Zur Analyse von randomisierten Suchheuristiken und Online-Heuristiken

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r N a t u r w i s s e n s c h a f t e n

der Universität Dortmund
am Fachbereich Informatik
von

Oliver Giel

2005

Tag der mündlichen Prüfung:	13. September 2005
Dekan:	Prof. Dr. Bernhard Steffen
Gutachter:	Prof. Dr. Günter Rudolph, Prof. Dr. Ingo Wegener

Danksagung

Mein Dank gilt Ingo Wegener für die Betreuung meiner Arbeit und die Zusammenarbeit an gemeinsamen Veröffentlichungen. Ebenso danke ich meinen Koautorinnen Lene Favrholdt und Susanne Albers. Den Mitarbeiterinnen und Mitarbeitern des Lehrstuhls für effiziente Algorithmen und Komplexitätstheorie danke ich für die freundschaftliche Arbeitsatmosphäre. Hervorheben möchte ich Carsten Witt, dem ich besonders für fachliche Diskussionen, aber auch für Ratschläge in Fragen der Orthographie danke.

Nicht zuletzt bin ich auch dem Fachbereich Informatik, der Universität Dortmund und der Deutschen Forschungsgemeinschaft zu Dank verpflichtet, die mit dem Sonderforschungsbereich „Computational Intelligence“ diese Arbeit erst ermöglicht haben.

Inhaltsverzeichnis

1 Vorwort **1**

I Randomisierte Suchheuristiken und das Matchingproblem

2 Einleitung **9**
2.1 Das Matchingproblem . . . 9
2.2 Einfache randomisierte Suchheuristiken . . . 10
2.3 Wahl der Fitnessfunktion . . . 13
2.4 Wie der (1+1)-EA Matchings verbessert . . . 16
2.5 Der Metropolisalgorithmus und Simulated Annealing . . . 18

3 Die Black-Box-Komplexität des Matchingproblems **21**

4 Randomisierte Suchheuristiken als Approximationsschemata **27**

5 Graphen mit polynomieller erwarteter Optimierzeit **33**
5.1 Pfadgraphen . . . 33
5.2 Vollständige Bäume . . . 44
5.3 Uneingeschränkte Bäume . . . 50

6 Graphen mit exponentieller erwarteter Optimierzeit **61**
6.1 Beweistechniken für exponentielle untere Schranken . . . 61
6.2 Ein schwieriger Graph für randomisierte Suchheuristiken . . . 67
6.3 Die Analyse . . . 71

7 Exponentielle Optimierzeit mit überwältigender Wahrscheinlichkeit **85**
7.1 Motivation . . . 85
7.2 Die Analyse . . . 87

8 Zusammenfassung **100**

II Zur Analyse einer grundlegenden randomisierten Suchheuristik für mehrkriterielle Probleme

9 Einleitung 103
9.1 Szenario und grundlegende Definitionen 103
9.2 Die Suchheuristik SEMO für partiell geordnete Zielräume 108
9.3 Literaturüberblick über Laufzeitanalysen für MOEAs 111

10 Die erwartete Optimierzeit im Worst Case 114

11 Ein Problem mit einstellbarem Schwierigkeitsgrad 120

12 Das Leading-Ones-Trailing-Zeroes-Problem 129

13 Das Multi-Objective-Counting-Ones-Problem 133
13.1 Das Problem 133
13.2 Die erwartete Optimierzeit 135
13.3 Eine obere Schranke für die Optimierzeit 136
13.4 Eine untere Schranke für die Optimierzeit 140

14 Zusammenfassung 147

III Über die Fehlerrate deterministischer Seitenwechselstrategien

15 Problemstellung, Motivation und Modellbildung 151
15.1 Das Seitenwechselproblem 151
15.2 Seitenwechselstrategien 154
15.3 Ergebnisse der kompetitiven Analyse 156
15.4 Maximums- und Durchschnittsmodell 161

16 Analyse der Fehlerrate im Maximumsmodell 167
16.1 Vorbereitungen 167
16.2 Eine größtmögliche untere Schranke 169
16.3 LRU und Markierungsstrategien 171
16.4 FIFO 175
16.5 Die optimale Offline-Strategie LFD 178

17 Analyse der Fehlerrate im Durchschnittsmodell 181
17.1 Eine größtmögliche untere Schranke 181
17.2 LRU und Markierungsstrategien 186
17.3 FIFO 194

17.4 Die optimale Offline-Strategie LFD 196

18 Die theoretischen Ergebnisse im Überblick **202**

19 Vergleich mit gemessenen Fehlerraten **204**

20 Zusammenfassung **211**

Anhang

A Mathematische Hilfsmittel **215**

B Symbolverzeichnis **219**

C Literaturverzeichnis **221**

1 Vorwort

Der Entwurf von Lösungsverfahren für Probleme unterschiedlichster Art gehört zu den Aufgaben der Informatik. Der Algorithmenentwurf muss sich selbstverständlich auf Probleme beschränken, die von Rechnern gelöst werden können. Erst die Analyse der Algorithmen ermöglicht durch die gewonnenen Einsichten den Entwurf verbesserter Algorithmen oder Aufschluss darüber, für welche Probleme und für welche Instanzen eines Problems ein Algorithmus geeignet ist. Die Analyse von Vertretern zweier Arten von Algorithmen, nämlich von randomisierten Suchheuristiken und Online-Heuristiken, ist das Thema dieser Dissertation.

Algorithmen sind häufig auf die Lösung eines bestimmten Problems spezialisiert, z. B. das Sortieren einer Folge von Zahlen. Daher kann man solche Algorithmen auch *problemspezifische Algorithmen* nennen. Man geht beim Entwurf problemspezifischer Algorithmen meistens davon aus, dass die Instanz des Problems dem Algorithmus als Eingabe vorliegt, im Beispiel die zu sortierende Folge von Zahlen. Der Zugriff auf die Probleminstanz ist prinzipiell nicht beschränkt. Wir können sagen, dass typische problemspezifische Algorithmen die Probleminstanz kennen, die sie zu bearbeiten haben.

Oft kann man Probleme, die von problemspezifischen Algorithmen gelöst werden, in sogenannte *Optimierungsprobleme* umwandeln. Unter einem Optimierungsproblem versteht man die Aufgabe, eine Funktion $f\colon S \to \mathbb{R}$ zu maximieren (oder zu minimieren). Wir gehen hier stets von Maximierungsproblemen aus. Das bedeutet, die Aufgabe ist es, ein Element $s \in S$ zu finden, sodass $f(s)$ maximal wird. Für das Beispiel des Sortierproblems ist S die Menge aller endlichen Zahlenfolgen. Die Funktion f gibt nun die „Sortiertheit“ einer Folge s an, die es zu maximieren gilt. Aus den Zahlen einer Folge s aus n Zahlen kann man $\binom{n}{2}$ Zahlenpaare bilden. So könnte f beispielsweise die Anzahl der Zahlenpaare angeben, deren Zahlen in s in der richtigen Reihenfolge stehen. Die Folge ist sortiert, wenn die Zahlen aller $\binom{n}{2}$ Zahlenpaare in s richtig geordnet sind.

Es wäre nicht sinnvoll, Algorithmen für beliebige Optimierungsprobleme vom Typ $f\colon S \to \mathbb{R}$ als problemspezifisch zu bezeichnen. Hinter so einem Optimierungsproblem können sich die unterschiedlichsten Probleme verbergen, für die es problemspezifische Algorithmen gibt oder für die solche nicht bekannt sind.

Optimierungsprobleme müssen häufig in einem Szenario gelöst werden, das *Black-Box-Szenario* genannt wird. In diesem Szenario erhält ein Algorithmus keine Beschreibung der Funktion f, obwohl seine Aufgabe die Optimierung von f ist. Es wird ihm nur ermöglicht, f für die Elemente $s \in S$ auszuwerten. Der Grund ist, dass eine Beschreibung von f, z. B. als geschlossene Formel, gar nicht bekannt ist. Die Funktion f kann vielleicht nur durch ein Experiment oder eine Rechnersimulation eines Experiments (näherungsweise) ausgewertet werden. Hierzu kann man sich beispielsweise vorstellen, dass f der Wirkungsgrad einer Maschine ist, der von den Stellgrößen s der Maschine abhängt. Der Zusammenhang zwischen den Stellgrößen und dem Wirkungsgrad ist aber nur schlecht verstanden und im Wesentlichen unbekannt. Viele in der Praxis vorkommende Probleme müssen in diesem Szenario gelöst werden. Oft fehlt es an Wissen oder z. B. im Rahmen eines Projektes auch an Zeit (und nicht zuletzt an Geld), um die Struktur des Problems genau genug untersuchen zu können. Selbst wenn eine Beschreibung von f vorliegt, sind häufig keine problemspezifischen Algorithmen bekannt.

Von problemspezifischen Algorithmen erwarten wir, dass sie eine korrekte und im Sinne der gestellten Anforderungen optimale Lösung abliefern. So erwarten wir von einem Sortieralgorithmus, dass er wirklich sortiert und erst stoppt, wenn er seine Eingabe sortiert hat. Eine „fast sortierte“ Folge würden wir nicht als Lösung akzeptieren. Für Black-Box-Algorithmen, das sind Algorithmen, die im Black-Box-Szenario arbeiten, ist diese Anforderung i. Allg. unsinnig. Ein Black-Box-Algorithmus kann sich erst sicher sein, ein Maximum gefunden zu haben, nachdem er sämtliche Elemente von S mindestens einmal ausgewertet hat. Erst dann „kennt“ er die Funktion f vollständig. Da die Menge S in der Regel viel zu groß ist, als dass dies in vernünftiger Zeit möglich wäre, ist es nicht das Ziel eines Black-Box-Algorithmus, eine beweisbar optimale Lösung zu finden.

Das Ziel eines Black-Box-Algorithmus ist es, eine möglichst gute Lösung in akzeptabler Zeit zu finden. Dabei hat er sich der Herausforderung zu stellen, dass er die Probleminstanz nicht kennt. Er lernt sie erst im Laufe der Zeit kennen. Die in dem oben beschriebenen Black-Box-Szenario arbeitenden Algorithmen können manchmal nicht einmal „wissen“, zu welchem Typ von Problemen die anfangs unbekannte Probleminstanz gehört. Das bedeutet, ihnen ist häufig nichts über die Struktur des Problems bekannt, z. B. ob hinter dem Optimierungsproblem ein Sortierproblem, ein Kürzeste-Wege-Problem oder ein sonst wie strukturiertes Problem steht. Falls immer wieder Instanzen desselben Problems gelöst werden müssen, werden Black-Box-Algorithmen jedoch mit Problemwissen ausgestattet, um so schneller zu guten Lösungen zu gelangen. Das bedeutet, die Algorithmen werden mithilfe der gewonnenen Erfahrung oder evtl. vorhandenen Problemwissens an das Problem angepasst.

In dieser Situation gibt es also nur die Hoffnung, für möglichst viele in der Praxis vorkommende Probleme oder zumindest möglichst viele Instanzen dieser Probleme effizient gute Lösungen zu finden. Ein Algorithmus wird oft als *Heuristik* bezeichnet, wenn keine Garantie besteht, dass er die Eigenschaften erfüllt, die wir von problemspezifischen Algorithmen z. T. ganz selbstverständlich annehmen. Dazu kann gehören, dass keine Garantie für eine polynomielle Laufzeit oder für die Güte der gefundenen Lösung gegeben werden kann oder dass überhaupt eine zulässige Lösung gefunden wird. Das *Informatik-Handbuch* (Rechenberg und Pomberger, 1997) definiert den Begriff Heuristik wie folgt.

> Eine Methode, die mit der Hoffnung auf Erfolg, nicht mit Garantie für Erfolg zur Lösung einer komplexen, nicht oder schlecht strukturierten Aufgabe eingesetzt wird.

In der Informatik gut bekannte Heuristiken sind Branch-and-Bound-Algorithmen. Sie werden z. B. auch zur Lösung NP-harter Probleme eingesetzt, wenn die Hoffnung besteht, dass die vorkommenden Instanzen des Problems zu erträglichen Rechenzeiten führen.

Randomisierte Suchheuristiken sind Heuristiken für Optimierungsprobleme, die zufällige Entscheidungen treffen. Viele Vertreter dieser Heuristiken arbeiten nach dem Prinzip der iterativen Verbesserung von Lösungen und werden oft erfolgreich eingesetzt. Zu dieser Gruppe von Algorithmen gehören u. a. (randomisierte) lokale Suche, Tabusuche, Gradientensuche, der Metropolisalgorithmus, Simulated Annealing und evolutionäre Algorithmen. Um aus lokalen Optima herauszufinden, weichen einige der aufgezählten Heuristiken gelegentlich von dem Prinzip der steten Verbesserung ab. (Es soll auch nicht gesagt sein, dass alle diese Verfahren grundsätzlich randomisiert arbeiten müssen.) Besonders die verschiedenen Varianten evolutionärer Algorithmen erfreuen sich großer Beliebtheit und gehören vielleicht zu den wichtigsten Vertretern randomisierter Suchheuristiken.

Der erste Teil dieser Dissertation befasst sich mit der Analyse eines sehr einfachen, aber grundlegenden evolutionären Algorithmus für kombinatorische Optimierungsprobleme. Exemplarisch wird dessen Laufzeit für ein gut bekanntes kombinatorisches Optimierungsproblem theoretisch analysiert. Das Problem ist das Matchingproblem, bei dem in einem ungerichteten Graphen möglichst viele Kanten auszuwählen sind, sodass keine zwei der ausgewählten Kanten einen gemeinsamen Endpunkt besitzen.

Der Gegenstand des zweiten Teils ist ein evolutionärer Algorithmus, den man als Verallgemeinerung des im ersten Teil untersuchten evolutionären Algorithmus auf mehrkriterielle Optimierungsprobleme ansehen kann. Mehrkriterielle Optimierungsprobleme sind Probleme $f\colon S \to \mathbb{R}^m$. Jeder Lösung s ist hier nicht nur *ein* Bewertungskriterium zugeordnet, sondern m Kriterien, die gleichzeitig zu optimieren sind. In der Regel kann keine einzelne Lösung $s \in S$ alle Kriterien maximieren. Man ist hier an der Lösungmenge interessiert, sodass jede einzelne Lösung der Menge hinsichtlich eines Kriteriums nur verbessert werden kann, wenn eine Verschlechterung hinsichtlich eines anderen Kriteriums in Kauf genommen wird. Im zweiten Teil der Dissertation werden Laufzeitanalysen für ausgewählte Probleme vorgestellt. Insbesondere wird auch die Worst-Case-Lauzeit der Heuristik untersucht.

Ein anderes Gebiet, in dem Heuristiken Anwendungen finden, sind Probleme, die im Online-Szenario gelöst werden müssen. Das Online-Szenario hat mit dem Black-Box-Szenario gemein, dass einem Algorithmus nicht von Anfang an Zugriff auf die Probleminstanz gewährt wird. Die Probleminstanz wird nur schrittweise offen gelegt. Als Reaktion darauf muss in jedem Schritt ein Teil der Lösung generiert werden.

Der dritte Teil dieser Dissertation beschäftigt sich mit dem Seitenwechselproblem (engl. *paging problem*), das gewöhnlich in einem Online-Szenario zu lösen ist. Das Seitenwechselproblem kann als Modell eines Rechners gesehen werden, der das Konzept des virtuellen Speichers umsetzt. Die Aufgabe ist es, den schnellen Speicher eines Speichersystems zu verwalten, das aus einem kleinen, schnellen Speicher und einem großen, langsamen Speicher besteht. Der langsame Speicher ist in Blöcke (Seiten) eingeteilt, die von einer Anfragesequenz angefragt werden. Diese Anfragen werden immer aus dem schnellen Speicher bedient. Dazu müssen die Seiten im schnellen Speicher zur Verfügung stehen, wenn sie angefragt werden. Falls der schnelle Speicher gefüllt ist und eine Seite angefragt wird, die nicht im schnellen Speicher vorgehalten wird, ist zu entscheiden, welche Seite im schnellen Speicher für die angefragte Seite Platz machen soll. Die angefragte Seite wird dann aus dem langsamen Speicher an den frei gewordenen Platz im schnellen Speicher kopiert. Dieser Vorgang verzögert die Abarbeitung der Sequenz. Die Aufgabe besteht darin, den schnellen Speicher so zu verwalten, dass möglichst wenige Verzögerungen entstehen. Man nennt so eine Verzögerung einen *Seitenfehler*. Falls die gesamte Anfragesequenz bekannt ist, ist dieses Problem ein kombinatorisches Optimierungsproblem. Man kann dann effizient eine Lösung mit der geringstmöglichen Zahl an Seitenfehlern berechnen. Im Online-Szenario wird die Anfragesequenz einem Algorithmus jedoch nur Anfrage um Anfrage eröffnet. Jede Anfrage muss sofort bedient werden, d. h., der Algorithmus muss ggf. entscheiden, welche Seite aus dem schnellen Speicher entfernt werden soll. Da zukünftige Anfragen dem Algorithmus nicht bekannt sind, kann er nur heuristisch vorgehen.

Unter der Analyse einer Online-Heuristik für das Seitenwechselproblem verstehen wir hier, die Anzahl der Seitenfehler zu bestimmen. Im ungünstigsten Fall kann offenbar jede Anfrage einen Seitenfehler verursachen. Daher ergibt es nur Sinn, Anfragesequenzen mit bestimmten Eigenschaften zu untersuchen, die das wiederholte Vorkommen von Anfragen an dieselbe Seite in der Sequenz modellieren. Wiederholungen von Anfragen an dieselbe Seite innerhalb kurzer Abschnitte der Anfragesequenz sind typisch, wenn die Sequenz von einem Prozess in einem Rechner erzeugt wird. Man nennt diese Eigenschaft *Anfragelokalität.* Im dritten Teil der Dissertation werden gängige, deterministische Online-Heuristiken für das Seitenwechselproblem unter dem Aspekt der Anfragelokalität analysiert.

Veröffentlichungen und Eigenanteil des Doktoranden

Die vorliegende Dissertation beruht auf den folgenden Veröffentlichungen, die hier nach den drei Teilen der Dissertation gegliedert aufgeführt sind.

Teil I:

1. Giel, O. und Wegener, I. (2003). Evolutionary algorithms and the maximum matching problem. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS '03)*, Band 2607 von *Lecture Notes in Computer Science*, Springer, S. 415–426.

2. Giel, O. und Wegener, I. (2004). Searching randomly for maximum matchings. Technischer Bericht, Electronic Colloquium on Computational Complexity (ECCC), Report Nummer 76(2004).

An diesen gemeinsam erstellten Veröffentlichungen ist der Eigenanteil etwa die Hälfte. Der Anstoß zur Beschäftigung mit dem Thema wurde vom Koautor Ingo Wegener gegeben. Die Beweisideen wurden gemeinsam entwickelt und ausgearbeitet.

Teil II:

3. Giel, O. (2003). Expected runtimes of a simple multi-objective evolutionary algorithm. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC '03)*, Band 3, IEEE, S. 1918–1925.

   Eine erweiterte Zeitschriftenfassung dieser Arbeit ("On the Analysis of a Simple Multi-objective Evolutionary Algorithm") ist zur Begutachtung eingereicht.

