n^3 + \frac{e}{2} \cdot n^2 = \mathcal{O}(n^3)$$

Generationen beendet.

Insgesamt folgt das Theorem. □

4.5. Fazit

Wir haben in diesem Kapitel einen evolutionären Algorithmus untersucht, der im Gegensatz zu den zuvor betrachteten Algorithmen SEMO und DEMO eine Population, die in jeder Generation genau μ Individuen enthält, iterativ verbessert. Der Algorithmus lehnt sich an die bekannten Algorithmen NSGA2 und SPEA2 an. Das bedeutet, dass die Selektion zur Ersetzung in zwei Phasen abläuft. In der ersten Phase wird der Rang der Individuen bestimmt, wobei die Individuen mit dem größten Rang potenziell aus der Population entfernt werden. In der zweiten Phase werden die Abstände der Individuen berücksichtigt, um jedem Individuum eine Dichte zuzuordnen. Anschließend wird ein Individuum gelöscht, das den größten Rang aufweist und von diesen Individuen die kleinste Dichte besitzt. Wir haben den Algorithmus RADEMO genannt, da die Entscheidung, welches Individuum aus der Population entfernt wird, auf den Rängen und den Dichten der Individuen basiert. Er unterscheidet sich von den weiter oben genannten Algorithmen NSGA2 und SPEA2 einzig darin, dass er genau wie die Algorithmen SEMO und DEMO in jeder Generation genau ein neues Individuum mithilfe eines Mutationsoperators erzeugt und gänzlich auf einen Rekombinationsoperator verzichtet. Wir haben für die aus Abschnitt 3.3 bekannte Beispielfunktion, die eine exponentiell große Paretofront besitzt, gezeigt, dass die vorgestellten diversitätserhaltenden Maßnahmen ähnlich erfolgreich sind wie das Konzept, das dem Algorithmus DEMO zugrunde liegt. Um eine gute Approximationsgüte zu erreichen, muss die Population natürlich hinreichend viele Individuen enthalten. Auf der anderen Seite haben wir eine natürliche Beispielfunktion vorgestellt, die die Schwachpunkte des Algorithmus RADEMO deutlich herausstellt. Insgesamt liefert dieses Kapitel neue Einsichten, in welchen Situationen die grundlegenden Algorithmen SEMO, DEMO und RADEMO erfolgreich eingesetzt werden können und in welchen nicht.

Wir werden uns in Kapitel 5 der sogenannten Selektion zur Mutation zuwenden, nachdem wir in Kapitel 3 und in Kapitel 4 den Einfluss der Selektion zur Ersetzung auf die Optimierungszeit untersucht haben.

5. Vergleich der Algorithmen SEMO und FEMO

5.1. Einleitung

Nachdem wir uns in den letzten beiden Kapiteln mit der Selektion zur Ersetzung beschäftigt haben, werden wir nun den Einfluss der Selektion zur Mutation beleuchten. Wir untersuchen die Optimierungszeiten von zwei grundlegenden evolutionären Algorithmen. Den bekannten Algorithmus SEMO vergleichen wir mit dem von Laumanns et al. [2004] vorgeschlagenen Algorithmus FEMO. Der Algorithmus FEMO zeichnet sich durch ein *faires* Selektionsschema aus. Während der Algorithmus SEMO das als nächstes zu mutierende Individuum rein zufällig aus der aktuellen Population zieht, trifft der Algorithmus FEMO hier eine differenziertere Wahl. Er speichert für jedes Individuum in der aktuellen Population, wie oft dieses bereits zur Mutation gewählt wurde, und zieht das als nächstes zu mutierende Individuum rein zufällig aus den Individuen, die bisher die geringste Chance bekommen haben, sich durch eine Mutation zu verbessern. Wir betrachten erneut zwei Beispielfunktionen und das Optimierungsziel, eine feste additive Approximationsgüte zu erreichen.

Laumanns et al. [2004] haben die beiden Algorithmen SEMO und FEMO bereits einem ersten Vergleich unterzogen. In [Laumanns et al., 2004] wird eine Beispielfunktion namens „Leading Ones, Trailing Zeroes (LOTZ)“ vorgestellt und gezeigt, dass der Algorithmus SEMO die Paretofront im Erwartungswert nach $\Theta(n^3)$ Generationen findet, während der Algorithmus FEMO die Paretofront im Erwartungswert nach $\mathcal{O}(n^2 \cdot \log n)$ Generationen findet. Sie haben damit gezeigt, dass die faire Selektion die Optimierungszeit um den Faktor $\Omega((\log n)/n)$ beschleunigen kann. Wir beweisen in diesem Kapitel, dass der Unterschied zwischen den Optimierungszeiten der beiden Algorithmen weitaus drastischer ausfallen kann.

5.2. Der Algorithmus FEMO

Stellen wir uns beispielsweise die einfache, lineare Paretofront vor, die durch die in Abschnitt 3.4 definierte Beispielfunktion gegeben ist (siehe Definition 3.7). Wenn wir diese Funktion mithilfe des Algorithmus SEMO optimieren, dann besteht die Population zu Beginn mit überwältigender Wahrscheinlichkeit aus einem Individuum, das in den mittleren Bereich der Paretofront abgebildet wird. Wir beobachten, dass sich die Population im Laufe der Optimierung nach außen ausbreitet. Wir beobachten weiter, dass nach relativ kurzer Zeit der mittlere Bereich der Paretofront vollständig abgedeckt ist, während die äußeren Bereiche noch fehlen. In dieser Situation ist es

Algorithmus 5 Fair Evolutionary Multiobjective Optimizer (FEMO)

Eingabe: Fitnessfunktion $f\colon S \to \mathbb{R}^d$ mit endlichem Suchraum S
Ausgabe: Suchpunkte $P \subseteq S$

```
1: t ← 0
2: wähle ein Individuum x_t ∈ S rein zufällig
3: c(x_t) ← 0                                  ▷ initialisiere Nachkommenzähler von x_t
4: P_t ← {x_t}
5: repeat
6:     t ← t + 1
7:     wähle ein Individuum x ∈ {y ∈ P_{t−1} | ∀z ∈ P_{t−1}: c(z) ≥ c(y)} rein zufällig
8:     c(x) ← c(x) + 1                         ▷ erhöhe Nachkommenzähler von x
9:     erzeuge x_t aus x durch Mutation
10:    if ∄x ∈ P_{t−1}: f(x) ≻ f(x_t) then
11:        if x_t ∉ P_{t−1} then
12:            c(x_t) ← 0                      ▷ initialisiere Nachkommenzähler von x_t
13:        end if
14:        P_t ← (P_{t−1} \ {x ∈ P_{t−1} | f(x_t) ⪰ f(x)}) ∪ {x_t}
15:    else
16:        P_t ← P_{t−1}
17:    end if
18: until Abbruchkriterium erfüllt
19: return P_t
```

vorteilhaft, die Individuen, die sich am Rand des abgedeckten Bereiches befinden, zur Mutation auszuwählen, um möglichst schnell die fehlenden Punkte der Paretofront zu entdecken. Da der Algorithmus SEMO das als nächstes zu mutierende Individuum rein zufällig wählt, wird der Optimierungsprozess durch die Wahl eines Individuums aus dem abgedeckten mittleren Bereich verlangsamt. Insgesamt benötigt der Algorithmus SEMO daher im Erwartungswert $\Omega(n^{2-c} \cdot \log n)$ Generationen, um die gesamte Paretofront zu finden, wobei $c > 0$ eine beliebige Konstante ist (siehe Theorem 3.10). Wenn wir die genannte Beispielfunktion so modifizieren, dass der Zielpunkt, auf den ein Suchpunkt x abgebildet wird, nicht mehr von OneMax(x) sondern von LeadingOnes(x) abhängt, dann kann man auf ähnliche Weise wie im Beweis von Theorem 3.10 zeigen, dass der Algorithmus SEMO die Paretofront im Erwartungswert erst nach $\Omega(n^{3-c})$ Generationen findet.

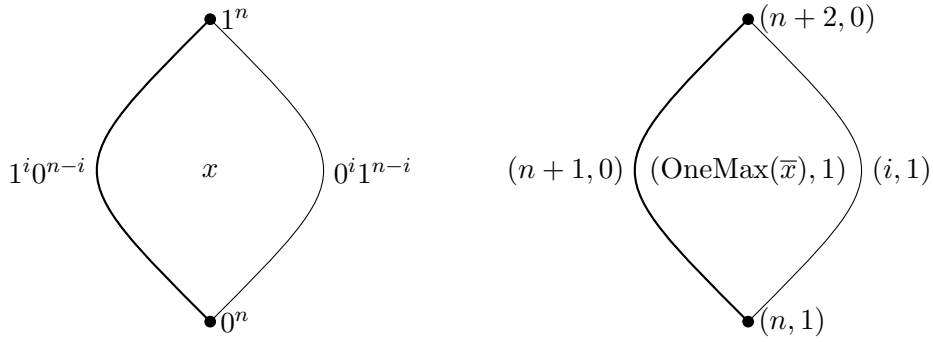
Wir werden nun eine weitere Möglichkeit vorstellen, um die Selektion zur Mutation zu realisieren. Die Idee besteht darin, jedem Individuum, das sich in der aktuellen Population befindet, die gleiche Chance einzuräumen, sich durch eine erfolgreiche Mutation zu verbessern. Wir realisieren diese Idee, indem wir jedes in der aktuellen Population befindliche Individuum mit einem sogenannten *Nachkommenzähler* versehen, der zählt, wie oft das Individuum bislang zur Mutation ausgewählt wurde. Der Nachkommenzähler wird mit 0 initialisiert, wenn ein Individuum der Population hinzugefügt wird. Das als nächstes zu mutierende Individuum wird nun rein zufällig aus

den Individuen gezogen, die über den kleinsten Nachkommenzähler verfügen. Wenn ein Individuum zur Mutation ausgewählt wird, dann wird der Nachkommenzähler entsprechend inkrementiert. Der Algorithmus FEMO arbeitet bis auf den gerade beschriebenen Mechanismus genau wie der Algorithmus SEMO (siehe Abschnitt 2.4). Algorithmus 5 fasst die Beschreibung des neuen Algorithmus zusammen.

Wir kommen auf das eingangs erwähnte Beispiel zurück. Man kann zeigen, dass der Algorithmus FEMO im Erwartungswert mit $\mathcal{O}(n^2 \cdot \log n)$ Generationen auskommt, um die Paretofront der in Abschnitt 3.4 definierten Beispielfunktion zu finden. Die gleiche Schranke gilt auch für die im vorletzten Absatz angesprochene Variante, die den Zielpunkt, auf den ein Suchpunkt x abgebildet wird, von LeadingOnes(x) abhängig macht. Ein Individuum mit i 1-Bits bzw. führenden 1-Bits muss im Erwartungswert höchstens $(e \cdot n)$-mal zur Mutation ausgewählt werden, um ein Individuum mit mehr oder weniger als i 1-Bits bzw. führenden 1-Bits zu erzeugen. Daher muss der Algorithmus FEMO für jeden Zielpunkt ein zugehöriges Individuum höchstens $\mathcal{O}(n \cdot \log n)$-mal zur Mutation auswählen, um mit überwältigender Wahrscheinlichkeit ein Individuum, das auf einen fehlenden Zielpunkt abgebildet wird, zu erschaffen. Da der Selektionsmechanismus des Algorithmus FEMO die Individuen in der Population gleichmäßig zur Mutation auswählt, folgt die genannte Schranke für die Optimierungszeit. Wir werden in den nächsten beiden Abschnitten Beispielfunktionen vorstellen und beweisen, dass sich die Optimierungszeiten der Algorithmen SEMO und FEMO wesentlich stärker unterscheiden können.

Der beschriebene Algorithmus FEMO verwaltet also für alle Individuen in der aktuellen Population einen eigenen Nachkommenzähler. Wenn wir annehmen, dass die aktuelle Population ein Individuum x enthält und dass ein von x verschiedenes Individuum y mit $y \equiv x$ erzeugt wird, dann passiert das Folgende. Das neue Individuum y wird akzeptiert und ersetzt das alte Individuum x. Außerdem wird der Zähler von y auf 0 gesetzt, da $y \neq x$ gilt. An dieser Stelle kann man auch auf die Idee kommen, die Nachkommenzähler nicht mit den einzelnen Suchpunkten in der Population sondern mit den entsprechenden Zielpunkten zu verknüpfen. Dann würde der Nachkommenzähler in der angesprochenen Situation nicht auf 0 gesetzt werden. Eine solche Variante des Algorithmus FEMO wird in [Friedrich et al., 2010] unter anderem mit der hier betrachteten Variante verglichen. Darin wird gezeigt, dass der Unterschied der Optimierungszeiten der beiden Varianten exponentiell groß sein kann. Wir werden an dieser Stelle auf die Betrachtung der zweiten Variante verzichten und uns auf den Vergleich mit dem Algorithmus SEMO konzentrieren.

Wir erwähnen an dieser Stelle, dass sich der Algorithmus FEMO nicht von den Algorithmen SEMO, DEMO und (1+1)-EA unterscheidet, wenn eine Fitnessfunktion $f\colon \mathbb{B}^n \to \mathbb{R}$ vorliegt. In diesem Fall enthalten alle Populationen genau ein Individuum und die Selektion eines Individuums zur Mutation degeneriert zu einer trivialen Aufgabe.



(a) Suchraum $\mathbb{B}^n$.

(b) Zielraum $f(\mathbb{B}^n)$.

Abbildung 5.1.: Suchraum und Zielraum der Beispielfunktion f.

5.3. Ein Problem, für das SEMO ineffizient und FEMO effizient ist

Wir stellen in diesem Abschnitt eine Beispielfunktion vor und demonstrieren, wie eine faire Auswahl des zu mutierenden Individuums den Optimierungsprozess signifikant beschleunigen kann. Die Beispielfunktion wurde erstmalig von Friedrich et al. [2007] benutzt. Friedrich et al. [2007] diskutieren die prinzipielle Frage, ob Plateaus in multikriteriellen Optimierungsproblemen schwieriger als in monokriteriellen Optimierungsproblemen sein können. Sie belegen mit der Beispielfunktion, dass die Existenz eines Plateaus die Optimierung eines multikriteriellen Optimierungsproblems erschweren kann. In [Friedrich et al., 2007] wird unter anderem gezeigt, dass der Algorithmus SEMO die Paretofront der folgenden Beispielfunktion in $\exp(\Omega(n^{1/24}))$ Generationen mit Wahrscheinlichkeit $1 - \exp(-\Omega(n^{1/24}))$ *nicht* findet.

Definition 5.1. *Sei $n \in \mathbb{N}^+$. Die Beispielfunktion $f = (f_1, f_2)\colon \mathbb{B}^n \to \mathbb{R}^2$ definieren wir dann durch*

$$f_1(x) = \begin{cases} \mathrm{OneMax}(\overline{x}) & \textit{falls } x \notin \{1^i0^{n-i} \mid 1 \leq i \leq n\} \\ n+1 & \textit{falls } x \in \{1^i0^{n-i} \mid 1 \leq i < n\} \\ n+2 & \textit{falls } x = 1^n \end{cases}$$

und

$$f_2(x) = \begin{cases} 1 & \textit{falls } x \notin \{1^i0^{n-i} \mid 1 \leq i \leq n\} \\ 0 & \textit{falls } x \in \{1^i0^{n-i} \mid 1 \leq i < n\} \\ 0 & \textit{falls } x = 1^n. \end{cases}$$

Die Beispielfunktion wird in Abbildung 5.1 veranschaulicht. Sie besitzt die beiden Pareto-optimalen Suchpunkte 0^n und 1^n, die durch ein Plateau miteinander verbunden sind. Wir benutzen in diesem Abschnitt für die Menge, die das Plateau und den Pareto-optimalen Suchpunkt 1^n enthält, die Abkürzung $SP := \{1^i0^{n-i} \mid 1 \leq i \leq n\}$.

Das nächste Theorem zeigt, dass der Algorithmus SEMO nicht in der Lage ist, die Beispielfunktion f effizient zu optimieren. Beachte, dass das Theorem die aus [Friedrich et al., 2007] bekannte untere Schranke für die Optimierungszeit leicht verbessert. Dem Beweis des Theorems liegt die folgende Idee zugrunde. Der Algorithmus SEMO gelangt mit überwältigender Wahrscheinlichkeit in eine Situation, in der seine Population aus dem Pareto-optimalen Suchpunkt 0^n und einem weiteren Suchpunkt auf dem Pfad SP besteht. Wenn der Suchpunkt auf dem Pfad diesen zufällig explorieren könnte, dann würde der zweite Pareto-optimale Suchpunkt 1^n auf diese Weise im Erwartungswert nach $\mathcal{O}(n^3)$ Schritten gefunden werden (siehe Beweis von Theorem 5.3). Das Problem besteht darin, dass auch der Suchpunkt 0^n im Erwartungswert alle 2 Schritte zur Mutation ausgewählt wird und mit einer größeren Wahrscheinlichkeit als $1/(e{\cdot}n)$ den Suchpunkt $10^{n-1} \in SP$ erschafft. Dieser wird akzeptiert und setzt damit die zufällige Irrfahrt auf dem Plateau zurück. Dieses Verhalten führt letztlich dazu, dass es mit überwältigender Wahrscheinlichkeit exponentiell lange dauert, bis die komplette Paretofront gefunden wird.

Theorem 5.2. *Wir wenden den Algorithmus SEMO auf die Fitnessfunktion f an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) > 0$ für die ersten $2^{\Omega(\sqrt{n})}$ Populationen P mit Wahrscheinlichkeit $1 - 2^{-\Omega(\sqrt{n})}$.*

Beweis. Betrachte die Potenzialfunktion $\mathrm{pot}(P) := \max\{\mathrm{OneMax}(x) \mid x \in P\}$. Da das erste Individuum rein zufällig gewählt wird, schließen wir mithilfe einer Chernoff-Ungleichung, dass anfangs $\mathrm{pot}(P) \leq (2/3) \cdot n$ mit Wahrscheinlichkeit $1 - e^{-\Omega(n)}$ gilt. Wir betrachten nun eine Phase der Länge $\ell := c \cdot n^{3/2} \cdot \log n$, wobei c eine positive Konstante ist, die wir später festlegen werden.

Wir zeigen zunächst, dass das Individuum 0^n mit überwältigender Wahrscheinlichkeit gefunden wird. Wir weisen auf die wichtige Tatsache hin, dass die Population aus höchstens zwei Individuen besteht, da $f_2(S) = \mathbb{B}$ gilt. Wenn die Population ein Individuum $x \in P$ enthält, dann beträgt die Wahrscheinlichkeit, dass der Algorithmus ein Individuum $x' \notin SP$ mit $\mathrm{OneMax}(x') = \mathrm{OneMax}(x) - 1$ erzeugt, mindestens

$$\begin{cases} (1/2) \cdot (\mathrm{OneMax}(x) - 1) \cdot (1/n) \cdot (1 - 1/n)^{n-1} & \text{falls } \mathrm{OneMax}(x) > 1 \\ (1/2) \cdot (1/n) \cdot (1 - 1/n)^{n-1} & \text{falls } \mathrm{OneMax}(x) = 1. \end{cases}$$

Die Terme können durch $(\mathrm{OneMax}(x) - 1)/(2 \cdot e \cdot n)$ bzw. $1/(2 \cdot e \cdot n)$ nach unten abgeschätzt werden. Wenn wir ein Individuum x aus $\mathbb{B}^n \setminus SP$ betrachten, dann gilt $f(x) = (n - \mathrm{OneMax}(x), 1)$ (siehe Definition 5.1). Wenn eine Population P_t ein Individuum x aus $\mathbb{B}^n \setminus SP$ enthält, dann enthalten alle nachfolgenden Populationen $P_{t'}$, $t' \geq t$, ein Individuum y aus $\mathbb{B}^n \setminus SP$ mit $\mathrm{OneMax}(y) \leq \mathrm{OneMax}(x)$. Also findet der Algorithmus SEMO im Erwartungswert nach höchstens

$$2 \cdot e \cdot n + \sum_{i=2}^{n} \frac{2 \cdot e \cdot n}{i - 1} = 2 \cdot e \cdot n \cdot (H_{n-1} + 1) \leq 2 \cdot e \cdot n \cdot (\ln(n - 1) + 2) =: \ell'$$

Generationen das Individuum 0^n. Wir schließen mithilfe der Markoff-Ungleichung, dass das Individuum 0^n nach $2 \cdot \ell'$ Generationen mit Wahrscheinlichkeit mindestens

$1/2$ gefunden wird. Wir teilen die Phase der Länge ℓ in $\ell/(2\cdot\ell')$ Teilphasen der Länge $2\cdot\ell'$ ein und schließen, dass das Individuum 0^n mit Wahrscheinlichkeit mindestens $1-2^{-\ell/(2\cdot\ell')}=1-2^{-\,\Omega(\sqrt{n})}$ gefunden wird.

Wir beweisen nun, dass sich $\mathrm{pot}(P)$ in der betrachteten Phase mit überwältigender Wahrscheinlichkeit um weniger als $(1/6)\cdot n$ erhöht. Die Wahrscheinlichkeit, ein Individuum x' mit $x'=\mathrm{pot}(P)+i$ zu erzeugen, beträgt höchstens $(1/n)^i$, da dieses aus SP stammen muss, damit es akzeptiert wird. Also beträgt die Wahrscheinlichkeit, ein Individuum x' mit $x'\geq\mathrm{pot}(P)+i$ zu erzeugen, höchstens

$$\sum_{j=i}^{n-\mathrm{pot}(P)}\left(\frac{1}{n}\right)^j\leq\left(\frac{1}{n}\right)^i\cdot\sum_{j=0}^{\infty}\left(\frac{1}{n}\right)^j=\frac{1}{(n-1)\cdot n^{i-1}}.$$

Die Wahrscheinlichkeit, das Potenzial um mindestens 1 zu erhöhen, beträgt daher höchstens $1/(n-1)$ und die Wahrscheinlichkeit, das Potenzial um mehr als $\sqrt{n}/\log n$ zu erhöhen, beträgt daher höchstens $1/((n-1)\cdot n^{\sqrt{n}/\log n})$. Aus einer Chernoff-Ungleichung folgt, dass in der betrachteten Phase mit Wahrscheinlichkeit $1-e^{-\ell/(12\cdot(n-1))}=1-2^{-\,\Omega(\sqrt{n}\cdot\log n)}$ höchstens $((1+1/2)\cdot(\ell/(n-1)))$-mal das Potenzial erhöht wird. Aus der booleschen Ungleichung folgt, dass in der betrachteten Phase mit Wahrscheinlichkeit $1-\ell/((n-1)\cdot n^{\sqrt{n}/\log n})=1-2^{-\,\Omega(\sqrt{n})}$ das Potenzial in einer Generation höchstens um $\sqrt{n}/\log n$ erhöht wird.

Wenn wir c hinreichend klein wählen, dann sehen wir, dass nach der betrachteten Phase das Individuum 0^n gefunden wurde und das Potenzial kleiner als $(5/6)\cdot n$ ist. Konkret wählen wir c gemäß

$$\left(1+\frac{1}{2}\right)\cdot\ell\cdot\frac{1}{n-1}\cdot\frac{\sqrt{n}}{\log n}<\frac{1}{6}\cdot n\iff c<\frac{1}{9}\cdot\frac{n-1}{n}.$$

Wir betrachten nun weitere Phasen der Länge ℓ. Die Wahrscheinlichkeit, das Individuum 10^{n-1} zu erzeugen, beträgt mindestens $1/(2\cdot e\cdot n)$. Im Erwartungswert wird das Individuum 10^{n-1} daher nach $\ell''=2\cdot e\cdot n$ Generationen gefunden. Die Wahrscheinlichkeit, das Individuum in $2\cdot\ell''$ Generationen zu erzeugen, beträgt gemäß der Markoff-Ungleichung mindestens $1/2$. Wir können eine Phase in ℓ/ℓ'' Teilphasen der Länge ℓ'' einteilen und sehen, dass das Individuum innerhalb der Phase mit Wahrscheinlichkeit $1-2^{-\ell/\ell''}=1-2^{-\,\Omega(\sqrt{n}\cdot\log n)}$ erzeugt wird. Die Erzeugung des Individuums 10^{n-1} führt dazu, dass das Potenzial auf 1 zurückgesetzt wird. Wir haben bereits gesehen, dass sich das Potenzial in der betrachteten Phase mit Wahrscheinlichkeit $1-2^{-\,\Omega(\sqrt{n})}$ um weniger als $(1/6)\cdot n$ erhöht.

Wir betrachten $2^{c'\cdot\sqrt{n}}$ Phasen für eine hinreichend kleine Konstante c' und schließen, dass das Individuum mit Wahrscheinlichkeit $1-2^{-\,\Omega(\sqrt{n})}$ nicht gefunden wird. Insgesamt folgt das Theorem. □

Das nächste Theorem zeigt, dass der Algorithmus SEMO in der Lage ist, die Teilfunktion f_1 effizient zu optimieren. Der Beweis des folgenden Theorems basiert auf [Jansen und Wegener, 2001]. Dort wird die Selektion zur Ersetzung des Algorithmus (1+1)-EA [Droste et al., 2002] untersucht. Es wird insbesondere gezeigt, dass es einen

erheblichen Unterschied machen kann, ob Suchpunkte mit der gleichen Fitness wie der aktuelle Suchpunkt akzeptiert oder verworfen werden. Der Beweis des Theorems nutzt aus, dass der Algorithmus SEMO das Plateau relativ schnell erreicht. Dann beginnt der Algorithmus eine zufällige Irrfahrt auf dem Plateau bis er auf das globale Optimum 1^n stößt. Im Beweis wird im Wesentlichen gezeigt, dass diese Irrfahrt im Erwartungswert nach $\mathcal{O}(n^3)$ Generationen von Erfolg gekrönt ist.

Theorem 5.3. *Wir wenden den Algorithmus SEMO auf die Fitnessfunktion f_1 an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) = 0$ im Erwartungswert nach $\mathcal{O}(n^3)$ Generationen.*

Beweis. Wir gliedern einen Lauf des Algorithmus SEMO in zwei Phasen. Dazu definieren wir zunächst $SP := \{1^i 0^{n-i} \mid 1 \leq i \leq n\}$. Die erste Phase ist abgeschlossen, wenn $P \cap SP \neq \emptyset$ gilt. Die zweite Phase ist abgeschlossen, wenn $P \cap \{1^n\} \neq \emptyset$ gilt.

Nehmen wir an, wir befinden uns in Phase 1. Dann umfasst die Population genau ein Individuum $x \notin SP$. Die Wahrscheinlichkeit, ein neues Individuum x' mit $\mathrm{OneMax}(x') = \mathrm{OneMax}(x) - 1$ zu erzeugen, beträgt mindestens $\mathrm{OneMax}(x) \cdot (1/n) \cdot (1-1/n)^{n-1} \geq \mathrm{OneMax}(x)/(e \cdot n)$. Wir finden daher im Erwartungswert nach höchstens

$$\sum_{i=1}^{n-1} \frac{e \cdot n}{i} = e \cdot n \cdot H_{n-1} \leq e \cdot n \cdot (\ln(n-1) + 1) = \mathcal{O}(n \cdot \log n)$$

Generationen das Individuum 0^n oder ein Individuum aus SP. Im ersten Fall finden wir im Erwartungswert nach höchstens $e \cdot n$ weiteren Generationen das Individuum 10^{n-1} oder ein anderes Individuum aus SP.

Nehmen wir an, wir befinden uns in Phase 2. Dann setzt sich die Population aus genau einem Individuum $x \in (SP \setminus \{1^n\})$ zusammen.

Wir nennen einen Mutationsschritt *erfolgreich*, wenn ein Suchpunkt $x' \in (SP \setminus \{x\})$ erzeugt wird. Die Wahrscheinlichkeit, dass ein Mutationsschritt erfolgreich ist und genau 1 Bit kippt, beträgt mindestens $(1/n) \cdot (1-1/n)^{n-1} \geq 1/(e \cdot n)$. Die Wahrscheinlichkeit, dass ein Mutationsschritt erfolgreich ist und genau i Bits kippt, beträgt höchstens $2 \cdot (1/n)^i \cdot (1-1/n)^{n-i} \leq 2/n^i$. Folglich beträgt die Wahrscheinlichkeit, dass ein Mutationsschritt erfolgreich ist und mindestens i Bits kippt, höchstens $\sum_{j=i}^{n} 2/n^j \leq 2/((n-1) \cdot n^{i-1})$.

Wir betrachten eine Phase, die $c \cdot n^3$ aufeinander folgende Generationen umfasst, wobei c eine positive Konstante ist, die wir später spezifizieren werden. Die Zufallsvariable N_i zähle die erfolgreichen Mutationsschritte, in denen genau i Bits gekippt werden. Mithilfe einer Chernoff-Ungleichung schließen wir, dass $N_1 \geq (1-1/2) \cdot (c \cdot n^3)/(e \cdot n) = (c/(2 \cdot e)) \cdot n^2$ mit Wahrscheinlichkeit mindestens $1 - p_1$ mit $p_1 := e^{-(c/(8 \cdot e)) \cdot n^2}$ gilt. Die boolesche Ungleichung liefert, dass $N_i = 0$ für alle $i \geq 4$ mit Wahrscheinlichkeit mindestens $1 - p_2$ mit $p_2 := (c \cdot n^3) \cdot 2/((n-1) \cdot n^3) = (2 \cdot c)/(n-1)$ gilt.

Wir beschränken nun die Wahrscheinlichkeit, dass der Algorithmus einen der letzten drei Punkte des Pfades SP – d. h., $1^{n-2}0^2$, $1^{n-1}0^1$ oder 1^n – erreicht, nach unten. Wir nennen einen erfolgreichen Schritt *gut*, wenn er die Anzahl der 1-Bits erhöht.

Wenn wir keinen der genannten Punkte erreicht haben, dann beträgt die Wahrscheinlichkeit, dass ein erfolgreicher Schritt, in dem höchstens 3 Bits kippen, gut ist, mindestens 1/2. Wir nehmen pessimistisch an, dass diese Wahrscheinlichkeit genau 1/2 beträgt. Folglich beträgt die Wahrscheinlichkeit, dass mindestens die Hälfte aller Schritte der Länge 2 gut sind, $1 - p_3$ mit $p_3 := 1/2$. Außerdem beträgt die Wahrscheinlichkeit, dass mindestens die Hälfte aller Schritte der Länge 3 gut sind, ebenfalls $1 - p_4$ mit $p_4 := 1/2$. Wir leiten nun eine untere Schranke für die Wahrscheinlichkeit her, dass mindestens $(1/2) \cdot N_1 + (1/2) \cdot n$ aller Schritte der Länge 1 gut sind. Wir bezeichnen mit der Zufallsvariablen X die Anzahl guter erfolgreicher 1-Bit-Mutationen. Die Wahrscheinlichkeit, dass genau i erfolgreiche 1-Bit-Mutationen gut sind, beträgt höchstens

$$\begin{aligned}
\mathrm{W}(X = i) &= \binom{N_1}{i} \cdot \left(\frac{1}{2}\right)^{N_1} \\
&\leq \binom{N_1}{N_1/2} \cdot \left(\frac{1}{2}\right)^{N_1} \\
&= \frac{N_1!}{(N_1/2)! \cdot (N_1/2)!} \cdot \left(\frac{1}{2}\right)^{N_1} \\
&\leq \frac{\sqrt{2 \cdot \pi \cdot N_1} \cdot (N_1/e)^{N_1} \cdot 2}{\sqrt{\pi \cdot N_1} \cdot (N_1/(2 \cdot e))^{N_1/2} \cdot \sqrt{\pi \cdot N_1} \cdot (N_1/(2 \cdot e))^{N_1/2} \cdot 2^{N_1}} \\
&= \frac{2 \cdot \sqrt{2}}{\sqrt{\pi}} \cdot \frac{1}{\sqrt{N_1}} \leq c' \cdot \frac{1}{n},
\end{aligned}$$

wobei $c' := (4 \cdot \sqrt{e})/\sqrt{\pi \cdot c}$ gilt. In die letzte Rechnung sind sowohl die Definition des Binomialkoeffizienten (siehe Definition A.11) als auch die Stirling-Formel (siehe Lemma A.10) eingeflossen. Dann gilt

$$\begin{aligned}
\mathrm{W}\left(X \geq \frac{N_1 + n}{2}\right) &= 1 - \mathrm{W}\left(X \leq \frac{N_1}{2}\right) - \mathrm{W}\left(\frac{N_1}{2} < X < \frac{N_1 + n}{2}\right) \\
&= 1 - \frac{1}{2} - \sum_{i=N_1/2+1}^{(N_1+n)/2-1} \mathrm{W}(X = i) \\
&\geq \frac{1}{2} - \frac{n}{2} \cdot c' \cdot \frac{1}{n} \\
&= 1 - p_5
\end{aligned}$$

mit $p_5 := (1 + c')/2$.

Nehmen wir an, dass wir $1^{n-i}0^i$, $1 \leq i \leq 2$, erreicht haben. Wir betrachten eine Phase, die $i \cdot 2 \cdot e \cdot n$ aufeinander folgende Generationen umfasst. Dann schließen wir mithilfe der Markoff-Ungleichung, dass die Wahrscheinlichkeit, dass die Phase mindestens i erfolgreiche Mutationsschritte enthält, mindestens $1 - p_6$ mit $p_6 := 1/2$ beträgt. Dann beträgt die Wahrscheinlichkeit, dass die ersten i erfolgreichen Schritte gute 1-Bit-Mutationen sind, mindestens $1 - p_7$ mit $p_7 := 1 - ((1/(e \cdot n)) \cdot ((n-1)/2))^2$,

da die Wahrscheinlichkeit, dass ein Mutationsschritt, in dem mindestens 1 Bit gekippt wird, erfolgreich ist, höchstens $2/(n-1)$ beträgt.

Insgesamt folgt, dass das Optimum in einer Phase der Länge $c \cdot n^3 + 4 \cdot e \cdot n = \mathcal{O}(n^3)$ mit Wahrscheinlichkeit

$$\prod_{i=1}^{7}(1-p_i) = \left(1 - e^{-(c \cdot n^2)/(8 \cdot e)}\right) \cdot \left(1 - \frac{2 \cdot c}{n-1}\right) \cdot \left(\frac{1}{2}\right) \cdot \left(\frac{1}{2}\right) \cdot \left(\frac{\sqrt{\pi \cdot c} - 4 \cdot \sqrt{e}}{2 \cdot \sqrt{\pi \cdot c}}\right) \cdot \left(\frac{1}{2}\right) \cdot \left(\frac{n-1}{2 \cdot e \cdot n}\right)^2 = \frac{\sqrt{\pi \cdot c} - 4 \cdot \sqrt{e}}{64 \cdot e^2 \cdot \sqrt{\pi \cdot c}} \cdot (1 - \mathrm{o}(1))$$

gefunden wird. Wenn wir z. B. $c := (1 + 4 \cdot \sqrt{e})^2/\pi$ wählen, dann wird das globale Optimum mit Wahrscheinlichkeit mindestens $(1 - \mathrm{o}(1))/(64 \cdot e^2 \cdot (1 + 4 \cdot \sqrt{e}))$ in einer Phase gefunden. Der Algorithmus benötigt im Erwartungswert also $\mathcal{O}(1)$ Phasen der Länge $\mathcal{O}(n^3)$ bis er das globale Optimum findet. Insgesamt folgt das Theorem. □

Die Schwierigkeit, das Plateau zu überwinden, wird folglich durch die Hinzunahme von f_2 induziert. Das Zusammenspiel der beiden Teilfunktionen f_1 und f_2 und die daraus resultierende Fitnessstruktur des Suchraumes machen die Funktionen für den einfachen multikriteriellen evolutionären Algorithmus SEMO so schwierig (siehe Theorem 5.2).

Wir beweisen als nächstes, dass der Algorithmus FEMO die Paretofront der Beispielfunktion f im Erwartungswert nach polynomiell vielen Generationen findet. Die Beweisidee lässt sich wie folgt zusammenfassen. Der Algorithmus FEMO erreicht genau wie der Algorithmus SEMO relativ schnell eine Population, die das Individuum 0^n und ein Individuum aus SP enthält. Während der Nachkommenzähler des Individuums 0^n monoton wächst, wird der Nachkommenzähler des Individuums x aus SP auf 0 gesetzt, sobald ein von x verschiedenes Individuum aus SP gefunden wird. Daher ist sichergestellt, dass das Individuum aus SP „ungestört“ das Plateau explorieren kann, sobald der Nachkommenzähler von 0^n hinreichend groß ist. Wir werden diese Ideen im Folgenden in einen Beweis gießen.

Theorem 5.4. *Wir wenden den Algorithmus FEMO auf die Fitnessfunktion f an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) = 0$ im Erwartungswert nach $\mathcal{O}(n^3 \cdot \log n)$ Generationen.*

Beweis. Bevor wir beweisen, dass der Algorithmus FEMO die Suchpunkte 0^n und 1^n effizient findet, fassen wir zwei zentrale Ergebnisse aus dem Beweis von Theorem 5.3 zusammen. Der Suchpunkt 0^n wird mit Wahrscheinlichkeit mindestens $1/2$ erzeugt, wenn mindestens $\gamma \cdot n \cdot \log n$ Suchpunkte $x \notin SP$ zur Mutation ausgewählt werden, wobei $\gamma > 0$ eine hinreichend große Konstante ist. Der Suchpunkt 1^n wird mit Wahrscheinlichkeit mindestens $1/2$ erzeugt, wenn mindestens $\delta \cdot n^3$ Suchpunkte $x \in SP$ zur Mutation ausgewählt werden und wenn $x' \notin SP$ für alle Mutanten x' von Individuen $x \notin SP$ gilt, wobei $\delta > 0$ eine hinreichend große Konstante ist.

Wir zeigen zunächst, dass einer der Suchpunkte 0^n und 1^n im Erwartungswert nach $\mathcal{O}(n^3 \cdot \log n)$ Generationen in die Population eingefügt wird. Wir betrachten

eine Phase der Länge

$$\ell := (4 \cdot \gamma \cdot \log n) \cdot (\delta \cdot n^3 + \gamma \cdot n \cdot \log n - 1) = \mathcal{O}(n^3 \cdot \log n)$$

und unterscheiden zwei Fälle. Wenn mindestens $\gamma \cdot n \cdot \log n$ Suchpunkte $x \notin SP$ zur Mutation ausgewählt werden, dann beträgt die Wahrscheinlichkeit, den Suchpunkt 0^n zu erzeugen, mindestens $1/2$. Die Wahrscheinlichkeit, dass $x' \in SP$ für einen Mutanten x' eines Individuums $x \notin SP$ gilt, beträgt höchstens $2/n$. Daher gilt im zweiten Fall $x' \in SP$ für weniger als $4 \cdot \gamma \cdot \log n$ Mutanten x' von Individuen $x \notin SP$ mit Wahrscheinlichkeit mindestens $1/2$ wegen der Markoff-Ungleichung. Wenn wir annehmen, dass das genannte Ereignis eingetreten ist, dann können wir das Schubfachprinzip anwenden und schließen, dass die Phase eine Teilphase der Länge

$$\delta \cdot n^3 + \gamma \cdot n \cdot \log n - 1$$

enthält, in der $x' \notin SP$ für alle Mutanten x' von Individuen $x \notin SP$ gilt. Da insgesamt weniger als $\gamma \cdot n \cdot \log n$ Suchpunkte $x \notin SP$ zur Mutation ausgewählt werden, werden in der Phase mindestens $\delta \cdot n^3$ Suchpunkte $x \in SP$ zur Mutation ausgewählt. Also beträgt die Wahrscheinlichkeit, den Suchpunkt 1^n zu erzeugen, mindestens $1/4$. Da die Wahrscheinlichkeit, den Suchpunkt 0^n oder den Suchpunkt 1^n in einer Phase der Länge ℓ zu erzeugen, mindestens $1/4$ beträgt, tritt das Ereignis im Erwartungswert nach $4 \cdot \ell = \mathcal{O}(n^3 \cdot \log n)$ Generationen ein, da wir eine Folge von unabhängigen Phasen der Länge ℓ betrachten können. Diese Argumentation werden wir im weiteren Verlauf des Beweises noch häufiger verwenden.

Wir betrachten nun die Situation, dass der Suchpunkt 0^n gefunden ist und der Suchpunkt 1^n noch fehlt. Wir betrachten eine Phase der Länge

$$\ell' := (4 \cdot e \cdot \ln(2 \cdot \delta \cdot n^3) + 1) \cdot (\delta \cdot n^3 + e \cdot n \cdot \ln(2 \cdot \delta \cdot n^3)) = \mathcal{O}(n^3 \cdot \log n).$$

Wenn der Suchpunkt 0^n höchstens $(e \cdot n \cdot \ln(2 \cdot \delta \cdot n^3))$-mal zur Mutation ausgewählt wird, dann beträgt die Wahrscheinlichkeit, dass $x' \in SP$ für höchstens $4 \cdot e \cdot \ln(2 \cdot \delta \cdot n^3)$ Mutanten x' von 0^n gilt, mindestens $1/2$ wegen der Markoff-Ungleichung. Wenn wir annehmen, dass das genannte Ereignis eingetreten ist, dann enthält die Phase eine Teilphase der Länge

$$\delta \cdot n^3 + e \cdot n \cdot \ln(2 \cdot \delta \cdot n^3),$$

in der mindestens $\delta \cdot n^3$ Suchpunkte $x \in SP$ zur Mutation ausgewählt werden und $x' \notin SP$ für alle Mutanten x' des Suchpunktes 0^n gilt. Daher beträgt die Wahrscheinlichkeit, dass der Suchpunkt 1^n gefunden wird oder dass der Nachkommenzähler $c(0^n)$ den Wert $e \cdot n \cdot \ln(2 \cdot \delta \cdot n^3)$ übersteigt, mindestens $1/4$. Wenn der Suchpunkt 1^n noch fehlt, dann betrachten wir Phasen der Länge

$$2 \cdot e \cdot n^2 + \delta \cdot n^3.$$

Die Wahrscheinlichkeit, der Population einen *neuen* Suchpunkt $x' \in SP$ hinzuzufügen, beträgt mindestens $1/(e \cdot n^2)$, da höchstens 2 bestimmte Bits gekippt werden

müssen. Der ungünstigste Fall liegt vor, wenn der Suchpunkt 0^n zur Mutation ausgewählt wird und das Individuum 10^{n-1} bereits in der Population enthalten ist. Daher beträgt die Wahrscheinlichkeit, dass in den ersten $2 \cdot e \cdot n^2$ Generationen der Phase ein neues Individuum mit einem Nachkommenzähler mit dem Wert 0 in die Population aufgenommen wird, mindestens $1/2$ wegen der Markoff-Ungleichung. Wenn wir annehmen, dass das genannte Ereignis eingetreten ist, dann können wir die Wahrscheinlichkeit, dass das Individuum 0^n in den folgenden $\delta \cdot n^3$ Generationen ausgewählt wird, wie folgt begrenzen. Die Wahrscheinlichkeit, den Nachkommenzähler des Individuums $x \in SP$ auf 0 zu setzen, beträgt mindestens $1/(e \cdot n)$. Die Wahrscheinlichkeit, dass das genannte Ereignis nicht in einer Phase der Länge $e \cdot n \cdot \ln(2 \cdot \delta \cdot n^3)$ eintritt, beträgt höchstens

$$(1 - 1/(e \cdot n))^{e \cdot n \cdot \ln(2 \cdot \delta \cdot n^3)} \leq e^{-\ln(2 \cdot \delta \cdot n^3)} = 1/(2 \cdot \delta \cdot n^3).$$

Die Wahrscheinlichkeit, dass dies in einer Phase der Länge $\delta \cdot n^3$ *nicht* passiert, beträgt höchstens $\delta \cdot n^3 \cdot 1/(2 \cdot \delta \cdot n^3) \leq 1/2$. Wir schließen, dass der Nachkommenzähler des in der Population befindlichen Individuums $x \in SP$ mit Wahrscheinlichkeit mindestens $1/2$ den Wert $e \cdot n \cdot \ln(2 \cdot \delta \cdot n^3)$ nicht übersteigt. Daher wird das Individuum 0^n nicht zur Mutation ausgewählt. Wenn wir annehmen, dass das genannte Ereignis eingetreten ist, dann beträgt die Wahrscheinlichkeit, dass das Individuum 1^n gefunden wird, mindestens $1/2$. Folglich wird der Suchpunkt 1^n im Erwartungswert nach $8 \cdot (2 \cdot e \cdot n^2 + \delta \cdot n^3) = \mathcal{O}(n^3)$ Generationen gefunden.

Wir betrachten nun die Situation, dass der Suchpunkt 1^n gefunden ist und der Suchpunkt 0^n noch fehlt. Wir warten bis die Population ein zusätzliches Individuum $x \notin SP$ enthält und $c(x) \leq c(1^n)$ gilt. Anschließend betrachten wir Phasen der Länge

$$2 \cdot \gamma \cdot n \cdot \log n.$$

Wir können sicher sein, dass das Individuum $x \notin SP$ mindestens $(\gamma \cdot n \cdot \log n)$-mal zur Mutation ausgewählt wird, da $c(1^n)$ monoton wächst. Folglich wird der Suchpunkt 0^n im Erwartungswert nach $2 \cdot 2 \cdot \gamma \cdot n \cdot \log n = \mathcal{O}(n \cdot \log n)$ Generationen gefunden.

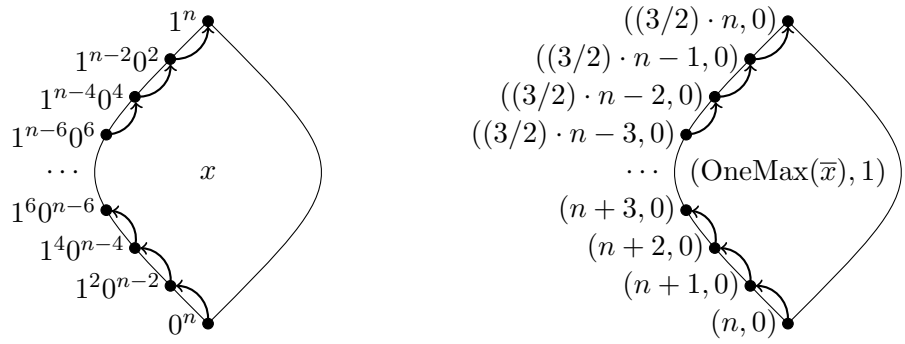
Insgesamt folgt das Theorem. □

5.4. Ein Problem, für das FEMO ineffizient und SEMO effizient ist

Wir haben im letzten Abschnitt gesehen, dass eine faire Auswahl die Optimierungszeit deutlich verkürzen kann. Wir beweisen hier, welche negativen Auswirkungen das faire Selektionsschema auf den Optimierungsprozess haben kann. Dazu betrachten wir die folgende Beispielfunktion.

Definition 5.5. *Sei $n = 2 \cdot m$ mit $m \in \mathbb{N}^+$. Die Beispielfunktion $g\colon \mathbb{B}^n \to \mathbb{R}^2$ definieren wir dann durch*

$$g(x) = \begin{cases} (\text{OneMax}(\overline{x}), 1) & \text{falls } x \notin \{1^{2 \cdot i}0^{n-2 \cdot i} \mid 0 \leq i \leq n/2\} \\ (n + i, 0) & \text{falls } x = 1^{2 \cdot i}0^{n-2 \cdot i},\ 0 \leq i \leq n/2. \end{cases}$$



(a) Suchraum $\mathbb{B}^n$.

(b) Zielraum $g(\mathbb{B}^n)$.

Abbildung 5.2.: Suchraum und Zielraum der Beispielfunktion g.

Die Beispielfunktion g wird in Abbildung 5.2 dargestellt. Sie besitzt zwei Pareto-optimale Zielpunkte, nämlich $(n-1, 1)$ und $(3/2 \cdot n, 0)$. Auf den ersten Zielpunkt werden alle Suchpunkte aus $\{x \in \mathbb{B}^n \mid \mathrm{OneMax}(x) = 1\}$ abgebildet. Von diesem Bereich des Suchraumes führt ein „löchriger“ Pfad zum Pareto-optimalen Suchpunkt 1^n, der auf den zweiten Zielpunkt abgebildet wird. Um dem Pfad zu folgen, müssen zwei bestimmte Bits gekippt werden.

Das folgende Theorem zeigt, dass der Algorithmus FEMO mit überwältigender Wahrscheinlichkeit in polynomiell vielen Schritten die Paretofront *nicht* findet. Dieses Verhalten wird wie folgt hervorgerufen. Der Algorithmus FEMO erreicht mit überwältigender Wahrscheinlichkeit eine Population, die aus einem Pareto-optimalen Individuum, das genau ein 1-Bit aufweist, und einem suboptimalen Individuum, das sich auf dem Pfad zum fehlenden Pareto-optimalen Individuum 1^n befindet, besteht. Der Nachkommenzähler des Individuums x mit $\mathrm{OneMax}(x) = 1$ wird jedes Mal auf 0 gesetzt, wenn ein von x verschiedenes Individuum mit einem 1-Bit generiert wird. Damit der Nachkommenzähler des Individuums auf dem Pfad zurückgesetzt wird, muss ein Individuum generiert werden, das näher am Ende des Pfades liegt (siehe Abbildung 5.2). Die Wahrscheinlichkeit für das erste Ereignis beträgt $\Omega(1/n)$ und die Wahrscheinlichkeit für das zweite Ereignis beträgt $\mathcal{O}(1/n^2)$. Daher passiert es mit überwältigender Wahrscheinlichkeit, dass der Nachkommenzähler des Individuums auf dem Pfad einen kritischen Wert übersteigt und das Individuum fortan nicht mehr zur Mutation ausgewählt wird. Im Beweis des folgenden Theorems formalisieren wir die skizzierten Ideen.

Theorem 5.6. *Wir wenden den Algorithmus FEMO auf die Fitnessfunktion g an. Dann gilt $\mathrm{I}_{\mathrm{add}}(P) > 0$ für die ersten $2^{\Omega(\sqrt{n})}$ Populationen P mit Wahrscheinlichkeit $1 - 2^{-\Omega(\sqrt{n})}$.*

Beweis. Wir definieren $SP := \{1^{2\cdot i}0^{n-2\cdot i} \mid 0 \leq i \leq n/2\}$. Das erste Individuum x besitzt mit Wahrscheinlichkeit $1-2^{-\Omega(n)}$ die Eigenschaften $x \notin SP$ und $\mathrm{OneMax}(x) \leq$

$(2/3)\cdot n$. Die Wahrscheinlichkeit, ein Individuum x' mit $\text{OneMax}(x') = \text{OneMax}(x)+i$ zu erzeugen, beträgt höchstens $1/n^i$. Also beträgt die Wahrscheinlichkeit, ein Individuum x' mit $\text{OneMax}(x') \geq \text{OneMax}(x)+i$ zu erzeugen, höchstens $1/((n-1)\cdot n^{i-1})$. Folglich beträgt die Wahrscheinlichkeit, dass in einer Phase der Länge $2^{\sqrt{n}}$ ein Individuum x' mit $\text{OneMax}(x') \geq \text{OneMax}(x) + \sqrt{n}$ erzeugt wird, höchstens $n^{-\Omega(\sqrt{n})}$.

Um den Suchpunkt 1^n zu finden, müssen nacheinander mindestens $\sqrt{n}/3$ verschiedene Suchpunkte $x \in SP$ mit $x > (2/3) \cdot n$ in die Population aufgenommen werden. Die Wahrscheinlichkeit, einen solchen Suchpunkt durch einen Mutationsschritt zu verbessern, beträgt höchstens $1/n^2$. Die Wahrscheinlichkeit, dass $n^2 - 1$ Versuche, einen solchen Suchpunkt zu verbessern, fehlschlagen, beträgt mindestens $(1 - 1/n^2)^{n^2-1} \geq 1/e$. Die Wahrscheinlichkeit, dass es einen Suchpunkt x^* mit den genannten Eigenschaften gibt, dessen Nachkommenzähler den Wert $n^2 - 1$ erreicht, beträgt folglich mindestens $1 - e^{-\sqrt{n}/3}$.

Wir beschränken nun den Wert des Nachkommenzählers des Individuums $x \notin SP$ und zeigen damit, dass kein Individuum mit $\text{OneMax}(x) > \text{OneMax}(x^*)$ gefunden wird. Betrachte einen Suchpunkt $x \notin SP$. Die Wahrscheinlichkeit, einen Suchpunkt x' mit $x' \neq x$, $x' \notin SP$ und $\text{OneMax}(x') = \text{OneMax}(x)$ zu erzeugen, beträgt mindestens

$$(\text{OneMax}(x) \cdot \text{OneMax}(\overline{x}) - 1) \cdot \frac{1}{e \cdot n^2} \geq \frac{1}{e \cdot (n-1)}.$$

Daher beträgt die Wahrscheinlichkeit, dass der Nachkommenzähler des Individuums $x \notin SP$ den Wert $n^2 - 1$ erreicht, höchstens

$$\begin{aligned}\left(1 - \frac{1}{e \cdot (n-1)}\right)^{n^2-1} &\leq \left(1 - \frac{1}{e \cdot (n-1)}\right)^{n^2} \cdot \left(1 - \frac{1}{e \cdot (n-1)}\right)^{-1} \\ &\leq e^{-n/e} \cdot \frac{e \cdot (n-1)}{e \cdot (n-1) - 1} \\ &= 2^{-\Omega(n)}.\end{aligned}$$

Die Wahrscheinlichkeit, dass das genannte Ereignis in einer Phase der Länge $2^{\sqrt{n}}$ eintritt, beträgt höchstens $2^{-\Omega(n)}$. Insgesamt folgt das Theorem. □

Wir beweisen abschließend, dass der Algorithmus SEMO sehr wohl in der Lage ist, die Paretofront im Erwartungswert in polynomiell vielen Schritten zu finden. Der Algorithmus SEMO findet genau wie der Algorithmus FEMO relativ schnell ein Individuum, das auf den Pareto-optimalen Zielpunkt $(n-1, 1)$ abgebildet wird. Nach wenigen Generationen enthält die Population zusätzlich ein Individuum auf dem Pfad zum zweiten Pareto-optimalen Zielpunkt. Da der Algorithmus SEMO das als nächstes zu mutierende Individuum rein zufällig aus der aktuellen Population zieht, bekommt das Individuum auf dem Pfad im Erwartungswert alle zwei Generationen eine Chance, sich zu verbessern. Folglich wird der zweite Pareto-optimale Zielpunkt $(3/2 \cdot n, 0)$ relativ schnell gefunden. Wenn wir die genannten Beobachtungen formalisieren, dann erhalten wir das folgende Theorem.

Theorem 5.7. *Wir wenden den Algorithmus SEMO auf die Fitnessfunktion g an. Dann gilt* $\mathrm{I}_{\mathrm{add}}(P) = 0$ *im Erwartungswert nach* $\mathcal{O}(n^3)$ *Generationen.*

Beweis. Wir gliedern einen Lauf des Algorithmus SEMO in vier Phasen. Die erste Phase ist abgeschlossen, wenn $P \cap (\mathbb{B}^n \setminus SP) \neq \emptyset$ gilt. Die zweite Phase ist abgeschlossen, wenn zusätzlich $P \cap \{x \in \mathbb{B}^n \mid \mathrm{OneMax}(x) = 1\} \neq \emptyset$ erfüllt ist. Die dritte Phase ist abgeschlossen, wenn zusätzlich $P \cap SP \neq \emptyset$ gilt. Die vierte Phase ist abgeschlossen, wenn zusätzlich $P \cap \{1^n\} \neq \emptyset$ gilt. Natürlich können einzelne Phasen auch leer sein.

Nehmen wir an, wir befinden uns in Phase 1. Dann besteht die Population aus einem Individuum $x \in SP$. Die Wahrscheinlichkeit, ein Individuum $x' \notin SP$ zu erzeugen, beträgt mindestens $1/(e \cdot n)$. Demzufolge beenden wir im Erwartungswert nach höchstens $e \cdot n = \mathcal{O}(n)$ Generationen Phase 1.

Nehmen wir an, wir befinden uns in Phase 2. Dann besteht die Population aus einem Individuum $x \notin SP$ und höchstens einem Individuum $y \in SP$. Die Wahrscheinlichkeit, ein Individuum $x' \notin SP$ mit $\mathrm{OneMax}(x') = \mathrm{OneMax}(x) - 1$ zu erzeugen, beträgt mindestens $(\mathrm{OneMax}(x) - 1)/(2 \cdot e \cdot n)$. Wir finden daher im Erwartungswert nach höchstens

$$\sum_{i=2}^{n-1} \frac{2 \cdot e \cdot n}{i-1} = 2 \cdot e \cdot n \cdot H_{n-2} \leq 2 \cdot e \cdot n \cdot (\ln(n-2) + 1) = \mathcal{O}(n \cdot \log n)$$

Generationen ein Individuum x mit $\mathrm{OneMax}(x) = 1$, wobei H_n die n-te harmonische Zahl bezeichnet.

Nehmen wir an, wir befinden uns in Phase 3. Dann enthält die Population genau ein Individuum x mit $\mathrm{OneMax}(x) = 1$. Die Wahrscheinlichkeit, das Individuum 0^n oder ein anderes Individuum aus SP zu erzeugen, beträgt mindestens $1/(e \cdot n)$. Demzufolge wird Phase 3 im Erwartungswert nach $e \cdot n$ Generationen beendet.

Nehmen wir an, wir befinden uns in Phase 4. Dann besteht die Population aus einem Individuum $x \in SP$ und einem Individuum $y \notin SP$. Die Wahrscheinlichkeit, ein Individuum $x' \in SP$ mit $\mathrm{OneMax}(x') = \mathrm{OneMax}(x) + 2$ zu erzeugen, beträgt mindestens $1/(2 \cdot e \cdot n^2)$. Der Algorithmus findet daher im Erwartungswert nach höchstens $(n/2) \cdot (2 \cdot e \cdot n^2) = \mathcal{O}(n^3)$ Generationen das noch fehlende Individuum 1^n.

Insgesamt folgt das Theorem. □

5.5. Fazit

Wir haben in diesem Kapitel den von Laumanns et al. [2004] vorgestellten Algorithmus FEMO untersucht. Der Algorithmus unterscheidet sich von dem Algorithmus SEMO einzig in der Selektion zur Mutation. Während der Algorithmus SEMO, das als nächstes zu mutierende Individuum rein zufällig aus seiner Population zieht, wählt der Algorithmus FEMO ein Individuum, das von allen in der Population befindlichen Individuen die wenigsten Mutanten hervorgebracht hat. Auf diese Weise soll sichergestellt werden, dass der Suchraum um alle in der Population befindlichen Individuen

möglichst gleichmäßig nach besseren Suchpunkten abgesucht wird. Laumanns et al. [2004] haben gezeigt, dass diese Vorgehensweise die Optimierungszeit um den fast linearen Faktor $\Omega((\log n)/n)$ beschleunigen kann.

Wir haben in diesem Kapitel bewiesen, dass die Auswirkungen der fairen Selektion noch weitaus drastischer sein können. Wir haben in Abschnitt 5.3 eine einfache Beispielfunktion präsentiert und bewiesen, dass der Algorithmus die Paretofront mit überwältigender Wahrscheinlichkeit in polynomiell vielen Generationen *nicht* findet. Andererseits gelingt es dem Algorithmus FEMO im Erwartungswert bereits nach $\mathcal{O}(n^3 \cdot \log n)$ Generationen die Paretofront aufzuspüren. In Abschnitt 5.4 haben wir andererseits bewiesen, dass die faire Selektion den Optimierungsprozess auch nachteilig beeinflussen kann. Wir haben eine Beispielfunktion vorgestellt, deren Paretofront von dem Algorithmus FEMO in polynomiell vielen Generationen mit überwältigender Wahrscheinlichkeit *nicht* gefunden wird. Hier zeigt sich, dass der Algorithmus SEMO dem Algorithmus FEMO deutlich überlegen ist, da dieser die Paretofront im Erwartungswert bereits nach $\mathcal{O}(n^3)$ Generationen findet.

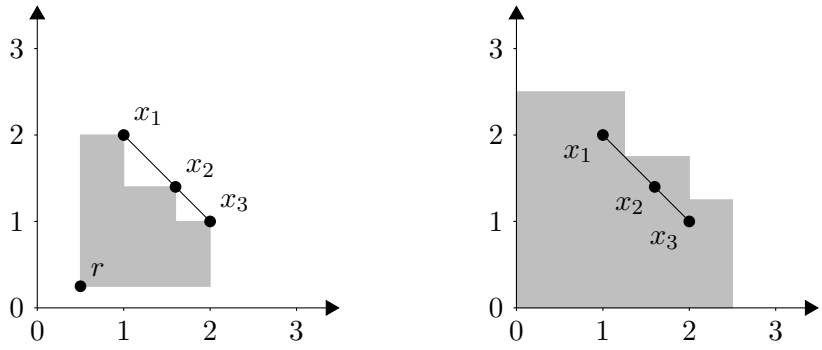
Insgesamt tragen die vorgestellten Beispiele dazu bei, die möglichen Implikationen, die der Einsatz eines fairen Selektionsschemas mit sich bringt, einordnen zu können.

6. Vergleich der Qualitätsindikatoren Hypervolumen und Approximationsgüte

6.1. Einleitung

Algorithmen, die auf einem Qualitätsindikator basieren, haben sich zu einer sehr beliebten Methode entwickelt, um multikriterielle Optimierungsprobleme zu lösen. In diesem Kapitel vertiefen wir das theoretische Verständnis von Algorithmen, die das Hypervolumen [Zitzler und Thiele, 1999] maximieren, indem sie μ Punkte auf der Paretofront verteilen. Das Hypervolumen stellt einen Qualitätsindikator dar, der sich großer Beliebtheit erfreut (siehe z. B. [Beume et al., 2007, Igel et al., 2007, Knowles und Corne, 2003, Zitzler et al., 2007]). In den letzten Jahren sind zahlreiche evolutionäre Algorithmen entwickelt worden, die das Hypervolumen benutzen, um ihre Suche zu lenken [Beume et al., 2007, Emmerich et al., 2005, Igel et al., 2007, Knowles et al., 2003, Zitzler et al., 2007, Zitzler und Künzli, 2004], obwohl das implizite Optimierungsziel schwer zu verstehen und noch weitgehend unerforscht ist. Wir untersuchen hier das Optimierungsziel dieser beliebten Methode.

Auger et al. [2009a] haben kürzlich gezeigt, dass die Steigung einer kontinuierlichen Paretofront beeinflusst, welche Punkte das Hypervolumen maximieren. Eine fundierte und in der Informatik weitverbreitete Methode, um die Güte einer Menge von Alternativen zu messen, besteht darin, ihre Approximationsgüte zu betrachten [Papadimitriou und Yannakakis, 2000]. Um das durch die Maximierung des Hypervolumens definierte Optimierungsziel besser zu verstehen, vergleichen wir die multiplikative Approximationsgüte der Verteilung von μ Punkten, die das Hypervolumen maximiert, – die Hypervolumenverteilung – mit der multiplikativen Approximationsgüte der Verteilung von μ Punkten, die die Approximationsgüte minimiert, – die Approximationsverteilung – auf verschiedenen Paretofronten. Für die Klasse von linearen Fronten und eine Klasse von konkaven Fronten beweisen wir, dass die Hypervolumenverteilung die optimale Approximationsgüte liefert. Wir werden sehen, dass die Analyse der Hypervolumen- wie auch der Approximationsverteilung auf relativ einfachen Klassen von Fronten bereits sehr komplex ist und schnell an ihre Grenzen stößt. Numerische Methoden ermöglichen hier die Betrachtung allgemeinerer Funktionenklassen. Wir untersuchen im Anschluss mithilfe numerischer Methoden eine recht allgemeine Klasse von Fronten, deren Krümmung und Skalierung durch verschiedene Parameter eingestellt werden kann. Wir zeigen, dass die Approximationsgüte der Hypervolumenverteilung deutlich von der optimalen Approximationsgüte



(a) Hypervolumen. (b) Approximationsgüte.

Abbildung 6.1.: Verteilung $X = \{1; 1{,}6; 2\}$ auf der linearen Front $f\colon [1,2] \to [1,2]$ mit $f(x) = 3 - x$, die das Hypervolumen $\mathrm{I}_{\mathrm{hyp}}(X) = 1{,}865$ (mit Referenzpunkt $r = (0{,}5; 0{,}25)$) und die Approximationsgüte $\mathrm{I}_{\mathrm{mul}}(X) = 1{,}25$ aufweist. Der grau hinterlegte Bereich symbolisiert den dominierten bzw. approximierten Anteil des Zielraumes.

abweichen kann. Dies ist eine wichtige Beobachtung, die bei Verwendung eines Algorithmus, der das Hypervolumen maximiert, beachtet werden sollte. Außerdem decken die Untersuchungen zahlreiche weitere interessante und teilweise überraschende Eigenschaften des Hypervolumens auf.

6.2. Szenario

Wir untersuchen in diesem Kapitel bikriterielle Maximierungsprobleme $P\colon S \to \mathbb{R}^2$, deren Paretofront $\{(x, f(x)) \mid x \in [x_{\min}, x_{\max}]\}$ entspricht, wobei $f\colon [x_{\min}, x_{\max}] \to \mathbb{R}$ eine stetige, differenzierbare und streng monoton fallende Funktion ist. Wir bezeichnen der Einfachheit halber mit f sowohl die Funktion $f\colon [x_{\min}, x_{\max}] \to \mathbb{R}$ als auch die Paretofront $\{(x, f(x)) \mid x \in [x_{\min}, x_{\max}]\}$. Wir nehmen weiterhin an, dass $x_{\min} > 0$ und $f(x_{\max}) > 0$ gelten.

Wir betrachten hier Teilmengen der Paretofront, die aus genau μ Elementen bestehen. Diese Mengen $\{(x_i, f(x_i)) \mid x_i \in X\}$ identifizieren wir mit der *Verteilung* $X = \{x_1, \ldots, x_\mu\} \subseteq [x_{\min}, x_{\max}]$. In diesem Kapitel geht es hauptsächlich um den Vergleich der Approximationsgüte der Verteilung, die das Hypervolumen maximiert, mit der Approximationsgüte der Verteilung, die die Approximationsgüte minimiert (siehe Abbildung 6.1(a) und 6.1(b) für ein Beispiel).

Wir wissen, dass das Hypervolumen $\mathrm{I}_{\mathrm{hyp}}(X)$ von der Wahl des Referenzpunktes $r = (r_1, r_2)$ abhängt (siehe Abschnitt 2.2). Da sowohl der Definitions- als auch der Wertebereich der hier betrachteten Funktionen f positiv und beschränkt sind, können wir $r_1 \leq 0$ und $r_2 \leq 0$ geeignet wählen, um sicherzustellen, dass $x_{\min}$ und $x_{\max}$ in der Verteilung X enthalten sind, die das Hypervolumen maximiert. In [Auger et al., 2009a] wird detailliert dargestellt, wie durch die Wahl eines geeigneten Referenz-

punktes sichergestellt werden kann, dass die Randpunkte der Paretofront in der Verteilung, die das Hypervolumen maximiert, enthalten sind (siehe auch Unterabschnitt 6.3.3). Um den angestrebten Vergleich nicht zu verzerren, fordern wir, dass auch die Verteilung, die die Approximationsgüte minimiert, $x_{\min}$ und $x_{\max}$ enthält.

Betrachte eine Paretofront f und eine feste Größe $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Wir begrenzen unsere Untersuchungen aus den im letzten Absatz genannten Gründen auf Verteilungen aus der Menge

$$\mathcal{X}(\mu, f) := \{X \subseteq [x_{\min}, x_{\max}] \mid |X| = \mu,\ x_{\min} \in X \text{ und } x_{\max} \in X\}.$$

Wir werden im Folgenden zwei bestimmte Verteilungen aus $\mathcal{X}(\mu, f)$ miteinander vergleichen, die *Hypervolumenverteilung* und die *Approximationsverteilung*.

Definition 6.1. *Seien f eine Paretofront und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Die* Hypervolumenverteilung

$$X_{\text{opt}}^{\text{HYP}}(\mu, f) := \operatorname*{argmax}_{X \in \mathcal{X}(\mu,f)} \mathrm{I}_{\text{hyp}}(X)$$

besteht aus μ Punkten, die das Hypervolumen bezüglich f maximieren. Die Approximationsverteilung

$$X_{\text{opt}}^{\text{APP}}(\mu, f) := \operatorname*{argmin}_{X \in \mathcal{X}(\mu,f)} \mathrm{I}_{\text{mul}}(X)$$

besteht aus μ Punkten, die die Approximationsgüte bezüglich f minimieren.

6.3. Analytische Untersuchungen

6.3.1. Ansatz

Wir erinnern daran, dass $\mathcal{X}(\mu, f)$ definitionsgemäß ausschließlich Verteilungen $X \subseteq [x_{\min}, x_{\max}]$ enthält, die sowohl $x_{\min}$ als auch $x_{\max}$ enthalten. Wir grenzen unsere Betrachtungen nun auf solche Verteilungen ein.

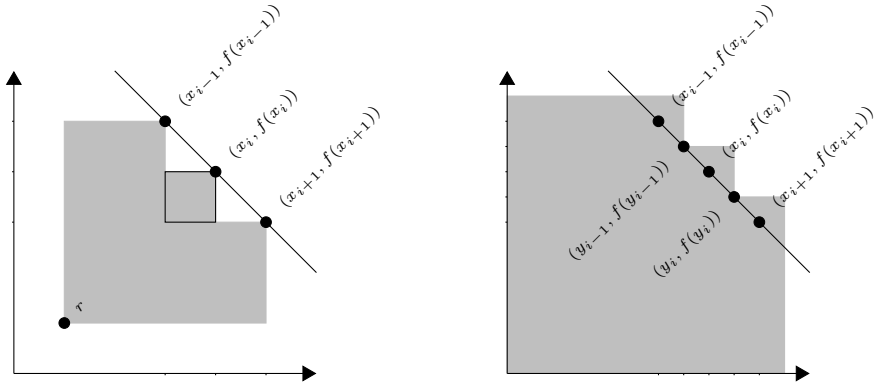
Das nächste Lemma, das von Auger et al. [2009a] gezeigt wurde, liefert eine genaue Charakterisierung der Hypervolumenverteilung.

Lemma 6.2 (Notwendige und hinreichende Bedingung für Hypervolumenverteilungen)**.** *Seien $f\colon [x_{\min}, x_{\max}] \to \mathbb{R}$ eine Paretofront und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Betrachte eine Verteilung $X = \{x_1, \ldots, x_\mu\}$ mit $x_i < x_{i+1}$ für alle $1 \leq i < \mu$ sowie $x_1 = x_{\min}$ und $x_\mu = x_{\max}$. Genau dann wenn $X = X_{\text{opt}}^{\text{HYP}}(\mu, f)$ gilt, dann gilt*

$$f'(x_i) \cdot (x_i - x_{i-1}) = f(x_{i+1}) - f(x_i)$$

für alle $1 < i < \mu$.

Beweis. Die Position eines Punktes $(x_i, f(x_i))$ zwischen seinen beiden Nachbarn $(x_{i-1}, f(x_{i-1}))$ und $(x_{i+1}, f(x_{i+1}))$ beeinflusst den Bereich, der ausschließlich von $(x_i, f(x_i))$ dominiert wird (siehe Abbildung 6.2(a)). Wenn wir den Bereich, der von



(a) Zum Beweis von Lemma 6.2. (b) Zum Beweis von Lemma 6.3.

Abbildung 6.2.: Situation im Beweis von Lemma 6.2 und 6.3. Der grau hinterlegte Bereich symbolisiert den dominierten bzw. δ-approximierten Anteil des Zielraumes.

$(x_{i-1}, f(x_{i-1}))$ dominiert wird, und den Bereich, der von $(x_{i+1}, f(x_{i+1}))$ dominiert wird, nicht berücksichtigen, dann misst

$$H_i(x_i) := (x_i - x_{i-1}) \cdot (f(x_i) - f(x_{i+1}))$$

den Beitrag des Punktes $(x_i, f(x_i))$ zum Hypervolumen. Für eine Hypervolumenverteilung muss gelten, dass x_i ein globales Maximum der Funktion $H_i \colon [x_{i-1}, x_{i+1}] \to \mathbb{R}$ ist, da das Hypervolumen anderenfalls vergrößert werden könnte. Da der Beitrag für $x_i = x_{i-1}$ und $x_i = x_{i+1}$ den Wert 0 annimmt, muss das Maximum in dem offenen Intervall (x_{i-1}, x_{i+1}) zu finden sein. Folglich gilt notwendigerweise $H_i'(x_i) = 0$. Es gilt

$$\begin{aligned} H_i'(x_i) &= ((x_i - x_{i-1}) \cdot (f(x_i) - f(x_{i+1})))' \\ &= (x_i \cdot f(x_i) - x_i \cdot f(x_{i+1}) - x_{i-1} \cdot f(x_i) + x_{i-1} \cdot f(x_{i+1}))' \\ &= f(x_i) + x_i \cdot f'(x_i) - f(x_{i+1}) - x_{i-1} \cdot f'(x_i). \end{aligned}$$

Insgesamt folgt die Aussage des Lemmas. □

Das folgende Lemma liefert eine genaue Charakterisierung der Approximationsverteilung.

Lemma 6.3 (Notwendige und hinreichende Bedingung für Approximationsverteilungen)**.** *Seien $f \colon [x_{\min}, x_{\max}] \to \mathbb{R}$ eine Paretofront und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Betrachte eine Verteilung $X = \{x_1, \ldots, x_\mu\}$ mit $x_i < x_{i+1}$ für alle $1 \leq i < \mu$ sowie $x_1 = x_{\min}$ und $x_\mu = x_{\max}$. Genau dann wenn $X = X_{\text{opt}}^{\text{APP}}(\mu, f)$ gilt, dann gibt es*

1. *eine Verteilung $Y = \{y_1, \ldots, y_{\mu-1}\}$ mit $x_i < y_i < x_{i+1}$ für alle $1 \leq i < \mu$, und*
2. *ein $\delta > 1$ mit $\frac{y_i}{x_i} = \delta$ und $\frac{f(y_i)}{f(x_{i+1})} = \delta$ für alle $1 \leq i < \mu$.*

Beweis. Die Approximationsgüte der Verteilung X beträgt offensichtlich δ, da die Elemente aus Y am schlechtesten approximiert werden. Die Situation wird in Abbildung 6.2(b) dargestellt.

Um $X = X_{\text{opt}}^{\text{APP}}(\mu, f)$ zu beweisen, nehmen wir an, dass es eine von X verschiedene Verteilung $X' = \{x'_1, \ldots, x'_\mu\}$ mit $x'_i < x'_{i+1}$ für alle $1 \leq i < \mu$ sowie $x'_1 = x_{\min}$ und $x'_\mu = x_{\max}$ gibt, deren Approximationsgüte höchstens δ beträgt. Wir zeigen, dass diese Annahme zu einem Widerspruch führt.

Da $X' \neq X$ gilt, gibt es einen Index i mit $x'_i \neq x_i$. Wir betrachten den kleinsten Index i mit der genannten Eigenschaft. Wir unterscheiden die Fälle $x'_i < x_i$ und $x'_i > x_i$. Wir betrachten zunächst den ersten Fall. Wir schließen $x'_{i+1} < x_{i+1}$ aus $x'_i < x_i$, da anderenfalls y_i nicht mit Güte δ approximiert würde. Wir wenden das Argument $(\mu - i)$-mal an und gelangen zu dem Widerspruch $x_{\max} = x'_\mu < x_\mu = x_{\max}$. Wir betrachten nun den zweiten Fall. Wenn $x'_i > x_i$ gilt, dann wird das offene Intervall $(y_{i-1}, f^{-1}(\delta \cdot f(x'_i)))$ nicht mit Güte δ approximiert. Das Intervall ist nicht leer, da die Ungleichungen $f^{-1}(\delta \cdot f(x'_i)) > x'_i > x_i > y_{i-1}$ aus den Voraussetzungen $\delta > 1$ und f streng monoton fallend folgen. Folglich erhalten wir erneut einen Widerspruch. Insgesamt folgt das Lemma. □

Wir werden Lemma 6.3 in diesem Abschnitt verwenden, um zu prüfen, ob die Hypervolumenverteilung der Approximationsverteilung entspricht.

6.3.2. Ergebnisse

Wir bestimmen in diesem Abschnitt die Hypervolumenverteilung und die Approximationsverteilung für zwei grundlegende Klassen von Funktionen f.

Wir definieren zunächst eine Funktionenklasse, die Funktionen enthält, die lineare Paretofronten repräsentieren.

Definition 6.4. *Definiere die Funktionenklasse*

$$\mathcal{F}_1 := \{f \colon [1, (1-d)/c] \to [1, c+d] \text{ mit } f(x) = c \cdot x + d\},$$

wobei $c < 0$ und $d > 1 - c$ frei gewählt werden können.

Die Verteilung, die das Hypervolumen für die genannten Paretofronten maximiert, wurde bereits in [Auger et al., 2009a] betrachtet. Auger et al. [2009a] zeigen, dass das maximale Hypervolumen aus einer gleichmäßigen Verteilung der μ Punkte resultiert.

Die Aussage des nächsten Theorems ist, dass die Verteilung, die das Hypervolumen maximiert, mit der Verteilung, die die Approximationsgüte minimiert, übereinstimmt.

Theorem 6.5. *Seien $f \in \mathcal{F}_1$ und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Dann gilt*

$$X_{\text{opt}}^{\text{HYP}}(\mu, f) = X_{\text{opt}}^{\text{APP}}(\mu, f) = \{x_1, \ldots, x_\mu\}$$

mit

$$x_i = 1 + \frac{i-1}{\mu - 1} \cdot \left(\frac{1-d}{c} - 1\right),$$

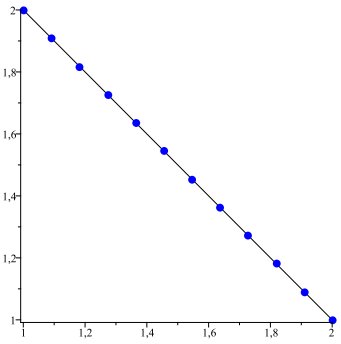


Abbildung 6.3.: Hypervolumenverteilung $X_{\text{opt}}^{\text{HYP}}(12, f)$ und Approximationsverteilung $X_{\text{opt}}^{\text{APP}}(12, f)$ auf der linearen Front $f\colon [1,2] \to [1,2]$ mit $f(x) = 3 - x$. Die Verteilungen stimmen hier überein.

wobei $f\colon [1,(1-d)/c] \to [1, c+d]$ *durch* $f(x) = c \cdot x + d$ *mit* $c < 0$ *und* $d > 1 - c$ *definiert ist.*

Beweis. Wir betrachten die Hypervolumenverteilung $X_{\text{opt}}^{\text{HYP}}(\mu, f) = \{x_1, \ldots, x_\mu\}$ mit $x_i < x_{i+1}$ für alle $1 \leq i < \mu$ sowie $x_1 = x_{\min}$ und $x_\mu = x_{\max}$. Laut Lemma 6.2 gilt

$$\begin{aligned} & f'(x_i) \cdot (x_i - x_{i-1}) = f(x_{i+1}) - f(x_i) \\ \iff\ & c \cdot (x_i - x_{i-1}) = (c \cdot x_{i+1} + d) - (c \cdot x_i + d) \\ \iff\ & x_{i+1} = 2 \cdot x_i - x_{i-1} \end{aligned}$$

für alle $1 < i < \mu$. Mithilfe der Gleichung $x_1 = 1$ schließen wir

$$\begin{array}{lcll} x_3 & = & 2 \cdot x_2 - x_1 & = & 2 \cdot x_2 - 1, \\ x_4 & = & 2 \cdot x_3 - x_2 & = & 3 \cdot x_2 - 2, \\ & \vdots & & \vdots & \\ x_\mu & = & 2 \cdot x_{\mu-1} - x_{\mu-2} & = & (\mu-1) \cdot x_2 - (\mu-2). \end{array}$$

Mithilfe der Gleichung $x_\mu = (1-d)/c$ schließen wir

$$\begin{array}{lcll} x_2 & = & \frac{x_\mu + \mu - 2}{\mu - 1} & = & \frac{1}{\mu-1} \cdot (\frac{1-d}{c} + \mu - 2), \\ x_3 & = & 2 \cdot x_2 - 1 & = & \frac{2}{\mu-1} \cdot (\frac{1-d}{c} + \mu - 2) - 1, \\ & \vdots & & \vdots & \\ x_{\mu-1} & = & (\mu-2) \cdot x_2 - (\mu-3) & = & \frac{\mu-2}{\mu-1} \cdot (\frac{1-d}{c} + \mu - 2) - (\mu-3). \end{array}$$

Folglich gilt

$$x_i = 1 + \frac{i-1}{\mu-1} \cdot \left(\frac{1-d}{c} - 1 \right) \tag{6.1}$$

für alle $1 \leq i \leq \mu$.

Wir beweisen nun, dass $X_{\text{opt}}^{\text{APP}}(\mu, f) = X_{\text{opt}}^{\text{HYP}}(\mu, f)$ gilt. Betrachte einen Punkt $\tilde{x}$ mit $x_i \leq \tilde{x} \leq x_{i+1}$. Dieser Punkt wird durch $X_{\text{opt}}^{\text{HYP}}(\mu, f)$ mit Güte

$$\min\left\{\frac{\tilde{x}}{x_i}, \frac{f(\tilde{x})}{f(x_{i+1})}\right\}$$

approximiert. Da f streng monoton fällt, wird die schlechteste Approximationsgüte erreicht, wenn

$$\frac{\tilde{x}}{x_i} = \frac{f(\tilde{x})}{f(x_{i+1})} \tag{6.2}$$

gilt. Wir lösen das durch Formel 6.1 und Formel 6.2 gebildete lineare Gleichungssystem nach $\tilde{x}$ auf und erhalten

$$\tilde{x} = \frac{d \cdot ((d+c-1) \cdot i - c \cdot \mu - d + 1)}{c \cdot ((\mu - 2) \cdot d - c + 1)}.$$

Es ergibt sich die Approximationsgüte

$$\frac{\tilde{x}}{x_i} = \frac{f(\tilde{x})}{f(x_{i+1})} = \frac{d \cdot (\mu - 1)}{d \cdot (\mu - 1) - c + 1}.$$

Die Approximationsgüte ist folglich für alle Intervalle $[x_i, x_{i+1}]$ identisch. Wir können nun Lemma 6.3 anwenden und $X_{\text{opt}}^{\text{APP}}(\mu, f) = X_{\text{opt}}^{\text{HYP}}(\mu, f)$ schließen. □

Abbildung 6.3 veranschaulicht die optimale Verteilung exemplarisch für eine lineare Front. Der Beweis des letzten Theorems zeigt außerdem, dass die optimale Approximationsgüte, die mit μ Elementen erreicht werden kann, genau

$$\frac{d \cdot (\mu - 1)}{d \cdot (\mu - 1) - c + 1}$$

beträgt.

Wir betrachten nun eine Funktionenklasse, die konkave Paretofronten enthält. Beachte, dass sich der Begriff „konkav“ hier auf die Menge bezieht, die sich unterhalb der Front befindet.

Definition 6.6. *Definiere die Funktionenklasse*

$$\mathcal{F}_2 := \{f \colon [1, c] \to [1, c] \text{ mit } f(x) = c/x\},$$

wobei $c > 1$ frei gewählt werden kann.

Das nächste Theorem zeigt, dass auch in diesem Fall die Hypervolumenverteilung mit der Approximationsverteilung übereinstimmt.

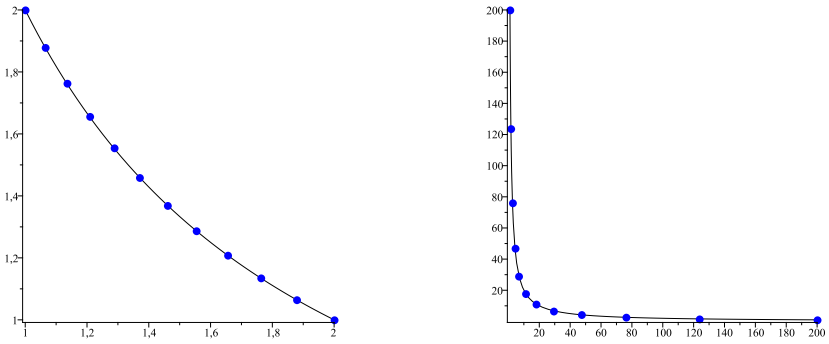
Theorem 6.7. *Seien $f \in \mathcal{F}_2$ und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Dann gilt*

$$X_{\text{opt}}^{\text{HYP}}(\mu, f) = X_{\text{opt}}^{\text{APP}}(\mu, f) = \{x_1, \ldots, x_\mu\}$$

mit

$$x_i = c^{(i-1)/(\mu-1)},$$

wobei $f \colon [1, c] \to [1, c]$ durch $f(x) = c/x$ mit $c > 1$ definiert ist.



(a) $c = 2$. (b) $c = 200$.

Abbildung 6.4.: Hypervolumenverteilung $X_{\text{opt}}^{\text{HYP}}(12, f)$ und Approximationsverteilung $X_{\text{opt}}^{\text{APP}}(12, f)$ auf den konkaven Fronten $f\colon [1, c] \to [1, c]$ mit $f(x) = c/x$ sowie $c = 2$ und $c = 200$. Die Verteilungen stimmen hier überein.

Beweis. Wir betrachten die Hypervolumenverteilung $X_{\text{opt}}^{\text{HYP}}(\mu, f) = \{x_1, \ldots, x_\mu\}$ mit $x_i < x_{i+1}$ für alle $1 \leq i < \mu$ sowie $x_1 = x_{\min}$ und $x_\mu = x_{\max}$. Laut Lemma 6.2 gilt

$$\begin{aligned} f'(x_i) \cdot (x_i - x_{i-1}) = f(x_{i+1}) - f(x_i) &\iff -c/x_i^2 \cdot (x_i - x_{i-1}) = c/x_{i+1} - c/x_i \\ &\iff x_{i+1} = x_i^2/x_{i-1} \end{aligned}$$

für alle $1 < i < \mu$. Mithilfe der Gleichung $x_1 = 1$ schließen wir

$$\begin{array}{rclcl} x_3 & = & x_2^2/x_1 & = & x_2^2, \\ x_4 & = & x_3^2/x_2 & = & x_2^3, \\ & \vdots & & \vdots & \\ x_\mu & = & x_{\mu-1}^2/x_{\mu-2} & = & x_2^{\mu-1}. \end{array}$$

Mithilfe der Gleichung $x_\mu = c$ schließen wir

$$\begin{array}{rclcl} x_2 & = & x_\mu^{1/(\mu-1)} & = & c^{1/(\mu-1)}, \\ x_3 & = & x_2^2 & = & c^{2/(\mu-1)}, \\ & \vdots & & \vdots & \\ x_{\mu-1} & = & x_2^{\mu-2} & = & c^{(\mu-2)/(\mu-1)}. \end{array}$$

Folglich gilt

$$x_i = c^{(i-1)/(\mu-1)} \tag{6.3}$$

für alle $1 \leq i \leq \mu$.

Wir beweisen nun, dass $X_{\text{opt}}^{\text{APP}}(\mu, f) = X_{\text{opt}}^{\text{HYP}}(\mu, f)$ gilt. Betrachte einen Punkt $\tilde{x}$ mit $x_i \leq \tilde{x} \leq x_{i+1}$. Dieser Punkt wird durch $X_{\text{opt}}^{\text{HYP}}(\mu, f)$ mit Güte

$$\min\left\{\frac{\tilde{x}}{x_i}, \frac{f(\tilde{x})}{f(x_{i+1})}\right\}$$

approximiert. Da f streng monoton fällt, wird die schlechteste Approximationsgüte erreicht, wenn

$$\frac{\tilde{x}}{x_i} = \frac{f(\tilde{x})}{f(x_{i+1})} \tag{6.4}$$

gilt. Wir lösen das durch Formel 6.3 und Formel 6.4 gebildete lineare Gleichungssystem nach $\tilde{x}$ auf und erhalten

$$\tilde{x} = c^{(2\cdot i-1)/(2\cdot\mu-2)}.$$

Es ergibt sich die Approximationsgüte

$$\frac{\tilde{x}}{x_i} = \frac{f(\tilde{x})}{f(x_{i+1})} = c^{1/(2\cdot\mu-2)}.$$

Die Approximationsgüte ist folglich für alle Intervalle $[x_i, x_{i+1}]$ identisch. Wir können nun Lemma 6.3 anwenden und $X^{\text{APP}}_{\text{opt}}(\mu, f) = X^{\text{HYP}}_{\text{opt}}(\mu, f)$ schließen. □

Abbildung 6.4 veranschaulicht die optimale Verteilung exemplarisch für zwei konkave Fronten. Der letzte Beweis zeigt, dass die Approximationsgüte $c^{1/(2\cdot\mu-2)}$ optimal ist, wenn μ Elemente zur Verfügung stehen.

6.3.3. Referenzpunkt und Randpunkte

Wir untersuchen nun, wie durch die Wahl eines geeigneten Referenzpunktes $r = (r_1, r_2)$ sichergestellt werden kann, dass die Randpunkte $x_{\min}$ und $x_{\max}$ in einer Verteilung, die das Hypervolumen maximiert, enthalten sind. Das nächste Lemma, das in [Auger et al., 2009a] gezeigt wurde, formuliert hinreichende Bedingungen, die sicherstellten, dass eine Verteilung, die das Hypervolumen maximiert, den Randpunkt $x_{\min}$ bzw. $x_{\max}$ enthält. Wir definieren zunächst

$$\mathcal{X}'(\mu, f) := \{X \subseteq [x_{\min}, x_{\max}] \mid |X| = \mu\},$$

um die Notation zu vereinfachen.

Lemma 6.8 (Hinreichende Bedingung für Randpunkte)**.** *Seien $f\colon [x_{\min}, x_{\max}] \to \mathbb{R}$ eine Paretofront und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Betrachte eine Verteilung $X = \{x_1, \ldots, x_\mu\}$ mit $x_i < x_{i+1}$ für alle $1 \leq i < \mu$. Wenn $X = \operatorname{argmax}_{X \in \mathcal{X}'(\mu,f)} \mathrm{I}_{\text{hyp}}(X)$ gilt, dann gelten*

1. *$x_1 = x_{\min}$ falls $r_1 < (f(x) - f(x_{\max}))/f'(x) + x$ für alle $x \in (x_{\min}, x_{\max})$, und*
2. *$x_\mu = x_{\max}$ falls $r_2 < f(x) + (x - x_{\min}) \cdot f'(x)$ für alle $x \in (x_{\min}, x_{\max})$.*

Beweis. Der Beitrag von x_1 zum Hypervolumen beträgt

$$H_1(x_1) := (x_1 - r_1) \cdot (f(x_1) - f(x_2)).$$

Es gilt

$$\begin{aligned} H_1'(x) &= 1 \cdot (f(x) - f(x_2)) + (x - r_1) \cdot f'(x) \\ &= f(x) - f(x_2) + x \cdot f'(x) - r_1 \cdot f'(x) \\ &\leq f(x) - f(x_{\max}) + x \cdot f'(x) - r_1 \cdot f'(x). \end{aligned}$$

Daher gelten $H_1'(x) < 0$ und folglich $x_1 = x_{\min}$, wenn $r_1 < (f(x) - f(x_{\max}))/f'(x) + x$ für alle $x \in (x_{\min}, x_{\max})$ gilt.

Der Beitrag von x_μ zum Hypervolumen beträgt

$$H_\mu(x_\mu) := (x_\mu - x_{\mu-1}) \cdot (f(x_\mu) - r_2).$$

Es gilt

$$\begin{aligned} H_\mu'(x) &= 1 \cdot (f(x) - r_2) + (x - x_{\mu-1}) \cdot f'(x) \\ &= f(x) + (x - x_{\mu-1}) \cdot f'(x) - r_2 \\ &\geq f(x) + (x - x_{\min}) \cdot f'(x) - r_2. \end{aligned}$$

Daher gelten $H_1'(x) > 0$ und folglich $x_\mu = x_{\max}$, wenn $r_2 < f(x) + (x - x_{\min}) \cdot f'(x)$ für alle $x \in (x_{\min}, x_{\max})$ gilt. □

Wir benutzen das letzte Lemma, um auszurechnen, wie der Referenzpunkt konkret gewählt werden kann, um sicherzustellen, dass die Randpunkte der Funktionen f aus $\mathcal{F}_1$ und $\mathcal{F}_2$ in der Hypervolumenverteilung $X_{\text{opt}}^{\text{HYP}}(\mu, f)$ enthalten sind. Wir betrachten zunächst lineare Paretofronten (siehe Definition 6.4).

Theorem 6.9. *Seien $f \in \mathcal{F}_1$ und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Dann gilt*

$$\{x_{\min}, x_{\max}\} \subseteq \operatorname*{argmax}_{X \in \mathcal{X}'(\mu, f)} \mathrm{I}_{\text{hyp}}(X)$$

wenn $r_1 \leq 2 - 1/c + d/c$ und $r_2 \leq 2 - c - d$ gelten, wobei $f\colon [1, (1-d)/c] \to [1, c+d]$ durch $f(x) = c \cdot x + d$ mit $c < 0$ und $d > 1 - c$ definiert ist.

Beweis. Es gelten

$$\begin{aligned} (f(x) - f(x_{\max}))/f'(x) + x &= (c \cdot x + d - (c \cdot x_{\max} + d))/c + x \\ &= 2 \cdot x - x_{\max} \\ &> 2 \cdot 1 - (1 - d)/c = 2 - 1/c + d/c \end{aligned}$$

und

$$\begin{aligned} f(x) + (x - x_{\min}) \cdot f'(x) &= c \cdot x + d + (x - x_{\min}) \cdot c \\ &= 2 \cdot c \cdot x + d - c \cdot x_{\min} \\ &> 2 \cdot c \cdot (1 - d)/c + d - c \cdot 1 = 2 - c - d \end{aligned}$$

für alle $x \in (x_{\min}, x_{\max})$. Also sind die Randpunkte laut Lemma 6.8 in der betrachteten Verteilung enthalten, wenn $r_1 \leq 2 - 1/c + d/c$ und $r_2 \leq 2 - c - d$ gelten. □

Wenn wir die Paretofront $3 - x$ betrachten (siehe Abbildung 6.3), dann können wir beispielsweise durch die Wahl des Referenzpunktes $r = (0, 0)$ sicherstellen, dass die Randpunkte $x_{\min} = 1$ und $x_{\max} = 2$ in der Verteilung, die das Hypervolumen maximiert, enthalten sind.

Wir betrachten nun konkave Paretofronten (siehe Definition 6.6).

Theorem 6.10. *Seien $f \in \mathcal{F}_2$ und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$. Dann gilt*

$$\{x_{\min}, x_{\max}\} \subseteq \operatorname*{argmax}_{X \in \mathcal{X}'(\mu,f)} \mathrm{I}_{\mathrm{hyp}}(X)$$

wenn $r_1 \leq 1/c$ und $r_2 \leq 1/c$ gelten, wobei $f\colon [1, c] \to [1, c]$ durch $f(x) = c/x$ mit $c > 1$ definiert ist.

Beweis. Es gelten

$$\begin{aligned}(f(x) - f(x_{\max}))/f'(x) + x &= (c/x - c/x_{\max})/(-c/x^2) + x \\ &= x^2/x_{\max} \\ &> 1^2/c = 1/c\end{aligned}$$

und

$$\begin{aligned}f(x) + (x - x_{\min}) \cdot f'(x) &= c/x + (x - x_{\min}) \cdot (-c/x^2) \\ &= c \cdot x_{\min}/x^2 \\ &> c \cdot 1/c^2 = 1/c\end{aligned}$$

für alle $x \in (x_{\min}, x_{\max})$. Also sind die Randpunkte laut Lemma 6.8 in der betrachteten Verteilung enthalten, wenn $r_1 \leq 1/c$ und $r_2 \leq 1/c$ gelten. □

Wenn wir die Paretofront $2/x$ betrachten (siehe Abbildung 6.4(a)), dann können wir beispielsweise durch die Wahl des Referenzpunktes $r = (1/2, 1/2)$ sicherstellen, dass die Randpunkte $x_{\min} = 1$ und $x_{\max} = 2$ in der Verteilung, die das Hypervolumen maximiert, enthalten sind.

6.4. Numerische Untersuchungen

6.4.1. Ansatz

Wir konnten im letzten Abschnitt die Hypervolumen- und Approximationsverteilungen für zwei relativ einfache Klassen von Fronten analytisch ermitteln. Dieser Ansatz wird für komplexere Fronten relativ schnell sehr kompliziert wenn nicht sogar unmöglich. Die numerische Lösung der entstehenden Gleichungssysteme stellt einen gangbaren Weg aus diesem Dilemma dar. Der vorliegende Abschnitt widmet sich daher der numerischen Analyse einer komplexeren Klasse von Fronten.

Wir untersuchen hier Fronten, deren Form durch x^p, $p > 0$, gegeben ist. Um eine Skalierung in beiden Dimensionen zu ermöglichen, untersuchen wir die folgende Funktionenklasse.

Definition 6.11. *Definiere die Funktionenklasse*

$$\mathcal{F}_3 := \left\{ f_p \colon [x_1, x_\mu] \to [y_\mu, y_1] \text{ mit } f(x) = y_\mu - (y_\mu - y_1) \cdot \left(1 - \left(\frac{x - x_1}{x_\mu - x_1}\right)^p\right)^{1/p} \right\},$$

wobei $p > 0$, $x_\mu > x_1 \geq 1$ *und* $y_1 > y_\mu \geq 1$ *frei gewählt werden können.*

Wir benutzen in diesem Abschnitt zusätzlich die Bezeichnung $y_i := f(x_i)$, um die Notation zu erleichtern. Wir werden uns hauptsächlich auf zwei Funktionen aus $\mathcal{F}_3$ konzentrieren. Die *symmetrische Front* $f_p^{\text{sym}} \colon [1, 2] \to [1, 2]$ und die *asymmetrische Front* $f_p^{\text{asy}} \colon [1, 201] \to [1, 2]$, wobei wir natürlich die Krümmung p variieren. Die hier untersuchte Funktionenklasse $\mathcal{F}_3$ subsumiert eine große Zahl von weitverbreiteten Beispielfunktionen. Die DTLZ1–7 genannten Probleme aus [Deb et al., 2002, 2005] werden beispielsweise häufig herangezogen, um die Leistungsfähigkeit von Algorithmen miteinander zu vergleichen. Die Probleme DTLZ1–4 sind z. B. in der Funktionenklasse $\mathcal{F}_3$ aus Definition 6.11 enthalten. Für $p = 1$ entsprechen die betrachteten Fronten der Beispielfunktion DTLZ1; für $p = 2$ ähneln die Fronten den Beispielfunktionen DTLZ2–4.

Wir bestimmen in diesem Abschnitt für verschiedene Funktionen f_p und $\mu \in \mathbb{N}^+$ mit $\mu \geq 2$ die Verteilungen $X_{\text{opt}}^{\text{HYP}}(\mu, f_p)$ und $X_{\text{opt}}^{\text{APP}}(\mu, f_p)$. Im ersten Fall bestimmen wir x_i, $1 < i < \mu$ gemäß

$$\underset{x_2,\ldots,x_{\mu-1}}{\text{argmax}} \left(\sum_{i=2}^{\mu-1} (x_i - x_{i-1}) \cdot (f(x_i) - f(x_\mu)) \right).$$

Im zweiten Fall bestimmen wir x_i, $1 < i < \mu$, und y_i, $1 \leq i < \mu$, gemäß

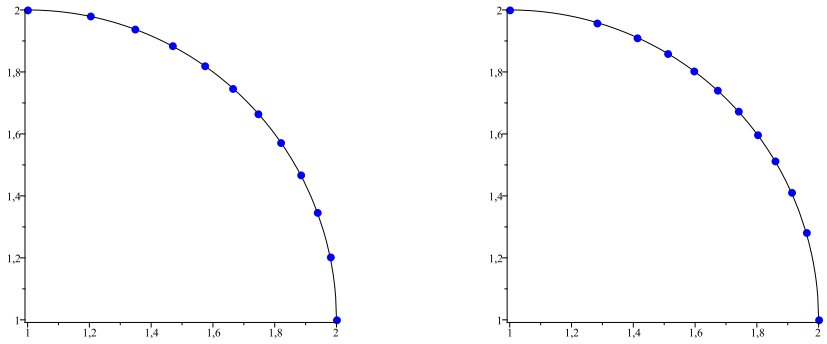
$$\frac{y_1}{x_1} = \frac{y_2}{x_2} = \cdots = \frac{y_{\mu-1}}{x_{\mu-1}} = \frac{f(y_1)}{f(x_2)} = \frac{f(y_2)}{f(x_3)} = \cdots = \frac{f(y_{\mu-1})}{f(x_\mu)}$$

(siehe Lemma 6.3). Die Berechnung der Lösungen erfolgt hier mithilfe numerischer Standardmethoden (siehe z. B. [Fletcher, 1987]). Hier wurde auf die Software „Maple“ von „Maplesoft“ zurückgegriffen, um die genannten Gleichungssysteme auf numerische Weise zu lösen [Maplesoft, 2009].

6.4.2. Ergebnisse

Im Folgenden präsentieren wir die Ergebnisse der numerischen Untersuchungen. Wir untersuchen zunächst den Fall $p = 2$. Abbildung 6.5 und 6.6 zeigen verschiedene Verteilungen auf f_2. Es zeigt sich, dass sich die Hypervolumen- und die Approximationsverteilung unterscheiden. Abbildung 6.5(a) und 6.5(b) zeigen die Verteilungen auf der symmetrischen Front $f_2^{\text{sym}} \colon [1, 2] \to [1, 2]$ mit

$$f_2^{\text{sym}}(x) = 1 + \sqrt{1 - (x - 1)^2}.$$



(a) $X_{\text{opt}}^{\text{HYP}}(12, f_2^{\text{sym}})$. (b) $X_{\text{opt}}^{\text{APP}}(12, f_2^{\text{sym}})$.

Abbildung 6.5.: Hypervolumen- und Approximationsverteilung auf der symmetrischen Front f_2^{sym}. Die Verteilungen unterscheiden sich hier. Die Approximationsgüte der Hypervolumenverteilung beträgt ungefähr $\mathrm{I}_{\text{mul}}(X_{\text{opt}}^{\text{HYP}}(12, f_2^{\text{sym}})) \approx 1{,}025$ und ist damit um 0,457% größer als die der Approximationsverteilung $\mathrm{I}_{\text{mul}}(X_{\text{opt}}^{\text{APP}}(12, f_2^{\text{sym}})) \approx 1{,}021$.

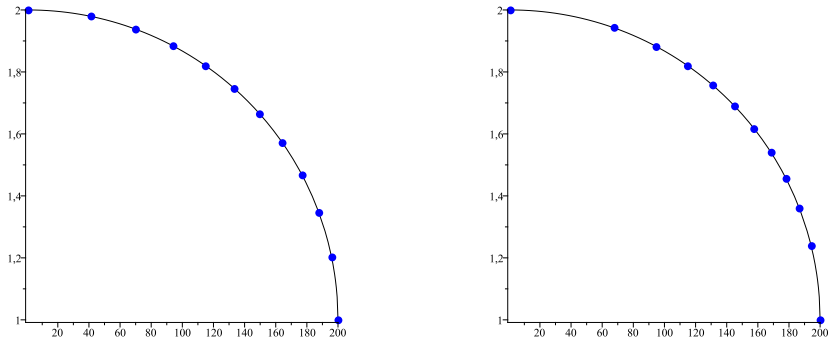
Abbildung 6.6(a) und 6.6(b) zeigen die Verteilungen auf der asymmetrischen Front $f_2^{\text{sym}} \colon [1, 201] \to [1, 2]$ mit

$$f_2^{\text{sym}}(x) = 1 + \sqrt{1 - (x/200 - 1/200)^2}.$$

Es fällt auf, dass die relativen Positionen der Punkte der Hypervolumenverteilung in Abbildung 6.5(a) und 6.6(a) gleich bleiben, während sich die relativen Positionen der Punkte der Approximationsverteilung in Abbildung 6.5(b) und 6.6(b) mit der Skalierung verändern. Folglich scheinen die relativen Positionen der Punkte der Hypervolumenverteilung skalierungsinvariant zu sein. Da die Positionen der Punkte der Approximationsverteilung natürlich von der Skalierung abhängen, kann die Hypervolumenverteilung nicht die beste Approximationsgüte liefern.

Für die in Abbildung 6.5 und 6.6 gezeigten Beispiele beträgt die Approximationsgüte der Approximationsverteilung für den symmetrischen und den asymmetrischen Fall 1,021 (Abbildung 6.5(b)) bzw. 1,030 (Abbildung 6.6(b)) während die Approximationsgüte der Hypervolumenverteilung 1,025 (Abbildung 6.5(a)) bzw. 1,038 (Abbildung 6.6(a)) beträgt. Folglich führt die Maximierung des Hypervolumens in den betrachteten Fällen nicht zu einer minimalen Approximationsgüte. Die Approximationsgüte der Hypervolumenverteilung ist also um 0,457% bzw. 0,839% schlechter als die Approximationsgüte der Approximationsverteilung.

Wir haben bereits gesehen, dass die Skalierung der Front einen großen Einfluss auf die Approximationsverteilung, nicht aber auf die Hypervolumenverteilung hat. Wir untersuchen diesen Einfluss nun genauer. Der Einfluss des Skalierungsparameters $x_\mu \geq 2$ auf verschiedene Fronten $f_p \colon [1, x_\mu] \to [1, 2]$ wird in Abbildung 6.7 für $p \in \{1/3, 1/2, 1, 2, 3\}$ dargestellt. In den Abbildungen wird für $\mu = 3$ die erreichte Approximationsgüte in Abhängigkeit von x_μ gezeigt. Erwartungsgemäß vergrößern



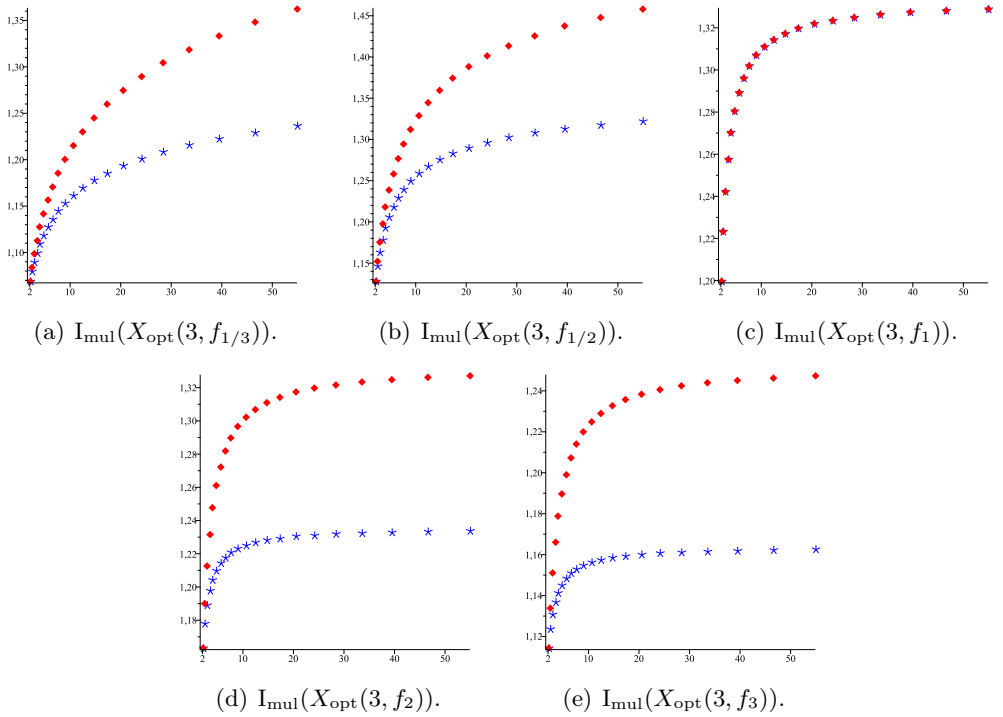
(a) $X_{\text{opt}}^{\text{HYP}}(12, f_2^{\text{asy}})$. (b) $X_{\text{opt}}^{\text{APP}}(12, f_2^{\text{asy}})$.

Abbildung 6.6.: Hypervolumen- und Approximationsverteilung auf der asymmetrischen Front f_2^{asy}. Die Verteilungen unterscheiden sich hier. Die Approximationsgüte der Hypervolumenverteilung beträgt ungefähr $\mathrm{I}_{\text{mul}}(X_{\text{opt}}^{\text{HYP}}(12, f_2^{\text{asy}})) \approx 1{,}038$ und ist damit um 0,839% größer als die der Approximationsverteilung $\mathrm{I}_{\text{mul}}(X_{\text{opt}}^{\text{APP}}(12, f_2^{\text{asy}})) \approx 1{,}030$.

sich die Approximationsgüten mit wachsender Asymmetrie. Für konvexe Fronten ($p > 1$) scheinen die Approximationsgüten mit wachsendem x_μ zu konvergieren. Die Approximationsgüten für f_2 erwecken den Eindruck, für die Approximationsverteilung gegen den goldenen Schnitt $\sqrt{5}-1 \approx 1{,}236$ und für die Hypervolumenverteilung gegen $4/3 \approx 1{,}333$ zu konvergieren. Für f_3 scheinen sie gegen 1,164 bzw. 1,253 zu konvergieren. Daher ist für f_2 und f_3 die Approximationsgüte der Hypervolumenverteilung höchstens 8% schlechter als die der Approximationsverteilung. Dieses Verhalten scheint für konkave Fronten ($p < 1$) grundlegend anders zu sein. Dort hat es den Anschein, dass das Verhältnis zwischen der Approximationsgüte der Hypervolumen- und der Approximationsverteilung divergiert.

Eine weitere wichtige Frage betrifft den Einfluss der Wahl der Populationsgröße auf das Verhältnis zwischen den Approximationsgüten der Hypervolumen- und der Approximationsverteilung. Wir untersuchen nun den Einfluss der Wahl von μ auf die Approximationsgüte genauer. Abbildung 6.8 zeigt die erreichten Approximationsgüten in Abhängigkeit von μ. Für symmetrische Fronten f_p mit $(x_1, y_1) = (y_\mu, x_\mu)$ und $\mu = 3$ ist die Approximationsgüte der Hypervolumenverteilung für alle $p > 1$ optimal. Außerdem ist die Approximationsgüte der Hypervolumenverteilung für alle linearen Fronten f_1 unabhängig von (x_1, y_1) und (y_μ, x_μ) optimal.

Für größere Populationen verringern sich die Approximationsgüten der Hypervolumen- und der Approximationsverteilungen. Nichtsdestotrotz ist die Approximationsgüte der Hypervolumenverteilung auf konkaven und asymmetrischen Fronten f_p^{asy} mit $p < 1$ auch für größere μ sehr schlecht (siehe beispielsweise Abbildung 6.8(f) und 6.8(g)). Kürzlich haben Bringmann und Friedrich [2010] die Approximationsgüte der Approximations- wie auch der Hypervolumenverteilung durch $1 + \Theta(1/\mu)$ begrenzt. Das bedeutet, dass für $\mu \to \infty$ die Approximationsgüten von $X_{\text{opt}}^{\text{HYP}}(\mu, f)$ und



(a) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(3, f_{1/3}))$. (b) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(3, f_{1/2}))$. (c) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(3, f_1))$.

(d) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(3, f_2))$. (e) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(3, f_3))$.

Abbildung 6.7.: Approximationsgüte der Hypervolumenverteilung (◆) und der Approximationsverteilung (⋆) auf diversen Fronten $f_p\colon [1, x_\mu] \to [1, 2]$ in Abhängigkeit von x_μ. Wie in Theorem 6.5 bewiesen, stimmen beide Kurven in (c) für lineare Fronten f_1 unabhängig von x_μ überein.

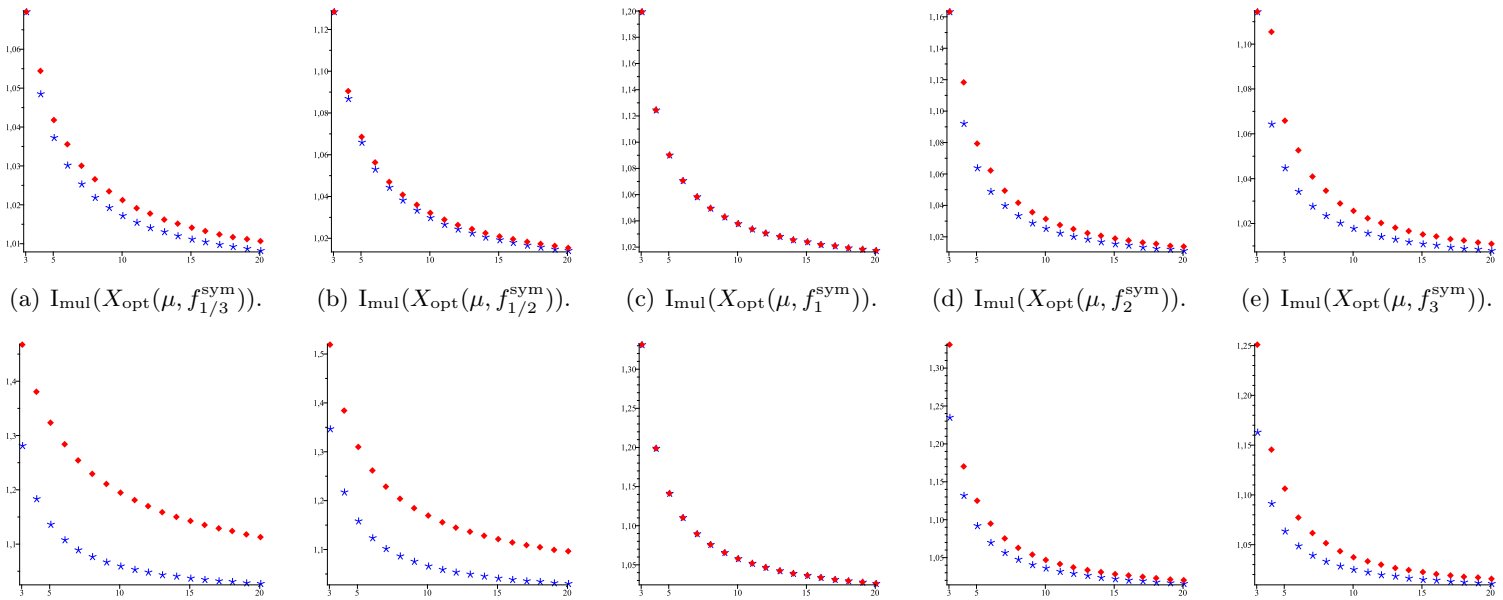
$X_{\mathrm{opt}}^{\mathrm{APP}}(\mu, f)$ gleichermaßen schnell gegen die optimale Approximationsgüte 1 konvergieren. Die genannten Ergebnisse schließen natürlich nicht aus, dass sich die Güten für hinreichend kleine Populationsgrößen μ deutlich unterscheiden können.

Die Untersuchungen zeigen, dass sich die Approximationsgüten der Hypervolumen- und der Approximationsverteilung in Abhängigkeit von p deutlich unterscheiden können. Die Krümmung der Front scheint also einen wesentlichen Einfluss auszuüben (siehe auch [Lizzáraga-Lizzáraga et al., 2009]). Es wurde vermutet, dass das Hypervolumen konvexe Regionen gegenüber konkaven Regionen der Front bevorzugt [Zitzler und Thiele, 1998b]. Neuere Untersuchungen von Auger et al. [2009a] zeigen, dass die Verteilung ausschließlich von der Steigung der Front beeinflusst wird und nicht davon abhängt, ob die Front konvex oder konkav ist. Um die Auswirkungen einer konvexen bzw. konkaven Front weiter zu beleuchten, zeigt Abbildung 6.9 die Approximationsgüte in Abhängigkeit von p. Erwartungsgemäß ist die Approximationsgüte der Hypervolumenverteilung für $p = 1$ optimal (siehe Theorem 6.5). Der Einfluss von p ist für symmetrische und asymmetrische Fronten sehr unterschiedlich. Für f_p^{sym} werden die konkaven Fronten ($p < 1$) durch die Hypervolumenverteilung sehr viel besser approximiert als die konvexen Fronten ($p > 1$) (siehe Ab-

bildung 6.9(a)–(d)). Für f_p^{asy} ist das Verhalten überraschenderweise gegensätzlich (siehe Abbildung 6.9(e)–(h)).

6.5. Fazit

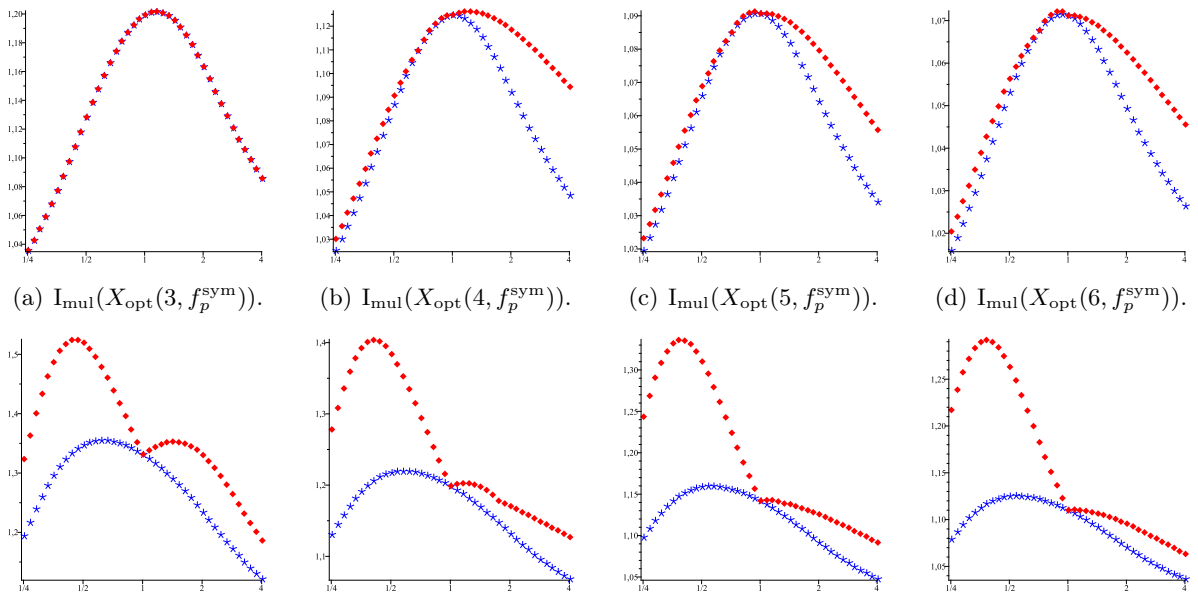
Die Verwendung des Hypervolumens, um die Qualität einer Population in einem evolutionären Algorithmus für multikriterielle Optimierungsprobleme zu bewerten, erfreut sich in den letzten Jahren großer Beliebtheit. Es ist schwierig, zu verstehen, wie die μ Individuen in der Population verteilt werden müssen, um das Hypervolumen zu maximieren. Das Optimierungsziel von evolutionären Algorithmen, die diesen Qualitätsindikator verwenden, ist folglich noch weitgehend unerforscht. Demzufolge trägt ein verbessertes Verständnis der Hypervolumenverteilung unmittelbar dazu bei, das Optimierungsziel vieler gängiger evolutionärer Algorithmen besser zu verstehen. Wir haben untersucht, wie die Individuen in der Population durch die Maximierung des Hypervolumens auf verschiedenartigen Fronten verteilt werden und die erreichte Approximationsgüte zu der Approximationsgüte, die durch eine optimale Verteilung der Individuen erreicht werden kann, in Beziehung gesetzt. Wir haben zunächst relativ einfache lineare und konkave Fronten studiert. Für diese Fronten konnten wir beweisen, dass die Maximierung des Hypervolumens automatisch zu der Minimierung der Approximationsgüte führt. Anschließend haben wir eine Klasse von Fronten betrachtet, deren Krümmungen und Skalierungen variiert werden können. Wir haben mithilfe numerischer Verfahren gezeigt, dass sowohl die Krümmung als auch die Skalierung der untersuchten Front enormen Einfluss auf die Approximationsgüte der Hypervolumenverteilung ausüben. Insbesondere haben wir für verschiedene Fronten dargestellt, dass die Approximationsgüte der Hypervolumenverteilung schlechter als die Approximationsgüte der Approximationsverteilung ist. Das Optimierungsziel, das Hypervolumen zu maximieren, kann sich also ganz wesentlich von dem Optimierungsziel, die Approximationsgüte zu minimieren, unterscheiden. Das letztgenannte Optimierungsziel ist gutfundiert und weitverbreitet. Wenn man einen Algorithmus einsetzen möchte, der das Hypervolumen maximiert, dann erscheint es vor diesem Hintergrund sinnvoll, sich zu vergewissern, dass das angestrebte Optimierungsziel die Wünsche der Entscheidungsinstanz respektiert.



(a) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{1/3}^{\mathrm{sym}}))$. (b) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{1/2}^{\mathrm{sym}}))$. (c) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{1}^{\mathrm{sym}}))$. (d) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{2}^{\mathrm{sym}}))$. (e) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{3}^{\mathrm{sym}}))$.

(f) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{1/3}^{\mathrm{asy}}))$. (g) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{1/2}^{\mathrm{asy}}))$. (h) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{1}^{\mathrm{asy}}))$. (i) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{2}^{\mathrm{asy}}))$. (j) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(\mu, f_{3}^{\mathrm{asy}}))$.

Abbildung 6.8.: Approximationsgüte der Hypervolumenverteilung (◆) und der Approximationsverteilung (⋆) auf diversen symmetrischen und asymmetrischen Fronten f_p in Abhängigkeit von μ. Wie in Theorem 6.5 bewiesen, stimmen beide Kurven in (c) und (h) für lineare Fronten f_1 unabhängig von μ überein. Weiterhin fällt auf, dass die Verteilungen für alle f_p^{sym} und $\mu = 3$ übereinstimmen (siehe (a)–(e)).



(a) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(3, f_p^{\mathrm{sym}}))$. (b) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(4, f_p^{\mathrm{sym}}))$. (c) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(5, f_p^{\mathrm{sym}}))$. (d) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(6, f_p^{\mathrm{sym}}))$.

(e) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(3, f_p^{\mathrm{asy}}))$. (f) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(4, f_p^{\mathrm{asy}}))$. (g) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(5, f_p^{\mathrm{asy}}))$. (h) $\mathrm{I}_{\mathrm{mul}}(X_{\mathrm{opt}}(6, f_p^{\mathrm{asy}}))$.

Abbildung 6.9.: Approximationsgüte der Hypervolumenverteilung (◆) und der Approximationsverteilung ($\star$) auf diversen symmetrischen und asymmetrischen Fronten f_p in Abhängigkeit von p. Es fällt auf, dass die Verteilungen für alle f_p^{sym} und $\mu = 3$ übereinstimmen (siehe (a)).

Teil II.

Evolutionäre Algorithmen für kombinatorische Optimierung

7. Ein evolutionärer Algorithmus für ein multikriterielles Kürzeste-Wege-Problem

7.1. Einleitung

In den letzten zehn Jahren wurde erheblicher Aufwand in die Analyse der Optimierungszeiten von evolutionären Algorithmen (EAs) investiert. Diese Resultate ermöglichen ein fundiertes Verständnis der Arbeitsweisen dieser Algorithmen. Eine der ersten Arbeiten, die dieser Forschungsrichtung zugeordnet werden können, untersucht die Laufzeit eines einfachen evolutionären Algorithmus, der (1+1)-EA genannt wird, auf einfachen pseudobooleschen Beispielfunktionen [Droste et al., 2002]. Später wurden ähnliche Untersuchungen auch für grundlegende evolutionäre Algorithmen für multikriterielle Optimierungsprobleme durchgeführt [Laumanns et al., 2004, Giel, 2003]. Die ersten Laufzeitanalysen haben vornehmlich künstliche Beispielfunktionen untersucht. In den letzten Jahren hat die Laufzeitanalyse von evolutionären Algorithmen für klassische algorithmische Probleme mehr und mehr Aufmerksamkeit auf sich gezogen, da evolutionäre Algorithmen häufig benutzt werden, um kombinatorische Optimierungsprobleme zu lösen.

Es gibt mittlerweile eine ganze Reihe von Artikeln, die zeigen, dass evolutionäre Algorithmen in der Lage sind, einfache kombinatorische Optimierungsprobleme effizient zu lösen, wobei „einfach“ hier bedeutet, dass es problemspezifische Algorithmen gibt, die die Probleme in Polynomialzeit lösen. Eine der ersten Arbeiten, die sich diesem Thema widmet, ist [Scharnow et al., 2004]. Die Autoren analysieren darin die Optimierungszeit von evolutionären Algorithmen für das Sortieren und das Kürzeste-Wege-Problem mit einer Quelle. Die Arbeit [Giel und Wegener, 2003] nimmt sich dem Problem an, eine größte Paarung (auf Englisch „maximum matching“) in einem ungerichteten Graphen zu finden. Weiterhin gibt es eine Reihe von Arbeiten, die evolutionäre Algorithmen für die graphentheoretischen Probleme, einen Eulerkreis [Neumann, 2008, Doerr et al., 2007b,c, Doerr und Johannsen, 2007] oder einen minimalen aufspannenden Baum [Neumann und Wegener, 2007, 2006] zu finden, analysieren.

Evolutionäre Algorithmen werden in der Praxis häufig eingesetzt, um gute Lösungen für NP-schwierige Probleme zu finden. Wenn man die recht gut fundierte These $\mathsf{RP} \neq \mathsf{NP}$ akzeptiert, dann folgt, dass ein evolutionärer Algorithmus nicht alle Eingabeinstanzen in Polynomialzeit *exakt* lösen kann. Die Komplexitätsklasse RP umfasst dabei alle Probleme, die von einem effizienten Algorithmus gelöst werden

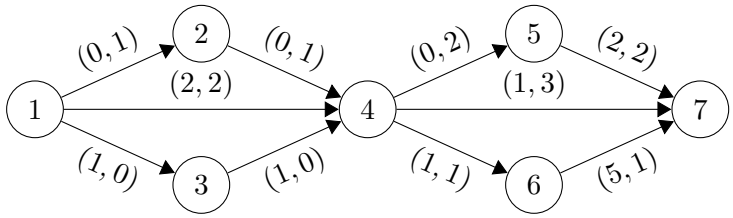


Abbildung 7.1.: Beispiel eines gewichteten gerichteten Graphen mit 2 Gewichten pro Kante.

Pfad	Gewicht	Pfad	Gewicht
$1 \to 2 \to 4 \to 5 \to 7$	$(2, 6)$	$1 \to 4 \to 6 \to 7$	$(8, 4)$
$\mathbf{1 \to 2 \to 4 \to 7}$	$\mathbf{(1, 5)}$	$1 \to 3 \to 4 \to 5 \to 7$	$(4, 4)$
$1 \to 2 \to 4 \to 6 \to 7$	$(6, 4)$	$\mathbf{1 \to 3 \to 4 \to 7}$	$\mathbf{(3, 3)}$
$1 \to 4 \to 5 \to 7$	$(4, 6)$	$\mathbf{1 \to 3 \to 4 \to 6 \to 7}$	$\mathbf{(8, 2)}$
$1 \to 4 \to 7$	$(3, 5)$		

Tabelle 7.1.: Übersicht über alle Pfade von Knoten 1 nach Knoten 7 mit den zugehörigen Gewichten. Die *Pareto-optimalen* bzw. *effizienten* Pfade sind fett gedruckt.

können, der einzig zu akzeptierende Eingaben fälschlicherweise mit Wahrscheinlichkeit höchstens 1/2 verwerfen darf. Die Annahme, dass ein evolutionärer Algorithmus nach polynomiell vielen Generationen für alle Eingabeinstanzen mit Wahrscheinlichkeit mindestens $1/\operatorname{poly}(n)$ entscheiden kann, ob ein Zielpunkt von der Paretofront dominiert wird oder nicht, würde die Ungleichung folglich zusammenbrechen lassen. Es ist daher sehr relevant, das Wissen zu vergrößern, wie lange diese Algorithmen benötigen, um die genannten Probleme approximativ zu lösen. Es gibt bereits einige Ergebnisse, die einfachen evolutionären Algorithmen die Eigenschaft attestieren, Lösungen für NP-schwierige kombinatorische Optimierungsprobleme effizient zu finden, deren Güte nur um einen konstanten Faktor schlechter als die einer optimalen Lösung ist. Derartige Ergebnisse werden beispielsweise in [Neumann, 2007] für das Problem, in einem ungerichteten Graphen mit zwei Gewichten pro Kante einen minimalen aufspannenden Baum zu finden, in [Witt, 2005] für das Partitionierungsproblem und in [Neumann und Reichel, 2008] für das Problem, in einem ungerichteten Graphen mit mehreren Quelle-Senke-Paaren einen minimalen Schnitt zu finden, präsentiert. Diese Ergebnisse demonstrieren auf eindrucksvolle Weise die Befähigung evolutionärer Algorithmen, gute approximative Lösungen schnell zu finden. Der Wert eines solchen evolutionären Algorithmus würde noch zunehmen, wenn die gewünschte Approximationsgüte vor Beginn der Optimierung eingestellt werden könnte. Wir verweisen auf die umfangreiche Arbeit von Oliveto et al. [2007a], die weitere Informationen über den hier betrachteten Themenkomplex bereithält.

Dieses Kapitel widmet sich dem sehr bekannten multikriteriellen Kürzeste-Wege-

Problem, das darin besteht, in einem gegebenen gewichteten gerichteten Graphen mit $k \geq 1$ Gewichtsfunktionen alle kürzesten Wege zu finden. Abbildung 7.1 zeigt eine mögliche Eingabeinstanz für das betrachtete Problem und Tabelle 7.1 führt alle Pfade von Knoten 1 nach Knoten 7 auf. Der Tabelle kann auch entnommen werden, welche Pfade Pareto-optimal sind. Bevor wir uns evolutionären Algorithmen zuwenden, fassen wir ein paar allgemeine das Problem betreffende Ergebnisse zusammen. Das Problem ist bekanntermaßen NP-schwierig, wenn wir es mit zwei oder mehr Gewichten pro Kante zu tun haben [Garey und Johnson, 1979]. Weiterhin gibt es Eingabeinstanzen, die bezüglich der Eingabegröße exponentiell viele Pareto-optimale Pfade aufweisen [Ehrgott, 2005]. Der Begriff „Eingabegröße“ bezieht sich hier auf die Anzahl von Bits, die benötigt werden, um den Graphen und die Kantengewichte auf natürliche Weise binär zu kodieren. Folglich stellen approximative Lösungen den einzigen Ausweg dar, wenn es darum geht, auch die oben genannten Probleminstanzen effizient zu behandeln. Solche Lösungen enthalten nicht alle Pareto-optimalen Pfade sondern eine Auswahl von Pfaden, die alle Bereiche der Paretofront widerspiegelt. Wir weisen darauf hin, dass im bikriteriellen Fall viele Probleminstanzen auf effiziente Weise exakt gelöst werden können [Müller-Hannemann und Weihe, 2006, Raith und Ehrgott, 2009]. Ein weiterer Vorteil approximativer Lösungen besteht darin, dass es für die Entscheidungsinstanz einfacher ist, die gemäß ihrer persönlichen Präferenzen geeignetste Lösung aus einer kleinen Anzahl von Lösungen, die die Paretofront repräsentieren, zu wählen, als alle Pareto-optimalen Lösungen durchzusehen.

Das erste echt polynomielle Approximationsschema (FPTAS) für den bikriteriellen Fall wurde von Hansen [1979] veröffentlicht. Wenn wir unsere Betrachtungen auf gerichtete azyklische Graphen begrenzen, dann wird in [Warburton, 1987] ein FPTAS für den allgemeinen multikriteriellen Fall vorgestellt. Papadimitriou und Yannakakis [2000] zeigen, dass die Paretofront jedes multikriteriellen Optimierungsproblems von einer Lösungsmenge polynomieller Größe approximiert werden kann. Weiterhin zeigen die letztgenannten Autoren, wie für verschiedene multikriterielle Optimierungsprobleme (inklusive dem Kürzeste-Wege-Problem) ein FPTAS konstruiert werden kann. Ein FPTAS mit einer verbesserten Laufzeit für das multikriterielle Kürzeste-Wege-Problem mit einer Quelle wird von Tsaggouris und Zaroliagis [2006] präsentiert. Der letztgenannte Artikel hat auch dazu beigetragen, die Optimierungszeit eines einfachen evolutionären Algorithmus für dieses Problem zu untersuchen, da die genannten Algorithmen das FPTAS, das dynamische Programmierung nutzt und mit skalierten und gerundeten Gewichtswerten arbeitet, „simulieren“ können.

Alle bisherigen Arbeiten, die die Optimierungszeiten von evolutionären Algorithmen für das Kürzeste-Wege-Problem analysieren, konzentrieren sich auf den monokriteriellen Fall. Die bereits weiter oben erwähnte Arbeit von Scharnow et al. [2004] analysiert die Optimierungszeit eines einfachen evolutionären Algorithmus für die Problemvariante mit einer Quelle. Doerr et al. [2007a] haben diese Analyse verfeinert, indem sie die Eingabeinstanzen gemäß ℓ klassifizieren, wobei ℓ die kleinste ganze Zahl ist, sodass jeder Knoten von der Quelle einem kürzesten Pfad mit höchstens ℓ Kanten folgend erreicht werden kann. Die Variante, bei der kürzeste Wege von jedem Knoten zu jedem anderen Knoten gesucht sind, wird in [Doerr et al., 2008a] als

natürliches Beispiel präsentiert, bei dem die Verwendung eines geeigneten Rekombinationsoperators die Optimierungszeit eines evolutionären Algorithmus beweisbar verbessert.

In diesem Kapitel präsentieren wir die erste Analyse der Optimierungszeit eines einfachen evolutionären Algorithmus für das multikriterielle Kürzeste-Wege-Problem. Wir zeigen, dass die Optimierungszeit des untersuchten evolutionären Algorithmus mit der Laufzeit des schnellsten bekannten problemspezifischen Algorithmus aus einer Arbeit von Tsaggouris und Zaroliagis [2006] vergleichbar ist.

Wir beenden diese Einleitung mit einem kurzen Überblick über das vorliegende Kapitel. In Abschnitt 7.2 führen wir das multikriterielle Kürzeste-Wege-Problem ein. Die vektorwertige Fitnessfunktion und der grundlegende evolutionäre Algorithmus, den wir in diesem Kapitel betrachten, werden in Abschnitt 7.3 zusammen mit einigen Eigenschaften des Algorithmus präsentiert. In Abschnitt 7.4 wenden wir den evolutionären Algorithmus auf die Fitnessfunktion an und bestimmen obere Schranken für die Optimierungszeit. Untere Schranken für die Optimierungszeit präsentieren wir in Abschnitt 7.5 und beenden dieses Kapitel anschließend mit einer Diskussion in Abschnitt 7.6.

7.2. Problemdefinition

Wir definieren in diesem Abschnitt das multikriterielle Kürzeste-Wege-Problem. Das Problem basiert auf einem gewichteten gerichteten Graphen $G = (V, E, w)$, wobei V eine Menge mit n Knoten und $E \subseteq \{(u, v) \in V^2 \mid v \neq u\}$ eine Menge mit m Kanten bezeichnet. Wir gehen davon aus, dass der Graph höchstens eine Kante zwischen zwei verschiedenen Knoten enthält. Die Gewichtsfunktion $w = (w_1, \ldots, w_k)\colon E \to \mathbb{N}^k$ ordnet jeder Kante $k \in \mathbb{N}^+$ nicht negative ganzzahlige Gewichte zu.

Wir führen nun wichtige Sprechweisen ein. Ein *(einfacher) Pfad der Länge* $\ell \in \mathbb{N}$ ist eine Folge $p = (v_0, \ldots, v_\ell)$ von $\ell + 1$ (verschiedenen) Knoten $v_i \in V$, wobei $(v_{i-1}, v_i) \in E$ für alle i, $1 \leq i \leq \ell$, gilt. Die Knoten v_0 und v_ℓ werden *Quelle* bzw. *Senke* genannt. Wir merken an, dass die Länge eines einfachen Pfades durch $n - 1$ begrenzt ist. Das Gewicht $w(p)$ eines Pfades $p = (v_0, \ldots, v_\ell)$ ist durch die Summe der Gewichte der benutzten Kanten gegeben, d. h. $w(p) = \sum_{i=1}^{\ell} w((v_{i-1}, v_i))$. Wir bezeichnen mit $PFAD_\ell(s, t)$ die Menge aller *einfachen* Pfade der Länge ℓ von $s \in V$ nach $t \in V$. Weiterhin schreiben wir $PFAD_{\leq \ell}(s, t) := \bigcup_{0 \leq i \leq \ell} PFAD_i(s, t)$. Um die Notation in den folgenden Abschnitten so einfach wie möglich zu gestalten, definieren wir außerdem $PFAD(s, t) := PFAD_{\leq n-1}(s, t)$.

Das Kürzeste-Wege-Problem besteht darin, die Paretofront der Zielfunktion

$$w\colon PFAD(s, t) \to \mathbb{N}^k$$

zu ermitteln. Wir weisen an dieser Stelle explizit darauf hin, dass sich die Paretofront von w nicht ändert, wenn man die Zielfunktion auch für nicht einfache Pfade definiert. Es gibt prinzipiell vier verschiedene Varianten des Kürzeste-Wege-Problems. Wir können ausschließlich an kürzesten Wegen von einem bestimmten Knoten $s \in V$

zu einem bestimmten Knoten $t \in V$ interessiert sein (eine Quelle und eine Senke). In diesem Fall müssen wir die Paretofront der Zielfunktionen $w\colon PFAD(s,t) \to \mathbb{N}^k$ ermitteln. Alternativ können wir uns für alle kürzeste Wege interessieren (alle Quellen und alle Senken). In diesem Fall müssen wir die Paretofronten der n^2 Zielfunktionen $w\colon PFAD(s,t) \to \mathbb{N}^k$, $s \in V$, $t \in V$, bestimmen. Außerdem können wir an allen kürzesten Wegen mit einer bestimmten Quelle $s \in V$ oder an allen kürzesten Wegen mit einer bestimmten Senke $t \in V$ interessiert sein (eine Quelle und alle Senken bzw. alle Quellen und eine Senke). In diesen Fällen müssen wir die Paretofronten der n Zielfunktionen $w\colon PFAD(s,t) \to \mathbb{N}^k$, $t \in V$, bzw. $w\colon PFAD(s,t) \to \mathbb{N}^k$, $s \in V$, bestimmen. Die letzten beiden Varianten sind ambitionierter als die erste Variante und weniger ambitioniert als die zweite Variante. Außerdem stimmt die Komplexität der letzten beiden Varianten überein, da die eine auf die andere Variante reduziert werden kann, indem man die Richtung der Kanten umdreht. Aus naheliegenden Gründen kürzen wir die vier Varianten mit $(1,1)$, $(1,n)$, $(n,1)$ und (n,n) ab. Wir konzentrieren uns in diesem Kapitel auf die Variante $(1,n)$.

Um uns die Arbeit zu erleichtern, führen wir die folgenden Abkürzungen ein. Wir bezeichnen mit $w_i^{\max} := \max_{e \in E} w_i(e)$ das größte Gewicht bezüglich der i-ten Gewichtsteilfunktion w_i. Weiterhin bezeichnen wir das größte Gewicht mit $w^{\max} := \max_{1 \le i \le k} w_i^{\max}$. Wir nehmen ohne Beschränkung der Allgemeinheit an, dass sowohl $V = \{1, \ldots, n\}$ als auch $s = 1$ gelten, um die folgenden Betrachtungen zu vereinfachen. Diese Annahme kann durch eine einfache Umbenennung der Knoten sichergestellt werden.

7.3. Ein evolutionärer Algorithmus

7.3.1. Fitnessfunktion

Wenn man ein Problem mithilfe eines evolutionären Algorithmus lösen möchten, dann besteht der erste Schritt darin, eine geeignete Repräsentation der Lösungskandidaten zu finden. Wir verwenden den Suchraum

$$PFAD(1,\cdot) := \bigcup_{1 \le t \le n} PFAD(1,t).$$

Wir weisen darauf hin, dass $PFAD(1,1) = \{(1)\}$ gilt, da wir uns auf einfache Pfade beschränken. Wir werden in den folgenden Ausführungen außerdem die Bezeichnungen $PFAD_\ell(1,\cdot)$ und $PFAD_{\le \ell}(1,\cdot)$ benutzen, um die Mengen, die alle einfachen Pfade der Länge genau bzw. höchstens ℓ enthalten, zu benennen. Wir werden in Unterabschnitt 7.4.1 noch thematisieren, warum es sinnvoll ist, den Suchraum auf einfache Pfade einzugrenzen.

Die zweite Aufgabe besteht darin, eine geeignete Fitnessfunktion zu definieren. Wir betrachten zwei beliebige Individuen $p = (v_0, \ldots, v_\ell)$ und $p' = (v'_0, \ldots, v'_{\ell'})$. Da wir an kürzesten Wegen von dem Knoten 1 zu allen Knoten t, $1 \le t \le n$, interessiert sind, muss die Fitnessfunktion sicherstellen, dass $f(p)$ und $f(p')$ unvergleichbar sind, wenn $v'_{\ell'} \neq v_\ell$ gilt. Die Gewichtsfunktion $w = (w_i)$ allein stellt daher

keine geeignete Fitnessfunktion dar. Demzufolge definieren wir die Fitnessfunktion $f = (f_{t,i})\colon PFAD(1,\cdot) \to \mathbb{N}^{n\cdot k}$ durch

$$f_{t,i}(p) := \begin{cases} w_i(p) & \text{falls } v_\ell = t \\ c \cdot (n-1) \cdot w^{\max} & \text{falls } v_\ell \neq t \end{cases}$$

für alle $1 \leq t \leq n$ und $1 \leq i \leq k$, wobei $c > 1$ gilt. Wir weisen darauf hin, dass wir uns für eine Problemformulierung entschieden haben, die die Optimierung von

$$d = n \cdot k$$

verschiedenen Fitnesswerten umfasst. Mithilfe des Faktors $c > 1$ wird sichergestellt, dass $c \cdot (n-1) \cdot w^{\max}$ größer als das Gewicht $w_i(p)$ jedes beliebigen einfachen Pfades p ist. Die Definition der Fitnessfunktion sorgt dafür, dass $0 \leq f_{t,i}(p) \leq (n-1) \cdot w^{\max}$ für $t = v_\ell$ und $f_{t,i}(p) = c \cdot (n-1) \cdot w^{\max}$ für $t \neq v_\ell$ gelten. Insgesamt gilt also $f(p') \succeq f(p)$ genau dann, wenn $w(p') \succeq w(p)$ und $v'_{\ell'} = v_\ell$ gelten.

Ein weiterer plausibler Ansatz besteht darin, die k-dimensionale Fitnessfunktion $w = (w_i)$ zu verwenden und den evolutionären Algorithmus so anzupassen, dass er Pfade mit unterschiedlichen Senken als unvergleichbar ansieht. Das Verhalten ist in beiden Fällen gleich. Wir entscheiden uns hier für die erste Alternative, die auf einem allgemeineren evolutionären Algorithmus basiert und zeigt, wie zusätzliche Informationen in die Fitnessfunktion integriert werden können.

7.3.2. Initialisierung, Mutation und Selektion

Wir motivieren und präsentieren in diesem Abschnitt alle Komponenten des evolutionären Algorithmus, den wir in den kommenden Abschnitten analysieren werden.

Die Initialisierung der Population ist einfach. Wir entscheiden uns für die naheliegende Möglichkeit und beginnen mit allen Pfaden der Länge 0, d. h., wir setzen $P_0 := PFAD_0(1,\cdot) = \{(1)\}$. Eine weitere gängige Möglichkeit bestünde darin, mit einem zufälligen Pfad zu beginnen. Wir werden am Ende von Abschnitt 7.4 sehen, dass die oberen Schranken in beiden Fällen gleichermaßen gelten.

Die Wahl eines geeigneten Variationsoperators ist komplizierter. Wir entscheiden uns für eine einfache Variante und benutzen nur einen einzigen Mutationsoperator. Dabei wählen wir in jeder Generation ein Individuum aus der aktuellen Population rein zufällig zur Mutation. Es ist eine klassische Richtlinie, dass die Anwendung eines Mutationsoperators in den meisten Fällen eine kleine Veränderung des selektierten Individuums nach sich ziehen sollte. Auf der anderen Seite sollte es möglich sein, mit positiver Wahrscheinlichkeit jedes andere Individuum in einem einzigen Mutationsschritt zu erreichen, damit das Erreichen eines Optimierungsziels mit gegen 1 konvergierender Wahrscheinlichkeit nicht schon an einem zu eingeschränkten Mutationsoperator scheitert. Wir weisen darauf hin, dass die genannte Eigenschaft des Mutationsoperators garantiert, dass das Optimierungsziel im Erwartungswert nach endlich vielen Schritten erreicht wird. Sie ist für die genannte Garantie hinreichend aber nicht notwendig. Schwächere Anforderungen an den Mutationsoperator,

die dennoch die Konvergenz gegen ein globales Optimum sicherstellen, werden z. B. in [Rudolph, 1996] untersucht.

Wenn die einzelnen Individuen Bitfolgen der Länge n repräsentieren, dann setzt die Standard-Bit-Mutation diese Richtlinie um, indem jedes Bit unabhängig von den anderen mit Wahrscheinlichkeit $1/n$ gekippt wird (siehe Abschnitt 2.4). Offensichtlich können wir diesen Mutationsoperator hier nicht anwenden, da wir einen anderen Suchraum untersuchen. Wir entscheiden uns dafür, das Verhalten des genannten Operators so weit wie möglich zu simulieren. Das Kippen eines Bits entspricht hier einer minimalen Modifikation eines Pfades. Genau wie Doerr et al. [2008a] entscheiden wir uns dafür, die letzte Kante eines Pfades zu entfernen oder eine Kante an den Pfad anzufügen. Die folgende Fallunterscheidung präzisiert die Wahrscheinlichkeiten für diese atomaren Mutationsschritte. Wenn wir einen Pfad $p = (v_0, \ldots, v_\ell)$ mutieren möchten, dann wählen wir eine der Alternativen $b \in \mathbb{B}$ rein zufällig und konstruieren den mutierten Pfad p' wie folgt.

- Wenn $b = 0$ und $\ell \geq 1$, dann setze $p' = (v_0, \ldots, v_{\ell-1})$.
- Wenn $b = 1$ und $\{v \in V \mid (v_\ell, v) \in E\} \neq \emptyset$, dann wähle $v' \in \{v \in V \mid (v_\ell, v) \in E\}$ rein zufällig und setze $p' = (v_0, \ldots, v_\ell, v')$.
- Anderenfalls setze $p' = p$.

Anschließend entfernen wir alle Kreise aus p' um sicherzustellen, dass der mutierte Pfad einfach ist.

Um das globale Verhalten der weiter oben genannten Standard-Bit-Mutation zu simulieren, müssen wir mehrere atomare Mutationsschritte pro Mutation ermöglichen. Die Anzahl der gekippten Bits bei Anwendung der Standard-Bit-Mutation ist gemäß einer Binomialverteilung verteilt, wobei die Dichtefunktion der Binomialverteilung für $n \in \mathbb{N}$, $0 \leq p \leq 1$ und $0 \leq i \leq n$ durch $f(i; n, p) = \binom{n}{i} \cdot p^i \cdot (1-p)^{n-i}$ gegeben ist. Bekanntlich konvergiert die Binomialverteilung gegen die Poisson-Verteilung mit $\lambda = n \cdot p$, wenn n gegen ∞ strebt, wobei die Dichtefunktion der Poisson-Verteilung für $\lambda > 0$ und $i \in \mathbb{N}$ durch $f(i; \lambda) = \lambda^i \cdot e^{-\lambda}/i!$ gegeben ist [Feller, 1968, 1971]. Es ist daher natürlich, die Poisson-Verteilung mit $\lambda = n \cdot 1/n = 1$ zu verwenden, um die Anzahl der atomaren Mutationsschritte zu bestimmen, wenn eine Simulation der Standard-Bit-Mutation angestrebt wird. Die gleiche Verteilung wurde bereits in [Scharnow et al., 2004, Doerr et al., 2007a, 2008a] benutzt. Der vorgeschlagene Mutationsoperator stimmt fast mit dem in [Doerr et al., 2008a] verwendeten Operator überein; dort werden die atomaren Mutationsschritte jedoch mit anderen Wahrscheinlichkeiten durchgeführt. Auf der anderen Seite unterscheidet er sich von dem in [Scharnow et al., 2004, Doerr et al., 2007a] benutzten Mutationsoperator, da dort eine andere Repräsentation zugrunde liegt. Die Individuen entsprechen dort $(v_2, \ldots, v_n) \in \{1, \ldots, n\}^{n-1}$, $v_i \neq i$, wobei v_i als Vorgänger von i interpretiert wird. Die genannte Repräsentation ist der Tatsache geschuldet, dass im monokriteriellen Fall alle kürzesten Wege einen Baum mit Wurzel 1 bilden. Diese elegante Darstellung überträgt sich leider nicht auf den allgemeinen multikriteriellen Fall.

Wir wenden uns nun der Selektion zur Ersetzung zu. Der einfachste Ansatz, diese zu implementieren, besteht darin, alle nicht dominierten Individuen in der aktuellen Population zu verwahren (siehe Algorithmus 2). Wir haben bereits festgestellt, dass ein Nachteil dieses Vorgehens darin besteht, dass die Populationsgröße stark anwachsen kann, obwohl sich alle Individuen auf einen bestimmten Teil der Paretofront konzentrieren (siehe Abschnitt 3.3). In diesem Fall wird nur eine schlechte Approximationsgüte erreicht. Es ist daher wichtig, einerseits die Populationsgröße zu kontrollieren und andererseits die Diversität in der Population zu erhöhen. Wir greifen erneut auf den Ansatz zurück, den Zielraum in Boxen zu unterteilen und in der Population höchstens ein Individuum pro Box zu behalten (siehe Algorithmus 3). In Kapitel 3 sind wir an der Minimierung der additiven Approximationsgüte interessiert gewesen. Hier interessieren wir uns für die Minimierung der multiplikativen Approximationsgüte. Wir definieren daher den sogenannten Boxindex $b = (b_1, \ldots, b_d)\colon \mathbb{N}^d \to \mathbb{Z}^d$ durch

$$b_i(y = (y_1, \ldots, y_d)) := \begin{cases} \lfloor \log_r y_i \rfloor & \text{falls } y_i \geq 1 \\ -1 & \text{falls } y_i = 0, \end{cases}$$

wobei $r > 1$ die Größe der Boxen definiert. Die Fallunterscheidung stellt sicher, dass auch Zielpunkte $y = (y_1, \ldots, y_d)$ mit $y_i = 0$ auf einen Boxindex abgebildet werden können. In den folgenden Ausführungen werden wir der Einfachheit halber zusätzlich die Notation $b(Y) := \{b(y) \mid y \in Y\}$ für Teilmengen $Y \subseteq \mathbb{N}^d$ des Zielraumes verwenden.

Hiermit haben wir die Initialisierung der Population, die Arbeitsweise des Variationsoperators und die Selektion zur Ersetzung, die auf die Minimierung der multiplikativen Approximationsgüte abgestimmt ist, beschrieben. Wir bezeichnen den Algorithmus erneut mit DEMO und verweisen auf die Darstellung in Algorithmus 3, die das Zusammenspiel der einzelnen Komponenten beschreibt.

7.3.3. Grundlegende Eigenschaften

In diesem Abschnitt werden wir zwei Lemmas beweisen, die die wichtigsten Eigenschaften der Selektion zur Ersetzung zusammenfassen. Wir machen uns in diesem Kapitel die durch $x \succeq_{\delta-1,\cdot} x'$ definierte Dominanzrelation $x \succeq_\delta x'$ zunutze, um die Notation kurz zu halten (siehe auch Abschnitt 2.2). Wir sprechen in diesem Fall von δ-Dominanz.

Die Selektion zur Ersetzung kann – wie im letzten Unterabschnitt thematisiert – nicht garantieren, dass jedes gefundene Individuum von einem Individuum aus der aktuellen Population schwach dominiert wird. Demzufolge müssen wir uns mit Approximationen zufriedengeben. Das nächste Lemma stellt im Wesentlichen sicher, dass alle Suchpunkte, die von einem gefundenen Individuum gut approximiert werden, auch von einem Individuum aus der aktuellen Population gut approximiert werden. Wir werden sehen, dass die für einen festen Suchpunkt erreichte Approximationsgüte im Laufe der Optimierung nicht wesentlich abnehmen kann. Eine vergleichbare Aussage haben wir bereits in Lemma 3.1 aus Kapitel 3 kennengelernt.

Lemma 7.1. *Wir wenden den Algorithmus DEMO mit Boxgröße $r > 1$ auf die Fitnessfunktion $g = (g_1, \ldots, g_d)\colon S \to \mathbb{N}^d$ an. Wenn $x_t \succeq_{r^j} s$ gilt, dann gilt nach der Selektion zur Ersetzung*

$$P_{t'} \succeq_{r^{j+1}} s'$$

für alle $t' \in \mathbb{N}$ mit $t' \geq t$ und für alle $s' \in S$ mit $b(g(s')) \preceq b(g(s))$.

Beweis. Wir wissen, dass $x_t \succeq_{r^j} s$ und $b(g(s')) \preceq b(g(s))$ gelten. Definitionsgemäß folgt, dass $g_i(x_t) \leq r^j \cdot g_i(s)$ und $b_i(g(s)) \leq b_i(g(s'))$ für alle $i \in \mathbb{N}$ mit $1 \leq i \leq d$ gelten. Wir schließen

$$b_i(g(x_t)) = \lfloor \log_r(g_i(x_t)) \rfloor \leq \lfloor \log_r(r^j \cdot g_i(s)) \rfloor = j + b_i(g(s)) \leq j + b_i(g(s')). \quad (7.1)$$

Die Selektion zur Ersetzung stellt sicher, dass es ein $s'' \in P_{t'}$ mit $b(g(s'')) \succeq b(g(x_t))$ gibt. Definitionsgemäß gilt dann

$$b_i(g(s'')) \leq b_i(g(x_t)). \quad (7.2)$$

Indem wir Formel 7.1 und 7.2 kombinieren, erhalten wir die Ungleichung $b_i(g(s'')) \leq j + b_i(g(s'))$ und schließen weiter

$$\begin{aligned} b_i(g(s'')) \leq j + b_i(g(s')) &\iff \lfloor \log_r(g_i(s'')) \rfloor \leq j + \lfloor \log_r(g_i(s')) \rfloor \\ &\implies \frac{\log(g_i(s''))}{\log(r)} - 1 \leq j + \frac{\log(g_i(s'))}{\log(r)} \\ &\iff g_i(s'') \leq r^{j+1} \cdot g_i(s'). \end{aligned}$$

Insgesamt folgt die Behauptung. □

Das nächste Lemma stellt – genau wie Lemma 3.2 aus Kapitel 3 – eine obere Schranke für die Populationsgröße bereit.

Lemma 7.2. *Wir wenden den Algorithmus DEMO mit Boxgröße $r > 1$ auf die Fitnessfunktion $g = (g_1, \ldots, g_d)\colon S \to \mathbb{N}^d$ an. Wenn $S = \bigcup_{i \in \{1,\ldots,n\}} S_i$ gilt, dann gilt nach der Selektion zur Ersetzung*

$$|P_t| \leq \sum_{i \in \{1,\ldots,n\}} \min_{j \in \{1,\ldots,d\}} \prod_{k \in \{1,\ldots,d\} \setminus \{j\}} |b_k(g(S_i))|$$

für alle $t \in \mathbb{N}$.