Der zweite Teil basiert auf dieser Arbeit. Lediglich die in Kapitel 13 vorgestellten Ergebnisse sind nur in der noch unveröffentlichten, erweiterten Fassung enthalten und wurden selbständig erzielt.

Teil III:

4. Albers, S., Favrholdt, L. M. und Giel, O. (2005). On paging with locality of reference. *Journal of Computer and System Sciences*, Band 70(2), S. 145–175.

   Dieser Zeitschriftenfassung ist folgende Konferenzfassung vorausgegangen:

   Albers, S., Favrholdt, L. M. und Giel, O. (2002). On paging with locality of reference. In *Proceedings of the 34th Annual Symposium on Theory of Computing (STOC '02)*, ACM, S. 258–267.

Der Eigenanteil an diesen Veröffentlichungen ist etwa ein Drittel. Die Arbeit wurde zunächst mit der Koautorin Susanne Albers begonnen, die auch die ursprüngliche Idee beigetragen hat, die später zur Formulierung der in Abschnitt 15.4 vorgestellten Modelle führte. Gemeinsam wurden die Ergebnisse erzielt, die in Kapitel 16 dargestellt werden. Der Autorenkreis hat sich später um die Koautorin Lene Favrholdt verstärkt, die im Wintersemester 2000/2001 Gast am Lehrstuhl für effiziente Algorithmen und Komplexitätstheorie war. In dieser Zeit wurden gemeinschaftlich die Ergebnisse erarbeitet, die hier in Kapitel 17 und Kapitel 19 vorgestellt werden.

Teil I

Randomisierte Suchheuristiken und das Matchingproblem

2 Einleitung

In diesem Kapitel werden das Matchingproblem und einfache randomisierte Suchheuristiken vorgestellt. Es wird motiviert, warum dieses Problem hier exemplarisch untersucht wird, und ein Überblick über vorhandene Resultate gegeben.

2.1 Das Matchingproblem

Das Matchingproblem (dt. *Zuordnungsproblem*) kann man sich anschaulich so vorstellen, dass man die Aufgabe hat, eine Gruppe von Personen in möglichst viele Zweierteams einzuteilen. Es können jedoch nicht beliebige Personen ein Team bilden, sondern nur solche, die miteinander harmonieren. Wenn wir gedanklich die Personen mit den Knoten eines Graphen verbinden, dann bedeutet eine ungerichtete Kanten zwischen zwei Knoten, dass die zugehörigen Personen zusammen ein Team bilden können. Formal ist das Matchingproblem wie folgt definiert.

Ein *Matching* bez. eines ungerichteten Graphen $G = (V, E)$ ist eine Teilmenge $M \subseteq E$, sodass keine zwei Kanten aus M einen gemeinsamen Endknoten besitzen. Das *Matchingproblem* ist das Problem, ein Matching M für G mit der größtmöglichen Zahl an Kanten zu finden. Ein solches Matching heißt *Maximummatching* (engl. *maximum cardinality matching*).

Wir nennen die Kanten, die zu einem Matching M gehören, *Matchingkanten*, wenn klar ist, von welchen Matching M und von welchem Graphen G die Rede ist. Wir sagen dann auch, dass M die Kanten *wählt*, die zu M gehören. Die nicht gewählten Kanten, das sind die Kanten $E \setminus M$, heißen *freie* Kanten. Ein Knoten, der von keiner Matchingkante überdeckt wird, der also kein Endpunkt einer Matchingkante ist, heißt ebenfalls *frei*. Um die Kanten verschiedener Matchings M und M' für denselben Graphen zu unterscheiden, sprechen wir dann von *M-Kanten* bzw. *M'-Kanten.*

Wie lösen problemspezifische Algorithmen das Matchingproblem? Typische Algorithmen benutzen sogenannte verbessernde Pfade. Unter einem *Pfad* in einem Graphen verstehen wir stets einen *einfachen Pfad*, d. h. einen Pfad ohne Knotenwiederholungen. Ein *alternierender Pfad* ist ein Pfad, auf dem sich entlang des Pfades freie Kanten und Matchingkanten abwechseln. Ein *M-verbessernder Pfad* ist ein alternierender Pfad bez. des Matchings M, der an einem freien Knoten mit einer freien Kante beginnt und an einem freien Knoten mit einer freien Kante endet. Wenn klar ist, welches Matching M gemeint ist, sprechen wir einfach von einem *verbessernden Pfad.* Verbessernde Pfade haben also eine freie Kante mehr

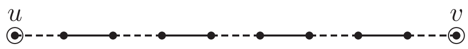

Bild 2.1: Verbessernder Pfad

als Matchingkanten. Wenn man auf einem verbessernden Pfad freie Kanten zu Matchingkanten macht und umgekehrt, erhält man ein neues Matching, das um eine Kante größer ist. Bild 2.1 zeigt einen verbessernden Pfad. Freie Kanten sind durch unterbrochene Linien dargestellt, Matchingkanten durch durchgezogene Linien. Die freien Endknoten des Pfades sind eingekreist.

Der Satz von Berge (1957) besagt, dass ein Matching M für G genau dann ein Maximummatching ist, wenn es keinen M-verbessernden Pfad in G gibt. Um ein Maximummatching zu konstruieren, genügt es also, so lange nach verbessernden Pfaden zu suchen und mit jedem gefundenen verbessernden Pfad das Matching um eine Kante zu vergrößern, bis es keinen verbessernden Pfad mehr gibt. Wenn man von einem freien Knoten ausgehend nach verbessernden Pfaden sucht, kann dieser Ansatz leicht zu exponentieller Laufzeit führen. Es war zunächst nicht klar, ob es überhaupt effiziente Algorithmen für das Matchingproblem geben kann. Erst Edmonds (1965) gelingt es, aus diesem Ansatz einen Polynomialzeitalgorithmus zu entwickeln, dessen Laufzeit er durch $O(n^4)$ beschränken kann. Wir bezeichnen hier wie allgemein üblich mit $n := |V|$ die Zahl der Knoten und mit $m := |E|$ die Zahl der Kanten des Graphen G. Hopcroft und Karp (1973) stellen einen Algorithmus für bipartite Graphen vor, dessen Laufzeit durch $O(n^{5/2})$ beschränkt ist. Der beste bekannte Matchingalgorithmus für (allgemeine) ungerichtete Graphen von Micali und Vazirani (1980) kommt mit einer Laufzeit von $O(n^{1/2}m)$ aus. Korrektheitsbeweise für diese Algorithmen sind keineswegs trivial.

2.2 Einfache randomisierte Suchheuristiken

Praktische Erfahrung, in welchen Situationen welche Heuristiken (hoffentlich) erfolgreich eingesetzt werden können, ist bei Anwendern randomisierter Suchheuristiken oft vorhanden. Für ein tieferes Verständnis, in welchen Situationen bestimmte randomisierte Suchheuristiken erfolgreich sein können, müssen sie jedoch analysiert werden.

Randomisierte Suchheuristiken für Optimierungsprobleme vom Typ $f\colon S \to \mathbb{R}$ sind, wie in Kapitel 1 dargelegt, normalerweise nicht auf ein bestimmtes Problem zugeschnitten. Laufzeitanalysen ohne Bezug zu einem bestimmten Problem ergeben jedoch keine aussagekräftigen Schranken, da die Probleme, die hinter so einem Optimierungsproblem stehen, sehr schwierige Probleme sein können. Das Beste, was man erreichen kann, ist, randomisierte Suchheuristiken für Problemklassen (d. h. Funktionenklassen) zu analysieren (z. B. Droste, Jansen und Wegener, 2002, Wegener und Witt, 2005). Die Anfänge der Laufzeitanalysen für randomisierte Suchheuristiken betrachteten zunächst häufig eigens zur Analyse ersonnene Funktionen. Daraus haben sich jedoch Analysemethoden und -werkzeuge entwickelt, mit denen zunehmend schwierigere Probleme und Fragestellungen untersucht werden konnten. Die Analyse randomisierter Suchheuristiken, besonders evolutionärer Algorithmen, für gut bekannte kombinatorische Optimierungsprobleme ist noch nicht sehr weit fortgeschritten. Sie profitiert aber inzwischen von den zuvor entwickelten Methoden.

Im Bereich der evolutionären Algorithmen wurden bereits das Sortierproblem, das Kürzeste-Wege-Problem (beide in Scharnow, Tinnefeld und Wegener, 2002) und in jüngster Zeit auch das Spannbaumproblem (Neumann und Wegener, 2004) untersucht. Diese Probleme erlauben es, nicht optimale Lösungen durch kleine, lokale Veränderungen zu verbessern. Das ist beim Matchingproblem anders. Wenn z. B. der in Bild 2.1 dargestellte Pfad nicht nur ein Ausschnitt aus dem Graphen ist, sondern der Graph nur aus diesem Pfad besteht, ist eine Verbesserung des Matchings nur möglich, wenn sämtliche Kanten ihren Status ändern.

Für Simulated Annealing und den Metropolisalgorithmus wurde das Matchingproblem bereits untersucht. Darauf wird später noch eingegangen. Sasaki (1991) untersucht den Metropolisalgorithmus neben dem Matchingproblem für das Problem *graph bisection* und das Travelling-Salesperson-Problem. Der Metropolisalgorithmus wurde für das Problem *graph bisection* (Jerrum und Sorkin, 1998) und das Cliquenproblem (Jerrum, 1992) jeweils für zufällige Graphen analysiert.

Wie sehen die hier angesprochenen randomisierten Suchheuristiken für Optimierungsprobleme aus? Einfache randomisierte Suchheuristiken, die nach dem Prinzip der iterativen Verbesserung einer Lösung arbeiten, können oft durch folgenden allgemeinen Rahmen beschrieben werden.

1. Wähle $s \in S$ gemäß einer Wahrscheinlichkeitsverteilung. (Initialisierung)
2. Wiederhole:

   Wähle $s' \in S$ gemäß einer Wahrscheinlichkeitsverteilung für S, die von s abhängt. (Suche)

   Entscheide abhängig von $f(s)$ und $f(s')$, ob s' den Suchpunkt s im nächsten Schleifendurchlauf ersetzen soll und setze ggf. $s := s'$. (Selektion)

Die durch dieses Schema beschriebenen Heuristiken stoppen offenbar nie und sind daher streng genommen keine Algorithmen. In Anwendungen wird man der Schleife ein Stoppkriterium hinzufügen, das die Schleife irgendwann abbricht. Im einfachsten Fall kann dies ein Schleifenzähler sein, der beim Erreichen eines bestimmten Zählerstandes die Schleife beendet. Im Kapitel 3 werden wir noch begründen, warum wir dem Stoppkriterium hier keine besondere Aufmerksamkeit schenken.

Aus diesem Schema kann man nun verschiedene Varianten einer einfachen randomisierten Suchheuristik ableiten. Diese unterscheiden sich durch verschiedene Suchund Selektionsoperatoren. Wir beschränken uns im Folgenden auf kombinatorische Optimierungsprobleme vom Typ $f\colon \{0,1\}^n \to \mathbb{R}$. Die beiden folgenden Suchoperatoren sind hier typisch.

lokal: Wähle s' als Hammingnachbarn von s, d. h., kippe ein zufällig gleichverteilt ausgewähltes Bit von s.

global: Erzeuge s aus s' durch unabhängiges Kippen jedes Bits von s mit Wahrscheinlichkeit $1/n$.

Für den Selektionsoperator sind folgende Varianten üblich.

elitär: Setze $s := s'$ genau dann, wenn $f(s') \geq f(s)$ gilt.

nicht elitär: Setze $s := s'$, wenn $f(s') \geq f(s)$ gilt. Andernfalls setze $s := s'$ mit einer Wahrscheinlichkeit $p(f(s), f(s'))$, die von den Funktionswerten der beiden Suchpunkte abhängt.

Die vielleicht einfachste Variante einer randomisierten Suchheuristik ist die *randomisierte lokale Suche (RLS)*. Sie kombiniert in obigem Schema den lokalen Suchoperator mit der elitären Selektion. Der sogenannte (1+1)-EA wählt den globalen Suchoperator und die elitäre Selektion. Die Wahl des nicht elitären Selektionsoperators zusammen mit dem lokalen Suchoperator führt zu Simulated Annealing und dem Metropolisalgorithmus. Deren Diskussion stellen wir bis zum Abschnitt 2.5 zurück. Zunächst betrachten wir den (1+1)-EA, den wir hier noch einmal klar formulieren wollen. Die folgende Beschreibung indiziert jeden Suchpunkt s des (1+1)-EA mit dem Zeitpunkt t, zu dem er der aktuelle Suchpunkt ist. Es bezeichnet also s_t den Inhalt der Variablen s zum Zeitpunkt t.

Definition 2.1 ((1+1)-EA). *Der (1+1)-EA läuft für eine zu maximierende Funktion $f\colon \{0,1\}^n \to \mathbb{R}$ wie folgt ab.*

1. *Setze $t := 1$.*
   Wähle $s_1 \in \{0,1\}^n$ gemäß einer Wahrscheinlichkeitsverteilung.
   Bestimme $f(s_1)$. *(Initialisierung)*

2. *Wiederhole:*

   Setze $t := t + 1$.

   Erzeuge s' aus s_{t-1} durch unabhängiges Kippen
   jedes Bits von s_{t-1} mit Wahrscheinlichkeit $1/n$.
   Bestimme $f(s')$. *(Suche)*

   Falls $f(s') \geq f(s_{t-1})$ gilt,
   dann setze $s_t := s'$,
   sonst setze $s_t := s_{t-1}$. *(Selektion)*

Man kann den (1+1)-EA als Vertreter sogenannter evolutionärer Algorithmen (EAs) verstehen. Unter dem Begriff *evolutionäre Algorithmen* ist eine Vielzahl randomisierter Suchheuristiken zusammengefasst. Die vielleicht wichtigsten Vertreter sind *genetische Algorithmen* (Goldberg, 1989) und *Evolutionsstrategien* (Rechenberg, 1973, Schwefel, 1995). Die Verfahren sind von biologischen Evolutionsprozessen inspiriert und imitieren einzelne Aspekte des biologischen Vorbilds. Ihnen allen ist gemein, dass sie eine Menge von Lösungen $s \in S$ verwalten. Diese Menge heißt *Population* und die Elemente der Population heißen *Individuen*. Im Fall des (1+1)-EA besteht die Population nur aus einem Individuum, das wir hier stets „Suchpunkt“ nennen.

Aus der Population werden regelmäßig Individuen herausgegriffen. Durch Anwenden von Suchoperatoren auf diese Individuen werden neue Lösungen, d. h. Elemente des Suchraums S, erzeugt und deren Güten durch Auswerten der Funktion f bestimmt. Den Suchoperator des (1+1)-EA nennen wir *$1/n$-Standardmutation* und s' einen *Mutanten* von s. Im Kontext evolutionärer Algorithmen heißt f *Fitnessfunktion* und die Güte $f(s)$ einer Lösung s heißt ihre *Fitness*. Auch diese Sprechweise übernehmen wir hier. Die Aufgabe des Suchoperators ist es also, bessere Lösungen zu finden. Die Fitnesswerte der neuen Lösungen werden nun mit denen der Individuen der Population verglichen und eine Auswahl unter den Individuen der Population und den neuen Lösungen getroffen (*Selektion*). Die ausgewählten Lösungen bilden nun die neue Population und die „Evolutionsschleife“ beginnt von vorn. Für eine umfassende Darstellung zu evolutionären Algorithmen ist z. B. Bäck, Fogel und Michalewicz (1997) zu nennen.

Einen Durchlauf der Schleife des (1+1)-EA nennen wir einen *Schritt*. Wenn der in einem Schritt erzeugte *Mutant* s' zum neuen Suchpunkt s_t wird, sagen wir, dass s' *akzeptiert* wird. Wir sprechen dann auch davon, dass *der (Mutations)schritt akzeptiert wird*. Andernfalls sagen wir, dass s' bzw. *der (Mutations)schritt nicht akzeptiert* oder *verworfen* wird.

Bis auf den Suchoperator läuft die RLS genau wie der (1+1)-EA ab. Beide verhalten sich nach Art eines „randomisierten Hillclimbers“, da sie neue Punkte stets verwerfen, wenn diese kleinere Fitness als der aktuelle Suchpunkt haben.

2.3 Wahl der Fitnessfunktion

Die Menge aus allen ungerichteten Graphen $G = (V, E)$ ist die Menge der Instanzen des Matchingproblems. Es sei $n := |V|$ und $m := |E|$. Wir müssen nun das Matchingproblem als ein Optimierungsproblem vom Typ $f\colon S \to \mathbb{R}$ formulieren, damit es von einfachen Suchheuristiken bearbeitet werden kann. Dazu entwerfen wir für jeden Graphen G seine Fitnessfunktion $f_G\colon \{0,1\}^m \to \mathbb{R}$. Der Suchraum $S = \{0,1\}^m$ beschreibt die Menge aller 2^m möglichen Kantenauswahlen. Dazu verbinden wir jede Kante mit einer Bitposition. Ein 1-Bit an dieser Position gibt an, dass die Kante ausgewählt ist. Da wir an Maximummatchings interessiert sind, sollte f_G monoton in der Matchinggröße sein. Für die RLS und den (1+1)-EA ist nur die relative Ordnung von Lösungen entscheidend. Also können wir für Suchpunkte $s = (s_1, \ldots, s_m)$, die Matchings beschreiben, o. B. d. A. festlegen, dass $f_G(s)$ gleich der Matchinggröße von s ist. (Wir unterscheiden hier und im Folgenden nicht streng zwischen Suchpunkten und den Kantenauswahlen, die sie repräsentieren. So sprechen wir auch einfach von der Matchinggröße des Suchpunktes s, wenn klar ist, dass s ein Matching repräsentiert.) Somit ist $f_G(s) = s_1 + \cdots + s_m$ für Matchings s.

Wie sollen Suchpunkte s behandelt werden, die keine Matchings sind? Eine einfache Antwort ist, dass man die Suche mit einem Matching beginnen lässt und Nichtmatchings als Suchpunkte verbietet. In der Literatur über den Metropolisalgorithmus und Simulated Annealing wird häufig so verfahren. Als Startpunkt der Suche kommt dann die leere Menge als Kantenauswahl in Frage. Wir bezeichnen das Matching, das keine Kante enthält, im Folgenden auch als das *leere Matching*.