Beweis. Betrachte S_i. Das Bild $g(S_i)$ umfasst Fitnessvektoren aus

$$|b(g(S_i))| \leq \prod_{k \in \{1,\ldots,d\}} |b_k(g(S_i))|$$

verschiedenen Boxen. Die Population enthält höchstens

$$\min_{j \in \{1,\ldots,d\}} \prod_{k \in \{1,\ldots,d\} \setminus \{j\}} |b_k(g(S_i))|$$

Individuen, da zwei Fitnessvektoren, die sich in höchstens einer Komponente unterscheiden, vergleichbar sind. Da $S = \bigcup_{i \in \{1,\ldots,n\}} S_i$ gilt, folgt insgesamt die Behauptung. □

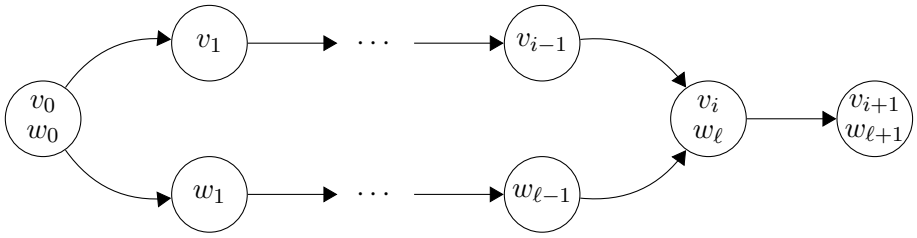


Abbildung 7.2.: Der Zusammenhang zwischen den in Lemma 7.3 genannten Pfaden $p' = (v_0, v_1, \ldots, v_{i-1}, v_i, v_{i+1})$ und $q' = (w_0, w_1, \ldots, w_{\ell-1}, w_\ell, w_{\ell+1})$.

Wenn $u = (u_1, \ldots, u_d)$ den Utopiapunkt bezeichnet, dann gilt

$$|b_j(g(S))| \leq \left\lfloor \frac{\log(u_j)}{\log(r)} \right\rfloor + 2$$

für alle $1 \leq j \leq d$. Folglich kann die Populationsgröße mithilfe von Lemma 7.2 beispielsweise durch

$$\left(\frac{\log(\max\{u_1, \ldots, u_d\})}{\log(r)} + 2 \right)^{d-1}$$

abgeschätzt werden.

7.4. Obere Schranken für die Optimierungszeit

7.4.1. Vorbereitungen

Wir beginnen mit zwei Lemmas, die in den Beweisen der folgenden Theoreme benötigt werden. Das erste Lemma stellt den Kern des Beweises von Theorem 7.5 dar. Es zeigt, wie eine r^i-Approximation der Paretofront von $f\colon PFAD_{\leq i}(1,\cdot) \to \mathbb{N}^d$ zu einer r^{i+1}-Approximation der Paretofront von $f\colon PFAD_{\leq i+1}(1,\cdot) \to \mathbb{N}^d$ weiterentwickelt werden kann.

Lemma 7.3. *Wir wenden den Algorithmus DEMO mit Boxgröße $r > 1$ auf die Fitnessfunktion $f = (f_{t,i})\colon PFAD(1,\cdot) \to \mathbb{N}^d$ mit $c > r^{n-2}$ an. Seien $p = (v_0, \ldots, v_i) \in PFAD_i(1,\cdot)$ mit $0 \leq i \leq n-2$ und $q = (w_0, \ldots, w_\ell) \in PFAD(1,\cdot)$ mit $f(q) \preceq_{r^i} f(p)$ einfache Pfade. Dann gibt es für alle Erweiterungen $p' = (v_0, \ldots, v_i, v_{i+1})$ von p eine Erweiterung $q' = (w_0, \ldots, w_\ell, w_{\ell+1})$ von q mit $f(q') \preceq_{r^i} f(p')$.*

Beweis. Betrachte den Teilpfad $p = (v_0, \ldots, v_i)$ von p'. Wir folgern aus $f(q) \preceq_{r^i} f(p)$, dass $f_{v_i,j}(q) \leq r^i \cdot f_{v_i,j}(p)$ für alle j, $1 \leq j \leq k$, gilt. Wir benutzen die Voraussetzungen $i \leq n-2$ und $r < c^{1/(n-2)}$, um

$$r^i \cdot f_{v_i,j}(p) \leq r^{n-2} \cdot w_j(p) < c \cdot (n-1) \cdot w^{\max} \tag{7.3}$$

zu schließen. Gemäß Definition der Fitnessfunktion würde die Annahme $w_\ell \neq v_i$ zu $f_{v_i,j}(q) = c \cdot (n-1) \cdot w^{\max}$ und damit letztlich zu einem Widerspruch zu Formel 7.3

führen. Folglich gilt $w_\ell = v_i$. Weiterhin gilt

$$w_j(q) = f_{w_\ell,j}(q) \leq r^i \cdot f_{v_i,j}(p) = r^i \cdot w_j(p).$$

Betrachte die Erweiterung $q' = (w_0, \ldots, w_\ell, w_{\ell+1})$ von q mit $w_{\ell+1} = v_{i+1}$. Der Zusammenhang zwischen p' und q' wird in Abbildung 7.2 veranschaulicht. Wir folgern abschließend $f(q') \succeq_{r^i} f(p')$, weil

$$\begin{aligned} f_{v_{i+1},j}(q') = w_j(q') &= w_j(q) + w_j((w_\ell, w_{\ell+1})) \\ &\leq r^i \cdot w_j(p) + w_j((v_i, v_{i+1})) \\ &< r^i \cdot w_j(p') = r^i \cdot f_{v_{i+1},j}(p') \end{aligned}$$

für alle $1 \leq j \leq k$ und

$$f_{t,j}(q') = c \cdot (n-1) \cdot w^{\max} = f_{t,j}(p')$$

für alle $1 \leq t \leq n$, $t \neq v_{i+1}$, und $1 \leq j \leq k$ gelten. □

Wir begrenzen nun mithilfe von Lemma 7.2 die Populationsgröße.

Lemma 7.4. *Wir wenden den Algorithmus DEMO mit Boxgröße $r > 1$ auf die Fitnessfunktion $f = (f_{t,i})$: $PFAD(1,\cdot) \to \mathbb{N}^d$ mit $c > 1$ an. Dann gilt nach der Selektion zur Ersetzung*

$$|P_t| \leq (n-1) \cdot (\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 2)^{k-1} + 1$$

für alle $t \in \mathbb{N}$.

Beweis. Betrachte die Partition $\bigcup_{j=1}^{n} PFAD(1,j)$ des Suchraumes $PFAD(1,\cdot)$. Aufgrund von Lemma 7.2 wird die Größe der Population durch

$$\sum_{j=1}^{n} \prod_{(t,i)|(t,i)\neq(j,1)} |b_{t,i}(f(PFAD(1,j)))|$$

begrenzt. Wir weisen darauf hin, dass wir uns in Übereinstimmung mit Lemma 7.2 dazu entschieden haben, die Dimension $(j,1)$ in dem letzten Ausdruck fortzulassen. Weiterhin gilt

$$\begin{cases} 0 \leq f_{t,i}(p) \leq (n-1) \cdot w^{\max} & \text{falls } t = j \\ f_{t,i}(p) = c \cdot (n-1) \cdot w^{\max} & \text{falls } t \neq j \end{cases}$$

für alle $p \in PFAD(1,j)$. Folglich gilt

$$\begin{cases} |b_{t,i}(f(PFAD(1,j)))| \leq \lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 2 & \text{falls } t = j \\ |b_{t,i}(f(PFAD(1,j)))| = 1 & \text{falls } t \neq j \end{cases} \tag{7.4}$$

für alle (t,i), wobei der erste Teil von Formel 7.4 aus

$$b((n-1) \cdot w^{\max}) - b(0) + 1 = \lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 2$$

resultiert. Insgesamt ist die Populationsgröße daher nicht größer als

$$1 \cdot 1^{d-1} + (n-1) \cdot \left((\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 2)^{k-1} \cdot 1^{d-k} \right),$$

da der Term 1^{d-1} eine obere Schranke für das Produkt $\prod |b_{t,i}(f(PFAD(1,1)))|$ darstellt und der Term $(\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 2)^{k-1} \cdot 1^{d-k}$ für alle $2 \leq j \leq n$ eine obere Schranke für das Produkt $\prod |b_{t,i}(f(PFAD(1,j)))|$ darstellt. □

Wir weisen an dieser Stelle explizit darauf hin, dass Lemma 7.4 auf der unteren Schranke 0 und der oberen Schranke $(n-1) \cdot w^{\max}$ für das Gewicht eines Pfades basiert. Die untere Schranke folgt aus der Annahme, dass alle Kantengewichte mindestens 0 betragen, und die obere Schranke folgt aus der Beschränkung auf *einfache* Pfade. Wenn wir beliebige Pfade erlauben würden, dann könnten wir die Populationsgröße im Allgemeinen nicht mehr so begrenzen.

Wenn wir auch nicht einfache Pfade zuließen, dann könnte die Population tatsächlich eine beliebige Größe erreichen, wie das folgende Beispiel zeigt. Betrachte einen Graphen, der aus einem Knoten v und zwei Kanten $e_1 = (v,v)$ und $e_2 = (v,v)$ besteht, und die Gewichtsfunktion w mit $w(e_1) = (1,0)$ und $w(e_2) = (0,1)$. Dann gilt für die Fitness $f(p_{i,j})$ eines Pfades $p_{i,j}$, der i-mal die Kante e_1 und j-mal die Kante e_2 enthält, $f(p_{i,j}) = (i,j)$, wobei $i \in \mathbb{N}$ und $j \in \mathbb{N}$ sind. Offensichtlich ist der Pfad $p_{0,0}$ optimal. Sei $\ell \in \mathbb{N}$ beliebig. Wir nehmen an, dass der evolutionäre Algorithmus mit einer leeren Population beginnt und in Generation i das Individuum $x_i = p_{i,\ell-i}$, $0 \leq i \leq \ell$, erschafft. Dann besteht die Population nach Generation ℓ aus genau $\lfloor \log(\ell)/\log(r) \rfloor + 2 = \Omega(\log(\ell)/\log(r))$ Individuen, da alle x_i unvergleichbar sind. Die Population kann also prinzipiell beliebig groß werden, wenn die Individuen in der beschriebenen ungünstigen Reihenfolge generiert werden. Wir weisen darauf hin, dass das Beispiel leicht so angepasst werden kann, dass der Graph höchstens eine Kante zwischen zwei verschiedenen Knoten enthält und dass die Paretofront nicht bereits in der initialen Population des evolutionären Algorithmus enthalten ist. Die Einschränkung des Suchraumes auf einfache Pfade ist also sinnvoll.

7.4.2. Analysen

Wir zeigen zunächst eine relativ einfache obere Schranke für die erwartete Optimierungszeit, die wir im Anschluss verfeinern werden.

Theorem 7.5. *Sei $\epsilon\colon \mathbb{N}^+ \to \mathbb{R}^+$ mit $\epsilon = \epsilon(n) = \mathcal{O}(1)$. Wir wenden den Algorithmus DEMO mit Boxgröße $r = r(n,\epsilon) = (1+\epsilon)^{1/(n-1)}$ auf die Fitnessfunktion $f = (f_{t,i})\colon PFAD(1,\cdot) \to \mathbb{N}^d$ mit $c = c(n,\epsilon) > (1+\epsilon)^{(n-2)/(n-1)}$ an. Dann gilt $\mathrm{I}_{\mathrm{mul}}(P) \leq 1+\epsilon$ im Erwartungswert nach*

$$\mathcal{O}\left(n^3 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon} \right)^{k-1} \cdot \log\left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\epsilon} \right) \right)$$

Generationen.

Beweis. Wir bezeichnen die aktuelle Population mit P. Wir gliedern einen Lauf des Algorithmus DEMO in $n-1$ Phasen, wobei die i-te Phase endet, wenn die Bedingung $f(P) \succeq_{r^i} f(PFAD_{\leq i}(1,\cdot))$ eintritt. Nach der $(n-1)$-ten Phase gilt also die Bedingung $f(P) \succeq_{r^{n-1}} f(PFAD_{\leq n-1}(1,\cdot))$ und daher $f(P) \succeq_{1+\epsilon} f(PFAD(1,\cdot))$, weil

$$r^{n-1} = \Big((1+\epsilon)^{1/(n-1)}\Big)^{n-1} = 1+\epsilon$$

aus der Wahl der Boxgröße r folgt. Die Initialisierung der Population stellt eingangs $f(P) \succeq_{r^0} f(PFAD_{\leq 0}(1,\cdot))$ sicher, da $P = PFAD_{\leq 0}(1,\cdot)$ erfüllt ist.

Betrachte $f(P) \succeq_{r^i} f(PFAD_{\leq i}(1,\cdot))$ mit $0 \leq i \leq n-2$. Wir bestimmen jetzt eine obere Schranke für die erwartete Anzahl von Generationen bis $f(P) \succeq_{r^{i+1}} f(PFAD_{\leq i+1}(1,\cdot))$ gilt. Dazu betrachten wir die nicht dominierten Boxindizes von $b(f(PFAD_{i+1}(1,\cdot)))$. Es gibt höchstens

$$\nu := (n-1)\cdot(\lfloor \log_r((n-1)\cdot w^{\max})\rfloor + 2)^{k-1} + 1$$

solche Indizes (siehe Lemma 7.4). Wir fixieren aus den zugehörigen Boxen jeweils einen Pfad $p_j \in PFAD_{i+1}(1,\cdot)$. Lemma 7.3 stellt sicher, dass es für alle Pfade p_j einen Pfad $p'_j \in P$ gibt, sodass eine atomare Mutation von p'_j genügt, um ein Individuum zu erschaffen, das p_j r^i-dominiert. Lemma 7.1 sichert, dass alle folgenden Populationen alle p''_j mit $b(p''_j) \preceq b(p_j)$ r^{i+1}-dominieren. Wenn alle p_j r^i-dominiert wurden, dann gilt für alle folgenden Populationen, dass diese die gesamte Menge $PFAD_{\leq i+1}(1,\cdot)$ r^{i+1}-dominieren.

Die Wahrscheinlichkeit, ein bestimmtes Individuum zur Mutation auszuwählen, beträgt $1/|P|$, die Wahrscheinlichkeit, genau einen atomaren Mutationsschritt durchzuführen, ist $1/e$ und die Wahrscheinlichkeit, eine bestimmte Kante (u,v) anzufügen, ist mindestens $1/(2\cdot(n-1))$. Die letzte Wahrscheinlichkeit ergibt sich, da der Mutationsoperator mit Wahrscheinlichkeit $1/2$ eine Kante anfügt und die Kante (u,v) aus allen ausgehenden Kanten mit Wahrscheinlichkeit mindestens $1/(n-1)$ wählt. Da die drei genannten Ereignisse unabhängig sind, beträgt die Wahrscheinlichkeit, einen Pfad zu erschaffen, der p_j r^i-dominiert, mindestens $1/(2\cdot e\cdot(n-1)\cdot|P|)$. Mithilfe von Lemma 7.4 können wir die Wahrscheinlichkeit durch $1/(2\cdot e\cdot(n-1)\cdot\nu)$ nach unten begrenzen. Es dauert im Erwartungswert also höchstens $2\cdot e\cdot(n-1)\cdot\nu$ Generationen bis das Ereignis eintritt. Wir interessieren uns nun für die erwartete Zeit bis alle ν Ereignisse eingetreten sind. Diese Situation entspricht genau dem bekannten Sammelbilderproblem (siehe Lemma A.7). Folglich müssen wir im Erwartungswert höchstens

$$2\cdot e\cdot(n-1)\cdot\nu\cdot\sum_{j=1}^{\nu}\frac{1}{j} = 2\cdot e\cdot(n-1)\cdot\nu\cdot H_\nu$$

Schritte warten, bis dauerhaft $f(P) \succeq_{r^{i+1}} f(PFAD_{\leq i+1}(1,\cdot))$ gilt, wobei H_n die n-te harmonische Zahl ist.

Wenn wir alle $n-1$ Phasen berücksichtigen, dann benötigt der Algorithmus DEMO im Erwartungswert insgesamt höchstens

$$(n-1)\cdot 2\cdot e\cdot(n-1)\cdot\nu\cdot H_\nu = \mathcal{O}\Big(n^3\cdot(\log_r(n\cdot w^{\max}))^{k-1}\cdot\log\Big(n\cdot(\log_r(n\cdot w^{\max}))^{k-1}\Big)\Big)$$

Generationen, um die Population in eine $(1+\epsilon)$-Approximation der Paretofront zu transformieren, da $H_n = \mathcal{O}(\log n)$ gilt (siehe Lemma A.17). Wir können die obere Schranke zu

$$\mathcal{O}\left(n^3 \cdot \left(\frac{\log(n \cdot w^{\max})}{\log(r)}\right)^{k-1} \cdot \log\left(n \cdot \left(\frac{\log(n \cdot w^{\max})}{\log(r)}\right)^{k-1}\right)\right)$$
$$= \mathcal{O}\left(n^3 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\log(1+\epsilon)}\right)^{k-1} \cdot \left(\log(n) + k \cdot \log\left(\frac{n \cdot \log(n \cdot w^{\max})}{\log(1+\epsilon)}\right)\right)\right).$$

vereinfachen. Indem wir $\log(n) = \mathcal{O}(\log(n \cdot \log(n \cdot w^{\max})/\log(1+\epsilon)))$ und $k = \mathcal{O}(1)$ ausnutzen, können wir die letzte Schranke zu

$$\mathcal{O}\left(n^3 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\log(1+\epsilon)}\right)^{k-1} \cdot \log\left(\frac{n \cdot \log(n \cdot w^{\max})}{\log(1+\epsilon)}\right)\right)$$
$$= \mathcal{O}\left(n^3 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)^{k-1} \cdot \log\left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)\right)$$

reduzieren, da $\log(1+\epsilon) \geq \log(1+\epsilon/c) \geq \epsilon/c$ für $0 \leq \epsilon \leq c$ und $c \geq 1$ konstant gilt. □

Um die obere Schranke für die Optimierungszeit in eine obere Schranke für die tatsächliche Laufzeit umzuwandeln, müssen wir die Laufzeit für die Initialisierung und die Schleifendurchläufe abschätzen. Wenn wir eine naheliegende Implementierung des Algorithmus DEMO betrachten, dann erfordert die Initialisierung eine Laufzeit von $\Theta(m)$ Rechenschritten und ein Schleifendurchlauf eine erwartete Laufzeit von

$$\Theta(|P| \cdot d) = \mathcal{O}\left(n^2 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)^{k-1}\right)$$

Rechenschritten, da der Algorithmus die Dominanzrelation für alle d Fitnesswerte und für alle Individuen in der aktuellen Population überprüfen muss. Folglich schließen wir mithilfe des letzten Theorems, dass die erwartete Laufzeit, bis der Algorithmus DEMO die initiale Population in eine $(1+\epsilon)$-Approximation der Paretofront verwandelt hat, durch

$$\text{poly}(n, 1/\epsilon) = \mathcal{O}\left(n^5 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)^{2\cdot(k-1)} \cdot \log\left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)\right)$$

Rechenschritte begrenzt ist, wobei $\text{poly}(\cdot,\cdot)$ ein geeignetes Polynom bezeichnet. Wenn wir den Algorithmus DEMO nach $4 \cdot \lceil \text{poly}(n, 1/\epsilon) \rceil$ Rechenschritten terminieren, dann können wir die Markoff-Ungleichung (siehe Lemma A.5) anwenden, um zu schließen, dass die letzte Population mit Wahrscheinlichkeit größer als $3/4$ eine $(1+\epsilon)$-Approximation der Paretofront ist. Insgesamt haben wir bewiesen, dass der Algorithmus DEMO ein FPRAS für die Variante $(1, n)$ des multikriteriellen Kürzeste-Wege-Problems ist.

Wir stellen an dieser Stelle die Definitionen eines *echt polynomiellen Approximationsschemas* (auf Englisch „fully polynomial-time approximation scheme“, FPTAS) und eines *echt polynomiellen randomisierten Approximationsschemas* (auf Englisch „fully polynomial-time randomized approximation scheme“, FPRAS) für multikriterielle Optimierungsprobleme gegenüber. Ein Algorithmus heißt genau dann FPTAS (FPRAS) für ein multikriterielles Optimierungsproblem, wenn er für jede Eingabeinstanz und für jede Approximationsgüte $1+\epsilon$ mit $\epsilon \in \mathbb{R}^+$

- bezogen auf die Eingabegröße und $1/\epsilon$ in Polynomialzeit läuft und
- mit Wahrscheinlichkeit 1 (mindestens 3/4) eine $(1+\epsilon)$-Approximation der Paretofront liefert.

Beachte, dass die Schranke 3/4 für die Erfolgswahrscheinlichkeit innerhalb gewisser Grenzen beliebig gewählt werden kann. Alle konstanten Schranken c mit $1/2 < c < 1$ können in der obigen Definition benutzt werden. Da die Erfolgswahrscheinlichkeit eines FPRAS mithilfe von Standardtechniken amplifiziert werden kann, führen alle Wahlen von c zur gleichen Klasse von Problemen, die ein FPRAS besitzen.

Das nächste Theorem zeigt, dass die Schranke für die Optimierungszeit, die im letzten Theorem nachgewiesen wurde, verbessert werden kann. Der Beweis des letzten Theorems folgt einer klassischen Beweismethode, die darauf basiert, die erwartete Optimierungszeit zu begrenzen, die benötigt wird, um eine r^i-Approximation der Paretofront von $f\colon PFAD_{\leq i}(1,\cdot) \to \mathbb{N}^d$ in eine r^{i+1}-Approximation der Paretofront von $f\colon PFAD_{\leq i+1}(1,\cdot) \to \mathbb{N}^d$ zu verwandeln. Die verwendete Beweismethode ist klar strukturiert, sie berücksichtigt die Konstruktion von guten Pfaden mit mehr als $i+1$ Kanten jedoch nicht solange noch nicht alle Pfade mit höchstens $i+1$ Kanten r^{i+1}-dominiert werden. Wir bezeichnen einen Pfad als „gut“, der zum ersten Mal einen Pfad in einer nicht dominierten Box aus $b(f(PFAD_{\leq j+1}(1,\cdot)))$ für ein $j > i$ r^j-dominiert. Da die genannten Ereignisse – wie der Beweis des nächsten Theorems zeigen wird – relativ häufig eintreten, ist die hergeleitete Schranke nicht scharf. Der Beweis des folgenden Theorems basiert auf einer Beweismethode, die bereits in [Doerr et al., 2007a] Verwendung findet, und erlaubt, eine verbesserte obere Schranke zu zeigen.

Theorem 7.6. *Sei* $\epsilon\colon \mathbb{N}^+ \to \mathbb{R}^+$ *mit* $\epsilon = \epsilon(n) = \mathcal{O}(1)$. *Wir wenden den Algorithmus DEMO mit Boxgröße* $r = r(n,\epsilon) = (1+\epsilon)^{1/(n-1)}$ *auf die Fitnessfunktion* $f = (f_{t,i})\colon PFAD(1,\cdot) \to \mathbb{N}^d$ *mit* $c = c(n,\epsilon) > (1+\epsilon)^{(n-2)/(n-1)}$ *an. Wenn* $\log(w^{\max})$ *und* $1/\epsilon$ *polynomiell in* n *beschränkt sind, dann gilt* $\mathrm{I}_{\mathrm{mul}}(P) \leq 1+\epsilon$ *mit Wahrscheinlichkeit* $1-\exp(-\Omega(n))$ *nach*

$$t := 4 \cdot e \cdot (n-2) \cdot (n-1) \cdot \nu = \mathcal{O}\left(n^3 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)^{k-1}\right)$$

Generationen, wobei

$$\nu := (n-1) \cdot (\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 2)^{k-1} + 1$$

gilt. Die Größenordnung der erwarteten Optimierungszeit entspricht der von t.

Beweis. Betrachte einen einfachen Pfad $q = (v_0, \ldots, v_\ell)$, $0 \leq \ell \leq n-1$, mit $v_0 = 1$. Wir nennen einen Mutationsschritt *i-te Verbesserung von q*, $1 \leq i \leq \ell$, wenn ein Individuum, das $(v_0, \ldots, v_i)$ zum ersten Mal seit Beginn der Optimierung r^{i-1}-dominiert, erschaffen wird. Aufgrund des Beweises von Theorem 7.5 beträgt die Wahrscheinlichkeit für die nächste Verbesserung mindestens $p := 1/(2 \cdot e \cdot (n-1) \cdot \nu)$, wenn weniger als ℓ Verbesserungen von q erreicht wurden. Betrachte eine Phase der Länge t, d. h. t aufeinander folgende Generationen. Definiere die unabhängigen Zufallsvariablen X_j, $1 \leq j \leq t$, durch $\mathrm{W}(X_j = 1) = p$ und $\mathrm{W}(X_j = 0) = 1 - p$ sowie die Zufallsvariable $X := \sum_{j=1}^{t} X_j$. Dann gilt

$$\mathrm{E}(X) = t \cdot p = 4 \cdot e \cdot (n-2) \cdot (n-1) \cdot \nu \cdot \frac{1}{2 \cdot e \cdot (n-1) \cdot \nu} = 2 \cdot (n-2)$$

aufgrund der Linearität des Erwartungswertes. Die Wahrscheinlichkeit, dass X_j den Wert 1 annimmt, ist eine untere Schranke für die Wahrscheinlichkeit, die nächste Verbesserung von q zu erreichen, wenn weniger als ℓ Verbesserungen erreicht wurden. Daher können wir unter Benutzung dieser Zufallsvariablen die Wahrscheinlichkeit, dass die ℓ-te Verbesserung nicht in höchstens t Mutationsschritten eintritt, durch

$$\begin{aligned}
\mathrm{W}(X \leq \ell - 1) &\leq \mathrm{W}(X \leq n-2) \\
&= \mathrm{W}\left(X \leq \frac{n-2}{\mathrm{E}(X)} \cdot \mathrm{E}(X)\right) \\
&= \mathrm{W}(X \leq (1 - 1/2) \cdot \mathrm{E}(X)) \\
&\leq \exp\left(-\frac{(1/2)^2 \cdot \mathrm{E}(X)}{2}\right) = \exp\left(-\frac{n-2}{4}\right)
\end{aligned}$$

nach oben abschätzen, wobei in die Rechnung eine Chernoff-Schranke eingeht (siehe Lemma A.6).

Die Menge $b(f(PFAD(1, \cdot)))$ enthält höchstens ν nicht dominierte Boxindizes (siehe Lemma 7.4). Wir halten aus den zugehörigen Boxen jeweils einen Pfad $q_j = (v_{j,1}, \ldots, v_{j,\ell_j})$ fest. Wir schließen nun mithilfe der booleschen Ungleichung (siehe Lemma A.4), dass die Wahrscheinlichkeit, dass die ℓ_j-te Verbesserung von einem q_j nicht innerhalb von t Mutationsschritten eintritt, höchstens $\nu \cdot \exp(-(n-2)/4) = \exp(-\Omega(n))$ beträgt, da

$$\nu = \mathcal{O}\left(n \cdot \left(\frac{n \cdot \log(n)}{\epsilon} + \frac{n \cdot \log(w^{\max})}{\epsilon}\right)^{k-1}\right)$$

polynomiell in n beschränkt ist. Also erreicht der Algorithmus DEMO eine $(1+\epsilon)$-Approximation der Paretofront innerhalb von t Generationen mit der Gegenwahrscheinlichkeit $1 - \exp(-\Omega(n))$. □

In dem Spezialfall $k = 1$, fällt die obere Schranke aus Theorem 7.6 zu $\mathcal{O}(n^3)$ zusammen. Wir erinnern an dieser Stelle daran, dass k angibt, wie viele Gewichte den einzelnen Kanten zugeordnet sind, und dass die zu optimierende Fitnessfunktion f

Pfade auf Zielpunkte aus $\mathbb{R}^d$ mit $d = n \cdot k$ abbildet. Wenn $k = 1$ gilt, dann bildet die Fitnessfunktion einen Pfad $p = (v_0, \ldots, v_\ell)$ auf den Zielpunkt $y = (y_1, \ldots, y_n)$ mit $y_i = w(p)$ falls $i = v_\ell$ und $y_i = c \cdot (n-1) \cdot w^{\max}$ falls $i \neq v_\ell$ ab. Demzufolge besteht die Paretofront in diesem Fall aus n Zielpunkten, wenn alle Knoten von der Quelle 1 aus erreichbar sind; die n Zielpunkte enthalten in den entsprechenden Komponenten die Gewichte der kürzesten Pfade von der Quelle 1 zu den n Senken $1 \leq i \leq n$. Eine einfache Betrachtung des letzten Beweises offenbart, dass in diesem Fall nicht nur eine $(1+\epsilon)$-Approximation der Paretofront sondern die gesamte Paretofront mit hoher Wahrscheinlichkeit in $\mathcal{O}(n^3)$ Generationen gefunden wird. Das gleiche Ergebnis gilt natürlich auch dann, wenn wir die Kantengewichte $w_i(\cdot)$ durch eine gewichtete Summe $\sum_{i=1}^{k} \lambda_i \cdot w_i(\cdot)$ ersetzen, um das multikriterielle in ein monokriterielles Problem zu transformieren.

Wir weisen darauf hin, dass Theorem 7.6 für alle initialen Populationen gilt, die den trivialen Pfad (1) enthalten. Wenn wir mit einer Population ohne (1) beginnen, dann müssen wir warten bis (1) der Population hinzugefügt wird, bevor wir Theorem 7.6 anwenden können, um die verbleibende Optimierungszeit abschätzen zu können. Es ist aus dem folgenden Grund offensichtlich, dass der Pfad (1) im Erwartungswert nach höchstens $\mathcal{O}(n \cdot \nu)$ Generationen gefunden wird. Wenn die aktuelle Population P einen Pfad umfasst, der aus $\ell > 0$ Kanten besteht, dann ist die Wahrscheinlichkeit, einen Pfad, der aus $\ell - 1$ Kanten besteht, zu erschaffen, durch $1/(2 \cdot e \cdot |P|)$ nach unten beschränkt. Die obere Schranke für die erwartete Optimierungszeit gilt also auch dann, wenn wir die Optimierung mit einer beliebigen nicht leeren Population beginnen.

7.5. Untere Schranken für die Optimierungszeit

In diesem Abschnitt untersuchen wir, wie viele Generationen mit hoher Wahrscheinlichkeit benötigt werden, um eine $(1+\epsilon)$-Approximation der Paretofront zu finden, wenn eine beliebig schwierige Eingabeinstanz zu optimieren ist. Dazu konstruieren wir eine konkrete Eingabeinstanz und beweisen eine untere Schranke für die Anzahl der Generationen, die der Algorithmus DEMO benötigt. Wir beginnen mit ein paar Vorbereitungen. Dann definieren wir die genannte Familie von Beispielinstanzen. Anschließend leiten wir eine untere Schranke für den allgemeinen Fall her, dass $k \in \mathbb{N}^+$ Gewichtsfunktionen gleichzeitig zu optimieren sind.

7.5.1. Vorbereitungen

Das nächste Lemma stellt zwei Ungleichungen für Summen von geometrisch verteilten Zufallsvariablen vor, die wir im Folgenden benötigen werden. Die Ungleichungen begrenzen die Wahrscheinlichkeit, dass eine Summe der genannten Zufallsvariablen einen Wert annimmt, der um den Faktor $(1+\delta)$ größer ist als ihr Erwartungswert.

Lemma 7.7. *Seien X_i, $1 \leq i \leq t$, unabhängige Zufallsvariable und $X := \sum_{i=1}^{t} X_i$.*

1. *Wenn* $\mathrm{W}(X_i = j) = (1-p)^{j-1} \cdot p$, $j \in \mathbb{N}^+$, *gilt, wobei* $0 < p < 1$ *ist, d. h., alle* X_i *folgen einer bei* 1 *beginnenden geometrischen Verteilung, dann gilt für alle* $\delta > 0$

$$\mathrm{W}(X \geq (1+\delta) \cdot \mathrm{E}(X)) \leq \exp\left(-\frac{\delta^2 \cdot t}{2 \cdot (1+\delta)}\right).$$

2. *Wenn* $\mathrm{W}(X_i = j) = (1-p)^{j} \cdot p$, $j \in \mathbb{N}$, *gilt, wobei* $0 < p < 1$ *ist, d. h., alle* X_i *folgen einer bei* 0 *beginnenden geometrischen Verteilung, dann gilt für alle* $\delta > 0$

$$\mathrm{W}(X \geq (1+\delta) \cdot \mathrm{E}(X)) \leq \exp\left(\frac{\delta^2 \cdot (1-p)^2 \cdot t}{2 \cdot (1+\delta \cdot (1-p))}\right).$$

Beweis. Wir beweisen die beiden Behauptungen nacheinander.

1. Wir leiten die erste Ungleichung mithilfe einer Chernoff-Ungleichung (siehe Lemma A.6) wie folgt her. Definiere die unabhängigen Zufallsvariablen Y_i, $1 \leq i \leq t'$, durch $\mathrm{W}(Y_i = 1) = p$ und $\mathrm{W}(Y_i = 0) = 1-p$, wobei $t' = \lceil (1+\delta) \cdot \mathrm{E}(X) \rceil$ ist. Sei $Y = \sum_{i=1}^{t'} Y_i$. Dann gelten $\mathrm{E}(X) = t/p$ und $\mathrm{E}(Y) = t' \cdot p \geq (1+\delta) \cdot t$. Betrachte die Umformungen

$$\begin{aligned}\mathrm{W}(X \geq (1+\delta) \cdot \mathrm{E}(X)) &\leq \mathrm{W}\left(Y \leq \frac{t}{\mathrm{E}(Y)} \cdot \mathrm{E}(Y)\right) \\ &\leq \mathrm{W}\left(Y \leq \left(1 - \frac{\delta}{1+\delta}\right) \cdot \mathrm{E}(Y)\right).\end{aligned}$$

   Eine Anwendung von Lemma A.6 führt zu

$$\begin{aligned}\mathrm{W}\left(Y \leq \left(1 - \frac{\delta}{1+\delta}\right) \cdot \mathrm{E}(Y)\right) &\leq \exp\left(-\frac{(\delta/(1+\delta))^2 \cdot \mathrm{E}(Y)}{2}\right) \\ &\leq \exp\left(-\frac{\delta^2 \cdot t}{2 \cdot (1+\delta)}\right).\end{aligned}$$

2. Die zweite Ungleichung kann mithilfe der ersten Ungleichung wie folgt hergeleitet werden. Definiere die unabhängigen Zufallsvariablen Y_i, $1 \leq i \leq t$, durch $\mathrm{W}(Y_i = j) = (1-p)^{j-1} \cdot p$, $j \in \mathbb{N}^+$. Sei $Y = \sum_{i=1}^{t} Y_i$. Dann gelten $\mathrm{E}(X) = t \cdot (1-p)/p$ und $\mathrm{E}(Y) = t/p$. Betrachte die Umformungen

$$\begin{aligned}\mathrm{W}(X \geq (1+\delta) \cdot \mathrm{E}(X)) &= \mathrm{W}(Y - t \geq (1+\delta) \cdot \mathrm{E}(Y - t)) \\ &= \mathrm{W}(Y \geq (1+\delta \cdot (1-p)) \cdot \mathrm{E}(Y)).\end{aligned}$$

   Eine Anwendung der ersten Ungleichung führt zu

$$\mathrm{W}(Y \geq (1+\delta \cdot (1-p)) \cdot \mathrm{E}(Y)) \leq \exp\left(\frac{\delta^2 \cdot (1-p)^2 \cdot t}{2 \cdot (1+\delta \cdot (1-p))}\right).$$

Insgesamt haben wir das Lemma bewiesen. □

7.5.2. Analysen

Wir betrachten die folgende Familie von Eingabeinstanzen, die von den drei Parametern k, n und b abhängen. Die Knoten der betrachteten Graphen können auf vier Komponenten $A = \{a_1, \ldots, a_\ell\}$, $B = \{b_{1,0}, \ldots, b_{1,b}, \ldots, b_{k-1,0}, \ldots, b_{k-1,b}\}$, $C = \{c_1, \ldots, c_\ell\}$ und $D = \{d_1, \ldots, d_\ell\}$ aufgeteilt werden. Um beispielsweise einen Pfad von dem Knoten a_1 zu dem Knoten c_ℓ zu finden, müssen die Komponenten A, B und C sukzessive durchlaufen werden. Die Komponente B stellt dabei sicher, dass es möglichst viele unvergleichbare Pfade von a_1 nach c_ℓ gibt. Außerdem sind alle Knoten aus A, B und C mit allen Knoten aus D verbunden, um das Vorankommen auf einem Pfad von a_1 nach c_ℓ zu erschweren. Wir werden die Eingabeinstanzen zunächst formal definieren und anschließend weitere wichtige Eigenschaften besprechen.

Definition 7.8. *Seien $k \in \mathbb{N}^+$, $n \in \mathbb{N}^+$ und $b \in \mathbb{N}^+$ positive natürliche Zahlen mit $\ell := (n-(k-1)\cdot(b+1))/3 \in \mathbb{N}^+$. Wir definieren den gewichteten gerichteten Graphen $G_{k,n,b} = (V, E, w)$ wie folgt. Sei die Knotenmenge $V = A \cup B \cup C \cup D$ durch*

- $A = \{a_i \mid (1 \leq i \leq \ell)\}$,
- $B = \{b_{i,j} \mid (1 \leq i < k) \wedge (0 \leq j \leq b)\}$,
- $C = \{c_i \mid (1 \leq i \leq \ell)\}$ *sowie*
- $D = \{d_i \mid (1 \leq i \leq \ell)\}$

definiert. Sei die Kantenmenge $E = E_A \cup E_B \cup E_C \cup E_D$ durch

- $E_A = \{(a_i, a_{i'}) \mid (1 \leq i < \ell) \wedge (i' = i + 1)\}$,
- $E_B = E'_B \cup E''_B$ *mit*
  - $E'_B = \{(a_i, b_{i',j'}) \mid (i = \ell) \wedge (i' = 1) \wedge (0 \leq j' \leq b)\}$ *und*
  - $E''_B = \{(b_{i,j}, b_{i',j'}) \mid (1 \leq i < k-1) \wedge (0 \leq j \leq b) \wedge (i' = i+1) \wedge (0 \leq j' \leq j)\}$,
- $E_C = E'_C \cup E''_C$ *mit*
  - $E'_C = \{(b_{i,j}, c_{i'}) \mid (i = k-1) \wedge (0 \leq j \leq b) \wedge (i' = 1)\}$ *und*
  - $E''_C = \{(c_i, c_{i'}) \mid (1 \leq i < \ell) \wedge (i' = i + 1)\}$ *sowie*
- $E_D = E'_D \cup E''_D \cup E'''_D$ *mit*
  - $E'_D = \{(a_i, d_{i'}) \mid (1 \leq i \leq \ell) \wedge (1 \leq i' \leq \ell)\}$,
  - $E''_D = \{(b_{i,j}, d_{i'}) \mid (1 \leq i < k) \wedge (0 \leq j \leq b) \wedge (1 \leq i' \leq \ell)\}$ *und*
  - $E'''_D = \{(c_i, d_{i'}) \mid (1 \leq i \leq \ell) \wedge (1 \leq i' \leq \ell)\}$

definiert. Sei die Gewichtsfunktion $w \colon E \to \mathbb{N}^k$ durch

- $w_d((a_i, b_{i',j'})) = 2^{b-j'}$ *falls* $d = 1$,
- $w_d((b_{i,j}, b_{i',j'})) = 2^{j-j'}$ *falls* $d = i'$,

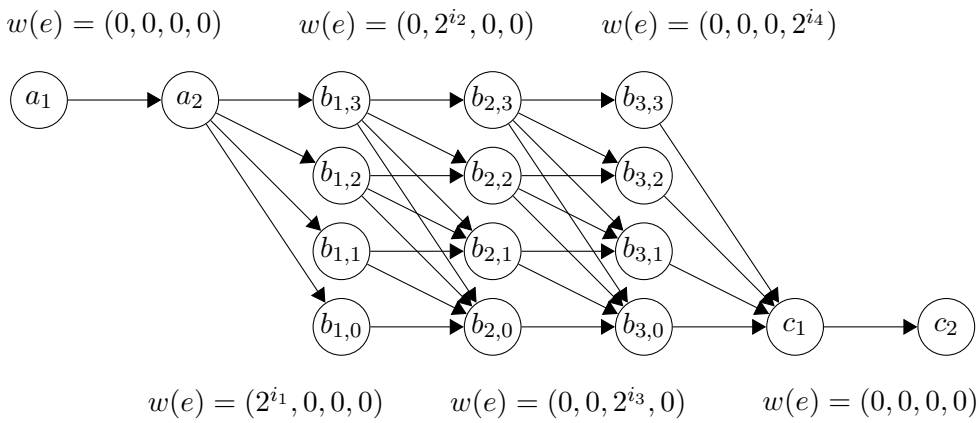


Abbildung 7.3.: Eingabeinstanz $G_{k,n,b}$ mit $k = 4$, $n = 18$, $b = 3$ und $\ell = 2$. Die Konstruktion des Graphen gewährleistet den Zusammenhang $w(PFAD(a_1, c_2)) = \{(2^{i_1}, 2^{i_2}, 2^{i_3}, 2^{i_4}) \mid i_1+i_2+i_3+i_4 = 3\}$. Beachte, dass die Knoten aus D und die Kanten aus E_D aus Darstellungsgründen nicht eingezeichnet sind.

- $w_d((b_{i,j}, c_{i'})) = 2^j$ *falls* $d = k$ *sowie*
- $w_d(e) = 0$ *anderenfalls*

definiert. Sei a_1 *die Quelle.*

Die in Definition 7.8 definierte Eingabeinstanz ist in Abbildung 7.3 skizziert. Beachte, dass die Gewichtsvektoren der Kanten, die nicht zu einem Knoten aus B oder $\{c_1\}$ führen, 0^k entsprechen und dass das maximale Gewicht 2^b ist. Die Gewichte der Kanten, die zu einem Knoten aus B oder $\{c_1\}$ führen, garantieren, dass

$$w(PFAD(a_i, c_{i'})) = \left\{ (2^{i_1}, \ldots, 2^{i_k}) \mid \sum_{j=1}^{k} i_j = b \right\}.$$

für alle $1 \leq i, i' \leq \ell$ gilt.

Im Folgenden werden wir uns auf Eingabeinstanzen mit einem angemessenen maximalen Gewicht beschränken, wobei wir hier unter „angemessen“ höchstens $2^{\mathcal{O}(n)}$ verstehen. Wir zeigen, dass es Eingabeinstanzen gibt, die bezogen auf das maximale Gewicht eine relativ große $(1 + \epsilon)$-Approximation der Paretofront erfordern. Das nächste Lemma stellt eine untere Schranke für die minimale Größe einer $(1 + \epsilon)$-Approximation der Paretofront vor.

Lemma 7.9. *Seien* $0 < \epsilon < 1$ *und* $w \leq 2^{\gamma \cdot n}$ *für eine Konstante* $\gamma < 1/(k-1)$. *Dann gibt es eine Eingabeinstanz mit* $w^{\max} \leq w$, *sodass die minimale Größe einer* $(1+\epsilon)$*-Approximation der Paretofront von* f *mit* $c > 1 + \epsilon$

$$\Omega(n \cdot (\log(w))^{k-1})$$

beträgt.

Beweis. Betrachte die in Definition 7.8 definierte Eingabeinstanz $G_{k,n,b}$ mit $b = \lfloor \log(w) \rfloor$. Folglich gilt für das maximale Gewicht $w^{\max} = 2^b = 2^{\lfloor \log(w) \rfloor} \leq w$. Aus der Konstruktion des Graphen folgt, dass es einen Pfad von der Quelle a_1 zu jedem Knoten aus C mit jedem Gewicht aus

$$W := \left\{ (2^{i_1}, \ldots, 2^{i_k}) \mid \sum_{j=1}^{k} i_j = b \right\}$$

gibt. Beachte, dass alle Gewichtsvektoren aus W paarweise unvergleichbar sind. Aufgrund von

$$\ell = \frac{n - (k-1) \cdot (b+1)}{3} \geq \frac{n - (k-1) \cdot (\gamma \cdot n + 1)}{3} = \Omega(n)$$

und

$$|W| = \binom{b+k-1}{k-1} \geq \left(\frac{b+k-1}{k-1} \right)^{k-1} \geq \left(\frac{b}{k-1} \right)^{k-1} = \Omega\left(\log(w) \right)^{k-1}$$

beträgt die Größe der Paretofront mindestens $\Omega(n \cdot (\log(w))^{k-1})$. Da alle Gewichtsvektoren aus W auch paarweise unvergleichbar bezüglich der $(1+\epsilon)$-Dominanz sind, muss eine $(1+\epsilon)$-Approximation der Paretofront alle genannten Gewichtsvektoren enthalten. Damit ist der Beweis komplett. □

Die Beobachtung, dass $\Omega(n \cdot (\log(w))^{k-1})$ eine untere Schranke für die Optimierungszeit des Algorithmus DEMO ist, folgt unmittelbar aus dem letzten Lemma, da der Algorithmus in jeder Generation höchstens einen Pareto-optimalen Pfad finden kann. Wir zeigen nun, dass die Optimierungszeit des betrachteten Algorithmus mit hoher Wahrscheinlichkeit um einen Faktor aus $\Omega(n^2)$ größer ist.

Theorem 7.10. *Seien $0 < \epsilon < 1$ und $w \leq 2^{\gamma \cdot n}$ für eine Konstante $\gamma < 1/(k-1)$. Wir wenden den Algorithmus DEMO mit Boxgröße $1 < r \leq 1+\epsilon$ auf die Fitnessfunktion $f = (f_{t,i})$: $PFAD(1, \cdot) \to \mathbb{N}^d$ mit $c > r$ an. Dann gibt es eine Eingabeinstanz mit $w^{\max} \leq w$, sodass $\mathrm{I}_{\mathrm{mul}}(P) > 1+\epsilon$ mit Wahrscheinlichkeit $1 - \exp(-\Omega(\sqrt{n}/\log(n)))$ für die ersten*

$$\frac{1}{12 \cdot (k-1)^{k-1}} \cdot (\ell - 1)^3 \cdot \lfloor \log(w) \rfloor^{k-1} = \Omega(n^3 \cdot (\log(w))^{k-1})$$

Populationen P gilt, wobei $\ell := (n - (k-1) \cdot (\lfloor \log(w) \rfloor + 1))/3 = \Omega(n)$ ist.

Beweis. Betrachte eine Eingabeinstanz $G_{k,n,b}$ mit $k \in \mathbb{N}^+$, $n \in \mathbb{N}^+$ und $b := \lfloor \log(w) \rfloor$. Dann gilt $\ell = (n - (k-1) \cdot (b+1))/3 = \Omega(n)$, weil wir aufgrund der geforderten Grenze für w

$$b = \lfloor \log(w) \rfloor \leq \lfloor \gamma \cdot n \rfloor \leq \gamma \cdot n < \frac{n}{k-1}$$

schließen können. Wir betrachten eine Phase der Länge $\Omega(n^3 \cdot (\log(w))^{k-1})$ und vollziehen einen typischen Lauf des Algorithmus DEMO nach. Wir erinnern daran, dass sich der Begriff „Phase“ auf eine bestimmte Anzahl aufeinander folgender Generationen bezieht. Wir zeigen, dass der Algorithmus DEMO mit hoher Wahrscheinlichkeit keine $(1+\epsilon)$-Approximation der Paretofront in der betrachteten Phase erreicht.

Betrachte die minimale Anzahl $\mathrm{pot}(P)$ von Kanten, die an einen beliebigen Pfad aus der aktuellen Population P angefügt werden müssen, um zu einem Pfad zu gelangen, der den Knoten a_ℓ enthält. Zu Beginn gilt sicherlich $\mathrm{pot}(P) = \ell - 1 = \Omega(n)$. Um das Potenzial $\mathrm{pot}(P)$ zu verringern, muss eine Kante zu dem nächsten Knoten aus A angefügt werden. Die Wahrscheinlichkeit, dies innerhalb einer Mutation zu erreichen, beträgt höchstens

$$\sum_{i=1}^{\infty} \frac{1}{e \cdot i!} \cdot \binom{i}{1} \cdot \frac{1}{2 \cdot (\ell+1)} = \frac{1}{2 \cdot (\ell+1)} \leq \frac{1}{\ell} = \mathcal{O}\Big(\frac{1}{n}\Big),$$

da jeder Knoten aus A mit jedem Knoten aus D verbunden ist. In die letzte Rechnung fließt außerdem $1/e \cdot \sum_{i=0}^{\infty} 1/i! = 1$ ein (siehe Lemma A.15). Wir überschätzen die Verringerung des Potenzials $\mathrm{pot}(P)$, wenn wir eine geometrische Verteilung mit dem Parameter $q := 1 - 1/\ell$ unterstellen. Betrachte eine Phase der Länge

$$t := \frac{(\ell-1)^2}{1+\sqrt{\ell}} = \Omega(n^{3/2})$$

und definiere die unabhängigen Zufallsvariablen X_i, $1 \leq i \leq t$, durch $\mathrm{W}(X_i = j) = (1-q)^j \cdot q$, $j \in \mathbb{N}$. Sei $X := \sum_{i=1}^{t} X_i$. Dann gilt offensichtlich

$$\mathrm{E}(X) = t \cdot \frac{1-q}{q} = \frac{(\ell-1)^2}{1+\sqrt{\ell}} \cdot \frac{1}{\ell-1} = \frac{\ell-1}{1+\sqrt{\ell}} = \Omega(\sqrt{n}),$$

da $\mathrm{E}(X_i) = (1-q)/q$ gilt. Folglich beträgt die Wahrscheinlichkeit, innerhalb der ersten t Generationen einen Pfad zu einem Knoten aus B oder C zu finden, höchstens

$$\begin{aligned}
\mathrm{W}(X \geq \ell - 1) &= \mathrm{W}\Big(X \geq \frac{\ell-1}{\mathrm{E}(X)} \cdot \mathrm{E}(X)\Big) \\
&= \mathrm{W}\Big(X \geq (1+\sqrt{\ell}) \cdot \mathrm{E}(X)\Big) \\
&\leq \exp\left(-\frac{\sqrt{\ell}^2 \cdot (1-q)^2 \cdot t}{2 \cdot (1+\sqrt{\ell} \cdot (1-q))}\right) = \exp\big(-\Omega(\sqrt{n})\big)
\end{aligned}$$

aufgrund der zweiten Ungleichung aus Lemma 7.7. Wir fahren unter der Annahme fort, dass das obige Ereignis innerhalb der ersten t Generationen nicht eingetreten ist.

Wir beweisen nun, dass innerhalb der obigen Phase der Länge t mit Wahrscheinlichkeit $1 - \exp(-\Omega(\sqrt{n}/\log(n)))$ ein Pfad zu jedem Knoten aus D konstruiert wird. Unter der Annahme, dass die Population Pfade zu $0 \leq i < \ell$ Knoten aus D enthält, beträgt die Wahrscheinlichkeit, einen Pfad zu einem unentdeckten Knoten aus

D zu erschaffen, mindestens $1/(2 \cdot e) \cdot 1/2 \cdot (\ell - i)/(2 \cdot (\ell + 1))$, da im ungünstigsten Fall die letzte Kante des Pfades, der zur Mutation ausgewählt wurde, entfernt und die Kante zu einem unentdeckten Knoten aus D angefügt werden muss. Folglich enthält die Population Pfade zu allen Knoten aus D im Erwartungswert nach $d := 8 \cdot e \cdot (\ell + 1) \cdot \sum_{i=1}^{\ell} 1/i = \mathcal{O}(n \cdot \log(n))$ Generationen. Mithilfe der Markoff-Ungleichung (siehe Lemma A.5) schließen wir, dass die Wahrscheinlichkeit, Pfade zu allen Knoten aus D in einer Phase der Länge $\lceil 2 \cdot d \rceil$ zu finden, größer als $1/2$ ist. Indem wir die obige Phase der Länge t in $\Omega(\sqrt{n}/\log(n))$ Unterphasen der Länge $\lceil 2 \cdot d \rceil$ einteilen, schließen wir, dass mit Wahrscheinlichkeit $1 - \exp(-\Omega(\sqrt{n}/\log(n)))$ Pfade zu allen Knoten aus D konstruiert werden. Im Folgenden nehmen wir an, dass die Population einen Pfad p zu jedem Knoten aus D mit $w(p) = 0^k$ enthält.

Wie wir im Beweis von Lemma 7.9 gesehen haben, enthält die Menge

$$W := \left\{ (2^{i_1}, \ldots, 2^{i_k}) \mid \sum_{j=1}^{k} i_j = b \right\}$$

mindestens $\Omega((\log(w))^{k-1})$ paarweise verschiedene Gewichtsvektoren. Wir bezeichnen diese Gewichtsvektoren mit $w_i \in \mathbb{N}^k$, $1 \leq i \leq |W|$. Um zu einer $(1 + \epsilon)$-Approximation der Paretofront zu gelangen, muss der Algorithmus DEMO einen Pfad zu jedem Knoten aus C mit jedem Gewichtsvektor aus W konstruieren.

Bevor wir uns den verschiedenen Gewichtsvektoren w_i zuwenden, beweisen wir, dass innerhalb von $\mathcal{O}(n^3 \cdot (\log(w))^{k-1})$ Generationen mit hoher Wahrscheinlichkeit höchstens $\lfloor \ell/3 \rfloor = \Omega(n)$ Kanten, die zu einem Knoten aus C führen, in einer einzelnen Mutation angefügt werden. Die Wahrscheinlichkeit, eine solche Kante innerhalb einer einzelnen Mutation hinzuzufügen beträgt höchstens $1/(2 \cdot (\ell + 1)) = \mathcal{O}(1/n)$. Folglich beträgt die Wahrscheinlichkeit, mehr als $\lfloor \ell/3 \rfloor$ Kanten hinzuzufügen, höchstens $\exp(-\Omega(n \cdot \log(n)))$. Mithilfe der booleschen Ungleichung (siehe Lemma A.4) schließen wir, dass innerhalb von $\mathcal{O}(n^3 \cdot (\log(w))^{k-1})$ Generationen mit Wahrscheinlichkeit $1 - \exp(-\Omega(n \cdot \log(n)))$ höchstens $\lfloor \ell/3 \rfloor$ der besagten Kanten in einer einzelnen Mutation angefügt werden, da die Anzahl der betrachteten Generationen polynomiell in n beschränkt ist. Wir gehen davon aus, dass das obige Ereignis eingetreten ist.

Halte einen Gewichtsvektor w_i fest und betrachte dann das Potenzial $\text{pot}_i(P) := \min\{\text{pot}_i(p) \mid p \in P\}$, wobei $\text{pot}_i(p)$ die Anzahl der Kanten misst, die an den Pfad p angefügt werden müssen, um einen Pfad zu erhalten, der zu dem letzten Knoten c_ℓ aus C führt und den Gewichtsvektor w_i aufweist. Wir warten bis das Potenzial höchstens $\ell - \lfloor \ell/3 \rfloor$ beträgt. Zu diesem Zeitpunkt ist das Potenzial größer als $\ell - 2 \cdot \lfloor \ell/3 \rfloor$, d. h., es gibt noch mindestens $\lceil \ell/3 \rceil = \Omega(n)$ bessere Potenzialwerte. Um das Optimierungsziel zu erreichen, muss $\text{pot}_i(P) = 0$ gelten.

Wir leiten eine obere Schranke für die Wahrscheinlichkeit, das Potenzial der aktuellen Population zu verringern, her. Um das Potenzial zu verringern, muss der Algorithmus einen Pfad p mit $\text{pot}_i(P) \leq \text{pot}_i(p) < \text{pot}_i(P) + \lfloor \ell/3 \rfloor$ zur Mutation auswählen und eine gewisse Anzahl Kanten anfügen, die zu dem nächsten Knoten aus C führen. Die Wahrscheinlichkeit, eine solche Kanten in einer einzelnen Mutation anzufügen, beträgt höchstens $1/(2 \cdot (\ell + 1)) \leq 1/\ell$. Wenn wir ein Individuum

$p \in P$ zur Mutation auswählen, dann beträgt die Wahrscheinlichkeit, das Potenzial $\mathrm{pot}_i(P)$ zu verringern, höchstens $(1/\ell)^{d+1}$, wobei $d := \mathrm{pot}_i(p) - \mathrm{pot}_i(P)$ mit $0 \leq d < \lfloor \ell/3 \rfloor$ ist. Wir haben bereits gezeigt, dass die aktuelle Population ℓ kürzeste Pfade zu allen Knoten aus D, die den Gewichtsvektor 0^k aufweisen, enthält. Folglich beträgt die Wahrscheinlichkeit, einen Pfad p mit einem bestimmten Potenzialwert $\mathrm{pot}_i(p)$ zwischen $\mathrm{pot}_i(P)$ und $\mathrm{pot}_i(P) + \lfloor \ell/3 \rfloor - 1$ zur Mutation auszuwählen, höchstens $1/(\ell+1) \leq 1/\ell$. Demzufolge beträgt die Wahrscheinlichkeit, das Potenzial zu verringern, höchstens

$$\sum_{i=0}^{\lfloor \ell/3 \rfloor - 1} \frac{1}{\ell} \cdot \left(\frac{1}{\ell}\right)^{i+1} \leq \frac{1}{\ell^2} \cdot \frac{\ell}{\ell - 1} = \frac{1}{(\ell-1)\cdot\ell} =: p.$$

Betrachte eine Phase der Länge $t := (t'+1)/(2 \cdot p) = \Omega(n^3)$, wobei $t' := (\ell-1)/6 = \Omega(n)$ ist. Definiere die unabhängigen Zufallsvariablen X_i, $1 \leq i \leq t$, durch $\mathrm{W}(X_i = 1) = p$ und $\mathrm{W}(X_i = 0) = 1 - p$. Sei $X := \sum_{i=1}^{t} X_i$. Dann gilt $\mathrm{E}(X) = t \cdot p = (t' + 1)/2$. Unter Verwendung dieser Zufallsvariablen beträgt die Wahrscheinlichkeit, das Potenzial mindestens $(t'+1)$-mal in der betrachteten Phase zu verringern, höchstens

$$\begin{aligned}
\mathrm{W}\big(X \geq t' + 1\big) &= \mathrm{W}\left(X \geq \frac{t'+1}{\mathrm{E}(X)} \cdot \mathrm{E}(X)\right) \\
&= \mathrm{W}(X \geq (1+1) \cdot \mathrm{E}(X)) \\
&\leq \exp\left(-\frac{\min\{1, 1^2\} \cdot \mathrm{E}(X)}{3}\right) = \exp(-\Omega(n)),
\end{aligned}$$

wobei hier eine Chernoff-Ungleichung (siehe Lemma A.6) einfließt.

Wenn das Potenzial $\mathrm{pot}_i(P)$ verringert wird, dann überschätzen wir die Verringerung, wenn wir eine geometrische Verteilung mit Parameter $q := 1 - 1/\ell$ unterstellen. Definiere die unabhängigen Zufallsvariablen Y_i, $1 \leq i \leq t'$, durch $\mathrm{W}(Y_i = j) = (1-q)^{j-1} \cdot q$ für alle $j \in \mathbb{N}^+$. Sei $Y = \sum_{i=1}^{t'} Y_i$. Dann gilt $\mathrm{E}(Y) = t' \cdot \ell/(\ell-1) = \ell/6$. Folglich beträgt die Wahrscheinlichkeit, $\mathrm{pot}_i(P) = 0$ in t Generationen zu erreichen höchstens

$$\begin{aligned}
\mathrm{W}\left(Y \geq \left\lceil \frac{\ell}{3} \right\rceil\right) &\leq \mathrm{W}\left(Y \geq \frac{\ell}{3 \cdot \mathrm{E}(Y)} \cdot \mathrm{E}(Y)\right) \\
&\leq \mathrm{W}(Y \geq (1+1) \cdot \mathrm{E}(Y)) \\
&\leq \exp\left(-\frac{1^2 \cdot t'}{2 \cdot (1+1)}\right) = \exp(-\Omega(n)),
\end{aligned}$$

wobei hier die erste Ungleichung aus Lemma 7.7 einfließt.

Wir erinnern an dieser Stelle daran, dass wir weiter oben bewiesen haben, dass in der betrachteten Phase mit hoher Wahrscheinlichkeit höchstens $\lfloor \ell/3 \rfloor$ Kanten, die zu einem Knoten aus C führen, in einer einzelnen Mutation hinzugefügt werden. Wenn wir annehmen, dass $\mathrm{pot}_i(P) \leq \ell - \lfloor \ell/3 \rfloor$ für alle Potenziale $\mathrm{pot}_i(P)$ gilt, dann kann, nachdem ein Individuum zur Mutation ausgewählt worden ist, folglich

höchstens ein Potenzial $\text{pot}_i(P)$ potenziell verringert werden. Indem wir nun alle $|W|$ Gewichtsvektoren berücksichtigen, erhalten wir das Resultat, dass der Algorithmus DEMO mit hoher Wahrscheinlichkeit mindestens

$$\begin{aligned} t \cdot |W| &= \frac{(\ell-1)/6+1}{2 \cdot 1/((\ell-1)\cdot\ell)} \cdot \binom{b+k-1}{k-1} \\ &\geq \frac{1}{12 \cdot (k-1)^{k-1}} \cdot (\ell-1)^3 \cdot \lfloor \log(w) \rfloor^{k-1} = \Omega(n^3 \cdot (\log(w))^{k-1}) \end{aligned}$$

Generationen benötigt, um seine initiale Population in eine $(1+\epsilon)$-Approximation der Paretofront zu transformieren. Der Begriff „hohe Wahrscheinlichkeit“ bezieht sich in unserem Fall auf $1-\exp(-\Omega(\sqrt{n}/\log(n)))$, da sich alle Fehlerwahrscheinlichkeiten zu $\exp(-\Omega(\sqrt{n}/\log(n)))$ aufsummieren. □

Im Rahmen des letzten Theorems haben wir eine Approximationsgüte $1+\epsilon$ mit $0 < \epsilon < 1$ und eine Gewichtsschranke w mit $w \leq 2^{\gamma \cdot n}$, wobei $\gamma < 1/(k-1)$ eine Konstante ist, betrachtet. Beachte, dass die Forderung $\epsilon < 1$ nicht sehr ambitioniert ist und dass $2^{\gamma \cdot n}$ eine relativ schwache Begrenzung der Gewichte darstellt. Theorem 7.10 stellt sicher, dass es eine Eingabeinstanz mit $w^{\max} \leq w$ gibt, sodass der Algorithmus DEMO mit Boxgröße $1 < r \leq 1+\epsilon$ angewandt auf die Fitnessfunktion $f = (f_{t,i})\colon PFAD(1,\cdot) \to \mathbb{N}^d$ mit $c > r$ mit hoher Wahrscheinlichkeit mindestens

$$\Omega(n^3 \cdot (\log(w))^{k-1})$$

Generationen benötigt, um eine $(1+\epsilon)$-Approximation der Paretofront zu produzieren. Wir weisen darauf hin, dass das letzte Theorem alle sinnvollen Werte für r und c abdeckt. Wenn $w = \Omega(n)$ und $\epsilon = \Omega(1)$ gelten, d. h., die Gewichtsschranke und die Approximationsgüte nicht zu klein sind, dann vereinfacht sich die untere Schranke zu

$$\Omega\left(n^3 \cdot \left(\frac{\log(n \cdot w)}{\epsilon}\right)^{k-1}\right). \tag{7.5}$$

In diesem Fall beträgt die Lücke zwischen der oberen Schranke aus Theorem 7.6 und der unteren Schranke aus Formel 7.5 für die Optimierungszeit genau $\Theta(n^{k-1})$. Es ist ein offenes Problem, diese Lücke für $k > 1$ zu schließen.

7.6. Fazit

Wir haben in diesem Kapitel einen grundlegenden evolutionären Algorithmus mit einer Population veränderlicher Größe untersucht und gezeigt, dass dieser Algorithmus, wenn er auf einem naheliegenden Modell eines relevanten und NP-schwierigen Problems operiert, ein FPRAS ergibt. Wenn wir die obere Schranke für die Optimierungszeit des evolutionären Algorithmus (siehe Theorem 7.6) mit der oberen Schranke

$$\mathcal{O}\left(n \cdot m \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)^{k-1}\right)$$

für die Laufzeit des in [Tsaggouris und Zaroliagis, 2006] vorgestellten FPTAS vergleichen, stellen wir fest, dass beide Schranken bis auf den Faktor $\Theta(n^2/m)$ miteinander übereinstimmen. Der Algorithmus aus [Tsaggouris und Zaroliagis, 2006] stellt übrigens das schnellste bekannte FPTAS für das betrachtetet Problem dar. Für dichte Graphen beträgt der Unterschied nur noch $\Theta(1)$. Wir erinnern daran, dass n und m die Anzahl der Knoten bzw. Kanten bezeichnen. Weiterhin steht k für die Anzahl der Kantengewichte, $w^{\max}$ für das maximale Gewicht und $1+\epsilon$ für die gewünschte Approximationsgüte. Es erscheint plausibel, dass dieser Unterschied den wahren Unterschied zwischen der Optimierungszeit und der Laufzeit widerspiegelt. Während der evolutionäre Algorithmus auf einen zufälligen Variationsoperator zurückgreift, um den Entscheidungsraum zu durchsuchen, durchsucht das FPTAS den Entscheidungsraum systematisch. Wir erkennen, dass der evolutionäre Algorithmus, obwohl er auf zufällige Weise Lösungen erzeugt, das Optimierungsziel mit vergleichbarem Erfolg erreicht. Jedoch benötigt eine naheliegende Implementierung des vorgeschlagenen evolutionären Algorithmus im Erwartungswert

$$\mathcal{O}\left(n^2 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\epsilon}\right)^{k-1}\right)$$

Rechenschritte für die Simulation einer einzigen Generation. Die praktische Anwendbarkeit des evolutionäre Algorithmus leidet unter diesem zusätzlichen Faktor.

Um die Optimierungszeit weiter einzugrenzen, haben wir eine Eingabeinstanz präsentiert, deren Optimierung für den evolutionären Algorithmus relativ schwierig ist. Anschließend haben wir eine untere Schranke für die Optimierungszeit des betrachteten evolutionären Algorithmus auf der genannten Eingabeinstanz bewiesen, die mit hoher Wahrscheinlichkeit eingehalten wird. Unter einigen relativ schwachen Annahmen, konnte die verbleibende Lücke zwischen der oberen und der unteren Schranke auf $\Theta(n^{k-1})$ eingegrenzt werden.

Es wäre wünschenswert, die Lücke zwischen beiden Schranken zu verkleinern bzw. idealerweise zu schließen. Es wäre weiterhin interessant, zu untersuchen, wie sich andere allgemeine randomisierte Suchheuristiken auf dem untersuchten Problem verhalten. Ameisenalgorithmen für verschiedene Kürzeste-Wege-Probleme werden hier in Kapitel 9 und 10 behandelt. Ein weiteres sehr interessantes Feld für zukünftige Untersuchungen stellt die Analyse der durchschnittlichen Optimierungszeit des betrachteten evolutionären Algorithmus bezüglich angemessener Verteilungen der Eingabeinstanzen dar.

8. Dynamische Programmierung und evolutionäre Algorithmen

8.1. Einleitung

In diesem wie auch im letzten Kapitel untersuchen wir evolutionäre Algorithmen für kombinatorische Optimierungsprobleme. Die Ergebnisse, die wir kennenlernen werden, sind jedoch allgemeiner, da sie nicht auf ein festes Problem zugeschnitten sind. Wir widmen uns hier insbesondere der wichtigen Frage, wie der Suchraum und die zugehörige Fitnessfunktion beschaffen sein müssen, damit ein evolutionärer Algorithmus eine optimalen Lösung effizient finden kann. Die Wahl eines geeigneten Suchraumes bildet den ersten wichtigen Schritt in dem Prozess, ein gegebenes Problem so zu modellieren, dass es von einen evolutionären Algorithmus erfolgreich gelöst werden kann. Diesem Thema wurde in der Literatur über evolutionäre Algorithmen bereits viel Aufmerksamkeit zuteil. Beispielsweise wurden verschiedene Repräsentationen für das Problem des Handlungsreisenden (siehe z. B. [Michalewicz und Fogel, 2004]) und für einige NP-schwierige Varianten des Problems, einen minimalen aufspannenden Baum zu finden, (siehe z. B. [Raidl und Julstrom, 2003]) untersucht.

Jede dieser Repräsentationen induziert in Verbindung mit den eingesetzten Variationsoperatoren des evolutionären Algorithmus – Mutation und Rekombination – eine charakteristische Nachbarschaft der Suchpunkte. Die genannten Faktoren beeinflussen, wie wahrscheinlich es ist, dass ein bestimmter Suchpunkt erzeugt wird, wenn eine bestimmte Menge von Suchpunkten die aktuelle Population bildet. Die Repräsentation der Lösungskandidaten wird häufig so gewählt, dass die Suchpunkte gültigen Lösungen für das zu optimierende Problem entsprechen. Wenn es Suchpunkte gibt, die ungültigen Lösungen entsprechen, dann kann der Suchprozess beispielsweise durch den Einsatz einer geeigneten Straffunktion in Bereiche des Suchraumes gelenkt werden, in denen gültige Lösungen zu finden sind.

Kürzlich konnte für verschiedene kombinatorische Optimierungsprobleme bewiesen werden, dass diese relativ effizient mithilfe von evolutionären Algorithmen gelöst werden können, wenn geeignete Repräsentationen und Mutationsoperatoren zum Einsatz kommen. Beispiele für diesen Ansatz sind in [Doerr et al., 2008a] und [Theile, 2009] für das Kürzeste-Wege-Problem bzw. das Problem des Handlungsreisenden zu finden. Die Repräsentationen, die in den genannten Arbeiten verwendet werden, unterscheiden sich deutlich von dem Suchraum $\mathbb{B}^n$, der den ersten theoretischen Untersuchungen der Optimierungszeit von evolutionären Algorithmen zugrunde liegt. Die gewählten Repräsentationen sind vielmehr so beschaffen, dass sie dem evolutionären Algorithmus ermöglichen, Teillösungen des zu optimierenden Problems zu

finden und diese nach und nach in optimale Lösungen zu transformieren. Durch die Wahl einer geeigneten Fitnessfunktion kann dann sichergestellt werden, dass die evolutionären Algorithmen das zu behandelnde Problem prinzipiell dem Paradigma der dynamischen Programmierung folgend lösen.

Dynamische Programmierung [Bellman und Dreyfus, 1962] ist eines der bekanntesten Entwurfsmuster für Algorithmen, mit dessen Hilfe eine Vielzahl von algorithmischen Problemen gelöst werden kann. Allgemeine Rahmen für dynamische Programme wurden beispielsweise von Woeginger [2000] und Kogan [2004] vorgestellt. Das genannte Paradigma ermöglicht den Entwurf von effizienten Algorithmen, die Probleme optimal lösen, indem sie Teillösungen schrittweise zu einer optimalen Lösung für das Gesamtproblem zusammensetzen. Wir werden in diesem Kapitel letztlich eine Verbindung zwischen den genannten evolutionären Algorithmen aus [Doerr et al., 2008a, Theile, 2009] und dynamischen Programmen herstellen.

Evolutionäre Algorithmen werden häufig eingesetzt, um „schwierige“ Probleme zu lösen. Dabei kann ein vorliegendes Problem tatsächlich inhärent komplex sein oder die mit der Lösung des Problems beauftragte Person hält ein eigentlich einfaches Problem nur für kompliziert. Im zweiten Fall überbringt das vorliegende Kapitel eine positive Botschaft: Wenn das Problem effizient von einem dynamischen Programm gelöst werden kann, dann können „natürliche“ evolutionäre Algorithmen das häufig auch. Wir gehen in zwei Schritten vor. Zunächst zeigen wir formal, wie erfolgreiche evolutionäre Algorithmen für Probleme, die mithilfe dynamischer Programmierung effizient lösbar sind, entworfen werden können. Wir beweisen insbesondere, dass es für ein Problem, das von einem dynamischen Programm in der Zeit T gelöst werden kann, einen evolutionären Algorithmus gibt, der das Problem im Erwartungswert in der Zeit $\mathcal{O}(T \cdot s \cdot \log(|\mathrm{DP}|))$ löst, wobei s die Anzahl der Stufen und $|\mathrm{DP}|$ die Anzahl der Zustände des dynamischen Programms bezeichnen (siehe Abschnitt 8.3). Anschließend vollziehen wir das Entwurfsmuster anhand von drei konkreten Beispielen für kombinatorische Optimierungsprobleme nach. Dabei werden wir sehen, dass die entstehenden evolutionären Algorithmen sehr natürlich sind. Auf diese Weise belegen wir, dass der Erfolg von evolutionären Algorithmen häufig darauf zurückzuführen ist, dass sie dynamische Programme „nachempfinden“.

Dieses Kapitel ist wie folgt aufgebaut. In Abschnitt 8.2 stellen wir einen allgemeinen Rahmen für die Probleme, die wir hier betrachten möchten, und für dynamische Programme vor. Die dynamischen Programme werden in Abschnitt 8.3 mit evolutionären Algorithmen in Verbindung gebracht. In Abschnitt 8.4 zeigen wir, wie evolutionäre Algorithmen für bekannte kombinatorische Optimierungsprobleme konstruiert werden können, deren Verhalten an die Arbeitsweise dynamischer Programme angelehnt ist. Wir beschließen dieses Kapitel mit einem kurzen Fazit.

8.2. Dynamische Programmierung

Dynamische Programmierung ist ein allgemeines Entwurfsmuster für Algorithmen. Die grundlegende Idee besteht darin, ein Problem in Teilprobleme der gleichen Art

aufzuteilen und eine Lösung des Originalproblems aus den Lösungen der Teilprobleme zusammenzusetzen. Mithilfe dynamischer Programmierung konnten in den vergangenen fast 50 Jahren viele mono- wie auch multikriterielle Optimierungsprobleme effizient gelöst werden. Für einige Probleme folgen sogar die besten bekannten Algorithmen diesem Ansatz.

8.2.1. Betrachtete Probleme

Wir betrachten in diesem Kapitel multikriterielle Optimierungsprobleme P, wobei $k \in \mathbb{N}^+$ die Anzahl der zu optimierenden Ziele bezeichnet. Wir können die genannten Optimierungsprobleme durch eine Zielfunktion $g\colon \mathcal{S} \to \mathbb{R}^k$ darstellen. Wir weisen darauf hin, dass auch monokriterielle Optimierungsprobleme in diesen Rahmen passen ($k = 1$).

Wir bezeichnen in diesem Kapitel Pareto-Dominanz durch $\succeq$, wobei $(y'_1, \ldots, y'_k) \succeq (y_1, \ldots, y_k)$ genau dann gilt, wenn $y'_i \leq y_i$ (Minimierungsprobleme) bzw. $y'_i \geq y_i$ (Maximierungsprobleme) für alle $1 \leq i \leq k$ gilt. Das Ziel besteht darin, eine minimale Teilmenge von $\mathcal{S}$ zu ermitteln, die auf die Paretofront abgebildet wird.

8.2.2. Rahmen für dynamische Programme

Betrachte ein dynamisches Programm DP für ein Optimierungsproblem P. Wir nehmen an, dass DP neue Teillösungen auf $s \in \mathbb{N}$ Stufen generiert. Die Teillösungen, die auf der i-ten Stufe erzeugt werden, bezeichnen wir mit $\mathcal{S}_i \subseteq \mathcal{S}$. Wir benutzen s endliche Mengen $\mathcal{F}_i$ von Abbildungen $F\colon \mathcal{S} \to \mathcal{S}'$ und eine Abbildung $H\colon \mathcal{S}' \to \mathbb{R}$ mit $S \in \mathcal{S}$ falls $H(S) \leq 0$, um DP zu beschreiben, wobei $\mathcal{S}'$ eine Obermenge des Originalsuchraumes $\mathcal{S}$ bezeichnet. Diese Modifikation ist aus formalen Gründen erforderlich, da wir zulassen möchten, dass die Abbildungen F Zustände erzeugen können, die keinem gültigen Suchpunkt entsprechen. Die Elemente der Menge $\mathcal{S}'$ nennen wir hier Zustände. Wir nehmen an, dass s und alle $\mathcal{F}_i$ von der Länge der Eingabeinstanz abhängen und H von der Eingabeinstanz abhängt. Wir nennen die verschiedenen F *Zustandsüberführungsfunktionen* und H *Gültigkeitsfunktion*.

Das dynamische Programm DP arbeitet wie folgt. In der Initialisierungsphase wird der Zustandsraum $\mathcal{S}_0$ auf eine endliche Teilmenge von $\mathcal{S}$ gesetzt. Auf der i-ten Stufe werden die Elemente des Zustandsraumes $\mathcal{S}_{i-1}$ als Teillösungen von P aufgefasst, die zu gültigen Lösungen von P ergänzt werden können. Auf Stufe i wird der Zustandsraum $\mathcal{S}_i$ auf $\mathcal{S}_{i-1}$ aufbauend durch

$$\mathcal{S}_i = \{F(S) \mid S \in \mathcal{S}_{i-1} \wedge F \in \mathcal{F}_i \wedge H(F(S)) \leq 0\} \tag{8.1}$$

bestimmt. In diesem Prozess sorgt die Gültigkeitsfunktion H dafür, dass ungültige Teillösungen, die in Phase i generiert werden, nicht in den aktuellen Zustandsraum $\mathcal{S}_i$ aufgenommen werden.

Um die Laufzeit zu verbessern, verwenden die meisten dynamischen Programme das sogenannte Bellmanprinzip (siehe z. B. [Bellman und Dreyfus, 1962]). Die grundlegende Idee besteht darin, schlechte Teillösungen so früh wie möglich zu erkennen

und fortan nicht weiter zu betrachten. Ein solches Vorgehen ist sinnvoll, wenn die Optimalität der endgültigen Lösungsmenge dadurch nicht in Mitleidenschaft gezogen wird. Wir beschreiben dieses Prinzip als eine Dominanzrelation auf den Zuständen in den verschiedenen Zustandsräumen $\mathcal{S}_i$.