Für die RLS und den (1+1)-EA mag es nahe liegend erscheinen, die Fitness von Nichtmatchings auf -1 zu setzen, sodass Nichtmatchings verworfen werden, nachdem einmal ein Matching erreicht worden ist. Falls, wie bei der RLS und dem (1+1)-EA üblich, die Suche mit einem gleichverteilt aus dem Suchraum ausgewählten Suchpunkt beginnt, ergibt sich jedoch folgende Schwierigkeit. Aus Chernoffschranken (siehe Anhang A) folgt, dass der Startpunkt der Suche dann mit einer Wahrscheinlichkeit von $1 - e^{-\Omega(m)}$ mindestens $m/4$ 1-Bits hat. Es gibt Graphen, deren Matchings im Vergleich zu der Kantenzahl des Graphen so klein sind, dass diese erst nach sehr langer Zeit gefunden würden. Dazu betrachten wir den vollständigen Graphen K_n. Das ist der ungerichtete Graph mit n Knoten, in dem je zwei Knoten durch eine Kante verbunden sind. Der Graph K_n hat also $m = \binom{n}{2} = \Theta(n^2)$ Kanten. Jedes Matching kann höchstens $n/2 = O(\sqrt{m})$ Kanten haben, da jede Kante zwei Knoten überdeckt. Die Matchinggröße ist also für eine Konstante $c > 0$ durch $c\sqrt{m}$ beschränkt. Wir betrachten nun exemplarisch die RLS. Um von einem Startpunkt mit mindestens $m/4$ 1-Bits zu einem Suchpunkt mit höchstens $c\sqrt{m}$ 1-Bits zu gelangen, muss die RLS zuvor einen Suchpunkt mit $2c\sqrt{m} - 1$ 1-Bits erreichen. Für alle Suchpunkte mit höchstens $2c\sqrt{m}$ 1-Bits gilt, dass der nächste Schritt der RLS mit einer Wahrscheinlichkeit von $(m - 2c\sqrt{m})/m = 1 - o(1)$ die Zahl der 1-Bits erhöhen und mit der Gegenwahrscheinlichkeit von $o(1)$ weiter senken wird. Mit Argumenten, die wir in Abschnitt 6.1 unter dem Begriff des Gambler's-Ruin-Problem kennen lernen werden, kann man leicht einsehen, dass in solchen „unfairen“ Situationen die Wahrscheinlichkeit überwältigend ist, wieder $2c\sqrt{m}$ 1-Bits zu erreichen, bevor $c\sqrt{m}$ erreicht werden. Dies führt dazu, dass die erwartete Zeit, bis irgendein Matching gefunden wird, exponentiell ist, wenn die Suche mit einem zufälligen Startpunkt beginnt und alle Nichtmatchings gleich bewertet werden.

Es gibt zwei Möglichkeiten, damit umzugehen. Man kann die Suche mit dem leeren Matching beginnen lassen. Ein fest gewählter Startpunkt, kann (in geeignet konstruierten Beispielen) jedoch ein schlechter Startpunkt sein. In Kapitel 7 werden wir ein Beispiel sehen, in dem das leere Matching ein ungünstiger Startpunkt ist. Andernfalls muss die Fitnessfunktion im Gebiet der Nichtmatchings „Hinweise in Richtung der Matchings“ geben. Da Matchings jeden Knoten höchstens mit einer Kante überdecken sollen, ist es eine natürliche Vorgehensweise, „Strafen“ für mehrfach überdeckte Knoten einzuführen. Es sei $d_G(v, s)$ der Knotengrad des Knotens v in G, wenn nur die Kanten berücksichtigt werden, die durch den Suchpunkt s ausgewählt sind. Dann ist der Strafterm (engl. *penalty*) für den Knoten v

$$p_G(v, s) \ := r \cdot \max\{0, d_G(v, s) - 1\}.$$

Darin ist r eine beliebige Konstante größer als m. Offenbar ist die Knotenstrafe 0, wenn der Knoten nur einfach oder gar nicht überdeckt ist, und wächst mit jeder zusätzlichen Kante um r. Die Fitnessfunktion lautet nun

$$f_G(s) := s_1 + \cdots + s_m - \sum_{v \in V} p_G(v, s).$$

Jedes Matching hat eine nicht negative Fitness von höchstens m. Der Skalierungsfaktor r in den Knotenstrafen sorgt nun dafür, dass alle Nichtmatchings negative Fitness erhalten.

Wir betrachten die Fitnessfunktion f_G selbst nie als Teil der Heuristiken. Doch wenn wir den (1+1)-EA oder die RLS auf das Matchingproblem anwenden wollen, müssen wir selbstverständlich auch die Fitnessfunktion implementieren. Wichtig ist, welche Rechenzeit eine Auswertung von f_G benötigt. Wir behaupten, dass die erste Auswertung in Zeit $O(m)$ durchgeführt werden kann und jede weitere Auswertung in einer erwarteten Zeit von $\Theta(1)$ möglich ist. Um die Fitness von s_1 zu bestimmen, berechnen wir in Zeit $O(m)$ die Menge der Knoten, die von mindestens einer Kante von G überdeckt werden. Im Folgenden können wir daher o. B. d. A. annehmen, dass G keine isolierten Knoten enthält. Nun legen wir für jeden Knoten v einen Zähler an, der die Anzahl $d_G(v, s)$ der Kanten des aktuellen Suchpunktes s angeben soll, die diesen Knoten überdecken. Beim Anlegen werden alle Zählerstände auf 0 gesetzt. Diese Zählerstände können nun für den ersten Suchpunkt s_1 in Zeit $O(m)$ korrekt initialisiert werden, indem wir den Bitstring des aktuellen Suchpunkts durchlaufen und die Zähler der Endpunkte aller ausgewählten Kanten entsprechend erhöhen. Dabei zählen wir auch gleich die 1-Bits im aktuellen Suchpunkt, sodass wir danach die Summe $s_1 + \cdots + s_m$ kennen. Nun durchlaufen wir alle Zähler und können dabei leicht für jeden Knoten v seine Knotenstrafe $p_G(v, s_1)$ berechnen. Insgesamt können wir so die Fitness des ersten Suchpunktes in Zeit $O(m)$ berechnen. Wenn in einem Suchschritt k Bits gekippt werden, können in Zeit $\Theta(k)$ alle betroffenen Zählerstände für s' aktualisiert werden und daraus die Fitnessänderung berechnet werden. Wenn die Fitness insgesamt sinkt, wird der Mutant s' verworfen und die Änderungen an den Zählern können ebenfalls in Zeit $\Theta(k)$ zurückgenommen werden. Die RLS kippt nur konstant viele Bits pro Schritt, nämlich eines. Für den (1+1)-EA ist die erwartete Zahl kippender Bits $m \cdot 1/m = 1$.

2.4 Wie der (1+1)-EA Matchings verbessert

Das Konzept verbessernder Pfade ist dem (1+1)-EA nicht „bekannt“, d. h., er wurde nicht dazu entworfen, verbessernde Pfade zu finden. Trotzdem können allgemein gehaltene randomisierte Suchheuristiken dieses Konzept „benutzen“, um Matchings zu verbessern. Die kürzesten verbessernden Pfade bestehen aus nur *einer* freien Kante, die zwei freie Knoten verbindet. Wir nennen so einen verbessernden Pfad aus nur einer freien Kante eine *wählbare* Kannte. Wählbare Kanten können dem Matching hinzugefügt werden, wodurch ein größeres Matching entsteht. Für den (1+1)-EA ist eine Situation, in der es eine wählbare Kante e gibt, eine gute Situation. Falls im nächsten Schritt nur das Bit kippt, das zu dieser Kante gehört, repräsentiert der Mutant ein größeres Matching und wird daher akzeptiert. Wir sagen, dass *die Kante e kippt,* wenn wir das Bit an der Bitposition meinen, die der Kante e zugeordnet ist. Die Wahrscheinlichkeit, dass in einer $1/m$-Standardmutation nur eine bestimmte Kante e kippt, ist das Produkt der Wahrscheinlichkeit, dass e kippt, und der Wahrscheinlichkeit, dass alle andern Kanten nicht kippen. Die Wahrscheinlichkeit ist also $(1/m) \cdot (1 - 1/m)^{m-1}$. Dieser Ausdruck ist nach oben durch $1/m$ und nach unten durch $1/(em)$ beschränkt (vgl. Anhang A). Also ist die Wahrscheinlichkeit, dass nur eine bestimmte Kante kippt, $\Theta(1/m)$.

Ein Matching, das keine wählbaren Kanten übrig lässt, heißt im Englischen *maximal matching*, weil es nicht durch Hinzunahme weiterer Kanten erweitert werden kann. Das bedeutet, sämtliche verbessernden Pfade haben mindestens die Länge 3. Der (1+1)-EA ist in der Lage, alle Kanten eines verbessernden Pfades aus ℓ Kanten in einem Schritt zu kippen. Die Wahrscheinlichkeit, dass eine $1/m$-Standardmutation nur die ℓ ausgewählten Kanten kippt, ist $(1/m)^{\ell} \cdot (1-1/m)^{m-\ell} = \Theta(1/m^{\ell})$. Eine unmittelbare Matchingverbesserung wird also mit zunehmender Pfadlänge rasch unwahrscheinlich. In Situationen ohne wählbare Kanten können 1-Bit-Mutationen offenbar nur Mutanten mit schlechterer Fitness erzeugen. Am häufigsten werden dann noch 2-Bit-Mutationen akzeptiert. Diese können die Länge eines verbessernden Pfades verändern. Der verbessernde Pfad in Bild 2.2 oben verkürzt sich um zwei Kanten, wenn seine beiden dem Endpunkt u nächstliegenden Kanten kippen. Das Ergebnis ist der Pfad in Bild 2.2 Mitte. Wenn im folgenden Schritt die beiden Kanten kippen, die sich nach rechts an den Knoten v anschließen, verlängert sich der Pfad wieder (Bild 2.2 unten). Fortgesetzte Verkürzungen um 2 Kanten können jedoch eine wählbare Kante erzeugen, die dann mithilfe einer abschließenden 1-Bit-Mutation zu einer Matchingverbesserung führen kann.

Matchings gleicher Größe und ohne wählbare Kanten bilden also bez. der Fitnessfunktion ein *Plateau*, auf dem Punkte bez. 2-Bit-Mutationen zusammenhängend sind. Die 2-Bit-Mutationen, die freie Knoten um zwei Kanten bewegen, können verbessernde Pfade verkürzen und verlängern, bis schließlich ein Matching mit einem verbessernden Pfad mit Länge 1, einer wählbaren Kante, erreicht ist. Diese Matchings kann man als „Ausgänge“ eines solchen Plateaus betrachten. Die Wahrschein-

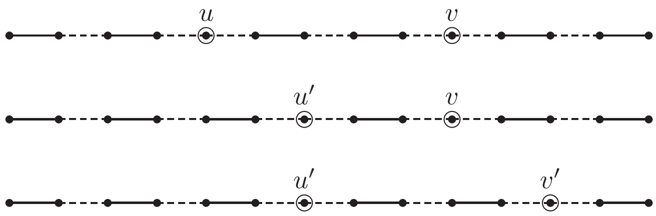

Bild 2.2: Wirkung von 2-Bit-Mutationen

lichkeit, durch Kippen der wählbaren Kante im nächsten Schritt das Matching zu verbessern, ist dann mit $\Theta(1/m)$ häufig groß genug. Kern der Analysen wird es sein, die Zahl der Schritte abzuschätzen, bis ein Matching mit einer wählbaren Kante erreicht ist. Jansen und Wegener (2001) analysieren die Laufzeit des (1+1)-EA für Fitnessfunktionen mit Plateaus aus zusammenhängenden Punkten im Suchraum.

Es wird nun klar, dass für die RLS, deren lokaler Suchoperator nur eine Kante hinzunehmen oder entfernen kann, Situationen ohne eine wählbare Kante lokale Maxima darstellen, die sie nicht mehr verlassen kann. Die RLS, wie wir sie bisher beschrieben haben, ist daher ungeeignet, f_G zu maximieren. Ihr Suchoperator arbeitet „etwas zu lokal“. Aus unseren Überlegungen zu 2-Bit-Mutationen folgt, dass zusätzliche 2-Bit-Mutationen zumindest die Chance eröffnen, jedes lokale Maximum zu verlassen. Wir betrachten daher eine Variante der RLS, die wir ebenfalls RLS nennen. In den restlichen Kapiteln in diesem Teil der Dissertation verstehen wir unter der RLS die folgende randomisierte Suchheuristik.

Definition 2.2 (RLS). *Für eine zu maximierende Funktion $f\colon \{0,1\}^n \to \mathbb{R}$ läuft die RLS wie folgt ab.*

1. *Setze $t := 1$.*
   Wähle $s_1 \in \{0,1\}^n$ gemäß einer Wahrscheinlichkeitsverteilung.
   Bestimme $f(s_1)$. *(Initialisierung)*

2. *Wiederhole:*

   Setze $t := t + 1$.

   Entscheide mithilfe einer fairen Münze, ob s' durch Kippen von 1 oder 2 Bits aus s entstehen soll.
   Im ersten Fall kippe ein gleichverteilt ausgewähltes Bit von s.
   Im zweiten Fall kippe 2 verschiedene Bits, die gleichverteilt aus allen $\binom{n}{2}$ Bitpaaren von s ausgewählt werden.
   Bestimme $f(s')$. *(Suche)*

   Falls $f(s') \geq f(s_{t-1})$ gilt,
   dann setze $s_t := s'$,
   sonst setze $s_t := s_{t-1}$. *(Selektion)*

Wir werden in den Analysen des (1+1)-EA für das Matchingproblem häufig feststellen, dass 1- und 2-Bit-Mutationen die wesentlichen Schritte sind. Die RLS können wir als einen (1+1)-EA auffassen, der auf diese wesentlichen Schritte reduziert ist. Offenbar ist für die RLS die Wahrscheinlichkeit, dass ein bestimmtes Bit (eine bestimmte Kante) kippt, $(1/2)\cdot(1/m) = \Theta(1/m)$. Die Wahrscheinlichkeit, dass zwei bestimmte Bits (zwei bestimmte Kanten) kippen, ist $(1/2)\cdot(2/(m(m-1))) = \Theta(1/m^2)$. Diese Wahrscheinlichkeiten sind von derselben Größenordnung wie entsprechende Wahrscheinlichkeiten für den (1+1)-EA. Die Wahl einer fairen Münze im Suchschritt ist also kein entscheidender Punkt. Im Grunde ist jedes Zufallsexperiment geeignet, das mit konstanter Wahrscheinlichkeit eine der beiden Alternativen wählt. Die Motivation, die RLS so zu definieren, ist also weniger darin zu sehen, dass diese RLS eine „natürliche“ randomisierte Suchheuristik wäre. Vielmehr dient die RLS dazu, Beweisideen für den (1+1)-EA ohne den manchmal schwierig zu handhabenden Einfluss von Mehrbitmutationen darstellen zu können. Die Analyse der RLS bildet dann häufig die Grundlage der Analyse des (1+1)-EA. Nicht immer wird es gelingen, alle Beweise auf den (1+1)-EA zu übertragen. In diesen Situation bleibt zumindest die Vermutung, dass Ergebnisse für die RLS auch für den (1+1)-EA gelten.

2.5 Der Metropolisalgorithmus und Simulated Annealing

Wir beziehen uns noch einmal auf das Schema einer einfachen randomisierten Suchheuristik aus Abschnitt 2.2. Der (1+1)-EA und die RLS wählen den elitären Selektionsoperator. Von diesen beiden Heuristiken ist nur der (1+1)-EA in der Lage, aus lokalen Optima zu entkommen, weil sein globaler Suchoperator im nächsten Schritt jeden Punkt aus $\{0,1\}^n$ mit einer positiven Wahrscheinlichkeit erzeugt. Einen anderen Weg, um lokale Optima zu verlassen, wählen der Metropolisalgorithmus (Metropolis, Rosenbluth, Rosenbluth, Teller und Teller, 1953) und Simulated Annealing (Kirkpatrick, Gelatt und Vecchi, 1983). Diese beiden Heuristiken setzen einen lokalen Suchoperator und eine nicht elitäre Selektion ein. Sie akzeptieren den neu erzeugten Suchpunkt s' mit Wahrscheinlichkeit 1, wenn er besser als der bisherige Suchpunkt s ist. Einen schlechteren Suchpunkt s' akzeptieren sie mit einer Wahrscheinlichkeit von $\mathrm{e}^{-(f(s)-f(s'))/T}$. Dabei ist $T > 0$ ein Parameter, der für den Metropolisalgorithmus konstant gewählt wird. Das bedeutet, je schlechter s' im Vergleich zum aktuellen Suchpunkt s ist, umso unwahrscheinlicher ist es, dass s' akzeptiert wird.

Simulated Annealing stellt den Parameter T, der in diesem Kontext „Temperatur“ genannt wird, dynamisch ein. Typisch – aber nicht zwingend – ist, dass T zu Beginn groß ist und nach einem festen Plan mit der Zeit abgesenkt wird. So werden zu Beginn auch große Verschlechterungen akzeptiert. Dies ermöglicht es, am Anfang lokale Optima zu verlassen. Das Absenken der Temperatur ist von der Hoffnung motiviert, dass der Suchpunkt nach einiger Zeit in der Nähe eines globalen Optimums liegt. Um zu verhindern, dass dieses Gebiet verlassen wird, sollte der Prozess sich nun mehr und mehr wie ein „Hillclimber“ verhalten. Tatsächlich gibt es Beispielfunktionen f, für die Simulated Annealing mit dem richtigen Abkühlungsschema das Optimum schneller findet als der Metropolisalgorithmus mit der bestmöglichen Temperatureinstellung. Dies ist nicht nur für konstruierte Funktionen der Fall, sondern auch für eine Funktion, die eine einfach strukturierte Instanz des Spannbaumproblems beschreibt (Wegener, 2005).

Die verbleibende Möglichkeit, nämlich den globalen Suchoperator und einen nicht elitären Selektionsoperator zu wählen, mischt beide Konzepte, lokale Optima zu verlassen. Sie hat keinen eigenen Namen und ist weniger üblich. Bei der Vielzahl an Arten selbst einfacher randomisierter Suchheuristiken muss man davon ausgehen, dass auch solche Varianten existieren. Bevor wir die Resultate erwähnen, die für Simulated Annealing und den Metropolisalgorithmus bez. des Matchingproblems bereits erzielt worden sind, führen wir noch einige Sprechweisen ein, die im Folgenden immer wieder vorkommen.

Wir sagen, dass eine Funktion $f\colon \mathbb{N} \to \mathbb{R}^+$ *(höchstens) polynomiell in n ist*, wenn $f(n) = O(p(n))$ für ein Polynom $p(n)$ gilt. Die Funktion f heißt *exponentiell (groß)* in Bezug auf n, wenn $f(n) = \mathrm{e}^{\Omega(n^\varepsilon)}$ für ein konstantes $\varepsilon > 0$ gilt. Wir sprechen z. B. von einer exponentiellen Laufzeit, wenn sie durch $\mathrm{e}^{\Omega(n^\varepsilon)}$ von unten beschränkt ist. Analog dazu heißt f *exponentiell klein,* wenn $f(n) = \mathrm{e}^{-\Omega(n^\varepsilon)}$ gilt. So sprechen wir z. B. von einer exponentiell kleinen Wahrscheinlichkeit, wenn diese durch $\mathrm{e}^{-\Omega(n^\varepsilon)}$ nach oben beschränkt ist. Eine Wahrscheinlichkeit, die von unten durch $1 - \mathrm{e}^{-\Omega(n^\varepsilon)}$ beschränkt ist, nennen wir *exponentiell nahe an 1* oder auch *überwältigend.*

Sasaki und Hajek (1988) betrachten für das Matchingproblem die natürliche Variante von Simulated Annealing, die in jedem Schritt höchstens eine Kante hinzufügen oder entfernen kann. Sie zeigen, dass Simulated Annealing in polynomieller Zeit Maximummatchings approximieren kann. Das bedeutet, ein Matching, sodass Maximummatchings höchstens um einen Faktor von $(1 + \varepsilon)$ größer sind, wird in einer erwarteten Zeit gefunden, die polynomiell in m, der Zahl der Kanten des Graphen, beschränkt ist. Der Polynomgrad hängt jedoch von ε ab. Da der Parameter T konstant gesetzt wird, gilt die Aussage sogar für den Metropolisalgorithmus. Jerrum und Sinclair (1989) liefern für dieses Resultat einen Beweis, der eine andere Methode benutzt. Solche Approximationsresultate sind interessant, da in den Anwendungsszenarien randomisierter Suchheuristiken exakte Optimierung häufig nicht das eigentliche Ziel ist. Man ist in erster Linie an *guten* Lösungen interessiert.