Wir nennen $\mathrm{EXT}_i(S) \in \mathcal{S}_s$ eine *gültige Erweiterung* von $S \in \mathcal{S}_i$ wenn es $F_j \in \mathcal{F}_j$, $i+1 \leq j \leq s$, mit

$$\mathrm{EXT}_i(S) = F_s(F_{s-1}(\dots F_{i+1}(S)\dots))$$

gibt und

$$H(F_j(F_{j-1}(\dots F_{i+1}(S)\dots))) \leq 0$$

für alle $i+1 \leq j \leq s$ gilt. Betrachte zwei Zustände $S, S' \in \mathcal{S}_i$. Wir nehmen an, dass es für alle gültigen Erweiterungen $\mathrm{EXT}_i(S)$ von S zu einer gültigen Lösung eine gültige Erweiterung $\mathrm{EXT}_i(S')$ von S' zu einer gültigen Lösung gibt, sodass $g(\mathrm{EXT}_i(S'))$ mindestens so gut wie $g(\mathrm{EXT}_i(S))$ ist, d. h., $g(\mathrm{EXT}_i(S')) \succeq g(\mathrm{EXT}_i(S))$ gilt. In diesem Fall sprechen wir davon, dass der Zustand $S' \in \mathcal{S}_i$ den Zustand $S \in \mathcal{S}_i$ *bezüglich Erweiterungen dominiert*. Ein Zustand ohne gültige Erweiterungen wird definitionsgemäß von allen Zuständen bezüglich Erweiterungen dominiert. Die geschilderte Situation ermöglicht es nun, den Zustand S fortzulassen, ohne die Optimalität der endgültigen Lösungsmenge zu beeinträchtigen.

Um die obige Dominanzrelation definitionsgemäß zu prüfen, müssen alle gültigen Erweiterungen betrachtet werden. Dieses Vorgehen verbietet sich offensichtlich aus Effizienzgründen. Um das Ausschließen schlechter Zustände zu vereinfachen, betrachten wir eine Halbordnung $\succeq_{\mathrm{dom}}$ auf $\mathcal{S}$. Wir werden sehen, dass eine Halbordnung $\succeq_{\mathrm{dom}}$, die Dominanz bezüglich Erweiterungen induziert, benutzt werden kann, um schlechte Zustände auszuschließen. Wir sagen, dass ein Zustand S' einen Zustand S *dominiert* genau dann, wenn $S' \succeq_{\mathrm{dom}} S$ gilt. Betrachte eine Teilmenge $\mathcal{S}_i$ von $\mathcal{S}$. Wir nennen $\mathcal{T}_i \subseteq \mathcal{S}_i$ eine *dominierende Teilmenge* von $\mathcal{S}_i$ bezüglich $\succeq_{\mathrm{dom}}$ genau dann, wenn es für alle Zustände $S \in \mathcal{S}_i$ einen Zustand $S' \in \mathcal{T}_i$ mit $S' \succeq_{\mathrm{dom}} S$ gibt. Wir verwenden die Notation $M(\mathcal{S}_i, \succeq_{\mathrm{dom}})$, um die Menge zu bezeichnen, die alle dominierenden Teilmengen von $\mathcal{S}_i$ enthält, deren Größe minimal ist.

Es ist für die Laufzeit des dynamischen Programms von entscheidender Bedeutung, dass die Halbordnung $\succeq_{\mathrm{dom}}$ effizient entschieden werden kann. Wir geben uns daher mit Halbordnungen $\preceq_{\mathrm{dom}}$ zufrieden, die durch

$$S' \succeq_{\mathrm{dom}} S \iff f(S') \succeq_{\mathrm{par}} f(S)$$

induziert werden, wobei $f\colon \mathcal{S} \to \mathbb{R}^d$ eine Funktion und $\succeq_{\mathrm{par}}$ eine Halbordnung auf $\mathbb{R}^d$ sind. Wir nehmen an, dass f effizient ausgewertet und $\succeq_{\mathrm{par}}$ effizient entschieden werden können.

Die folgende Proposition zeigt, dass es genügt, eine dominierende Teilmenge der Zustände, die in jeder Phase i konstruiert werden, zu verwahren. Es ist folglich nicht notwendig alle Zustände aus $\mathcal{S}_i$ weiter zu betrachten.

Proposition 8.1. *Seien $\mathcal{T}_i \subseteq \mathcal{S}$, $0 \leq i \leq s$, mit*

$$\mathcal{T}_0 \in M(\mathcal{S}_0, \succeq_{\mathrm{dom}})$$

und

$$\mathcal{T}_i \in M(\{F(S) \mid S \in \mathcal{T}_{i-1} \wedge F \in \mathcal{F}_i \wedge H(F(S)) \leq 0\}, \succeq_{\text{dom}})$$

für alle $1 \leq i \leq s$ *gegeben. Wenn für alle* $S, S' \in \mathcal{S}_i$ *aus* $S' \succeq_{\text{dom}} S$ *folgt, dass* S *von* S' *bezüglich Erweiterungen dominiert wird, dann gilt*

$$\max(g(\mathcal{T}_s), \succeq) = \max(g(\mathcal{S}_s), \succeq).$$

Beweis. Die Aussage $\max(g(\mathcal{T}_s), \succeq) \subseteq \max(g(\mathcal{S}_s), \succeq)$ folgt aus $\mathcal{T}_i \subseteq \mathcal{S}_i$ für alle $0 \leq i \leq s$. Wir müssen noch die Aussage $\max(g(\mathcal{S}_s), \succeq) \subseteq \max(g(\mathcal{T}_s), \succeq)$ beweisen. Sei $y \in \max(g(\mathcal{S}_s), \succeq)$ beliebig. Wir zeigen für alle $0 \leq i \leq s$, dass es einen Zustand $S \in \mathcal{T}_i$ und eine gültige Erweiterung $\text{EXT}_i(S)$ von S mit $g(\text{EXT}_i(S)) = y$ gibt. Wir beweisen die Aussagen mithilfe einer vollständigen Induktion.

Aus $y \in \max(g(\mathcal{S}_s), \succeq)$ folgt, dass ein Zustand $S' \in \mathcal{S}_0$ und eine gültige Erweiterung $\text{EXT}_0(S')$ von S' mit $g(\text{EXT}_0(S')) = y$ existieren. Aus $\mathcal{T}_0 \in M(\mathcal{S}_0, \succeq_{\text{dom}})$ folgt, dass es einen Zustand $S \in \mathcal{T}_0$ mit $S \succeq_{\text{dom}} S'$ gibt. Also existiert eine gültige Erweiterung $\text{EXT}_0(S)$ von S mit $g(\text{EXT}_0(S)) = y$.

Sei $S' \in \mathcal{T}_{i-1}$ ein Zustand, für den es eine gültige Erweiterung $\text{EXT}_{i-1}(S')$ mit $g(\text{EXT}_{i-1}(S')) = y$ gibt. Dann existieren $F_j \in \mathcal{F}_j$, $i \leq j \leq s$, mit

$$\text{EXT}_{i-1}(S') = F_s(F_{s-1}(\dots F_i(S') \dots))$$

und

$$H(F_j(F_{j-1}(\dots F_i(S') \dots))) \leq 0$$

für alle $i \leq j \leq s$. Also gilt $F_i(S') \in \mathcal{S}_i$. Daraus folgt, dass es einen Zustand $S \in \mathcal{T}_i$ mit $S \succeq_{\text{dom}} F_i(S')$ gibt. Also existiert eine gültige Erweiterung $\text{EXT}_i(S)$ von S mit $g(\text{EXT}_i(S)) = y$. □

Beachte, dass die Größe der $\mathcal{T}_i$ eindeutig bestimmt ist. Die letzte Proposition impliziert, dass die Paretofront von g in $g(\mathcal{T}_s)$ enthalten ist, wenn sie in $g(\mathcal{S}_s)$ enthalten ist.

Wir betrachten zusätzlich die folgenden drei Bedingungen, die wünschenswerte Eigenschaften der Dominanzrelation formulieren.

Bedingung 8.2 (C.1)**.** *Seien* $S, S' \in \mathcal{S}_i$*. Wenn* $S \preceq_{\text{dom}} S'$ *gilt, dann gilt* $F(S) \preceq_{\text{dom}} F(S')$ *für alle* $F \in \mathcal{F}_{i+1}$*.*

Bedingung 8.2 stellt sicher, dass sich die Dominanzrelation zwischen zwei Zuständen von einer Stufe auf die nächste überträgt.

Bedingung 8.3 (C.2)**.** *Seien* $S, S' \in \mathcal{S}_i$*. Wenn* $S \preceq_{\text{dom}} S'$ *und* $H(S) \leq 0$ *gelten, dann gilt* $H(S') \leq 0$*.*

Bedingung 8.3 drückt aus, dass ungültige Zustände niemals gültige Zustände dominieren.

Bedingung 8.4 (C.3)**.** *Seien* $S, S' \in \mathcal{S}_s$*. Wenn* $S \preceq_{\text{dom}} S'$ *gilt, dann gilt* $g(S) \preceq g(S')$*.*

Algorithmus 6 Dynamisches Programm für g

$\mathcal{T}_0 \leftarrow \emptyset$
for $S \in \mathcal{S}_0$ **do**
  if $\nexists S' \in \mathcal{T}_0 \colon S' \succeq_{\text{dom}} S$ **then**
    $\mathcal{T}_0 \leftarrow (\mathcal{T}_0 \setminus \{S' \in \mathcal{T}_0 \mid S \succ_{\text{dom}} S'\}) \cup \{S\}$
  end if
end for
for $i = 1, \ldots, s$ **do**
  $\mathcal{T}_i \leftarrow \emptyset$
  for $S \in \mathcal{T}_{i-1}$ und $F \in \mathcal{F}_i$ **do**
    if $H(F(S)) \leq 0$ und $\nexists S' \in \mathcal{T}_i \colon S' \succeq_{\text{dom}} F(S)$ **then**
      $\mathcal{T}_i \leftarrow (\mathcal{T}_i \setminus \{S' \in \mathcal{T}_i \mid F(S) \succ_{\text{dom}} S'\}) \cup \{F(S)\}$
    end if
  end for
end for
return $\{S \in \mathcal{T}_s \mid \nexists S' \in \mathcal{T}_s \colon g(S') \succ g(S)\}$

Bedingung 8.4 stellt eine Verbindung zwischen der Dominanzrelation und der zu optimierenden Zielfunktion g her. Sie sagt aus, dass auf der letzten Stufe die Dominanz $S \preceq_{\text{dom}} S'$ von Zuständen S, S' die Paretodominanz $g(S) \preceq g(S')$ der zugehörigen Funktionswerte $g(S), g(S')$ nach sich zieht.

Die nächste Proposition zeigt, dass die Dominanzrelation $\preceq_{\text{dom}}$ Dominanz bezüglich Erweiterungen impliziert, wenn Bedingung 8.2, 8.3 und 8.4 erfüllt sind.

Proposition 8.5. *Seien Bedingung 8.2, 8.3 und 8.4 für $\preceq_{\text{dom}}$ erfüllt. Seien $i \in \{0, \ldots, s\}$ und $S, S' \in \mathcal{S}_i$. Dann dominiert der Zustand S' den Zustand S bezüglich Erweiterungen, wenn $S' \succeq_{\text{dom}} S$ gilt.*

Beweis. Wenn auf Stufe i der Zustand S keine gültige Erweiterung besitzt, dann folgt die Aussage trivialerweise.

Wir nehmen an, dass S eine gültige Erweiterung

$$\text{EXT}_i(S) = F_s(F_{s-1}(\ldots F_{i+1}(S)\ldots))$$

besitzt. Dann führen $s - i$ Anwendungen von Bedingung 8.2 zu

$$F_s(F_{s-1}(\ldots F_{i+1}(S)\ldots)) \preceq_{\text{dom}} F_s(F_{s-1}(\ldots F_{i+1}(S')\ldots)).$$

Weiterhin führen $s-i$ Anwendungen von Bedingung 8.3 zu $H(F_j(\ldots F_{i+1}(S')\ldots)) \leq 0$ für alle $j = i+1, \ldots, s$, da $H(F_j(\ldots F_{i+1}(S)\ldots)) \leq 0$ gilt. Folglich besitzt S' eine gültige Erweiterung

$$\text{EXT}_i(S') = F_s(F_{s-1}(\ldots F_{i+1}(S')\ldots))$$

mit $\text{EXT}_i(S) \preceq_{\text{dom}} \text{EXT}_i(S')$. Abschließend ist $g(\text{EXT}_i(S'))$ mindestens so gut wie $g(\text{EXT}_i(S))$, d. h., $g(\text{EXT}_i(S')) \succeq g(\text{EXT}_i(S))$, aufgrund von Bedingung 8.4. Insgesamt gilt die Proposition, da wir gezeigt haben, dass der Zustand S den Zustand S' bezüglich Erweiterungen dominiert. □

Wenn wir die in Proposition 8.1 deklarativ beschriebene Berechnung operationalisieren, dann erhalten wir einen allgemeinen Rahmen für alle dynamischen Programme für alle multikriteriellen Optimierungsprobleme P. Der resultierende Rahmen ist in Algorithmus 6 zu finden.

Aufgrund von Proposition 8.1 nennen wir ein dynamisches Programm *korrekt* genau dann, wenn die Paretofront von g in $g(\mathcal{S}_s)$ enthalten ist und die Dominanzrelation $\preceq_{\text{dom}}$ Dominanz bezüglich Erweiterungen impliziert.

8.3. Evolutionäre Algorithmen

In diesem Abschnitt etablieren wir eine Verbindung zwischen evolutionären Algorithmen und dynamischen Programmen. Wir formulieren zunächst einen allgemeinen Rahmen für evolutionäre Algorithmen. Anschließend beschreiben wir, wie die einzelnen Komponenten ausgestaltet werden müssen, damit ein evolutionärer Algorithmus eine mit einem dynamischen Programm vergleichbare Laufzeit aufweist.

8.3.1. Rahmen für evolutionäre Algorithmen

Ein evolutionärer Algorithmus umfasst verschiedene generische Module. Diese Module müssen präzisiert werden, um ein gegebenes Problem angehen zu können. Neben der Ausgestaltung des evolutionären Algorithmus, d. h., der Wahl der Selektions- und Variationsoperatoren, beeinflusst natürlich auch die Modellierung des Problems, d. h., die Definition der Fitnessfunktion, letztlich den Erfolg der Optimierung. Dieses komplexe Zusammenspiel wird in zahlreichen experimentellen Studien sowie in einigen theoretischen Arbeiten (siehe z. B. [Neumann, 2008, Doerr et al., 2007b,c, Doerr und Johannsen, 2007]) beleuchtet.

Wir nehmen – wie eingangs besprochen – an, dass das zu lösende Problem durch eine zu optimierende multikriterielle Funktion $g\colon \mathcal{S} \to \mathbb{R}^k$ gegeben ist. Wir betrachten hier grundlegende evolutionäre Algorithmen, die aus den folgenden Komponenten bestehen.

Der evolutionäre Algorithmus, der in Algorithmus 7 angegeben ist, beginnt die Optimierung mit einer initialen Population $\mathcal{P}_0 \subseteq \mathcal{S}'$, wobei $\mathcal{S}'$ den sogenannten Genotypraum bezeichnet. Während der Optimierung verwendet der evolutionäre Algorithmus einen Selektionsoperator $\text{sel}(\cdot)$ und einen Mutationsoperator $\text{mut}(\cdot)$, um in jeder Generation genau ein neues Individuum zu erschaffen. Eine d-dimensionale Fitnessfunktion $f\colon \mathcal{S}' \to \mathbb{R}^d$ und eine Halbordnung $\succeq_{\text{par}}$ auf $\mathbb{R}^d$ induzieren eine Quasiordnung auf dem Genotypraum, die die Suche lenkt. Der Algorithmus akzeptiert ein neues Individuum, wenn sich in der aktuellen Population kein Individuum befindet, dessen Fitness die Fitness des neuen Individuums bezüglich $\succ_{\text{par}}$ dominiert. Danach wird jedes Individuum aus der aktuellen Population entfernt, dessen Fitness von der Fitness des neuen Individuums bezüglich $\succeq_{\text{par}}$ schwach dominiert wird. Der Algorithmus beendet die Optimierung, wenn ein Abbruchkriterium erfüllt ist, auf das wir hier nicht näher eingehen werden. Anschließend sorgt eine Ausgabefunktion

Algorithmus 7 Evolutionärer Algorithmus für g

```
1: P ← ∅
2: for I ∈ P_0 do
3:     if ∄I' ∈ P: f(I') ≻_par f(I) then
4:         P ← (P \ {I' ∈ P | f(I) ⪰_par f(I')}) ∪ {I}
5:     end if
6: end for
7: repeat
8:     I ← mut(sel(P))
9:     if ∄I' ∈ P: f(I') ≻_par f(I) then
10:        P ← (P \ {I' ∈ P | f(I) ⪰_par f(I')}) ∪ {I}
11:    end if
12: until Abbruchkriterium erfüllt
13: return {out(I) | I ∈ P ∧ ∄I' ∈ P: g(out(I')) ≻ g(out(I))}
```

out$(\cdot)$ dafür, die Individuen, die sich in der letzten Population befinden, auf Elemente aus dem Definitionsbereich $\mathcal{S}$ der zu optimierenden Funktion $g\colon \mathcal{S} \to \mathbb{R}^k$ abzubilden.

Die in dieser Dissertation vorgestellten evolutionären Algorithmen SEMO (siehe Abschnitt 2.4), DEMO (siehe Abschnitt 3.2) und FEMO (siehe Abschnitt 5.2) lassen sich beispielsweise in den Rahmen pressen. Wir machen das exemplarisch für den Algorithmus SEMO deutlich. Die Fitnessfunktion $f\colon \mathbb{B}^n \to \mathbb{R}^k$ stimmt hier mit der zu optimierenden Funktion $g\colon \mathbb{B}^n \to \mathbb{R}^k$ überein. Demzufolge gilt für die Ausgabefunktion $\text{out}(\cdot) = \text{id}(\cdot)$. Die Halbordnung $\succeq_{\text{par}}$ entspricht der (schwachen) Paretodominanz $\succeq$. Abschließend zieht der Selektionsoperator sel$(\cdot)$ rein zufällig ein Individuum aus der aktuellen Population und der Mutationsoperator mut$(\cdot)$ kippt die einzelnen Bits unabhängig voneinander mit Wahrscheinlichkeit $1/n$.

8.3.2. Definition der Module

Wir untersuchen nun, wie die einzelnen Module des im letzten Unterabschnitt beschriebenen evolutionären Algorithmus implementiert werden müssen, damit dieser auf ähnliche Weise wie ein dynamisches Programm von der Struktur eines Problems profitieren kann. Um dieses Ziel zu erreichen, setzen wir die Module des evolutionären Algorithmus und die Komponenten eines dynamischen Programms zueinander in Beziehung. Betrachte ein durch eine multikriterielle Funktion $g\colon \mathcal{S} \to \mathbb{R}^k$ gegebenes Problem, das mithilfe eines dynamischen Programms DP gelöst werden kann. Wenn es erforderlich ist, verwenden wir die Indizes DP und EA, um zwischen den Komponenten des dynamischen Programms und des evolutionären Algorithmus zu unterscheiden. Der evolutionäre Algorithmus arbeitet wie folgt.

Wir benutzen den Genotypraum $\mathcal{S}'_{\text{EA}} := \{0, \ldots, s\} \times \mathcal{S}'_{\text{DP}}$. Die zusätzliche Komponente des Genotypraumes wird benötigt, um eine Stufe des zugrunde liegenden dynamischen Programms widerzuspiegeln. Die initiale Population setzen wir auf $\mathcal{P}_0 = \{0\} \times \mathcal{S}_0$, wobei $\mathcal{S}_0$ den initialen Zustandsraum des dynamischen Programms DP

bezeichnet. Der Selektionsoperator $\mathrm{sel}(\cdot)$ wählt ein Individuum $I \in \mathcal{P}$ rein zufällig zur Mutation. Wenn wir den Mutationsoperator $\mathrm{mut}(\cdot)$ auf ein Individuum (i, S) anwenden, dann wählt dieser rein zufällig eine Zustandsüberführungsfunktion $F \in \mathcal{F}_{i+1}$ und generiert den Mutanten $\mathrm{mut}((i, S)) = (i{+}1, F(S))$. Wenn wir den Mutationsoperator auf ein Individuum (s, S) anwenden, dann passiert nichts, d. h., es ergibt sich der Mutant (s, S). Wir verwenden die Fitnessfunktion $f_{\mathrm{EA}} \colon \mathcal{S}'_{\mathrm{EA}} \to \{0, \ldots, n\} \times (\mathbb{R} \cup \{\pm\infty\})^d$ mit

$$f_{\mathrm{EA}}((i, S)) = \begin{cases} (i) \circ f_{\mathrm{DP}}(S) & \text{falls } H(S) \leq 0 \\ (i) \circ (\pm\infty)^d & \text{falls } H(S) > 0, \end{cases}$$

wobei $\circ$ die Konkatenation von zwei Vektoren bezeichnet. Die Bedeutung der Fitnessfunktion ist die Folgende. Wenn ein Individuum (i, S) gültig ist, d. h., $H(S) \leq 0$ gilt, dann wird es auf $(i, y_1, \ldots, y_d)$ abgebildet, wobei $(y_1, \ldots, y_d) = f_{\mathrm{DP}}(S)$ gilt. Anderenfalls wird es auf $(i, \pm\infty, \ldots, \pm\infty)$ abgebildet, wobei wir annehmen, dass $y \succ^{\mathrm{DP}}_{\mathrm{par}} (\pm\infty, \ldots, \pm\infty)$ für alle $y \in f_{\mathrm{DP}}(\mathcal{S})$ gilt. Wir benutzen die durch

$$(i', y') \succeq^{\mathrm{EA}}_{\mathrm{par}} (i, y) \iff i' = i \wedge y' \succeq^{\mathrm{DP}}_{\mathrm{par}} y$$

definierte Halbordnung $\succeq^{\mathrm{EA}}_{\mathrm{par}}$, um die Suche des evolutionären Algorithmus zu lenken. Abschließend definieren wir die Ausgabefunktion $\mathrm{out}_{\mathrm{EA}}((i, S)) := S$ für alle $(i, S) \in \{0, \ldots, s\} \times \mathcal{S}$. Aus formalen Gründen definieren wir zudem $\mathrm{out}_{\mathrm{EA}}((i, S)) := S'$ für alle $(i, S) \in \{0, \ldots, s\} \times (\mathcal{S}'_{\mathrm{DP}} \setminus \mathcal{S})$, wobei $S' \in \mathcal{S}$ einen beliebigen gültigen Suchpunkt bezeichnet. Die Ausgabefunktion wird am Ende eines Laufes des evolutionären Algorithmus gebraucht, um die zusätzlichen Informationen zu entfernen und ein Individuum in einen Suchpunkt des zu lösenden Problems zu transformieren.

8.3.3. Optimierungszeit des evolutionären Algorithmus

Wir zeigen in diesem Unterabschnitt, dass der evolutionäre Algorithmus, den wir im letzten Unterabschnitt entwickelt haben, ein durch g gegebenes Problem effizient löst, wenn das zugrunde liegende dynamische Programm dazu in der Lage ist. Das nächste Theorem setzt die erwartete Optimierungszeit des evolutionären Algorithmus und die Laufzeit des dynamischen Programms zueinander in Verbindung.

Theorem 8.6. *Betrachte ein – wie in Algorithmus 6 beschriebenes – dynamisches Programm und einen – wie in Algorithmus 7 und Unterabschnitt 8.3.2 beschriebenen – evolutionären Algorithmus. Dann werden von dem dynamischen Programm*

$$\sum_{i=1}^{s} |\mathcal{T}_{i-1}| \cdot |\mathcal{F}_i| \leq \kappa \cdot \sum_{i=0}^{s-1} s_i$$

Zustände generiert und die erwartete Optimierungszeit des evolutionären Algorithmus beträgt

$$\mathcal{O}\left(\kappa \cdot \sum_{i=0}^{s} s_i \cdot \sum_{i=1}^{s} \log(|\mathcal{T}_i|)\right),$$

wobei $\kappa := \max_{i\in\{1,\dots,s\}} |\mathcal{F}_i|$ *und* $s_i := \max\{|\mathcal{U}_i| \mid \mathcal{U}_i \subseteq \mathcal{S}_i$ *und* $\forall S, S' \in \mathcal{U}_i \colon S' \parallel_{\text{par}} S\}$ *gelten.*

Beweis. Wir teilen einen Lauf des evolutionären Algorithmus in s Phasen ein. Wir nennen Phase i, $1 \leq i \leq s$, *beendet*, wenn $f(\mathcal{P}) \succeq_{\text{par}} \{f((i,S)) \mid S \in \mathcal{S}_i\}$ gilt. Sei X_i, $1 \leq i \leq s$, die Zufallsvariable, die angibt, nach wie vielen Generationen Phase i beendet ist. Dann ist die Optimierungszeit durch die Zufallsvariable $X := \sum_{i=1}^{s} X_i$ gegeben.

Wenn sich der evolutionäre Algorithmus in Phase i befindet, dann gibt es ein $\mathcal{T}_{i-1} \in M(\mathcal{S}_{i-1}, \succeq_{\text{dom}})$, sodass alle $(i-1,S)$ mit $S \in \mathcal{T}_{i-1}$ in der Population $\mathcal{P}$ enthalten sind. Sei $0 \leq t \leq |\mathcal{T}_i|$ die Anzahl der noch nicht dominierten Fitnessvektoren aus $\{f((i,S)) \mid S \in \mathcal{S}_i\}$. Um ein Individuum zu erschaffen, das einen der genannten Fitnessvektoren dominiert, genügt es, wenn der Selektionsoperator ein bestimmtes Individuum auswählt und der Mutationsoperator eine bestimmte Zustandsüberführungsfunktion auswählt. Die Wahrscheinlichkeit für eine solches Ereignis ist demzufolge nicht kleiner als $((|\mathcal{T}_i| - t)/|\mathcal{P}|) \cdot (1/|\mathcal{F}_i|)$. Wenn wir die obere Schranke $|\text{DP}| := \sum_{i=0}^{s} s_i$ für die Populationsgröße $|\mathcal{P}|$ einsetzen, dann erhalten wir $((|\mathcal{T}_i| - t)/|\mathcal{P}|) \cdot (1/|\mathcal{F}_i|)$, wobei $|\text{DP}|$ die größte Anzahl unvergleichbarer Zustände aus dem dynamischen Programm DP bezeichnet. Die erwartete Wartezeit beträgt folglich höchstens $(|\text{DP}| \cdot |\mathcal{F}_i|)/(|\mathcal{T}_i| - t)$ Generationen, da sie geometrisch verteilt ist. Die erwartete Wartezeit, um Phase i zu beenden, ist daher durch

$$\text{E}(X_i) \leq \sum_{t=1}^{|\mathcal{T}_i|} \frac{|\text{DP}| \cdot |\mathcal{F}_i|}{t} = |\text{DP}| \cdot |\mathcal{F}_i| \cdot H_{|\mathcal{T}_i|}$$

beschränkt, wobei H_k die k-te harmonische Zahl bezeichnet (siehe Lemma A.17).

Insgesamt erhalten wir die Schranke

$$\begin{aligned}
\text{E}(X) &\leq \sum_{i=1}^{s} |\text{DP}| \cdot |\mathcal{F}_i| \cdot H_{|\mathcal{T}_i|} \\
&\leq \kappa \cdot |\text{DP}| \cdot \sum_{i=1}^{s} H_{|\mathcal{T}_i|} \\
&= \mathcal{O}\left(\kappa \cdot |\text{DP}| \cdot \sum_{i=1}^{s} \log(|\mathcal{T}_i|)\right).
\end{aligned}$$

für die erwartete Optimierungszeit. □

Eine Inspektion des letzten Beweises offenbart, dass wir die Schranke auf die gleiche Weise wie für Theorem 7.5 aus Abschnitt 7.4 verbessern können.

Theorem 8.7. *Betrachte ein – wie in Algorithmus 6 beschriebenes – dynamisches Programm und einen – wie in Algorithmus 7 und Unterabschnitt 8.3.2 beschriebenen*

– evolutionären Algorithmus. Wenn $|\mathcal{T}_s| = 2^{\mathcal{O}(s)}$ *gilt, dann beträgt mit Wahrscheinlichkeit* $1 - 2^{-\Omega(s)}$ *die Optimierungszeit des evolutionären Algorithmus*

$$\mathcal{O}\left(s \cdot \kappa \cdot \sum_{i=0}^{s} s_i\right),$$

wobei $\kappa := \max_{i\in\{1,\dots,s\}} |\mathcal{F}_i|$ *und* $s_i := \max\{|\mathcal{U}_i| \mid \mathcal{U}_i \subseteq \mathcal{S}_i$ *und* $\forall S, S' \in \mathcal{U}_i \colon S' \parallel_{\text{par}} S\}$ *gelten.*

Beweis. Betrachte $\mathcal{S}_s$. Sei $M(\mathcal{S}_s, \succeq_{\text{dom}})$ die Menge, die alle dominierenden Teilmengen von $\mathcal{S}_s$ enthält, deren Größe minimal ist. Halte ein $\mathcal{T}_s \in M(\mathcal{S}_s, \succeq_{\text{dom}})$ und ein $S \in \mathcal{T}_s$ fest. Dann gibt es ein $S' \in \mathcal{S}_0$ und $F_i \in \mathcal{F}_i$, $1 \leq i \leq s$, mit $S = F_s(F_{s-1}(\dots F_1(S')\dots))$ und $H(F_i(F_{i-1}(\dots F_1(S)\dots))) \leq 0$ für alle $1 \leq i \leq s$. Wenn die aktuelle Population ein Individuum (i, S'') mit $S'' \succeq_{\text{dom}} F_i(F_{i-1}(\dots F_1(S')\dots))$ enthält, beträgt die Wahrscheinlichkeit, ein Individuum $(i+1, S''')$ mit $S''' \succeq_{\text{dom}} F_{i+1}(F_1(\dots F_1(S')\dots))$ zu erschaffen, mindestens

$$\frac{1}{\sum_{i=0}^{s} s_i} \cdot \frac{1}{|\mathcal{F}_{i+1}|} \geq \frac{1}{\kappa \cdot \sum_{i=0}^{s} s_i} =: p.$$

Wir modellieren die Situation wie folgt. Seien X_i, $1 \leq i \leq t$, unabhängige Zufallsvariable mit $\mathrm{W}(X_i = 1) = p$ und $\mathrm{W}(X_i = 0) = 1 - p$. Sei $X := \sum_{i=1}^{t} X_i$. Dann beträgt die Wahrscheinlichkeit, dass die Population nach t Generationen noch kein Individuum (s, S'''') mit $S'''' \succeq_{\text{dom}} S$ enthält, höchstens

$$\mathrm{W}(X \leq s-1) = \mathrm{W}\left(X \leq \frac{s-1}{\mathrm{E}(X)} \cdot \mathrm{E}(X)\right) = \mathrm{W}\left(X \leq \frac{s-1}{t \cdot p} \cdot \mathrm{E}(X)\right).$$

Da $|\mathcal{T}_s| = 2^{\mathcal{O}(s)}$ gilt, gibt es eine Konstante $c > 1/4$ mit $|\mathcal{T}_s| \leq \exp(c \cdot s)$. Wenn wir $t := d \cdot (s-1) \cdot \kappa \cdot \sum_{i=0}^{s} s_i$ Generationen betrachten, wobei $d > 1$ eine hinreichend große Konstante ist, dann folgt mithilfe einer Chernoff-Ungleichung (siehe Lemma A.6)

$$\begin{aligned}
&\mathrm{W}\left(X \leq \frac{s-1}{t \cdot p} \cdot \mathrm{E}(X)\right) \\
&= \mathrm{W}\left(X \leq \frac{s-1}{(d \cdot (s-1) \cdot \kappa \cdot \sum_{i=0}^{s} s_i) \cdot (1/(\kappa \cdot \sum_{i=0}^{s} s_i))} \cdot \mathrm{E}(X)\right) \\
&= \mathrm{W}\left(X \leq \left(1 - \frac{d-1}{d}\right) \cdot \mathrm{E}(X)\right) \\
&\leq \exp\left(-\frac{((d-1)/d)^2 \cdot \mathrm{E}(X)}{2}\right) \\
&= \exp\left(-\frac{((d-1)/d)^2 \cdot (d \cdot (s-1))}{2}\right) \\
&= \exp\left(-\frac{(d-1)^2}{2 \cdot d} \cdot (s-1)\right).
\end{aligned}$$

Wir wählen $d := 8 \cdot c$. Dann gilt

$$\frac{(d-1)^2}{2 \cdot d} > \frac{d^2}{8 \cdot d} = c.$$

Wir wenden nun die boolesche Ungleichung (siehe Lemma A.4) an und schließen, dass das Optimierungsziel nach t Generationen höchstens mit Wahrscheinlichkeit

$$|\mathcal{T}_s| \cdot \exp\left(-\frac{(d-1)^2}{2 \cdot d} \cdot (s-1)\right) \leq \exp(c \cdot s) \cdot \exp\left(-\frac{(d-1)^2}{2 \cdot d} \cdot (s-1)\right) = 2^{-\Omega(s)}$$

nicht erreicht wird. Insgesamt folgt die Behauptung. □

8.4. Beispiele

In diesem Abschnitt demonstrieren wir, wie die in den vorangegangenen Abschnitten vorgestellten Rahmen für dynamische Programme und evolutionäre Algorithmen genutzt werden können, um den Erfolg von bekannten evolutionären Algorithmen zu erklären. Konkret konstruieren wir für bekannte kombinatorische Optimierungsprobleme evolutionäre Algorithmen, die auf dynamischen Programmen basieren. Die Ergebnisse aus dem letzten Abschnitt stellen sicher, dass die evolutionären Algorithmen ähnlich gute Eigenschaften wie die zugrunde liegenden dynamischen Programme aufweisen. Wir merken an, dass es natürlich nicht sinnvoll ist, einen solchen evolutionären Algorithmus zu nutzen, wenn man das dynamische Programm kennt. Andererseits sind die entstehenden evolutionären Algorithmen sehr natürlich, sodass man darauf hoffen kann, auf sie zu stoßen, ohne die dynamischen Programme zu kennen.

Wir begnügen uns hier mit der Beschreibung der Bestandteile dynamischer Programme für ganz unterschiedliche Probleme. Die Existenz und die Arbeitsweise der evolutionären Algorithmen für die vorgestellten Probleme folgt dann aus Abschnitt 8.3.

8.4.1. Kürzeste-Wege-Problem

Ein klassisches Problem, das in den in Abschnitt 8.2 präsentierten Rahmen für dynamische Programme passt, ist das Kürzeste-Wege-Problem für alle Paare. Eine Eingabe besteht aus einem zusammenhängenden ungerichteten Graphen $G = (V, E)$ und positiven Kantengewichten $w\colon E \to \mathbb{R}^+$. Die Aufgabe besteht darin, für jedes Paar (u, v) von Knoten einen kürzesten Pfad zu finden, der u und v miteinander verbindet.

Doerr et al. [2008a] haben gezeigt, dass das Problem von einem evolutionären Algorithmus im Erwartungswert in Polynomialzeit gelöst werden kann. Im Folgenden diskutieren wir einige grundlegende Eigenschaften des betrachteten Problems und zeigen, dass sich der evolutionäre Algorithmus auf die hier vorgestellte Weise aus einem dynamischen Programm herleiten lässt.

Die meisten Algorithmen für das Kürzeste-Wege-Problem fußen auf der grundlegenden Erkenntnis, dass Teilpfade kürzester Wege auch kürzeste Wege beschreiben.

Folglich kann ein kürzester Weg von u nach v aus einem kürzesten Weg von u nach x, wobei x ein geeigneter Nachbar von v ist, konstruiert werden, indem die Kante (x, v) an den Pfad angefügt wird. Diese Beobachtung erlaubt die Formulierung eines sehr natürlichen dynamischen Programms, da die Teillösungen, um die der Suchraum oft künstlich ergänzt wird, hier einen integralen Bestandteil bilden.

Für das Kürzeste-Wege-Problem ist der Suchraum $\mathcal{S}$ natürlicherweise als Menge aller Pfade in G gegeben. Wir fassen Pfade als Folgen von Knoten auf. Da alle Kantengewichte positiv sind, können wir uns offensichtlich auf kreisfreie Pfade beschränken. Demzufolge können wir den Suchraum gegebenenfalls auf kreisfreie Pfade mit höchstens n Knoten bzw. $n-1$ Kanten begrenzen. Als erweiterten Suchraum $\mathcal{S}'$ wählen wir die Menge, die alle Folgen von Knoten enthält. Die zu minimierende Zielfunktion $g = (g_{i,j})\colon \mathcal{S} \to \mathbb{R}^{n^2}$ ist durch

$$g_{i,j}((v_0, \ldots, v_\ell)) := \begin{cases} \sum_{i=1}^{\ell} w((v_{i-1}, v_i)) & \text{falls } v_0 = i \text{ und } v_\ell = j \\ \infty & \text{falls } v_0 \neq i \text{ oder } v_\ell \neq j \end{cases}$$

gegeben. Die initiale Zustandsmenge $\mathcal{S}_0$ ist durch die Menge gegeben, die alle Pfade der Länge 0 enthält, d. h., $\mathcal{S}_0 := \{(v) \mid v \in V\}$. Das Programm durchläuft $n-1$ Stufen. Wir definieren nun für alle $1 \leq i < n$ die Zustandsüberführungsfunktionen $\mathcal{F}_i := \{\mathrm{id}\} \cup \{F_v \mid v \in V\}$, wobei $F_v\colon \mathcal{S} \to \mathcal{S}'$ die Abbildung bezeichnet, die den Knoten v an eine Folge von Knoten anhängt. Um ungültige Zustände herauszufiltern, definieren wir die Gültigkeitsfunktion $H(S)$ als -1, wenn S einen Pfad in G beschreibt, und als 1, wenn das nicht der Fall ist.

Es verbleibt, zu definieren, wann ein Zustand einen anderen Zustand dominiert. Wir definieren $\succeq_{\mathrm{dom}}$ auf natürliche Weise dergestalt, dass $S' \succeq_{\mathrm{dom}} S$ genau dann gilt, wenn die Pfade S und S' die gleichen Knoten verbinden und S' nicht länger als S ist. Diese Anforderung kann in unserem Rahmen durch die Definition von $f = (f_{i,j})\colon \mathcal{S} \to \mathbb{R}^{n^2}$ gemäß

$$f_{i,j}((v_0, \ldots, v_\ell)) := g_{i,j}((v_0, \ldots, v_\ell))$$

und $\succeq_{\mathrm{par}}$ gemäß

$$(x'_1, \ldots, x'_{n^2}) \succeq_{\mathrm{par}} (x'_1, \ldots, x'_{n^2}) \iff \forall i \in \{1, \ldots, n^2\}\colon x'_i \leq x_i$$

erfüllt werden. Da das Gewicht eines Pfades, der aus dem Anfügen einer Kante an einen bestehenden Pfad hervorgeht, monoton von dem Gewicht des existierenden Pfades abhängt, wird das Bellmanprinzip durch die Relation $\succeq_{\mathrm{dom}}$ richtig umgesetzt. Folglich enthalten die Zustandsmengen höchstens einen Pfad pro Paar von Knoten.

Das resultierende dynamische Programm arbeitet wie folgt. Es beginnt mit allen Pfaden der Länge 0 als Zustandsmenge. Anschließend wird $(n-1)$-mal die folgende Schleife durchlaufen. Auf jeder Stufe wird jeder Knoten an jeden Pfad in der aktuellen Zustandsmenge angefügt. Wenn ein auf diese Weise generierter Pfad kürzer als der aktuell in der Zustandsmenge befindliche Pfad mit den gleichen Endknoten ist, dann wird der letztgenannte Pfad durch den erstgenannten ersetzt. Abgesehen von

Implementierungsdetails (Speicherung eines Pfades statt Speicherung des Gewichts eines Pfades) erhalten wir eine Variante des bekannten Floyd-Warshall-Algorithmus.

Wir können das dynamische Programm auf die in Abschnitt 8.3 beschriebene Weise in einen evolutionären Algorithmus transformieren. Weil $|\mathcal{T}_{n-1}| = n^2 = 2^{\mathcal{O}(n)}$ gilt, können wir mithilfe von Theorem 8.7 schließen, dass die Optimierungszeit des evolutionären Algorithmus mit überwältigender Wahrscheinlichkeit

$$\mathcal{O}\left(s \cdot \kappa \cdot \sum_{i=0}^{s} s_i\right) = \mathcal{O}(n^5)$$

beträgt, wobei $s_i := \max\{|\mathcal{U}_i| \mid \mathcal{U}_i \subseteq \mathcal{S}_i \text{ und } \forall S, S' \in \mathcal{U}_i \colon S' \parallel_{\text{par}} S\}$ gilt. Die Optimierungszeit des in [Doerr et al., 2008a] betrachteten evolutionären Algorithmus beträgt $\Theta(n^4)$. Der Unterschied resultiert daraus, dass die Größe der Population von $\Theta(n^3)$ auf $\Theta(n^2)$ gesenkt werden kann, wenn auf die Stufeneinteilung verzichtet wird. In diesem Fall kann darauf verzichtet werden, da es zulässig ist, wenn im Laufe der Optimierung ein aus ℓ Kanten bestehender Pfad von u nach v durch einen nicht längeren aus $\ell' > \ell$ Kanten bestehenden Pfad von u nach v ersetzt wird.

8.4.2. Problem des Handlungsreisenden

Wir widmen uns hier dem sehr bekannten NP-schwierigen Problem des Handlungsreisenden. Eine Eingabe für das Problem besteht aus einem vollständigen Graphen $G = (V, E)$ mit nicht negativen Kantengewichten $w \colon E \to \mathbb{R}_0^+$, wobei wir ohne Beschränkung der Allgemeinheit $V = \{1, \ldots, n\}$ unterstellen. Wir sind an einer Permutation $(v_1, \ldots, v_n)$ der Knoten interessiert, sodass die Länge $\sum_{i=2}^{n} w(v_i, v_{i-1}) + w(v_n, v_1)$ der beschriebenen Rundreise minimiert wird. Ohne Beschränkung können wir weiterhin annehmen, dass $v_1 = 1$ gilt, d. h., dass der Handlungsreisende seine Rundreise in dem festen Knoten 1 antritt.

In [Theile, 2009] wurde ein evolutionärer Algorithmus für das Problem vorgestellt und theoretisch untersucht. Im Folgenden zeigen wir, wie sich der Algorithmus mithilfe des hier vorgestellten Ansatzes auf naheliegende Weise aus einem dynamischen Programm herleiten lässt. Dem bekannten dynamische Programm von Held und Karp [1962] für das Problem des Handlungsreisenden liegen die folgenden Ideen zugrunde. Das Programm berechnet nacheinander für jede Teilmenge $A \subseteq \{2, \ldots, n\}$ und für jeden Knoten $t \in A$ die Länge des kürzesten Weges, der an dem Knoten 1 beginnt, alle Knoten aus A durchläuft und an dem Knoten t endet. Wenn $W(A, t)$ die Länge dieses Pfades bezeichnet, dann gilt $W(A, t) = \min_{x \in A \setminus \{t\}} W(A \setminus \{t\}, x) + w((x, t))$. Da es $\mathcal{O}(n \cdot 2^n)$ Möglichkeiten gibt, A und t zu wählen, und die Berechnung von $W(A, t)$ jeweils die Zeit $\mathcal{O}(n)$ benötigt, beträgt die Gesamtlaufzeit $\mathcal{O}(n^2 \cdot 2^n)$.

Wir formulieren das Problem und den skizzierten Algorithmus nun in unserem Rahmen. Der Suchraum $\mathcal{S}$ umfasst alle einfachen Pfade in G, die mit dem Knoten 1 beginnen, und der erweiterte Suchraum $\mathcal{S}'$ umfasst alle Folgen von höchstens n Knoten, die mit dem Knoten 1 beginnen. Die zu minimierende Zielfunktion $g \colon \mathcal{S} \to \mathbb{R}$

ist durch

$$g((v_1 = 1, \ldots, v_\ell)) := \begin{cases} \sum_{i=2}^{\ell} w(v_i, v_{i-1}) + w(v_n, v_1) & \text{falls } \{v_1, \ldots, v_\ell\} = \{1, \ldots, n\} \\ \infty & \text{falls } \{v_1, \ldots, v_\ell\} \neq \{1, \ldots, n\} \end{cases}$$

gegeben. Die initiale Zustandsmenge $\mathcal{S}_0$ umfasst $n-1$ Zustände. Sie enthält für alle $i \in \{2, \ldots, n\}$ den Pfad $(1, i)$. Das eigentliche Programm durchläuft $n-2$ Stufen. Die $n-2$ Mengen $\mathcal{F}_i$ entsprechen $\{\mathrm{id}\} \cup \{F_v \mid v \in V \setminus \{1\}\}$, wobei F_v die durch $F_v((v_0, \ldots, v_\ell)) := (v_0, \ldots, v_\ell, v)$ definierte Zustandsüberführungsfunktion bezeichnet. Ungültige Zustände zeichnen sich dadurch aus, dass sie keine einfachen Pfade repräsentieren. Folglich definieren wir

$$H(S) := \begin{cases} -1 & \text{falls } S \in \mathcal{S} \\ 1 & \text{falls } S \notin \mathcal{S}. \end{cases}$$

Wir definieren nun die Dominanzrelation $\succeq_{\mathrm{dom}}$, um aussichtslose Teillösungen so früh wie möglich zu verwerfen. Wir legen fest, dass $p' = (v'_1, \ldots, v'_{\ell'}) \succeq_{\mathrm{dom}} p = (v_1, \ldots, v_\ell)$ genau dann gilt, wenn $\{v'_1, \ldots, v'_{\ell'}\} = \{v_1, \ldots, v_\ell\}$, $v'_{\ell'} = v_\ell$ und $w(p') \leq w(p)$ gelten. Das kann in unserem Rahmen mithilfe der durch

$$f_{A,t}((v_0, \ldots, v_\ell)) := \begin{cases} \sum_{i=1}^{\ell} w(v_i, v_{i-1}) & \text{falls } \{v_0, \ldots, v_\ell\} = A \text{ und } v_\ell = t \\ \infty & \text{falls } \{v_0, \ldots, v_\ell\} \neq A \text{ und } v_\ell \neq t \end{cases}$$

definierten Funktion $f = (f_{A,t})\colon \mathcal{S} \to \mathbb{R}^{n \cdot 2^n}$ und der durch

$$(x'_1, \ldots, x'_{n \cdot 2^n}) \succeq_{\mathrm{par}} (x'_1, \ldots, x'_{n \cdot 2^n}) \iff \forall i \in \{1, \ldots, n \cdot 2^n\}\colon x'_i \leq x_i$$

definierten Halbordnung $\succeq_{\mathrm{par}}$ realisiert werden.

Weil $|\mathcal{T}_{n-2}| = n \cdot 2^n = 2^{\mathcal{O}(n)}$ gilt, können wir mithilfe von Theorem 8.7 schließen, dass die Optimierungszeit des zugehörigen evolutionären Algorithmus mit überwältigender Wahrscheinlichkeit

$$\mathcal{O}\left(s \cdot \kappa \cdot \sum_{i=0}^{s} s_i\right) = \mathcal{O}(n^3 \cdot 2^n)$$

beträgt, wobei $s_i := \max\{|\mathcal{U}_i| \mid \mathcal{U}_i \subseteq \mathcal{S}_i \text{ und } \forall S, S' \in \mathcal{U}_i\colon S' \parallel_{\mathrm{par}} S\}$ gilt. Die genannte Schranke entspricht der in [Theile, 2009] gezeigten Schranke für die Optimierungszeit.

8.4.3. Rucksack-Problem

Wir haben bereits gesehen, dass die bekannten evolutionären Algorithmen für das APSP („all-pairs shortest path problem“) [Doerr et al., 2008a] und das TSP („travelling salesperson problem“) [Theile, 2009] in den vorgeschlagenen Rahmen passen.

Das Rucksack-Problem ist ein weiteres bekanntes kombinatorisches Optimierungsproblem, das mithilfe dynamischer Programmierung gelöst werden kann. Wir betrachten dieses Problem als drittes Beispiel, um zu verdeutlichen, wie evolutionäre Algorithmen konstruiert werden können, die dazu in der Lage sind, das Verhalten dynamischer Programme nachzuahmen.

Eine Eingabe für das Rucksack-Problem besteht aus genau $n \in \mathbb{N}^+$ Gegenständen $\{1, \ldots, n\}$, wobei jeder Gegenstand i durch ein Gewicht $w_i \in \mathbb{R}_0^+$ und einen Nutzen $p_i \in \mathbb{R}_0^+$ charakterisiert ist. Zusätzlich ist eine Gewichtsschranke $W \in \mathbb{R}_0^+$ gegeben. Das Ziel besteht darin, eine Auswahl $K \subseteq \{1, \ldots, n\}$ von Gegenständen zu bestimmen, die den Gesamtnutzen $\sum_{i \in K} p_i$ unter Einhaltung der Nebenbedingung $\sum_{i \in K} w_i \leq W$ maximiert. Wir pressen das Problem in unseren Rahmen, indem wir den Suchraum

$$\mathcal{S} := \left\{ x \in \mathbb{B}^n \mid \sum_{i=1}^{n} x_i \cdot w_i \leq W \right\},$$

den erweiterten Suchraum $\mathcal{S}' := \mathbb{B}^n$ und die durch

$$g(x) := \sum_{i=1}^{n} x_i \cdot p_i$$

definierte zu maximierende pseudoboolesche Zielfunktion $g \colon \mathcal{S} \to \mathbb{R}$ betrachten.

Ein klassisches dynamisches Programm für das Rucksack-Problem arbeitet wie folgt. Das Programm beginnt mit dem Zustandsraum $\mathcal{S}_0 = \{0^n\}$, der einen die leere Auswahl von Gegenständen repräsentierenden Zustand 0^n enthält. Das dynamische Programm durchläuft n Stufen. Auf der i-ten Stufe wird versucht, den i-ten Gegenstand zu allen bisher generierten Auswahlen von Gegenständen hinzuzufügen. Formal definieren wir $\mathcal{F}_i = \{F_0, F_i\}$ mit $F_0(x) := x$ und $F_i(x) := (x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$. Die neuen Auswahlen x von Gegenständen werden akzeptiert, wenn sie nicht die Gewichtsschranke überschreiten, d. h., $H(x) \leq 0$ gilt, wobei $H(x) := \sum_{j=1}^{n} x_j \cdot w_j - W$ ist. Um aussichtslose Teillösungen so früh wie möglich zu verwerfen, benutzen wir die durch

$$f(x) = \left(\sum_{i=1}^{n} x_i \cdot p_i, \sum_{i=1}^{n} x_i \cdot w_i \right)$$

definierte Funktion $f \colon \mathcal{S} \to \mathbb{R}^2$ und die durch

$$(p_1', w') \succeq_{\text{par}} (p_1, w) \iff p_i' \geq p_i \wedge w' \leq w$$

definierte Halbordnung $\succeq_{\text{par}}$, um $\succeq_{\text{dom}}$ festzulegen.

Die Laufzeit des beschriebenen dynamischen Programms beträgt $\mathcal{O}(n \cdot W)$, da für alle Zustandsmengen $\sum_{i=1}^{n} |\mathcal{T}_{i-1}| \leq n \cdot (W + 1)$ gilt, wobei die Ungleichung daraus folgt, dass wir auf der i-ten Stufe für jedes Gewicht zwischen 0 und W höchstens eine Auswahl von nicht mehr als i Gegenständen speichern. Weil $|\mathcal{T}_n| = n \cdot W = 2^{\mathcal{O}(n)}$

gilt, können wir mithilfe von Theorem 8.7 schließen, dass die Optimierungszeit des zugehörigen evolutionären Algorithmus mit überwältigender Wahrscheinlichkeit

$$\mathcal{O}\left(s \cdot \kappa \cdot \sum_{i=0}^{s} s_i\right) = \mathcal{O}(n^2 \cdot W)$$

beträgt, wobei $s_i := \max\{|\mathcal{U}_i| \mid \mathcal{U}_i \subseteq \mathcal{S}_i \text{ und } \forall S, S' \in \mathcal{U}_i \colon S' \parallel_{\text{par}} S\}$ gilt.

8.5. Fazit

Wir haben in diesem Kapitel untersucht, wie ein evolutionärer Algorithmus beschaffen sein muss, um die Fähigkeit zu erlangen, die Arbeitsweise eines dynamischen Programms zu simulieren. Wir haben zunächst allgemeine Rahmen für dynamische Programme und evolutionäre Algorithmen vorgestellt. Anschließend haben wir gezeigt, wie ein evolutionärer Algorithmus anhand eines gegebenen dynamischen Programms konstruiert werden kann und die Optimierungszeit des resultierenden evolutionären Algorithmus analysiert. Durch die Betrachtung prominenter kombinatorischer Optimierungsprobleme haben wir gezeigt, dass viele aus der Literatur bekannte evolutionäre Algorithmen mithilfe der vorgestellten allgemeinen Konstruktionsmethode aus den zugehörigen dynamischen Programmen abgeleitet werden können. Natürlich zieht ein auf diese Weise konstruierter evolutionärer Algorithmus keine praktischen Konsequenzen nach sich. Dennoch wird durch die Rückführung bestehender evolutionärer Algorithmen auf dynamische Programme die beweisbare Effizienz dieser Algorithmen auf neue Weise erklärt. Weiterhin konnte auch gezeigt werden, wie effiziente evolutionäre Algorithmen für neue kombinatorische Optimierungsprobleme – wie z. B. das Rucksack-Problem – konstruiert werden können.

Teil III.

Andere moderne randomisierte Suchheuristiken

9. Ameisensysteme für Kürzeste-Wege-Probleme

9.1. Einleitung

Ameisenkolonie-Optimierung (auf Englisch „Ant Colony Optimization“, ACO) ist ein modernes Optimierungsparadigma mit vielen erfolgreichen Anwendungen auf kombinatorische Optimierungsprobleme aus ganz verschiedenen Bereichen (siehe z. B. [Dorigo und Stützle, 2004]). Die meisten Ameisensysteme sind vom Verhalten realer Ameisen, die nach Futter suchen, inspiriert, das es einer Ameisenkolonie ermöglicht, kürzeste Wege zwischen ihrem Nest und einer Futterquelle zu finden. Ameisen kommunizieren, indem sie, während sie die Umgebung nach Futter absuchen, sogenannte Pheromone auf dem Boden platzieren. Diese Chemikalien können von anderen Ameisen wahrgenommen werden. Sie leiten die Suche der anderen Ameisen auf die folgende Weise. Wenn eine Ameise eine Futterquelle findet, dann neigt sie dazu, sich ihrer eigenen Pheromonspur folgend zum Nest zurückzubewegen und dabei die Intensität der Spur mit weiteren Pheromonen zu verstärken. Andere Ameisen werden dann von diesem Pfad angezogen und hinterlassen darauf ihrerseits Pheromone. Wenn der Pfad kurz ist, dann nimmt die Pheromondichte auf dem Pfad schnell zu. Durch dieses Verhalten wird, auch wenn mehrere Ameisen verschiedene Pfade von dem Nest zu einer Nahrungsquelle gefunden haben, sichergestellt, dass es wahrscheinlich ist, dass fast alle Ameisen relativ schnell einem kürzesten Pfad folgen. Diese kollektive Intelligenz hat zu dem Paradigma „Ameisenkolonie-Optimierung“ geführt.

Die ausgefeilten Kommunikationsmechanismen von realen Ameisen sind auf viele unterschiedliche Optimierungsprobleme wie z. B. das Problem des Handlungsreisenden [Dorigo und Gambardella, 1997], Routenplanungsprobleme [Dorigo et al., 1991, Di Caro und Dorigo, 1998] und viele andere kombinatorische Probleme übertragen worden; das Buch von Dorigo und Stützle [2004] gibt hier einen guten Überblick. Diese Algorithmen verhalten sich auf praktischen Problemen häufig sehr gut und sie liefern oft bessere Ergebnisse als deterministische Algorithmen oder andere randomisierte Suchheuristiken. Außerdem sind sie auch dann anwendbar, wenn das zu optimierende Problem nicht hinreichend gut verstanden ist, um einen maßgeschneiderten Algorithmus zu entwerfen. In einem extremen Fall ist das Problem als Black-Box gegeben, mit dessen Hilfe lediglich Lösungskandidaten ausgewertet werden können, und die Auswertung von Lösungskandidaten ist der einzige Weg, um Informationen über das Problem zu gewinnen. Dieses Szenario wird „Black-Box-Optimierung“ (auf Englisch „Black-Box Optimization“) genannt (siehe z. B. [Droste et al., 2006] wie auch Abschnitt 2.3).

Obschon es eine Vielzahl von Anwendungen gibt, ist das theoretische Wissen über Ameisenalgorithmen immer noch außerordentlich limitiert. Es herrscht breiter Konsens darüber, dass eine solide theoretische Fundierung vonnöten ist [Wegener, 2003, Dorigo und Blum, 2005, Gutjahr, 2007]. Erste theoretische Untersuchungen haben sich Konvergenzbeweisen [Gutjahr, 2003] und vereinfachten Modellen von Ameisenalgorithmen [Merkle und Middendorf, 2002] zugewandt. In 2006 wurden die ersten rigorosen Untersuchungen der Laufzeit von Ameisenalgorithmen für die Optimierung von einfachen pseudobooleschen Funktionen unabhängig von Gutjahr [2008] und Neumann und Witt [2006] vorgestellt. Die letztgenannten Autoren stellten einen 1-ANT genannten Algorithmus vor. Dieser Algorithmus speichert die beste bisher gefundene Lösung. In jeder Iteration wird eine neue Lösung konstruiert. Wenn die Qualität der neuen Lösung mindestens so gut wie die der besten bisher gefundenen Lösung ist, wird die beste bisher gefundene Lösung durch die neue Lösung ersetzt und die Pheromone werden aktualisiert. Mit anderen Worten heißt das, dass jede neue beste bisher gefundene Lösung nur ein einziges Mal Einfluss auf die Pheromone nimmt. Der Einfluss einer Lösung auf die Pheromone wird gewöhnlich mithilfe eines Parameters $0 < \rho \leq 1$ eingestellt, der *Evaporationsfaktor* genannt wird. Je kleiner der Wert von ρ ist, desto größer ist der Einfluss der bisherigen Pheromone bzw. desto kleiner ist der Einfluss der Lösung, die herangezogen wird, um neue Pheromone abzulegen. Untersuchungen des Algorithmus 1-ANT [Neumann und Witt, 2006, Doerr et al., 2007d, Neumann und Witt, 2009] haben gezeigt, dass er sogar auf sehr einfachen Problemen stagniert und die erwartete Zeit, bis ein Optimum gefunden wird, exponentiell ist, wenn der Evaporationsfaktor zu klein gewählt wird. Andere Ameisenalgorithmen, Varianten des bekannten Max-Min-Ameisensystems (MMAS) [Stützle und Hoos, 2000], verstärken die beste bisher gefundene Lösung in jeder Iteration. Dadurch wird das Problem der Stagnation vermieden und es werden effiziente Laufzeiten auf verschiedenen Problemen erreicht [Gutjahr und Sebastiani, 2008, Neumann et al., 2009].

Neumann et al. [2008] untersuchten die Auswirkungen der Kreuzung von Ameisensystemen mit lokaler Suche. Sie demonstrierten für künstlich geschaffene Probleme, dass die Verwendung von lokaler Suche mit hoher Wahrscheinlichkeit exponentielle Laufzeiten in polynomielle Laufzeiten verwandeln kann. Für andere konstruierte Funktionen ist der Effekt umgekehrt. Im Hinblick auf Ameisensysteme für kombinatorische Optimierungsprobleme präsentierten Neumann und Witt [2008] eine Analyse für minimale aufspannende Bäume. Sie bewiesen polynomielle obere Schranken für verschiedene Mechanismen, die es den Ameisen ermöglichen, aufspannende Bäume zu konstruieren. Sie thematisierten auch die Auswirkungen der Verwendung von heuristischen Informationen, um den Konstruktionsprozess zu unterstützen. Kürzlich präsentierte Zhou [2009] eine Analyse eines Ameisensystems, das polynomielle Laufzeiten auf einfachen Instanzen für das Problem des Handlungsreisenden zeigt. Außerdem stellten Attiratanasunthron und Fakcharoenphol [2008] eine Analyse der Laufzeit eines Ameisenalgorithmus für ein Kürzeste-Wege-Problem vor, genauer gesagt für das Kürzeste-Wege-Problem mit einer Senke auf gerichteten azyklischen Graphen. Ihr Algorithmus n-ANT lehnt sich sowohl an den Algorithmus 1-ANT [Neumann und Witt,

2006] als auch an den Algorithmus AntNet [Di Caro und Dorigo, 1998] an. Soweit wir wissen, stellt diese Untersuchung die erste und einzige rigorose Laufzeitanalyse eines Ameisensystems für ein Kürzeste-Wege-Problem dar. Dies überrascht, da Kürzeste-Wege-Probleme die Entwicklung von Ameisensystemen wesentlich inspiriert haben.

Das Ziel dieses Kapitels besteht darin, tiefere Einsichten in die Arbeitsweisen von Ameisenalgorithmen zu gewinnen und ihre Leistungsfähigkeit für Kürzeste-Wege-Probleme einzuordnen. Wir wählen Kürzeste-Wege-Probleme nicht nur aufgrund ihrer offensichtlichen Bedeutung für natürliche Ameisenkolonien. Diese Probleme sind sehr gut erforscht und weisen eine klare Struktur auf, was sie als ausgezeichneten Startpunkt für theoretische Untersuchungen qualifiziert. Außerdem sind Kürzeste-Wege-Probleme ein fundamentales Forschungsgebiet in der Informatik und die Entwicklung von Algorithmen für diese Probleme hält nach wie vor Herausforderungen bereit [Chan, 2007, Bast et al., 2007, Vassilevska, 2008].

9.1.1. Frühere Arbeiten

Kürzeste-Wege-Probleme sind bereits im Kontext von anderen Metaheuristiken untersucht worden. Wir präsentieren hier einen kurzen Abriss dieser Ergebnisse. Beachte, dass es in der Theorie über randomisierte Suchheuristiken gängige Praxis ist, die Anzahl der Auswertungen der Zielfunktion zu analysieren. Dies entspricht hier offensichtlich der Anzahl der Pfadlängenauswertungen. (Wir gehen in Abschnitt 9.2 darauf ein, wie dieses Maß mit dem gängigen Begriff der Laufzeit auf Rechnermodellen mit wahlfreiem Speicherzugriff in Verbindung gesetzt werden kann.) Diese Sichtweise ist sinnvoll, da die Zielfunktionsauswertungen häufig die rechenzeitintensivsten Operationen in einer solchen randomisierten Suchheuristik darstellen. Wenn nichts anderes erwähnt wird, entspricht die Größenordnung der Anzahl der Zielfunktionsauswertungen in den wiedergegebenen Ergebnissen der Größenordnung der Anzahl der Iterationen.

Scharnow et al. [2004] präsentierten eine Analyse eines einfachen evolutionären Algorithmus für das Kürzeste-Wege-Problem für eine Quelle. Beachte, dass das Kürzeste-Wege-Problem für eine Quelle auf das Kürzeste-Wege-Problem für eine Senke reduziert werden kann, indem die Richtungen aller Kanten in dem Graphen umgedreht werden. Deren Algorithmus (1+1)-EA verwaltet eine aktuelle Lösung v für das Problem, die durch einen Kürzeste-Wege-Baum repräsentiert wird, der sich aus den bisher gefundenen kürzesten Wegen ergibt. In jeder Iteration kommt ein Mutationsoperator zum Einsatz, der durch zufällige Änderungen in dem Kürzeste-Wege-Baum eine neue Lösung v' erzeugt. Wenn alle kürzesten Wege in v' nicht länger als der korrespondierende kürzeste Weg in v sind, dann wird v durch v' ersetzt. Die Autoren zeigen, dass dieser einfache Algorithmus das Kürzeste-Wege-Problem für eine Quelle auf jedem Graphen mit n Knoten im Erwartungswert in $\mathcal{O}(n^3)$ Iterationen löst. Dieses Ergebnis wurde später von Doerr et al. [2007a] verbessert, die gezeigt haben, dass im ungünstigsten Fall im Erwartungswert genau $\Theta(n^2 \cdot \ell^*)$ Iterationen benötigt werden, wobei $\ell^* := \max\{\ell, \ln(n)\}$ ist und ℓ die größte Anzahl von Kanten auf einem kürzesten Pfad bezeichnet. Der Vergleich von zwei Lösungen in dem Algorithmus

(1+1)-EA basiert auf einer multikriteriellen Formulierung des monokriteriellen Problems, da alle Pfade in der einen Lösung mit den korrespondierenden Pfaden in der anderen Lösung verglichen werden. Baswana et al. [2009] untersuchten eine Variante des Algorithmus (1+1)-EA, die eine monokriterielle Fitnessfunktion optimiert, und bewiesen die obere Schranke $\mathcal{O}(n^3 \cdot (\log(n) + \log(w_{\max})))$, wobei $w_{\max}$ das größte Gewicht in dem Graphen bezeichnet.

Doerr et al. [2008a] untersuchten einen genetischen Algorithmus, den wir mit GA abkürzen, für das Kürzeste-Wege-Problem für alle Paare, der sowohl einen Mutations- als auch einen Rekombinationsoperator benutzt, um neue Lösungen zu erzeugen. Eine Lösung wird von einer „Population“ von Pfaden mit unterschiedlichen Quelle-Senke-Paaren repräsentiert. Der Rekombinationsoperator fügt Teile aus verschiedenen Pfaden aneinander. Die Autoren bewiesen, dass der Einsatz von Rekombination verglichen mit einem evolutionären Algorithmus, der ausschließlich auf Mutation basiert, zu einer Verbesserung der Laufzeit führt. Eine Tatsache, die zuvor nur für künstliche Probleme [Jansen und Wegener, 2002] und ein einfaches Graphfärbungsproblem [Fischer und Wegener, 2005, Sudholt, 2005] gezeigt werden konnte. Genauer wird die erwartete Anzahl von Iterationen durch die Einführung eines Rekombinationsoperators von $\Theta(n^4)$ auf $\mathcal{O}(n^{3,5} \cdot \log^{0,5}(n))$ gesenkt. Die letztgenannte obere Schranke wurde anschließend von Doerr und Theile [2009] auf $O(n^{3,25} \cdot \log^{0,25}(n))$ Iterationen verbessert; in [Doerr und Theile, 2009] wurde auch eine Eingabeinstanz präsentiert, die zeigt, dass die verbesserte Schranke scharf ist. Außerdem untersuchte Horoba [2010] ein NP-schwieriges multikriterielles Kürzeste-Wege-Problem, bei dem jede Kante mit einem Gewichtsvektor versehen ist. Es wurde gezeigt, dass ein einfacher evolutionärer Algorithmus für multikriterielle Optimierungsprobleme ein echt polynomielles Approximationsschema für das Problem liefert. Das letztgenannte Ergebnis ist hier in Kapitel 7 zu finden.

In der Arbeit von Attiratanasunthron und Fakcharoenphol [2008] über Ameisensysteme für ein Kürzeste-Wege-Problem, die wir bereits weiter oben erwähnt haben, wurde die obere Schranke $\mathcal{O}(m \cdot \Delta \cdot \ell \cdot \log(\Delta \cdot \ell)/\rho)$ für alle gerichteten azyklischen Graphen mit m Kanten, Höchstgrad Δ und höchstens ℓ Kanten auf jedem *kürzesten* Pfad gezeigt. Der Evaporationsfaktor ρ ist ein Parameter des Algorithmus, der innerhalb der Grenzen $0 < \rho \leq 1$ beliebig gewählt werden kann. Die Analyse folgt der Analyse des bekannten Bellman-Ford-Algorithmus [Cormen et al., 2001]. Wir werden den Algorithmus und die Analyse noch genauer besprechen.

Wir fassen die Anzahl der benötigten Iterationen, um alle kürzesten Wege zu finden, und die Anzahl der Zielfunktionsauswertungen pro Iteration zusammen. Tabelle 9.1 gibt einen Überblick über die besten bekannten Schranken für verschiedene Suchheuristiken und monokriterielle Kürzeste-Wege-Probleme; die Tabelle enthält auch Ergebnisse, die in diesem Kapitel bewiesen werden.

9.1.2. Gliederung

Der Rest des vorliegenden Kapitels ist wie folgt aufgebaut. In Abschnitt 9.2 definieren wir den Ameisenalgorithmus $\text{MMAS}_{\text{SDSP}}$ für das Kürzeste-Wege-Problem

für einer Senke, der sich auf zwei Weisen essenziell von dem Algorithmus n-ANT [Attiratanasunthron und Fakcharoenphol, 2008] unterscheidet. Für unseren modifizierten Algorithmus beweisen wir deutlich verbesserte Laufzeitschranken (siehe Tabelle 9.1 und Abschnitt 9.3). Außerdem zeigen wir, dass der modifizierte Algorithmus nicht nur auf gerichtete azyklische Graphen sondern auch auf Graphen mit Kreisen anwendbar ist. Eine zugehörige untere Schranke zeigt, dass unsere oberen Schranken asymptotisch scharf sind, wenn der Evaporationsfaktor ρ nicht zu klein ist. In Abschnitt 9.4 übertragen wir die Ergebnisse auf ein verallgemeinertes Ameisensystem $\text{MMAS}_{\text{APSP}}$ für das Kürzeste-Wege-Problem für alle Paare, in dem sich Ameisen mit unterschiedlichen Zielen unabhängig voneinander bewegen. Das Hauptergebnis in diesem Abschnitt behandelt eine Modifikation des Ameisensystems $\text{MMAS}_{\text{APSP}}$, die es Ameisen ermöglicht, vorübergehend fremden Pheromonspuren zu folgen. Wir beweisen, dass dieser einfache Mechanismus überraschenderweise zu einer deutlichen Laufzeitverbesserung führt. Wir beschließen dieses Kapitel in Abschnitt 9.5 mit einigen Bemerkungen zu möglichen Verallgemeinerungen und offenen Problemen.

9.2. Algorithmen

Wir betrachten hier verschiedene Kürzeste-Wege-Probleme auf gewichteten gerichteten Graphen $G = (V, E, w)$, wobei $w(e)$ das Gewicht einer Kante e ist. Die Anzahl der Knoten bezeichnen wir stets mit n. Wir definieren einen *Pfad* der Länge ℓ von u nach v als eine Folge von Knoten $(v_0, \ldots, v_\ell)$, wobei $v_0 = u$, $v_\ell = v$ und $(v_{i-1}, v_i) \in E$ für alle i mit $1 \leq i \leq \ell$ gelten müssen. Der Einfachheit halber werden wir auch die zugehörige Folge von Kanten als Pfad bezeichnen. Wir bezeichnen mit $\deg(u)$ den Ausgrad eines Knotens u, d. h., die Anzahl der von u ausgehenden Kanten, und mit $\Delta(G)$ den größten Ausgrad aller Knoten $u \in V$. Seien $\ell(G, v) := \max_u\{\#\text{Kanten auf } p \mid p \text{ ist ein kürzester Pfad von } u \text{ nach } v\}$ und $\ell(G) := \max_v \ell(G, v)$. Für ungewichtete und ungerichtete Graphen werden $\ell(G, v)$ *Exzentrizität* von v und $\ell(G)$ *Durchmesser* von G genannt.

Das Kürzeste-Wege-Problem für eine Senke (SDSP) besteht darin, von jedem Knoten einen kürzesten Pfad zu der ausgezeichneten Senke zu finden. Die Länge $w(p)$ eines Pfades p ist durch die Summe der Gewichte der Kanten aus p definiert, wenn der Pfad an der Senke endet. Wenn der Pfad nicht an der Senke endet, setzen wir $w(p) := \infty$. Im Folgenden werden wir hauptsächlich Graphen mit positiven Gewichten betrachten, da die Präsenz von Kreisen negativer Länge dazu führt, dass beliebig kurze Pfade konstruiert werden können. Außerdem ist das Problem, einen kürzesten *einfachen* Pfad zu finden, NP-schwierig. Wir können das NP-schwierige Problem, einen Hamiltonkreis in einem Graphen $G = (V, E)$ zu finden, auf das Problem, einen kürzesten einfachen Pfad in einem gewichteten Graphen $G = (V, E, w)$ zu finden, reduzieren. Wenn wir $w(e) = -1$ für alle $e \in E$ setzen, dann gibt es einen kürzesten einfachen Pfad mit Gewicht $-n$ genau dann, wenn es einen Hamiltonkreis gibt. Beachte, dass wir einen einfachen Kreis als einen einfachen Pfad ansehen.

Attiratanasunthron und Fakcharoenphol [2008] präsentieren den Ameisenalgorith-

Algorithmus	Problem	Iterationen	Ausw.
n-ANT[1]	SDSP f. DAGs	$\mathcal{O}\left(\frac{m\Delta\ell\log(\Delta\ell)}{\rho}\right)$	n
$\text{MMAS}_{\text{SDSP}}$	SDSP	$\mathcal{O}\left(\Delta\ell\ell^* + \frac{\ell}{\rho}\right)$	n
$\text{MMAS}_{\text{SDSP}}$	SDSP f. $G^{\text{lb}}_{n,\ell}$	$\Omega\left(\Delta\ell^2 + \frac{\ell}{\rho\log(1/\rho)}\right)$	n
$\text{MMAS}_{\text{SDSP}}$ m. $\tau_{\min}$ adaptiv	SDSP	$\mathcal{O}\left(\ell m + \frac{n}{\rho}\right)$	n
(1+1)-EA[2]	SSSP	$\Theta\left(n^2\ell^*\right)$	1
$\text{MMAS}_{\text{APSP}}$	APSP	$\mathcal{O}\left(\Delta\ell\ell^* + \frac{\ell}{\rho}\right)$	n^2
$\text{MMAS}_{\text{APSP}}$ m. Interaktion	APSP	$\mathcal{O}\left(n\log n + \frac{\log(\ell)\log(\Delta\ell)}{\rho}\right)$	n^2
GA[3]	APSP	$\Theta\left(n^{3,25}\log^{0,25} n\right)$	1

[1] Siehe [Attiratanasunthron und Fakcharoenphol, 2008].
[2] Siehe [Doerr et al., 2007a].
[3] Siehe [Doerr und Theile, 2009].

Tabelle 9.1.: Übersicht über die besten bekannten Laufzeitschranken für Graphen mit n Knoten, m Kanten, Höchstgrad Δ, höchstens ℓ Kanten auf jedem kürzesten Pfad und $\ell^* := \max\{\ell, \ln n\}$. Die Schranken für die Ameisenalgorithmen sind in Abhängigkeit von dem Parameter ρ angegeben. Die rechteste Spalte enthält die Anzahl der Pfadlängenauswertungen pro Iteration. Für alle Ameisenalgorithmen ist der Parameter $\tau_{\min}$ optimal gewählt. Die Schranke für den Algorithmus $\text{MMAS}_{\text{APSP}}$ mit Interaktion gilt für alle $\rho \leq (1 - p_{\text{int}})/(12\Delta \log n)$; sie vereinfacht sich zu $\mathcal{O}(n \log^3 n)$, wenn ρ und p_{int} optimal gewählt werden.

mus n-ANT für das Kürzeste-Wege-Problem für eine Senke. Der Algorithmus ist sowohl von dem Algorithmus 1-ANT [Neumann und Witt, 2006] als auch von dem Routing-Algorithmus AntNet [Di Caro und Dorigo, 1998] inspiriert. Es wird an jedem Knoten $u \in V$ eine Ameise a_u simuliert, die nach der Senke sucht. Die Ameise konstruiert einen Pfad, indem sie zufallsgesteuert durch den Graphen läuft, wobei die Wahrscheinlichkeit, eine bestimmte Kante zu wählen, von den Pheromonwerten der Kanten abhängt, die den aktuellen Knoten verlassen. Die Ameise a_u merkt sich den kürzesten Pfad, den sie bislang von ihrer Quelle u zu der Senke gefunden hat. Wenn sie einen Pfad findet, der nicht länger als der beste bisher gefundene Pfad ist, dann werden die Pheromonwerte aktualisiert, wobei die Pheromonspur auf dem neuen Pfad intensiviert wird. Die Autoren benutzen ein rein lokales Aktualisierungsschema, bei dem jede Ameise a_u für die Aktualisierung der Pheromonwerte auf den Kanten, die ihre Quelle u verlassen, verantwortlich ist. Wenn der neue Pfad schlechter als der beste bisher gefundene Pfad ist, dann ändern sich die Pheromone auf den

Algorithmus 8 Pfadkonstruktion von u nach v für $\text{MMAS}_{\text{SDSP}}$

1: initialisiere $i \leftarrow 0$, $p_0 \leftarrow u$ und $V_1 \leftarrow \{p \in V \setminus \{p_0\} \mid (p_0, p) \in E\}$
2: **while** $p_i \neq v$ und $V_{i+1} \neq \emptyset$ **do**
3: $\quad i \leftarrow i + 1$
4: $\quad$ wähle $p_i \in V_i$ mit Wahrscheinlichkeit $\tau((p_{i-1}, p_i)) / \sum_{p \in V_i} \tau((p_{i-1}, p))$
5: $\quad V_{i+1} \leftarrow \{p \in V \setminus \{p_0, \ldots, p_i\} \mid (p_i, p) \in E\}$
6: **end while**
7: **return** $(p_0, \ldots, p_i)$

Kanten, die u verlassen, nicht.

Da Attiratanasunthron und Fakcharoenphol [2008] ausschließlich azyklische Graphen betrachten, ist das Ameisensystem n-ANT nicht darauf ausgelegt, mit Kreisen zurechtzukommen. Insbesondere stellen die Autoren fest, dass ihre Methode zur Pfadkonstruktion in Graphen mit Kreisen exponentiell lange dauern kann. Aus diesem Grund passen wir ihren Algorithmus an, indem wir den Ameisen nur erlauben, einfache Pfade zu konstruieren, d. h., eine Ameise kann einen Knoten nicht mehr als einmal besuchen. Die Wahl der Kante, der als nächstes gefolgt wird, kann folglich nur auf die Kanten fallen, die zu unbesuchten Knoten führen. Diese Einschränkung induziert das Risiko, dass die Ameise das gesuchte Ziel nicht erreicht. Wir erinnern daran, dass in diesem Fall die Länge des gefundenen Pfades p definitionsgemäß $w(p) = \infty$ entspricht. Aufgrund des lokalen Aktualisierungsschemas der Pheromone ist auch hier sichergestellt, das eine ausgehende Kante für alle Knoten u mit $\deg(u) \geq 1$ und $u \neq n$ belohnt wird. Die Pfadkonstruktionsmethode ist in Algorithmus 8 beschrieben.

Wir nennen unseren Ameisenalgorithmus $\text{MMAS}_{\text{SDSP}}$, da wir das aus dem Algorithmus MMAS [Neumann et al., 2009] bekannte Aktualisierungsschema, das die beste bisher gefundene Lösung intensiviert, verwenden und nicht das Aktualisierungsschema, das in dem Algorithmus 1-ANT zum Einsatz kommt. Der Unterschied besteht darin, dass wir die Pheromone nach jeder Iteration mithilfe der derzeit besten bisher gefundenen Lösung aktualisieren; dabei kann es sich entweder um den neuen Pfad handeln oder um den vorherigen besten bisher gefundenen Pfad, falls der neue Pfad schlechter ist.

Das Aktualisierungsschema für die Pheromone entspringt im Wesentlichen der Arbeit von Attiratanasunthron und Fakcharoenphol [2008]. Wir initialisieren die Pheromone $\tau\colon E \to \mathbb{R}_0^+$ so, dass alle Kanten, die einen Knoten u verlassen, die gleiche Pheromonmenge erhalten. Wenn $e = (u, \cdot)$ gilt, dann setzen wir $\tau(e) = 1/\deg(u)$. Wenn e die einzige Kante ist, die u verlässt, dann halten wir $\tau(e) = 1$ fest. Das bedeutet, dass Ameisen, die einen Knoten mit genau einer ausgehenden Kante verlassen möchten, dies auf die einzig mögliche Weise tun. Die genannten Knoten können daher vernachlässigt werden, wenn es darum geht, obere Schranken für die Laufzeit zu beweisen. Falls u mehr als eine ausgehende Kante besitzt, werden die Pheromone für $e = (u, v)$ wie folgt berechnet. Sei p_u^* der beste Pfad von u, der bislang gefunden wurde. Eingangs setzen wir p_u^* auf den leeren Pfad, sodass $w(p_u^*) = \infty$ gemäß der Definition von w gilt.

Algorithmus 9 $\text{MMAS}_{\text{SDSP}}$

1: initialisiere Pheromone τ und beste bisher gefundene Pfade $p_1^*, \ldots, p_n^*$
2: **repeat**
3: **for** $u = 1, \ldots, n$ **do**
4: konstruiere einfachen Pfad $p_u = (p_{u,0}, \ldots, p_{u,\ell_u})$ von u nach n bezüglich τ
5: **if** $w(p_u) \leq w(p_u^*)$ **then** $p_u^* \leftarrow p_u$ **end if**
6: **end for**
7: aktualisiere Pheromone τ bezüglich $p_1^*, \ldots, p_n^*$
8: **until** Abbruchkriterium erfüllt
9: **return** $p_1^*, \ldots, p_n^*$

Ein gemeinsames Alleinstellungsmerkmal von Ameisenalgorithmen ist, dass die Pheromone, die von vorangegangenen Ameisen abgelegt worden sind, im Laufe der Zeit verfliegen. Während der Aktualisierung der Pheromone wird gemeinhin angenommen, dass ein ρ-Anteil der alten Pheromone verdunstet und folglich ein $(1-\rho)$-Anteil fortbesteht. Neue Pheromone werden auf den Kanten platziert, die belohnt werden sollen. Eine weitere, weitverbreitete Methode in Ameisenalgorithmen besteht darin, eine untere und eine obere Schranke für die Pheromone festzulegen und diese einzuhalten [Stützle und Hoos, 2000, Attiratanasunthron und Fakcharoenphol, 2008]. Wir halten die Pheromone in einem Intervall $[\tau_{\min}, \tau_{\max}]$, wobei wir $\tau_{\min} > 0$ als Parameter des Algorithmus auffassen und in diesem Kapitel stets $\tau_{\max} = 1 - \tau_{\min}$ gilt. Diese Pheromonschranken stellen sicher, dass der Algorithmus jede feste Lösung mit positiver Wahrscheinlichkeit erzeugt. Dadurch wird garantiert, dass die erwartete Optimierungszeit endlich ist. Die Aktualisierung der Pheromone erfolgt gemäß der genauen Formel

$$\tau(e) \leftarrow \begin{cases} \min\{(1-\rho)\cdot\tau(e) + \rho,\ \tau_{\max}\} & \text{falls } e = (u,v) \in p_u^* \\ \max\{(1-\rho)\cdot\tau(e),\ \tau_{\min}\} & \text{falls } e = (u,v) \notin p_u^*. \end{cases} \tag{9.1}$$

Wir weisen darauf hin, dass Formel 9.1 genau der in dem Algorithmus n-ANT [Attiratanasunthron und Fakcharoenphol, 2008] benutzten Formel entspricht; von den Algorithmen n-ANT und $\text{MMAS}_{\text{SDSP}}$ wird die Aktualisierungsmethode jedoch zu unterschiedlichen Zeitpunkten aufgerufen.

Der vollständige Algorithmus $\text{MMAS}_{\text{SDSP}}$ wird in Algorithmus 9 beschrieben. Wir interessieren uns für die *Optimierungszeit* des Algorithmus $\text{MMAS}_{\text{SDSP}}$, die definitionsgemäß widerspiegelt, wie viele Iterationen benötigt werden, um kürzeste Pfade von $1, \ldots, n$ nach n zu finden. Wie bereits zuvor erwähnt stellt die Anzahl der Zielfunktionsauswertungen ein gebräuchliches Maß für die Analyse der Leistungsfähigkeit von Metaheuristiken dar. Beachte, dass in einer Iteration des Ameisensystems $\text{MMAS}_{\text{SDSP}}$ parallel n Ameisen simuliert werden, die n Lösungen konstruieren und demzufolge n Zielfunktionsauswertungen benötigen. Also ist die Anzahl der Zielfunktionsauswertungen um den Faktor n größer als die Anzahl der Iterationen. Wir konzentrieren uns im Folgenden auf die Anzahl der Iterationen.

Es erwächst die naheliegende Frage, wie die Optimierungszeit und die Laufzeit des Ameisensystems $\text{MMAS}_{\text{SDSP}}$ auf einem Rechnermodell mit wahlfreiem Speicherzugriff in Beziehung stehen. Wenn wir eine naive, auf Adjazenzlisten beruhende Implementierung betrachten, beträgt die Laufzeit für die Initialisierung des Ameisensystems $\mathcal{O}(m+n)$, da m Pheromonwerte und n beste bisher gefundene Pfade vorbereitet werden müssen. Im ungünstigsten Fall beträgt die erwartete Laufzeit für jede Iteration $\mathcal{O}(m \cdot n)$, weil n Ameisen jeweils einen *einfachen* Pfad konstruieren und die zufällige Konstruktion eines einfachen Pfades im Erwartungswert $\mathcal{O}(m)$ Rechenschritte erfordert. Beachte, dass die letzte Abschätzung für viele Eingabeinstanzen und Knoten zu pessimistisch ist. Elaboriertere Implementierungen könnten die Gesamtlaufzeit pro Iteration in vielen Fällen reduzieren. Außerdem ist der Einsatz mehrerer Rechner hilfreich, da jede Ameise ihren Pfad unabhängig von den anderen Ameisen konstruiert. Wenn wir beispielsweise n Maschinen zur Verfügung haben, um alle n Ameisen parallel zu simulieren, dann verbessert sich die Laufzeit pro Iteration um den Faktor $1/n$.

Um das Bild zu komplettieren, ist ein kurzer Vergleich mit problemspezifischen Algorithmen angebracht. Wir können natürlich nicht darauf hoffen, dass vielseitig einsetzbare Metaheuristiken – wie Ameisenalgorithmen – Algorithmen schlagen, die auf ein bestimmtes Problem zugeschnitten sind. Es ist wohlbekannt, dass Dijkstras Algorithmus das Kürzeste-Wege-Problem für eine Senke auf Graphen mit n Knoten und m Kanten in der Zeit $\mathcal{O}(m + n \cdot \log n)$ und das Kürzeste-Wege-Problem für alle Paare in der Zeit $\mathcal{O}(n \cdot m + n^2 \cdot \log n)$ löst [Cormen et al., 2001]. Die beste, derzeit bekannte, allgemeine Laufzeitschranke für das Kürzeste-Wege-Problem für alle Paare ist $\mathcal{O}(n^3 \cdot (\log^3 \log n)/\log^2 n)$ [Chan, 2007]. Alle Ameisenalgorithmen, die in diesem Kapitel diskutiert werden, können gemäß der Definition von Karger et al. [1993] als *pfadvergleichende Algorithmen* aufgefasst werden. Die letztgenannten Autoren haben auch die untere Schranke $\Omega(m \cdot n)$ für alle deterministischen und randomisierten pfadvergleichenden Algorithmen bewiesen. Diese Schranke gilt sogar für allgemeinere Funktionen, die Pfaden Gewichte zuordnen.

Bevor wir den Algorithmus $\text{MMAS}_{\text{SDSP}}$ analysieren, motivieren wir, warum es essenziell ist, dass von jedem Knoten eine Ameise ihre Suche startet, sogar wenn wir uns nur für einen kürzesten Pfad von einer einzigen Quelle zu einer einzigen Senke interessieren und uns auf einfache gerichtete azyklische Graphen beschränken. Wir stellen uns ein alternatives Ameisensystem $\text{MMAS}_{\text{SPSP}}$ (SPSP steht für „single-pair shortest path“) vor, in dem eine oder mehrere Ameisen ausschließlich von der Quelle starten und nach der Senke fahnden. Betrachte den folgenden Graphen $G = (V, E, w)$, der in Abbildung 9.1 skizziert ist. Er enthält eine einzige schwere Kante $(n-1, n)$ mit Gewicht $n-2$ und leichte Kanten $(u, n-1)$ für $u \leq n-2$, $(u, u+1)$ für $u \leq n-3$ und $(n-2, n)$ jeweils mit Gewicht 1.

An jedem Knoten $u \leq n-2$ muss sich eine Ameise entscheiden, ob sie sich nach $n-1$ bewegt oder auf dem kürzesten Pfad $1, 2, \ldots, n-2, n$ voranschreitet. Da anfangs die Pheromondichte auf allen Kanten übereinstimmt, beträgt die Wahrscheinlichkeit, dass eine Ameise dem kürzesten Pfad bis zum Knoten $n/2$ folgt, genau $2^{-n/2+1}$. Wir nehmen an, dass die Ameise den kürzesten Pfad verlässt bevor sie den Knoten $n/2$

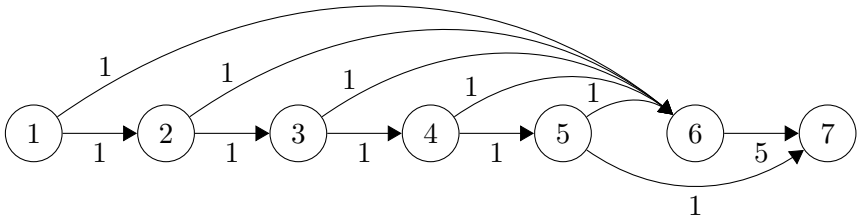


Abbildung 9.1.: Beispielgraph für $n = 7$.

erreicht. Da das Gewicht eines Pfades, der ℓ Kanten und den Knoten $n-1$ beinhaltet, $\ell + n - 3$ beträgt, werden im Folgenden abgesehen von dem optimalen Pfad, der $1, 2, \ldots, n-2, n$ durchläuft, keine Pfade mit einer größeren Anzahl von Kanten akzeptiert. Das bedeutet, dass die Pheromone auf den Kanten, die die Knoten $n/2, \ldots, n-2$ verlassen, solange übereinstimmen bis eine Ameise das Optimum gefunden hat. Die Wahrscheinlichkeit, das Optimum zu finden, beträgt $2^{-n/2+1}$. Wenn wir die boolesche Ungleichung verwenden, um die Wahrscheinlichkeit abzuschätzen, dass das Optimum in $2^{c \cdot n}$ Schritten gefunden wird, wobei $c > 0$ eine kleine Konstante ist, dann erhalten wir, dass die Optimierungszeit mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$ mindestens $2^{c \cdot n}$ beträgt. Beachte, dass Gleiches auch dann gilt, wenn polynomiell viele Ameisen pro Iteration parallel nach der Senke suchen. Indem wir das Gewicht der schweren Kante von $n-2$ auf $n-2+k$, $k \in \mathbb{N}$, erhöhen, sehen wir, dass das Ameisensystem nicht einmal einen Pfad berechnet, der das Gewicht eines kürzesten Pfades mit einem kleineren Approximationsfaktor als $(n-1+k)/(n-2) = 1+(k+1)/(n-2)$ annähert.

Auch die Zuhilfenahme der Kantengewichte als heuristische Informationen kann hier nicht helfen. Viele Ameisenalgorithmen benutzen sowohl Pheromone als auch heuristische Informationen, um die Konstruktion von Lösungskandidaten zu lenken [Dorigo und Stützle, 2004]. Ein gebräuchlicher Ansatz besteht darin, die Wahrscheinlichkeit in Zeile 4 von Algorithmus 8 gemäß

$$\frac{[\tau((p_{i-1}, p_i))]^\alpha \cdot [\eta((p_{i-1}, p_i))]^\beta}{\sum_{p \in V_i} [\tau((p_{i-1}, p))]^\alpha \cdot [\eta((p_{i-1}, p))]^\beta}$$

anzupassen, wobei $\eta(e) = 1/w(e)$ die heuristische Information bezeichnet und die Parameter $\alpha \in \mathbb{R}_0^+$ und $\beta \in \mathbb{R}_0^+$ die relative Wichtigkeit der Pheromone und der heuristischen Informationen steuern. Allerdings haben die beiden von einem Knoten $n/2 \leq u \leq n-2$ ausgehenden Kanten das gleiche Gewicht und mit hoher Wahrscheinlichkeit die gleiche Pheromondichte und sind folglich – nach wie vor – für alle Ameisen ununterscheidbar. Dieses Beispiel zeigt, dass die Verwendung von heuristischen Informationen für einige Probleminstanzen unnütz ist. Wenn wir das Gewicht der leichten Kanten nach Knoten $n-1$ verringern und das Gewicht der schweren Kante nach Knoten n erhöhen, dann sehen wir außerdem, dass heuristische Informationen im Allgemeinen sogar irreführend sein können. Wir werden heuristische

Informationen im Rahmen dieser Dissertation nicht weiter betrachten. Ameisensysteme, die auch auf die genannten Informationen zurückgreifen, werden beispielsweise in [Neumann und Witt, 2008, Kötzing et al., 2010] analysiert.

9.3. Das Kürzeste-Wege-Problem für eine Senke

9.3.1. Obere Schranken

Wenn Ameisen von unterschiedlichen Knoten starten, haben die Ameisen, die in der Nähe der Senke starten, eine gute Chance, einen kürzesten Pfad zu finden. Die Pheromone, die auf den ausgehenden Kanten eines Knotens v abgelegt sind, können dann dazu dienen, andere Ameisen, die den Knoten v durchlaufen, zu leiten. Auf diese Weise kann der von v ausgehende kürzeste Pfad zu längeren kürzesten Pfaden, die v enthalten, erweitert werden. Mit anderen Worten: Kürzeste Pfade werden schrittweise durch den gesamten Graphen verbreitet. Dies ist die grundlegende Idee der Analyse von Attiratanasunthron und Fakcharoenphol [2008], die wiederum auf der Analyse des bekannten Bellman-Ford-Algorithmus basiert [Cormen et al., 2001]. Die Ergebnisse in [Attiratanasunthron und Fakcharoenphol, 2008] besitzen ausschließlich für gerichtete azyklische Graphen Gültigkeit. Wir beginnen mit diesen Graphen und verallgemeinern die Resultate anschließend auf gerichtete Graphen mit Kreisen. Dabei verbessern wir die oberen Schranken von Attiratanasunthron und Fakcharoenphol [2008] und präsentieren untere Schranken, die – verglichen mit unseren besten oberen Schranken – für die meisten Evaporationsfaktoren ρ asymptotisch scharf sind.

Das folgende Lemma wird Verwendung finden, um die Wahrscheinlichkeit, dass eine Ameise eine bestimmte Kante auswählt, abzuschätzen.

Lemma 9.1. *Wenn $\tau_{\min} + \tau_{\max} = 1$ gilt, dann gilt für jeden Knoten u, $u \neq n$, mit* $\deg(u) > 1$ *stets*

$$1 \leq \sum_{e=(u,\cdot)\in E} \tau(e) \leq 1 + \deg(u) \cdot \tau_{\min}.$$

Beweis. In [Attiratanasunthron und Fakcharoenphol, 2008] wurde bereits die erste Ungleichung gezeigt. Eingangs beträgt die Summe der Pheromone genau 1. Nimm für eine Induktion an, dass $\sum \tau(e) \geq 1$ gilt. Wenn die Pheromone nicht aus dem gültigen Bereich $[\tau_{\min}, \tau_{\max}]$ herausfallen, dann erhalten wir $(1-\rho) \cdot \sum \tau(e) + \rho \geq 1$ als neue Summe. Falls ein Pheromonwert unter $\tau_{\min}$ fällt, kann das Setzen des Pheromonwertes auf $\tau_{\min}$ die Summe nur vergrößern. Wenn mindestens ein Pheromonwert die obere Schranke $\tau_{\max}$ erreicht oder überschritten hat, beträgt die Summe der Pheromone mindestens $\tau_{\min} + \tau_{\max} = 1$, da $\deg(u) > 1$ gilt.

Um die zweite Ungleichung einzusehen, beobachten wir, dass sich die Summe der Pheromone nur aufgrund der unteren Pheromonschranke vergrößern kann, da $(1-\rho) \cdot \sum \tau(e) + \rho \leq \sum \tau(e)$ aus $\sum \tau(e) \geq 1$ folgt. Betrachte eine Kante e mit $(1-\rho) \cdot \tau(e) < \tau_{\min}$. Verglichen mit diesem Wert erhöht sich der Pheromonwert höchstens um $\tau_{\min} \cdot \rho$, wenn er auf $\tau_{\min}$ gesetzt wird. Wenn aktuell $\sum \tau(e) \leq 1 + \deg(u) \cdot \tau_{\min}$ gilt, dann beträgt die Summe der nächsten Pheromonwerte höchstens $(1-\rho) \cdot (1 + \deg(u) \cdot$

$\tau_{\min}) + \rho + \deg(u) \cdot \tau_{\min} \cdot \rho = 1 + \deg(u) \cdot \tau_{\min}$. Somit ergibt sich auch die zweite Ungleichung durch Induktion. □

Als unmittelbare Konsequenz erhalten wir für die Ameise a_u den folgenden direkten Zusammenhang zwischen Pheromonen und Wahrscheinlichkeiten. Konkret betrachten wir die Wahrscheinlichkeit, dass die Ameise, die am Knoten u startet, eine bestimmte Kante $(u, \cdot)$ wählt, wenn $\tau_{\min} \leq 1/\deg(u)$ gilt. Die letztgenannte Bedingung ist sinnvoll, da $\tau_{\min}$ nicht oberhalb des initialen Pheromonwertes $1/\deg(u)$ gewählt werden sollte.

Korollar 9.2. *Wenn $\tau_{\min} \leq 1/\deg(u)$ und $\tau_{\min} + \tau_{\max} = 1$ für jede Kante $e = (u, \cdot)$, $u \neq n$, gelten, dann gilt*

$$\tau(e)/2 \leq \mathrm{W}(\textit{Ameise } a_u \textit{ wählt Kante } e) \leq \tau(e).$$

Die untere Schranke gilt auch für jede andere Ameise, die den Knoten u verlässt, und jede Kante $e = (u, v)$, sofern die Ameise zuvor v noch nicht besucht hat. Die obere Schranke gilt auch für jede andere Ameise, die Knoten u verlässt, und jede Kante $e = (u, \cdot)$, sofern die Ameise zuvor noch keinen Nachfolger von u besucht hat.

Die vorletzte Aussage gilt, da sich die Wahrscheinlichkeit, eine Kante $e = (u, v)$ zu einem unbesuchten Knoten v zu wählen, erhöht, wenn andere Nachfolger von u zuvor bereits besucht worden sind. Insbesondere ist $\tau_{\min}/2$ stets eine untere Schranke für die Wahrscheinlichkeit, eine feste ausgehende Kante zu wählen. Dies ist eine Verbesserung von Lemma 1 aus [Attiratanasunthron und Fakcharoenphol, 2008]. Wir weisen darauf hin, dass durch die Verwendung des verbesserten Lemmas in [Attiratanasunthron und Fakcharoenphol, 2008] die Laufzeitschranken für den Algorithmus n-ANT durch m/n geteilt werden können, wobei m die Anzahl der Kanten bezeichnet.

Das nachfolgende Korollar begrenzt die Wahrscheinlichkeit, dass eine Ameise, die bislang einem kürzesten Pfad gefolgt ist, einer inkorrekten Kante folgt, die den letzten Knoten des Pfades verlässt. Dabei nennen wir eine Kante $e = (u, \cdot)$ *korrekt*, wenn es einen kürzesten Pfad von u nach n gibt, der mit e beginnt; anderenfalls nennen wir sie *inkorrekt*.

Korollar 9.3. *Betrachte einen gerichteten Graphen $G = (V, E, w)$, der azyklisch ist oder in dem alle Kantengewichte positiv sind. Seien $u, v \in V \setminus \{n\}$. Wir nehmen an, dass die Ameise a_u einen kürzesten Pfad von u nach v konstruiert hat, wobei v ein Knoten auf einem kürzesten Pfad von u nach n ist. Wenn $\tau_{\min} \leq 1/\deg(u)$ und $\tau_{\min} + \tau_{\max} = 1$ gelten, dann gilt*

$$\mathrm{W}(\textit{Ameise } a_u \textit{ wählt irgendeine inkorrekte Kante } (v, \cdot)) \leq \tau_{inc},$$

wobei τ_{inc} die Summe der Pheromonwerten auf den inkorrekten Kanten $(v, \cdot)$ bezeichnet.

Beweis. Die Behauptung folgt unmittelbar aus Korollar 9.2, wenn der Graph azyklisch ist. Wenn wir Graphen mit Kreisen betrachten, kann es passieren, dass einige der Kanten $(v, \cdot)$ zu Knoten führen, die die Ameise bereits auf ihrem Weg von u nach v besucht hat. Sei w ein solcher Knoten. Es kann nicht passieren, dass w auf einem kürzesten Pfad von v nach n liegt, da wir annehmen, dass v auf einem kürzesten Pfad von u nach n liegt, dass w auf einem kürzesten Pfad von u nach v liegt und dass alle Kantengewichte positiv sind. Demzufolge muss es sich bei (v, w) um eine inkorrekte Kante handeln. Wir beweisen im Folgenden, dass die Wahrscheinlichkeit, irgendeine inkorrekte Kante zu wählen, nicht dadurch wächst, dass wir verhindern, dass die Ameise eine inkorrekte Kante zu einem bereits besuchten Knoten entlangläuft.

Bezeichne τ_{cor} die Summe der Pheromonwerte auf korrekten Kanten $(v, \cdot)$ und bezeichne $\tau_{\text{inc+vis}}$ die Summe der Pheromonwerte auf inkorrekten Kanten $(v, \cdot)$, die zu einem bereits besuchten Knoten führen. Dann beträgt die Wahrscheinlichkeit, dass die Ameise a_u irgendeine inkorrekte Kante $(v, \cdot)$ wählt, höchstens

$$\frac{\tau_{\text{inc}} - \tau_{\text{inc+vis}}}{\tau_{\text{cor}} + \tau_{\text{inc}} - \tau_{\text{inc+vis}}} \leq \frac{\tau_{\text{inc}}}{\tau_{\text{cor}} + \tau_{\text{inc}}} \leq \tau_{\text{inc}},$$

wobei die zweite Ungleichung aus der Ungleichung $\tau_{\text{cor}} + \tau_{\text{inc}} \geq 1$ folgt, die wiederum Lemma 9.1 entspringt. □

Das folgende Theorem liefert obere Schranken für den Algorithmus $\text{MMAS}_{\text{SDSP}}$, die jeweils aus zwei additiven Termen bestehen. Intuitiv spiegeln die ersten Terme die Wartezeiten wider, bis für alle Knoten kürzeste Pfade gefunden sind. Die zweiten Terme wachsen mit $1/\rho$. Sie reflektieren die Wartezeiten, bis sich die Pheromone nach einer Veränderung eines besten bisher gefundenen Pfades angepasst haben. Diese Zeit wird von Neumann et al. [2009] anschaulich *Gefrierzeit* (auf Englisch „freezing time“) genannt.

Theorem 9.4. *Betrachte einen gerichteten azyklischen Graphen G mit n Knoten und beliebigen (möglicherweise negativen) Kantengewichten. Seien $\Delta := \Delta(G)$ und $\ell := \ell(G)$. Wenn $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ und $\tau_{\max} = 1 - \tau_{\min}$ gelten, dann beträgt die erwartete Optimierungszeit des Algorithmus MMAS_{SDSP} angewandt auf G höchstens $\mathcal{O}(n/\tau_{\min} + n \cdot \log(1/\tau_{\min})/\rho)$. Die Schranke vereinfacht sich für $\tau_{\min} = 1/n^2$ zu $\mathcal{O}(n^3 + n \cdot \log(n)/\rho)$ und für $\tau_{\min} = 1/(\Delta \cdot \ell)$ zu $\mathcal{O}(n \cdot \Delta \cdot \ell + n \cdot \log(\Delta \cdot \ell)/\rho)$.*

Die beiden Spezialfälle $\tau_{\min} = 1/(\Delta \cdot \ell)$ und $\tau_{\min} = 1/n^2$ stellen für uns die interessantesten Fälle dar, da die erste Parameterwahl die beste obere Schranke unter den festgelegten Vorbedingungen liefert. Die zweite Parameterwahl macht sich andererseits – abgesehen von der Knotenanzahl n – keine Eigenschaften des vorliegenden Graphen zunutze; sie kann auch dann benutzt werden, wenn Δ und ℓ unbekannt sind. Während man die Knotenanzahl n kennt und auch der Höchstgrad $\Delta \leq n$ einfach zu bestimmen ist, ist die Grenze $\ell < n$ für die Anzahl von Kanten auf kürzesten Pfaden in der Regel unbekannt. Insofern wird man, wenn man das Ameisensystem einsetzen möchte, vermutlich auf die Pheromonschranke $\tau_{\min} = 1/n^2$ bzw. $\tau_{\min} = 1/(\Delta \cdot n)$ zurückgreifen.

Beweis von Theorem 9.4. Wir lehnen in diesem Beweis unsere Argumentation an die Analyse von Attiratanasunthron und Fakcharoenphol [2008] an. Wenn nennen eine Kante (u, v) *inkorrekt*, wenn sie nicht zu einem beliebigen kürzesten Pfad von u nach n gehört. Wir nennen einen Knoten u *abgearbeitet*, wenn ein kürzester Pfad von u nach n gefunden wurde und wenn die Pheromonwerte $\tau(\cdot)$ auf allen Kanten $(u, \cdot)$, die inkorrekt sind, $\tau_{\min}$ betragen.

Wir schätzen die erwartete Zeit, bis ein Knoten u abgearbeitet ist, ab, wobei wir voraussetzen, dass alle Knoten, die von u auf einem kürzesten Pfad von u nach n erreichbar sind, bereits abgearbeitet sind. Wir betrachten zunächst die erwartete Zeit bis ein kürzester Pfad von u nach n zum ersten Mal gefunden wird. Wir sagen auch, dass der Knoten u in diesem Fall *optimiert* ist. Aufgrund von Korollar 9.2 beträgt die Wahrscheinlichkeit, eine Kante zu wählen, die zu einem kürzesten Pfad von u nach n gehört, mindestens $\tau_{\min}/2$. Ein solcher kürzester Pfad wird gefunden, wenn die Ameise keine inkorrekte Kante wählt, bis sie n erreicht. Da alle Knoten auf allen kürzesten Pfaden abgearbeitet sind, sind an einem Knoten v die Pheromonwerte auf allen inkorrekten Kanten durch $\tau_{\min}$ gegeben und die Wahrscheinlichkeit, eine inkorrekte Kante zu wählen, beträgt höchstens $\deg(v) \cdot \tau_{\min}$ aufgrund von Korollar 9.2. Demzufolge beträgt die Wahrscheinlichkeit, eine Kante auf einem kürzesten Pfad zu wählen, mindestens $1 - \deg(v) \cdot \tau_{\min} \geq 1 - 1/\ell$, wenn $\tau_{\min} \leq 1/(\deg(v) \cdot \ell)$ gilt. Da alle kürzesten Pfade aus höchstens ℓ Kanten bestehen, beträgt die Wahrscheinlichkeit, dass keine inkorrekte Kante gewählt wird, mindestens $(1 - 1/\ell)^{\ell-1} \geq 1/e$, wobei $e = \exp(1)$ ist. Insgesamt ist die Wahrscheinlichkeit, einen kürzesten Pfad von u nach n zu finden, nicht kleiner als $\tau_{\min}/(2 \cdot e)$.

Die erwartete Zeit, bis u optimiert ist, beträgt folglich höchstens $2 \cdot e/\tau_{\min}$. Anschließend wird ein kürzester Pfad von u nach n aufgrund des Aktualisierungsschemas, das die beste bisher gefundene Lösung intensiviert, automatisch in jeder Iteration verstärkt. Der genaue Pfad kann sich ändern, aber es ist sichergestellt, dass nur kürzeste Pfade belohnt werden und sich infolgedessen die Pheromone auf inkorrekten Kanten in jedem Schritt verringern. In [Attiratanasunthron und Fakcharoenphol, 2008] wurde bereits in Lemma 2 festgehalten, dass $\ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen genügen, um einen Knoten abzuarbeiten, da die einfache Rechnung

$$(1 - \rho)^{\ln(\tau_{\max}/\tau_{\min})/\rho} \cdot \tau_{\max} < e^{-\ln(\tau_{\max}/\tau_{\min})} \cdot \tau_{\max} = \tau_{\min}$$

sicherstellt, dass die Pheromondichte auch im ungünstigsten Fall von $\tau_{\max}$ auf $\tau_{\min}$ fällt. Demzufolge wird die erwartete Zeit, bis u abgearbeitet ist, durch $2 \cdot e/\tau_{\min} + \ln(\tau_{\max}/\tau_{\min})/\rho$ beschränkt.

Sei $v_1, \ldots, v_n$ eine topologische Ordnung der Knoten aus V mit $v_n = n$. Da wir gerichtete azyklische Graphen betrachten, enthalten alle kürzesten Pfade von v_i nach v_n ausschließlich Knoten aus $\{v_{i+1}, \ldots, v_n\}$. Wenn $v_{i+1}, \ldots, v_n$ abgearbeitet sind, dann können wir die Zeit, bis v_i abgearbeitet ist, mithilfe der obigen Argumentation abschätzen. Die erwartete Zeit, bis alle Knoten abgearbeitet sind, ist durch $n \cdot 2 \cdot e/\tau_{\min} + n \cdot \ln(\tau_{\max}/\tau_{\min})/\rho = \mathcal{O}(n/\tau_{\min} + n \cdot \log(1/\tau_{\min})/\rho)$ begrenzt. □

Beachte, dass der Algorithmus $\text{MMAS}_{\text{SDSP}}$, nachdem er einen kürzesten Pfad von u

nach n gefunden hat, sicherstellt, dass die Pheromone auf den von u ausgehenden inkorrekten Kanten in den nächsten $F = \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen auf den Wert $\tau_{\min}$ fallen und diese dann dauerhaft „eingefroren“ werden. Der Algorithmus n-ANT aus [Attiratanasunthron und Fakcharoenphol, 2008] hingegen aktualisiert die Pheromone nur dann, wenn er einen neuen besten bisher gefundenen Pfad konstruiert hat. Das impliziert, dass ein kürzester Pfad von u nach n mehrfach – im ungünstigsten Fall in F verschiedenen Iterationen – gefunden werden muss, um die Pheromone auf die gleiche Weise einzufrieren. Folglich verbessert die Verwendung des die beste bisher gefundene Lösung intensivierenden Aktualisierungsschemas, das in MMAS-Algorithmen zum Einsatz kommt, in diesem Fall die Leistung. Diese Resultate ergänzen den Vergleich der Ameisensysteme 1-ANT und MMAS auf pseudobooleschen Problemen in [Neumann et al., 2009].

In dem Beweis von Theorem 9.4 haben wir pessimistischerweise angenommen, dass wir, nachdem der Knoten v optimiert worden ist, $\lceil \ln(\tau_{\max}/\tau_{\min})/\rho \rceil$ Iterationen warten müssen, um anschließend einen längeren kürzesten Pfad, der v enthält, mit einer nicht zu kleinen Wahrscheinlichkeit zu finden. Diese pessimistische Annahme kann wie folgt relaxiert werden. Statt $\lceil \ln(\tau_{\max}/\tau_{\min})/\rho \rceil$ Iterationen zu warten, reicht es aus, nach der Optimierung eines neuen Knotens, $\lceil 3/\rho \rceil$ Iterationen zu warten, um zu garantieren, dass ein kürzester Pfad mit einer guten Wahrscheinlichkeit gefunden wird. Der Einfachheit halber nehmen wir im Folgenden ohne Beschränkung der Allgemeinheit an, dass $3/\rho$ ganzzahlig ist.

Definition 9.5. *Betrachte einen gewichteten gerichteten Graphen mit einem Zielknoten n und Pheromonen auf allen Kanten. Dann heißt ein Knoten u genau dann* fast abgearbeitet*, wenn für jeden kürzesten Pfad $\mathcal{P}$ von u nach n die folgende Bedingung erfüllt ist. Wenn $\mathcal{P} = (p_1, \ldots, p_r, n)$ mit $u = p_1$ gilt, dann sind die Knoten p_i für alle $1 \leq i \leq r$ seit mindestens $3 \cdot i/\rho$ Iterationen optimiert.*

Die Definition eines fast abgearbeiteten Knotens spiegelt die Tatsache wider, dass Knoten, die sich (bezogen auf die minimale Anzahl von Kanten auf kürzesten Pfaden) nahe am Zielknoten befinden, typischerweise früher als Knoten, deren kürzeste Wege viele Kanten enthalten, optimiert werden. Da die erstgenannten Knoten schon länger optimiert sind, hatten die Pheromone auf den inkorrekten Kanten folglich schon mehr Zeit, um „einzufrieren“. Ähnliche Argumente wurden auch von Neumann et al. [2009] für die Analyse eines Ameisenalgorithmus für pseudoboolesche Optimierung und von Sudholt und Witt [2008] für die Analyse eines binären Partikelschwarmoptimierers benutzt.

Lemma 9.6. *Betrachte einen gewichteten gerichteten Graphen G mit einem Zielknoten n, der azyklisch ist oder in dem alle Kantengewichte positiv sind. Dann gilt für jeden festen Knoten u das Folgende. Wenn alle Nachfolger von u auf kürzesten Pfaden von u nach n fast abgearbeitet sind und $\tau_{\min} \leq 1/(\Delta(G) \cdot \ell(G))$ gilt, dann beträgt die Wahrscheinlichkeit, dass u in einer Iteration optimiert wird, mindestens $\tau_{\min}/(4 \cdot e)$.*

Wenn u optimiert ist, dann gilt nach $3/\rho$ weiteren Iterationen, dass u fast abgearbeitet ist.

Beweis. Sei $\ell := \ell(G)$. Betrachte einen Knoten v und ignoriere vorübergehend die untere Pheromonschranke $\tau_{\min}$. Die Summe der Pheromone auf inkorrekten Kanten $(v, \cdot)$ ist aufgrund von Lemma 9.1 immer durch $1 + \deg(v) \cdot \tau_{\min} \leq 1 + 1/\ell \leq 2$ nach oben beschränkt. Wenn v seit t Iterationen optimiert ist, beträgt die Summe dieser Pheromone höchstens $2 \cdot (1-\rho)^t$. Die Einhaltung der unteren Pheromonschranke führt für jede Kante zu einer Abweichung von höchstens $\tau_{\min}$. Demzufolge beträgt die Summe der Pheromone auf inkorrekten Kanten – unter Berücksichtigung der Pheromonschranken – höchstens $\deg(v) \cdot \tau_{\min} + 2 \cdot (1-\rho)^t \leq 1/\ell + 2 \cdot (1-\rho)^t$. Die Wahrscheinlichkeit, irgendeine inkorrekte Kante zu wählen, beträgt aufgrund von Korollar 9.3 höchstens $1/\ell + 2 \cdot (1-\rho)^t$, wenn wir voraussetzen, dass die Ameise bislang einem kürzesten Pfad gefolgt ist. Die Wahrscheinlichkeit, eine Kante auf einem kürzesten Pfad zu wählen, beträgt folglich mindestens $1 - 1/\ell - 2 \cdot (1-\rho)^t$.

Wir schließen genau wie im Beweis von Theorem 9.4, dass eine Ameise die erste Kante mindestens mit Wahrscheinlichkeit $\tau_{\min}/2$ richtig wählt. Dies beweist die erste Behauptung für $\ell = 1$. Wir nehmen im Folgenden $\ell \geq 2$ an und beobachten, dass $1 - 1/\ell - 2 \cdot (1-\rho)^t \geq 1 - 1/\ell - 2 \cdot e^{-3 \cdot j}$ für $t = 3 \cdot j/\rho$ und für alle $j \in \mathbb{N}$ gilt. Wenn unsere Vorbedingungen für u gelten, beträgt die Wahrscheinlichkeit für eine Ameise a_u, einen kürzesten Pfad von u zu finden, mindestens

$$\begin{aligned}
\frac{\tau_{\min}}{2} \cdot \prod_{j=1}^{\ell-1} \left(1 - \frac{1}{\ell} - 2 \cdot e^{-3 \cdot j}\right) &\geq \frac{\tau_{\min}}{2} \cdot \prod_{j=1}^{\ell-1} \left(1 - \frac{1}{\ell} - 4 \cdot e^{-3 \cdot j} + \frac{4 \cdot e^{-3 \cdot j}}{\ell}\right) \\
&= \frac{\tau_{\min}}{2} \cdot \prod_{j=1}^{\ell-1} \left(\left(1 - \frac{1}{\ell}\right) \cdot \left(1 - 4 \cdot e^{-3 \cdot j}\right)\right) \\
&\geq \frac{\tau_{\min}}{2 \cdot e} \cdot \prod_{j=1}^{\ell-1} \left(1 - 4 \cdot e^{-3 \cdot j}\right).
\end{aligned}$$

Unter Verwendung von $1 - x \geq e^{-2 \cdot x}$ für $0 \leq x \leq 1/2$, begrenzen wir den $\prod$-Term abschließend durch

$$\begin{aligned}
\prod_{j=1}^{\ell-1} \left(1 - 4 \cdot e^{-3 \cdot j}\right) \geq \prod_{j=1}^{\ell-1} \exp\left(-8 \cdot e^{-3 \cdot j}\right) &\geq \exp\left(-8 \cdot \sum_{j=1}^{\infty} e^{-3 \cdot j}\right) \\
&= \exp\left(-\frac{8}{e^3 - 1}\right) \geq \frac{1}{2}.
\end{aligned}$$

Dies beweist die erste Behauptung. Die zweite Behauptung folgt aus der Definition eines fast abgearbeiteten Knotens, da alle Knoten auf kürzesten Pfaden von u (u eingeschlossen) nur für höchstens $3/\rho$ weitere Iterationen optimiert sein müssen. □

Lemma 9.6 kann benutzt werden, um die aus Theorem 9.4 bekannte obere Schranke von $\mathcal{O}(n/\tau_{\min} + n \log(1/\tau_{\min})/\rho)$ auf $\mathcal{O}(n/\tau_{\min} + n/\rho)$ zu verkleinern. Dies ist verglichen mit dem vorausgegangenen Ergebnis aus [Horoba und Sudholt, 2009] eine

Verbesserung. Wir vollziehen einen weiteren Schritt und beweisen, dass die letztgenannte Schranke für alle gerichteten Graphen mit positiven Kantengewichten gilt.

Im Beweis von Theorem 9.4 haben wir ausgenutzt, dass ein Knoten effizient optimierbar ist, wenn alle Nachfolger auf allen kürzesten Pfaden abgearbeitet sind. Genauer gesagt haben wir eine feste Reihenfolge $v_1, \ldots, v_n$ der Knoten betrachtet, sodass alle kürzesten Pfade von v_i, wobei v_i den letzten noch nicht optimierte Knoten bezeichnet, ausschließlich Knoten aus $v_{i+1}, \ldots, v_n$ benutzen. Wenn ein allgemeiner gerichteter Graph vorliegt, können wir eine topologische Sortierung nicht als Argumentationsgrundlage gebrauchen. Es könnte sogar schlimmer noch passieren, dass es eine Reihenfolge mit den gewünschten Eigenschaften nicht gibt. Beispielsweise existiert die gewünschte Reihenfolge nicht, wenn der Graph einen gerichteten Kreis aus Kanten mit Gewicht 0 enthält. Wenn wir voraussetzen, dass die Gewichte echt positiv sind, können wir jedoch die Länge eines kürzesten Pfades benutzen, um die Knoten auf die gewünschte Weise zu sortieren.

Theorem 9.7. *Betrachte einen beliebigen gerichteten Graphen G mit n Knoten und (echt) positiven Kantengewichten. Seien $\Delta := \Delta(G)$ und $\ell := \ell(G)$. Wenn $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ und $\tau_{\max} = 1 - \tau_{\min}$ gelten, beträgt die erwartete Optimierungszeit des Algorithmus $MMAS_{SDSP}$ angewandt auf G höchstens $\mathcal{O}(n/\tau_{\min} + n/\rho)$. Die Schranke vereinfacht sich für $\tau_{\min} = 1/n^2$ zu $\mathcal{O}(n^3 + n/\rho)$ und für $\tau_{\min} = 1/(\Delta \cdot \ell)$ zu $\mathcal{O}(n \cdot \Delta \cdot \ell + n/\rho)$.*

Beweis. Sei $v_1, v_2, \ldots, v_n$ eine Auflistung aller Knoten, sodass die Länge eines kürzesten Pfades nach n abnimmt. Beachte, dass $v_n = n$ gilt. Da alle Gewichte positiv sind, durchlaufen alle kürzesten Wege von v_i nach v_n ausschließlich Knoten aus $\{v_{i+1}, \ldots, v_n\}$. Wenn wir einem beliebigen kürzesten Pfad folgen, dann gilt außerdem, dass die Indizes der durchlaufenen Knoten streng monoton wachsen.

Wir gliedern einen Lauf des Algorithmus in Phasen, die in absteigender Reihenfolge durchlaufen werden. Phase n beginnt unmittelbar nach der Initialisierung. Für $1 \leq i \leq n$ gilt, dass Phase i endet, sobald alle Knoten aus $\{v_i, \ldots, v_n\}$ fast abgearbeitet sind. Nachdem Phase i beendet ist, beginnt Phase $i-1$, wenn $i > 1$ ist. Beachte, dass, falls v_i zufälligerweise vor v_{i+1} fast abgearbeitet ist, Phase i leer ist. Beachte außerdem, dass die Zeit, bis Phase 1 beendet ist, eine obere Schranke für die Optimierungszeit darstellt. Halte i fest und nimm an, dass Phase $i+1$ beendet ist. Da alle Knoten aus $\{v_{i+1}, \ldots, v_n\}$ fast abgearbeitet sind, sind die Bedingungen aus Lemma 9.6 erfüllt. Folglich ist die Wahrscheinlichkeit, v_i in einer Iteration zu optimieren, durch $\tau_{\min}/(4 \cdot e)$ nach unten begrenzt. Die erwartete Wartezeit auf dieses Ereignis beträgt höchstens $4 \cdot e/\tau_{\min}$. Aufgrund der zweiten Behauptung aus Lemma 9.6 ist auch v_i nach $3/\rho$ weiteren Schritten fast abgearbeitet und Phase i endet.

Die erwartete Zeit, die der Algorithmus in Phase i verbringt, ist durch $4 \cdot e/\tau_{\min} + 3/\rho$ beschränkt. Also ist die erwartete Zeit, bis Phase 1 abgeschlossen ist, durch $4 \cdot e \cdot n/\tau_{\min} + 3 \cdot n/\rho = \mathcal{O}(n/\tau_{\min} + n/\rho)$ beschränkt. □

Bis jetzt haben wir in unseren Analysen unterstellt, dass die Knoten einer nach dem anderen optimiert und abgearbeitet werden. Die bisher gezeigten oberen Schranken

für die Optimierungszeit basieren auf dieser pessimistischen Annahme. In bestimmten Graphen kann es jedoch Teilgraphen geben, die parallel optimiert werden können. Wir werden unsere oberen Schranken unter einigen zusätzlichen Annahmen um den Faktor ℓ/n verbessern. Dabei werden wir auch beweisen, dass die Schranken für die Optimierungszeit mit hoher Wahrscheinlichkeit gelten (d. h., mit Wahrscheinlichkeit mindestens $1 - n^{-c}$ für ein $c > 0$). Im Beweis werden wir auf Ideen aus [Doerr et al., 2007a] zurückgreifen, die für einen evolutionären Algorithmus gezeigt haben, dass die zufällige Zeit, bis ein kürzester Pfad der Länge $\ell = \Omega(\log n)$ gefunden wird, im Wesentlichen um den Erwartungswert konzentriert ist.[1]

Theorem 9.8. *Betrachte einen gerichteten Graphen G mit n Knoten und positiven Kantengewichten, in dem alle kürzesten Pfade eindeutig sind. Seien $\Delta := \Delta(G)$, $\ell := \ell(G)$ und $\ell^* := \max\{\ell, \ln n\}$. Wenn $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ und $\tau_{\max} = 1 - \tau_{\min}$ gelten, dann beträgt die Optimierungszeit des Algorithmus $MMAS_{APSP}$ angewandt auf G mit Wahrscheinlichkeit mindestens $1 - 1/n^2$ höchstens $\mathcal{O}(\ell^*/\tau_{\min} + \ell/\rho)$. Die Schranke vereinfacht sich für $\tau_{\min} = 1/n^2$ zu $\mathcal{O}(\ell^* \cdot n^2 + \ell/\rho)$ und für $\tau_{\min} = 1/(\Delta \cdot \ell)$ zu $\mathcal{O}(\Delta \cdot \ell \cdot \ell^* + \ell/\rho)$. Die Schranke für die Optimierungszeit gilt auch im Erwartungswert.*

Beweis. Halte einen Knoten u und den eindeutig bestimmten kürzesten Pfad $u = v_{\ell'}, v_{\ell'-1}, \ldots, v_0 = n$ mit $\ell' \leq \ell$ fest. Wir schätzen die erwartete Zeit, bis u fast abgearbeitet ist, pessimistisch ab, wobei wir die Phasenargumentation aus dem letzten Beweis auf formalere Weise benutzen. Sei T_i die zufällige Zeit, bis v_i optimiert ist. Betrachte Zufallsvariable X_i, $i \in \mathbb{N}^+$, die unabhängig voneinander mit Wahrscheinlichkeit $\tau_{\min}/(4 \cdot e)$ auf 1 und anderenfalls auf 0 gesetzt werden. Sei $T_i^* := \min\{j \in \mathbb{N}^+ \mid \sum_{k=1}^{j} X_k = i\}$. Der zufällige minimale Index T_1^*, für den $X_{T_1^*} = 1$ gilt, dominiert die zufällige Zeit, bis v_1 optimiert ist, stochastisch. Da v_1 nach einer zusätzlichen Wartezeit von $3/\rho$ Schritten ebenfalls fast abgearbeitet ist, wird T_1 von $T_1^* + 3/\rho$ stochastisch dominiert. Wir schließen per Induktion, dass $T_{\ell'}$ von $T_{\ell'}^* + 3 \cdot \ell'/\rho$ stochastisch dominiert wird. Definitionsgemäß wird dann auch die Zeit, bis u fast abgearbeitet ist, stochastisch dominiert.

Seien $T := 32 \cdot e \cdot \ell^*/\tau_{\min}$ und $X := \sum_{i=1}^{T} X_i$. Dann gilt $\mathrm{E}(X) = T \cdot \tau_{\min}/(4 \cdot e) = 8 \cdot \ell^*$. Aufgrund der ersten Chernoff-Ungleichung aus Lemma A.6 gilt

$$\mathrm{W}(X < \ell^*) \leq \mathrm{W}(X \leq (1 - 7/8) \cdot \mathrm{E}(X)) \leq e^{-8 \cdot \ell^* \cdot (7/8)^2/2} < e^{-3 \cdot \ell^*} \leq n^{-3}.$$

Also ist die Wahrscheinlichkeit, dass u nach $T + 3 \cdot \ell/\rho$ Schritten nicht fast abgearbeitet ist, durch $1/n^3$ beschränkt. Durch eine Anwendung der booleschen Ungleichung (siehe Lemma A.4) schließen wir, dass die Wahrscheinlichkeit, dass nach dieser Zeit ein noch nicht fast abgearbeiteter Knoten verblieben ist, höchstens $1/n^2$ beträgt. Das

[1] Hier manifestiert sich ein subtiler Unterschied zu [Doerr et al., 2007a]: In deren Definition von ℓ betrachten die letztgenannten Autoren nur kürzeste Wege *mit einer minimalen Anzahl von Kanten*, d. h., sie definieren formal $\ell(G, v) := \max_u \min_{p \in \mathcal{P}_{u,v}} \{\#\text{Kanten auf } p\}$, wenn $\mathcal{P}_{u,v}$ die Menge aller kürzesten Pfade von u nach v bezeichnet. Unsere Definition entspricht formal $\ell(G, v) := \max_u \max_{p \in \mathcal{P}_{u,v}} \{\#\text{Kanten auf } p\}$. Beide Definition von ℓ stimmen jedoch überein, wenn alle kürzesten Pfade eindeutig sind oder die gleiche Anzahl von Kanten aufweisen.

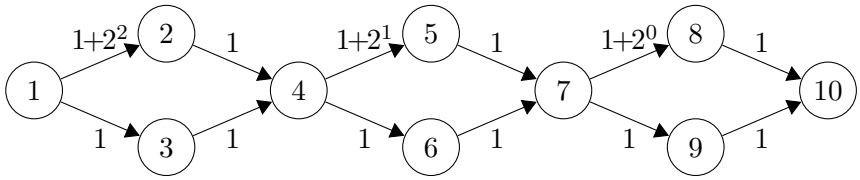


Abbildung 9.2.: Beispielgraph für $n = 10$.

Ergebnis für den Erwartungswert folgt aus dem ersten Resultat, das für beliebige initiale Pheromone gilt. Wenn der Algorithmus nicht alle kürzesten Pfade innerhalb der ersten $T + 3 \cdot \ell/\rho$ Schritte findet, dann wiederholen wir die Argumentation mit einer anderen Phase derselben Länge. Die erwartete Anzahl von benötigten Phasen beträgt offensichtlich $\mathcal{O}(1)$. □

Wir weisen darauf hin, dass die Bedingung, dass alle kürzesten Pfade eindeutig sein müssen, aufgegeben werden kann, wenn wir bereit sind, den zusätzlichen Faktor $\log n$ in dem ersten Term der Optimierungszeitschranke zu akzeptieren. In diesem Fall können wir wie folgt argumentieren. Partitioniere alle Knoten in Schichten, wobei Schicht i die Knoten enthält, deren maximale Anzahl von Kanten auf allen kürzesten Pfade nach n genau i beträgt. Der Vorteil dieser Schichteneinteilung besteht darin, dass sobald alle Knoten in den Schichten $i - 1, \ldots, 1$ optimiert und fast abgearbeitet sind, alle Knoten in Schicht i mit Wahrscheinlichkeit mindestens $\tau_{\min}/(4 \cdot e)$ optimiert werden können. Die erwartete Zeit, bis der letzte Knoten in Schicht i optimiert ist, kann mithilfe von Standardargumenten durch $\mathcal{O}(\log(n)/\tau_{\min})$ begrenzt werden. (Im Wesentlichen zeigt man, dass ein fester Knoten in Schicht i nach $(4 \cdot e \cdot \ln(2 \cdot n))/\tau_{\min}$ Iterationen höchstens mit Wahrscheinlichkeit $1/(2 \cdot n)$ *nicht* optimiert ist und folgert dann mithilfe der booleschen Ungleichung, dass alle Knoten in Schicht i mindestens mit Wahrscheinlichkeit $1/2$ optimiert sind.) Wenn wir diese erwarteten Wartezeiten sowie für jede Schicht die Wartezeit $3/\rho$, die die benötigten Zeit, eine Schicht fast abzuarbeiten, widerspiegelt, aufsummieren, dann erhalten wir die Schranke $\mathcal{O}(\ell \cdot \log n/\tau_{\min} + \ell/\rho)$.

Alle oberen Schranken für die Optimierungszeit, die wir bislang gesehen haben, wachsen mit $\ell(G)$. Unsere Argumente basieren darauf, die benötigte Zeit, um die Knoten $v_0, \ldots, v_\ell$ einen nach dem anderen abzuarbeiten, zu begrenzen. Es stellt sich die naheliegende Frage, ob dies immer notwendig ist. Betrachte den in Abbildung 9.2 skizzierten Beispielgraphen. Betrachte darin insbesondere die Knoten, die über zwei ausgehende Kanten verfügen, die wir untere Kante und obere Kante nennen. Jeder Pfad, der mit einer unteren Kante anfängt, ist kürzer als jeder Pfad, der mit der zugehörigen oberen Kante anfängt. Diese Beobachtung impliziert, dass nachdem eine Ameise die untere Kante genommen hat, zukünftig ausschließlich die Pheromonspur auf der unteren Kante intensiviert wird. Die erwartete Zeit, bis dies passiert, ist durch $\mathcal{O}(1/\tau_{\min})$ Iterationen beschränkt. Demzufolge weist jeder Knoten nach $\mathcal{O}((\log n)/\tau_{\min} + \log(1/\tau_{\min})/\rho)$ Iterationen mit hoher Wahrscheinlichkeit den Pheromonwert $\tau_{\min}$ auf seiner oberen Kante auf. In dieser Situation beträgt die

Wahrscheinlichkeit, einen kürzesten Pfad zu finden, für jede Ameise mindestens $1/e$. Also müssen wir einen Term der Größenordnung $\mathcal{O}(\log n)$ zu der erwarteten Zeit, bis der letzte Knoten optimiert ist, hinzuaddieren. Die Optimierungszeit des Ameisensystems $\text{MMAS}_{\text{SDSP}}$ angewandt auf den Beispielgraphen beträgt folglich mit hoher Wahrscheinlichkeit $\mathcal{O}((\log n)/\tau_{\min} + \log(1/\tau_{\min})/\rho)$. Beachte, dass in diesem Fall die obere Schranke $\mathcal{O}(n/\tau_{\min} + n/\rho)$ aus Theorem 9.8 zu pessimistisch ist. Im Allgemeinen ist die Schranke jedoch scharf, wie wir im Folgenden zeigen werden.

9.3.2. Untere Schranken

Wir wenden uns nun unteren Schranken für die erwartete Optimierungszeit des Algorithmus $\text{MMAS}_{\text{SDSP}}$ zu. Der zweite Term in den oberen Schranken, die wir bislang kennengelernt haben, wächst mit $1/\rho$. Es liegt nahe, sich zu fragen, ob diese Abhängigkeit von ρ tatsächlich notwendig ist. Das folgende Ergebnis beantwortet diese Frage positiv, sofern ρ nicht so klein ist, dass das Verhalten des Ameisensystems einer zufälligen Suche gleicht. In diesem Fall erhalten wir eine untere Schranke, die fast die Größenordnung $\exp(\Omega(\ell))$ aufweist.

Die Hauptidee für den Beweis des folgenden Theorems besteht darin, dass die Pheromone eine gewisse Zeit benötigen, um sich anzupassen, sodass ein kürzester Pfad mit ℓ Kanten mit einer nicht zu kleinen Wahrscheinlichkeit gefunden werden kann.

Theorem 9.9. *Betrachte einen gerichteten Graphen G mit n Knoten und positiven Kantengewichten. Seien $\Delta := \Delta(G)$ und $\ell := \ell(G)$. Wir nehmen an, dass G einen eindeutigen kürzesten Pfad $p_0, \ldots, p_\ell = n$ enthält, sodass für alle $0 \leq i \leq \ell - \mathcal{O}(1)$ gilt, dass von dem Knoten p_i mindestens zwei Kanten ausgehen und dass alle ausgehenden Kanten zu einem Knoten aus $V \setminus \{p_0, \ldots, p_i\}$ führen. Wenn $\tau_{\min} \leq 1/(\Delta \cdot \ell)$, $\tau_{\max} = 1 - \tau_{\min}$ und $0 < \rho \leq 1 - \Omega(1)$ gelten, dann beträgt die erwartete Optimierungszeit des Algorithmus $MMAS_{APSP}$ angewandt auf G mindestens $\Omega(\min\{(\log \ell)/\rho, \exp(\ell^{1-\epsilon}/4)\})$ für alle Konstanten $\epsilon > 0$.*

Beweis. Betrachte die Ameise, die am Knoten p_0 startet und versucht, den Pfad $p_0, \ldots, p_\ell$ zu finden. Gemäß der Annahme für G gilt, dass die Ameise während sie diesem Pfad folgt, an jedem Knoten $0 \leq i \leq \ell - \mathcal{O}(1)$ die Wahl zwischen mindestens zwei Kanten hat, die zu unbesuchten Knoten führen. Sei $c > 1$ eine Konstante, die wir später in Abhängigkeit von ϵ festlegen werden. Während der ersten $t := \min\{(1/\rho - 1) \cdot \epsilon \cdot \ln(\ell), \exp(\ell^{1-\epsilon}/4)/c\} = \Omega(\min\{(\log \ell)/\rho, \exp(\ell^{1-\epsilon}/4)\})$ Iterationen – hier benutzen wir $1/\rho - 1 = \Omega(1/\rho)$, das aufgrund der Annahme an ρ gilt – beträgt der Pheromonwert auf jeder solchen Kante mindestens

$$\frac{1}{\deg(u)} \cdot (1-\rho)^t \geq \frac{1}{\deg(u)} \cdot e^{-\epsilon \cdot \ln(\ell)} = \frac{1}{\deg(u)} \cdot \frac{1}{\ell^\epsilon}.$$

Beachte, dass dies sogar dann gilt, wenn die untere Schranke für die Pheromonwerte erreicht ist. Da die Wahrscheinlichkeit, eine bestimmte inkorrekte Kante zu wählen,

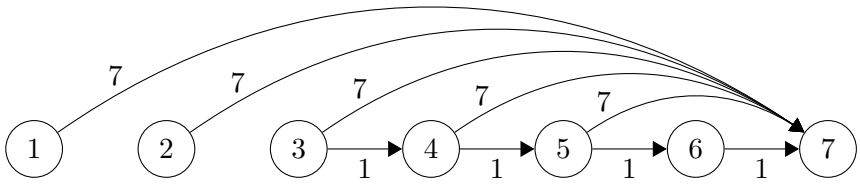


Abbildung 9.3.: Beispielgraph $G^{\text{lb}}_{n,\ell}$ aus Definition 9.10 für $n = 7$ und $\ell = 4$.

mindestens $p := 1/(2 \cdot \deg(u) \cdot \ell^\epsilon)$ beträgt, beträgt die Wahrscheinlichkeit, dass die Ameise eine korrekte Kante auf dem Pfad wählt, höchstens

$$1 - (\deg(u) - 1) \cdot p = 1 - (\deg(u) - 1) \cdot \frac{1}{2 \cdot \deg(u) \cdot \ell^\epsilon} \leq 1 - \frac{1}{4 \cdot \ell^\epsilon}.$$

Demzufolge ist die Wahrscheinlichkeit, dass der Pfad $p_0, \ldots, p_\ell$ in einer bestimmten Iteration $t' \leq t$ konstruiert wird, durch $(1 - 1/(4 \cdot \ell^\epsilon))^{\ell - \mathcal{O}(1)} = \mathcal{O}(\exp(-\ell^{1-\epsilon}/4))$ beschränkt. Die Wahrscheinlichkeit, dass dies während der ersten t Iterationen geschieht, ist durch $t \cdot \mathcal{O}(\exp(-\ell^{1-\epsilon}/4)) \leq 1/2$ beschränkt, wenn wir die Konstante c in der Definition von t geeignet festlegen. Wie die Konstante festzulegen ist, werden wir uns im Anschluss anschauen. Also gilt mit Wahrscheinlichkeit mindestens $1/2$, dass nach t Schritten noch nicht alle kürzesten Pfade gefunden sind, und die untere Schranke $t/2 = \Omega(\min\{(\log \ell)/\rho, \exp(\ell^{1-\epsilon}/4)\})$ für die erwartete Optimierungszeit folgt.

Wir müssen die Konstante $c > 0$ so wählen, dass

$$t \cdot \left(1 - \frac{1}{4 \cdot \ell^\epsilon}\right)^{\ell - c'} \leq \frac{1}{c} \cdot e^{\ell^{1-\epsilon}/4} \cdot e^{-\ell^{1-\epsilon}/4} \cdot e^{c'/(4 \cdot \ell^\epsilon)} = \frac{e^{c'/(4 \cdot \ell^\epsilon)}}{c} \overset{!}{\leq} \frac{1}{2}$$

gilt, wobei $i \leq \ell - c'$ für eine Konstante $c' > 0$ ist. Wie die Rechnung zeigt, eignet sich beispielsweise $c := 2 \cdot e^{c'/4}$. □

Um beurteilen zu können, ob die obere Schranke aus Theorem 9.8 asymptotisch scharf ist, betrachten wir die folgende Klasse von Eingabeinstanzen (siehe Abbildung 9.3). Die grundlegende Idee ist, dass das Ameisensystem gezwungen ist, die Knoten einen nach dem anderen, von rechts nach links zu optimieren.

Definition 9.10. *Sei* $G^{\text{lb}}_{n,\ell} = (V, E, w)$, $n \in \mathbb{N}$, $1 \leq \ell < n$, *mit* $V = \{1, \ldots, n\}$, $E = \{(i, i+1) \mid n - \ell \leq i \leq n-1\} \cup \{(i, n) \mid 1 \leq i \leq n-2\}$ *und Kantengewichten* $w((u,v)) = 1$ *wenn* $v = u + 1$ *und* $w((u,v)) = n$ *wenn* $v \neq u + 1$.

Theorem 9.8 liefert eine obere Schranke aus $\mathcal{O}(\ell/\tau_{\min} + \ell/\rho)$ für $G^{\text{lb}}_{n,\ell}$ wenn $\ell = \Omega(\log n)$ und $\tau_{\min} \leq 1/(\Delta(G^{\text{lb}}_{n,\ell}) \cdot \ell)$ gelten. Die folgende untere Schranke stimmt asymptotisch mit der oberen Schranke überein, wenn $\rho = \Omega(\tau_{\min})$ ist. Für kleinere ρ manifestiert sich eine Lücke, die jedoch durch $O(\log n)$ begrenzt ist.

Theorem 9.11. *Seien* $\Delta = \Delta(G^{\text{lb}}_{n,\ell})$ *und* $\ell = \ell(G^{\text{lb}}_{n,\ell}) \geq c \cdot \log^2 n$ *für eine hinreichend große Konstante* c*. Wenn* $1/\operatorname{poly}(n) \leq \tau_{\min} \leq 1/(\Delta \cdot \ell)$*,* $\tau_{\max} = 1 - \tau_{\min}$ *und* $1/\operatorname{poly}(n) \leq \rho \leq 1/2$ *gelten, dann beträgt die erwartete Optimierungszeit des Algorithmus* $MMAS_{APSP}$ *angewandt auf* $G^{\text{lb}}_{n,\ell}$ *mindestens* $\Omega(\ell/\tau_{\min} + \ell/(\rho \cdot \log(1/\rho)))$.

Beweis. Wir ignorieren die Knoten $1, \ldots, n - \ell - 1$, da die Konstruktion eines kürzesten Pfades für diese Knoten trivial ist. Betrachte alle Pfade von u nach n mit $n - \ell \leq u \leq n - 2$. Der Pfad (u, n) weist die Länge n auf. Alle anderen Pfade beginnen mit der Kante $(u, u + 1)$. Die Länge des Pfades, der nur die Kanten mit Gewicht 1 umfasst, beträgt $n - u$. Wenn der Pfad jedoch mit einer Kante (v, n) für $u < v \leq n - 2$ endet, dann beträgt die Länge $v - u + n > n$. Also ist der Pfad (u, n) der eindeutige zweitkürzeste Pfad von u nach n.

Wir nennen einen Knoten $n - \ell \leq u \leq n - 2$ *falsch*, wenn der von der Ameise a_u beste bisher gefundene Pfad (u, n) entspricht. Nach der Initialisierung werden beide Kanten mit der gleichen Wahrscheinlichkeit von der ersten Ameise gewählt. Aufgrund der Chernoff-Ungleichung wählen mindestens $\ell/3$ Ameisen a_u mit $n - \ell \leq u \leq n - 2$ sofort die inkorrekte Kante mit Wahrscheinlichkeit $1 - e^{-\Omega(\ell)}$ (siehe Lemma A.6). In einem solchen Fall bleibt der zugehörige Knoten falsch bis der kürzeste Pfad gefunden wird. Im Folgenden nehmen wir an, dass anfangs mindestens $\ell/3$ Knoten falsch sind.

Intuitiv verringert sich die Wahrscheinlichkeit, einen Knoten zu optimieren, mit der Anzahl von falschen Knoten auf seinem kürzesten Pfad. Das folgende Lemma liefert eine untere Schranke für die Zeit bis ein falscher Knoten optimiert wird, wenn sich auf seinem kürzesten Pfad mehr als $8 \cdot \log(1/\rho)$ falsche Knoten befinden.

Lemma 9.12. *Sei* u *ein falscher Knoten, sodass der kürzeste Pfad von* u *nach* n $8 \cdot \log(1/\rho) + i$ *falsche Knoten für ein* $i \in \mathbb{N}$ *enthält. Dann beträgt die Wahrscheinlichkeit,* u *innerhalb der nächsten* $1/\rho - 1$ *Iterationen zu optimieren, höchstens* $(1 - 1/(4 \cdot e))^i$.

Beweis. Sei F die Menge, die alle falschen Knoten auf dem kürzesten Pfad von u nach n enthält. Solange ein Knoten falsch ist, beträgt der Pheromonwert auf seiner inkorrekten Kante mindestens $1/2$. (Er strebt sogar kontinuierlich gegen $\tau_{\max}$ bis der kürzeste Pfad gefunden wird.) Das bedeutet für jeden Knoten aus F, dass die gegenwärtige Wahrscheinlichkeit, die inkorrekte Kante zu wählen, gemäß Korollar 9.2 mindestens $1/4$ beträgt. Sobald ein Knoten aus F optimiert ist, verringert sich die Wahrscheinlichkeit in jeder folgenden Iteration um den Faktor $(1 - \rho)$. Sei $t := 1/\rho - 1$. Wenn wir den Faktor $(1 - \rho)^t$ als grobe Schätzung verwenden, können wir schließen, dass während der nächsten t Iterationen die Wahrscheinlichkeit, die inkorrekte Kante zu wählen, stets mindestens $1/4 \cdot (1 - \rho)^t \geq 1/(4 \cdot e)$ beträgt. Das bedeutet, dass die Wahrscheinlichkeit, den kürzesten Pfad von u nach n zu finden, höchstens $(1 - 1/(4 \cdot e))^{8 \cdot \log(1/\rho) + i}$ ist. Wir wenden die boolesche Ungleichung auf t Iterationen an und erhalten, dass die Wahrscheinlichkeit, dass u innerhalb der

nächsten t Iterationen optimiert wird, höchstens

$$\begin{aligned} t \cdot \left(1 - \frac{1}{4 \cdot e}\right)^{8 \cdot \log(1/\rho) + i} &\leq 2^{\log(1/\rho)} \cdot \left(1 - \frac{1}{4 \cdot e}\right)^{8 \cdot \log(1/\rho) + i} \\ &\leq 2^{\log(1/\rho)} \cdot e^{-(2/e) \cdot \log(1/\rho)} \cdot \left(1 - \frac{1}{4 \cdot e}\right)^{i} \\ &\leq \left(1 - \frac{1}{4 \cdot e}\right)^{i} \end{aligned}$$

beträgt. □

Das nächste Lemma stellt eine unmittelbare Folgerung dar.

Lemma 9.13. *Die erwartete Anzahl von falschen Knoten, die innerhalb von* $1/\rho - 1$ *Iterationen optimiert werden, ist durch* $8 \cdot \log(1/\rho) + 4 \cdot e$ *beschränkt.*

Beweis. Sei F die Menge, die alle falschen Knoten enthält, und bezeichne p_u die Wahrscheinlichkeit, dass ein falscher Knoten u innerhalb von $1/\rho - 1$ Iterationen optimiert wird. Wenn u höchstens $8 \cdot \log(1/\rho)$ falsche Knoten auf seinem kürzesten Pfad aufweist, dann schätzen wir $p_u \leq 1$. Anderenfalls enthält der kürzeste Pfad $8 \cdot \log(1/\rho) + i$ falsche Knoten und Lemma 9.12 liefert $p_u \leq (1 - 1/(4 \cdot e))^i$. Die erwartete Anzahl von optimierten Knoten ist folglich durch

$$\sum_{u \in F} p_u \leq 8 \cdot \log(1/\rho) + \sum_{i=0}^{\infty} \left(1 - \frac{1}{4 \cdot e}\right)^{i} = 8 \cdot \log(1/\rho) + 4 \cdot e$$

beschränkt. □

Wir erinnern daran, dass wir angenommen haben, dass zu Beginn mindestens $\ell/3$ Knoten falsch sind. Nun betrachten wir $\ell/6 \cdot 1/(8 \cdot \log(1/\rho) + 4 \cdot e)$ aufeinander folgende Phasen, die jeweils $1/\rho - 1$ Iterationen umfassen. Wir weisen darauf hin, dass die Phasen insgesamt $\Omega(\ell/(\rho \cdot \log(1/\rho)))$ Iterationen umfassen. Lemma 9.13 impliziert, dass die erwartete Anzahl von Knoten, die in allen Phasen optimiert werden, durch $\ell/6$ beschränkt ist. Wir wenden die Markoff-Ungleichung an und schließen, dass die Wahrscheinlichkeit, dass $\ell/3$ korrigiert werden, höchstens $1/2$ beträgt. Wir berücksichtigen die Wahrscheinlichkeit, dass es am Anfang nicht mindestens $\ell/3$ falsche Knoten gibt, und erhalten, dass das Ameisensystem mit Wahrscheinlichkeit mindestens $1/2 - e^{-\Omega(\ell)} = \Omega(1)$ mindestens $\Omega(\ell/(\rho \log(1/\rho)))$ Iterationen benötigt. Dies zeigt die untere Schranke $\Omega(\ell/(\rho \cdot \log(1/\rho)))$ für die erwartete Optimierungszeit.

Nun müssen wir nur noch eine untere Schranke aus $\Omega(\ell/\tau_{\min})$ herleiten, um den Beweis des Theorems zu komplettieren. Aufgrund von Lemma 9.13 werden in $F := \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen im Erwartungswert nicht mehr als $\mathcal{O}(\ln(\tau_{\max}/\tau_{\min}) \cdot \log(1/\rho)) = \mathcal{O}(\log^2 n)$ falsche Knoten optimiert, wobei hier die Annahmen an $\tau_{\min}$ und ρ eingehen. Mit Wahrscheinlichkeit mindestens $1/2$ beträgt diese zufällige Anzahl höchstens $\mathcal{O}(\log^2 n)$. Wenn wir annehmen, dass dies eingetreten ist, beträgt die

Anzahl der nach F Iterationen verbliebenen falschen Knoten $\ell/3 - \mathcal{O}(\log^2 n) = \Omega(\ell)$, wenn die Konstante c in der Voraussetzung $\ell \geq c \cdot \log^2 n$ hinreichend groß gewählt wird. Für diese falschen Knoten sind die Pheromone auf den Kanten eingefroren, wobei die Pheromonwerte auf der korrekten und der inkorrekten Kante nun $\tau_{\min}$ bzw. $\tau_{\max}$ betragen.

Jetzt beträgt die Wahrscheinlichkeit, einen Knoten v zu optimieren, auf dessen kürzestem Pfad sich mehr als 3 falsche Knoten befinden, höchstens $(\tau_{\min})^4 \leq \tau_{\min}/\ell^3$. Betrachte die $\mathcal{O}(\ell)$ Ameisen $a_{n-\ell}, \ldots, a_{n-2}$ und $\mathcal{O}(\ell/\tau_{\min})$ Iterationen. Dann schließen wir mithilfe der booleschen Ungleichung, dass mit Wahrscheinlichkeit mindestens $1 - \mathcal{O}(1/\ell)$ in einer Phase aus $\mathcal{O}(\ell/\tau_{\min})$ Iterationen höchstens 3 falsche Knoten pro Iteration korrigiert werden. Die Wahrscheinlichkeit, einen falschen Knoten mit $i-1$ falschen Nachfolgern auf seinem kürzesten Pfad zu korrigieren, beträgt höchstens $(\tau_{\min})^i$. Also ist die Wahrscheinlichkeit, einen falschen Knoten zu korrigieren, höchstens $\sum_{i=1}^{\infty} \tau_{\min}^i = \mathcal{O}(\tau_{\min})$ und die erwartete Zeit, bis ein falscher Knoten korrigiert wird, ist mindestens $\Omega(1/\tau_{\min})$. Wenn sich die Anzahl von falschen Knoten stets nur um höchstens 3 verringert, müssen $\Omega(\ell)/3 = \Omega(\ell)$ dieser Ereignisse eintreten, um alle kürzesten Pfade zu finden. Insgesamt beträgt die erwartete Zeit in dieser Situation damit mindestens $\Omega(\ell/\tau_{\min})$. Da das Ameisensystem mit Wahrscheinlichkeit mindestens $1/2 - \mathcal{O}(1/\ell) - e^{-\Omega(\ell)} = \Omega(1)$ in die beschriebene Situation gerät, erhalten wir eine unbedingte Schranke der gleichen Größenordnung. □

Beachte, dass die untere Schranke nur für $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ gilt und dass der erste Term der unteren Schranke mit $1/\tau_{\min}$ wächst. Die Leserinnen und Leser könnten versucht sein, $\tau_{\min} > 1/(\Delta \cdot \ell)$ zu wählen, um eine erwartete Optimierungszeit aus $o(\Delta \cdot \ell^2)$ zu erreichen. Wenn $\tau_{\min}$ jedoch zu groß gewählt wird, können sich die Pheromone nicht richtig anpassen und die Suche ist zu nah an einer zufälligen Suche. Betrachte beispielsweise den Knoten $u = n - \ell$ aus $G_{n,\ell}^{\text{lb}}$, dessen kürzester Pfad ℓ Kanten enthält. Sogar wenn alle Pheromone so gut wie möglich angepasst sind, beträgt die Wahrscheinlichkeit, die korrekte Kante an einem Knoten $v \leq n-2$ zu wählen, dennoch höchstens $1 - \tau_{\min}/2$. Die Wahrscheinlichkeit, $(\ell-1)$-mal die richtige Entscheidung zu treffen, beträgt nur $(1-\tau_{\min}/2)^{\ell-1} \approx e^{-\tau_{\min} \cdot \ell/2}$. Der Kehrwert dieser Wahrscheinlichkeit stellt eine untere Schranke für die erwartete Optimierungszeit dar; sie ist superpolynomiell falls $\tau_{\min} = \omega((\log n)/\ell)$ gilt. Diese Beobachtung rechtfertigt auch, warum unsere bisherigen oberen Schranken nur in dem Fall $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ greifen.

9.3.3. Eine adaptive Wahl der Pheromonschranken

Alle Schranken für die Optimierungszeit, die wir bislang kennengelernt haben, setzen voraus, dass $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ für die untere Pheromonschranke $\tau_{\min}$ gilt. Zudem haben wir im letzten Abschnitt gezeigt, dass eine wesentlich größere untere Pheromonschranke im Allgemeinen nicht sinnvoll ist. Graphen, in denen nur wenige Knoten einen hohen Grad besitzen, könnten eine Ausnahme von dieser Regel darstellen. Wir erinnern uns, dass die Wahrscheinlichkeit, einen kürzesten Pfad von u nach n zu kon-

struieren, gegeben, dass alle Nachfolger von u auf kürzesten Pfaden fast abgearbeitet sind, durch $\tau_{\min}/(4 \cdot e)$ nach unten beschränkt ist, wenn $\tau_{\min} \leq 1/(\deg(u) \cdot \ell)$ gilt (siehe Lemma 9.6). Wenn die gleichen Pheromonschranken für alle Kanten gelten, dann ist $\tau_{\min} = 1/(\Delta \cdot \ell)$ die beste Wahl für $\tau_{\min}$, die diese Anforderung für alle Knoten erfüllt.

Man könnte jedoch auch Ameisensysteme betrachten, in denen die Pheromonschranken für jeden Knoten individuell eingestellt werden können. Der Pheromonwert auf einer Kante $e = (u, \cdot)$ wird dann durch die Pheromonschranken $\tau_{\min}(u)$ und $\tau_{\max}(u)$ begrenzt. Wenn $\tau_{\min}(u) = 1/(\deg(u) \cdot \ell)$ und $\tau_{\max}(u) = 1 - \tau_{\min}(u)$ gelten, dann ist die erwartete Wartezeit, bis u optimiert wird, gegeben, dass alle Nachfolger auf kürzesten Pfaden fast abgearbeitet sind, durch $4 \cdot e/\tau_{\min}(u) = 4 \cdot e \cdot \deg(u) \cdot \ell$ beschränkt. Das Aufsummieren dieser Wartezeiten für alle Knoten und das Hinzufügen der Wartezeiten, bis die Knoten fast abgearbeitet werden, ergibt die folgende Schranke.

Theorem 9.14. *Betrachte einen gerichteten Graphen G mit n Knoten, m Kanten und positiven Kantengewichten. Sei $\ell := \ell(G)$. Wenn $\tau_{\min}(u) = 1/(\deg(u) \cdot \ell)$ und $\tau_{\max}(u) = 1 - \tau_{\min}(u)$ für alle Knoten u gelten, dann beträgt die erwartete Optimierungszeit des Algorithmus $MMAS_{SDSP}$ mit adaptiven Pheromonschranken angewandt auf G höchstens $\mathcal{O}(\ell \cdot m + n/\rho)$.*

9.4. Das Kürzeste-Wege-Problem für alle Paare

Wir konstruieren nun einen Algorithmus $\text{MMAS}_{\text{APSP}}$ für das Kürzeste-Wege-Problem für alle Paare, wobei wir den Algorithmus $\text{MMAS}_{\text{SDSP}}$ geeignet erweitern. Dazu führen wir für jede Senke $v \in V$ eine eigene Pheromonfunktion $\tau_v \colon E \to \mathbb{R}_0^+$ ein. In jeder Iteration simulieren wir für jede Quelle u und für jede Senke v eine Ameise $a_{u,v}$, die auf dem Knoten u startet und nach dem Knoten v sucht. Eine Ameise, die nach v sucht, orientiert sich anhand der Pheromonfunktion τ_v und aktualisiert diese wie in Abschnitt 9.2 beschrieben. Weiterhin speichert der Algorithmus $\text{MMAS}_{\text{APSP}}$ die bislang besten Pfade $p^*_{u,v}$ von u nach v für alle $u, v \in V$.