Sasaki und Hajek (1988) zeigen aber auch, dass es Graphen gibt, für die die erwartete Zeit, bis ein Maximummatching gefunden wird, exponentiell in m ist. Das gilt sogar für eine sehr weit gefasste Klasse von Strategien, nach denen die Temperatur T geregelt wird. So darf T auch vom aktuellen Suchpunkt abhängen.

Sasaki (1991) führt eine Untere-Schranken-Technik für den Metropolisalgorithmus vor, mit der ein ähnliches Resultat gezeigt wird.

Das Ziel in diesem ersten Teil der Dissertation ist es, vergleichbare Ergebnisse, wie sie für den Metropolisalgorithmus und Simulated Annealing erreicht wurden, auch für den (1+1)-EA und die RLS zu liefern.

3 Die Black-Box-Komplexität des Matchingproblems

Im Gegensatz zu problemspezifischen Algorithmen sind (allgemeine) randomisierte Suchheuristiken nicht auf ein bestimmtes Problem zugeschnitten. Dazu gehören viele Vertreter evolutionärer Algorithmen, die randomisierte lokale Suche, Simulated Annealing und viele weitere Heuristiken. Randomisierte Suchheuristiken für Optimierungsprobleme arbeiten meistens in einem Szenario, das *Black-Box-Szenario* genannt wird und bereits in Kapitel 1 umrissen wurde. Die im Folgenden vorgestellten Definitionen für Black-Box-Probleme, Black-Box-Komplexität und randomisierte Suchheuristiken sind angelehnt an die Darstellung in der Monographie von Wegener (2003) (vgl. auch Droste, Jansen, Tinnefeld und Wegener, 2003).

Ein *Black-Box-Problem* wird durch die Problemgröße n, den zugehörigen Suchraum S_n und die Menge F der möglichen Probleminstanzen beschrieben, die man als Menge von Funktionen $f\colon S_n \to \mathbb{R}$ auffassen kann. Die Aufgabe ist es, einen Suchpunkt $s \in S_n$ zu finden, sodass $f(s)$ maximal oder minimal wird. Black-Box-Probleme sind also Optimierungsprobleme. Wir gehen im Folgenden immer von Maximierungsproblemen aus.

Die Funktion f wird als „schwarzer Kasten“ (engl. *black box*) verstanden, da die durch f repräsentierte Probleminstanz nach außen nicht sichtbar wird. Das bedeutet, dass ein Algorithmus, der ein Black-Box-Problem löst, keinen unmittelbaren Zugriff auf die Probleminstanz hat. Der Algorithmus kann lediglich Anfragen an die Black Box stellen, indem er f für Suchpunkte $s \in S_n$ auswerten lässt und so Information über die in f verborgene Probleminstanz gewinnt.

Eine randomisierte Suchheuristik für ein Black-Box-Problem geht wie folgt vor:

1. Zum Zeitpunkt $t = 1$ wird gemäß einer Wahrscheinlichkeitsverteilung p_1 auf S_n ein Suchpunkt $s_1 \in S_n$ gewählt und mit einer Anfrage an die Black Box $f(s_1)$ bestimmt.

2. Für $t > 1$ wird entschieden, ob die Suche beendet werden soll. Für diese Entscheidung dürfen $(s_1, f(s_1)), \dots, (s_{t-1}, f(s_{t-1}))$ benutzt werden. Falls die Suche beendet werden soll, wird daraus ein s_i mit bestem $f(s_i)$-Wert ausgewählt und als Ergebnis der Suche präsentiert.

   Andernfalls wird abhängig von $(s_1, f(s_1)), \dots, (s_{t-1}, f(s_{t-1}))$ eine Wahrscheinlichkeitsverteilung p_t gewählt, gemäß der der neue Suchpunkt $s_t \in S_n$ ausgewählt wird. Mit einer Anfrage an die Black Box wird $f(s_t)$ bestimmt.

Wenn wir von dem Stoppkriterium absehen, passen sich der (1+1)-EA, die RLS, Simulated Annealing und der Metropolisalgorithmus in dieses Schema einer randomisierten Suchheuristik ein. Sie verzichten darauf, die gesamte Historie $(s_1, f(s_1)), \ldots, (s_{t-1}, f(s_{t-1}))$ abzuspeichern und beschränken sich auf *ein* Paar $(s_i, f(s_i))$. Auch populationsbasierte evolutionäre Algorithmen passen offenbar in dieses Schema. Diese speichern ebenfalls nur einen kleinen Teil der Historie ab, welcher der Population entspricht. Eventuell speichern sie zusätzlich noch den besten je angefragten Suchpunkt.

Im Gegensatz zu problemspezifischen Algorithmen beweisen randomisierte Suchheuristiken durch ihr Stoppen prinzipiell nicht, dass sie eine optimale Lösung gefunden haben. Daher wird in theoretischen Untersuchungen der Laufzeit das Stoppkriterium meist außer Acht gelassen. Stattdessen untersucht man die Laufzeit, bis zum ersten Mal ein Suchpunkt mit optimalem f-Wert angefragt wird. Wir nennen diese Laufzeit auch Optimierzeit. Bei der Analyse randomisierter Suchheuristiken hat man sich darauf geeinigt, als Maß für die Laufzeit die Zahl der Anfragen an die Black Box zu verwenden. Die Beantwortung dieser Anfragen mithilfe der Black Box wird im Vergleich zu den übrigen Operationen als teuer (d. h. zeitaufwendig) bewertet.

Unter dem Begriff „randomisierte Suchheuristik“ verstehen wir im Folgenden nur solche randomisierten Suchheuristiken für Black-Box-Probleme, die sich in das oben beschriebene Schema (evtl. ohne das Stoppkriterium) einfügen.

Definition 3.1 (Optimierzeit). *Die* Optimierzeit *einer randomisierten Suchheuristik $\mathcal{A}$ für eine Instanz f eines Black-Box-Problems ist die Zufallsvariable $T_{\mathcal{A},f} \in \mathbb{N}$, welche die Zahl der Anfragen an f angibt, bis $\mathcal{A}$ zum ersten Mal einen Suchpunkt mit optimalem Funktionswert anfragt.*

Die erwartete Optimierzeit *ist der Erwartungswert $E(T_{\mathcal{A},f})$. (Der Erwartungswert wird bez. der zufälligen Entscheidungen von $\mathcal{A}$ gebildet.)*

Für ein anderes Ziel als einen Suchpunkt mit optimalem f-Wert, z. B. einen Suchpunkt mit einer bestimmten Approximationsgüte oder einer anderen Eigenschaft, ist die Laufzeit bzw. erwartete Laufzeit analog definiert.

Die folgende Sprechweise ist üblich, sie kann jedoch leicht missverstanden werden. Man sagt beispielsweise, dass die Optimierzeit von $\mathcal{A}$ für ein bestimmtes Black-Box-Problem mit einer Wahrscheinlichkeit von mindestens p durch $O(g(n))$ beschränkt ist. Die Wahrscheinlichkeit p soll hier nicht die Korrektheit der Aussage infrage stellen. Es ist gemeint, dass für jede Instanz f dieses Black-Box-Problems die Zufallsvariable $T_{\mathcal{A},f}$ mit einer Wahrscheinlichkeit von mindestens p so ausfällt, dass sie durch $O(g(n))$ beschränkt ist.

Definition 3.2 (Black-Box-Komplexität). *Für ein Black-Box-Problem mit Funktionenmenge F und eine randomisierte Suchheuristik $\mathcal{A}$ ist* $\max_{f \in F} E(T_{\mathcal{A},f})$ *die erwartete Optimierzeit von $\mathcal{A}$ für eine Worst-Case-Instanz des Problems für $\mathcal{A}$.*

Die Black-Box-Komplexität *des Problems ist das Minimum dieser Werte über alle randomisierten Suchheuristiken $\mathcal{A}$.*

Die Theorie der Black-Box-Komplexität wurde entwickelt, um untere Schranken für ein Problem für alle randomisierten Suchheuristiken, insbesondere für alle evolutionären Algorithmen, beweisen zu können. Mit der (gewohnten) algorithmischen Komplexität ist sie i. Allg. nicht vergleichbar. Droste, Jansen und Wegener (2005) zeigen, dass NP-schwierige Probleme polynomielle Black-Box-Komplexität haben können und andererseits algorithmisch triviale Probleme exponentielle Black-Box-Komplexität aufweisen können.

Wir beabsichtigen, die Optimierzeit des (1+1)-EA und der RLS für das Matchingproblem zu analysieren. Es ist bekannt, dass Simulated Annealing für konstruierte Instanzen des Matchingproblems keine polynomielle erwartete Optimierzeit hat (Sasaki und Hajek, 1988). Es stellt sich daher die Frage, ob das Matchingproblem polynomielle Black-Box-Komplexität haben kann. Im positiven Fall besteht zumindest die Möglichkeit, dass der (1+1)-EA und die RLS in erwartet polynomieller Zeit Maximummatchings finden. Im negativen Fall wüssten wir, dass alle randomisierten Suchheuristiken Worst-Case-Eingaben haben, auf denen sie versagen, d. h., für die sie superpolynomielle erwartete Optimierzeit haben.

Um diese Frage zu beantworten, muss man berücksichtigen, *wie* das Matchingproblem im Black-Box-Szenario modelliert wurde. Da es viele Möglichkeiten gibt, kann man eigentlich nicht von *der* Black-Box-Komplexität des Matchingproblems sprechen. Diese Sprechweise rechtfertigt sich jedoch wie folgt. Wir zeigen, dass „naheliegende“ Modellierungen des Matchingproblems als Black-Box-Problem zu polynomieller Black-Box-Komplexität führen. Damit sind Modellierungen gemeint, die man wählen würde, wenn einem außer der Definition des Matchingproblems nichts über das Problem bekannt wäre. Das ist die Situation, in der sich Anwender randomisierter Suchheuristiken befinden, wenn sie ein neues Problem lösen müssen, über dessen Struktur sie nichts oder nur sehr wenig wissen. Wenn Anwender ausreichend Wissen über die Problemstruktur besitzen, sollten sie nicht zu reinen randomisierten Suchheuristiken greifen, sondern problemspezifische Algorithmen einsetzen – sofern diese zur Verfügung stehen oder sie selbst in der Lage sind, diese zu entwerfen.

Wir nehmen an, dass eine Probleminstanz $G = (V, E)$ des Matchingproblems durch eine Instanz $f_G\colon \{0,1\}^m \to \mathbb{R}$ des zugehörigen Black-Box-Problems mit Eingabelänge $m := |E|$ dargestellt wird. Die Eingabe sei ein charakteristischer Vektor, der eine Kantenauswahl beschreibt. Über die Funktion f_G nehmen wir an, dass sie die folgenden drei Eigenschaften aufweist. In Anlehnung an die Sichtweise evolutionärer Algorithmen nennen wir den $f_G(s)$-Wert eines Suchpunktes s im Folgenden seine Fitness.

1. Matchings haben größere Fitness als Nichtmatchings.

2. Matchings gleicher Größe haben gleiche Fitness.

3. Für zwei Matchings verschiedener Größe hat das größere Matching größere Fitness als das kleinere.

Zumindest die letzte Eigenschaft dürfte unstrittig sein, denn das Ziel sind Maximummatchings. Die erste Eigenschaft könnte man ablehnen. Ein großes Matching mit nur einer zusätzlichen Kante, welche die Matchingeigenschaft zerstört, ist vielleicht ein besserer Suchpunkt als ein kleines Matching. Gegen den zweiten Punkt könnte man vorbringen, dass diese Bewertung nicht fein genug ist. Matchings mit wählbaren Kanten sind vermutlich besser als Matchings derselben Größe, die nur lange verbessernde Pfade haben. Doch wenn wir so argumentieren, tragen wir schon in die Modellierung des Problems Wissen über das Matchingproblem hinein und haben dabei möglicherweise eine spezielle Suchheuristik im Blick, die davon profitieren könnte.

Theorem 3.3. *Für jede Modellierung des Matchingproblems als Black-Box-Problem, das obige Eigenschaften hat, ist die Black-Box-Komplexität $O(m^2)$.*

Beweis. Unter allen randomisierten Suchheuristiken gibt es auch eine, die keinen Gebrauch von zufälligen Entscheidungen macht, sondern deterministisch arbeitet und nach folgender Strategie vorgeht.

Mit der ersten Anfrage ermittelt sie die Fitness des leeren Matchings, indem sie den Suchpunkt aus m 0-Bits anfragt. Mit den folgenden $\binom{m}{2}$ Anfragen ermittelt sie für alle $\binom{m}{2}$ Kantenpaare, ob die beiden darin enthaltenen Kanten einen gemeinsamen Endpunkt besitzen. Dazu genügt es, nacheinander alle Suchpunkte mit genau zwei 1-Bits anzufragen: Ein Fitnesswert größer als die Fitness des leeren Matchings bedeutet, dass der Suchpunkt ein Matching darstellt und dass die beiden Kanten des Paares keinen gemeinsamen Endpunkt haben. Ein Fitnesswert kleiner als die Fitness des leeren Matchings bedeutet, dass der Suchpunkt kein Matching darstellt, weil das Kantenpaar einen gemeinsamen Endpunkt hat. Die Strategie speichert die Ergebnisse der Anfragen in einer Tabelle, in der man für je zwei verschiedene Kanten ablesen kann, ob die Kanten benachbart sind.

Jetzt zählt die Strategie alle 2^m Teilmengen der Kantenmenge auf. Mithilfe der Tabelle kann sie nun leicht für jede Teilmenge überprüfen, ob es darin ein Kantenpaar gibt, das die Matchingeigenschaft verletzt. Dazu genügt es, alle enthaltenen Kantenpaare aufzuzählen und für jedes Kantenpaar den zuvor gespeicherten Tabelleneintrag abzulesen. Danach sind der Strategie alle Maximummatchings für den ihr unbekannten Graphen G bekannt. Sie wählt deterministisch eines aus und fragt in der letzten Anfrage den zugehörigen Suchpunkt an.

Offensichtlich werden insgesamt nur $\binom{m}{2} + 2 = O(m^2)$ Anfragen an die Black Box gerichtet. □

Der Beweis zu Theorem 3.3 deckt eine Schwäche der Black-Box-Komplexität auf. Sie berücksichtigt nicht den Rechenaufwand, den die Algorithmen zwischen den Anfragen an die Black Box treiben. Im Beweis werden Mengen exponentieller Größe aufgezählt. Die dafür erforderliche Zeit wird in der Black-Box-Komplexität jedoch nicht berücksichtigt. Der Grund dafür ist die erwähnte Annahme, dass nur die Beantwortung der Anfragen an die Black Box aufwendig ist. Hinzu kommt, dass typische randomisierte Suchheuristiken zwischen den Anfragen keine Probleme mit großer algorithmischer Komplexität lösen.

Hier stellt sich die Frage, ob Theorem 3.3 gültig bleibt, wenn man höchstens polynomielle Rechenzeit zwischen den Anfragen an die Black Box gestattet. Dies ist der Fall, denn der Beweis von Theorem 3.3 kann auch ohne die exponentiell lange Rechnung vor der letzten Anfrage geführt werden. Im Folgenden wird beschrieben, wie dieser Schritt, der aus der Tabelle ein Maximummatching berechnet, in polynomieller Zeit durchgeführt werden kann.

Zu einem ungerichteten Graphen $G = (V, E)$ ist der *Kantengraph* (engl. *line graph*) G' wie folgt definiert. Der ungerichtete Graph G' hat die Knotenmenge $V' = E$. Zwischen zwei Knoten aus V' gibt es in G' genau dann eine Kante, wenn die zugehörigen Kanten in G benachbart sind, d. h., wenn sie einen gemeinsamen Endpunkt besitzen. Wenn wir weiterhin mit m die Kardinalität von E bezeichnen, hat G' genau m Knoten und höchstens $\binom{m}{2}$ Kanten. Der Graph G heißt auch *Wurzelgraph* (engl. *root graph*) von G'. Während man zu jedem Graphen den zugehörigen Kantengraphen bestimmen kann, besitzt nicht jeder Graph einen Wurzelgraphen. Nur Graphen, für die Wurzelgraphen existieren, sind Kantengraphen.

Bis auf eine Ausnahme ist der Wurzelgraph jedes zusammenhängenden Kantengraphen sogar eindeutig bestimmt (siehe z. B. Harary, 1969). Die Ausnahme bildet der vollständige Graph K_3, denn dieser besitzt zwei Wurzelgraphen. Der erste ist isomorph zu K_3 selbst, der zweite ist isomorph zu dem Sterngraphen mit drei Kanten, den man auch als vollständigen bipartiten Graphen $K_{1,3}$ notieren kann. Lehot (1974) gibt einen Algorithmus an, der zu einem ungerichteten Graphen $G' = (V', E')$ in Zeit $O(|E'|)$ einen Wurzelgraphen konstruiert, sofern G' ein Kantengraph ist.

Um die Idee zur effizienten Konstruktion eines Maximummatchings zu beschreiben, nehmen wir an, der verborgene Graph $G = (V, E)$ sei zusammenhängend und habe mindestens fünf Knoten. Dann ist ausgeschlossen, dass sein Kantengraph G' isomorph zu K_3 ist. Die Tabelle, die im Beweis zu Theorem 3.3 erstellt wird, ist die Adjazenzmatrix des Kantengraphen G' zu G. In diesem Fall können wir in Zeit $O(m^2)$ den unbekannten Wurzelgraphen G aus der Tabelle rekonstruieren, indem wir Lehots Algorithmus anwenden. Sobald wir G kennen, können wir einen beliebigen der bekannten Polynomialzeitalgorithmen für das Matchingproblem anwenden und ein Maximummatching für G bestimmen.

Was ändert sich, wenn der verborgene Graph G nicht zusammenhängend ist? Es könnte in G Knoten geben, mit denen keine Kante inzidiert. Diese sind folglich im Kantengraphen G' nicht repräsentiert und können bei der Rekonstruktion von G aus G' nicht wiederhergestellt werden. Das ist auch nicht notwendig, da diese isolierten Knoten von G für Matchings keine Rolle spielen. Die übrigen Zusammenhangskomponenten von G enthalten also mindestens je eine Kante. Offenbar bilden die Kanten in einer Zusammenhangskomponente von G eine Zusammenhangskomponente von G'. Die Zusammenhangskomponenten von G' können wir effizient bestimmen, indem wir mit $|V'|$ Komponenten beginnen, die anfänglich je einen Knoten von G' enthalten. Dann durchlaufen wir die Kantenmenge E', indem wir die Adjazenzmatrix von G' durchlaufen. Dabei verschmelzen wir die Komponenten, die zu den Endpunkten jeder Kante gehören.