Das folgende Ergebnis folgt unmittelbar aus Theorem 9.8.

Theorem 9.15. *Betrachte einen gerichteten Graphen G mit n Knoten und positiven Kantengewichten, in dem alle kürzesten Pfade eindeutig sind. Seien $\Delta := \Delta(G)$, $\ell := \ell(G)$ und $\ell^* := \max\{\ell, \ln(n)\}$. Wenn $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ und $\tau_{\max} = 1 - \tau_{\min}$ gelten, beträgt die Optimierungszeit des Algorithmus $MMAS_{APSP}$ angewandt auf G mit Wahrscheinlichkeit mindestens $1 - 1/n$ höchstens $\mathcal{O}(\ell^*/\tau_{\min} + \ell/\rho)$. Die Schranke vereinfacht sich für $\tau_{\min} = 1/n^2$ zu $O(\ell^* \cdot n^2 + \ell/\rho)$ und für $\tau_{\min} = 1/(\Delta \cdot \ell)$ zu $O(\Delta \cdot \ell \cdot \ell^* + \ell/\rho)$. Die Schranke für die Optimierungszeit gilt auch im Erwartungswert.*

Beweis. Halte eine Senke v fest. Kürzeste Wege von allen Quellen u zu der Senke v werden aufgrund von Theorem 9.8 innerhalb von $O(\ell^*/\tau_{\min} + \ell/\rho)$ Iterationen mindestens mit Wahrscheinlichkeit $1 - 1/n^2$ gefunden. Wir wenden nun die boolesche Ungleichung (siehe Lemma A.4) an und schließen, dass mindestens mit Wahr-

Algorithmus 10 Pfadkonstruktion von u nach v für $\text{MMAS}_{\text{APSP}}$ mit Interaktion

```
wähle p ∈ [0, 1) rein zufällig
if p < p_int then
    wähle w ∈ V rein zufällig
    konstruiere einfachen Pfad p' = (p'_0, ..., p'_ℓ') von u nach w bezüglich τ_w
    konstruiere einfachen Pfad p'' = (p''_0, ..., p''_ℓ'') von w nach v bezüglich τ_v
    if p'_ℓ' = w then p ← (p'_0, ..., p'_ℓ', p''_1, ..., p''_ℓ'') else p ← p' end if
else
    konstruiere einfachen Pfad p von u nach v bezüglich τ_v
end if
return p
```

scheinlichkeit $1 - 1/n$ zwischen jedem Paar von Knoten ein kürzester Pfad gefunden wird. □

Wir weisen darauf hin, dass Ameisen, die nach unterschiedlichen Senken suchen, in dem beschriebenen Ameisensystem nicht interagieren. Eine Ameise, die nach der Senke v sucht, orientiert sich einzig an der Pheromonfunktion τ_v. Folglich könnten wir auch n Instanzen des Ameisensystems $\text{MMAS}_{\text{SDSP}}$ parallel laufen lassen, um das gleiche Resultat zu erreichen. Es stellt sich die naheliegende Frage, ob die Ameisen durch geeignete Interaktionsmechanismen dazu in die Lage versetzt werden, das Problem schneller lösen zu können.

Der folgende einfache Mechanismus wird sich interessanterweise als hilfreich erweisen. Betrachte die Ameise $a_{u,v}$, die nach der Senke v sucht. Anstatt stets der Pheromonspur τ_v zu folgen, um nach v zu gelangen, entscheidet sich die Ameise mit einer gewissen Wahrscheinlichkeit p_{int} zeitweise einer fremden Pheromonspur zu folgen. Sie wählt in diesem Fall zunächst rein zufällig eine Zwischenstation w, benutzt dann die Pheromonfunktion τ_w, um nach w zu gelangen, und verwendet anschließend die Pheromonfunktion τ_v, um das eigentliche Ziel v zu erreichen (siehe Algorithmus 10). Die Ameise $a_{u,v}$ ist jedoch nach wie vor ausschließlich für die Aktualisierung der Pheromonfunktion τ_v zuständig.

Der vorgeschlagene Interaktionsmechanismus ermöglicht der Ameise $a_{u,v}$ von hilfreichen Informationen zu profitieren, die von anderen Ameisen gesammelt wurden, die nach einem Knoten w suchen. Das gilt besonders, wenn w ein Knoten auf einem kürzesten Weg von u nach v ist. Das folgende Theorem liefert eine deutlich verbesserte obere Schranke für die Optimierungszeit und gilt zudem auch für Graphen, in denen die kürzesten Wege nicht eindeutig bestimmt sind.

Theorem 9.16. *Betrachte einen gerichteten Graphen G mit n Knoten und positiven Kantengewichten. Seien $\Delta := \Delta(G)$ und $\ell := \ell(G)$. Wenn $1/n^3 \leq \tau_{\min} \leq 1/(\Delta \cdot \ell)$, $\tau_{\max} = 1 - \tau_{\min}$ und $\rho \leq (1 - p_{\text{int}})/(12 \cdot \Delta \cdot \log n)$ gelten, beträgt die Optimierungszeit des Algorithmus $MMAS_{APSP}$ mit Interaktion (mit einer konstanter Wahrscheinlichkeit $0 < p_{\text{int}} < 1$) angewandt auf G mit Wahrscheinlichkeit mindestens $1 - 1/n^2$ höchstens $\mathcal{O}(n \cdot \log(n) + \log(\ell) \cdot \log(1/\tau_{\min})/\rho)$. Die Schranke vereinfacht*

sich für $\tau_{\min} = 1/n^2$ *zu* $O(n \cdot \log(n) + \log(\ell) \cdot \log(n)/\rho)$ *und für* $\tau_{\min} = 1/(\Delta \cdot \ell)$ *zu* $O(n \cdot \log(n) + \log(\ell) \cdot \log(\Delta \cdot \ell)/\rho)$. *Die Schranke für die Optimierungszeit gilt auch im Erwartungswert.*

Beweis. Wir führen ähnliche Bezeichnungen wie zuvor ein. Betrachte ein Paar (u, v) von Knoten. Sei

$$\ell_{u,v} := \max\{\#\text{Kanten von } p \mid p \text{ ist ein kürzester Pfad von } u \text{ nach } v\}.$$

Wir nennen eine Kante *inkorrekt* bezüglich v, wenn sie nicht zu einem beliebigen kürzesten Pfad nach v gehört. Wir nennen ein Paar (u, v) *optimiert*, wenn ein kürzester Pfad von u nach v gefunden wurde. Wir nennen ein Paar (u, v) *abgearbeitet*, wenn es optimiert wurde und wenn die Pheromonwerte $\tau_v(\cdot)$ auf allen Kanten $(u, \cdot)$, die inkorrekt bezüglich v sind, $\tau_{\min}$ betragen.

Betrachte die ersten $t = \lceil \ln(2)/\rho \rceil = \mathcal{O}(1/\rho)$ Iterationen. Betrachte ein Paar (u, v) mit $\ell_{u,v} = 1$. Die Wahrscheinlichkeit, (u, v) in Iteration i zu optimieren, beträgt mindestens $(1 - p_{\text{int}}) \cdot (1 - \rho)^{i-1}/(2 \cdot \Delta)$, da die Ameise $a_{u,v}$ sich mit Wahrscheinlichkeit $1 - p_{\text{int}}$ entscheidet, nach v zu suchen, und aufgrund von Korollar 9.2 (u, v) mindestens mit Wahrscheinlichkeit $(1 - \rho)^{i-1}/(2 \cdot \Delta)$ wählt. Folglich beträgt die Wahrscheinlichkeit, (u, v) in der betrachteten Phase *nicht* zu optimieren, höchstens

$$\begin{aligned}\prod_{i=1}^{t} \left(1 - \frac{(1 - p_{\text{int}}) \cdot (1 - \rho)^{i-1}}{2 \cdot \Delta}\right) &\leq \exp\left(-\frac{1 - p_{\text{int}}}{2 \cdot \Delta} \cdot \sum_{i=0}^{t-1} (1 - \rho)^i\right) \\ &= \exp\left(-\frac{1 - p_{\text{int}}}{2 \cdot \Delta} \cdot \frac{1 - (1 - \rho)^t}{1 - (1 - \rho)}\right) \\ &\leq \exp\left(-\frac{1 - p_{\text{int}}}{4 \cdot \Delta \cdot \rho}\right),\end{aligned}$$

wobei die erste Ungleichung aus Korollar A.19 folgt. Indem wir ausnutzen, dass $\rho \leq (1 - p_{\text{int}})/(12 \cdot \Delta \cdot \log(n)) \leq (1 - p_{\text{int}})/(4 \cdot \Delta \cdot \ln(2 \cdot n^4))$ gilt, schließen wir, dass die obige Wahrscheinlichkeit durch $1/(2 \cdot n^4)$ nach oben begrenzt ist. Aufgrund der booleschen Ungleichung (siehe Lemma A.4), werden alle Paare (u, v) mit $\ell_{u,v} = 1$ in der betrachteten Phase mit Wahrscheinlichkeit mindestens $1 - f_1$ optimiert, wobei $f_1 := 1/(2 \cdot n^2)$ ist. Wir wissen bereits, dass ein optimiertes Paar (u, v) nach höchstens $\lceil \ln(\tau_{\max}/\tau_{\min})/\rho \rceil$ Iterationen abgearbeitet ist.

Halte ein Paar (u, v) von Knoten fest und betrachte einen kürzesten Pfad $p_{u,v} = (v_0, \ldots, v_{\ell_{u,v}})$ von u nach v mit $\ell_{u,v}$ Kanten. Sei i mit $(3/2)^i < \ell_{u,v} \leq (3/2)^{i+1}$. Wir leiten eine untere Schranke für die Wahrscheinlichkeit, (u, v) zu optimieren, her, wobei wir annehmen, dass alle Paare (u', v') mit $\ell_{u',v'} \leq (3/2)^i$ abgearbeitet sind. Die Ameise entscheidet sich mit Wahrscheinlichkeit p_{int}, einen Zwischenknoten w aufzusuchen. Dieser befindet sich auf dem mittleren Drittel $(v_{\ell_{u,v} - \lfloor (3/2)^i \rfloor}, \ldots, v_{0 + \lfloor (3/2)^i \rfloor})$ des Pfades $p_{u,v}$ mit Wahrscheinlichkeit

$$\frac{\lfloor (3/2)^i \rfloor - \ell_{u,v} + \lfloor (3/2)^i \rfloor + 1}{n} = \frac{2 \cdot \lfloor (3/2)^i \rfloor - \ell_{u,v} + 1}{n} \geq \frac{\ell_{u,v}}{3 \cdot n}.$$

Demzufolge beträgt die Anzahl der Kanten von allen kürzesten Pfaden $p_{u,w}$ ($p_{w,v}$) von u (w) nach w (v) höchstens $(3/2)^i$. Weil (x,w) ((x,v)) für alle Knoten x auf einem kürzesten Pfad von u (w) nach w (v) abgearbeitet ist, folgt die Ameise einem kürzesten Pfad von u nach v mindestens mit Wahrscheinlichkeit $(1-1/\ell)^{\ell-1} \geq 1/e$. Insgesamt haben wir gezeigt, dass die Wahrscheinlichkeit, (u,v) zu optimieren, mindestens $p_{\text{int}} \cdot \ell_{u,v}/(3 \cdot n) \cdot 1/e > p_{\text{int}} \cdot (3/2)^i/(3 \cdot e \cdot n)$ beträgt.

Wir gliedern einen Lauf des Ameisensystems in Phasen. Die i-te Phase endet, wenn alle Paare (u,v) mit $(3/2)^{i-1} < \ell_{u,v} \leq (3/2)^i$ abgearbeitet sind. Da $\ell_{u,v} \leq \ell$ gilt, müssen wir $\alpha := \lceil \log(\ell)/\log(3/2) \rceil$ Phasen betrachten.

Betrachte Phase i der Länge $t = \lceil (3 \cdot e \cdot n)/(p_{\text{int}} \cdot (3/2)^i) \cdot \ln(2 \cdot \alpha \cdot n^4) \rceil$. Die Wahrscheinlichkeit, ein Paar (u,v) mit $(3/2)^{i-1} < \ell_{u,v} \leq (3/2)^i$ in der Phase *nicht* zu optimieren, beträgt höchstens $(1-p_{\text{int}} \cdot (3/2)^i/(3 \cdot e \cdot n))^t \leq 1/(2 \cdot \alpha \cdot n^4)$. Aufgrund der booleschen Ungleichung (siehe Lemma A.4), werden alle solche Paare (u,v) in t Iterationen mindestens mit Wahrscheinlichkeit $1 - 1/(2 \cdot \alpha \cdot n^2)$ optimiert. Wir wissen, dass ein optimiertes Paar (u,v) nach höchstens $\lceil \ln(\tau_{\max}/\tau_{\min})/\rho \rceil$ Iterationen abgearbeitet ist. Indem wir die boolesche Ungleichung erneut anwenden, schließen wir, dass alle Phasen nach

$$\begin{aligned}
&\sum_{i=1}^{\alpha} \left(\left\lceil \frac{3 \cdot e \cdot n \cdot \ln(2 \cdot \alpha \cdot n^4)}{p_{\text{int}} \cdot (3/2)^i} \right\rceil + \left\lceil \frac{\ln(\tau_{\max}/\tau_{\min})}{\rho} \right\rceil \right) \\
&\leq \frac{3 \cdot e \cdot n \cdot \ln(2 \cdot \alpha \cdot n^4)}{p_{\text{int}}} \cdot \sum_{i=1}^{\alpha} \left(\frac{2}{3} \right)^i + \alpha \cdot \frac{\ln(1/\tau_{\min})}{\rho} + 2 \cdot \alpha \\
&= \mathcal{O}(n \cdot \log(n) + \log(\ell) \cdot \log(1/\tau_{\min})/\rho)
\end{aligned}$$

Iterationen mit Wahrscheinlichkeit mindestens $1 - f_2$ abgearbeitet sind, wobei $f_2 := 1/(2 \cdot n^2)$ ist. Der erste Teil des Theorems folgt, da sich die beiden Fehlerwahrscheinlichkeiten f_1 und f_2 zu $1/n^2$ aufsummieren.

Wir müssen noch die erwartete Optimierungszeit begrenzen, falls die letzte Argumentation fehlschlägt. Die Wahrscheinlichkeit für einen solchen Fehlschlag ist durch $1/n^2$ beschränkt. Wir wissen, dass ein Paar (u,v) mit $\ell_{u,v} = 1$ unabhängig von den aktuellen Pheromonwerten mit Wahrscheinlichkeit $\Omega(\tau_{\min})$ optimiert wird. Folglich sind im Erwartungswert nach höchstens $\mathcal{O}(\log(n)/\tau_{\min})$ Iterationen alle solche Paare optimiert. Da die Argumentation der letzten Absätze für alle anfänglichen Pheromonwerte gültig ist, erhalten wir die obere Begrenzung $\mathcal{O}(\log(n)/\tau_{\min} + n \cdot \log(n) + \log(\ell) \cdot \log(1/\tau_{\min})/\rho)$ für die erwartete Optimierungszeit. Zusammen mit der ersten Schranke, die mit Wahrscheinlichkeit $1 - 1/n^2$ eingehalten wird, folgt der zweite Teil des Theorems, da die Annahme $\tau_{\min} \geq 1/n^3$ sicherstellt, dass $\mathcal{O}(1/n^2 \cdot \log(n)/\tau_{\min}) = \mathcal{O}(n \cdot \log n)$ gilt. □

Betrachte die Parameter $\tau_{\min} = 1/(\Delta \cdot \ell)$, $\tau_{\max} = 1 - 1/(\Delta \cdot \ell)$ und $\rho = (1-p_{\text{int}})/(12 \cdot \Delta \cdot \log(n))$. Beachte, dass die Hinzunahme eines neuen Knotens ohne eingehende Kanten zu einem Graphen die Optimierungszeit nicht verringern kann. Wenn wir den aus Abschnitt 9.3 bekannten Graphen $G^{\text{lb}}_{n,n-1}$ betrachten und einen zusätzlichen Knoten

mit $\Omega(n)$ ausgehenden Kanten und keinen eingehenden Kanten hinzufügen, dann erhalten wir einen Graphen mit linearem Höchstgrad, sodass die untere Schranke $\Omega(\ell/\tau_{\min}) = \Omega(n^3)$ für die Optimierungszeit des Ameisensystems $\text{MMAS}_{\text{SDSP}}$ aufgrund von Theorem 9.11 gilt. Diese untere Grenze gilt trivialerweise auch für das Ameisensystem $\text{MMAS}_{\text{APSP}}$ ohne Interaktion. Wenn $\Delta, \ell = \Omega(n)$ gilt, dann vereinfachen sich die oberen Schranken aus Theorem 9.15 und Theorem 9.16 zu $\mathcal{O}(n^3)$ bzw. $\mathcal{O}(n \cdot \log^3(n))$. Demzufolge profitiert das Ameisensystem von unserem einfachen Interaktionsmechanismus und dem verbesserten Informationsaustausch zwischen den einzelnen Ameisen.

Der vorgeschlagene Interaktionsmechanismus weist gewisse Ähnlichkeiten mit Rekombinationsoperatoren, die in genetischen Algorithmen zum Einsatz kommen, auf (siehe z. B. [Doerr et al., 2008a, Doerr und Theile, 2009]). Auch hier können mithilfe des Interaktionsmechanismus kürzeste Teilpfade zu einem längeren kürzesten Pfad kombiniert werden, wenn der Zwischenknoten richtig gewählt wird. Wie auch die Stelle, an der ein Rekombinationsoperator eine Bitfolge durchtrennt, meist rein zufällig gewählt wird, so ziehen wir den Zwischenknoten hier rein zufällig. Obschon der Interaktionsmechanismus die Optimierungszeit signifikant verbessert, sind uns keine praktischen Anwendungen eines vergleichbaren Mechanismus im Kontext von Ameisenkolonie-Optimierung bekannt. Möglicherweise ist dies darauf zurückzuführen, dass der Interaktionsmechanismus nicht vom natürliche Vorbild abgeleitet werden kann. Andererseits zeigt das Beispiel, dass es fruchtbar sein kann, Mechanismen aus unterschiedlichen Heuristiken miteinander zu verbinden.

9.5. Fazit

Die Befähigung von echten Ameisenkolonien, kürzeste Wege zu einer Futterquelle zu finden, hat die Entwicklung von Ameisenalgorithmen wesentlich motiviert. Aufbauend auf einer ersten Studie von Attiratanasunthron und Fakcharoenphol [2008] haben wir die Laufzeit von verschiedenen Ameisenalgorithmen für Kürzeste-Wege-Probleme auf rigorose Weise analysiert. Unsere Ergebnisse (siehe Tabelle 9.1) stellen deutliche Verbesserungen und Verallgemeinerungen der bekannten Ergebnisse für das Problem, kürzeste Wege zu einer Senke zu finden, dar. Wenn wir die Anzahl der Zielfunktionsauswertungen als Effizienzmaß heranziehen, ist die Schranke für den Algorithmus $\text{MMAS}_{\text{SDSP}}$ besser als die Schranke für den evolutionären Algorithmus (1+1)-EA [Doerr et al., 2007a], wenn $\Delta \cdot \ell = \text{o}(n)$ gilt und ρ nicht zu klein ist.

Für das Problem, kürzeste Wege zwischen allen Paaren von Knoten zu finden, konnten erste Ergebnisse für den Algorithmus $\text{MMAS}_{\text{APSP}}$, der eine Verallgemeinerung des Algorithmus $\text{MMAS}_{\text{SDSP}}$ darstellt, erzielt werden. Wir haben bewiesen, dass es überraschenderweise die Ergebnisse signifikant verbessert, wenn die Ameisen die Möglichkeit bekommen, zwischenzeitlich fremden Pheromonspuren zu zufälligen Zielen zu folgen. Das Ergebnis ist außerdem das erste Resultat für ein kombinatorisches Optimierungsproblem, bei dem eine langsame Anpassung der Pheromonwerte entscheidend ist, d. h., kleine Werte für den Evaporationsfaktor ρ liefern die

besten oberen Schranken. Für eine optimale Wahl von ρ stellt die obere Schranke $\mathcal{O}(n^3 \cdot \log^3(n))$ eine Verbesserung gegenüber der Schranke $\Theta(n^{3,25} \cdot \log^{0,25}(n))$ für einen genetischen Algorithmus [Doerr und Theile, 2009] dar, wenn wir die Anzahl der Funktionsauswertungen zugrunde legen. Daher ist Ameisenkolonie-Optimierung – unter theoretischen Gesichtspunkten betrachtet – die derzeit beste bekannte Metaheuristik für das Kürzeste-Wege-Problem für alle Paare. Dennoch kann sie natürlich nicht mit problemspezifischen Algorithmen wie z. B. Dijkstras Algorithmus (Laufzeit: $\mathcal{O}(n^3)$) konkurrieren, da die erwartete Laufzeit $\mathcal{O}(n^5 \cdot \log^3(n))$ beträgt, wenn wir die Laufzeit pro Iteration einbeziehen (siehe auch Abschnitt 9.2).

Unsere oberen Schranken können für viele andere Distanz- oder Gewichtsfunktionen verallgemeinert werden. Die einzige Voraussetzung an die Gewichtsfunktion, derer wir uns in den Analysen bedient haben, ist die folgende Bedingung, die für alle Knoten u, w und v gelten muss. Wenn w ein Knoten auf einem kürzesten Pfad von u nach v ist, dann ist die Konkatenation von einem beliebigen kürzesten Pfad von u nach w und einem beliebigen kürzesten Pfad von w nach v ein kürzester Pfad von u nach v. Beispielsweise übertragen sich unsere Ergebnisse auch dann, wenn die Länge eines Pfades nicht durch die Summe der Gewichte sondern durch das Minimum der Gewichte definiert ist.

Da die Längen der konstruierten Pfade stets von Grund auf neu berechnet werden, eignen sich unsere Ameisenalgorithmen gerade für den Einsatz in Umgebungen, in denen sich beispielsweise das Netzwerk oder die Kantengewichte dynamisch ändern. Die Bestimmung der Rahmenbedingungen, unter denen Ameisen in der Lage sind, dynamische Kürzeste-Wege-Probleme effizient zu lösen, stellt ein interessantes Thema für zukünftige Arbeiten dar. Diese Betrachtungsweise spiegelt auch die Tatsache wider, dass Ameisenalgorithmen oft eingesetzt werden, um zeitlich veränderliche Probleme zu lösen. Auch Untersuchungen der Robustheit von Ameisensystemen erscheinen lohnenswert. Der Frage, ob Ameisensystemen kürzeste Wege auch dann noch finden können, wenn die Gewichte verrauscht sind, werden wir in Kapitel 10 nachgehen. Außerdem wäre es interessant, Ameisenalgorithmen auf Kürzeste-Wege-Problemen mit Nebenbedingungen oder Varianten wie Routenplanungsproblemen zu analysieren. Wir erwarten, dass sich die in diesem Kapitel verwendeten Techniken und die gewonnenen Einsichten als nützlich erweisen werden, um weitere Resultate für auf Graphen definierte Probleme, wie beispielsweise das Problem, einen minimalen aufspannenden Baum zu finden, oder das Problem des Handlungsreisenden, zu gewinnen.

10. Ein Ameisensystem für ein verrauschtes Kürzeste-Wege-Problem

10.1. Einleitung

In diesem Kapitel erweitern wir die aus Kapitel 9 bekannten Ergebnisse für das Kürzeste-Wege-Problem für eine Senke. Wir vollziehen einen nächsten Schritt, indem wir eine stochastische Variante des Problems auf gerichteten azyklischen Graphen analysieren. Viele verschiedene Varianten von stochastischen Kürzeste-Wege-Problemen sind in der Literatur über Optimierung untersucht worden (siehe z. B. [Nikolova et al., 2006, Miller-Hooks und Mahmassani, 2000]). Eine bekannte Variante besteht beispielsweise darin, einen Pfad, der die kleinste erwartete Zeit aufweist (auf Englisch „Least Expected Time“, LET), zu finden [Bertsekas und Tsitsiklis, 1991, Papadimitriou und Yannakakis, 1991]. Ein anderes Problem besteht darin, die Wahrscheinlichkeit, das Ziel innerhalb eines gegebenen Zeitlimits zu erreichen, zu maximieren [Boyan und Mitzenmacher, 2001, Fan et al., 2005].

Wir konzentrieren uns hier auf ein Szenario, in dem den Kantengewichten ein stochastisches Rauschen hinzugefügt wird. Das Rauschen ist nicht negativ und die Aufgabe besteht darin, die echten kürzesten Pfade – d. h., die kürzesten Pfade ohne Rauschen – zu finden oder zu approximieren. Wir stellen die Hauptfrage, in welchen Szenarien die Ameisen noch dazu in der Lage sind, kürzeste Pfade effizient zu finden, wenn die Länge der Pfade verrauscht ist. Die Leserinnen und Leser können sich unter dem Rauschen beispielsweise die Formalisierung von Messfehlern vorstellen. Das LET-Problem kann als Spezialfall gesehen werden, in dem die kürzesten Pfade auch die kürzeste erwartete Länge aufweisen. Soweit wir wissen, ist dies die erste Laufzeitanalyse einer randomisierten Suchheuristik auf einem stochastischen kombinatorischen Optimierungsproblem.

Dieses Kapitel ist wie folgt gegliedert. In Abschnitt 10.2 führen wir das unseren Untersuchungen zugrunde liegende Szenario, das Kürzeste-Wege-Problem und das Ameisensystem formal ein. Abschnitt 10.3 enthält einige Eigenschaften der bekannten Gamma-Verteilung, die eine gebräuchliche Wahl für die Modellierung von stochastischem Rauschen darstellt. In Abschnitt 10.4 präsentieren wir eine allgemeine obere Schranke für die Zeit, bis eine angemessene Approximation gefunden wird, wenn die Zufallsvariablen, die das Rauschen auf den verschiedenen Kanten in dem Graphen modellieren, unabhängig sind. Wir betrachten in Abschnitt 10.5 unabhängiges gamma-verteiltes Rauschen und beweisen eine exponentielle untere Schranke

Algorithmus 11 Pfadkonstruktion von u nach v

initialisiere $i \leftarrow 0$, $p_0 \leftarrow u$ und $V_1 \leftarrow \{p \in V \mid (p_0, p) \in E\}$
while $p_i \neq v$ und $V_{i+1} \neq \emptyset$ **do**
    $i \leftarrow i + 1$
    wähle $p_i \in V_i$ mit Wahrscheinlichkeit $\tau((p_{i-1}, p_i)) / \sum_{p \in V_i} \tau((p_{i-1}, p))$
    $V_{i+1} \leftarrow \{p \in V \mid (p_i, p) \in E\}$
end while
return $(p_0, \ldots, p_i)$

für einen konstruierten Graphen. Abschnitt 10.6 ist dem Fall gewidmet, in dem das Rauschen auf den einzelnen Kanten stark korreliert ist. Wir beenden das Kapitel in Abschnitt 10.7.

10.2. Problem und Algorithmus

Betrachte einen gewichteten gerichteten azyklischen Graphen $G = (V, E, w)$ mit $V = \{1, \ldots, n\}$, $E = \{e_1, \ldots, e_m\} \subseteq V \times V$ und $w \colon E \to \mathbb{R}_0^+$. Wir interessieren uns für kürzeste Wege von allen Quellen $v \in V$ zu der festen Senke n. Wir fassen eine Folge $p = (v_0, \ldots, v_\ell)$ von Knoten $v_i \in V$ mit $(v_{i-1}, v_i) \in E$, $1 \leq i \leq \ell$, als einen *Pfad* von v_0 nach v_ℓ mit ℓ Kanten auf. Wir fassen zudem die zugehörige Folge $p = ((v_0, v_1), \ldots, (v_{\ell-1}, v_\ell))$ von Kanten $(v_{i-1}, v_i) \in E$ als einen *Pfad* auf. Im Folgenden verwenden wir beide Repräsentationen, um die Notation zu erleichtern. Wir definieren die *Länge* $w(p)$ eines Pfades $p = (e_1, \ldots, e_\ell)$ durch $w(p) := \sum_{i=1}^{\ell} w(e_i)$, wenn er an der Senke endet. Anderenfalls definieren wir $w(p) := \infty$. Den Ausgrad eines Knotens v schreiben wir als $\deg(v)$ und den maximalen Ausgrad eines beliebigen Knotens in dem Graphen notieren wir als Δ.

Wir untersuchen eine stochastische Version des beschriebenen Kürzeste-Wege-Problems. Der Begriff „stochastisch“ bedeutet, dass die Auswertung der Länge eines Pfades p nicht die *tatsächliche Länge* $w(p)$ sondern eine *verrauschte Länge* $\tilde{w}(p)$ liefert. Nimm an, dass eine Familie $(\eta(e))_{e \in E}$ von nicht negativen Zufallsvariablen $\eta(e) \geq 0$ gegeben ist. Die verrauschte Länge $\tilde{w}(e)$ einer Kante $e \in E$ ist durch $\tilde{w}(e) = (1 + \eta(e)) \cdot w(e)$ definiert und die verrauschte Länge $\tilde{w}(p)$ eines Pfades $p = (e_1, \ldots, e_\ell)$ durch

$$\tilde{w}(p) = \sum_{i=1}^{\ell} (1 + \eta(e_i)) \cdot w(e_i).$$

Beachte, dass $\tilde{w}(e) \geq w(e)$ für jede Kante e und dass $\tilde{w}(p) \geq w(p)$ für jeden Pfad p gelten. Beachte außerdem, dass die Intensität des Rauschens von dem Gewicht der zugehörigen Kante abhängt.

Wir betrachten das Ameisensystem $\text{MMAS}_{\text{SDSP}}$ aus Kapitel 9. Wenn der zugrunde liegende Graph azyklisch ist, kann das Ameisensystem wie folgt beschrieben werden (siehe Algorithmus 11 und 12). In jeder Iteration startet von jedem Knoten eine Ameise, die versucht, einen Pfad zum Ziel zu konstruieren. Dabei vollzieht die Ameise

Algorithmus 12 $\text{MMAS}_{\text{SDSP}}$

```
initialisiere Pheromone τ und beste bisher gefundene Pfade p*_1, ..., p*_n
repeat
    for u = 1, ..., n do in parallel
        konstruiere einfachen Pfad p_u von u nach n bezüglich τ
        werte w̃(p_u) aus
        if w̃(p_u) ≤ w̃(p*_u) then p*_u ← p_u end if
        aktualisiere Pheromone τ auf allen Kanten (u, ·) bezüglich p*_u
    end for
until Abbruchkriterium erfüllt
return p*_1, ..., p*_n
```

eine zufällige Irrfahrt durch den Graphen, die von den Pheromonen beeinflusst wird. Genauer gesagt wird die nächste Kante immer mit einer Wahrscheinlichkeit gewählt, die durch den Quotienten aus dem Pheromonwert auf der Kante und der Summe der Pheromonwerte auf allen ausgehenden Kanten gegeben ist. Wenn die Ameise in eine Sackgasse läuft, d. h., einen Knoten erreicht, der über keine ausgehenden Kanten verfügt, ist die Länge des resultierenden Pfades ∞ aufgrund der Definition von w. Dies kann nur dann passieren, wenn es Knoten gibt, von denen aus der Zielknoten nicht erreichbar ist. Die Pfadkonstruktion wird detailliert in Algorithmus 11 dargestellt. Beachte, dass Algorithmus 11 die Arbeitsweise von Algorithmus 8 im Kontext von gerichteten azyklischen Graphen einfacher darstellt.

Sobald die Ameise, die von einem Knoten u gestartet ist, einen Pfad konstruiert hat, wird der beste bisher gefundene Pfad p_u^* aktualisiert, wenn der neue Pfad p_u nicht schlechter als der zuvor genannte Pfad ist. Beachte, dass diese Entscheidung auf der Basis der verrauschten Pfadlängenauswertungen $\tilde{w}(p_u^*)$ und $\tilde{w}(p_u)$ getroffen wird. Ein wichtiges Detail besteht darin, dass wir die verrauschte Länge $\tilde{w}(p_u^*)$ speichern und $\tilde{w}(p_u^*)$ *nicht* für jeden Vergleich neu auswerten. Anschließend aktualisiert die Ameise die Pheromonwerte auf den Kanten, die ihren Startknoten verlassen, im Einklang mit dem besten bisher gefundenen Pfad p_u^* gemäß der folgenden Formel. Pheromone auf einer Kante e notieren wir als $\tau(e)$. Eingangs setzen wir $\tau(e) = 1/\deg(v)$ für alle $e = (v, \cdot)$ sowie $p_v^* = ()$ und $\tilde{w}(p_v^*) = \infty$ für alle $v \in V$. Der Evaporationsfaktor $0 \leq \rho \leq 1$ sowie die Pheromonschranken $\tau_{\min}$ and $\tau_{\max}$ sind Parameter des Algorithmus. Die Formel für die Aktualisierung der Pheromonwerte ist durch

$$\tau(e) \leftarrow \begin{cases} \min\{(1-\rho)\cdot\tau(e) + \rho,\ \tau_{\max}\} & \text{falls } e = (u,v) \in p_u^* \\ \max\{(1-\rho)\cdot\tau(e),\ \tau_{\min}\} & \text{falls } e = (u,v) \notin p_u^* \end{cases}$$

gegeben. Die sogenannten Pheromonschranken $\tau_{\min}$ und $\tau_{\max}$ stellen sicher, dass die Pheromonwerte auf jeder Kante einen gewissen Abstand von 0 aufweisen, sodass die Chance besteht, eine einmal gefällte (falsche) Entscheidung rückgängig zu machen. Genau wie in Kapitel 9 fixieren wir $\tau_{\max} := 1 - \tau_{\min}$ und variieren ausschließlich $\tau_{\min}$, wobei wir die offensichtlichen Grenzen $0 < \tau_{\min} \leq 1/\Delta$ respektieren. Beachte,

dass der Algorithmus einfach parallelisiert werden kann, da die Pfadkonstruktionen unabhängig voneinander sind und die einzelnen Ameisen die Pheromonwerte von disjunkten Kantenmengen aktualisieren. Das gesamte Ameisensystem wird in Algorithmus 12 beschrieben.

Wir interessieren uns für die Anzahl von Iterationen bis das folgende Ziel erreicht wird. Sei OPT_v die Länge $w(p)$ eines kürzesten Pfades p von $v \in V$ nach n. Bezeichne $P_v(\alpha)$ die Menge, die jeden Pfad p von $v \in V$ nach n mit $w(p) \leq \alpha \cdot OPT_v$ enthält, wobei $\alpha \in \mathbb{R}$ mit $\alpha \geq 1$ ist. Wir nennen einen Knoten v genau dann *α-approximiert*, wenn $w(p_v^*) \leq \alpha \cdot OPT_v$ gilt. Wir nennen einen Knoten v genau dann *permanent α-approximiert*, wenn $\tilde{w}(p_v^*) \leq \alpha \cdot OPT_v$ erfüllt ist. Der Unterschied zwischen diesen beiden Definitionen kann wie folgt erläutert werden. Wenn v α-approximiert ist, kann die Länge des besten bisher gefundenen Pfades größer als eine α-Approximation sein, da wir von diesem Pfad die verrauschte Länge $\tilde{w}(p_v^*)$ speichern. Die Ameise kann den besten bisher gefundenen Pfad später ändern und einen Pfad mit einer kleineren verrauschten Länge akzeptieren, dessen tatsächliche Länge jedoch größer ist. Auf diese Weise kann v die Eigenschaft verlieren, α-approximiert zu sein. Andererseits ist sichergestellt, dass ein Knoten, der permanent α-approximiert ist, im weiteren Verlauf der Optimierung auch permanent α-approximiert bleibt.

Wir sagen, dass das Ameisensystem $\text{MMAS}_{\text{SDSP}}$ eine α-Approximation gefunden hat, wenn alle $v \in V$ zeitgleich α-approximiert sind. Wenn eine 1-Approximation gefunden ist, sind insbesondere alle kürzesten Pfade gefunden worden. Beachte, dass es im Allgemeinen nicht sinnvoll ist, auf eine permanente 1-Approximation zu hoffen, da die Wahrscheinlichkeit, den Rauschwert 0 zu beobachten, 0 sein kann.

10.3. Die Gamma-Verteilung

Die Gamma-Verteilung wird häufig benutzt, um stochastische Verzögerungen zu modellieren [Fan et al., 2005]. Sie kann mithilfe eines Formparameters $k \in \mathbb{R}^+$ und eines Skalierungsparameters $\theta \in \mathbb{R}^+$ eingestellt werden. Wir gebrauchen die Schreibweise $X \sim \Gamma(k, \theta)$ für gamma-verteilte Zufallsvariable X. Die zugehörige Wahrscheinlichkeitsdichte ist durch die Funktion

$$f_X(x) = \begin{cases} 0 & \text{falls } x \leq 0 \\ x^{k-1} \cdot \frac{e^{-x/\theta}}{\theta^k \cdot \Gamma(k)} & \text{falls } x > 0 \end{cases}$$

gegeben, wobei Γ die sogenannte Gamma-Funktion $\Gamma(k) = \int_0^\infty t^{k-1} \cdot e^{-t}\,\mathrm{d}t$ bezeichnet. Wenn $k \in \mathbb{N}$ gilt, vereinfacht sich die Wahrscheinlichkeitsdichte zu

$$f_X(x) = x^{k-1} \cdot \frac{e^{-x/\theta}}{\theta^k \cdot (k-1)!}$$

für alle $x > 0$. In dem Spezialfall $k = 1$ entspricht die Gamma-Verteilung der Exponentialverteilung mit Mittelwert θ. Im Allgemeinen entspricht die Gamma-Verteilung mit $k \in \mathbb{N}$ der Summe von k solchen exponentialverteilten Zufallsvariablen. In diesem

Fall ist die Verteilung auch als Erlang-Verteilung bekannt. Der Erwartungswert von X entspricht $\mathrm{E}(X) = k \cdot \theta$.

Die Verteilungsfunktion ist für alle $k \in \mathbb{N}$ durch

$$F_X(x) = \mathrm{W}(X \leq x) = 1 - \sum_{i=0}^{k-1} \frac{(x/\theta)^i}{i!} \cdot e^{-x/\theta}.$$

gegeben. Die Gamma-Verteilung besitzt die folgenden Eigenschaften.

1. Eine Summe von m unabhängigen, gamma-verteilten Zufallsvariablen mit dem gleichen Skalierungsparameter ist auch gamma-verteilt: Wenn wir für $1 \leq i \leq m$ unabhängige Zufallsvariable $X_i \sim \Gamma(k_i, \theta)$ betrachten, dann gilt

$$\sum_{i=1}^{m} X_i \sim \Gamma\left(\sum_{i=1}^{m} k_i, \theta\right).$$

2. Eine Gamma-Verteilung ist auch skalierungsinvariant: Wenn wir eine Zufallsvariable $X \sim \Gamma(k, \theta)$ und einen Skalierungsfaktor $\alpha > 0$ betrachten, dann gilt

$$\alpha \cdot X \sim \Gamma(k, \alpha \cdot \theta).$$

Das nachfolgende Lemma liefert Abschätzungen für die Verteilungsfunktion einer gamma-verteilten Zufallsvariablen und wird sich im Folgenden als nützlich erweisen.

Lemma 10.1. *Betrachte eine gamma-verteilte Zufallsvariable X mit $X \sim \Gamma(k, \theta)$, wobei $k \in \mathbb{N}$ und $\theta \in \mathbb{R}^+$ sind. Dann gilt für alle $x \in \mathbb{R}^+$*

$$\frac{(x/\theta)^k}{k!} \cdot e^{-x/\theta} \leq \mathrm{W}(X \leq x) \leq \frac{(x/\theta)^k}{k!}.$$

Beweis. Es gilt

$$\mathrm{W}(X \leq x) = 1 - \sum_{i=0}^{k-1} \frac{(x/\theta)^i}{i!} \cdot e^{-x/\theta} = 1 - \left(e^0 - \frac{e^\xi}{k!} \cdot (x/\theta)^k\right)$$

für ein $\xi \in [-x/\theta, 0]$, indem wir die Lagrange-Form des Restgliedes des $(k-1)$-ten Taylorpolynoms für e^x an der Entwicklungsstelle $-x/\theta$ benutzen. Wir schließen

$$\frac{e^{-x/\theta}}{k!} \cdot (x/\theta)^k \leq \mathrm{W}(X \leq x) \leq \frac{e^0}{k!} \cdot (x/\theta)^k.$$

□

Wir beweisen auch die folgende Abschätzung für den „Schwanz“ der Verteilung.

Lemma 10.2. *Betrachte eine gamma-verteilte Zufallsvariable X mit $X \sim \Gamma(k, \theta)$, wobei $k \in \mathbb{N}$ und $\theta \in \mathbb{R}^+$ sind. Dann gilt für alle $x \in \mathbb{R}^+$ mit $x \geq k \cdot \theta = \mathrm{E}(X)$*

$$\mathrm{W}(X > x) \leq e^{-x/\theta} \cdot (x/\theta)^{k-1} \cdot \frac{k}{(k-1)!}.$$

Beweis. Indem wir $x/\theta \geq k \geq 1$ ausnutzen, schließen wir

$$\begin{aligned}\sum_{i=0}^{k-1} \frac{(x/\theta)^i}{i!} &= \sum_{i=0}^{k-1} \frac{(x/\theta)^{k-1}}{i! \cdot (x/\theta)^{k-i-1}} \\ &\leq (x/\theta)^{k-1} \cdot \sum_{i=0}^{k-1} \frac{1}{i! \cdot k^{k-i-1}} \\ &\leq (x/\theta)^{k-1} \cdot \frac{k}{(k-1)!}.\end{aligned}$$

Indem wir dies in die Verteilungsfunktion einsetzen, erhalten wir die behauptete Schranke. □

10.4. Eine allgemeine obere Schranke für unabhängiges Rauschen

Im Folgenden nehmen wir an, dass die Zufallsvariablen $\eta(e)$, $e \in E$, unabhängig sind. Jedes Mal, wenn ein neuer Pfad konstruiert wird, werden neue Zufallsvariable $\eta(e)$ benutzt, um seine verrauschte Länge zu bestimmen, d. h., alle $\eta(e)$ sind unabhängig für jede Kante, für jede Ameise und für jede Iteration. Das bedeutet, dass in einer Iteration verschiedene Ameisen unterschiedliches Rauschen wahrnehmen können, auch wenn sie den gleichen Kanten folgen. Wir erinnern an dieser Stelle daran, dass die Länge eines besten bisher gefundenen Pfades nicht neu ausgewertet wird, wenn sie mit der Länge eines neuen Pfades verglichen wird.

Wir fangen mit einer sehr allgemeinen oberen Schranke für die Optimierungszeit an, die für alle Rauschverteilungen gilt. Die folgenden Aussagen wurden bereits implizit in Kapitel 9 bewiesen. Wir formulieren sie hier für gerichtete azyklische Graphen.

Lemma 10.3. *Das Ameisensystem $MMAS_{SDSP}$ weist die folgenden Eigenschaften auf gerichteten azyklischen Graphen auf.*

1. *Die Wahrscheinlichkeit, dass eine Ameise auf $u \neq n$ der Kante $e = (u, v)$ folgt, beträgt höchstens $\tau(e) \leq \tau_{\max}$ und mindestens $\tau(e)/2 \geq \tau_{\min}/2$.*

2. *Wenn der Pheromonwert auf einer Kante in mindestens $T^* := \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen verringert wurde, beträgt er $\tau_{\min}$.*

3. *Betrachte eine Menge $\mathcal{S}_v$ von Pfaden von einem beliebigen Knoten $v \in V$ zu der Senke n, sodass die Kanten, die $\mathcal{S}_v$ verlassen, den Pheromonwert $\tau_{\min}$*

aufweisen. Wenn jeder Pfad aus $\mathcal{S}_v$ aus höchstens $\ell - 1$ Kanten besteht und $\tau_{\min} \leq 1/(\Delta \cdot \ell)$ gilt, dann beträgt die Wahrscheinlichkeit, dass eine Ameise einem Pfad aus $\mathcal{S}_v$ von v nach n folgt, mindestens $1/e$.

Beweis. 1. Die erste Aussage folgt unmittelbar aus Korollar 9.2.

2. Betrachte eine Kante e. Bezeichne $\tau_{\min} \leq \tau_i(e) \leq \tau_{\max}$ den Pheromonwert auf e zu Beginn der i-ten Iteration. Wenn der Pheromonwert in j aufeinander folgenden Iterationen nicht intensiviert wird, dann gilt zu Beginn der $(i+j)$-ten Iteration $\tau_{i+j}(e) = \max\{(1-\rho)^j \cdot \tau_i(e), \tau_{\min}\}$. Die zweite Aussage folgt mithilfe von Korollar A.19 aus

$$(1-\rho)^{T^*} \cdot \tau_i(e) \leq (1-\rho)^{\ln(\tau_{\max}/\tau_{\min})/\rho} \cdot \tau_{\max} \leq e^{-\ln(\tau_{\max}/\tau_{\min})} \cdot \tau_{\max} = \tau_{\min}.$$

3. Betrachte einen Knoten u, $u \neq n$, der auf einem Pfad aus $\mathcal{S}_v$ liegt. Dann beträgt der Pheromonwert auf jeder Kante $(u, \cdot)$, die nicht auf einem Pfad aus $\mathcal{S}_v$ liegt, höchstens $\tau_{\min}$. Demzufolge folgt die Ameise einer Kante, die sich auf einem Pfad aus $\mathcal{S}_v$ befindet, mindestens mit Wahrscheinlichkeit $1-(\deg(u)-1)\cdot\tau_{\min} \geq 1 - \Delta \cdot \tau_{\min}$, wobei hier die erste Aussage dieses Korollars eingeht. Also beträgt die Wahrscheinlichkeit, dass eine Ameise, die auf dem Knoten v startet, den Knoten n erreicht, mindestens $(1 - \Delta \cdot \tau_{\min})^{\ell-1} \geq (1 - 1/\ell)^{\ell-1} \geq 1/e$.

□

Diese drei Aussagen werden uns in den folgenden Beweisen entscheidende Dienste erweisen. Das nächste Theorem liefert einen Kompromiss zwischen der gewünschten Approximationsgüte und der benötigten Optimierungszeit gemäß der Variablen α.

Theorem 10.4. *Betrachte einen Graphen $G = (V, E, w)$ mit Gewicht $w(e) > 0$ und Rauschen $\eta(e) \geq 0$ für jede Kante $e \in E$. Wähle den Parameter $\tau_{\min} \leq 1/(\Delta \cdot L)$, wobei L die maximale Anzahl von Kanten auf jedem Pfad in G bezeichnet. Sei $\chi = (1 + \alpha \cdot \eta_{\max})^L$, $\alpha \in \mathbb{R}$, $\alpha > 1$, wobei $\eta_{\max} := \max_{1 \leq i \leq m} \mathrm{E}(\eta(e_i))$ ist. Dann findet das Ameisensystem $MMAS_{SDSP}$ eine χ-Approximation im Erwartungswert nach höchstens $\mathcal{O}((L \cdot (\log n)/\tau_{\min}) \cdot (\alpha/(\alpha-1)) + L \cdot \log(1/\tau_{\min})/\rho)$ Iterationen.*

Beachte, dass die Approximationsgüte $\chi \leq e^{\alpha \cdot \eta_{\max} \cdot L}$ gegen 1 konvergiert, wenn $\eta_{\max} = \mathrm{o}(1/(\alpha \cdot L))$ gilt.

Beweis. Wir definieren eine Partition von V gemäß der Höchstanzahl von Kanten auf beliebigen Pfaden zum Ziel. Sei

$$V_i := \{v \in V \mid \exists p \in P_v \colon |p| \geq i \text{ und } \forall p \in P_v \colon |p| \leq i\}$$

mit $0 \leq i \leq L$, wobei P_v die Menge, die alle Pfade von v nach n enthält, und $|p|$ die Anzahl von Kanten auf p bezeichnen. Dann bilden $V_0, \ldots, V_L$ eine Partition von V. Beachte, dass es für jede Kante $(u, v) \in E$ Indizes i und j mit $u \in V_i$, $v \in V_j$ und $i > j$ gibt.

Betrachte einen Knoten $u \in V_i$. Nimm an, dass für jeden Index $0 \leq j < i$ jeder Knoten $v \in V_j$ seit mindestens $T^* = \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen permanent $\chi^{j/L}$-approximiert ist. Wir begrenzen nun die erwartete Zeit, bis u permanent $\chi^{i/L}$-approximiert wird. Für jede Kante $(v, \cdot)$, $v \in V_j$, $0 \leq j < i$, die nicht zu einem Pfad aus $P_v(\chi^{j/L})$ ergänzt werden kann, gilt $\tau((v,\cdot)) = \tau_{\min}$ aufgrund der zweiten Aussage aus Lemma 10.3. Schreibe die Ameise, die auf dem Knoten u beginnt, als a_u. Die Wahrscheinlichkeit, dass die Ameise a_u die erste Kante (u, v), $v \in V_j$, $0 \leq j < i$, eines Pfades aus $P_u(1)$ wählt (d. h., eines kürzesten Pfades), ist gemäß der ersten Aussage aus Lemma 10.3 durch $\tau_{\min}/2$ nach unten beschränkt. Indem wir uns auf die dritte Aussage aus Lemma 10.3 mit $\mathcal{S}_v = P_v(\chi^{j/L})$ berufen, schließen wir, dass die Wahrscheinlichkeit, dass die Ameise a_u von v auf einem Pfad aus $P_v(\chi^{j/L})$ nach n läuft, mindestens $1/e$ beträgt. Insgesamt beträgt die Wahrscheinlichkeit, dass die Ameise a_u einen beliebigen Pfad p aus $P_u(\chi^{j/L})$ findet, mindestens $\tau_{\min}/(2 \cdot e)$.

Einen solchen Pfad $p = (e_1, \ldots, e_k)$ zu finden, reicht nicht aus; die verrauschte Längenauswertung von p darf zusätzlich nicht zu schlecht sein. Der Knoten u wird permanent $\chi^{i/L}$-approximiert, wenn $\tilde{w}(p) \leq \chi^{i/L} \cdot OPT_u$ gilt. Es gilt

$$\begin{aligned}
\chi^{i/L} \cdot OPT_u &= \chi^{i/L} \cdot (w(e_1) + OPT_v) \\
&\geq \chi^{i/L} \cdot \left(w(e_1) + \frac{\sum_{i=2}^{k} w(e_i)}{\chi^{j/L}} \right) \\
&= \chi^{j/L} \cdot w(e_1) + \chi^{(i-j)/L} \cdot \sum_{i=1}^{k} w(e_i).
\end{aligned}$$

Folglich gilt

$$\begin{aligned}
\mathrm{W}(\tilde{w}(p) \geq \chi^{i/L} \cdot OPT_u) &\leq \mathrm{W}\left(\sum_{i=1}^{k} (1 + \eta(e_i)) \cdot w(e_i) \geq \chi^{1/L} \cdot \sum_{i=1}^{k} w(e_i) \right) \\
&= \mathrm{W}\left(\sum_{i=1}^{k} \eta(e_i) \cdot w(e_i) \geq (\chi^{1/L} - 1) \cdot \sum_{i=1}^{k} w(e_i) \right).
\end{aligned}$$

Mithilfe der Markoff-Ungleichung kann diese Wahrscheinlichkeit nach oben durch

$$\frac{\mathrm{E}(\sum_{i=1}^{k} \eta(e_i) \cdot w(e_i))}{(\chi^{1/L} - 1) \cdot \sum_{i=1}^{k} w(e_i)} = \frac{\sum_{i=1}^{k} \mathrm{E}(\eta(e_i)) \cdot w(e_i)}{\alpha \cdot \eta_{\max} \cdot \sum_{i=1}^{k} w(e_i)} \leq \frac{1}{\alpha} < 1.$$

abgeschätzt werden. Demzufolge gilt $\tilde{w}(p) < \chi^{i/L} \cdot w(p)$ mit Wahrscheinlichkeit mindestens $1 - 1/\alpha > 0$. Also ist die Wahrscheinlichkeit, dass der Knoten $u \in V_i$ permanent $\chi^{i/L}$-approximiert wird, durch $(\tau_{\min}/(2 \cdot e)) \cdot (1 - 1/\alpha)$ nach unten begrenzt.

Wir haben gezeigt, dass die Wahrscheinlichkeit, für einen *festen* Knoten aus V_i eine permanente $\chi^{i/L}$-Approximation zu finden, mindestens $(\tau_{\min}/(2 \cdot e)) \cdot ((\alpha - 1)/\alpha)$ beträgt. Nach $T := ((2 \cdot e)/\tau_{\min}) \cdot (\alpha/(\alpha - 1)) \cdot \ln(2 \cdot |V_i|)$ Schritten ist ein fester Knoten aus V_i höchstens mit Wahrscheinlichkeit

$$\left(1 - \frac{\tau_{\min}}{2 \cdot e} \cdot \frac{\alpha - 1}{\alpha}\right)^{T} \leq \exp\left(-\frac{\tau_{\min}}{2 \cdot e} \cdot \frac{\alpha - 1}{\alpha} \cdot \frac{2 \cdot e}{\tau_{\min}} \cdot \frac{\alpha}{\alpha - 1} \cdot \ln(2 \cdot |V_i|) \right) = \frac{1}{2 \cdot |V_i|}$$

nicht permanent $\chi^{i/L}$-approximiert. Indem wir die boolesche Ungleichung benutzen, schließen wir, dass nach T Schritten alle Knoten aus V_i die gewünschte Approximation mindestens mit Wahrscheinlichkeit $1 - |V_i|/(2 \cdot |V_i|) = 1/2$ erreicht haben. Also folgt, dass die erwartete Zeit, bis der letzte Knoten aus V_i permanent $\chi^{i/L}$-approximiert ist, höchstens $\mathcal{O}(((\log n)/\tau_{\min}) \cdot (\alpha/(\alpha - 1)))$ beträgt. Nach einer zusätzlichen Wartezeit von T^* Iterationen haben sich alle Pheromonwerte auf Kanten, die V_i verlassen, angepasst und wir setzen unsere Betrachtungen mit der Menge V_{i+1} fort. Indem wir die erwarteten Zeiten für alle $L + 1$ Mengen aufsummieren, erhalten wir die behauptete obere Schranke. □

Beachte, dass wir nicht hoffen können, dass das Ameisensystem eine Approximationsgüte $1+\epsilon$ mit $\epsilon < \mu := \eta_{\max} \cdot L$ erreicht. Sei $0 < \epsilon < \mu$ und betrachte den Graphen $G = (V, E)$ mit $V = \{1, \ldots, n\}$ und $E = \{(i, i+1) \mid n - L \leq i < n\} \cup \{(n - L, n)\}$. Weiterhin gelten $w(e) = 1/L$ und $\mathrm{W}(\eta(e) = \mu/L) = 1$ für $e = (i, i+1)$, $n-L \leq i < n$, sowie $w(e) = 1 + \epsilon$ und $\mathrm{W}(\eta(e) = 0) = 1$ für $e = (n - L, n)$. Betrachte die Pfade $p' = (n-L, \ldots, n)$ und $p'' = (n-L, n)$. Dann gelten folglich $w(p') = 1 < 1+\epsilon = w(p'')$ und $\mathrm{W}(\tilde{w}(p') = 1 + \mu > 1 + \epsilon = \tilde{w}(p'')) = 1$ für die tatsächliche bzw. die verrauschte Länge. Wenn die Ameise a_{n-L} dem Pfad p'' zum ersten Mal folgt, ersetzt dieser Pfad den aktuellen besten bisher gefundenen Pfad p^*_{n-L} und die Approximationsgüte beträgt dauerhaft $1 + \epsilon$.

Dieses Beispiel zeigt auch, dass das betrachtete Szenario zu allgemein ist, da deterministisches Rauschen jede beliebige Probleminstanz in jede beliebige Probleminstanz mit größeren oder gleichen Gewichten transformieren kann. Diese Beobachtung motiviert die Betrachtung der gleichen gamma-verteilten Rauschverteilung $\eta(e) \sim \Gamma(k, \theta)$ für alle Kanten e. In diesem Fall gilt $\mathrm{E}(\tilde{w}(p)) = (1 + k \cdot \theta) \cdot w(p)$ für jeden Pfad p. Das bedeutet, dass wir prinzipiell nach LET-Pfaden suchen, d. h., nach Pfaden, deren erwartete Länge am kürzesten ist.

10.5. Eine untere Schranke für unabhängiges Rauschen

In diesem Abschnitt werden wir eine untere Schranke für die zufällige Zeit beweisen, bis eine gute Approximation gefunden wird. Das Ergebnis gilt für einen „schwierigen“ Graphen, den wir im Folgenden beschreiben werden (für ein Beispiel siehe Abbildung 10.1). Zunächst besteht der Graph aus einigen der folgenden Subgraphen, die wir *Komponenten* nennen. Jede Komponente besitzt eine eindeutige Quelle und eine eindeutige Senke. Es gibt zwei Möglichkeiten bzw. Pfade, um eine Komponente zu durchlaufen: Eine Kette von m Knoten und eine einzelne Kante, die die Quelle direkt mit der Senke verbindet. Das Kantengewicht für die einzelne Kante ist so gewählt, dass das Gewicht der Kante um den Faktor $(1+\epsilon)$ größer als das Gewicht der gesamten Kette ist. Die einzelne Kante werden wir im Folgenden aus naheliegenden Gründen auch lange Kante nennen.

Die Hauptbeobachtung ist nun die Folgende. Wenn eine Ameise die Kette entlangläuft, ist die verrauschte Länge der Kette um ihren Erwartungswert konzentriert, da sie durch eine Summe von vielen unabhängigen, identisch-verteilten Zufallsvariablen

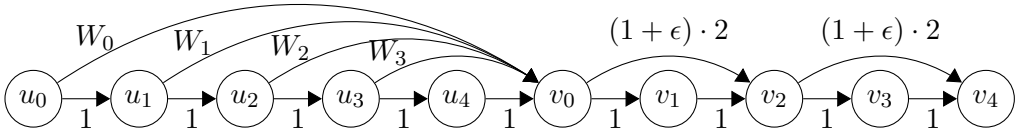


Abbildung 10.1.: Beispielgraph $G_{n,k,\theta,\epsilon}$ aus Definition 10.5 für $n = 10$ mit $W_i := 2 \cdot n \cdot (2 + 2 \cdot k \cdot \theta \cdot n)^{i+1}$.

gegeben ist. Andererseits ist die Varianz für die verrauschte Länge der langen Kante relativ groß. Das bedeutet, dass die Wahrscheinlichkeit groß ist, dass die Ameise ein kleines Rauschen wahrnimmt, wenn sie der langen Kante folgt, auch wenn die Kette im Erwartungswert eine kürzere verrauschte Länge aufweist. Wenn der Wert von ϵ nicht zu groß ist, dann resultiert dieses Verhalten darin, dass die Ameise die lange Kante als besten bisher gefundenen Pfad speichert und diesen mit Pheromonen intensiviert.

Um eine große untere Schranke beweisen zu können, die zudem mit überwältigender Wahrscheinlichkeit gilt, hängen wir einige der genannten Komponenten auf naheliegende Weise hintereinander. Die Senke einer Komponente wird dabei mit der Quelle der nächsten identifiziert. Wir werden beweisen, dass nach einiger Zeit mit hoher Wahrscheinlichkeit für alle Ameisen, die in den Komponenten starten, die jeweiligen unteren Kanten in ihren besten bisher gefundenen Pfaden enthalten sind und dass die Pheromone auf diesen Kanten den Maximalwert erreicht haben. Daraus folgt, dass die Ameise, die in der ersten Komponente startet, mit hoher Wahrscheinlichkeit fast allen langen Kanten folgt. Auf diese Weise werden wir die behauptete untere Schranke für die Approximationsgüte beweisen.

Die Konstruktion der „schwierigen“ Graphinstanz ist jedoch noch nicht komplett. Wie im letzten Absatz angesprochen, benötigen die Ameisen, die in den Komponenten starten, eine gewisse Zeitspanne, um die lange Kante entlangzulaufen und darauf ein Rauschen wahrzunehmen, das deutlich kleiner als der Erwartungswert ist. Wir benötigen also mit anderen Worten einen Trick, um die Ameisen Glauben zu machen, dass die langen Kanten zu kürzeren Pfaden führen. Wir können nicht ausschließen, dass am Anfang eines Laufes Ameisen in vielen Komponenten der Kette folgen. Daher ergänzen wir den Graphen um einen Teilgraphen, in dem die Ameisen eine bestimmte minimale Anzahl von Iterationen benötigen, um eine gute Approximation zu finden. Diesen Subgraphen hängen wir vor alle Komponenten, sodass das Verhalten der Ameisen, die in den Komponenten starten, nicht beeinflusst wird. Wir liefern nun eine formale Definition des beschriebenen Graphen. Abbildung 10.1 visualisiert den Graphen für $n = 10$.

Definition 10.5. *Seien $n \in \mathbb{N}$, $k \in \mathbb{N}$, $\theta \in \mathbb{R}^+$ und $\epsilon \in \mathbb{R}^+$. Ohne Beschränkung der Allgemeinheit betrachten wir nur n für die $m := \sqrt{n/2 - 1}$ eine ganze Zahl ist. Sei $G_{n,k,\theta,\epsilon} = (V, E, w)$ mit $V = \{u_0, \ldots, u_{n/2-1}, v_0, \ldots, v_{n/2-1}\}$ und den folgenden Mengen von Kanten. Wenn wir $u_{n/2}$ mit v_0 identifizieren, dann gibt es die Kanten*

(v_i, v_{i+1}) für $0 \leq i \leq n/2 - 2$ mit Gewicht 1, $(v_{i \cdot m}, v_{(i+1) \cdot m})$ für $0 \leq i \leq m - 1$ mit Gewicht $(1 + \epsilon) \cdot m$, (u_i, u_{i+1}) für $0 \leq i \leq n/2 - 1$ mit Gewicht 1 und $(u_i, u_{n/2})$ für $0 \leq i \leq n/2 - 2$, wobei die Kante $(u_i, u_{n/2})$ das Gewicht $W_i := 2 \cdot n \cdot (2 + 2 \cdot k \cdot \theta \cdot n)^{i+1}$ aufweist.

Unter Bezugnahme auf Abbildung 10.1 bezeichnen wir die Kanten mit Gewicht W_i oder $(1 + \epsilon) \cdot m$ als obere Kanten (oder oberer Pfad) und die verbleibenden Kanten als untere Kanten (oder unterer Pfad). Wir sprechen auch von einem linken Teil, der alle Knoten u_i enthält, und einem rechten Teil, der alle Knoten v_i umfasst.

Der erste Schritt, um eine untere Schranke zu beweisen, besteht darin, zu zeigen, dass die Ameisen, die in den Komponenten starten, nach hinreichend vielen Schritten die oberen Pfade favorisieren. Die Bedingung $k \cdot \theta / \epsilon \geq 4 \cdot \sqrt{n}$ kann als Kompromiss zwischen der erwarteten Rauschstärke und der Lücke zwischen den Kantengewichten auf den oberen und unteren Pfaden interpretiert werden. Je größer das erwartete Rauschen $k \cdot \theta$ ist, desto größer darf die Lücke sein.

Lemma 10.6. *Betrachte den Graphen $G_{n,k,\theta,\epsilon} = (V, E, w)$ mit Rauschen $\eta(e) \sim \Gamma(k, \theta)$ für jede Kante $e \in E$. Seien $k \geq 12$, $\epsilon \leq 1$ und $k \cdot \theta / \epsilon \geq 4 \cdot \sqrt{n}$. Wähle die Parameter $1/\operatorname{poly}(n) \leq \tau_{\min} \leq 1/(\Delta \cdot n)$ und $\rho \geq 1/\operatorname{poly}(n)$. Sei $x := \kappa \cdot m^2/\tau_{\min} + m \cdot T^*$, wobei $m = \sqrt{n/2 - 1}$ und $T^* = \ln(\tau_{\max}/\tau_{\min})/\rho$ sind und $\kappa > 0$ konstant ist. Dann erreichen wir mit Wahrscheinlichkeit $1 - \exp(-\Omega(\sqrt{n}))$ nach x Iterationen eine Situation, in der für jede Ameise, die von einem Knoten $v_{i \cdot m}$, $0 \leq i \leq m - 1$, startet, das Folgende gilt:*

1. *der beste bisher gefundene Pfad der Ameise beginnt mit der oberen Kante,*
2. *der Pheromonwert auf der ersten unteren Kante ist $\tau_{\min}$ und*
3. *die Wahrscheinlichkeit, dass in einer festen Iteration der beste bisher gefundene Pfad in einen Pfad geändert wird, der mit der unteren Kante beginnt, ist $\exp(-\Omega(\sqrt{n}))$.*

Beweis. Wir unterteilen die ersten $x = \kappa \cdot m^2/\tau_{\min} + m \cdot T^*$ Iterationen in m aufeinander folgende Phasen $1, \ldots, m$, die jeweils $\kappa \cdot m/\tau_{\min} + T^*$ Iterationen umfassen. Dann beweisen wir, dass nach Phase i die drei Aussagen für die Ameise, die auf dem linkesten Knoten der Komponente $m - i + 1$ startet, und für alle nachfolgenden Ameisen gelten.

Betrachte Phase i und nimm an, dass die vorangegangenen Phasen erfolgreich gewesen sind. Nimm außerdem an, dass das Ereignis, das in der dritten Aussage genannt wird, für keine Komponente in keiner Phase eintritt; mithilfe der booleschen Ungleichung können wir diese Fehlerwahrscheinlichkeit durch $(x \cdot m) \cdot \exp(-\Omega(\sqrt{n})) = \exp(-\Omega(\sqrt{n}))$ nach oben begrenzen, da wir $\tau_{\min}, \rho \geq 1/\operatorname{poly}(n)$ fordern. Wir nennen die Ameise, die auf dem linkesten Knoten aus Komponente $m - i + 1$ startet, die aktuelle Ameise. Die zweite Aussage für alle nachfolgenden Komponenten impliziert, dass die Wahrscheinlichkeit, dass die aktuelle Ameise einem Pfad folgt, der nur obere

Kanten enthält, mindestens $\tau_{\min}/(2 \cdot e)$ aufgrund von Lemma 10.3 beträgt. Wir zählen, wie oft die aktuelle Ameise diesem Pfad während der ersten $\kappa \cdot m/\tau_{\min}$ Iterationen der aktuellen Phase folgt. Wir definieren $X = \sum_{i=1}^{\kappa \cdot m/\tau_{\min}} X_i$, wobei X_i unabhängige, Bernoulli-verteilte Zufallsvariable mit $\mathrm{W}(X_i = 1) = \tau_{\min}/(2 \cdot e)$ und $\mathrm{W}(X_i = 0) = 1 - \tau_{\min}/(2 \cdot e)$ sind. Folglich gilt $\mathrm{E}(X) = (\kappa \cdot m/\tau_{\min}) \cdot (\tau_{\min}/(2 \cdot e)) = (\kappa \cdot m)/(2 \cdot e)$. Dann beträgt die Wahrscheinlichkeit, diesem Pfad weniger als $((\kappa \cdot m)/(4 \cdot e))$-mal zu folgen, höchstens

$$\begin{aligned}\mathrm{W}(X < (\kappa \cdot m)/(4 \cdot e)) &\leq \mathrm{W}(X \leq (1 - 1/2) \cdot \mathrm{E}(X)) \\ &\leq e^{-(1/2)^2 \cdot \mathrm{E}(X)/2} = e^{-(\kappa \cdot m)/(16 \cdot e)}\end{aligned}$$

aufgrund einer gebräuchlichen Chernoff-Ungleichung (siehe Lemma A.6). Folglich tritt dieses Ereignis mit Wahrscheinlichkeit mindestens $1 - \exp(-\Omega(\sqrt{n}))$ mindestens $((\kappa \cdot m)/(4 \cdot e))$-mal während der ersten $(\kappa \cdot m)/\tau_{\min}$ Iterationen der aktuellen Phase ein.

Wir beweisen nun, dass es einen Schwellenwert b gibt, sodass die aktuelle Ameise mit hoher Wahrscheinlichkeit eine verrauschte Länge von höchstens b wahrnimmt, wenn sie dem Pfad folgt, der ausschließlich aus oberen Kanten besteht. Andererseits beträgt die Wahrscheinlichkeit, eine verrauschte Länge von höchstens b wahrzunehmen, wenn die Ameise einem Pfad folgt, der mindestens einen unteren Pfad umfasst, höchstens $\exp(-\Omega(\sqrt{n}))$. Das beweist die erste und die dritte Aussage für die aktuelle Komponente und – nach einer zusätzlichen Wartezeit von T^* Iterationen – auch die zweite Aussage.

Das Rauschen auf jedem oberen Pfad ist gemäß $\Gamma(k, (1 + \epsilon) \cdot m \cdot \theta)$ verteilt und das Rauschen auf jedem unteren Pfad gemäß $\Gamma(m \cdot k, \theta)$. Die verrauschte Länge des Pfades, der aus jeder verbleibenden Komponente den oberen Pfad enthält, ist $(1 + \epsilon) \cdot m \cdot i + X_{i'}$ mit $X_{i'} \sim \Gamma(i \cdot k, (1 + \epsilon) \cdot m \cdot \theta)$ und die verrauschte Länge eines Pfades, der aus mindestens einer verbleibenden Komponente den unteren Pfad enthält, ist mindestens $i \cdot m + Y_{i''}$ mit $Y_{i''} \sim \Gamma(m \cdot k, \theta)$. Wir wählen

$$b := (1 + \epsilon) \cdot (m \cdot i + m \cdot \theta).$$

Zuallererst betrachten wir $\mathrm{W}((1+\epsilon) \cdot m \cdot i + \min\{X_{i'}\} \leq b)$ für $1 \leq i' \leq (\kappa \cdot m)/(4 \cdot e)$. Aufgrund von Lemma 10.1 und der Ungleichung $n! \leq n^n$ folgt

$$\begin{aligned}&\mathrm{W}(X_{i'} > b - (1 + \epsilon) \cdot m \cdot i) \\ &= 1 - \mathrm{W}(X_{i'} \leq (1 + \epsilon) \cdot m \cdot \theta) \\ &\leq 1 - \frac{(((1 + \epsilon) \cdot m \cdot \theta)/((1 + \epsilon) \cdot m \cdot \theta))^{i \cdot k}}{(i \cdot k)!} \cdot e^{-((1+\epsilon) \cdot m \cdot \theta)/((1+\epsilon) \cdot m \cdot \theta)} \\ &\leq 1 - \left(\frac{1}{i \cdot k}\right)^{i \cdot k} \cdot e^{-1}.\end{aligned}$$

Also gilt

$$\begin{aligned}\mathrm{W}(\epsilon \cdot m \cdot i + \min\{X_{i'}\} \leq b) &= 1 - \mathrm{W}(X_1 > b - \epsilon \cdot m \cdot i)^{(\kappa \cdot m)/(4 \cdot e)} \\ &\geq 1 - \left(1 - \left(\frac{1}{i \cdot k}\right)^{i \cdot k} \cdot e^{-1}\right)^{(\kappa \cdot m)/(4 \cdot e)} \\ &= 1 - \exp(-\Omega(\sqrt{n})).\end{aligned}$$

Betrachte $\mathrm{W}(m \cdot i + Y_{i''} > b)$. Aufgrund von Lemma 10.1 und $n! \geq (n/e)^n$ gilt

$$\begin{aligned}\mathrm{W}(m \cdot i + Y_{i'} \leq b) &= \mathrm{W}(Y_{i'} \leq \epsilon \cdot m \cdot i + (1+\epsilon) \cdot m \cdot \theta) \\ &\leq \frac{((\epsilon \cdot m \cdot i + (1+\epsilon) \cdot m \cdot \theta)/\theta)^{m \cdot k}}{(m \cdot k)!} \\ &\leq \frac{(\epsilon \cdot m \cdot i/\theta + (1+\epsilon) \cdot m)^{m \cdot k} \cdot e^{m \cdot k}}{(m \cdot k)^{m \cdot k}} \\ &\leq (\epsilon \cdot m \cdot e/(k \cdot \theta) + (1+\epsilon) \cdot e/k)^{m \cdot k} \\ &= \exp(-\Omega(\sqrt{n})),\end{aligned}$$

wobei die letzte Aussage aus den Voraussetzungen $k \geq 12$, $\epsilon \leq 1$ und $k \cdot \theta/\epsilon \geq 4 \cdot \sqrt{n}$ aufgrund von

$$(\epsilon \cdot m \cdot e)/(k \cdot \theta) + (1+\epsilon) \cdot e/k \leq e/(4 \cdot \sqrt{2}) + e/6 < 1$$

folgt. □

Das folgende Lemma zeigt eine untere Schranke für die Zeit, bis der linke Teil des Graphen optimiert ist. Dies wird auch eine untere Schranke für die Zeit liefern, bis eine gute Approximation des linken Teils des Graphen gefunden wird, und demzufolge auch des gesamten Graphen.

Lemma 10.7. *Betrachte den Graphen $G_{n,k,\theta,\epsilon} = (V, E, w)$ mit Rauschen $\eta(e) \sim \Gamma(k, \theta)$ für jede Kante $e \in E$. Sei $\epsilon \leq 1$. Wähle die Parameter $\tau_{\min} \leq 1/(2 \cdot n)$ und $1/\operatorname{poly}(n) \leq \rho \leq 1/2$. Dann findet das Ameisensystem $\mathrm{MMAS}_{\mathrm{SDSP}}$ mit Wahrscheinlichkeit $1 - \exp(-\Omega(\sqrt{n}/\log n))$ keine 2-Approximation innerhalb der erste $n/(6 \cdot \tau_{\min}) + \sqrt{n} \cdot \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen.*

Beweis. In Kapitel 9 wird die erwartete Optimierungszeit des $\mathrm{MMAS}_{\mathrm{SDSP}}$ genannten Ameisensystems auf einem Graphen, der sehr große Ähnlichkeit mit dem linken Teil von G aufweist, in einem Szenario ohne Rauschen durch $\Omega(n/\tau_{\min} + n/(\rho \cdot \log(1/\rho)))$ nach unten begrenzt. Wir stützen unsere Analyse teilweise auf diesen Beweis und konzentrieren uns auf den linken Teil von G. Eine gemeinsame Eigenschaft des Graphen aus Kapitel 9 ohne Rauschen und des linken Teils unseres Graphen mit Rauschen besteht darin, dass der kürzeste Pfad von u_0 nach $u_{n/2}$ durch den Teilgraphen offensichtlich alle unteren Kanten umfasst. Außerdem besteht der zweitbeste Pfad von u_i nach $u_{n/2}$ genau aus der Kante $(u_i, u_{n/2})$, da sich die Gewichte der oberen Kanten

von links nach rechts erhöhen. Folglich werden viele Ameisen dazu verleitet, Pheromone auf den oberen Kanten abzulegen, was es für die anderen Ameisen schwierig macht, der Kette von unteren Kanten zu folgen. Wir beweisen zunächst, dass die Kante $(u_i, u_{n/2})$ mit überwältigender Wahrscheinlichkeit die zweitbeste Wahl in dem verrauschten Szenario darstellt.

Beachte zuerst, dass aus $\eta(e) \sim \Gamma(k, \theta)$ aufgrund von Lemma 10.2 folgt, dass $\mathrm{W}(\eta(e) \geq k \cdot \theta \cdot n) \leq e^{-k \cdot n} \cdot (k \cdot n)^{k-1} \cdot k/(k-1)! \leq e^{-\Omega(n)}$ gilt. Wenn wir annehmen, dass sich dieses Ereignis für keine Kante während der betrachteten Zeitspanne von $n/(6 \cdot \tau_{\min}) + \sqrt{n} \cdot T^*$ mit $T^* = \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen ereignet, dann machen wir nur mit einer exponentiell kleinen Wahrscheinlichkeit einen Fehler. Demzufolge, gilt mit überwältigender Wahrscheinlichkeit für jeden Pfad p

$$w(p) \leq \tilde{w}(p) \leq (1 + k \cdot \theta \cdot n) \cdot w(p) = \beta \cdot w(p),$$

wobei $\beta := 1 + k \cdot \theta \cdot n$ ist. Für $0 \leq i \leq n/2 - 2$ und $0 \leq j \leq n/2 - i - 2$ bezeichne $p_{i,j}$ den Pfad, der am Knoten u_i beginnt und j untere Kanten umfasst bevor er entlang der oberen Kante nach $u_{n/2}$ führt, d. h.,

$$p_{i,j} := (u_i, u_{i+1}), \ldots, (u_{i+j-1}, u_{i+j}), (u_{i+j}, u_{n/2}).$$

Wir erinnern daran, dass die obere Kante $(u_j, u_{n/2})$ das Gewicht $W_j = 2 \cdot n \cdot (2 + 2 \cdot k \cdot \theta \cdot n)^{j+1} = 2 \cdot n \cdot (2 \cdot \beta)^{j+1}$ aufweist. Beachte außerdem, dass jeder Pfad von $u_{n/2}$ zum Zielknoten eine tatsächliche Länge zwischen $n/2$ und $(1 + \epsilon) \cdot n/2 \leq n$ besitzt, da $\epsilon \leq 1$ gilt. Für alle i und j' mit $j < j' \leq n/2 - i - 2$ und für alle Pfade p^* und p^{**} von $u_{n/2}$ zum Zielknoten gilt (wobei $\circ$ die Konkatenation von Pfaden bezeichnet)

$$\begin{aligned}\tilde{w}(p_{i,j} \circ p^*) &< \beta \cdot w(p_{i,j} \circ p^*) \\ &\leq \beta \cdot (j + 2 \cdot n \cdot (2 \cdot \beta)^{j+1} + n) \leq 2 \cdot n \cdot (2 \cdot \beta)^{j+2} \\ &\leq w(p_{i,j'} \circ p^{**}) \leq \tilde{w}(p_{i,j'} \circ p^{**}).\end{aligned}$$

Also wird eine Ameise den Pfad $p_{i,j}$ immer dem Pfad $p_{i,j'}$ vorziehen; insbesondere ist $p_{i,0}$ der zweitbeste Pfad.