Jetzt können wir für jede Zusammenhangskomponente von G' einen Wurzelgraphen bestimmen, der einer Zusammenhangskomponente von G entspricht. Sofern eine Zusammenhangskomponente von G' mindestens vier Knoten besitzt, ist sie nicht isomorph zu dem vollständigen Graphen K_3 und besitzt folglich einen eindeutig bestimmten Wurzelgraphen, den wir mit Lehots Algorithmus bestimmen können. Danach können wir effizient ein Maximummatching für die entsprechende Zusammenhangskomponente von G berechnen. Andernfalls, wenn die Zusammenhangskomponente von G' höchstens drei Knoten enthält, könnten wir es mit dem K_3 zu tun haben. Die Knoten der Zusammenhangskomponente von G' geben uns unmittelbar die Kanten der entsprechenden Zusammenhangskomponente von G an. In konstanter Zeit können wir für die höchstens 2^3 möglichen Kantenauswahlen in dieser Zusammenhangskomponente von G mithilfe der Tabelle prüfen, welche davon Maximummatchings beschreiben, und ein Maximummatching auswählen.

Damit ist entschieden, dass die in Kapitel 2 gewählte Modellierung des Matchingproblems nicht zu Black-Box-Problemen führt, die für alle randomisierten Suchheuristiken schwierig sind. Sogar wenn die Gesamtrechenzeit berücksichtigt wird, gibt es deterministische Algorithmen, die dieses Problem im Black-Box-Szenario in polynomieller Zeit lösen können. Dies ist auch dann noch gültig, wenn wir die Rechenzeit der Black Box einbeziehen. Es ist klar, dass Black-Box-Anfragen in polynomieller Zeit beantwortet werden können, denn der Black Box ist G bekannt. Die Black Box gleicht in der gewählten Modellierung also nicht etwa einem Orakel, das für die Suchheuristik ein algorithmisch schwieriges Problem löst und so für polynomielle Black-Box-Komplexität sorgt.

Es gibt einen wichtigen Punkt, in dem sich Simulated Annealing, der Metropolisalgorithmus, der (1+1)-EA und die RLS von dem vorgestellten deterministischen Algorithmus unterscheiden. Die genannten randomisierten Suchheuristiken benutzen nur sehr wenig Speicherplatz, nämlich $O(m)$. Sie speichern im Wesentlichen nur einen Suchpunkt (und dessen Fitness) aus der Historie der angefragten Suchpunkte. Zum Nachweis der polynomiellen Black-Box-Komplexität haben wir allein zum Speichern der Adjazenzmatrix des Kantengraphen Platz $\Omega(m^2)$ benötigt. Für dicht besetzte Graphen lässt sich auch mithilfe besserer Datenstrukturen keine wesentliche Einsparung erreichen. Die Vermutung ist, dass Platz $\omega(m)$ erforderlich ist, um polynomielle Optimierzeit zu erreichen. Im folgenden Kapitel zeigen wir, dass der (1+1)-EA und die RLS trotz ihres geringen Speicherbedarfs zumindest *gute* Lösungen effizient finden.

4 Randomisierte Suchheuristiken als Approximationsschemata

In diesem Kapitel zeigen wir, dass sowohl der (1+1)-EA als auch die RLS in der Lage sind, Maximummatchings effizient zu approximieren. Das bedeutet Folgendes. Wenn wir für einen gegebenen Graphen G mit M^* ein Maximummatching für G bezeichnen und mit M ein beliebiges Matching, dann ist $|M^*|/|M| \geq 1$ die *Approximationsgüte* der Lösung M. Für gegebene Approximationsgüte $1+\varepsilon > 1$ zeigen wir, dass die erwartete Zeit, bis eine Lösung mit mindestens dieser Güte gefunden ist, für beide Heuristiken polynomiell beschränkt ist. Dabei wächst der Polynomgrad mit besser (d. h. kleiner) werdender Approximationsgüte.

Aus der Sicht evolutionärer Algorithmen würde man den ersten Suchpunkt typischerweise zufällig wählen. Wir haben schon weiter oben diskutiert, dass es dann bei ungeeigneter Wahl der Fitnessfunktion exponentielle Zeit brauchen kann, Matchings zu finden. Als erstes Etappenziel zeigen wir, dass beide Heuristiken in der Lage sind, die Hinweise unserer Fitnessfunktion zu nutzen und in kurzer Zeit *irgendein* Matching finden, und zwar unabhängig vom Startpunkt der Suche.

Lemma 4.1. *Unabhängig vom Startpunkt der Suche finden der (1+1)-EA und die RLS in einer erwarteten Zeit von $O(m \log m)$ ein Matching.*

Beweis. Es sei $p(s) := r \cdot k(s)$ die Summe aller Knotenstrafen des aktuellen Suchpunktes s. Dann ist $k(s)$ kleiner als $2m$, die Summe aller Knotengrade. Solange s noch kein Matching ist, erhöht jedes Absenken von $k(s)$ die Fitness $f(s) = s_1 + \cdots + s_m - p(s)$, weil $r \geq m+1$ ist. Aus demselben Grund würde jede Erhöhung des $k(s)$-Wertes die Fitness senken. Da jede Kante an höchstens zwei Knotenstrafen beteiligt sein kann, gibt es mindestens $\lceil k/2 \rceil$ Kanten, die den $k(s)$-Wert absenken, wenn man eine von diesen Kanten aus s entfernt. Für beide Heuristiken ist die Wahrscheinlichkeit $\Theta(1/m)$, im nächsten Schritt nur ein bestimmtes Bit zu kippen. Die Wahrscheinlichkeit ist $\Theta(k(s)/m)$, nur eines der Bits zu kippen, mit denen die Knotenstrafen gesenkt werden können. Daher ist die erwartete Zeit bis zum Absinken des $k(s)$-Wertes $O(m/k(s))$. Sobald $k(s)$ null ist, ist ein Matching erreicht. Also können wir die erwartete Zeit zum Finden eines Matchings durch die Summe

$$\sum_{1 \leq k(s) < 2m} O\left(\frac{m}{k(s)}\right) = O\left(m \sum_{1 \leq k(s) < 2m} \frac{1}{k(s)}\right) = O(m \log m)$$

abschätzen. □

Nachdem ein Matching gefunden wurde, ist die Grundidee der Analyse die folgende. Bis zum Erreichen der gewünschten Approximationsgüte gibt es einen verbessernden Pfad, der so kurz ist, dass der (1+1)-EA eine gute Chance hat, genau die Kanten dieses Pfades in einem Schritt zu kippen. Zwar wächst mit besser werdender Approximationsgüte die obere Schranke für die Pfadlänge, die man garantieren kann, sodass die Erfolgswahrscheinlichkeit für den (1+1)-EA sinkt. Dennoch bleibt diese Wahrscheinlichkeit genügend groß, um für jede fest gewählte Approximationsgüte höchstens polynomielle erwartete Laufzeiten nachweisen zu können. Das liegt daran, dass es bis zum Erreichen einer konstanten Approximationsgüte stets einen verbessernden Pfad gibt, dessen Länge durch eine Konstante begrenzt ist. Die RLS ist zwar nicht in der Lage, alle Kanten dieses Pfades in einem Schritt zu kippen, jedoch kann sie das in aufeinanderfolgenden Schritten bewerkstelligen.

Zunächst müssen wir daher die Länge des kürzesten verbessernden Pfades in einem Matching abschätzen. Die Aussagen des folgenden Lemmas findet man schon in der Arbeit von Hopcroft und Karp (1973). Wir beweisen sie hier, da wir später noch auf die Beweisidee zurückgreifen werden.

Lemma 4.2. *Es sei $G = (V, E)$ ein ungerichteter Graph, M ein Matching und M^* ein Maximummatching für G.*

1. *Man kann $|M^*| - |M|$ knotendisjunkte verbessernde Pfade bez. M finden.*
2. *Falls $|M| < |M^*|$ gilt, dann gibt es darunter einen verbessernden Pfad, dessen Länge durch $L := 2\lfloor |M|/(|M^*| - |M|)\rfloor + 1$ beschränkt ist.*

Zu Punkt 1 sei noch bemerkt, dass es durchaus mehr als $|M^*| - |M|$ verbessernde Pfade geben kann. Offenbar kann es aber nicht mehr als $|M^*| - |M|$ *knotendisjunkte* verbessernde Pfade geben, denn sonst ließe sich M um mehr als $|M^*| - |M|$ Kanten vergrößern.

Beweis zu Lemma 4.2. Wir betrachten die Zusammenhangskomponenten des ungerichteten Graphen $G' = (V, E')$, dessen Kantenmenge die symmetrische Differenz $E' := M \oplus M^*$ ist. Da M und M^* Matchings sind, ist der Knotengrad von G' höchstens 2. Daher zerfällt G' in knotendisjunkte Zusammenhangskomponenten, die Kreise oder Pfade sind. Entlang eines solchen Pfades oder Kreises wechseln sich M- und M^*-Kanten ab. Die Zusammenhangskomponenten kann man daher in folgende Typen einteilen:

– isolierte Knoten,

– alternierende Pfade und Kreise jeweils gerader Länge,

– alternierende Pfade ungerader Länge mit M^*-Kanten an den Enden und

– alternierende Pfade ungerader Länge mit M-Kanten an den Enden.

Unter den E'-Kanten, die zu Pfaden oder Kreisen mit gerader Länge gehören, wählen M und M^* gleich viele Kanten aus. Das Gleiche gilt für die Kanten, auf denen sich M und M^* nicht unterscheiden; das sind die Kanten, die nur zu E und nicht zu E' gehören.

Der letzte Typ Zusammenhangskomponenten kommt gar nicht vor. Um das einzusehen, betrachten wir zuerst einen Endpunkt so eines vermeintlichen Pfades. Der Endpunkt ist in G' nur von einer M-Kante überdeckt. Daher kann diese Kante nicht gleichzeitig M^*-Kante sein. Da M ein Matching ist, gibt es keine weitere Kante von M, die den Endpunkt überdeckt, und jede M^*-Kante, die den Endpunkt überdeckte, wäre in G' sichtbar. Also sind die Endpunkte freie Knoten bez. des Matchings M^*. Zwischen den Endpunkten wechseln sich in G freie Kanten und M^*-Kanten ab. Es handelt sich also um einen verbessernden Pfad bez. M^*. Das ist ein Widerspruch dazu, dass M^* ein Maximummatching ist.

Die Zahl der Kanten $|M^*| - |M|$, um die sich M und M^* in ihrer Größe unterscheiden, muss also allein durch Pfade vom dritten Typ verursacht sein, d. h., es muss $|M^*| - |M|$ dieser Pfade in G' geben. Mit denselben Argumenten wie beim letzten Typ, aber vertauschten Rollen von M und M^*, folgt, dass diese Pfade verbessernde Pfade bez. M sind. Die Knotendisjunktheit folgt aus der Tatsache, dass die Pfade Zusammenhangskomponenten von G' entsprechen. Damit ist der erste Teil des Lemmas bewiesen.

Da höchstens alle M-Kanten in den Pfaden vom zweiten Typ versammelt sein können, gibt es unter diesen verbessernden Pfaden bez. M einen Pfad, der höchstens $\lfloor |M|/(|M^*| - |M|) \rfloor$ M-Kanten besitzt. Dessen Länge ist höchstens L. □

Mit den beiden vorhergehenden Lemmata sind wir nun gerüstet, die zentrale Aussage dieses Kapitels zu beweisen. Vergleichbare Resultate wurden von Sasaki und Hajek (1988) für Simulated Annealing bewiesen.

Theorem 4.3. *Für $\varepsilon > 0$ finden der (1+1)-EA und die RLS unabhängig vom Startpunkt der Suche in einer erwarteten Zeit von $O(m^{2\lceil 1/\varepsilon \rceil})$ ein $(1+\varepsilon)$-optimales Matching.*

Beweis. Die erste Phase der Suche endet, sobald ein Matching gefunden ist, nach erwartet $O(m \log m)$ Schritten (Lemma 4.1). Diese Zeit ist für alle Werte von ε klein genug. Danach sei M das Matching, das der aktuelle Suchpunkt beschreibt, und es sei M^* ein beliebiges Maximummatching. Wir betrachten nur Situationen mit $|M^*| > (1+\varepsilon)|M|$. Andernfalls ist das Ziel der Suche schon erreicht. Daraus folgt

$$\frac{|M|}{|M^*| - |M|} < \frac{|M|}{(1+\varepsilon)|M| - |M|} = \frac{1}{\varepsilon}$$

und somit

$$\left\lfloor \frac{|M|}{|M^*| - |M|} \right\rfloor \leq \left\{ \begin{array}{ll} \lfloor 1/\varepsilon \rfloor, & \text{falls } 1/\varepsilon \notin \mathbb{N} \\ \lfloor 1/\varepsilon \rfloor - 1, & \text{falls } 1/\varepsilon \in \mathbb{N} \end{array} \right\} \leq \lceil 1/\varepsilon \rceil - 1.$$

Nach Lemma 4.2 gibt es einen verbessernden Pfad, dessen Länge höchstens $L := 2\lfloor |M|/(|M^*| - |M|)\rfloor + 1 \leq 2\lceil 1/\varepsilon\rceil - 1$ ist. Wir zeigen, dass für beide Heuristiken die erwartete Zeit bis zum Finden eines größeren Matchings durch $O(m^L)$ beschränkt ist. Dann ist die erwartete Zeit zum Finden eines $(1+\varepsilon)$-optimalen Matchings $O(|M^*| \cdot m^L) = O(m^{2\lceil 1/\varepsilon\rceil})$ und die Aussage des Theorems ist bewiesen.

Die Wahrscheinlichkeit, dass der nächste Schritt des (1+1)-EA nur die Kanten eines bestimmten Pfades der Länge ℓ kippt, ist $\Theta(1/m^\ell)$. Die erwartete Wartezeit auf dieses Ereignis ist $\Theta(m^\ell)$. Da es bis zum Erreichen der Approximationsgüte $1+\varepsilon$ in jedem Schritt einen verbessernden Pfad der Länge $\ell \leq L$ gibt, ist die erwartete Wartezeit auf ein größeres Matching $O(m^L)$.

Die RLS kann nicht in einem Schritt alle Kanten des ausgewählten verbessernden Pfades kippen. Wenn sie in $\lfloor \ell/2\rfloor$ aufeinanderfolgenden Schritten eines der Kantenpaare an einem der beiden Enden kippt, dann entsteht durch fortgesetztes Verkürzen ein verbessernder Pfad der Länge 1. (Verbessernde Pfade haben immer ungerade Länge ℓ.) Wenn die RLS im darauffolgenden Schritt nur die verbleibende freie Kante kippt, dann ist das Matching ebenfalls verbessert. Die Wahrscheinlichkeit, dass diese $\lfloor \ell/2\rfloor+1$ Schritte in dieser Reihenfolge eintreten, ist $\Omega((1/m^2)^{\lfloor \ell/2\rfloor} \cdot 1/m) = \Omega(1/m^\ell)$. Dabei haben wir benutzt, dass ℓ ungerade ist. Wir nennen dies eine *erfolgreiche* Phase. Eine *erfolglose* Phase bricht nach dem ersten Schritt ab, der sich von dem entsprechenden Schritt einer erfolgreichen Phase unterscheidet. Die erwartete Zahl erfolgloser Phasen vor einer erfolgreichen Phase ist $O(m^\ell)$. Die erwartete Länge einer Phase ist $O(1)$ Schritte, denn die Wahrscheinlichkeit, dass der jeweils nächste Schritt die Phase erfolgreich fortsetzt, ist $O(1/m)$. Dies gilt erst recht unter der Bedingung, dass eine Phase erfolglos ist. Die Länge einer erfolgreichen Phase ist $\lfloor \ell/2\rfloor + 1$ Schritte. Somit ist die erwartete Zeit bis zum Ende der nächsten erfolgreichen Phase $O(m^\ell + \ell) = O(m^L)$. □

Randomisierte Varianten polynomieller Approximationsschemata für Optimierungsprobleme kann man wie folgt definieren (vgl. Motwani und Raghavan, 1995, Witt, 2004).

Definition 4.4 (PRAS, FPRAS). *Ein* polynomielles, randomisiertes Approximationsschema *(engl.* polynomial randomized approximation scheme, PRAS*) für ein Optimierungsproblem ist ein randomisierter Algorithmus, der als Eingabe eine Probleminstanz I sowie ein $\varepsilon > 0$ erhält und in polynomiell in der Länge von I beschränkter Zeit mit einer Wahrscheinlichkeit von mindestens* 3/4 *eine Lösung berechnet, deren Approximationsgüte nicht schlechter als $1+\varepsilon$ ist.*

Ein echt polynomielles, randomisiertes Approximationsschema *(engl.* fully polynomial randomized approximation scheme, FPRAS*) ist ein PRAS, dessen Laufzeit polynomiell in der Länge von I und $1/\varepsilon$ beschränkt ist.*

Die Konstante 3/4 in Definition 4.4 wurde in Übereinstimmung mit der Definition von Motwani und Raghavan (1995) gewählt, die jedoch auf Zahlprobleme ausgerichtet ist. Für Optimierungsprobleme darf man in Definition 4.4 den Wert 3/4 durch jede konstante Wahrscheinlichkeit $\delta > 0$ ersetzen. Durch k unabhängige Läufe des randomisierten Algorithmus kann man die Erfolgswahrscheinlichkeit dann auf mindestens $1 - (1 - \delta)^k$ erhöhen, indem man die beste Lösung aus allen Läufen wählt. Das bedeutet, dass man durch konstant viele Wiederholungen jede konstante Erfolgswahrscheinlichkeit erreichen kann.

Im Folgenden bezeichnen wir mit c den konstanten Faktor, der sich in dem O-Term der Laufzeitschranke in Theorem 4.3 verbirgt. Im Beweis zu Theorem 4.3 haben wir c zwar nicht genau bestimmt, es ist aber leicht möglich, c durch präzisere Abschätzungen nach oben zu beschränken. Treffender ist es daher zu sagen, dass c eine obere Schranke für den Faktor in der Laufzeitschranke sein soll.

Mithilfe der Markoffungleichung (siehe Anhang A) folgt aus Theorem 4.3, dass der (1+1)-EA und die RLS in $4cm^{2\lceil 1/\varepsilon\rceil}$ Schritten mit einer Wahrscheinlichkeit von mindestens 3/4 eine $(1 + \varepsilon)$-Approximation eines Maximummatchings finden. Wir können also ein Stoppkriterium einführen, das die Heuristiken nach $4cm^{2\lceil 1/\varepsilon\rceil}$ Schritten anhält. Der dann aktuelle Suchpunkt ist die Ausgabe dieser Algorithmen. Haben wir damit gezeigt, dass die RLS und der (1+1)-EA polynomielle, randomisierte Approximationsschemata für das Matchingproblem sind? Noch nicht ganz, denn wir messen die Laufzeit bisher in der Zahl der Fitnessauswertungen. Es ist aber klar, dass der Aufwand für eine Fitnessberechnung durch ein Polynom $p(m)$ beschränkt ist, sodass die Laufzeit im uniformen Kostenmaß durch ein Polynom $p'(m) = 4cm^{2\lceil 1/\varepsilon\rceil} \cdot p(m)$ beschränkt ist. Für ein PRAS müssen wir die Laufzeit in der Länge der Eingabe I angeben. In unserem Fall ist I eine (explizite) Darstellung des Graphen G. Da die Eingabe jede Kante des Graphen aufzählen muss, gilt $m = O(|I|)$. Damit haben wir folgendes Korollar gezeigt.

Korollar 4.5. *Die RLS und der (1+1)-EA, ausgestattet mit einer effizienten Implementierung der Fitnessfunktion und der Stoppregel, die die Heuristiken nach* $4cm^{2\lceil 1/\varepsilon\rceil}$ *Schleifendurchläufen anhält, sind polynomielle Approximationsschemata für das Matchingproblem. Sie finden für jedes konstante* $\varepsilon > 0$ *mit einer Wahrscheinlichkeit von mindestens* $3/4$ *in polynomieller Zeit eine* $(1+\varepsilon)$*-Approximation eines Maximummatchings.*

An dieser Stelle tritt die Frage auf, ob die beiden Heuristiken vielleicht sogar *echt polynomielle,* randomisierte Approximationsschemata sind. Schließlich haben wir uns im Beweis von Theorem 4.3 nur auf *einen* verbessernden Pfad konzentriert und auf sehr spezielle Ereignisse gesetzt, die das Matching verbessern. Wir glauben daher nicht, dass Theorem 4.3 eine scharfe obere Schranke angibt. Mit einem Vorgriff auf Ergebnisse aus Kapitel 6 und Kapitel 7 können wir ausschließen, dass eine der beiden Heuristiken ein FPRAS ist. Dort zeigen wir, dass es Graphen gibt, für die beide Heuristiken exponentielle erwartete Laufzeit haben; sogar, dass die Laufzeit nur mit exponentiell kleiner Wahrscheinlichkeit (d. h. $e^{-\Omega(m^c)}$) weniger als exponentiell groß ist. Das hat folgende Auswirkung. Wenn wir $\varepsilon = 1/m$ wählen, dann suchen wir nach Approximationen von Maximummatchings, deren Größe um weniger als eine Kante von Maximummatchings abweicht. Für diese Wahl muss ein FPRAS das Problem exakt lösen, d. h. in polynomieller Zeit in m und $1/\varepsilon = m$ mit einer Wahrscheinlichkeit von mindestens 3/4 ein Maximummatching liefern. Für den (1+1)-EA und die RLS widerspricht das offensichtlich den angekündigten Resultaten.

5 Graphen mit polynomieller erwarteter Optimierzeit

Im vorhergehenden Kapitel haben wir gesehen, dass der (1+1)-EA und die RLS Maximummatchings effizient approximieren können. Sie finden gute Approximationen in erwartet polynomieller Zeit. Wir haben auch schon auf die Grenzen der Heuristiken vorgegriffen, nämlich, dass Maximummatchings nicht immer in polynomieller Zeit erreichbar sind (siehe Kapitel 6 und Kapitel 7). Diese Ergebnisse beziehen sich jedoch auf Graphen, die speziell dafür entworfen wurden, Heuristiken in die Irre zu führen. Für einfach strukturierte Graphen sind unsere Heuristiken aber häufig wohl in der Lage, effizient Maximummatchings zu finden. In diesem und den beiden folgenden Kapiteln geben wir uns daher nicht mehr mit Approximationen zufrieden, sondern richten den Blick auf den gesamten Optimierungsprozess bis zum Erreichen eines Optimums.

Als Beispiel für eine Graphklasse, für die beide Heuristiken effizient Maximummatchings finden, untersuchen wir in diesem Kapitel Bäume. Wir beginnen mit sehr einfach strukturierten Bäumen, nämlich Bäumen, die nur aus einem Pfad bestehen. Wir bezeichnen diese Bäume hier als *Pfadgraphen*, um sie sprachlich von den immer wieder betrachteten verbessernden Pfaden unterscheiden zu können. Die Analyse ist für Pfadgraphen noch nicht sehr schwierig, bietet aber die Gelegenheit, die Konzepte später folgender Analysen vorzustellen und noch oft verwendete „Werkzeuge“ und Vorgehensweisen einzuführen. Danach wenden wir uns vollständigen Bäumen und Bäumen ohne weitere Einschränkungen zu.

5.1 Pfadgraphen

Definition 5.1 (Pfadgraph). *Der Pfadgraph P_m mit $m \in \mathbb{N}$ Kanten ist der ungerichtete Graph mit Knotenmenge $V_m := \{0, \ldots, m\}$ und Kantenmenge*

$$E_m := \big\{\{0,1\}, \{1,2\}, \ldots, \{m-1, m\}\big\}.$$

Maximummatchings für Pfadgraphen wählen jede zweite Kante entlang des Pfades aus und haben Kardinalität $\lceil m/2 \rceil$. Für ungerade m ist das Maximummatching eindeutig bestimmt, andernfalls gibt es offenbar zwei Maximummatchings, die gerade invers zueinander sind.

Die Analysen in diesem Kapitel konzentrieren sich ähnlich wie in Kapitel 4 zu jeder Zeit auf einen kürzesten verbessernden Pfad. Wir sind an der erwarteten

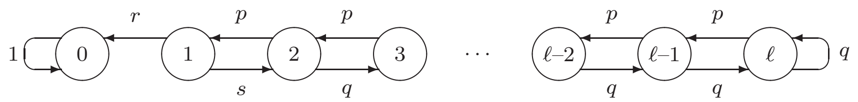

Bild 5.1: Die Markoffkette M aus Lemma 5.2

Zeit interessiert, bis dieser verbessernde Pfad Länge 0 erreicht. Das entspricht dem Ereignis, dass das Matching mithilfe des Pfades verbessert wurde. Indem wir die Wahrscheinlichkeiten für Verlängerungen und Verkürzungen dieses Pfades in einem Schritt der Heuristiken abschätzen, gelingt es, die Längenänderung dieses Pfades als Irrfahrt zu modellieren. Diese Irrfahrt hat Ähnlichkeit mit dem Problem, das in der Literatur unter dem Begriff Gambler's-Ruin-Problem bekannt ist (z. B. in Feller, 1968, Schickinger und Steger, 2001). In Abschnitt 6.1 werden wir dieses Problem noch genauer kennen lernen. An dieser Stelle bereiten wir es mit Lemma 5.2 so auf, dass wir es in diesem Kapitel gut einsetzen können. Die uns interessierenden Irrfahrten können nämlich durch die folgende Markoffkette beschrieben werden.

Lemma 5.2. *Für Wahrscheinlichkeiten p und r, $0 < p, r < 1$, sei $q := 1 - p$ und $s := 1 - r$. Gegeben sei die endliche, zeithomogene Markoffkette M mit Zustandsmenge $S := \{0, \dots, \ell\}$, Startzustand $\ell \geq 2$ und den positiven Übergangswahrscheinlichkeiten $p(0,0) := 1$, $p(1,0) := r$, $p(1,2) := s$, $p(i, i-1) := p$ für $i \in \{2, \dots, \ell\}$, $p(i, i+1) := q$ für $i \in \{2, \dots, \ell-1\}$, und $p(\ell, \ell) := q$. (siehe Bild 5.1). Die erwartete Zeit, bis erstmalig der Zustand 0 erreicht wird, ist*

$$h_{\ell,0} = \begin{cases} \ell^2 + \left(\frac{2}{r} - 3\right)\ell - \frac{1}{r} + 2, & \text{falls } p = q = 1/2, \\[1ex] \frac{1}{q-p}\left(\frac{\left(\frac{q}{p}\right)^{\ell} - 1}{\frac{q}{p} - 1} + \frac{s}{r}\left(\frac{q}{p}\right)^{\ell-1} - \ell - \frac{s}{r}\right) + \frac{1}{r}, & \text{falls } p \neq q. \end{cases}$$

Insbesondere gilt daher $h_{\ell,0} = \ell^2 + \ell$, falls $p = q = r = s = 1/2$.

Beweis. Es sei $h_{i,j}$ die erwartete Zeit, um von Zustand i ausgehend den Zustand j zu erreichen. Wir behaupten, dass gilt:

$$h_{1,0} = \frac{1}{r} + \frac{s}{r} \cdot h_{2,1} \quad \text{und für } j \geq 2 \quad h_{j,j-1} = \begin{cases} 2(\ell - j + 1), & \text{falls } p = q = 1/2, \\[1ex] \frac{\left(\frac{q}{p}\right)^{\ell-j+1} - 1}{q - p}, & \text{falls } p \neq q. \end{cases}$$

Indem man alle Terme $h_{j,j-1}$ für $j \in \{1, \dots, \ell\}$ aufsummiert, erhält man aus dieser Behauptung die Aussage des Lemmas. Aus dem Satz von der totalen Wahrscheinlichkeit (siehe Anhang A) folgt $h_{1,0} = r \cdot 1 + s \cdot (1 + h_{2,1} + h_{1,0})$. Durch Umformen erhält man den ersten Teil der Behauptung. Den zweiten Teil beweisen wir per Induktion über j. Da man Zustand ℓ nur verlassen kann, indem man zu Zustand $\ell - 1$ geht, gilt $h_{\ell,\ell-1} = 1/p$. Dies stimmt mit der Behauptung sowohl für $p = q = 1/2$ als auch für $p \neq q$ überein. Für $\ell - 1 \geq j \geq 2$ folgt wieder mit dem Satz von der totalen Wahrscheinlichkeit, dass $h_{j,j-1} = p \cdot 1 + q \cdot (1 + h_{j+1,j} + h_{j,j-1})$. Durch Umformen erhält man

$$h_{j,j-1} \;=\; \frac{1}{p}(1 + q \cdot h_{j+1,j}).$$

Nun wenden wir die Induktionsvoraussetzung an. Dazu betrachten wir die Fälle $p = q = 1/2$ und $p \neq q$ getrennt. Im ersten Fall erhalten wir

$$h_{j,j-1} \;=\; \frac{1}{p}\Big(1 + q\big(2(\ell - (j+1) + 1)\big)\Big) \;=\; 2(\ell - j + 1),$$

im zweiten Fall erhalten wir

$$h_{j,j-1} \;=\; \frac{1}{p}\left(1 + q\frac{\left(\frac{q}{p}\right)^{\ell-j} - 1}{q-p}\right) \;=\; \frac{\frac{q}{p} - 1}{p(\frac{q}{p} - 1)} + \frac{\left(\frac{q}{p}\right)^{\ell-j+1} - \frac{q}{p}}{q-p} \;=\; \frac{\left(\frac{q}{p}\right)^{\ell-j+1} - 1}{q-p}.$$

Damit ist der Beweis der Behauptung und des Lemmas abgeschlossen. □

Wir werden im Folgenden häufig nicht alle, sondern nur ausgewählte Schritte der Heuristiken betrachten, die wir *relevante* Schritte nennen. Die Definition eines relevanten Schrittes wird i. Allg. von dem Ziel abhängen, das wir in der Analyse verfolgen, und vom aktuellen Suchpunkt.

Wenn jeder Schritt des Algorithmus mit einer Wahrscheinlichkeit von mindestens p relevant ist, dann ist die erwartete Wartezeit auf einen relevanten Schritt höchstens $1/p$ Schritte. Wenn die Zufallsvariable R die Zahl relevanter Schritte bezeichnet, um ein bestimmtes Ziel zu erreichen, und $E(R)$ ihren Erwartungswert, dann ist das Produkt $E(R) \cdot 1/p$ eine obere Schranke für die erwartete Zahl Schritte bis zum Erreichen des Ziels. Die zugrundeliegenden Zufallsvariablen sind unabhängig, denn es ist für den nächsten relevanten Schritt unerheblich, wie lange darauf gewartet werden musste.

Theorem 5.3. *Unabhängig vom Startpunkt der Suche ist für P_m die erwartete Optimierzeit der RLS $O(m^4)$.*

Beweis. Die erwartete Zeit bis zum Erreichen eines Matchings ist klein genug (Lemma 4.1). Für $m \leq 4$ ist die (verbleibende) erwartete Optimierzeit offensichtlich durch eine Konstante beschränkt. Wir betrachten im Folgenden nur noch den Fall $m \geq 5$. Wenn die Größe des aktuellen Matchings $|M^*| - i$ ist, dann gibt es mindestens i verbessernde Pfade (Lemma 4.2). (Tatsächlich sind es sogar $2i - 1$, darunter i knotendisjunkte verbessernde Pfade.) Die Länge mindestens eines verbessernden

Pfades ist daher höchstens $\ell := \lfloor m/i \rfloor$. Nach jedem Schritt wählen wir einen kürzesten verbessernden Pfad aus und nennen diesen P. Der nächste Schritt heißt nun P-relevant, wenn er akzeptiert wird und die Länge des zuvor ausgewählten Pfades P verändert.

Die Wahrscheinlichkeit eines P-relevanten Schritts ist $\Omega(1/m^2)$: Wenn die Länge von P mindestens 3 ist, ist diese Wahrscheinlichkeit durch die Wahrscheinlichkeit des Ereignisses nach unten beschränkt, dass das Kantenpaar aus den ersten (oder letzten) beiden Kanten von P kippt. Andernfalls, bei Länge 1, ist sie durch die Wahrscheinlichkeit des Ereignisses nach unten beschränkt, dass nur die eine Kante von P kippt. Dieses Ereignis hat dann sogar Wahrscheinlichkeit $\Omega(1/m)$. Wenn wir zeigen können, dass eine erwartete Zahl von $O(\ell^2)$ P-relevanten Schritten das Matching um mindestens eine Kante verbessert, dann reichen erwartet $\sum_{1 \le i \le \lceil m/2 \rceil} O((m/i)^2) = O(m^2)$ P-relevante Schritte für ein Maximummatching. Aus unserer Vorüberlegung zu relevanten Schritten folgt, dass die erwartete Optimierzeit $O(m^4)$ ist.

Die Länge $|P|$ des ausgewählten, kürzesten verbessernden Pfades nimmt nur ungerade Werte an. Wenn $|P|$ mindestens 3 ist, kann es keine wählbaren Kanten geben. Daher können nur Mutationsschritte akzeptiert werden, in denen die Matchinggröße unverändert bleibt. Weil die RLS höchstens zwei Kanten in einem Schritt kippen kann, bedeutet das, dass eine freie Kante e und eine Matchingkante e' kippen müssen. Da in dieser Situation jede freie Kante e mindestens eine Nachbarkante e' im Matching besitzt, muss e' auch kippen, wenn e kippt. Das bedeutet, eine freie Kante e und eine Matchingkante e' können nur kippen, wenn die freie Kante e mit einem freien Endknoten inzidiert und an ihrem anderen Endpunkt an die Matchingkante e' grenzt. Für P impliziert diese Beobachtung, dass in einem P-relevanten Schritt nur eines der Kantenpaare $p_1, \dots, p_4$ in Bild 5.2 kippen kann. Für $|P| \ge 3$ existieren zumindest die Paare p_2 und p_3. Daher schrumpft P mindestens mit Wahrscheinlichkeit 1/2 um zwei Kanten in jedem P-relevanten Schritt und wächst um zwei Kanten mit Wahrscheinlichkeit höchstens 1/2.

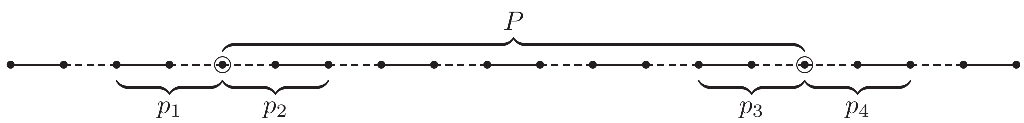

Bild 5.2: In einem P-relevanten Schritt kippt eines der Kantenpaare $p_1, \dots, p_4$

Wenn P nur noch Länge 1 hat, dann gibt es mindestens eine wählbare Kante e. Falls im nächsten Schritt nur e kippt, dann schrumpft P auf Länge 0, d. h., P verschwindet und das Matching ist um eine Kante verbessert. Die Wahrscheinlichkeit dieses Ereignisses ist mindestens $1/(2m)$. Die Wahrscheinlichkeit, dass P im nächsten Schritt auf Länge 3 anwächst, ist höchstens $2 \cdot (1/2) \cdot \binom{m}{2}^{-1} = 2/(m(m-1))$, weil dazu eines der Paare p_1 oder p_2 kippen muss. Unter der Bedingung, dass der nächste

Schritt relevant ist, ist die Wahrscheinlichkeit für eine Verkürzung, mindestens

$$\frac{\frac{1}{2m}}{\frac{1}{2m}+\frac{2}{m(m-1)}} = \frac{1}{1+\frac{4}{m-1}} \geq \frac{1}{2} \qquad \text{für } m \geq 5.$$

Wenn wir nun für jeden P-relevanten Schritt die Wahrscheinlichkeit für eine Verkürzung durch die untere Schranke 1/2 und die Wahrscheinlichkeit für eine Verlängerung durch die obere Schranke 1/2 ersetzen, dann können wir die erwartete Zahl P-relevanter Schritte bis zur Matchingverbesserung mithilfe der Markoffkette aus Lemma 5.2 abschätzen. Wir identifizieren die Länge $j \in \{0, 1, 3, 5, \ldots, \ell\}$ von P mit dem Zustand $\lceil j/2 \rceil$ der Markoffkette M aus Lemma 5.2 für $p = r = 1/2$ und Zustandsmenge $\{0, \ldots, \lceil \ell/2 \rceil\}$. Dadurch ersetzen wir den nicht zeithomogenen Prozess, der durch den Optimierungsprozess beschrieben wird, durch einen zeithomogenen Prozess. Für den zeithomogenen Prozess gilt für alle k, dass die Wahrscheinlichkeit nach t Schritten in einem (d. h. in irgendeinem, aber keinem bestimmten) der Zustände aus $\{0, \ldots, k-1\}$ zu sein, nur kleiner geworden ist als nach derselben Zahl an Schritten im ursprünglichen Prozess. Wenn wir pessimistisch annehmen, dass die Länge von P zu Beginn ℓ ist, ist die erwartete Zahl relevanter Schritte $O(\ell^2)$, um das Matching zu verbessern. □

Die RLS unterscheidet sich darin vom (1+1)-EA, dass im Mutationsschritt viele Bits gleichzeitig kippen können. In der Regel macht das die Analyse des (1+1)-EA schwieriger. Die Kernidee der Laufzeitanalyse des (1+1)-EA auf Pfadgraphen haben wir in der Analyse für die RLS schon kennen gelernt. Um mit vielen kippenden Kanten umgehen zu können, führen wir in der Analyse *saubere Schritte* ein. Saubere Schritte sind relevante Schritte, die sich im Wesentlichen wie die relevanten Schritte der RLS verhalten. Die Grundidee ist es nun zu zeigen, dass eine Phase, die $\Theta(\ell^2)$ relevante Schritte des (1+1)-EA einschließt, mit mindestens konstanter Wahrscheinlichkeit nur saubere relevante Schritte enthält. So eine Phase nennen wir eine *saubere Phase*. Da eine saubere Phase so viele relevante Schritte enthält wie die erwartete Zahl relevanter Schritte zum Verbessern des Matchings, kann sie das Matching mit mindestens konstanter Wahrscheinlichkeit verbessern.

Theorem 5.4. *Unabhängig vom Startpunkt der Suche ist für P_m die erwartete Optimierzeit des (1+1)-EA $O(m^4)$.*

Beweis. Die Definition von P, ℓ und P-relevant übernehmen wir aus dem Beweis zu Theorem 5.3. Für den (1+1)-EA gelten dieselben asymptotischen Schranken für die Wahrscheinlichkeiten für spezielle 1- und 2-Bit-Mutationen wie für die RLS. Daher folgt mit denselben Argumenten wie im Beweis zu Theorem 5.3, dass es genügt zu zeigen, dass die erwartete Zahl P-relevanter Schritte $O(\ell^2)$ ist, um das Matching zu verbessern.