Nachdem wir diese Beziehung sichergestellt haben, können wir einige Argumente aus dem Beweis von Theorem 9.11 wiederverwenden. Wir nennen einen Knoten *falsch*, wenn der zugehörige beste bisher gefundene Pfad mit der oberen Kante beginnt. Mithilfe einer Chernoff-Ungleichung folgern wir, dass es initial mit Wahrscheinlichkeit $1 - \exp(-\Omega(n))$ mindestens $(4/9) \cdot (n/2)$ falsche Knoten gibt. Beachte außerdem, dass für einen falschen Knoten die Wahrscheinlichkeit, die erste Kante auf dem eindeutigen kürzesten Pfad zu wählen, eingangs $1/2$ beträgt und im Laufe der Zeit gegen $\tau_{\min}$ strebt, solange der Knoten weiterhin falsch bleibt (siehe auch die obere Schranke für die Wahrscheinlichkeit aus der ersten Aussage aus Lemma 10.3). Nach T^* Iterationen wird die Pheromonschranke $\tau_{\min}$ erreicht.

Betrachte einen Knoten v mit mindestens $8 \cdot \log(1/\rho) + \kappa \cdot \sqrt{n}/\log n$ falschen Nachfolgern auf seinem kürzesten Pfad, wobei $\kappa > 0$ eine kleine Konstante ist. Um v zu optimieren, muss die Ameise, die auf v startet, die richtige Entscheidung an jedem

falschen Nachfolger treffen. Der Beweis von Theorem 9.11 aus Kapitel 9 zeigt, dass die Wahrscheinlichkeit, den Knoten v innerhalb von $1/\rho - 1 \geq 1/(2 \cdot \rho)$ Iterationen zu optimieren, gerade $\exp(-\Omega(\sqrt{n}/\log n))$ beträgt. Dies liegt daran, dass auch wenn Nachfolger von v optimiert werden, die Pheromone eine gewisse Zeit benötigen, um sich anzupassen (wir erinnern daran, dass die Wahrscheinlichkeit, eine richtige Kante zu wählen, höchstens $1/2$ beträgt). Indem wir die boolesche Ungleichung für alle Knoten v verwenden, schließen wir, dass innerhalb von $1/(2 \cdot \rho)$ Iterationen mit überwältigender Wahrscheinlichkeit nur $8 \cdot \log(1/\rho) + \kappa \cdot \sqrt{n}/\log n$ falsche Knoten korrigiert werden.

Nach $2 \cdot \sqrt{n} \cdot T^*$ Phasen, die jeweils $1/(2 \cdot \rho)$ Iterationen umfassen, wurden mit überwältigender Wahrscheinlichkeit höchstens

$$2 \cdot \sqrt{n} \cdot T^* \cdot (8 \cdot \log(1/\rho) + \kappa \cdot \sqrt{n}/\log n) \leq 1/18 \cdot n/2$$

falsche Knoten korrigiert, wobei die Ungleichung gilt, wenn κ hinreichend klein gewählt wird und n hinreichend groß ist. Anschließend befinden wir uns in einer Situation, in der immer noch $(4/9) \cdot (n/2) - (1/18) \cdot (n/2) = (7/18) \cdot (n/2)$ falsche Knoten übrig sind. Zudem ist an diesen Knoten die Wahrscheinlichkeit, die untere Kante zu wählen, auf $\tau_{\min}$ gefallen, da die Pheromone mehr als T^*-mal verringert worden sind. Nun gilt: wenn v mindestens i falsche Nachfolger auf seinem kürzesten Pfad hat, beträgt die Wahrscheinlichkeit, v in der nächsten Iteration zu optimieren, aufgrund der erste Aussage von Lemma 10.3 höchstens $(\tau_{\min})^i$.

Das folgende Argument basiert auf dem Beweis von Theorem 17 aus [Droste et al., 2002]. Wir stellen uns eine unbeschränkte Folge von Nullen und Einsen vor, wobei jedes Bit unabhängig mit Wahrscheinlichkeit $\tau_{\min}$ auf 1 gesetzt wird. Dann folgt die zufällige Anzahl von Einsen vor der ersten Null der gleichen geometrischen Verteilung wie die Anzahl von optimierten Knoten (tatsächlich dominiert die erste Verteilung die zweite stochastisch, da die Wahrscheinlichkeiten nicht genau $(\tau_{\min})^i$ entsprechend, sondern kleiner sind). Die Wahrscheinlichkeit, $(7/18) \cdot (n/2)$ falsche Knoten in $n/(6 \cdot \tau_{\min})$ Iterationen zu optimieren, wird demzufolge durch die Wahrscheinlichkeit, mindestens $(7/18) \cdot (n/2)$ Einsen in den ersten $n/(6 \cdot \tau_{\min}) + (7/18) \cdot (n/2)$ Bits der Bitfolge zu haben, begrenzt. Eine direkte Anwendung einer Chernoff-Ungleichung garantiert, dass diese Wahrscheinlichkeit durch $\exp(-\Omega(n))$ begrenzt ist.

Indem wir alle Fehlerwahrscheinlichkeiten aufsummieren, erhalten wir, dass mit Wahrscheinlichkeit $1 - \exp(-\Omega(\sqrt{n}/\log n))$ nach $n/(6 \cdot \tau_{\min}) + \sqrt{n} \cdot T^*$ Iterationen *nicht* alle Knoten optimiert sind. Da der zweitbeste Pfad von jedem Knoten im linken Teil eine Approximationsgüte von mindestens 2 aufweist, folgt die Behauptung. □

Beachte, dass wir sicherstellen können, dass in der genannten Zeitspanne eine beliebig schlechte Approximationsgüte *nicht* erreicht wird, indem wir alle Gewichte W_i mit einem hinreichend großen Faktor multiplizieren. Tatsächlich haben wir das folgende Theorem bewiesen, das festhält, dass wir nicht darauf hoffen können, eine beliebige, feste Approximationsgüte in weniger als $n/(6 \cdot \tau_{\min}) + \sqrt{n} \cdot \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen erreichen zu können.

Theorem 10.8. *Für jede Approximationsgüte $r \in \mathbb{R}^+$ gibt es einen Graphen mit n Knoten, sodass das Ameisensystem $MMAS_{SDSP}$ mit $1/\operatorname{poly}(n) \leq \rho \leq 1/2$ und $\tau_{\min} \leq 1/(2 \cdot n)$ mit überwältigender Wahrscheinlichkeit keine $(1+r)$-Approximation innerhalb der ersten $n/(6 \cdot \tau_{\min}) + \sqrt{n} \cdot \ln(\tau_{\max}/\tau_{\min})/\rho$ Iterationen findet.*

Die aus Kapitel 9 bekannten Ergebnisse für das Ameisensystem $\text{MMAS}_{\text{SDSP}}$ liefern, dass der Algorithmus tatsächlich alle kürzesten Pfade findet, wenn ihm ein bisschen mehr Zeit zur Verfügung steht: eine obere Schranke aus $\mathcal{O}(n/\tau_{\min} + n \cdot \ln(\tau_{\max}/\tau_{\min})/\rho)$ gilt in dem Szenario ohne Rauschen (siehe auch Kapitel 9). Für stochastische kürzeste Wege ist die Situation eine gänzlich andere. Indem wir die Aussagen von Lemma 10.6 und Lemma 10.7 für die unterschiedlichen Teile von G kombinieren, schließen wir, dass die Wartezeit, bis eine einigermaßen vernünftige Approximation gefunden wird, exponentiell groß ist. Die Bedingung $k \cdot \theta/\epsilon \geq 8 \cdot \sqrt{n}$ bedeutet beispielsweise, dass wir, wenn das erwartete Rauschen $k \cdot \theta$ konstant ist, eine $(1 + \mathcal{O}(1/\sqrt{n}))$-Approximation *nicht* effizient erreichen können.

Theorem 10.9. *Betrachte den Graphen $G_{n,k,\theta,2\cdot\epsilon} = (V, E, w)$ mit Rauschen $\eta(e) \sim \Gamma(k, \theta)$ für jede Kante $e \in E$. Seien $k \geq 12$, $\epsilon \leq 1/2$ und $k \cdot \theta/\epsilon \geq 8 \cdot \sqrt{n}$. Wähle die Parameter $1/\operatorname{poly}(n) \leq \tau_{\min} \leq 1/(\Delta \cdot n)$ und $1/\operatorname{poly}(n) \leq \rho \leq 1/2$. Dann findet das Ameisensystem $MMAS_{SDSP}$ mit Wahrscheinlichkeit $1 - \exp(-\Omega(\sqrt{n}/\log n))$ keine $(1+\epsilon)$-Approximation innerhalb der ersten $e^{c\cdot\sqrt{n}}$ Iterationen, wobei $c > 0$ eine hinreichend kleine Konstante ist.*

Beweis. Aufgrund von Lemma 10.7 findet das Ameisensystem keine $(1+\epsilon)$-Approximation innerhalb der ersten $n/(6 \cdot \tau_{\min}) + \sqrt{n} \cdot T^*$ Iterationen, wobei $T^* = \ln(\tau_{\max}/\tau_{\min})/\rho$ ist. Indem wir Lemma 10.6 mit $\epsilon' := 2 \cdot \epsilon$ benutzen, schließen wir, dass sich nach dieser Zeitspanne in alle Komponenten die Pheromone auf den oberen Kanten gesammelt haben. Die Wahrscheinlichkeit, dass sich ein bester bisher gefundener Pfad in einer einzelnen Iteration ändert, beträgt aufgrund der dritten Aussage von Lemma 10.6 höchstens $m \cdot \exp(-\Omega(\sqrt{n}))$. Wir erinnern daran, dass die oberen Kanten in $G_{n,k,\theta,2\epsilon}$ das Gewicht $(1 + 2 \cdot \epsilon) \cdot m$ aufweisen. Wir erinnern ferner daran, dass die Ameise, die in der ersten Komponente startet, nur eine $(1+\epsilon)$-Approximation konstruiert, wenn sie dem unteren Pfad in mindestens $m/2$ Komponenten folgt. Die Wahrscheinlichkeit für dieses Ereignis beträgt höchstens $(\tau_{\min})^{m/2} \cdot \binom{m}{m/2} \leq (4 \cdot \tau_{\min})^{m/2} = \exp(-\Omega(m))$, da $\tau_{\min} \leq 1/(2 \cdot n)$ gilt. Die Wahrscheinlichkeit, dass sich eines der beiden genannten Ereignisse innerhalb von $e^{c\cdot\sqrt{n}}$ Iterationen ereignet, beträgt nach wie vor $\exp(-\Omega(\sqrt{n}))$, wenn c hinreichend klein ist. □

10.6. Korreliertes Rauschen

In zahlreichen stochastischen Optimierungsszenarien ist das Rauschen nicht unabhängig sondern korreliert. In diesem Abschnitt betrachten wir ein Szenario, das den Gegenpol zu unabhängigem Rauschen darstellt. Wir nehmen an, dass das gleiche Rauschen für alle Kanten gilt, d. h., für jede Ameise gibt es einen einzelnen

Rauschwert, der auf die Länge des gesamten Pfades addiert wird. Die Leserinnen und Leser können die Ameisen beispielsweise als reisende Agenten sehen, die sich jeweils mit einem individuellen Tempo fortbewegen. Formal sind die η-Werte nach wie vor gamma-verteilt: $\eta(e_1), \dots, \eta(e_m) \sim \Gamma(k, \theta)$ für $k \in \mathbb{N}$ und $\theta \in \mathbb{R}^+$. Die Werte gleichen sich jedoch: $\eta(e_1) = \dots = \eta(e_m)$. Beachte, dass dies einer perfekten Korrelation der Verzögerungen entspricht. Die verrauschte Länge $\tilde{w}(p)$ eines Pfades $p = (e_1, \dots, e_\ell)$ gleicht dann

$$\tilde{w}(p) = \sum_{i=1}^{\ell} (1 + \eta(e_i)) \cdot w(e_i) = (1 + \eta(e_1)) \cdot w(p).$$

Wenn wir den Graphen G aus Definition 10.5 erneut betrachten, dann erwarten wir, dass stark korreliertes Rauschen hilfreich ist, da der Rauschwert für den unteren Pfad in einer Komponente dann eine vergleichbare Varianz wie der Rauschwert für den oberen Pfad zeigt. Dies ermöglicht den Ameisen letztlich, den unteren Pfad mit einem hinreichend kleinen Gesamtrauschen wahrzunehmen. Tatsächlich beweisen wir, dass die Ameisen kürzere von längeren Pfaden unterscheiden können und effizient kürzeste Wege finden, wenn das Rauschen perfekt korreliert ist. Das folgende Theorem liefert eine obere Schranke, die von den Rauschparametern und dem Wert von ϵ abhängt, der die Lücke zwischen den Gewichten auf den oberen und unteren Kanten in den Komponenten des Graphen G bestimmt.

Theorem 10.10. *Betrachte den Graphen $G_{n,k,\theta,\epsilon} = (V, E, w)$ mit Rauschen $\eta(e) \sim \Gamma(k, \theta)$ für jede Kante $e \in E$. Sei $\epsilon \leq \theta \cdot m$, wobei $m := \sqrt{n/2 - 1}$ ist. Wähle den Parameter $\tau_{\min} \leq 1/(\Delta \cdot n)$. Dann findet das Ameisensystem $MMAS_{SDSP}$ eine 1-Approximation im Erwartungswert nach höchstens $\mathcal{O}(n/\tau_{\min} \cdot (k \cdot \theta \cdot m/\epsilon)^k + \log(1/\tau_{\min})/\rho)$ Iterationen.*

Beweis. Wir betrachten die Ameisen von rechts nach links und begrenzen die erwartete Zeit, bis eine Ameise einem kürzesten Pfad folgt und diesen mit solch einem kleinen Rauschen wahrnimmt, dass die verrauschte Länge ihres besten bisher gefundenen Pfades kleiner als die tatsächliche Länge jedes suboptimalen Pfades ist. Betrachte die Ameise, die auf dem Knoten $v_{n/2-1-i\cdot m}$, $1 \leq i \leq m$, startet, und nimm an, dass alle Nachfolger das genannte Ziel bereits erreicht haben und dass die Pheromone auf den anderen Kanten, die diese Knoten verlassen, auf $\tau_{\min}$ abgefallen sind. Aufgrund von Lemma 10.3 folgt die Ameise dem optimalen Pfad mit Wahrscheinlichkeit mindestens $\tau_{\min}/(2 \cdot e)$. Die verrauschte Länge $\tilde{w}(p)$ eines Pfades p ist $(1 + X) \cdot i \cdot m$ mit $X \sim \Gamma(k, \theta)$, wenn er aus jeder verbleibenden Komponente den unteren Pfad enthält, und größer als $(i + \epsilon) \cdot m$, wenn er mit einer oberen Kante beginnt. Mithilfe von Lemma 10.1 schließen wir

$$\begin{aligned} \mathrm{W}((1 + X) \cdot i \cdot m \leq (i + \epsilon) \cdot m) &= \mathrm{W}(X \leq \epsilon/i) \\ &\geq \mathrm{W}(X \leq \epsilon/m) \geq \frac{(\epsilon/(\theta \cdot m))^k}{k!} \cdot e^{-\epsilon/(\theta \cdot m)} \geq \Big(\frac{\epsilon}{k \cdot \theta \cdot m}\Big)^k \cdot e^{-1}. \end{aligned}$$

Die erwartete Wartezeit beträgt $\mathcal{O}(1/\tau_{\min} \cdot (k \cdot \theta \cdot m/\epsilon)^k)$. Nach T^* weiteren Schritten gilt $\tau((v_{n/2-1-i\cdot m}, v_{n/2-1-(i-1)\cdot m})) = \tau_{\min}$ und die Ameise hat ihr Ziel erreicht.

Ein vergleichbares Argument gilt im linken Teil. Betrachte die Ameise, die auf dem Knoten $u_{n/2-1-i}$ mit $1 \leq i \leq n/2-1$ startet. Dann folgt unter den gleichen Annahmen wie zuvor, dass die Ameisen einen optimalen Pfad mit Wahrscheinlichkeit mindestens $\tau_{\min}/(2 \cdot e)$ entlangläuft. Die verrauschte Länge $\tilde{w}(p)$ eines Pfades p, der aus jeder verbleibenden Komponente den unteren Pfad enthält, ist genau $(1+X) \cdot (i+1+n/2-1) \leq (1+X) \cdot n$ mit $X \sim \Gamma(k, \theta)$ und die verrauschte Länge $\tilde{w}(p)$ eines Pfades p, der mit einer oberen Kante beginnt, ist größer als $2 \cdot n \cdot (1+k \cdot \theta \cdot n) + n/2 - 1$. Wir benutzen Lemma 10.2 und schließen

$$\begin{aligned} \mathrm{W}((1+X) \cdot n \leq 2 \cdot n \cdot (1 + k \cdot \theta \cdot n) + n/2 - 1) \\ \geq \mathrm{W}(X \leq k \cdot \theta \cdot n) \geq 1 - \exp(-\Omega(n)). \end{aligned}$$

Dies liefert eine erwartete Anzahl von $\mathcal{O}(1/\tau_{\min})$ Schritten. Nach T^* weiteren Schritten gilt $\tau((u_{n/2-1-i\cdot m}, u_{n/2})) = \tau_{\min}$. Das Aufsummieren der Wartezeiten vollendet den Beweis. □

Beachte, dass beispielsweise für $k = 12$ und $\theta/\epsilon = 2 \cdot \sqrt{n}/3$ Theorem 10.10 eine polynomielle obere Schranke liefert, während Theorem 10.9 eine exponentielle untere Schranke garantiert. Demzufolge ermöglicht korreliertes Rauschen den Ameisen in diesem Beispiel kürzeste Pfade finden zu können.

10.7. Fazit

Wir haben eine erste Analyse von Ameisenalgorithmen für ein stochastisches kombinatorisches Problem vorgestellt: Das Finden von kürzesten Wegen mit verrauschten Kantengewichten. Unsere allgemeine obere Schranke aus Theorem 10.4 gilt für beliebige, unabhängige Rauschverteilungen. Dieses Ergebnis kann aufgrund seiner Allgemeinheit nicht wesentlich verbessert werden. Für gamma-verteiltes Rauschen haben wir ein Szenario konstruiert, in dem die Ameisen aufgrund des Rauschens mit unterschiedlichen Varianzen für die oberen und unteren Pfade dazu gebracht werden, irrtümlicherweise den Pfad mit der größeren Varianz, der jedoch auch eine größere tatsächlichen Länge aufweist, zu bevorzugen. Die erwartete Zeit, bis eine gute Approximation gefunden wird, ist in diesem Fall exponentiell groß. Eine weitere Einsicht besteht darin, dass dieser Effekt verschwindet, wenn wir das unabhängige Rauschen durch perfekt korreliertes Rauschen ersetzen.

Nachfolgende Arbeiten könnten sich auf die Analyse von anderen Varianten von stochastischen Kürzeste-Wege-Problemen konzentrieren. Auch andere Rauschverteilungen könnten untersucht werden, besonders Verteilungen, die negative Werte annehmen können oder moderat korreliert sind. Außerdem ist die Frage, ob Ameisenalgorithmen eine vergleichbare Robustheit auch im Kontext von anderen stochastischen Problemen aufweisen, noch unbeantwortet.

11. Künstliche Immunsysteme

11.1. Einleitung

Der Begriff „Künstliche Immunsysteme“ (auf Englisch „Artificial Immune Systems“, AIS) bezeichnet ein relativ neues Forschungsgebiet, das bereits zahlreiche vielversprechende Ergebnisse für Probleme aus verschiedenen Anwendungsbereichen hervorgebracht hat; das Buch von de Castro und Timmis [2002] gibt hier einen guten Überblick. Künstliche Immunsysteme ahmen das biologische Abwehrsystem höherer Lebewesen nach, das dazu dient, Schädigungen des Gewebes durch Krankheitserreger zu verhindern. Sie basieren auf verschiedenen Theorien, die das Funktionieren der genannten Systeme erklären. Die bekanntesten Prinzipien werden mit den englischen Schlagwörtern „clonal selection“, „negative selection“, „immune networks“ und „danger theory“ verbunden. Wenngleich die Entdeckung von Anomalien und die Klassifikation von Objekten für künstliche Immunsysteme die natürlichsten Anwendungsgebiete darstellen, werden sie auch häufig eingesetzt, um Optimierungsprobleme zu lösen. Für die genannten Probleme gibt es eine Vielzahl von verschiedenen Varianten, die häufig auf dem „clonal selection“ genannten Prinzip beruhen [Cutello et al., 2004, de Castro und Von Zuben, 2002, Kelsey und Timmis, 2003].

Die bereits gelegten theoretischen Grundlagen widmen sich hauptsächlich dem Konvergenzverhalten von künstlichen Immunsystemen (siehe z. B. [Cutello et al., 2007b]). Die Entwicklung einer soliden theoretischen Basis für künstliche Immunsysteme wird als „große theoretische Herausforderung für die Zukunft“ angesehen [Timmis et al., 2008]. Wir fassen künstliche Immunsysteme – genau wie evolutionäre Algorithmen und Ameisensysteme – als randomisierte Suchheuristiken auf und konzentrieren uns auf die Optimierungszeit.

Die Analyse von künstlichen Immunsystemen ist kompliziert, da diese Systeme – genau wie evolutionäre Algorithmen – aus vielen Komponenten bestehen, die auf komplexe Weise ineinander greifen. Eine Möglichkeit, die genannten Systeme zu erforschen, besteht darin, die einzelnen Komponenten zunächst getrennt voneinander zu analysieren. Beispielsweise existieren erste theoretische Resultate für die Optimierungszeiten von Algorithmen, die Mutationsoperatoren aus künstlichen Immunsystemen einsetzen [Zarges, 2008, 2009, Jansen und Zarges, 2009a]. Wir werden uns in diesem Kapitel nicht auf Mutationsoperatoren sondern auf sogenannte Alterungsoperatoren konzentrieren, die sich seit einigen Jahren im Kontext von künstlichen Immunsystemen sehr großer Beliebtheit erfreuen [Castrogiovanni et al., 2007, Cutello et al., 2004, 2007a]. Darauf, wie die noch relativ neuen Arbeiten [Jansen und Zarges, 2009b, 2010c,a,b], die auch Alterungsoperatoren untersuchen, mit den Ergebnissen aus diesem Kapitel zusammenhängen, werden wir an den entsprechenden Stellen noch ein-

gehen.

Das Konzept des Älterwerdens spielt auch in anderen naturinspirierten Verfahren, besonders in evolutionären Algorithmen, eine wichtige Rolle. Die zugehörigen Operatoren verwenden eine *Alter* genannte Größe, die für jedes Individuum separat verwaltet wird. Diese wird herangezogen, um im Laufe des Optimierungsprozesses Entscheidungen zu treffen, die z. B. auf die Selektion oder die Mutation Einfluss haben können. Soweit wir wissen, haben Schwefel und Rudolph [1995] im Kontext von Evolutionsstrategien (ES) einen der ersten Versuche unternommen, das genannte Konzept mit Leben zu füllen. Außerdem gibt es verschiedene Ansätze, das Konzept in genetische Algorithmen (GA) [Ghosh et al., 1996, Kubota und Fukuda, 1997] und in evolutionäre Programmierung (EP) [Choi, 2002] zu integrieren.

In diesem Kapitel analysieren wir den Einfluss eines relativ neuen und bereits jetzt weitverbreiteten immuninspirierten Alterungsoperators im Kontext eines grundlegenden populationsbasierten evolutionären Algorithmus, wobei wir auf diese Weise eine Verbindung zwischen diesen Forschungsgebieten etablieren. Die Leistungsfähigkeit eines Algorithmus, der einen Alterungsoperator verwendet, kann sehr stark von der Lebensspanne τ der Individuen in der Population abhängen. Wir demonstrieren hier den Grad der Abhängigkeit, indem wir Familien von Beispielfunktionen identifizieren, für die sich ein Phasenübergang von polynomiellen zu exponentiellen Optimierungszeiten manifestiert, wenn man τ vergrößert oder verkleinert. Weiterhin konstruieren wir Beispielfunktionen und beweisen, dass die Optimierungszeit genau dann exponentiell ist, wenn die Lebensspanne τ außerhalb eines gegebenen Intervalls liegt, das zudem mithilfe eines Parameters der Beispielfunktionen beliebig transponiert werden kann. Außerdem präsentieren wir eine allgemeine Technik, die benutzt werden kann, um die Eigenschaften einer Beispielfunktion zu verstärken oder die Eigenschaften von verschiedenen Beispielfunktionen zu kombinieren.

Dieses Kapitel ist wie folgt aufgebaut. In dem nächsten Abschnitt beschreiben wir verschiedene Alterungsoperatoren und den Algorithmus, den wir anschließend untersuchen werden. In Abschnitt 11.3 zeigen wir Situationen auf, in denen eine große Lebensspanne benötigt wird, während wir in Abschnitt 11.4 eine Funktion konstruieren, für die die Lebensspanne nicht zu groß sein darf. In Abschnitt 11.5 präsentieren wir schließlich eine Technik, die es ermöglicht, die Eigenschaften von verschiedenen Beispielfunktion zu kombinieren, und wenden diese auf die zuvor beschriebenen Beispielfunktionen an. Wir beenden dieses Kapitel mit Vorschlägen für zukünftige Untersuchungen.

11.2. Der Alterungsoperator

In diesem Kapitel setzen wir das Konzept des Älterwerdens dazu ein, um den Selektionsprozess einer randomisierten Suchheuristik zu modifizieren. Der im Kontext von Evolutionsstrategien von Schwefel und Rudolph [1995] vorgestellte Alterungsoperator weist jedem neuen Individuum ein Alter zu, das zunächst auf 0 gesetzt wird. Ein Individuum, das das Alter $\tau + 1$ aufweist, wird während der Selektion zur Ersetzung

entfernt, wobei τ die Lebensspanne bezeichnet. Hier spielt die Fitness, die das Individuum zum genannten Zeitpunkt aufweist, keine Rolle. Die Lebensspanne τ gibt also an, wie viele Selektionen zur Ersetzung ein Individuum höchstens überstehen kann, bevor es aufgrund seines Alters herausgefiltert wird. Ein Individuum wird hier nach höchstens $\tau + 1$ Selektionen zur Ersetzung entfernt. Es wurde deutlich gemacht, dass dieser Ansatz einen Kompromiss zwischen den aus Evolutionsstrategien bekannten Selektionsvarianten – Plus- und Kommaselektion – darstellt. Die extremen Wahlen $\tau = \infty$ und $\tau = 0$ entsprechen dabei der Plus- bzw. der Kommaselektion.

In künstlichen Immunsystemen wird gewöhnlich eine leichte Variante dieses Konzeptes eingesetzt [Cutello et al., 2004, 2007a]. Auch hier hat jedes Individuum ein individuelles Alter und auch hier wird zusätzlich eine Lebensspanne τ vorgegeben. Im Gegensatz zur genannten Version erbt ein Nachkomme jedoch standardmäßig das Alter seines Elters und bekommt nur dann das Alter 0 zugewiesen, wenn seine Fitness echt größer als die Fitness seines Elters ist. Beachte, dass dieser Alterungsoperator versucht, jedem neuen Individuum, das besser als sein Elter ist, die gleiche Chance einzuräumen, seine Nachbarschaft zu erforschen und sich potenziell zu verbessern. Eine experimentelle Analyse dieses Operators wurde von Castrogiovanni et al. [2007] durchgeführt. Eine elitäre Variante dieses Alterungsoperators kann erreicht werden, indem man einem aktuell besten Individuum in der Population (unmittelbar vor der Selektion zur Ersetzung) das Alter 0 zuweist [Cutello et al., 2007b] oder alternativ einfach die Entfernung eines aktuell besten Individuums unterlässt und sein Alter wie zuvor fortführt. Außerdem gibt es eine stochastische Variante des Alterungsoperators, die ein Individuum x in Abhängigkeit von seinem Alter a_x mit Wahrscheinlichkeit $P_{\text{elim}}(a_x) := 1 - 2^{-1/a_x}$ entfernt [Cutello et al., 2007b].

Wir weisen darauf hin, dass die beiden genannten Varianten in [Jansen und Zarges, 2009b, 2010c] miteinander verglichen werden. In der Studie werden zwei Beispielfunktionen angegeben, für die sich die Optimierungszeiten der zugehörigen Algorithmen deutlich unterscheiden. Auch eine dritte Variante, die das Alter eines Nachkommen x' auf 0 setzt, wenn $f(x') \geq f(x)$ und $x' \neq x$ gelten, wobei x den Elter von x' bezeichnet, wird darin untersucht. Diese Variante führt auf beiden betrachteten Beispielfunktionen zu effizienten Optimierungszeiten.

Wir konzentrieren uns hier auf die statische, nicht elitäre Variante des immuninspirierten Alterungsoperators (auf Englisch auch unter „static pure aging“ bekannt) und betten diese in einen einfachen populationsbasierten evolutionären Algorithmus, genannt (μ+1)-EA, ein, der darauf ausgelegt ist, beliebige pseudoboolesche Fitnessfunktionen $f\colon \mathbb{B}^n \to \mathbb{R}$ zu maximieren [Witt, 2006]. Der Alterungsoperator wird auf alle in der Population befindlichen Individuen angewandt. Außerdem halten wir die Populationsgröße konstant, indem wir der Population neue, rein zufällig erzeugte Individuen hinzufügen, wenn die Anzahl der am Ende einer Generation überlebenden Individuen kleiner als μ ist. Der (μ+1)-EA mit Älterwerden wird in Algorithmus 13 definiert. Beachte, dass der bekannte (μ+1)-EA mit dem (μ+1)-EA mit Älterwerden übereinstimmt, wenn man die Lebensspanne τ auf ∞ setzt. Da wir den Algorithmus (μ+1)-EA im Folgenden auch untersuchen werden, definieren wir ihn separat in Algorithmus 14.

Algorithmus 13 (μ+1)-EA mit Älterwerden

Eingabe: Fitnessfunktion $f\colon S \to \mathbb{R}$ mit endlichem Suchraum S
Ausgabe: Suchpunkte $P \subseteq S$
Parameter: Populationsgröße $\mu \in \mathbb{N}^+$ und Lebensspanne $\tau \in (\mathbb{N} \cup \{\infty\})$

```
1: t ← 0
2: P_t = ∅
3: while |P_t| < μ do
4:     wähle z ∈ S rein zufällig, a_z ← 0, P_t ← (P_t ∪ {z})
5: end while
6: repeat
7:     t ← t + 1
8:     a_z ← a_z + 1 für alle z ∈ P_{t−1}
9:     wähle x ∈ P_{t−1} rein zufällig
10:    erzeuge x' aus x durch Mutation
11:    if f(x') > f(x) then a_{x'} ← 0 else a_{x'} ← a_x end if
12:    P_t ← {z ∈ (P_{t−1} ∪ {x'}) | a_z ≤ τ}
13:    if |P_t| > μ then
14:        wähle z ∈ argmin_{z'∈P_t} f(z') rein zufällig, P_t ← (P_t \ {z})
15:    else
16:        while |P_t| < μ do
17:            wähle z ∈ S rein zufällig, a_z ← 0, P_t ← (P_t ∪ {z})
18:        end while
19:    end if
20: until Abbruchkriterium erfüllt
21: return P_t
```

11.3. Große Lebensspanne

Die Schwierigkeit einer Fitnessfunktion kann für verschiedene (randomisierte) Suchheuristiken sehr unterschiedlich sein. Das gilt nicht nur im Hinblick auf Optimierungszeiten sondern auch im Hinblick auf lokale Fortschrittsmaße. Wenn ein Alterungsoperator eingesetzt wird, stellt die benötigte Zeit, um einen Nachkommen mit einem echt größeren Fitnesswert zu erschaffen, ein wichtiges lokales Fortschrittsmaß dar. Ausschließlich in diesem Fall wird das Alter des Nachkommen auf 0 gesetzt, anderenfalls erbt der Nachkomme das Alter seines Elters. Es leuchtet intuitiv ein, dass eine randomisierte Suchheuristik mit dem hier betrachteten Alterungsoperator Schwierigkeiten hat, das Optimierungsziel zu erreichen, wenn die Lebensspanne τ kleiner als die Zeit ist, die für solch einen lokalen Fortschritt benötigt wird. Der Suchheuristik steht einfach nicht die Zeit zur Verfügung, die benötigt wird, um einen besseren Nachkommen zu konstruieren. Folglich werden die Eltern während der betrachteten Zeitspanne durch neue, zufällige Individuen ersetzt. Man könnte dem Glauben anheimfallen, dass eine randomisierte Suchheuristik mit dem hier betrachteten Alterungsoperator und einer zu kleinen Lebensspanne τ chancenlos ist, wenn

Algorithmus 14 (μ+1)-EA

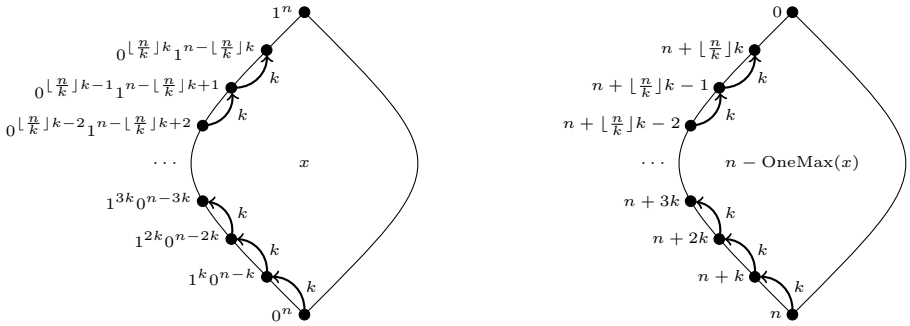
Eingabe: Fitnessfunktion $f\colon S \to \mathbb{R}$ mit endlichem Suchraum S
Ausgabe: Suchpunkte $P \subseteq S$
Parameter: Populationsgröße $\mu \in \mathbb{N}^+$

```
1: t ← 0
2: P_t = ∅
3: while |P_t| < μ do
4:     wähle z ∈ S rein zufällig, P_t ← (P_t ∪ {z})
5: end while
6: repeat
7:     t ← t + 1
8:     wähle x ∈ P_{t−1} rein zufällig
9:     erzeuge x' aus x durch Mutation
10:    P_t ← (P_{t−1} ∪ {x'})
11:    wähle z ∈ argmin_{z'∈P_t} f(z') rein zufällig, P_t ← (P_t \ {z})
12: until Abbruchkriterium erfüllt
13: return P_t
```

das geschilderte Ereignis in jedem Lauf irgendwann fast sicher eintritt. Die Dinge sind jedoch weniger einfach.

Betrachte eine weitestgehend leicht zu optimierende Funktion, die bereits einfache Suchheuristiken auf einem Pfad zum globalen Optimum geleitet. Die einzige Schwierigkeit besteht darin, dass der Pfad an einer Stelle unterbrochen ist. Um die „Schlucht“ zu überwinden und den „Gipfel“ zu erreichen, müssen in einem Suchpunkt gleichzeitig k feste Bits gekippt werden, wobei $k > 1$ eine ganzzahlige Konstante ist. Die Wahrscheinlichkeit für solch eine Mutation entspricht $(1/n)^k \cdot (1 - 1/n)^{n-k} < 1/n^k$. Die Wartezeit ist geometrisch verteilt und demzufolge größer als n^k. Wenn wir eine relativ kleine Lebensspanne τ betrachten, z. B. $\tau = n$, dann ist es sehr unwahrscheinlich, dass ein Individuum hinreichend lange überlebt, um einen Nachkommen zu erzeugen, der die Schlucht „überspringt“. Dieser Umstand stellt jedoch möglicherweise kein Problem dar, wenn die durchschnittliche Zeit $\mathcal{O}(t)$, um in die geschilderte Situation zu gelangen, kurz ist. Ein Individuum, das die Schlucht erreicht hat, liefert mit Wahrscheinlichkeit $\Theta(1/n^k)$ einen Nachkommen auf der anderen Seite der Schlucht, wenn es zur Mutation ausgewählt wird. Im Erwartungswert müssen also $\Theta(n^k)$ solche Individuen zur Mutation ausgewählt werden, damit einem Individuum der Sprung über die Schlucht gelingt. Wenn wir an dieser Stelle $\mu = \Theta(1)$ annehmen, folgt, dass es der Algorithmus im Erwartungswert nach $\mathcal{O}(t \cdot n^k)$ Schritten schafft, die Schlucht zu überwinden, auch wenn die Lebensspanne viel zu klein ist.

Eine Beispielfunktion, die zeigt, dass es verheerend sein kann, die Lebensspanne zu klein zu wählen, muss also etwas komplizierter als die im letzten Absatz beschriebene Funktion sein. Wir kombinieren die bekannte Beispielfunktion Ridge [Quick et al., 1998] mit genau den Schluchten, die wir weiter oben beschrieben haben. Der Haupt-



(a) Suchraum $\mathbb{B}^n$.

(b) Zielraum $\mathrm{f}_k(\mathbb{B}^n)$.

Abbildung 11.1.: Suchraum und Zielraum der Beispielfunktion f_k.

unterschied besteht darin, dass wir eine große Zahl von solchen Schluchten einführen, die alle sequenziell ohne einen einzigen Fehler übersprungen werden müssen, um die Funktion zu optimieren (siehe Abbildung 11.1). Wir können beweisen, dass die Wahrscheinlichkeit für dieses Ereignis verschwindend gering ist, wenn die Lebensspanne τ zu klein ist. Wenn die Lebensspanne τ hinreichend groß ist, dann fällt das Überspringen dieser Schluchten jedoch relativ leicht. Wir präzisieren diese Ideen nun und beweisen, dass sie richtig sind. Der Vollständigkeit halber geben wir zunächst die Definition von Ridge an.

Definition 11.1. *Sei $n \in \mathbb{N}^+$. Die Funktion* $\mathrm{Ridge}\colon \mathbb{B}^n \to \mathbb{N}^+$ *ist für alle $x \in \mathbb{B}^n$ durch*

$$\mathrm{Ridge}(x) := \begin{cases} n+i & \text{falls } x = 1^i 0^{n-i},\ i \in \{0,1,\ldots,n\} \\ n - \mathrm{OneMax}(x) & \text{anderenfalls} \end{cases}$$

definiert.

Definition 11.2. *Seien $n \in \mathbb{N}^+$ und $k \in \mathbb{N}^+$. Die Funktion* $\mathrm{f}_k\colon \mathbb{B}^n \to \mathbb{N}$ *ist für alle $x \in \mathbb{B}^n$ durch*

$$\mathrm{f}_k(x) := \begin{cases} n-i & \text{falls } x = 1^i 0^{n-i},\ i \in \{0,1,\ldots,n\} \text{ und } i/k \notin \mathbb{N} \\ \mathrm{Ridge}(x) & \text{anderenfalls} \end{cases}$$

definiert.

Beachte, dass $\mathrm{f}_1 = \mathrm{Ridge}$ gilt. Außerdem gelten die Zusammenhänge $\mathrm{f}_n = \mathrm{Trap}$ sowie $\mathrm{f}_k(x) = \mathrm{OneMax}(\overline{x})$ für alle $k > n$ und für alle $x \in \mathbb{B}^n$. Wir überzeugen uns zunächst davon, dass f_k nicht zu schwierig zu optimieren ist. Wir betrachten einen gewöhnlichen $(\mu{+}1)$-EA sowie einen $(\mu{+}1)$-EA mit Älterwerden, wobei die Lebensspanne τ hinreichend groß ist. Sei $T(A, f)$ die zufällige Anzahl von Generationen,

bis der Algorithmus A erstmalig ein globales Optimum der Funktion f anfragt. Beachte, dass die Anzahl von Generationen kleiner als die Anzahl von Funktionsauswertungen, d. h., die Optimierungszeit, sein kann, da die Bestimmung der Fitness der Individuen in der initialen Population μ Funktionsauswertungen erfordert. Außerdem können in einer Generation mehr als eine Funktionsauswertung erforderlich sein, da die neuen, zufälligen Individuen, die der Alterungsoperator potenziell hinzufügt (siehe Zeilen 16–18 in Algorithmus 13), auch ausgewertet werden müssen. Da ein beliebiges Individuum aufgrund seines Alters nach $\tau + 1$ Generationen aus der aktuellen Population entfernt wird, kann die Anzahl von Funktionsauswertungen durch $\mu + (1 + \mu/(\tau + 1)) \cdot T(A, f) < \mu + (1 + \mu/\tau) \cdot T(A, f)$ nach oben beschränkt werden. Dieses Verhalten kann beispielsweise beobachtet werden, wenn eine Funktion mit einem großen Plateau, wie z. B. Needle, optimiert wird.

Theorem 11.3. *Seien $n \in \mathbb{N}^+$ und $k \in \mathbb{N}^+$. Dann gelten für jede Populationsgröße μ und für jede Lebensspanne $\tau = \omega((n^k + \mu \cdot \log n) \cdot \log n)$*

$$\mathrm{E}(T(\textit{(}\mu\textit{+1)-EA, } \mathrm{f}_k)) = \mathcal{O}(n^{k+1} + \mu \cdot n \cdot \log n),$$
$$\mathrm{E}(T(\textit{(}\mu\textit{+1)-EA mit Älterwerden, } \mathrm{f}_k)) = \mathcal{O}(n^{k+1} + \mu \cdot n \cdot \log n).$$

Beweis. Wir partitionieren den Suchraum $\mathbb{B}^n$ zunächst in $n + \lfloor n/k \rfloor + 1$ disjunkte Mengen, die die Struktur von f_k widerspiegeln. Seien $B_i := \{1^{i \cdot k} 0^{n - i \cdot k}\}$ für alle $0 \leq i \leq \lfloor n/k \rfloor$ und $A_i := \{x \in \mathbb{B}^n \setminus (B_0 \cup \cdots \cup B_{\lfloor n/k \rfloor}) \mid \mathrm{OneMax}(x) = n - i\}$ für alle $0 \leq i < n$. Es ist einfach, einzusehen, dass A_i, $0 \leq i < n$, und B_i, $0 \leq i \leq \lfloor n/k \rfloor$, eine f_k-basierte Partition

$$A_0 <_{\mathrm{f}_k} \cdots <_{\mathrm{f}_k} A_{n-1} <_{\mathrm{f}_k} B_0 <_{\mathrm{f}_k} \cdots <_{\mathrm{f}_k} B_{\lfloor n/k \rfloor}$$

bilden [Droste et al., 2002], wobei $X <_{\mathrm{f}_k} Y$ für alle $X, Y \subseteq \mathbb{B}^n$ durch

$$\forall x \in X \colon \forall y \in Y \colon \mathrm{f}_k(x) < \mathrm{f}_k(y)$$

definiert ist.

Wir konzentrieren uns zuerst auf den Algorithmus (μ+1)-EA. Wir betrachten den Fall, dass die Population ein Individuum aus A_i enthält und begrenzen die erwartete Zeit, bis die Population ein besseres Individuum, d. h., ein Individuum aus $A_{i+1} \cup \cdots \cup A_{n-1} \cup B_0 \cup \cdots \cup B_{\lfloor n/k \rfloor}$, enthält. Wenn die Population genau j Individuen aus A_i enthält, dann beträgt die Wahrscheinlichkeit, dass eines dieser Individuen zur Mutation ausgewählt wird, genau j/μ. Die Wahrscheinlichkeit, dass sich das ausgewählte Individuum anschließend dupliziert, beträgt mindestens $(1 - 1/n)^n \geq 1/3$. Sei $b := \min\{\lceil n/(n-i) \rceil, \mu\}$. Insgesamt dauert es im Erwartungswert höchstens

$$\sum_{j=1}^{b-1} 3 \cdot \mu/j = 3 \cdot \mu \cdot (\ln(b-1) + 1)$$

Generationen, bis die Population mindestens b Individuen aus A_i oder ein besseres Individuum enthält. Die Wahrscheinlichkeit, ein besseres Individuum zu erzeugen,

beträgt anschließend mindestens $(b/\mu)\cdot((n-i)/n)\cdot(1-1/n)^i \geq (b\cdot(n-i))/(e\cdot\mu\cdot n)$, da es genügt, genau ein 1-Bit zu kippen. Also dauert es im Erwartungswert höchstens

$$3\cdot\mu\cdot(\ln(b-1)+1)+\frac{e\cdot\mu\cdot n}{b\cdot(n-i)}$$

Generationen, bis ein besseres Individuum gefunden wird. Insgesamt haben wir die obere Schranke

$$\begin{aligned}
&\sum_{i=0}^{n-1}\left[3\cdot\mu\cdot(\ln(b-1)+1)+\frac{e\cdot\mu\cdot n}{b\cdot(n-i)}\right]\\
&\leq \sum_{i=0}^{n-1}\left[3\cdot\mu\cdot\left(\ln\left(\frac{n}{n-i}\right)+1\right)+\frac{e\cdot\mu\cdot n}{\lceil n/(n-i)\rceil\cdot(n-i)}+\frac{e\cdot\mu\cdot n}{\mu\cdot(n-i)}\right]\\
&\leq 3\cdot\mu\cdot\ln\left(\prod_{i=0}^{n-1}\frac{n}{n-i}\right)+3\cdot\mu\cdot n+e\cdot\mu\cdot n+\sum_{i=0}^{n-1}\frac{e\cdot n}{(n-i)}\\
&\leq 3\cdot\mu\cdot\ln\left(\frac{n^n\cdot e^n}{n^n}\right)+3\cdot\mu\cdot n+e\cdot\mu\cdot n+e\cdot n\cdot(\ln(n)+1)\\
&= \mathcal{O}(\mu\cdot n+n\cdot\log n)
\end{aligned}$$

für die Zeit hergeleitet, bis die Population ein Individuum aus $B_0\cup\cdots\cup B_{\lfloor n/k\rfloor}$ enthält. Beachte, dass die letzte Ungleichung aus einer Anwendung der Stirling-Formel resultiert (siehe Lemma A.10). Wenn die Population ein Individuum aus B_i enthält, dann folgt auf analoge Weise, dass die erwartete Zeit, bis ein Individuum aus $B_{i+1}\cup\cdots\cup B_{\lfloor n/k\rfloor}$ erzeugt wird, für alle $b'\in\{1,\ldots,\mu\}$ höchstens

$$3\cdot\mu\cdot(\ln(b'-1)+1)+\frac{e\cdot\mu\cdot n^k}{b'}$$

Generationen beträgt, da in diesem Fall genau k bestimmte 0-Bits gekippt werden müssen. Wenn wir z. B. $b':=\min\{\lceil n^k\rceil,\mu\}$ wählen, dauert es im Erwartungswert höchstens $\mathcal{O}(\mu\cdot n\cdot\log n+n^{k+1})$ Generationen, bis das Optimierungsziel erreicht wird. Insgesamt wird die Optimierungszeit des Algorithmus (μ+1)-EA demzufolge von der zweiten Phase dominiert und beträgt höchstens $\mathcal{O}(\mu\cdot n\cdot\log n+n^{k+1})$. (Die hier benutzten Argumente werden auch nach Theorem 4.1 in Kapitel 4 thematisiert.)

Wir wenden uns nun dem Algorithmus (μ+1)-EA mit Älterwerden zu. Wir haben im ersten Teil dieses Beweises gesehen, dass der Algorithmus (μ+1)-EA das Optimierungsziel im Erwartungswert nach $c\cdot(\mu\cdot n\cdot\log n+n^{k+1})$ Generationen erreicht, wobei $c>0$ eine geeignete Konstante ist. Mithilfe der Markoff-Ungleichung folgt, dass das Optimierungsziel in $2\cdot c\cdot(\mu\cdot n\cdot\log n+n^{k+1})$ Generationen mit Wahrscheinlichkeit mindestens 1/2 erreicht wird. Der Beweis überträgt sich auf den (μ+1)-EA mit Älterwerden, wenn wir zusätzlich zeigen, dass in der betrachteten Phase niemals ein bestes Individuum aufgrund seines fortgeschrittenen Alters aus der Population entfernt wird. Für die Lebensspanne gilt nach Voraussetzung $\tau=\alpha(n)\cdot(n^k+\mu\cdot\log n)\cdot\log n$,

wobei α eine geeignete Funktion mit $\lim_{n\to\infty} \alpha(n) = \infty$ ist. Da das Alter des besten Individuums nach einer Verbesserung auf 0 gesetzt wird, müssen wir zeigen, dass die $n + \lfloor n/k \rfloor \leq 2 \cdot n$ erforderlichen Verbesserungen jeweils nach höchstens $\tau + 1$ Generationen erreicht werden. Wie wir bereits gesehen haben, wird eine Verbesserung im Erwartungswert nach $c' \cdot (\mu \cdot \log n + n^k)$ Generationen erreicht, wobei $c' > 0$ eine geeignete Konstante ist. Demzufolge wird eine Verbesserung mit Wahrscheinlichkeit mindestens $1/2$ nach höchstens $2 \cdot c' \cdot (\mu \cdot \log n + n^k)$ Generationen und mit Wahrscheinlichkeit mindestens $1 - (1 - 1/2)^{\log n} = 1 - 1/n$ nach höchstens $2 \cdot c' \cdot (\mu \cdot \log n + n^k) \cdot \log n$ Generationen erreicht. Die Wahrscheinlichkeit, dass dies nach höchstens $\tau + 1$ Generationen passiert, beträgt demzufolge mindestens $1 - (1 - (1 - 1/n))^{\alpha(n)/(2 \cdot c')} = 1 - (1/n)^{\alpha'(n)}$ für eine weitere geeignete Funktion α' mit $\lim_{n\to\infty} \alpha'(n) = \infty$. Wir schließen mithilfe der booleschen Ungleichung, dass alle $2 \cdot n$ Verbesserungen jeweils nach höchstens $\tau + 1$ Generationen mit Wahrscheinlichkeit mindestens $1 - 2 \cdot n \cdot (1/n)^{\alpha(n)} = 1 - (1/n)^{\alpha''(n)}$ erfolgen, wobei α'' erneut eine geeignete Funktion mit $\lim_{n\to\infty} \alpha''(n) = \infty$ ist. Insgesamt folgt die Aussage für den Algorithmus (μ+1)-EA mit Älterwerden, da wir im Erwartungswert höchstens $\mathcal{O}(1)$ Phasen betrachten müssen, bis das Optimierungsziel erreicht wird. □

Beachte, dass das letzte Theorem verglichen mit dem korrespondierenden Theorem aus [Horoba et al., 2009] für alle Populationsgrößen μ und nicht nur für $\mu = n^{\mathcal{O}(1)}$ gilt, wobei superpolynomielle Populationsgrößen natürlich für große Eingaben nicht zu empfehlen sind. Auch der Voraussetzung $k = \mathcal{O}(1)$ konnten wir uns entledigen.

Nachdem wir gezeigt haben, dass es sich bei f_k um eine Funktion von angemessenem Schwierigkeitsgrad handelt, wenden wir uns nun unserem eigentlichen Ziel zu. Wir beweisen, dass der Algorithmus (μ+1)-EA mit Älterwerden eine hinreichend große Lebensspanne benötigt, um die Funktion f_k erfolgreich optimieren zu können, wenn sich die Populationsgröße innerhalb sinnvoller Grenzen bewegt. Die obere Schranke aus Theorem 11.3 wird von dem Term dominiert, der die Populationsgröße enthält, sobald $\mu = \omega(n^k / \log n)$ gilt. Wir werden unsere Betrachtungen daher auf kleinere Populationsgrößen einschränken.

Wir weisen an dieser Stelle erneut darauf hin, dass es für eine aussagekräftige untere Schranke für die Optimierungszeit nicht ausreicht, eine große untere Schranke für die erwartete Optimierungszeit $\mathrm{E}(T((\mu{+}1)\text{-EA mit Älterwerden, } \mathrm{f}_k))$ zu beweisen. Große Erwartungswerte können in die Irre führen, wenn ihre Größe auf unwahrscheinlichen Ereignissen basiert, die zu äußerst langen Läufen führen. Es ist daher sehr viel informativer, eine untere Schranke für die Wahrscheinlichkeit anzugeben, dass ein Lauf lange dauert. Das ist genau die Art von Aussage, die das folgende Theorem macht.

Theorem 11.4. *Seien $n \in \mathbb{N}^+$ und $k \in \mathbb{N}^+$ mit $k = \mathcal{O}(1)$. Dann gilt für jede Populationsgröße $\mu = \mathcal{O}(n^k / \log n)$, für jede Lebensspanne $\tau = \mathrm{o}(n^k \cdot \log n)$ und für jede Konstante $0 < \epsilon < 1$*

$$\mathrm{W}(T((\mu{+}1)\text{-EA mit Älterwerden, } \mathrm{f}_k) = 2^{\Omega(n^\epsilon)}) = 1 - 2^{-\Omega(n^{1-\mathrm{o}(1)})}.$$

Beweis. Wir betrachten $2^{\Omega(n^\epsilon)}$ Generationen. Wenn ein Individuum $x \in \mathbb{B}^n$ rein zufällig gezogen wird, dann gilt $\text{OneMax}(x) \leq (2/3) \cdot n$ mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$ (aufgrund einer Chernoff-Ungleichung, siehe Lemma A.6). Also gilt dies für alle Individuen, die in den betrachteten Generationen rein zufällig gezogen werden, mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$ (aufgrund der booleschen Ungleichung, siehe Lemma A.4).

Wir nehmen nun an, dass $\text{OneMax}(x) \leq (2/3) \cdot n$ für alle Individuen in der aktuellen Population gilt. Um das globale Optimum $1^{\lfloor n/k \rfloor \cdot k} 0^{n - \lfloor n/k \rfloor \cdot k}$ zu erreichen, müssen mindestens $\lfloor n/k \rfloor - \lceil (2 \cdot n)/(3 \cdot k) \rceil$ Schluchten, die jeweils k Bits breit sind, übersprungen werden. Also müssen $c \cdot (n/k)$ Schluchten übersprungen werden, wobei $c > 0$ eine geeignete Konstante ist. Die Wahrscheinlichkeit, eine Schlucht zu überspringen, beträgt höchstens $1/n^k$. Wenn in einem Mutationsschritt mindestens eine Schlucht übersprungen wird, können wir die Anzahl von übersprungenen Schluchten durch eine geometrisch verteilte Zufallsvariable X_i mit $\text{W}(X_i = j) = (1-p)^{j-1} \cdot p$, $j \in \mathbb{N}^+$, und $p = (1 - 1/n^k)$ modellieren. Dann gilt $\text{E}(X) = 1/(1 - 1/n^k)$. Die Wahrscheinlichkeit, dass nach $t := (c \cdot n \cdot (1 - 1/n^k))/(2 \cdot k) \geq (c \cdot n)/(4 \cdot k) = \Omega(n)$ Mutationsschritten, in denen jeweils mindestens eine Schlucht übersprungen wird, das globale Optimum gefunden ist, beträgt folglich höchstens

$$\text{W}(X \geq c \cdot (n/k)) = \text{W}(X \geq (1+1) \cdot \text{E}(X)) \leq \exp\left(\frac{c \cdot n \cdot (1 - 1/n^k)}{8 \cdot k} \right) = 2^{-\Omega(n)},$$

wobei $X = \sum_{i=1}^{t} X_i$ ist (siehe Lemma 7.7). Auf dem Weg zum globalen Optimum befinden wir uns daher mit überwältigender Wahrscheinlichkeit t-mal in einer Situation, in der die Population genau ein bestes Individuum $x = 1^{i \cdot k} 0^{n - i \cdot k}$ mit $\text{OneMax}(x) > (2/3) \cdot n$ und Alter 0 enthält.

Sei $\tau = \text{o}(n^k \cdot \log n)$. Dann gilt $\tau \leq \alpha(n) \cdot n^k \cdot \log n$ für eine geeignete Funktion α mit $\lim_{n \to \infty} \alpha(n) = 0$. Wir betrachten in einer der oben genannten Situationen eine Phase der Länge $\tau + 1 \leq \alpha(n) \cdot n^k \cdot \log n + 1$. Die Wahrscheinlichkeit, dass in der betrachteten Phase kein Individuum eine Schlucht überspringt, beträgt mindestens

$$\left(1 - \frac{1}{n^k}\right)^{\alpha(n) \cdot n^k \cdot \log n + 1} \geq e^{-(\alpha(n) \cdot n^k \cdot \log n + 1)/(n^k - 1)} \geq e^{-4 \cdot \alpha(n) \cdot \log n} = \frac{1}{n^{4 \cdot \alpha(n)}}.$$

Wenn dieser Fall eintritt, werden alle im letzten Absatz genannten Individuen durch rein zufällig gezogene Individuen ersetzt und wir befinden uns in der Ausgangssituation. Die Wahrscheinlichkeit, dass dies in einer der t Situationen passiert, beträgt mindestens

$$\begin{aligned} 1 - \left(1 - \frac{1}{n^{4 \cdot \alpha(n)}}\right)^t &\geq 1 - \left(1 - \frac{1}{n^{4 \cdot \alpha(n)}}\right)^{(c \cdot n)/(4 \cdot k)} \\ &\geq 1 - e^{-(c \cdot n)/(4 \cdot k \cdot n^{4 \cdot \alpha(n)})} \\ &= 1 - 2^{-\Omega(n^{1 - \text{o}(1)})}. \end{aligned}$$

Mithilfe der booleschen Ungleichung schließen wir, dass das globale Optimum in $2^{\Omega(n^\epsilon)}$ Generationen nicht gefunden wird. Insgesamt folgt das Theorem. □

11.4. Kleine Lebensspanne

Im vorangegangenen Abschnitt haben wir Situationen aufgezeigt, in denen eine große Lebensspanne vonnöten ist. Andererseits gibt es Situationen, in denen eine kleine Lebensspanne den betrachteten Algorithmus davor bewahren kann, viel Zeit in Bereichen des Suchraumes zu verschwenden, die weit vom globalen Optimum entfernt sind. Wir vervollständigen das Gesamtbild, indem wir nun eine Beispielfunktion konstruieren, die nur effizient optimiert werden kann, wenn die Lebensspanne nicht zu groß ist.

Unsere Beispielfunktion besteht im Wesentlichen aus einem Pfad, ähnlich dem Pfad in der Beispielfunktion Ridge, der hier jedoch zu einem lokalen Optimum führt. Das globale Optimum befindet sich in einer gewissen Entfernung von diesem Pfad. Demzufolge ist es wahrscheinlicher, dem Pfad zu folgen, als zum globalen Optimum zu springen. Der Pfad besteht aus zwei Teilpfaden, die durch zwei Schluchten getrennt sind. Der erste Teilpfad umfasst polynomiell viele Suchpunkte während der zweite exponentiell viele umfasst. Die Idee besteht darin, dass die Individuen, die den zweiten Teilpfad, der sich vom globalen Optimum entfernt, erreicht haben, auf diesem exponentiell lange gefangen sind.

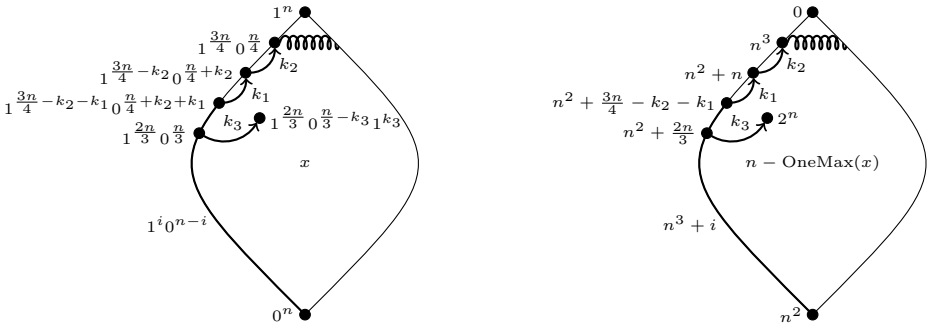
Um dieses Ziel zu erreichen, machen wir uns einen sogenannten langen k-Pfad [Horn et al., 1994, Rudolph, 1996] zunutze, d. h., eine sehr lange Folge von verschiedenen Hammingnachbarn mit wachsenden Fitnesswerten, die sich durch den booleschen Hyperwürfel zieht. Wir übernehmen die Notation von langen k-Pfaden so wie sie von Sudholt [2008] präsentiert wird.

Definition 11.5 (Langer k-Pfad)**.** *Seien $n \in \mathbb{N}$ und $k \in \mathbb{N}^+$ mit $n/k \in \mathbb{N}$. Der lange k-Pfad P_k^n der Dimension n ist eine Folge von Punkten aus $\mathbb{B}^n$, die rekursiv definiert ist. Seien $P_k^0 := ()$, die leere Bitfolge, und $P_k^{n-k} = (p_1, \ldots, p_\ell)$. Dann ist P_k^n die Konkatenation von S_0, B und S_1, d. h., $P_k^n = (S_0, B, S_1)$, wobei*

- $S_0 = (0^k p_1, 0^k p_2, \ldots, 0^k p_\ell)$,
- $B = (0^{k-1}1 p_\ell, 0^{k-2}1^2 p_\ell, \ldots, 01^{k-1} p_\ell)$ *und*
- $S_1 = (1^k p_\ell, 1^k p_{\ell-1}, \ldots, 1^k p_1)$

sind.

Beachte, dass sich die Suchpunkte aus S_0 und S_1 in den führenden k Bits unterscheiden und dass die Suchpunkte aus B eine „Brücke“ zwischen S_0 und S_1 bilden. Folglich sind alle Suchpunkte paarweise verschieden. Eine wichtige Eigenschaft dieser Pfade ist, dass für jeden Index $0 \leq i < k$ und für jeden Punkt p_j auf dem Pfad, der über mindestens i Nachfolger verfügt, die Hammingdistanz $\mathrm{H}(p_j, p_{j+i})$ von p_j und p_{j+i} genau i beträgt. Außerdem gilt $\mathrm{H}(p_j, p_{j+i}) \geq k$ für jeden Index $i \geq k$. Also ist für eine große Abkürzung die simultane Mutation von k Bits erforderlich. Die Wahrscheinlichkeit für solch ein Ereignis ist für $k = \Omega(\sqrt{n})$ exponentiell klein. Die Länge eines langen k-Pfades der Dimension n beträgt $|P_k^n| = k \cdot 2^{n/k} - k + 1$ [Sudholt, 2008]



(a) Suchraum $\mathbb{B}^n$.

(b) Zielraum $\mathrm{g}_{k_1,k_2,k_3}(\mathbb{B}^n)$.

Abbildung 11.2.: Suchraum und Zielraum der Beispielfunktion g_{k_1,k_2,k_3}.

und ist demzufolge für $k = \mathcal{O}(\sqrt{n})$ immer noch exponentiell groß. Eine Analyse des evolutionären Algorithmus (1+1)-EA auf langen k-Pfaden mit $k = \Theta(\sqrt{n})$ wurde beispielsweise von Droste et al. [2002] präsentiert. Das genannte Resultat zeigt auf eindrucksvolle Weise, dass die Optimierungszeit auch für unimodale Funktionen, die im Allgemeinen als einfach gelten, exponentiell groß sein kann.

Da die Fitness auf dem langen k-Pfad ansteigt, wird ein Individuum aus der Population des Algorithmus (μ+1)-EA mit Älterwerden, das den Pfad erreicht hat, erst am Ende des Pfades entfernt, wenn es dem Pfad hinreichend schnell folgt. (Wenn die Individuen dem Pfad bezogen auf die Lebensspanne nicht hinreichend schnell folgen, kann es passieren, dass ein Individuum aufgrund seines Alters entfernt und durch ein neues, zufälliges Individuum ersetzt wird.) Also ist die Optimierungszeit des Algorithmus (μ+1)-EA mit Älterwerden exponentiell groß mit einer Wahrscheinlichkeit, die exponentiell schnell gegen 1 konvergiert, wenn alle Individuen aus der Population auf dem Pfad gefangen sind.

Die weiter unten definierte Beispielfunktion g_{k_1,k_2,k_3} ist im Wesentlichen eine Kombination von Ridge und einem langen k-Pfad für $k = \sqrt{n/4}$. Streng genommen müssen $k = \sqrt{n/4} \in \mathbb{N}$ und $n/k \in \mathbb{N}$ sein. Wir ersparen uns die Mühe, $\lfloor\sqrt{n/4}\rfloor$ zu benutzen oder nur n mit $n = (4 \cdot i)^2$ für $i \in \mathbb{N}$ zu betrachten, da dies zu keinen wesentlichen Änderungen führen würde. Eine Veranschaulichung von g_{k_1,k_2,k_3} ist in Abbildung 11.2 zu finden. Beachte, dass die „Schlangenlinie“ den langen Pfad symbolisiert, der sich durch den booleschen Hyperwürfel zieht.

Definition 11.6. *Seien $n \in \mathbb{N}^+$ und $k_1, k_2, k_3 \in \mathbb{N}^+$ mit $k_1, k_2, k_3 = \mathcal{O}(1)$. Die*

Funktion $\mathrm{g}_{k_1,k_2,k_3} \colon \mathbb{B}^n \to \mathbb{R}$ *ist für alle* $x \in \mathbb{B}^n$ *durch*

$$\mathrm{g}_{k_1,k_2,k_3}(x) := \begin{cases} 2^n & \text{falls } x = 1^{2n/3}0^{n/3-k_3}1^{k_3} \\ n^3 + i & \text{falls } x = p'_i \\ n^2 + n & \text{falls } x = 1^{3n/4-k_2}0^{n/4+k_2} \\ n^2 + i & \text{falls } x = r_i \\ n - \mathrm{OneMax}(x) & \text{anderenfalls} \end{cases}$$

definiert, wobei

$$\begin{aligned} r_i &= 1^i 0^{n-i}, 0 \le i \le 3n/4 - k_1 - k_2, \\ p'_i &= x^{(\ell)} \circ x^{(r)}, x^{(\ell)} = 1^{3n/4}, x^{(r)} = p_i \in P^{n/4}_{\sqrt{n/4}}, \\ x^{(\ell)} &= (x_1, \ldots, x_{3n/4}), \\ x^{(r)} &= (x_{3n/4+1}, \ldots, x_n) \end{aligned}$$

sind und $\circ$ *die Konkatenation von zwei Bitvektoren bezeichnet.*

Beachte, dass die p'_i die Punkte auf dem langen k-Pfad mit $k = \sqrt{n/4}$ und die r_i die Punkte auf dem anderen Teilpfad bezeichnen. Es gibt zwei Schluchten der Größe k_1 bzw. k_2, die den letztgenannten Teilpfad und den langen k-Pfad trennen. Das erste Individuum, das es schafft, die erste Schlucht zu überspringen, übernimmt höchstwahrscheinlich die gesamte Population, indem es sich nach und nach dupliziert, wenn wir voraussetzen, dass die Populationsgröße μ nicht zu groß ist. In diesem Fall weisen alle Individuen in der Population ein identisches Alter auf. Dann kommt es höchstwahrscheinlich dazu, dass die komplette Population auf einen Schlag am Ende einer Generation aufgrund ihres Alters ausstirbt, wenn wir voraussetzen, dass die Lebensspanne τ hinreichend klein ist. Dieses Verhalten entspricht genau einem kompletten Neustart des Algorithmus. Wir behaupten nicht, dass zwei hintereinander geschaltete Schluchten notwendig sind, damit das Älterwerden von Individuen sinnvoll ist. Die Konstruktion vereinfacht den folgenden Beweis jedoch erheblich. Wir erwähnen, dass die Kombination von Neustarts mit zwei Pfaden, denen ein evolutionärer Algorithmus mit unterschiedlichen Wahrscheinlichkeiten folgt, bereits von Jansen [2002] untersucht wurde.

Weil wir g_{k_1,k_2,k_3} immer mit den gleichen Parametern k_1, k_2 und k_3 benutzen, werden wir der kürzeren Notation halber im Folgenden g statt g_{k_1,k_2,k_3} schreiben. Analog zu Abschnitt 11.3 partitionieren wir den Suchraum $\mathbb{B}^n$ in die fünf disjunkten Mengen

$$\begin{aligned} B &:= \{r_i \mid 0 \le i < 3n/4 - k_1 - k_2\}, \\ C &:= \{r_{3n/4-k_1-k_2}, 1^{3n/4-k_2}0^{n/4+k_2}\}, \\ D &:= \{p'_i \mid p_i \in P^{n/4}_{\sqrt{n/4}}\}, \\ E &:= \{1^{2n/3}0^{n/3-k_3}1^{k_3}\}, \\ A &:= \mathbb{B}^n \setminus (B \cup C \cup D \cup E). \end{aligned}$$

Die Mengen A, B, C, D und E formen erneut eine g-basierte Partition $A <_g B <_g C <_g D <_g E$ des Suchraumes.

Während eines typischen Laufes des Algorithmus (μ+1)-EA mit Älterwerden beginnt die Population in A und nähert sich 0^n, bis der erste Punkt aus B gefunden wird. Dann folgt die Population dem durch die Punkte aus B definierten Pfad bis entweder das globale Optimum aus E gefunden oder das Ende des Pfades erreicht wird. Im letztgenannten Fall kann es passieren, dass die Individuen C passieren, d. h., über beide Schluchten springen, und D erreichen, wenn die Lebensspanne τ hinreichend groß ist. Wir werden sehen, dass die Population anderenfalls ausstirbt und neu in A initialisiert wird, bis letztlich doch einmal ein Sprung zum globalen Optimum aus E gelingt.

Wir zeigen zunächst, dass der bekannte Algorithmus (μ+1)-EA genau wie die Variante (μ+1)-EA mit Älterwerden, wenn die Lebensspanne zu groß ist, Schwierigkeiten hat, die Funktion g zu optimieren, da er relativ einfach den durch den langen k-Pfad gebildeten Irrweg erreicht.

Theorem 11.7. *Seien $n \in \mathbb{N}^+$ und $k_1, k_2, k_3 \in \mathbb{N}^+$ mit $k_1, k_2, k_3 = \mathcal{O}(1)$, $k_1 \leq k_2$, $k_2 \geq 2$ und $k_3 \leq k_2$. Dann gelten für jede Populationsgröße $\mu = \mathcal{O}(n^{k_3-1}/(\log n))$ und für jede Lebensspanne $\tau = \omega(n^{k_2})$*

$$\mathrm{W}(T(\textit{(}\mu\textit{+1)-EA, g}) = 2^{\Omega(\sqrt{n})}) = 1 - \mathrm{o}(1),$$
$$\mathrm{W}(T(\textit{(}\mu\textit{+1)-EA mit Älterwerden, g}) = 2^{\Omega(\sqrt{n})}) = 1 - \mathrm{o}(1).$$

Beweis. Wir benutzen die Partition $A \cup B \cup C \cup D \cup E$ des Suchraumes $\mathbb{B}^n$, die wir weiter oben definiert haben. Bezeichne T die Optimierungszeit des betrachteten Algorithmus. Sei T_A die Anzahl von Generationen, in denen $P_t \subseteq A$ gilt. Sei T_B die Anzahl von Generationen, in denen $P_t \subseteq A \cup B$ und $P_t \cap B \neq \emptyset$ gelten. Sei T_C die Anzahl von Generationen, in denen $P_t \subseteq A \cup B \cup C$ und $P_t \cap C \neq \emptyset$ gelten. Außerdem sei T_D die Anzahl von Generationen, in denen $P_t \subseteq A \cup B \cup C \cup D$ und $P_t \cap D \neq \emptyset$ gelten. Dann gilt $T = T_A + T_B + T_C + T_D$.

Wir betrachten zunächst den Algorithmus (μ+1)-EA. Für alle Individuen x in der initialen Population P_0 gilt $\mathrm{OneMax}(x) \leq (7/12) \cdot n$ mit Wahrscheinlichkeit $1 - 2^{-\Omega(n)}$ (Chernoff-Ungleichung und boolesche Ungleichung). Dann schließen wir auf analoge Weise wie zuvor, dass $\mathrm{E}(T_A) = \mathcal{O}(\mu n + n \log n)$, $\mathrm{E}(T_B) = \mathcal{O}(\mu n \log n + n^2)$ und $\mathrm{E}(T_C) = \mathcal{O}(\mu \log n + n^{k_2})$ (da $k_2 \geq k_1$) gelten. Insgesamt gilt $\mathrm{E}(T_A + T_B + T_C) = \mathcal{O}(n^{k_2})$, da $k_2 \geq 2$ und $\mu = \mathcal{O}(n^{k_2-1}/(\log n))$ gelten. Also gibt es eine Konstante $c > 0$ mit $\mathrm{E}(T_A + T_B + T_C) \leq c \cdot n^{k_2}$. Dann gilt $T_A + T_B + T_C \leq 2 \cdot c \cdot n^{k_2} \cdot \log n$ mit Wahrscheinlichkeit $1 - 1/n = 1 - \mathrm{o}(1)$.

Sei $y = 1^{(2/3)\cdot n} 0^{(1/3)\cdot n - k_3} 1^{k_3}$ das globale Optimum. Wir beweisen nun, dass ein Individuum aus D vor dem Individuum y mit Wahrscheinlichkeit $1 - \mathrm{o}(1)$ erreicht wird. Sei x mit $\mathrm{H}(x, y) \geq k_2 + 1$. Dann beträgt die Wahrscheinlichkeit, y durch eine Mutation zu erzeugen, höchstens $1/n^{k_2+1}$. Sei $1^{(2/3)\cdot n \pm i} 0^{(1/3)\cdot n \mp i}$ mit $0 \leq i \leq k_2 - k_3$. Dann beträgt die Wahrscheinlichkeit, das Individuum y vor dem Individuum $1^{i+1} 0^{n-i-1}$ durch eine Mutation zu erzeugen, höchstens $(1/(n^{i+k_3}))/(1/(e \cdot n)) =$

e/n^{i+k_3-1}. Die Wahrscheinlichkeit, das Individuum y vor einem Individuum aus D zu erreichen, kann mithilfe der booleschen Ungleichung durch

$$(2 \cdot c \cdot n^{k_2} \cdot \log n) \cdot (1/n^{k_2+1}) + \sum_{i=0}^{k_2-k_3} 2 \cdot \mu \cdot (e/n^{i+k_3-1}) = \mathrm{o}(1)$$

abgeschätzt werden, da $\mu = \mathrm{o}(n^{k_3-1})$ für die Populationsgröße gilt.

Anschließend beginnt das Individuum aus D, die Population nach D zu ziehen. Die komplette Population ist nach $\mathcal{O}(\mu \cdot \log^2 n) = \mathrm{o}(n^{k_3-1} \cdot \log n)$ Generationen mindestens mit Wahrscheinlichkeit $1 - 1/n$ in D enthalten. In dieser Phase wird das globale Optimum mit Wahrscheinlichkeit $1 - \mathrm{o}(1)$ nicht gefunden, da mindestens k_3 bestimmte Bits kippen müssen.

Auf dem langen k-Pfad genügt die Mutation eines bestimmten Bits, um einen Nachkommen zu erhalten, der eine größere Fitness als sein Elter aufweist. Wie bereits oben festgestellt, sind lange k-Pfade gerade so konstruiert, dass eine Mutation von mindestens k Bits notwendig ist, um eine große Abkürzung zu nehmen. Das Vorhandensein einer Population vergrößert die Wahrscheinlichkeit solcher Mutationen nicht. Folglich überträgt sich die untere Schranke für die Optimierungszeit des Algorithmus (1+1)-EA für einen langen k-Pfad aus [Droste et al., 2002] auf den Algorithmus (μ+1)-EA. Wir erhalten deshalb für $k = \sqrt{n/4}$ das Ergebnis, dass in

$$\Theta\left(\left(\frac{n}{4}\right)^{3/2} \cdot 2^{\sqrt{n/4}}\right) = 2^{\Omega(\sqrt{n})}$$

Generationen mit Wahrscheinlichkeit $1 - 2^{-\Omega(\sqrt{n})}$ weder der letzte Suchpunkt auf dem langen Pfad noch das globale Optimum gefunden werden, da der Algorithmus das globale Optimum in dieser Situation nur durch das Kippen von mindestens $\Omega(n)$ Bits erreichen kann. Das beweist die Behauptung für den Algorithmus (μ+1)-EA.

Wir betrachten nun den Algorithmus (μ+1)-EA mit Älterwerden. Die obige Analyse überträgt sich, wenn wir zeigen, dass niemals ein bestes Individuum aufgrund seines Alters aus der Population entfernt wird. Die Wahrscheinlichkeit, dass wir ein bestes Individuum durch eine 1-Bit-Mutation in $\tau + 1 = \omega(n^{k_2})$ Generationen verbessern, beträgt mindestens $1 - 2^{-\Omega(n)}$. Die Wahrscheinlichkeit, dass wir ein bestes Individuum durch eine k_1-Bit-Mutation bzw. eine k_2-Bit-Mutation verbessern, beträgt mindestens $1 - \mathrm{o}(1)$. Da sich der Algorithmus (μ+1)-EA mit Älterwerden mit Wahrscheinlichkeit $1 - \mathrm{o}(1)$ genau wie der im ersten Teil dieses Beweises beschriebene Algorithmus (μ+1)-EA verhält, folgt die zweite Behauptung des Theorems. Insgesamt haben wir damit das Theorem bewiesen. □

Wir haben festgestellt, dass der Algorithmus (μ+1)-EA mit Älterwerden Schwierigkeiten hat, die Funktion g zu optimieren, wenn die Lebensspanne τ zu groß ist. Das liegt daran, dass die Population auf dem langen k-Pfad gefangen werden kann, da auf diesem kleine Fortschritte möglich sind. Insbesondere zeigt der Beweis des letzten Theorems, dass sich der genannte Algorithmus größtenteils wie der Algorithmus (μ+1)-EA verhält. Im Folgenden stellen wir die Vorteile des Algorithmus

(μ+1)-EA mit Älterwerden heraus. Wenn die Lebensspanne τ hinreichend klein ist, kann der Mechanismus des Älterwerdens den Algorithmus davor bewahren, den langen k-Pfad zu erreichen und auf diesem viel Zeit zu verschwenden. Beachte, dass die Lebensspanne τ auch nicht zu klein sein darf. Sehr kleine Lebensspannen τ halten den Algorithmus sogar davon ab, sehr einfache Funktionen zu optimieren.

Theorem 11.8. *Seien $n \in \mathbb{N}^+$ und $k_1, k_2, k_3 \in \mathbb{N}^+$ mit $k_1, k_2, k_3 = \mathcal{O}(1)$, $k_3 < k_1 < k_2$ und $k_3 + 3 \leq k_2$. Dann gilt für jede Populationsgröße μ mit $\mu = \mathrm{o}(n^{k_1 - k_3}/(\log^4 n))$ und für jede Lebensspanne τ mit $\tau = \omega(\mu \cdot \log^2 n + n \cdot \log n)$ und $\tau = \mathrm{o}(n^{k_2 - k_3}/(\log^2 n))$*

$$\mathrm{W}(T(\text{(μ+1)-EA mit Älterwerden, g}) = \mathcal{O}(\alpha(n, \mu, \tau))) = 1 - \mathrm{o}(1),$$

wobei $\alpha(n, \mu, \tau) = n^{k_1 + k_3} \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n)$ ist.

Beweis. In diesem Beweis vollziehen wir das Verhalten des Algorithmus (μ+1)-EA mit Älterwerden schrittweise nach. Wir werden dabei einen Lauf beschreiben, der mit Wahrscheinlichkeit $1 - \mathrm{o}(1)$ stattfindet. Die Sprechweisen „hohe Wahrscheinlichkeit“ und „überwältigende Wahrscheinlichkeit“ beziehen sich hier auf Wahrscheinlichkeiten aus $1 - 2^{-\Omega(\log n)}$ bzw. $1 - 2^{-\Omega(n)}$ (siehe auch Definition A.22).

Wir betrachten insgesamt $\mathcal{O}(n^{k_1 + k_3} \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n))$ Generationen. Daher gilt mit überwältigender Wahrscheinlichkeit $\mathrm{OneMax}(x) \leq (7/12) \cdot n$ für alle zufälligen Individuen $x \in \mathbb{B}^n$ (Chernoff-Ungleichung und boolesche Ungleichung). Außerdem kippen mit überwältigender Wahrscheinlichkeit höchstens $(1/12) \cdot n$ Bits pro Mutation. Wenn ein bestes Individuum aus der aktuellen Population durch das Kippen eines Bits verbessert werden kann, dauert es im Erwartungswert höchstens $\mathcal{O}(\mu \cdot \log n + n)$ Generationen, bis die Population ein besseres Individuum enthält. Die Schranke $\mathcal{O}((\mu \cdot \log n + n) \cdot \log n)$ gilt mit hoher Wahrscheinlichkeit. Da wir $\tau = \omega(\mu \cdot \log^2 n + n \cdot \log n)$ voraussetzen, wird in den betrachteten Generationen mit hoher Wahrscheinlichkeit kein solches Individuum aufgrund seines Alters aus der Population entfernt.

Betrachte $\mathcal{O}(\mu \cdot n \cdot \log^2 n + n^2 \cdot \log n + \tau)$ Generationen. Wir erreichen ein Individuum aus C (oder ein Individuum aus D oder E) im Erwartungswert nach

$$\mathcal{O}\big((\mu \cdot n + n \cdot \log n) + (\mu \cdot n \cdot \log n + n^2)\big) = \mathcal{O}\big(\mu \cdot n \cdot \log n + n^2\big)$$

Generationen. Also wird in der genannten Phase mit hoher Wahrscheinlichkeit ein Individuum aus C gefunden. Wenn es sich dabei nicht um das zweite Individuum aus C handelt, übernimmt das erste Individuum aus C nach $\mathcal{O}(\mu \cdot \log^2 n)$ Generationen mit hoher Wahrscheinlichkeit die komplette Population. Beachte, dass $\tau = \omega(\mu \cdot \log^2 n)$ gilt. Die Alter der Individuen können sich an dieser Stelle übrigens unterscheiden. In dieser Situation erreichen wir in der nächsten Generation das zweite Individuum aus C mit Wahrscheinlichkeit $\Omega(n^{-k_1})$. Wenn dies nicht passiert, dann sind nach $\mathcal{O}(\tau)$ weiteren Generationen alle Nachkommen der genannten Individuen aus der Population verschwunden und wir befinden uns in der Ausgangssituation. Insgesamt wird das zweite Individuum aus C in der betrachteten Phase mit Wahrscheinlichkeit $\Omega(n^{-k_1})$

gefunden. Also dauert es mit hoher Wahrscheinlichkeit

$$\mathcal{O}\Big(n^{k_1} \cdot (\mu \cdot n \cdot \log^2 n + n^2 \cdot \log n + \tau) \cdot \log n\Big) \\ = \mathcal{O}\Big(\mu \cdot n^{k_1+1} \cdot \log^3 n + n^{k_1+2} \cdot \log^2 n + \tau \cdot n^{k_1} \cdot \log n\Big)$$

Generationen, bis das zweite Individuum aus C gefunden wird. Die Wahrscheinlichkeit, ein Individuum aus D zu erreichen, beträgt höchstens $1/(n^{k_1+k_2})$. Dies passiert also in der betrachteten Phase mit Wahrscheinlichkeit höchstens

$$\mathcal{O}\Big(\mu \cdot n^{-k_2+1} \cdot \log^3 n + n^{-k_2+2} \cdot \log^2 n + \tau \cdot n^{-k_2} \cdot \log n\Big)$$

(boolesche Ungleichung). Dieses erste ungünstige Ereignis tritt mit Wahrscheinlichkeit $1 - \mathrm{o}(n^{-k_3}/(\log n))$ nicht ein, da wir $\mu = \mathrm{o}(n^{k_2-k_3-1}/(\log^4 n))$, $k_2 \geq k_3 + 3$ und $\tau = \mathrm{o}(n^{k_2-k_3}/(\log^2 n))$ voraussetzen. Beachte, dass $\mu = \mathrm{o}(n^{k_2-k_3-1}/(\log^4 n))$ aus $k_1 < k_2$ und $\mu = \mathrm{o}(n^{k_1-k_3}/(\log^4 n))$ folgt.

Nun dauert es im Erwartungswert $\mathcal{O}(\mu \cdot \log n)$ Generationen bis das zweite Individuum aus C die Population übernimmt. Die Schranke $\mathcal{O}(\mu \cdot \log^2 n)$ gilt mit hoher Wahrscheinlichkeit. In dieser Phase schafft mit Wahrscheinlichkeit $1 - \mathcal{O}(\mu \cdot n^{-k_1} \cdot \log^2 n)$ kein Individuum einen Sprung über eine Schlucht, da $k_2 \geq k_1$ gilt. Dieses zweite ungünstige Ereignis tritt mit Wahrscheinlichkeit $1 - \mathrm{o}(n^{-k_3}/(\log n))$ nicht ein, da wir $\mu = \mathrm{o}(n^{k_1-k_3}/(\log^3 n))$ voraussetzen. Also enthält die Population anschließend nur Individuen, die sich auf den gleichen Suchpunkt beziehen und die das gleiche Alter aufweisen.

Betrachte $\mathcal{O}(\tau)$ Generationen. In dieser Phase schafft mit Wahrscheinlichkeit $1 - \mathcal{O}(\tau \cdot n^{-k_2})$ kein Individuum den Sprung über die zweite Schlucht. Dieses dritte ungünstige Ereignis tritt mit Wahrscheinlichkeit $1 - \mathrm{o}(n^{-k_3}/(\log n))$ nicht ein, da wir $\tau = \mathrm{o}(n^{k_2-k_3}/(\log n))$ voraussetzen. Also sterben alle Individuen auf einen Schlag aus und die gesamte Population wird neu initialisiert. Dieses Verhalten gleicht einem kompletten Neustart des Algorithmus.

Wir betrachten nun die Wahrscheinlichkeit, dass das globale Optimum in einer der beschriebenen Phasen gefunden wird. Betrachte ein Individuum $1^{(2/3)\cdot n-i}0^{(1/3)\cdot n+i}$ mit $1 \leq i \leq (2/3) \cdot n$. Dann beträgt die Wahrscheinlichkeit, durch eine Mutation das Individuum $1^{(2/3)\cdot n}0^{(1/3)\cdot n}$ zu erzeugen, mindestens $1/(e \cdot n^i) = \Omega(n^{-i})$. Andererseits beträgt die Wahrscheinlichkeit, durch eine Mutation ein bestimmtes Individuum $1^{(2/3)\cdot n+j}0^{(1/3)\cdot n-j}$ mit $1 \leq i \leq (1/3) \cdot n$ zu erzeugen, höchstens $1/n^{i+j}$. Die Wahrscheinlichkeit, eines der genannten Individuen zu erzeugen, beträgt folglich höchstens $\sum_{j=1}^{(1/3)\cdot n}(1/n^{i+j}) \leq (1/n^i) \cdot (1/(n-1)) = \mathcal{O}(n^{-i-1})$. Also befinden wir uns mit Wahrscheinlichkeit $1 - \Omega(n^{-1})$ in einer Situation, in der $1^{(2/3)\cdot n}0^{(1/3)\cdot n}$ das beste Individuum in der Population ist. Es dauert trivialerweise mindestens μ Generationen, bis alle Individuen in der Population besser sind. Also wird das genannte Individuum in dieser Phase mit Wahrscheinlichkeit $\Omega(1)$ zur Mutation ausgewählt. Das globale Optimum wird in der genannten Mutation mit Wahrscheinlichkeit mindestens $1/(e \cdot n^{k_3})$ erreicht. Also wird das Optimum in einer der betrachteten Phasen mit Wahrscheinlichkeit $\Omega(n^{-k_3})$ gefunden.

Wenn wir nun $\Theta(n^{k_3} \cdot \log n)$ Phasen der Länge

$$\mathcal{O}(n^{k_1} \cdot (\mu \cdot n \cdot \log^3 n + n^2 \cdot \log^2 n + \tau \cdot \log n))$$

betrachten, wird das globale Optimum mit Wahrscheinlichkeit $1 - o(1)$ gefunden, da die drei ungünstigen Ereignisse mit Wahrscheinlichkeit $o(n^{-k_3}/(\log n))$ eintreten. Insgesamt beträgt die Optimierungszeit in diesem Fall

$$\mathcal{O}(n^{k_1+k_3} \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n))$$

Generationen. □

Die Aussagen von Theorem 11.7 und Theorem 11.8 sind relativ kompliziert. Wir veranschaulichen sie anhand eines Beispiels. Nehmen wir an, um den langen Irrpfad zu erreichen, müssen $k_1 = 3$ und $k_2 = 5$ Bits gekippt werden. Um das Optimierungsziel zu erreichen, müssen $k_3 = 2$ Bits gekippt werden. Betrachte die Populationsgröße $\mu = \lfloor n/(\log^4 n) \rfloor$. Dann sagt Theorem 11.7, dass die Optimierungszeiten der Algorithmen (μ+1)-EA und (μ+1)-EA mit Älterwerden mit Wahrscheinlichkeit $1 - o(1)$ mindestens $2^{\Omega(\sqrt{n})}$ betragen, wenn $\omega(n^5)$ für die Lebensspanne τ gilt. Außerdem sagt Theorem 11.8, dass die Optimierungszeit des Algorithmus (μ+1)-EA mit Älterwerden mit Wahrscheinlichkeit $1 - o(1)$ höchstens $\mathcal{O}(n^7 \cdot \log^3 n + \tau \cdot n^5 \cdot \log^2 n)$ beträgt, wenn $\tau = \omega(n \cdot \log n)$ und $\tau = o(n^3/(\log^2 n))$ für die Lebensspanne τ gelten. Das bedeutet insbesondere, dass die Optimierungszeit z. B. für $\tau = \lfloor n^5 \cdot \log n \rfloor$ exponentiell groß ist während sie für $\tau = \lfloor n^3/(\log^3 n) \rfloor$ polynomiell groß ist.

Wenn wir den Beweis von Theorem 11.8 Revue passieren lassen, sehen wir, dass der Algorithmus (μ+1)-EA mit Älterwerden durch die begrenzte Lebensspanne der Individuen davor bewahrt wird, den exponentiell langen Irrweg zu erreichen. Der Beweis zeigt, dass mit hoher Wahrscheinlichkeit ein Individuum entweder den Sprung zum globalen Optimum oder den Sprung über die erste Schlucht schafft. Im ersten Fall hat der Algorithmus das Optimierungsziel erreicht; im zweiten Fall übernimmt das Individuum mit hoher Wahrscheinlichkeit die komplette Population, die nach höchstens τ weiteren Generation auf einen Schlag ausstirbt. Auf diese Weise wird ein kompletter Neustart des Algorithmus realisiert. Wenn es zu einem kompletten Neustart kommt, dann haben die Individuen erneut eine nicht zu geringe Chance, das globale Optimum durch einen Sprung zu erreichen.

Aus dem im letzten Absatz Gesagten folgt, dass auch der Algorithmus (μ+1)-EA das globale Optimum finden kann, wenn wir den Algorithmus nach einer geeigneten Anzahl von Generationen abbrechen und neu starten. Insofern stellt sich die Frage, wie Funktionen beschaffen sein müssen, die von einem Algorithmus mit einem Alterungsoperator effizient optimiert werden können, während die skizzierte Neustart-Strategie scheitert. In [Jansen und Zarges, 2010a] wird eine Beispielfunktion konstruiert, die die gewünschten Eigenschaften besitzt. Sie analysieren die in diesem Kapitel betrachteten Algorithmen, wobei zusätzlich ein Rekombinationsoperator zu Einsatz kommt. Die Beispielfunktion ist so gestaltet, dass der Algorithmus (μ+1)-EA mit Älterwerden von partiellen Neustarts profitiert, da der Rekombinationsoperator aus

einem rein zufälligen Individuum und einem Individuum, das sich in einem lokalen Optimum eingefunden hat, mit einer nicht zu geringen Wahrscheinlichkeit ein optimales Individuum kreiert. Der Einsatz eines Alterungsoperators in Verbindung mit einem Rekombinationsoperator wirft Fragen auf. In [Jansen und Zarges, 2010a] wird das Alter eines Nachkommen von zwei Eltern auf 0 gesetzt, wenn der Nachkomme eine echt größere Fitness als beide Eltern aufweist. Anderenfalls, wird sein Alter auf das Alter eines ältesten Elters gesetzt. Andere Alternativen sind hier natürlich möglich. Der Fragestellung, welche Konsequenzen diese Entwurfsentscheidungen haben können, wird in [Jansen und Zarges, 2010b] nachgegangen.

11.5. Kombination von Beispielfunktionen

In diesem Abschnitt präsentieren wir eine allgemeine Methode, die benutzt werden kann, um die Eigenschaften einer Beispielfunktion zu verstärken oder die Eigenschaften verschiedener Beispielfunktionen zu kombinieren.

Betrachte k pseudoboolesche Funktionen $\mathrm{h}_i\colon \mathbb{B}^{n_i} \to \mathbb{R}^+$. Wir wählen zunächst $\mathrm{h}_i^{\mathrm{ub}} \in \mathbb{R}^+$ mit $\mathrm{h}_i^{\mathrm{ub}} \geq \max\{\mathrm{h}_i(x^{(i)}) \mid x^{(i)} \in \mathbb{B}^{n_i}\}$. Die Bestimmung der genannten Schranken ist problemlos möglich, wenn wir Beispielfunktionen betrachten. Wir arbeiten darauf hin, alle h_i zu einer einzigen pseudobooleschen Funktion zu verschmelzen. Die grundlegende Idee besteht darin, eine Funktion zu kreieren, die den Algorithmus (μ+1)-EA (mit Älterwerden) dazu bringt, alle h_i sequenziell zu optimieren.

Wir betrachten zwei Alternativen, die Optimierung einer Funktion h_i *abzuschließen*: Die Konstruktion eines Suchpunktes $x^{(i)}$ mit $x^{(i)} \in OPT_i$ oder $x^{(i)} \in TRAP_i$, wobei OPT_i und $TRAP_i$ zwei disjunkte Teilmengen des Suchraumes $\mathbb{B}^{n_i}$ sind. Im ersten Fall nennen wir die Optimierung von h_i einen *Erfolg* und im zweiten Fall nennen wir sie einen *Fehlschlag*.

Betrachte $\mathrm{h}\colon \mathbb{B}^n \to \mathbb{R}^+$, wobei $n = \sum_{i=1}^{k} n_i$ ist. Wir partitionieren einen Bitvektor $x = (x_1, \ldots, x_n)$ der Länge n in k Bitvektoren

$$x^{(i)} = \left(x_{\sum_{j=1}^{i-1} n_j+1}, \ldots, x_{\sum_{j=1}^{i} n_j}\right)$$

der Länge n_i, um die Funktionsdefinition

$$\mathrm{h}(x) = \begin{cases} \sum_{i=1}^{a(x)} \mathrm{h}_i^{\mathrm{ub}} + \mathrm{h}_{a(x)+1}(x^{(a(x)+1)}) & \text{falls } a(x) < k \\ \sum_{i=1}^{a(x)} \mathrm{h}_i^{\mathrm{ub}} + 1 & \text{falls } a(x) = k \text{ und } b(x) \leq \lfloor k/2 \rfloor \\ \sum_{i=1}^{a(x)} \mathrm{h}_i^{\mathrm{ub}} + 2 & \text{falls } a(x) = k \text{ und } b(x) > \lfloor k/2 \rfloor \end{cases}$$

zu vereinfachen, wobei

$$a(x) = \max\{1 \leq i \leq k \mid \forall 1 \leq j \leq i\colon x^{(j)} \in OPT_j \cup TRAP_j\}$$

die maximale Länge einer Folge $\mathrm{h}_1, \ldots, \mathrm{h}_{a(x)}$ von abgeschlossenen h_i und

$$b(x) = |\{1 \leq i \leq k \mid x^{(i)} \in OPT_i\}|$$

die Anzahl von erfolgreich abgeschlossenen h_i bezeichnen. Die Struktur von h lässt sich wie folgt beschreiben. Ein Suchpunkt $x' = (x'_1, \ldots, x'_n)$ ist besser als ein Suchpunkt $x = (x_1, \ldots, x_n)$, wenn $a(x') > a(x)$ gilt. Außerdem ist x' besser als x, wenn $a(x') = a(x)$ und $h_{a(x')+1}(x') > \mathrm{h}_{a(x)+1}(x)$ erfüllt sind. Ein Suchpunkt $x = (x^{(1)}, \ldots, x^{(k)})$ ist genau dann ein globales Optimum von h, wenn $\mathrm{h}_i(x^{(i)}) \in OPT_i \cup TRAP_i$ für alle $1 \leq i \leq k$ und $\mathrm{h}_i(x^{(i)}) \in OPT_i$ für mehr als $\lfloor k/2 \rfloor$ verschiedene i gelten. Wenn $\mathrm{h}_i(x^{(i)}) \in OPT_i$ für höchstens $\lfloor k/2 \rfloor$ verschiedene i zutrifft, dann wird x auf die zweitbeste Fitness abgebildet.

Wir wenden diese Methode nun an, um eine Beispielfunktion zu erschaffen, die sowohl die Eigenschaften der Beispielfunktion g aus Abschnitt 11.4 als auch die der Beispielfunktion f_k aus Abschnitt 11.3 aufweist. Betrachte $\mathrm{g}\colon \mathbb{B}^{n/2} \to \mathbb{R}^+$ als h_1 und $\mathrm{f}_k\colon \mathbb{B}^{n/2} \to \mathbb{R}^+$ als h_2. Wähle die Schranken $\mathrm{h}_1^{\mathrm{ub}} = 2^{n/2}$ und $\mathrm{h}_2^{\mathrm{ub}} = n/2 + \lfloor n/(2k) \rfloor$ (siehe Definition 11.2 und 11.6). Seien $OPT_1 = \{1^{2n/6}0^{n/6-k_3}1^{k_3}\}$ und $TRAP_1 = \emptyset$ sowie $OPT_2 = \{1^{\lfloor n/(2k) \rfloor k}0^{n/2-\lfloor n/(2k) \rfloor k}\}$ und $TRAP_2 = \emptyset$. Wir kombinieren h_1 und h_2 auf die eingangs beschriebene Weise. Wir nennen die resultierende Funktion $\mathrm{h}_{k_1,k_2,k_3,k}$ und schreiben erneut einfach h.

Das folgende Theorem zeigt, dass der Algorithmus (μ+1)-EA mit Älterwerden nicht in der Lage ist, h effizient zu optimieren, wenn τ zu klein oder zu groß gewählt wird. Wenn τ innerhalb eines bestimmten Intervalls liegt, dann optimiert der Algorithmus (μ+1)-EA mit Älterwerden die Funktion h effizient.

Theorem 11.9. *Seien $n \in \mathbb{N}^+$ und $k_1, k_2, k_3, k \in \mathbb{N}^+ \setminus \{1\}$ mit $k_1, k_2, k_3, k = \mathcal{O}(1)$, $k_3 < k_1 < k_2$, $k_2 - k_3 \geq 3$ und $k < k_1 + k_3$. Dann gilt für jede Populationsgröße $\mu = \mathrm{o}(n^{k_1-k_3}/(\log^4 n))$ und für jede Lebensspanne τ das Folgende.*

1. *Wenn $\tau = \mathrm{o}(n^k \cdot \log n)$ gilt, dann gilt für jede Konstante $0 < \epsilon < 1$*

$$\mathrm{W}(T(\text{(}\mu\text{+1)-EA m. Älterwerden, h}) = 2^{\Omega(n^\epsilon)}) = 1 - 2^{-\Omega(n^{1-\mathrm{o}(1)})}.$$

2. *Wenn $\tau = \omega((n^k + \mu \cdot \log n) \cdot (\log n))$ und $\tau = \mathrm{o}(n^{k_2-k_3}/(\log^2 n))$ gelten, dann gilt*

$$\mathrm{W}(T(\text{(}\mu\text{+1)-EA m. Älterwerden, h}) = \mathcal{O}(\alpha(n, \mu, \tau))) = 1 - \mathrm{o}(1),$$

   wobei $\alpha(n, \mu, \tau) = n^{k_1+k_3} \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n)$ ist.

3. *Wenn $\tau = \omega(n^{k_2})$ gilt, dann gilt*

$$\mathrm{W}(T(\text{(}\mu\text{+1)-EA m. Älterwerden, h}) = 2^{\Omega(\sqrt{n})}) = 1 - \mathrm{o}(1).$$

Beweis. 1. Nachdem das erste Individuum x die Funktion $\mathrm{h}_1 = \mathrm{g}$ abgeschlossen hat, besteht die zweite Hälfte von x aus rein zufällig verteilten Bits. Von nun an wird die Fitness von x durch die Funktion $\mathrm{h}_2 = \mathrm{f}_k$ bestimmt. Gemäß Theorem 11.4, schließt der Algorithmus (μ+1)-EA mit Älterwerden die Funktion $\mathrm{h}_2 = \mathrm{f}_k$ mit Wahrscheinlichkeit $1 - 2^{-\Omega(n^{1-\mathrm{o}(1)})}$ *nicht* innerhalb von $2^{\Omega(n^\epsilon)}$ Generationen ab, wobei

$0 < \epsilon < 1$ eine beliebige Konstante ist. Also beträgt die Optimierungszeit des Algorithmus (μ+1)-EA mit Älterwerden auf h mit Wahrscheinlichkeit $1 - 2^{-\Omega(n^{1-o(1)})}$ mindestens $2^{\Omega(n^\epsilon)}$.

2. Gemäß Theorem 11.8, schließt der Algorithmus (μ+1)-EA mit Älterwerden die Funktion $h_1 = g$ mit Wahrscheinlichkeit $1-o(1)$ innerhalb von $\mathcal{O}(n^{k_1+k_3} \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n))$ Generationen ab. Nachdem das erste Individuum x die Funktion $h_1 = g$ abgeschlossen hat, besteht die zweite Hälfte von x aus rein zufällig verteilten Bits. Von nun an wird die Fitness von x durch die Funktion $h_2 = f_k$ bestimmt. Gemäß Theorem 11.3, schließt der Algorithmus (μ+1)-EA mit Älterwerden die Funktion $h_2 = f_k$ mit Wahrscheinlichkeit $\Omega(1)$ innerhalb von $\mathcal{O}(n^{k+1} + \mu \cdot n \cdot \log n)$ Generationen ab. Also beträgt die Optimierungszeit des Algorithmus (μ+1)-EA mit Älterwerden auf h mit Wahrscheinlichkeit $1 - o(1)$ höchstens

$$\mathcal{O}\Big((n^{k_1+k_3} \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n)) + (n^{k+1} + \mu \cdot n \cdot \log n)\Big) = \mathcal{O}\Big(n^{k_1+k_3} \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n)\Big).$$

3. Gemäß Theorem 11.7, schließt der Algorithmus (μ+1)-EA mit Älterwerden die Funktion $h_1 = g$ mit Wahrscheinlichkeit $1 - o(1)$ *nicht* innerhalb von $2^{\Omega(\sqrt{n})}$ Generationen ab. Also beträgt die Optimierungszeit des Algorithmus (μ+1)-EA mit Älterwerden auf h mit Wahrscheinlichkeit $1 - o(1)$ mindestens $2^{\Omega(\sqrt{n})}$. □

Wir veranschaulichen das letzte Theorem. Betrachte z. B. $k_1 = 3$, $k_2 = 5$, $k_3 = 2$ und $k = 2$. Laut Theorem 11.9 garantiert eine Lebensspanne τ mit $\tau = \omega((n^2 + \mu \cdot \log n) \cdot \log n)$ und $\tau = o(n^3/(\log^2 n))$, dass die polynomielle Schranke $\mathcal{O}(n^5 \cdot (\mu \cdot n \cdot \log^4 n + n^2 \cdot \log^3 n + \tau \cdot \log^2 n))$ für die Optimierungszeit mit Wahrscheinlichkeit $1 - o(1)$ eingehalten wird. Wenn für die Populationsgröße $\mu = \mathcal{O}(n^2/(\log n))$ gilt, vereinfachen sich diese Anforderungen zu $\tau = \omega(n^2 \cdot \log n)$ und $\tau = o(n^3/(\log^2 n))$. Wenn $\mu = \Omega(n^2/(\log n))$ erfüllt ist, vereinfachen sie sich zu $\tau = \omega(\mu \cdot \log^2 n)$ und $\tau = o(n^3/(\log^2 n))$. Damit wir eine solche Lebensspanne τ wählen können, muss für die Populationsgröße μ natürlich $o(n^3/(\log^4 n))$ gelten. Wenn z. B. $\mu = \Theta(n^{3-\epsilon}/(\log^4 n))$ zutrifft, vereinfachen sich die Anforderungen zu $\tau = \omega(n^{3-\epsilon}/(\log^2 n))$ und $\tau = o(n^3/(\log^2 n))$ für beliebig kleine Konstanten $0 < \epsilon < 1$. Andererseits stellt Theorem 11.9 sicher, dass die Optimierungszeit mit überwältigender Wahrscheinlichkeit exponentiell groß ist, wenn $\tau = o(n^2 \cdot \log n)$ oder $\omega(n^5)$ gilt. Die Wahl einer geeigneten Lebensspanne τ kann also potenziell recht schwierig sein.

11.6. Fazit

Wir haben gesehen, dass die Wahl der Lebensspanne τ darüber entscheiden kann, ob die Optimierungszeit polynomiell oder exponentiell groß ist. Außerdem haben wir gesehen, dass der Bereich, der geeignete Werte für τ enthält, extrem klein sein kann. Daher kann die Wahl einer geeigneten Lebensspanne τ sowohl essenziell als auch schwierig sein.

Unsere Ergebnisse für die Funktion mit zwei Pfaden gelten nur mit Wahrscheinlichkeit $1 - o(1)$. Dies ist hier der Fall, da das Erreichen des korrekten Pfades mit Wahrscheinlichkeit $\Omega(1/p(n))$ für ein Polynom p möglich sein muss. Folglich ist ein einfacher evolutionärer Algorithmus mit Neustarts auch effizient. Wünschenswert wären Ergebnisse, die mit einer Wahrscheinlichkeit gelten, die exponentiell schnell gegen 1 konvergiert. Es ist ein offenes Problem, eine Beispielfunktion zu konstruieren und zu zeigen, dass sie nur dann effizient optimiert werden kann, wenn ein Alterungsoperator eingesetzt wird. Zwischenzeitlich konnten Jansen und Zarges [2010a] einen Schritt in diese Richtung machen. Sie haben eine Beispielfunktion konstruiert und bewiesen, dass sie die gewünschten Eigenschaften besitzt, wenn den hier betrachteten Algorithmen zusätzlich die Möglichkeit gegeben wird, Individuen mithilfe eines Rekombinationsoperators zu kreieren (siehe auch Abschnitt 11.4).

Die in diesem Kapitel gezeigten Ergebnisse gelten zudem nur, wenn die Populationsgröße μ beschränkt ist. Es ist eine offene Frage, ob polynomielle Optimierungszeiten auch ohne einen Alterungsoperator erreicht werden können, wenn eine viel größere Population zum Einsatz kommt. Außerdem ist ungewiss, ob eine zu große Lebensspanne τ durch eine größere Population kompensiert werden kann.

Schließlich funktioniert unsere Methode, um Funktionen zu kombinieren, nicht für Funktionen, die Neustarts erfordern. Das liegt daran, dass solche Neustarts globale Auswirkungen haben, die sich nicht nur auf die Funktion beziehen, die aktuell den Funktionswert definiert. Es ist ein offenes Problem, eine allgemeine Methode zu beschreiben, die auch für diese Fälle funktioniert.

Teil IV.

Anhang und Verzeichnisse

A. Mathematische Grundlagen

Wir stellen in diesem Kapitel die wichtigsten mathematischen Hilfsmittel vor, die wir in der vorliegenden Dissertation benötigen.

A.1. Wahrscheinlichkeitstheorie

In diesem Abschnitt präsentieren wir einige Ergebnisse aus dem Gebiet „Wahrscheinlichkeitstheorie“, die wir häufig benutzen. Sie sind beispielsweise in [Feller, 1968, 1971, Motwani und Raghavan, 1995, Mitzenmacher und Upfal, 2005] zu finden. Wir gehen im Folgenden davon aus, dass sich unsere Betrachtungen implizit auf einen Wahrscheinlichkeitsraum beziehen, der aus einem Ergebnisraum Ω, einer Ereignisalgebra $\Sigma \subseteq \mathrm{Pot}(\Omega)$ und einem Wahrscheinlichkeitsmaß $\mathrm{W}\colon \Sigma \to [0,1]$ besteht.

Definition A.1 (Bedingte Wahrscheinlichkeit)**.** *Seien E_1 und E_2 Ereignisse mit $\mathrm{W}(E_2) > 0$. Dann nennen wir*

$$\mathrm{W}(E_1 \mid E_2) := \frac{\mathrm{W}(E_1 \cap E_2)}{\mathrm{W}(E_2)}$$

die Wahrscheinlichkeit des Ereignisses E_1 unter der Bedingung, dass das Ereignis E_2 eingetreten ist.

Lemma A.2 (Multiplikationssatz)**.** *Seien E_i, $1 \leq i \leq n$, Ereignisse mit $\mathrm{W}(E_1 \cap \cdots \cap E_n) > 0$. Dann gilt*

$$\mathrm{W}(E_1 \cap \cdots \cap E_n) = \prod_{i=1}^{n} \mathrm{W}(E_i \mid E_1 \cap \cdots \cap E_{i-1}).$$

Lemma A.3 (Satz von der totalen Wahrscheinlichkeit)**.** *Seien E_i, $1 \leq i \leq n$, paarweise disjunkte Ereignisse mit $\mathrm{W}(E_i) > 0$ und sei E ein Ereignis mit $E \subseteq E_1 \cup \cdots \cup E_n$. Dann gilt*

$$\mathrm{W}(E) = \sum_{i=1}^{n} \mathrm{W}(E \cap E_i) = \sum_{i=1}^{n} \mathrm{W}(E \mid E_i) \cdot \mathrm{W}(E_i).$$

Lemma A.4 (Boolesche Ungleichung)**.** *Seien E_i, $1 \leq i \leq n$, Ereignisse. Dann gilt*

$$\mathrm{W}(E_1 \cup \cdots \cup E_n) \leq \sum_{i=1}^{n} \mathrm{W}(E_i).$$

Lemma A.5 (Markoff-Ungleichung). *Sei X eine nicht negative Zufallsvariable. Für $t \in \mathbb{R}^+$ gilt dann*

$$\mathrm{W}(X \geq t \cdot \mathrm{E}(X)) \leq \frac{1}{t}.$$

Lemma A.6 (Chernoff-Ungleichungen). *Seien X_i mit $1 \leq i \leq n$ unabhängige Bernoulli-verteilte Zufallsvariable mit $\mathrm{W}(X_i = 1) = p_i$ und $\mathrm{W}(X_i = 0) = 1 - p_i$. Seien $X := \sum_{i=1}^{n} X_i$ und $\mu := \mathrm{E}(X) = \sum_{i=1}^{n} p_i$. Für $\delta \in \mathbb{R}$ mit $0 < \delta \leq 1$ gelten dann*

$$\mathrm{W}(X \leq (1-\delta) \cdot \mu) \leq e^{-\delta^2 \cdot \mu/2},$$
$$\mathrm{W}(X \geq (1+\delta) \cdot \mu) \leq e^{-\delta^2 \cdot \mu/3}.$$

Lemma A.7 (Sammelbilderproblem). *Angenommen, es gibt n verschiedene Typen von Sammelbildern. Es bezeichne X die Zeit, die man benötigt, um von jedem Typ mindestens ein Sammelbild zu haben, wenn man zu jedem Zeitpunkt unabhängig ein Sammelbild erhält, das jedem Typ mit Wahrscheinlichkeit $1/n$ entspricht. Dann gelten*

$$\mathrm{E}(X) = n \cdot H_n,$$

wobei H_n die n-te harmonische Zahl bezeichnet, und

$$\lim_{n \to \infty} \mathrm{W}(X \leq n \cdot (\ln n - c)) = e^{-e^c}$$

für jede Konstante $c \in \mathbb{R}$.

A.2. Sonstiges

In diesem Abschnitt fassen wir alle mathematischen Hilfsmittel zusammen, die nicht in das Gebiet „Wahrscheinlichkeitstheorie“ fallen. Die hier angegebenen Ergebnisse sind beispielsweise in [Graham et al., 1994, Zeidler, 2003, Grosche et al., 2003] zu finden.

Die folgenden Sprechweisen aus dem Gebiet „Ordnungstheorie“ finden in dieser Dissertation Verwendung.

Definition A.8 (Eigenschaften von Relationen). *Sei M eine Menge. Eine Relation $R \subseteq M \times M$ heißt*

- reflexiv *genau dann, wenn* $\forall x \in M \colon (x, x) \in R$,
- symmetrisch *genau dann, wenn* $\forall x, y \in M \colon [(x, y) \in R \implies (y, x) \in R]$,
- transitiv *genau dann, wenn* $\forall x, y, z \in M \colon [(x, y) \in R \wedge (y, z) \in R \implies (x, z) \in R]$,
- antisymmetrisch *genau dann, wenn* $\forall x, y \in M \colon [(x, y) \in R \wedge (y, x) \in R \implies x = y]$, *und*

- total *genau dann, wenn* $\forall x, y \in M \colon [(x, y) \in R \vee (y, x) \in R]$.

Definition A.9 (Ordnungsrelationen)**.** *Sei M eine Menge. Eine Relation $R \subseteq M \times M$ heißt*

- Quasiordnung *genau dann, wenn R reflexiv und transitiv ist,*
- Halbordnung *genau dann, wenn R reflexiv, transitiv und antisymmetrisch ist, und*
- Totalordnung *genau dann, wenn R reflexiv, transitiv, antisymmetrisch und total ist.*

Die folgenden Aussagen über Fakultäten und Binomialkoeffizienten sind hilfreich.

Lemma A.10 (Stirling-Formel)**.** *Sei $n \in \mathbb{N}^+$. Dann gelten*

$$\lim_{n\to\infty} \frac{n!}{\sqrt{2 \cdot \pi \cdot n} \cdot (n/e)^n} = 1$$

und

$$\sqrt{2 \cdot \pi \cdot n} \cdot \left(\frac{n}{e}\right)^n \cdot e^{1/(12 \cdot n+1)} \leq n! \leq \sqrt{2 \cdot \pi \cdot n} \cdot \left(\frac{n}{e}\right)^n \cdot e^{1/(12 \cdot n)}.$$

Definition A.11 (Binomialkoeffizient)**.** *Seien $n \in \mathbb{N}$ und $k \in \mathbb{N}$ mit $0 \leq k \leq n$. Dann definieren wir den Binomialkoeffizienten*

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}.$$

Lemma A.12. *Seien $n \in \mathbb{N}$ und $k \in \mathbb{N}$ mit $0 \leq k \leq n$. Dann gilt*

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \frac{n^k}{k!} \leq \left(\frac{e \cdot n}{k}\right)^k.$$

Ein Beweis des folgenden Lemmas ist beispielsweise in [Flum und Grohe, 2006] zu finden.

Lemma A.13. *Seien $n \in \mathbb{N}^+$ und $p \in \mathbb{R}$ mit $0 < p \leq 1/2$. Dann gilt*

$$\sum_{i=0}^{\lfloor p \cdot n \rfloor} \binom{n}{i} \leq 2^{H(p) \cdot n},$$

wobei $H(p) := -p \cdot \log p - (1-p) \cdot \log(1-p)$ die binäre Entropie von p bezeichnet.

Die folgenden Aussagen über Reihen und die darauffolgenden Ungleichungen benutzen wir in zahlreichen Rechnungen.

Lemma A.14 (Geometrische Reihe)**.** *Seien $n \in \mathbb{N}$ und $q \in \mathbb{R}$ mit $q \neq 1$. Dann gilt*

$$\sum_{i=0}^{n} q^i = \frac{1 - q^{n+1}}{1 - q}.$$

Lemma A.15. *Es gilt* $\sum_{i=0}^{\infty} 1/i! = e$.

Lemma A.16. *Sei* $x \in \mathbb{R}$ *mit* $x > 1$. *Dann gelten*

$$\lim_{x\to\infty} \left(1 - \frac{1}{x}\right)^x = \frac{1}{e}$$

und

$$\left(1 - \frac{1}{x}\right)^x \leq \frac{1}{e} \leq \left(1 - \frac{1}{x}\right)^{x-1}.$$

Lemma A.17 (Harmonische Zahlen)**.** *Für die durch* $H_n := \sum_{i=1}^{n} 1/i$ *definierte* n*-te harmonische Zahl* H_n *gelten*

$$\lim_{n\to\infty} (H_n - \ln n) = \gamma$$

und

$$\ln(n) \leq H_n \leq \ln(n) + 1,$$

wobei $\gamma \approx 0{,}577$ *die Euler-Mascheroni-Konstante bezeichnet.*

Lemma A.18 (Bernoullische Ungleichung)**.** *Sei* $x \in \mathbb{R}$ *mit* $x \geq -1$. *Dann gilt*

$$(1 + x)^r \geq 1 + r \cdot x$$

für alle $r \in \mathbb{N}$. *Außerdem gelten* $(1 + x)^r \geq 1 + r \cdot x$ *für alle* $r \in \mathbb{R}$ *mit* $r \geq 1$ *und* $(1 + x)^r \leq 1 + r \cdot x$ *für alle* $r \in \mathbb{R}$ *mit* $0 \leq r \leq 1$.

Korollar A.19. *Sei* $x \in \mathbb{R}$. *Dann gilt*

$$1 + x \leq e^x.$$

Wir beenden dieses Kapitel mit einigen wichtigen Definitionen und Sprechweisen.

Definition A.20 (Landau-Symbole)**.** *Sei* $f\colon \mathbb{N} \to \mathbb{R}^+$. *Dann definieren wir*

$$\begin{aligned}
\mathrm{o}(f(n)) &:= \{g\colon \mathbb{N} \to \mathbb{R}^+ \mid \forall d \in \mathbb{R}^+\colon \exists n_0 \in \mathbb{N}\colon \forall n \geq n_0\colon g(n) \leq d \cdot f(n)\},\\
\mathcal{O}(f(n)) &:= \{g\colon \mathbb{N} \to \mathbb{R}^+ \mid \exists d \in \mathbb{R}^+\colon \exists n_0 \in \mathbb{N}\colon \forall n \geq n_0\colon g(n) \leq d \cdot f(n)\},\\
\Theta(f(n)) &:= \{g\colon \mathbb{N} \to \mathbb{R}^+ \mid \exists c, d \in \mathbb{R}^+\colon \exists n_0 \in \mathbb{N}\colon \forall n \geq n_0\colon c \cdot f(n) \leq g(n) \leq d \cdot f(n)\},\\
\Omega(f(n)) &:= \{g\colon \mathbb{N} \to \mathbb{R}^+ \mid \exists c \in \mathbb{R}^+\colon \exists n_0 \in \mathbb{N}\colon \forall n \geq n_0\colon c \cdot f(n) \leq g(n)\},\\
\omega(f(n)) &:= \{g\colon \mathbb{N} \to \mathbb{R}^+ \mid \forall c \in \mathbb{R}^+\colon \exists n_0 \in \mathbb{N}\colon \forall n \geq n_0\colon c \cdot f(n) \leq g(n)\}.
\end{aligned}$$

Wir weisen in diesem Zusammenhang darauf hin, dass wir die Definitionen wie folgt gebrauchen. Wir schreiben beispielsweise anstelle von $3 \cdot n \in \mathcal{O}(n) \subseteq \mathrm{o}(n^2)$ einfach $3 \cdot n = \mathcal{O}(n) = \mathrm{o}(n^2)$, da Missverständnisse in diesem Kontext ausgeschlossen sind.

Definition A.21 (Wachstumsklassen)**.** *Sei* $f\colon \mathbb{N} \to \mathbb{R}^+$. *Dann sagen wir*

- f *ist* exponentiell klein *genau dann, wenn* $\exists c \in \mathbb{R}^+ \colon f = \exp(-\Omega(n^c))$,
- f *ist* polynomiell klein *genau dann, wenn* $\exists c \in \mathbb{R}^+ \colon f = \mathcal{O}(n^{-c})$,
- f *ist* polynomiell (groß) *genau dann, wenn* $\exists c \in \mathbb{R}^+ \colon f = \mathcal{O}(n^c)$, *und*
- f *ist* exponentiell (groß) *genau dann, wenn* $\exists c \in \mathbb{R}^+ \colon f = \exp(\Omega(n^c))$.

Definition A.22 (Wachstumsklassen für Wahrscheinlichkeiten)**.** *Sei* $(w_n)_{n\in\mathbb{N}}$ *mit* $w_n \in [0,1]$ *eine Folge von Wahrscheinlichkeiten, die das Eintreten eines Ereignisses charakterisiert. Dann sagen wir*

- *das Ereignis tritt* mit hoher Wahrscheinlichkeit (m. h. W.) *ein genau dann, wenn* $1 - w_n$ *polynomiell klein ist und*
- *das Ereignis tritt* mit überwältigender Wahrscheinlichkeit (m. ü. W.) *ein genau dann, wenn* $1 - w_n$ *exponentiell klein ist*

B. Symbolverzeichnis

Bekannte Beispielfunktionen:

$\mathrm{BinVal}(x)$	Beispielfunktion BinVal: $\mathbb{B}^n \to \mathbb{R}$, definiert durch $\mathrm{BinVal}(x = (x_i)) = \sum_{i=1}^{n} x_i \cdot 2^{n-i}$
$\mathrm{LeadingOnes}(x)$	Beispielfunktion LeadingOnes: $\mathbb{B}^n \to \mathbb{R}$, definiert durch $\mathrm{LeadingOnes}(x = (x_i)) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$
$\mathrm{Needle}(x)$	Beispielfunktion Needle: $\mathbb{B}^n \to \mathbb{R}$, definiert durch $\mathrm{Needle}(x = (x_i)) = \prod_{i=1}^{n} x_i$
$\mathrm{OneMax}(x)$ ...	Beispielfunktion OneMax: $\mathbb{B}^n \to \mathbb{R}$, definiert durch $\mathrm{OneMax}(x = (x_i)) = \sum_{i=1}^{n} x_i$
$\mathrm{Ridge}(x)$	Beispielfunktion Ridge: $\mathbb{B}^n \to \mathbb{R}$, definiert durch $\mathrm{Ridge}(x = (x_i)) = \mathrm{OneMax}(\overline{x}) + \sum_{i=0}^{n} \prod_{j=1}^{i} x_j \cdot \prod_{j=i+1}^{n} \overline{x}_j \cdot 2i$
$\mathrm{Trap}(x)$	Beispielfunktion Trap: $\mathbb{B}^n \to \mathbb{R}$, definiert durch $\mathrm{Trap}(x = (x_i)) = \mathrm{OneMax}(x) + \mathrm{Needle}(\overline{x}) \cdot 2n$

Mathematische Funktionen:

$\exp(x)$	Exponentialfunktion e^x zur Basis e
$\ln(x)$	Logarithmus zur Basis e
$\log(x)$	Logarithmus zur Basis 2
$\mathrm{H}(x, y)$	Hammingabstand H: $\mathbb{B}^n \times \mathbb{B}^n \to \{0, \ldots, n\}$, definiert durch $\mathrm{H}(x = (x_i), y = (y_i)) = \sum_{i=1}^{n} \lvert y_i - x_i \rvert$
$\overline{x}$	Komplement $\overline{x} = (1 - x_i) \in \mathbb{B}^n$ von $x = (x_i) \in \mathbb{B}^n$
H_n	n-te harmonische Zahl

Mathematische Konstanten:

γ	Euler-Mascheroni-Konstante $\gamma \approx 0{,}577$
e	eulersche Zahl $e \approx 2{,}718$
π	Kreiszahl $\pi \approx 3{,}142$

Landau-Symbole:

$f = \mathrm{o}(g)$ ..... f wächst asymptotisch langsamer als g

$f = \mathcal{O}(g)$ ..... f wächst asymptotisch nicht schneller als g

$f = \Theta(g)$ ..... f wächst asymptotisch genauso schnell wie g

$f = \Omega(g)$ ..... f wächst asymptotisch nicht langsamer als g

$f = \omega(g)$ ..... f wächst asymptotisch schneller als g

$\mathrm{poly}(n)$ ..... $n^{\mathcal{O}(1)}$ (polynomiell in n beschränkt)

Mathematische Mengen:

$\emptyset$ ..... leere Menge

$\mathbb{B}$ ..... Menge $\{0, 1\}$ der binären Zahlen

$\mathbb{N}^+$ ..... Menge $\{1, 2, 3, \dots\}$ der positiven natürlichen Zahlen

$\mathbb{N}$ ..... Menge $\{0, 1, 2, \dots\}$ der natürlichen Zahlen

$\mathbb{Z}$ ..... Menge $\{\dots, -1, 0, 1, \dots\}$ der ganzen Zahlen

$\mathbb{R}^+$ ..... Menge der positiven reellen Zahlen

$\mathbb{R}_0^+$ ..... Menge der nicht negativen reellen Zahlen

$\mathbb{R}$ ..... Menge der reellen Zahlen

(a, b) ..... offenes reellwertiges Intervall mit den Grenzen a und b

$[a, b]$ ..... abgeschlossenes reellwertiges Intervall mit den Grenzen a und b

$\mathrm{Pot}(M)$ ..... Potenzmenge der Menge M, z. B. $\mathrm{Pot}(\mathbb{B}) = \{\emptyset, \{0\}, \{1\}, \mathbb{B}\}$

Wichtige Parameter:

μ ..... Populationsgröße $\mu \in \mathbb{N}^+$

ρ ..... Evaporationsfaktor $\rho \in \mathbb{R}$ mit $0 < \rho \leq 1$

$\tau_{\min}$ ..... Untere Schranke $\tau_{\min} \in \mathbb{R}$ mit $\tau_{\min} > 0$ für Pheromone (hier: $0 < \tau_{\min} < 1/2$)

$\tau_{\max}$ ..... Obere Schranke $\tau_{\max} \in \mathbb{R}$ mit $\tau_{\max} > \tau_{\min}$ für Pheromone (hier: $\tau_{\max} = 1 - \tau_{\min}$)

τ ..... Lebensspanne $\tau \in \mathbb{N} \cup \{\infty\}$

Qualitätsindikatoren:

$\mathrm{I}_{\mathrm{fro}}(\cdot)$ ..... Qualitätsindikator „Anteil der Paretofront“, def. durch
$\mathrm{I}_{\mathrm{fro}}(A) = |\{x \in \max(f(S), \succeq) \mid \exists a \in A\colon f(a) \succeq x\}| / |\max(f(S), \succeq)|$

$\mathrm{I}_{\mathrm{mul}}(\cdot)$ ..... Qualitätsindikator „multiplikative Approximationsgüte“, def. durch
$\mathrm{I}_{\mathrm{mul}}(A) = \inf\{\epsilon \in \mathbb{R} \mid \forall x \in \max(f(S), \succeq)\colon \exists a \in A\colon f(a) \succeq_{\epsilon,\cdot} x\}$

$\mathrm{I}_{\mathrm{add}}(\cdot)$ ..... Qualitätsindikator „additive Approximationsgüte“, def. durch
$\mathrm{I}_{\mathrm{add}}(A) = \inf\{\epsilon \in \mathbb{R} \mid \forall x \in \max(f(S), \succeq)\colon \exists a \in A\colon f(a) \succeq_{\epsilon,+} x\}$

$\mathrm{I}_{\mathrm{hyp}}(\cdot)$ ..... Qualitätsindikator „Hypervolumen“, def. durch
$\mathrm{I}_{\mathrm{hyp}}(A) = \mathrm{Vol}(\bigcup_{(a_1,\ldots,a_d)\in f(A)} [r_1, a_1] \times \cdots \times [r_d, a_d])$

Wichtige Relationen:

$\succeq$ ..... $(x_1, \ldots, x_d) \succeq (x'_1, \ldots, x'_d) \iff \forall i \in \{1, \ldots, d\}\colon x_i \geq x'_i$

$\succeq_{\epsilon,\cdot}$ ..... $(x_1, \ldots, x_d) \succeq_{\epsilon,\cdot} (x'_1, \ldots, x'_d) \iff \forall i \in \{1, \ldots, d\}\colon (1+\epsilon) \cdot x_i \geq x'_i$

$\succeq_{\epsilon,+}$ ..... $(x_1, \ldots, x_d) \succeq_{\epsilon,+} (x'_1, \ldots, x'_d) \iff \forall i \in \{1, \ldots, d\}\colon x_i + \epsilon \geq x'_i$

$\succeq_{\delta}$ ..... $(x_1, \ldots, x_d) \succeq_{\delta-1,\cdot} (x'_1, \ldots, x'_d)$ bzw. $(x_1, \ldots, x_d) \succeq_{\delta,+} (x'_1, \ldots, x'_d)$

$\succeq_I$ ..... $A \succeq_I B \iff I(f(A)) \geq I(f(B))$

Wahrscheinlichkeitstheorie:

$\mathrm{W}(E)$ ..... Wahrscheinlichkeit des Ereignisses E

$\mathrm{W}(E_1 \mid E_2)$ ... Wahrscheinlichkeit des Ereignisses E_1 unter der Bedingung, dass das Ereignis E_2 eingetreten ist

$\mathrm{E}(X)$ ..... Erwartungswert der Zufallsvariablen X

$\mathrm{Var}(X)$ ..... Varianz der Zufallsvariablen X

C. Abbildungsverzeichnis

D. Literaturverzeichnis

Nattapat Attiratanasunthron und Jittat Fakcharoenphol. A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs. *Information Processing Letters*, 105(3):88–92, 2008.

Anne Auger, Johannes Bader, Dimo Brockhoff und Eckart Zitzler. Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point. In *10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, Seiten 87–102. ACM Press, 2009a.

Anne Auger, Johannes Bader, Dimo Brockhoff und Eckart Zitzler. Articulating user preferences in many-objective problems by sampling the weighted hypervolume. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 555–562. ACM Press, 2009b.

Anne Auger, Johannes Bader, Dimo Brockhoff und Eckart Zitzler. Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 563–570. ACM Press, 2009c.

Holger Bast, Stefan Funke, Peter Sanders und Dominik Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.

Surender Baswana, Somenath Biswas, Benjamin Doerr, Tobias Friedrich, Piyush P. Kurur und Frank Neumann. Computing single source shortest paths using single-objective fitness functions. In *10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, Seiten 59–66. ACM Press, 2009.

Thomas Bäck, David B. Fogel und Zbigniew Michalewicz, Herausgeber. *Handbook of Evolutionary Computation*. IOP Publishing und Oxford University Press, 1997.

Richard E. Bellman und Stuart E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, erste Auflage, 1962.

Dimitri P. Bertsekas und John N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

Nicola Beume und Günter Rudolph. Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem. In *2nd IASTED International Conference on Computational Intelligence (CI '06)*, Seiten 231–236. ACTA Press, 2006.

Nicola Beume, Boris Naujoks und Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

Justin A. Boyan und Michael Mitzenmacher. Improved results for route planning in stochastic transportation. In *12th annual ACM-SIAM Symposium on Discrete Algorithms (SODA '01)*, Seiten 895–902. SIAM, 2001.

Karl Bringmann und Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In *19th International Symposium on Algorithms and Computation (ISAAC '08)*, Band 5369 aus *Lecture Notes in Computer Science*, Seiten 436–447. Springer-Verlag, 2008.

Karl Bringmann und Tobias Friedrich. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. In *5th International Conference on Evolutionary Multi-Criterion Optimization (EMO '09)*, Band 5467 aus *Lecture Notes in Computer Science*, Seiten 6–20. Springer-Verlag, 2009.

Karl Bringmann und Tobias Friedrich. The maximum hypervolume set yields near-optimal approximation. In *12th Genetic and Evolutionary Computation Conference (GECCO '10)*. ACM Press, 2010. Erscheint.

Mario Castrogiovanni, Giuseppe Nicosia und Rosario Rascunà. Experimental analysis of the aging operator for static and dynamic optimisation problems. In *11th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES '07)*, Band 4694 aus *Lecture Notes in Computer Science*, Seiten 804–811. Springer-Verlag, 2007.

Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *39th Annual ACM Symposium on Theory of Computing (STOC '07)*, Seiten 590–598. ACM Press, 2007.

Doo-Hyun Choi. Cooperative mutation based evolutionary programming for continuous function optimization. *Operations Research Letters*, 30(3):195–201, 2002.

Carlos A. Coello Coello, David A. Van Veldhuizen und Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer-Verlag, zweite Auflage, 2007.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest und Clifford Stein. *Introduction to Algorithms*. MIT Press, zweite Auflage, 2001.

Vincenzo Cutello, Giuseppe Nicosia und Mario Pavone. Exploring the capability of immune algorithms: A characterization of hypermutation operators. In *3rd International Conference on Artificial Immune Systems (ICARIS '04)*, Band 3239 aus *Lecture Notes in Computer Science*, Seiten 263–276. Springer-Verlag, 2004.

Vincenzo Cutello, Giuseppe Nicosia, Mario Pavone und Jonathan Timmis. An immune algorithm for protein structure prediction on lattice models. *IEEE Transactions on Evolutionary Computation*, 11(1):101–117, 2007a.

Vincenzo Cutello, Giuseppe Nicosia, Mario Romeo und Pietro S. Oliveto. On the convergence of immune algorithms. In *1st IEEE Symposium on Foundations of Computational Intelligence (FOCI '07)*, Seiten 409–415. IEEE Press, 2007b.

Leandro N. de Castro und Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, erste Auflage, 2002.

Leandro N. de Castro und Fernando J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, 2002.

Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, erste Auflage, 2001.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap und T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *6th International Conference on Parallel Problem Solving from Nature (PPSN '00)*, Band 1917 aus *Lecture Notes in Computer Science*, Seiten 849–858. Springer-Verlag, 2000.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns und Eckart Zitzler. Scalable multi-objective optimization test problems. In *4th IEEE Congress on Evolutionary Computation (CEC '02)*, Seiten 825–830. IEEE Press, 2002.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns und Eckart Zitzler. *Evolutionary Multiobjective Optimization*, Kapitel Scalable Test Problems for Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, Seiten 105–145. Advanced Information and Knowledge Processing. Springer-Verlag, 2005.

Gianni Di Caro und Marco Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

Benjamin Doerr und Daniel Johannsen. Adjacency list matchings – an ideal genotype for cycle covers. In *9th Genetic and Evolutionary Computation Conference (GECCO '07)*, Seiten 1203–1210. ACM Press, 2007.

Benjamin Doerr und Madeleine Theile. Improved analysis methods for crossover-based algorithms. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 247–254. ACM Press, 2009.

Benjamin Doerr, Edda Happ und Christian Klein. A tight analysis of the (1+1)-EA for the single source shortest path problem. In *9th IEEE Congress on Evolutionary Computation (CEC '07)*, Seiten 1890–1895. IEEE Press, 2007a.

Benjamin Doerr, Nils Hebbinghaus und Frank Neumann. Speeding up evolutionary algorithms through asymmetric mutation operators. *Evolutionary Computation*, 15(4):401–410, 2007b.

Benjamin Doerr, Christian Klein und Tobias Storch. Faster evolutionary algorithms by superior graph representations. In *1st IEEE Symposium on Foundations of Computational Intelligence (FOCI '07)*, Seiten 245–250. IEEE Press, 2007c.

Benjamin Doerr, Frank Neumann, Dirk Sudholt und Carsten Witt. On the runtime analysis of the 1-ANT ACO algorithm. In *9th Genetic and Evolutionary Computation Conference (GECCO '07)*, Seiten 33–40. ACM Press, 2007d.

Benjamin Doerr, Edda Happ und Christian Klein. Crossover can provably be useful in evolutionary computation. In *10th Genetic and Evolutionary Computation Conference (GECCO '08)*, Seiten 539–546. ACM Press, 2008a.

Benjamin Doerr, Thomas Jansen und Christian Klein. Comparing global and local mutations on bit strings. In *10th Genetic and Evolutionary Computation Conference (GECCO '08)*, Seiten 929–936. ACM Press, 2008b.

Benjamin Doerr, Anton Eremeev, Christian Horoba, Frank Neumann und Madeleine Theile. Evolutionary algorithms and dynamic programming. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 771–777. ACM Press, 2009.

Marco Dorigo und Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344:243–278, 2005.

Marco Dorigo und Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

Marco Dorigo und Thomas Stützle. *Ant Colony Optimization*. MIT Press, erste Auflage, 2004.

Marco Dorigo, Vittorio Maniezzo und Alberto Colorni. The ant system: An autocatalytic optimizing process. Technischer Bericht 91-016 Revised, Politecnico di Milano, Mailand, Italien, 1991.

Stefan Droste, Thomas Jansen und Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1–2):51–81, 2002.

Stefan Droste, Thomas Jansen und Ingo Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39(4):525–544, 2006.

Matthias Ehrgott. *Multicriteria Optimization*. Springer-Verlag, zweite Auflage, 2005.

Michael Emmerich, Nicola Beume und Boris Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In *3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO '05)*, Band 3410 aus *Lecture Notes in Computer Science*, Seiten 62–76. Springer-Verlag, 2005.

Yueyue Fan, Robert E. Kalaba und James E. Moore. Arriving on time. *Journal of Optimization Theory and Applications*, 127(3):497–513, 2005.

William Feller. *An Introduction to Probability Theory and its Applications*, Band 1. Wiley, dritte Auflage, 1968.

William Feller. *An Introduction to Probability Theory and its Applications*, Band 2. Wiley, zweite Auflage, 1971.

Simon Fischer und Ingo Wegener. The one-dimensional Ising model: Mutation versus recombination. *Theoretical Computer Science*, 344(2–3):208–225, 2005.

Roger Fletcher. *Practical Methods of Optimization*. Wiley, zweite Auflage, 1987.

Jörg Flum und Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, erste Auflage, 2006.

Tobias Friedrich, Nils Hebbinghaus und Frank Neumann. Plateaus can be harder in multi-objective optimization. In *9th IEEE Congress on Evolutionary Computation (CEC '07)*, Seiten 2622–2629. IEEE Press, 2007.

Tobias Friedrich, Christian Horoba und Frank Neumann. Runtime analyses for using fairness in evolutionary multi-objective optimization. In *10th International Conference on Parallel Problem Solving from Nature (PPSN '08)*, Band 5199 aus *Lecture Notes in Computer Science*, Seiten 671–680. Springer-Verlag, 2008a.

Tobias Friedrich, Pietro S. Oliveto, Dirk Sudholt und Carsten Witt. Theoretical analysis of diversity mechanisms for global exploration. In *10th Genetic and Evolutionary Computation Conference (GECCO '08)*, Seiten 945–952. ACM Press, 2008b.

Tobias Friedrich, Christian Horoba und Frank Neumann. Multiplicative approximations and the hypervolume indicator. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 571–578. ACM Press, 2009. Ausgezeichnet mit einem *Best-Paper-Award*.

Tobias Friedrich, Christian Horoba und Frank Neumann. Showcases of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science*, 2010. Erscheint.

Michael R. Garey und David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, erste Auflage, 1979.

Ashish Ghosh, Shigeyoshi Tsutsui und Hideo Tanaka. Individual aging in genetic algorithms. In *4th Australian New Zealand Conference on Intelligent Information Systems (ANZIIS '96)*, Seiten 276–279. IEEE Press, 1996.

Oliver Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *5th IEEE Congress on Evolutionary Computation (CEC '03)*, Seiten 1918–1925. IEEE Press, 2003.

Oliver Giel und Ingo Wegener. Evolutionary algorithms and the maximum matching problem. In *20th Symposium on Theoretical Aspects of Computer Science (STACS '03)*, Band 2607 aus *Lecture Notes in Computer Science*, Seiten 415–426. Springer-Verlag, 2003.

Ronald L. Graham, Donald E. Knuth und Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, zweite Auflage, 1994.

Günter Grosche, Viktor Ziegler, Eberhard Zeidler und Dorothea Ziegler, Herausgeber. *Teubner-Taschenbuch der Mathematik*, Band 2. B. G. Teubner, achte Auflage, 2003.

Walter J. Gutjahr. A generalized convergence result for the graph-based ant system metaheuristic. *Probability in the Engineering and Informational Sciences*, 17:545–569, 2003.

Walter J. Gutjahr. Mathematical runtime analysis of ACO algorithms: Survey on an emerging issue. *Swarm Intelligence*, 1(1):59–79, 2007.

Walter J. Gutjahr. First steps to the runtime complexity analysis of ant colony optimization. *Computers and Operations Research*, 35(9):2711–2727, 2008.

Walter J. Gutjahr und Giovanni Sebastiani. Runtime analysis of ant colony optimization with best-so-far reinforcement. *Methodology and Computing in Applied Probability*, 10:409–433, 2008.

Bruce Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14:502–525, 1982.

P. Hansen. Bicriterion path problems. In *International Conference on Multiple Criteria Decision Making (MCDM '79)*, Band 177 aus *Lecture Notes in Economics and Mathematical Systems*, Seiten 109–127. Springer-Verlag, 1979.

Edda Happ, Daniel Johannsen, Christian Klein und Frank Neumann. Rigorous analyses of fitness-proportional selection for optimizing linear functions. In *10th Genetic and Evolutionary Computation Conference (GECCO '08)*, Seiten 953–960. ACM Press, 2008.

Jun He und Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

Jun He und Xin Yao. Erratum to: Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 140(1–2):245–248, 2002.

Jun He und Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

Michael Held und Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1): 196–210, 1962.

Jeffrey Horn, David E. Goldberg und Kalyanmoy Deb. Long path problems. In *3rd International Conference on Parallel Problem Solving from Nature (PPSN '94)*, Band 866 aus *Lecture Notes in Computer Science*, Seiten 149–158. Springer-Verlag, 1994.

Christian Horoba. Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In *10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, Seiten 113–120. ACM Press, 2009.

Christian Horoba. Exploring the runtime of an evolutionary algorithm for the multiobjective shortest path problem. *Evolutionary Computation*, 18(3), 2010. Erscheint.

Christian Horoba und Frank Neumann. Benefits and drawbacks for the use of ϵ-dominance in evolutionary multi-objective optimization. In *10th Genetic and Evolutionary Computation Conference (GECCO '08)*, Seiten 641–648. ACM Press, 2008. Nominiert für einen *Best-Paper-Award*.

Christian Horoba und Frank Neumann. Additive approximations of Pareto-optimal sets by evolutionary multi-objective algorithms. In *10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, Seiten 79–86. ACM Press, 2009.

Christian Horoba und Frank Neumann. *Advances in Multi-Objective Nature Inspired Computing*, Band 272 aus *Studies in Computational Intelligence*, Kapitel Approximating Pareto-Optimal Sets Using Diversity Strategies in Evolutionary Multi-Objective Optimization, Seiten 23–44. Springer-Verlag, 2010.

Christian Horoba und Dirk Sudholt. Running time analysis of ACO systems for shortest path problems. In *2nd International Workshop on Engineering Stochastic Local Search Algorithms (SLS '09)*, Band 5752 aus *Lecture Notes in Computer Science*, Seiten 76–91. Springer-Verlag, 2009.

Christian Horoba und Dirk Sudholt. Ant colony optimization for stochastic shortest path problems. In *12th Genetic and Evolutionary Computation Conference (GECCO '10)*. ACM Press, 2010. Nominiert für einen *Best-Paper-Award*. Erscheint.

Christian Horoba, Thomas Jansen und Christine Zarges. Maximal age in randomized search heuristics with aging. In *11th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 803–810. ACM Press, 2009. Nominiert für einen *Best-Paper-Award*.

Christian Igel, Nikolaus Hansen und Stefan Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

Thomas Jansen. On the analysis of dynamic restart strategies for evolutionary algorithms. In *7th International Conference on Parallel Problem Solving from Nature (PPSN '02)*, Band 2439 aus *Lecture Notes in Computer Science*, Seiten 33–43. Springer-Verlag, 2002.

Thomas Jansen und Ingo Wegener. Evolutionary algorithms – how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, 5(6):589–599, 2001.

Thomas Jansen und Ingo Wegener. On the analysis of evolutionary algorithms—a proof that crossover really can help. *Algorithmica*, 34(1):47–66, 2002.

Thomas Jansen und Ingo Wegener. Real royal road functions—where crossover provably is essential. *Discrete Applied Mathematics*, 149(1–3):111–125, 2005.

Thomas Jansen und Christine Zarges. A theoretical analysis of immune inspired somatic contiguous hypermutations for function optimization. In *8th International Conference on Artificial Immune Systems (ICARIS '09)*, Band 5666 aus *Lecture Notes in Computer Science*, Seiten 80–94. Springer-Verlag, 2009a.

Thomas Jansen und Christine Zarges. Comparing different aging operators. In *8th International Conference on Artificial Immune Systems (ICARIS '09)*, Band 5666 aus *Lecture Notes in Computer Science*, Seiten 95–108. Springer-Verlag, 2009b.

Thomas Jansen und Christine Zarges. Aging beyond restarts. In *12th Genetic and Evolutionary Computation Conference (GECCO '10)*. ACM Press, 2010a. Erscheint.

Thomas Jansen und Christine Zarges. On the benefits of aging and the importance of details. In *9th International Conference on Artificial Immune Systems (ICARIS '10)*. Springer-Verlag, 2010b. Erscheint.

Thomas Jansen und Christine Zarges. On benefits and drawbacks of aging strategies for randomized search heuristics. *Theoretical Computer Science*, 2010c. Erscheint.

David R. Karger, Daphne Koller und Steven J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. *SIAM Journal of Computing*, 22(6): 1199–1217, 1993.

Johnny Kelsey und Jonathan Timmis. Immune inspired somatic contiguous hypermutation for function optimisation. In *5th Genetic and Evolutionary Computation Conference (GECCO '03)*, Band 2723 aus *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

Scott Kirkpatrick, Charles D. Gelatt Jr. und Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. Dissertation, University of Reading, 2002.

Joshua D. Knowles und David W. Corne. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, 2003.

Joshua D. Knowles, David W. Corne und Mark Fleischer. Bounded archiving using the Lebesgue measure. In *5th IEEE Congress on Evolutionary Computation (CEC '03)*, Seiten 2490–2497. IEEE Press, 2003.

D. I. Kogan. Dynamic programming and discrete multicriterial optimization, 2004. Auf Russisch.

Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

Timo Kötzing, Per Kristian Lehre, Pietro S. Oliveto und Frank Neumann. Ant colony optimization and the minimum cut problem. In *12th Genetic and Evolutionary Computation Conference (GECCO '10)*. ACM Press, 2010. Erscheint.

Naoyuki Kubota und Toshio Fukuda. Genetic algorithms with age structure. *Soft Computing*, 1(4):155–161, 1997.

Marco Laumanns, Lothar Thiele, Kalyanmoy Deb und Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002a.

Marco Laumanns, Lothar Thiele, Eckart Zitzler, Emo Welzl und Kalyanmoy Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *7th International Conference on Parallel Problem Solving from Nature (PPSN '02)*, Band 2439 aus *Lecture Notes in Computer Science*, Seiten 44–53. Springer-Verlag, 2002b.

Marco Laumanns, Lothar Thiele und Eckart Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182, 2004.

Giovanni Lizzáraga-Lizzáraga, Arturo Hernández-Aguirre und Salvador Botello-Rionda. G-metric: an M-ary quality indicator for the evaluation of non-dominated sets. In *10th Genetic and Evolutionary Computation Conference (GECCO '09)*, Seiten 665–672. ACM Press, 2009.

Maplesoft. Maple 12, 2009. `http://www.maplesoft.com/products/Maple/`.

Daniel Merkle und Martin Middendorf. Modelling the dynamics of ant colony optimization algorithms. *Evolutionary Computation*, 10(3):235–262, 2002.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller und Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Zbigniew Michalewicz und David B. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, zweite Auflage, 2004.

Elise D. Miller-Hooks und Hani S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, 34(2):198–215, 2000.

Michael Mitzenmacher und Eli Upfal. *Probability and Computing*. Cambridge University Press, erste Auflage, 2005.

Matthias Müller-Hannemann und Karsten Weihe. On the cardinality of the Pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147(1): 269–286, 2006.

Rajeev Motwani und Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, erste Auflage, 1995.

Frank Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620–1629, 2007.

Frank Neumann. Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. *Computers and Operations Research*, 35(9):2750–2759, 2008.

Frank Neumann und Joachim Reichel. Approximating minimum multicuts by evolutionary multi-objective algorithms. In *10th International Conference on Parallel Problem Solving from Nature (PPSN '08)*, Band 5199 aus *Lecture Notes in Computer Science*, Seiten 72–81. Springer-Verlag, 2008.

Frank Neumann und Ingo Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

Frank Neumann und Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40, 2007.

Frank Neumann und Carsten Witt. Runtime analysis of a simple ant colony optimization algorithm. In *17th International Symposium on Algorithms and Computation (ISAAC '06)*, Band 4288 aus *Lecture Notes in Computer Science*, Seiten 618–627. Springer-Verlag, 2006.

Frank Neumann und Carsten Witt. Ant colony optimization and the minimum spanning tree problem. In *2nd International Conference on Learning and Intelligent Optimization (LION '07)*, Band 5313 aus *Lecture Notes in Computer Science*, Seiten 153–166. Springer-Verlag, 2008.

Frank Neumann und Carsten Witt. Runtime analysis of a simple ant colony optimization algorithm. *Algorithmica*, 54(2):243–255, 2009.

Frank Neumann, Dirk Sudholt und Carsten Witt. Rigorous analyses for the combination of ant colony optimization and local search. In *6th International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS '08)*, Band 5217 aus *Lecture Notes in Computer Science*, Seiten 132–143. Springer-Verlag, 2008.

Frank Neumann, Dirk Sudholt und Carsten Witt. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. *Swarm Intelligence*, 3(1):35–68, 2009.

Evdokia Nikolova, Matthew Brand und David R. Karger. Optimal route planning under uncertainty. In *16th International Conference on Automated Planning and Scheduling (ICAPS '06)*, Seiten 131–141. AAAI Press, 2006.

Pietro S. Oliveto und Carsten Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 2010. Erscheint.

Pietro S. Oliveto, Jun He und Xin Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007a.

Pietro S. Oliveto, Jun He und Xin Yao. Evolutionary algorithms and the vertex cover problem. In *9th IEEE Congress on Evolutionary Computation (CEC '07)*, Seiten 1870–1877. IEEE Press, 2007b.

Mark H. Overmars und Chee-Keng Yap. New upper bounds in Klee's measure problem. *SIAM Journal on Computing*, 20(6):1034–1045, 1991.

Christos H. Papadimitriou und Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

Christos H. Papadimitriou und Mihalis Yannakakis. On the approximability of tradeoffs and optimal access of web sources. In *41st Symposium on Foundations of Computer Science (FOCS '00)*, Seiten 86–92. IEEE Press, 2000.

Richard J. Quick, Victor J. Rayward-Smith und George D. Smith. Fitness distance correlation and ridge functions. In *8th International Conference on Parallel Problem Solving from Nature (PPSN '98)*, Band 1498 aus *Lecture Notes in Computer Science*, Seiten 77–86. Springer-Verlag, 1998.

Nicholas J. Radcliffe. Forma analysis and random respectful recombination. In *4th International Conference on Genetic Algorithms (ICGA '91)*, Seiten 222–229. Morgan Kaufmann, 1991.

Günther R. Raidl und Bryant A. Julstrom. Edge sets: An effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, 7(3): 225–239, 2003.

Andrea Raith und Matthias Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers and Operations Research*, 36(4): 1299–1331, 2009.

Günter Rudolph. How mutation and selection solve long-path problems in polynomial expected time. *Evolutionary Computation*, 4(2):195–205, 1996.

Günter Rudolph und Alexandru Agapie. Convergence properties of some multiobjective evolutionary algorithms. In *2nd IEEE Congress on Evolutionary Computation (CEC '00)*, Seiten 1010–1016. IEEE Press, 2000.

Jens Scharnow, Karsten Tinnefeld und Ingo Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, 3(4):349–366, 2004.

Hans-Paul Schwefel und Günter Rudolph. Contemporary evolution strategies. In *3rd European Conference on Artificial Life (ECAL '95)*, Band 929 aus *Lecture Notes in Computer Science*, Seiten 893–907. Springer-Verlag, 1995.

Thomas Stützle und Holger H. Hoos. MAX-MIN ant system. *Journal of Future Generation Computer Systems*, 16:889–914, 2000.

Dirk Sudholt. Crossover is provably essential for the Ising model on trees. In *7th Genetic and Evolutionary Computation Conference (GECCO '05)*, Seiten 1161–1167. ACM Press, 2005.

Dirk Sudholt. *Computational Complexity of Evolutionary Algorithms, Hybridizations, and Swarm Intelligence*. Dissertation, Technische Universität Dortmund, 2008.

Dirk Sudholt und Carsten Witt. Runtime analysis of Binary PSO. In *10th Genetic and Evolutionary Computation Conference (GECCO '08)*, Seiten 135–142. ACM Press, 2008.

Madeleine Theile. Exact solutions to the traveling salesperson problem by a population-based evolutionary algorithm. In *9th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP '09)*, Band 5482 aus *Lecture Notes in Computer Science*, Seiten 145–155. Springer-Verlag, 2009.

Jonathan Timmis, Andrew Hone, Thomas Stibor und Ed Clark. Theoretical advances in artificial immune systems. *Theoretical Computer Science*, 403(1):11–32, 2008.

George Tsaggouris und Christos D. Zaroliagis. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. In *17th International Symposium on Algorithms and Computation (ISAAC '06)*, Band 4288 aus *Lecture Notes in Computer Science*, Seiten 389–398. Springer-Verlag, 2006.

Berthold Ulungu, Jacques Teghem, Philippe Fortemps und Daniel Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

Virginia Vassilevska. Nondecreasing paths in a weighted graph or: How to optimally read a train schedule. In *19th annual ACM-SIAM Symposium on Discrete Algorithms (SODA '08)*, Seiten 465–472. SIAM, 2008.

A. Warburton. Approximation of Pareto optima in multiple-objective shortest path problems. *Operations Research*, 35(1):70–79, 1987.

Ingo Wegener. Towards a theory of randomized search heuristics. In *28th International Symposium on Mathematical Foundations of Computer Science (MFCS '03)*, Band 2747 aus *Lecture Notes in Computer Science*, Seiten 125–141. Springer-Verlag, 2003.

Carsten Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *22nd Symposium on Theoretical Aspects of Computer Science (STACS '05)*, Band 3404 aus *Lecture Notes in Computer Science*, Seiten 44–56. Springer-Verlag, 2005.

Carsten Witt. Runtime analysis of the (μ+1) EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14(1):65–86, 2006.

Carsten Witt. Population size versus runtime of a simple evolutionary algorithm. *Theoretical Computer Science*, 403(1):104–120, 2008.

Gerhard J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.

Christine Zarges. Rigorous runtime analysis of inversely fitness proportional mutation rates. In *10th International Conference on Parallel Problem Solving from Nature (PPSN '08)*, Band 5199 aus *Lecture Notes in Computer Science*, Seiten 112–122. Springer-Verlag, 2008.

Christine Zarges. On the utility of the population size for inversely fitness proportional mutation rates. In *10th International Workshop on Foundations of Genetic Algorithms (FOGA '09)*, Seiten 39–46. ACM Press, 2009.

Eberhard Zeidler, Herausgeber. *Teubner-Taschenbuch der Mathematik*, Band 1. B. G. Teubner, zweite Auflage, 2003.

Yuren Zhou. Runtime analysis of an ant colony optimization algorithm for TSP instances. *IEEE Transactions on Evolutionary Computation*, 13(5):1083–1092, 2009.

Eckart Zitzler. Hypervolume metric calculation, 2001. Computer Engineering and Networks Laboratory (TIK), ETH Zürich, Zürich, Schweiz, `ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c`.

Eckart Zitzler und Simon Künzli. Indicator-based selection in multiobjective search. In *8th International Conference on Parallel Problem Solving from Nature (PPSN '04)*, Seiten 832–842. Springer-Verlag, 2004.

Eckart Zitzler und Lothar Thiele. Multiobjective optimization using evolutionary algorithms – a comparative case study. In *8th International Conference on Parallel Problem Solving from Nature (PPSN '98)*, Band 1498 aus *Lecture Notes in Computer Science*, Seiten 292–301. Springer-Verlag, 1998a.

Eckart Zitzler und Lothar Thiele. An evolutionary approach for multiobjective optimization: The strength Pareto approach. TIK Report 43, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, Zürich, Schweiz, 1998b.

Eckart Zitzler und Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Eckart Zitzler, Marco Laumanns und Lothar Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *EUROGEN '01*, Seiten 95–100. CIMNE, 2001.

Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca und Viviane Grunert da Fonseca. Performance assesment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Eckart Zitzler, Dimo Brockhoff und Lothar Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *4th International Conference on Evolutionary Multi-Criterion Optimization (EMO '07)*, Band 4403 aus *Lecture Notes in Computer Science*, Seiten 862–876. Springer-Verlag, 2007.