P-saubere Schritte definieren wir für Situationen, in denen es keine wählbaren Kanten gibt. Es seien u und v die Endpunkte von P und die Umgebung $U_u :=$

$\{\{w, z\} \in E \mid \mathrm{dist}(u, w) \leq 3\}$ sei die Menge der Kanten, sodass ein Endpunkt jeder Kante in der Menge höchstens einen Abstand von 3 Kanten zu u hat. Die Umgebung U_v ist analog definiert (siehe Bild 5.3). Ein P-relevanter Schritt heißt *P-sauber*, wenn

– höchstens 3 Kanten in $U := U_u \cup U_v$ kippen und

– höchstens 2 kippende Kanten aus U Nachbarn sind.

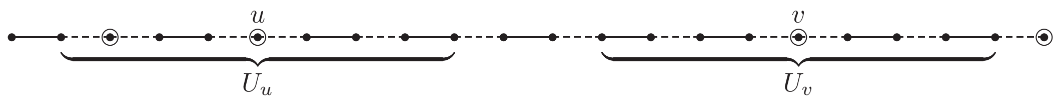

Bild 5.3: Umgebungen U_u und U_v

Wir beschreiben die Wirkung P-sauberer Schritte auf P. Die freien Knoten teilen den Pfadgraphen in alternierende Pfade ein. Als Beispiel kann man Bild 5.3 betrachten. Weil wir nur Situationen ohne wählbare Kanten betrachten, liegt zwischen einem freien Knoten und dem nächsten freien Knoten ein verbessernder Pfad aus mindestens 3 Kanten. An den Enden des Pfadgraphen, zwischen dem ersten Knoten des Graphen und dem ersten freien Knoten, kann es auch alternierende Pfade gerader Länge geben. Deren Länge ist mindestens 2. Ein P-sauberer Schritt kann nicht alle Kanten von P kippen, denn dazu müsste ein Block aus 3 oder mehr Kanten in U kippen. Daher verschwindet der gewählte Pfad auch nicht in einem P-sauberen Schritt, er verlängert oder verkürzt sich nur. Es ist jedoch möglich, dass in einem P-sauberen Schritt zwischen den Endpunkten von P zwei oder mehr neue freie Knoten entstehen. Zwischen diesen neuen freien Knoten liegen dann wieder verbessernde Pfade. Dieses Ereignis interpretieren wir als Verkürzung von P. Da verbessernde Pfade immer ungerade Länge haben, handelt es sich um eine Verkürzung um 2 oder ein Vielfaches von 2. Es ist nicht möglich, dass ein P-sauberer Schritt P um mehr als 2 Kanten, d. h. um mindestens 4 Kanten, verlängert. Dies würde erfordern, dass mehr als 3 Kanten aus U kippten. Wenn P-saubere Schritte P verlängern, dann um 2 Kanten und dazu ist es notwendig, dass eines der Kantenpaare p_1 oder p_4 in Bild 5.2 kippt. Damit ein P-sauberer Schritt P verkürzt, reicht es, dass eines der Kantenpaare p_2 oder p_3 kippt (und sonst keine weitere Kante). Da höchstens 3 Kanten in U kippen dürfen, kann auch höchstens eines der 4 Paare $p_1, \ldots, p_4$ kippen. Also ist die Wahrscheinlichkeit für eine Verlängerung in einem P-sauberen Schritt höchstens $1/2$ und für eine Verkürzung mindestens $1/2$, wobei Verlängerungen nur um 2 Kanten möglich sind, während Verkürzungen auch größer ausfallen können.

Eine Folge von Schritten, in der alle P-relevanten Schritte P-sauber sind, nennen wir eine P-saubere Phase. In unserer Analyse ist das Ziel einer P-sauberen Phase, eine wählbare Kante zu erzeugen oder das Matching zu verbessern. Damit unsere Definition des Begriffs eines P-sauberen Schritts diesem Ziel nicht im Wege steht, erweitern wir ihn. In Situationen ohne wählbare Kanten sind zusätzlich alle akzeptierten Schritte P-relevant und P-sauber, wenn sie eine wählbare Kante erzeugen oder das Matching verbessern. Damit erhöhen wir nur die Wahrscheinlichkeit P-relevanter Schritte, sodass unsere bisherigen Überlegungen zu relevanten Schritten unberührt bleiben.

Nun schätzen wir die Wahrscheinlichkeit nach oben ab, dass ein Schritt P-relevant und nicht P-sauber ist. Wir erinnern uns, dass wir nur über Situationen ohne wählbare Kanten sprechen. Um die erste Eigenschaft zu verletzen, müssen mindestens 4 aus höchstens 16 Kanten in U kippen. Dieses Ereignis hat nur eine Wahrscheinlichkeit von $O(1/m^4)$. Für die zweite Eigenschaft brauchen wir nun nur noch solche Mutationsschritte zu betrachten, die nicht schon die erste Eigenschaft verletzen. Das bedeutet, es bleiben nur Mutationsschritte, die einen Block B aus drei benachbarten Kanten in U kippen. Ein solcher Schritt erzeugt lokal in B einen Überschuss von entweder einer freien Kante oder einer Matchingkante. Wird dieser Überschuss nicht anderswo ausgeglichen, wird der Schritt gar nicht erst akzeptiert, da sich das Matching verschlechtert, oder der Schritt ist per Definition sauber, weil sich das Matching verbessert.

Um einen Überschuss von einer freien Kante auszugleichen, muss außerhalb von B in der Summe eine freie Kante mehr kippen als Matchingkanten. Falls B an einer der bis zu 4 Grenzen von U liegt, kann diese freie Kante an B grenzen, aber außerhalb von U liegen. Die Wahrscheinlichkeit für einen solchen Schritt ist $O(m^4)$. Wenn B nicht an einer Grenze von U liegt, dann muss mindestens ein zweiter Block B' aus 3 benachbarten Kanten außerhalb von U kippen, da es keine wählbaren Kanten gibt. Für B und B' gibt es nur $O(1)$ bzw. $O(m)$ Möglichkeiten, diese zu platzieren. Also ist die Wahrscheinlichkeit $O(1/m^5)$.

Um einen Überschuss von einer Matchingkante auszugleichen, könnte eine andere Matchingkante kippen. Doch dann würde eine wählbare Kante erzeugt und der Schritt wäre per Definition sauber. Also muss außerhalb von B ein Block B' aus mindestens 3 Kanten kippen. Die Wahrscheinlichkeit für dieses Ereignis ist wieder $O(1/m^5)$.

Zusammengefasst ist die Wahrscheinlichkeit für einen P-relevanten, aber nicht P-sauberen Schritt $O(1/m^4)$. Die bedingte Wahrscheinlichkeit, dass ein Schritt nicht P-sauber ist, gegeben, der Schritt ist P-relevant, ist $O(1/m^4)/\Omega(1/m^2) = O(1/m^2)$. Also ist eine Phase, die $O(\ell^2) = O(m^2)$ P-relevante Schritte einschließt, mit Wahrscheinlichkeit $\Omega(1)$ P-sauber (für m genügend groß).

Wir ersetzen Verkürzungen von P durch Verkürzungen um genau 2 Kanten und ersetzen die Wahrscheinlichkeit für eine Verkürzung in P-sauberen Schritten durch die untere Schranke 1/2. Wenn wir die Markoffkette in Lemma 5.2 für $p = r = 1/2$ betrachten, dann ist die erwartete Zeit, um vom Startzustand den Zustand 1 zu erreichen, nur kleiner als die erwartete Zeit, um den Zustand 0 zu erreichen. Mit denselben Argumenten wie im Beweis zu Theorem 5.3 für die RLS ist die erwartete

Zahl P-sauberer Schritte zum Erreichen von Zustand 1, d. h. $|P| = 1$, durch $O(\ell^2)$ beschränkt. Aus der Markoffungleichung folgt, dass dies mit einer Wahrscheinlichkeit von $\Omega(1)$ in einer Folge von $c\ell^2$ P-sauberen Schritten eintritt. Zusammen mit dem Ergebnis des letzten Absatzes erhalten wir, dass $O(\ell^2)$ P-relevante Schritte mit einer Wahrscheinlichkeit von $\Omega(1)$ eine wählbare Kante erzeugen (oder das Matching verbessern).

Wenn es mindestens eine wählbare Kante e gibt, ist der nächste Schritt mit Wahrscheinlichkeit $\Omega(1/m)$ relevant, nämlich wenn er nur e kippt und somit das Matching verbessert. Um im nächsten Schritt alle wählbaren Kanten verschwinden zu lassen, muss insbesondere e nicht wählbar werden. Dazu ist es notwendig, dass entweder e oder mindestens eine Nachbarkante von e kippt. So ein Schritt kann also höchstens Wahrscheinlichkeit $O(1/m)$ haben. Daher verbessert der nächste relevante Schritt, der auf eine erfolgreiche P-saubere Phase folgt, das Matching mindestens mit einer Wahrscheinlichkeit von $\Omega(1)$. Insgesamt haben wir also gezeigt, dass es eine geeignete Zahl $N = O(\ell^2)$ gibt, sodass die nächsten N P-relevanten Schritte das Matching mit mindestens einer konstanten Wahrscheinlichkeit $\delta > 0$ verbessern. Falls die ersten N relevanten Schritte nicht erfolgreich sind, ist die Erfolgswahrscheinlichkeit für die nächsten N relevanten Schritte wieder mindestens δ. Die erwarte Zahl an Wiederholungen bis zur Matchingverbesserung ist also höchstens $1/\delta$, eine Konstante. Daraus folgt, dass die erwartete Zahl P-relevanter Schritte höchstens $O(\ell^2)$ ist. $\square$

Die Laufzeitschranke $O(m^4)$ ist groß im Vergleich zu den besten bekannten deterministischen Algorithmen, die auf das Matchingproblem für beliebige Graphen spezialisiert sind ($O(n^{1/2} \cdot m)$, Micali und Vazirani (1980)). Wir zählen hier zwar nur die Zahl der Fitnessauswertungen, wissen aber, dass die erwartete Zeit für eine Fitnessauswertung $\Theta(1)$ ist, sodass der Vergleich zulässig ist.

Die Laufzeit der RLS und des (1+1)-EA kommt schon zu einem großen Teil dadurch zustande, dass diese Heuristiken unplanmäßig suchen. Wir diskutieren, warum für bestimmte Startpunkte der Suche eine untere Schranke von $\Omega(m^4)$ gilt. Wenn es nur (noch) $O(1)$ verbessernde Pfade und keine wählbare Kante gibt, ist die Wahrscheinlichkeit, dass ein Schritt für irgendeinen der verbessernden Pfade relevant ist, lediglich $O(1/m^2)$. Das führt zu einer durchschnittlichen Wartezeit von $\Omega(m^2)$ auf einen Schritt, der irgendeinen der verbessernden Pfade verändert. Den verbleibenden Faktor von ebenfalls $\Omega(m^2)$ kann man erklären, wenn man die zweitbesten Matchings betrachtet. Ein *zweitbestes Matching* ist ein Matching der Kardinalität $|M^*| - 1$, das also nur noch um eine Kante verbessert werden kann. Wenn m ungerade ist, dann ist das Maximummatching M^* für Pfadgraphen eindeutig bestimmt. Daraus folgt, dass jedes zweitbeste Matching in diesem Fall nur genau einen verbessernden Pfad haben kann. Die Längenänderung dieses einzigen verbessernden Pfades ist dann im Wesentlichen wieder durch die Markoffkette aus Lemma 5.2 beschrieben, wobei die Übergangswahrscheinlichkeiten auf 1/2 gesetzt sind: Wenn die Anfangslänge des verbessernden Pfades $\Omega(m)$ ist, dann ist die erwar-

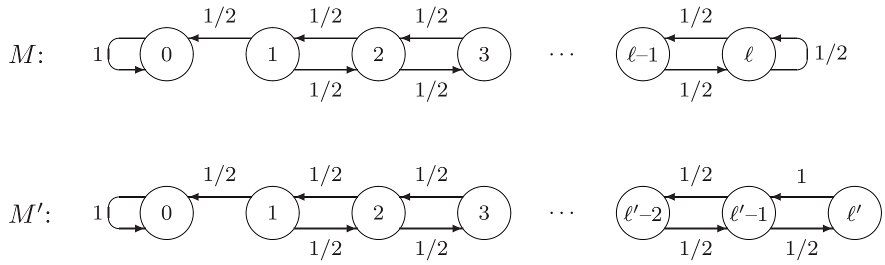

Bild 5.4: Oben die Markoffkette M aus Lemma 5.2 für $p = r = 1/2$, unten M' für $\ell' = \ell + 1$

tete Zahl relevanter Schritte, bis der Pfad kurz ist, $\Omega(m^2)$. Das ist die Beweisidee zu folgendem Theorem.

Theorem 5.5. *Für P_m, m ungerade, ist die erwartete Optimierzeit der RLS und des (1+1)-EA $\Theta(m^4)$, wenn der Startpunkt der Suche ein zweitbestes Matching beschreibt, dessen verbessernder Pfad Länge $\Omega(m)$ hat.*

Beweis. Die behaupteten oberen Schranken folgen aus Theorem 5.3 und Theorem 5.4.

Für die unteren Schranken betrachten wir zuerst folgende Markoffkette M' mit Zustandsmenge $\{0, 1, \ldots, \ell'\}$. Der Startzustand ist ℓ' und die positiven Übergangswahrscheinlichkeiten sind $p(\ell', \ell' - 1) := 1$, $p(0, 0) := 1$ und für $1 \leq i \leq \ell' - 1$ sei $p(i, i - 1) := p(i, i + 1) := 1/2$. Diese Markoffkette unterscheidet sich von der Markoffkette M aus Lemma 5.2 bei Wahl von $p = r = 1/2$ nur bez. der Wahrscheinlichkeit, mit der der Startzustand verlassen wird. Bild 5.4 stellt die Ketten für $\ell' = \ell + 1$ gegenüber. Die Markoffkette M' könnten wir zwar mit derselben Methode wie im Beweis zu Lemma 5.2 analysieren, es genügt hier aber, die beiden Ketten zu vergleichen. Offenbar ist bei Wahl von $\ell' = \ell + 1$ in M' die erwartete Zeit, um vom Zustand ℓ' zum Zustand 0 zu gelangen, nur größer als die erwartete Zeit, um in der Markoffkette M von Zustand ℓ zum Zustand 0 zu gelangen. Daher folgt mithilfe von Lemma 5.2, dass auch für M' die erwartete Zeit, um Zustand 0 zu erreichen, für eine Konstante $c > 0$ mindestens $c \cdot \ell'^2$ ist.

Wir behaupten, dass in M' für $0 < \varepsilon < 1$ die Wahrscheinlichkeit, Zustand 0 in $N := \lfloor \varepsilon c \ell'^2 \rfloor$ Schritten zu erreichen, höchstens ε ist. Wäre die Erfolgswahrscheinlichkeit größer als ε, dann wäre sie auch für jeden anderen Startzustand zwischen ℓ' und 0 größer als ε, denn diese müssen ohnehin passiert werden. Damit wäre die erwartete Zahl an Phasen der Länge N bis zum ersten Erfolg kleiner als $1/\varepsilon$. Dies widerspricht der Tatsache, dass die erwartete Zeit mindestens $c \cdot \ell'^2 \geq N/\varepsilon$ ist.

Die Anfangslänge des verbessernden Pfades P zwischen u und v sei mindestens $6L$ für $L = \Omega(m)$ und $L \in \mathbb{N}$. Die Anfangssituation stellt sich dann wie in Bild 5.5 dar.

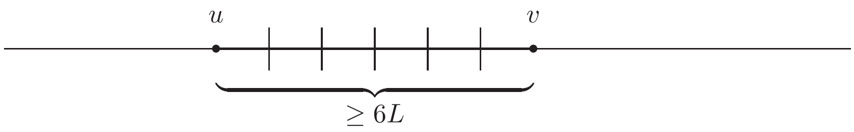

Bild 5.5: Anfangssituation

Wir beginnen mit der RLS. Wir wissen aus dem Beweis zu Theorem 5.3, dass sich P in jedem relevanten Schritt nur um 2 Kanten verkürzen oder verlängern kann. Damit P kürzer als L wird, muss mindestens eines der beiden folgenden Ereignisse eintreten: Der linke Endpunkt u bewegt sich $2L$ Positionen nach rechts oder der rechte Endpunkt v bewegt sich $2L$ Positionen nach links. Bevor eines dieser Ereignisse eingetreten ist, bewegt jeder relevante Schritt nur einen der beiden Endpunkte. Daher dürfen wir die Bewegung der Endpunkte als zwei unabhängige Irrfahrten auffassen. Gedanklich erlauben wir es den Endpunkten nicht, sich über ihre Anfangsposition hinaus nach außen zu bewegen. Das bedeutet, der Endpunkt bleibt auf seiner Anfangsposition stehen, wenn er diesen nach außen verlassen möchte. Dadurch verringern wir nur die Wartezeit auf eines der Ereignisse, da die Endpunkte dazu ohnehin zuvor zu ihren Ausgangspunkten zurückkehren müssen. Jetzt gehorcht die Bewegung eines Endpunktes bis zum Eintreten eines der beschriebenen Ereignisse der Markoffkette M' mit $\ell' = L$. Aus unseren Vorüberlegungen folgt: Die Wahrscheinlichkeit, dass mindestens einer der beiden Endpunkte sich in weniger als αm^2 P-relevanten Schritten um mindestens $2L$ Positionen nach innen bewegt, ist höchstens $1/2$, wenn wir die Konstante $\alpha > 0$ nur klein genug wählen. Aus der Definition des Erwartungswerts folgt nun, dass die erwartete Zahl P-relevanter Schritte bis zum Eintreten eines der beiden Ereignisse $\Omega(m^2)$ ist. Daraus folgt die untere Schranke für die Laufzeit der RLS, da jeder Schritt nur mit einer Wahrscheinlichkeit von $\Theta(1/m^2)$ P-relevant ist.

Für den (1+1)-EA haben die gerade betrachteten 2-Bit-Mutationen ebenfalls Wahrscheinlichkeit $\Theta(1/m^2)$, sodass wir die Analyse für 2-Bit-Mutationen von der RLS übernehmen können. Zusätzlich müssen wir aber auch Schritte berücksichtigen, in denen viele Bits kippen. Solange P mindestens Länge L hat, ist die Wahrscheinlichkeit eines Schritts, der genau die Kanten von P kippt, klein, nämlich $m^{-\Omega(m)}$. Da dies die einzige Möglichkeit ist, das Matching zu verbessern, können sonst nur Schritte akzeptiert werden, in denen eine gerade Anzahl von Bits kippt. Pessimistisch nehmen wir an, dass jeder akzeptierte Schritt, in dem $2k$, $k \geq 2$, Bits kippen, P um ebenso viele Kanten verkürzt. In der Situation eines zweitbesten Matchings können nur solche Schritte akzeptiert werden, in denen die kippenden Kanten ein oder zwei Blöcke bilden, sodass die erste oder letzte Kante eines Blocks einen der freien Endpunkte von P überdeckt. Es gibt also nur $O(k)$ Möglichkeiten für akzeptierte $2k$-Bit-Mutationen. Die erwartete Verkürzung von P durch $2k$-Bit-Mutationen pro Schritt ist höchstens $2k \cdot O(k/m^{2k}) = O(k^2/m^{2k})$. In der Summe über alle k ist sie höchstens

$$\sum_{2\le k\le\lfloor m/2\rfloor} O\left(\frac{k^2}{m^{2k}}\right) = O\left(\frac{1}{m^4}\right) + \sum_{4\le k\le\lfloor m/2\rfloor} O\left(\frac{1}{m^{2k-2}}\right) = O\left(\frac{1}{m^4}\right).$$

In βm^4 Schritten ist die erwartete Verkürzung insgesamt nur $O(1)$ und, wenn die Konstante β klein genug gewählt ist, dann folgt aus der Markoffungleichung, dass die Verkürzung mit einer Wahrscheinlichkeit von $1 - o(1)$ kleiner als L ausfällt. Für jede Wahl von $\beta \le \alpha$ folgt aus der Analyse der RLS, dass 2-Bit-Mutationen in βm^4 Schritten den Pfad mindestens mit einer Wahrscheinlichkeit von $1/2$ höchstens auf Länge $2L$ verkürzen. Zusammen mit den $2k$-Bit-Mutationen für $k \ge 2$ ist die Wahrscheinlichkeit $\Omega(1)$, dass die Pfadlänge in dem abgesteckten Zeitraum nie unter L fällt. Die Wahrscheinlichkeit, dass in βm^4 Schritten eine Mutation stattfindet, die genau die Kanten von P kippt, ist mit $m^{-\Omega(m)}$ klein genug. Insgesamt bleibt es dabei, dass mit Wahrscheinlichkeit $\Omega(1)$ der Pfad in βm^4 Schritten nicht kürzer als L wird. Wie im Fall der RLS folgt daraus die behauptete Aussage für den Erwartungswert. □

Die Projektgruppe 427 am Fachbereich Informatik der Universität Dortmund hat theoretische Analysen von EAs durch systematische Experimente überprüft(vgl. Briest, Brockhoff, Degener, Englert, Gunia, Heering, Jansen, Leifhelm, Plociennik, Röglin, Schweer, Sudholt, Tannenbaum und Wegener (2004a,b)). Die Vorgehensweise der Gruppe war es, zunächst Hypothesen aufzustellen, um dann geeignete Experimente zu entwerfen, mit denen die Hypothesen überprüft werden sollten. Neben anderen kombinatorischen Optimierungsproblemen hat sich die Gruppe auch mit der RLS und dem (1+1)-EA für das Matchingproblem beschäftigt. Aus Theorem 5.5 ist nicht erkennbar, ob auch eine untere Schranke von $\Omega(m^4)$ gilt, wenn die Suche mit einem zufälligen Startpunkt beginnt. Ein entsprechender Beweis ist nicht gelungen. Eine Hypothese der Gruppe war es daher, dass der (1+1)-EA und die RLS für Pfadgraphen eine erwartete Optimierzeit von $\Theta(m^4)$ hat, wenn der Startpunkt der Suche zufällig gewählt wird. Dazu wurden Pfadgraphen mit einer Knotenzahl von $n = m + 1$ zwischen 10 und 100 in 10er-Schritten untersucht und für jedes dieser n der Mittelwert über 1000 Läufe gebildet. Die Mittelwerte wurden nun durch Polynome vom Grad 3 und Grad 4 approximiert. Die Koeffizienten der Polynome wurden mithilfe von Regressionsanalysen ermittelt. Die besseren Ergebnisse ergaben sich zwar für den Polynomgrad 4 jedoch mit überraschend kleinen Koeffizienten vor dem n^4-Term von etwa 0,05 für den (1+1)-EA und etwa 0,02 für die RLS. Die Gruppe kommt zu dem Schluss, dass die Ergebnisse die Hypothese zwar untermauern, die kleinen Koeffizienten jedoch Zweifel aufwerfen. Dies könnte erklären, warum der Beweis einer unteren Schranke von $\Omega(m^4)$ für einen zufälligen Startpunkt bisher nicht gelungen ist.

5.2 Vollständige Bäume

Pfadgraphen erscheinen schwierig für Heuristiken, weil verbessernde Pfade die maximal mögliche Länge von m Kanten annehmen können, wenn der Optimierungsprozess bei zweitbesten Matchings angekommen ist. Wir haben gesehen, dass bei Pfadgraphen allein der Schritt von zweitbesten Matchings zu Maximummatchings für die Laufzeit $\Theta(m^4)$ verantwortlich sein kann. Andererseits erscheinen Pfadgraphen leicht, weil verbessernde Pfade nur in eine Richtung verlängert werden können.

Dies ändert sich, wenn wir zu Graphklassen wechseln, die höheren Knotengrad erlauben. Wenn es keine wählbaren Kanten gibt und ein freier Knoten Knotengrad $\deg(v)$ hat, dann kann sich dieser freie Knoten in einer 2-Bit-Mutation mit gleicher Wahrscheinlichkeit in $\deg(v)$ Richtungen bewegen. Dazu müssen eine der angrenzenden freien Kanten und ihre eindeutig bestimmte Nachbarkante aus dem Matching kippen. Für freie Knoten, die Endpunkt eines einzigen verbessernden Pfades sind, bedeutet dies, dass bis zu $\deg(v) - 1$ Richtungen den Pfad verlängern können, während nur eine Richtung ihn verkürzt. Je höher der Knotengrad, desto unfairer wird dieses Spiel. Andererseits erzwingen viele Knoten mit hohem Grad in Bäumen auch viele Blätter (Knoten vom Grad 1), da der durchschnittliche Knotengrad in Bäumen weniger als 2 ist. Das führt zu geringerem Graphdurchmesser, sodass die maximale Länge verbessernder Pfade abnimmt.

Da man für Bäume noch mit einem direkten, einfachen Ansatz effizient Maximummatchings finden kann, vermuten wir, dass die Heuristiken ebenfalls dazu in der Lage sind. Daher untersuchen wir in diesem Abschnitt vollständige k-äre Bäume und im folgenden Abschnitt beliebige Bäume.

Schon bei Pfadgraphen haben wir gesehen, dass die grundsätzlichen Beweisideen in der Analyse der RLS steckten. Die Analyse des (1+1)-EA wurde lediglich durch die Tatsache erschwert, dass viele Bits gleichzeitig kippen können. Dies führte bei Pfadgraphen zur Betrachtung einer Reihe von Möglichkeiten, wie sich die kippenden Bits verteilen können. Mithilfe dieser Betrachtungen konnten wir den Einfluss „ungewollter" Mutationen, die nicht in unser Analysekonzept passen, beschränken. Bei komplizierten Graphen wird die Betrachtung dieser zahlreichen Möglichkeiten zunehmend schwieriger. Im Folgenden beschränken wir uns daher auf die Untersuchung der RLS.

Wir beginnen mit einigen vorbereitenden Lemmata, wobei sich die ersten beiden auf beliebige Bäume beziehen. Auch in Abschnitt 5.3 werden wir darauf zurückgreifen. Zunächst zeigen wir, dass das Kernproblem der Matchingverbesserung darin besteht, eine wählbare Kante zu produzieren.

Lemma 5.6. *Gegeben sei ein beliebiger Baum und ein Matching mit mindestens einer wählbaren Kante. In erwartet $O(m)$ Schritten der RLS tritt eines der beiden folgenden Ereignisse ein: Das Matching ist verbessert oder das Matching ist nicht verbessert und alle kürzesten verbessernden Pfade haben Länge 3.*

Die Wahrscheinlichkeit, dass das erste Ereignis (Matchingverbesserung) zuerst eintritt, ist mindestens $1/2$.

Beweis. Um das Lemma zu beweisen, beweisen wir die folgende Behauptung, aus der das Lemma unmittelbar folgt.

Für alle Suchpunkte, die Matchings mit mindestens einer wählbaren Kante beschreiben, gilt: Im nächsten Schritt ist die Wahrscheinlichkeit,

– dass das Matching verbessert wird, mindestens $1/(2m)$, und

– dass alle wählbaren Kanten verschwinden, ohne dass das Matching verbessert wird, höchstens $1/(2m)$.

Der erste Teil der Behauptung ist offensichtlich, da die Wahrscheinlichkeit mindestens $1/(2m)$ ist, dass nur eine der wählbaren Kanten kippt.

Mutationsschritte, die nur Matchingkanten kippen, werden verworfen, da die Fitness sinkt. Mutationsschritte, die nur freie Kanten kippen, werden verworfen, weil sie zu Nichtmatchings führen, oder sie werden akzeptiert, weil sie das Matching verbessern. Also brauchen wir für die zweite Behauptung nur *gemischte* Mutationsschritte zu betrachten, die eine freie Kante e und eine Matchingkante e' kippen. Weil offenbar e' nicht wählbar werden darf, müssen e und e' benachbart sein. Das bedeutet, dass e keine wählbare Kante sein kann. Damit eine gegebene wählbare Kante e^* verschwindet, muss e also auch mit e^* benachbart sein. Zusammengefasst heißt das, dass die freie Kante e und die wählbare Kante e^* mit demselben freien Endknoten inzidieren und e an ihrem zweiten Endpunkt die Matchingkante e' als Nachbarn hat. Wenn nun e und e' kippen, entsteht ein verbessernder Pfad der Länge 3. Wie groß ist die Wahrscheinlichkeit eines Schritts, in dem eine bestimmte wählbare Kante e^* verschwindet? Weil die Ausgangssituation ein Matching ist, legt die Wahl von e auch e' fest. Weil der zugrunde liegende Graph ein Baum ist, bestimmt die Wahl von e' auch e, weil e zwischen e' und e^* liegen muss. Daher sind alle in Frage kommenden Kantenpaare $\{e, e'\}$ paarweise disjunkt. Weil e^* selbst ausgeschlossen ist, gibt es höchstens $(m-1)/2$ solcher Kantenpaare. Die Wahrscheinlichkeit, dass gerade eines dieser Paare kippt, ist höchstens $((m-1)/2) \cdot (1/2) \cdot \binom{m}{2}^{-1} = 1/(2m)$. □

Die Aussage von Lemma 5.6 ist also, dass der nächste relevante Schritt, nachdem eine wählbare Kante produziert wurde, das Matching mit Wahrscheinlichkeit $1/2$ verbessern wird. Mit einem „relevanten Schritt“ ist hier ein Schritt gemeint, der die Situation verändert, d. h. das Matching verbessert oder alle wählbaren Kanten verschwinden lässt. Ein solcher relevanter Schritt hat mindestens eine Wahrscheinlichkeit von $\Omega(1/m)$. Das bedeutet, die Situation, dass eine wählbare Kante existiert, braucht im Durchschnitt höchstens zweimal einzutreten, bevor der nächstfolgende relevante Schritt erfolgreich das Matching verbessert.

Eingehender müssen wir nun noch die Situation untersuchen, dass es keine wählbare Kante gibt. Wir zeigen, dass ein verbessernder Pfad in einem Mutationsschritt

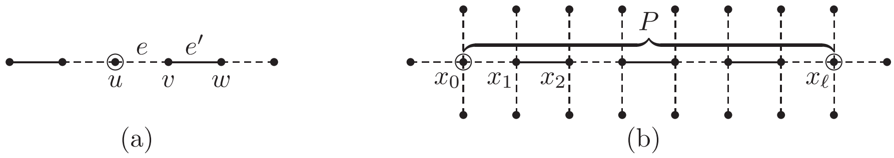

Bild 5.6: (a) In einem akzeptierten Mutationsschritt kippt eine freie Kante e, die mit einem freien Knoten u inzidiert, und eine benachbarte Matchingkante e'. (b) Ein verbessernder Pfad P und seine Nachbarschaft. Dargestellt sind nur P und unmittelbar angrenzende Knoten und Kanten.

der RLS nicht verschwinden kann. Ferner ist die Wahrscheinlichkeit für eine Verkürzung eines Pfades wesentlich durch die Knotengrade seiner Endpunkte bestimmt.

Lemma 5.7. *Gegeben sei ein Baum mit einem Matching ohne wählbare Kante. Es sei $P = (x_0, x_1, \dots, x_\ell)$ ein verbessernder Pfad. Jeder akzeptierte Mutationsschritt der RLS bewahrt die Matchinggröße und der verbessernde Pfad P*

– *bleibt unverändert,*

– *wird an einem seiner Enden um 2 Kanten verlängert oder*

– *um 2 oder ein Vielfaches von 2 Kanten an einem Ende verkürzt.*

Falls P verändert wird, ist die Wahrscheinlichkeit, dass P verkürzt wird, mindestens $2/(\deg(x_0) + \deg(x_\ell))$.

Beweis. Da es nach Voraussetzung keine wählbare Kante gibt, haben alle verbessernden Pfade mindestens Länge 3. Matchingverbesserungen sind somit ausgeschlossen. Also können nur solche Mutationsschritte akzeptiert werden, die die Matchinggröße erhalten. Das sind gemischte Mutationsschritte, die eine freie Kante e und eine Matchingkante e' kippen. Da e frei, aber nicht wählbar ist, müssen e und e' benachbart sein, d. h. $e = \{u, v\}$ und $e' = \{v, w\}$, und zusätzlich muss der Knoten u ein freier Knoten sein (siehe Bild 5.6(a)).

Wir untersuchen nun akzeptierte Schritte, die einen ausgewählten verbessernden Pfad P verändern. Formal gibt es die folgenden vier Möglichkeiten für die freie Kante $e = \{u, v\}$:

– e ist eine äußere Kante von P, nämlich $\{x_0, x_1\}$ oder $\{x_{\ell-1}, x_\ell\}$,

– e ist eine innere Kante von P,

– e gehört nicht zu P, aber ihr freier Endpunkt u,

– weder e noch ihr freier Endpunkt u gehören zu P.

Im ersten Fall gehören e' und e beide zu P (siehe Bild 5.6(b)). Sie sind die ersten beiden oder die letzten beiden Kanten von P. Offenbar verkürzt sich P um 2 Kanten, wenn e und e' kippen. Der zweite Fall kann nicht eintreten, weil alle inneren Kanten von e keinen freien Endpunkt besitzen. Im dritten Fall ist entweder $u = x_0$ oder $u = x_\ell$. Wenn e und e' kippen, verlängert sich P. Da der zweite Endpunkt w der Matchingkante zu einem freien Knoten wird, wächst P um genau zwei Kanten. Im letzten Fall verändert sich P nur, wenn der gemeinsame Knoten v von e und e' zu P gehört, d. h., die Matchingkante e' ist eine innere Kante von P. Wenn e und e' kippen, wird der Knoten w, der dann auf dem Pfad P liegt, zu einem freien Knoten. Dadurch wird entweder $(x_0, \dots, w)$ oder $(w, \dots, x_\ell)$ zu einem verbessernden Pfad, der kürzer ist als der ursprüngliche Pfad P. Da jeder verbessernde Pfad ungerade Länge hat, ist der neue verbessernde Pfad um eine gerade Zahl von Kanten kürzer als der ursprüngliche verbessernde Pfad P.

Wir haben gesehen, dass P nur wachsen kann, wenn $u = x_0$ oder $u = x_\ell$ ist. In einem solchen Schritt bestimmt die Wahl der freien Kante e, die mit x_0 oder x_ℓ inzidiert, welche Matchingkante ebenfalls kippen muss. Es gibt daher höchstens $\deg(x_0) + \deg(x_\ell) - 2$ 2-Bit-Mutationen, die P verlängern. Da es immer mindestens zwei 2-Bit-Mutationen gibt, die P verkürzen, folgt die letzte Aussage des Lemmas.

□

Wenn wir für einen Moment in Lemma 5.7 von der Möglichkeit absehen, dass der betrachtete verbessernde Pfad P auch um mehr als zwei Kanten verkürzt werden kann, bewegen sich die Endpunkte von P nach Art einer Irrfahrt auf dem Graphen. Jeder Schritt der RLS bewegt höchstens einen der beiden freien Endpunkte. Für P sind nur diese Schritte relevant. Wir warten auf das Ereignis, dass P zu einer wählbaren Kante wird, d. h., dass sich die Endpunkte von P an den Endpunkten einer Kante gegenüberstehen.

Das folgende, sehr ähnliche Verfolgungsspiel von zwei Spielsteinen ist bereits analysiert worden. Zu Beginn werden zwei Spielsteine an zwei beliebigen Knoten eines ungerichteten Graphen platziert, der das Spielfeld darstellt. In jedem Schritt wird einer der beiden Spielsteine zu einem zufällig gewählten Nachbarknoten bewegt, wobei jeder Nachbarknoten die gleiche Chance bekommt. Es ist nicht festgelegt, dass die Spielsteine abwechselnd bewegt werden. Man darf sich sogar vorstellen, dass ein böswilliger Gegner, der verhindern will, dass die Spielsteine sich treffen, bestimmt, welcher Stein im nächsten Schritt bewegt werden muss. Coppersmith, Tetali und Winkler (1993) zeigen, dass die erwartete Zeit, bis sich die Spielsteine an einem Knoten treffen, für beliebige zusammenhängende Graphen durch $O(n^3)$ beschränkt ist.

Leider ist dieses Resultat für das Matchingproblem nicht anwendbar. Das Hauptproblem besteht darin, dass die freien Endpunkte von P sich nicht auf einem festen Graphen bewegen. Wenn sich in einem Schritt der RLS ein Endpunkt bewegt, bewegt er sich zwar zufällig entlang einer der angrenzenden (freien) Kanten, er muss dann aber noch die an die gewählte freie Kante angrenzende Matchingkante nehmen. Die Kanten, die in diesem Moment an die freien Kanten grenzen, legen also die Bewegungsmöglichkeiten des Endpunktes fest. Das „Spielfeld“, auf dem sich die Endpunkte bewegen, ist im Laufe der Zeit Veränderungen unterworfen, die z. T. durch die Bewegung der Endpunkte anderer verbessernder Pfade verursacht werden. Selbst wenn wir uns hier auf Bäume beschränken können, macht es dies schwierig, die Resultate von Coppersmith, Tetali und Winkler oder ähnliche Resultate anwendbar zu machen. Im Fall beliebiger zusammenhängender Graphen kann dieser Versuch auch nicht erfolgreich sein, denn wir werden in Kapitel 6 Graphen mit exponentieller erwarteter Optimierzeit kennen lernen.

Wir versuchen nun, das Problem auf andere Weise zu lösen und wenden uns zuerst vollständigen Bäumen zu. Unter einem *vollständigen k-ären Baum* wollen wir hier einen ungerichteten Baum mit ausgezeichnetem Wurzelknoten verstehen, in dem alle Blätter denselben Abstand von der Wurzel haben. Der Grad der Wurzel ist k und alle anderen inneren Knoten haben Grad $k+1$. Offenbar sind Pfadgraphen vollständige unäre Bäume. Im Rest dieses Abschnitts betrachten wir daher nur noch den Fall $k \geq 2$.

Wir rufen uns einige einfache Fakten zu Bäumen in Erinnerung. Dazu stellen wir uns die Knoten eines Baumes auf Ebenen (Levels) angeordnet vor, sodass die oberste Ebene 0 nur von der Wurzel besetzt wird und auf Ebene i alle Knoten mit Abstand i von der Wurzel liegen. Ein Baum der *Tiefe* $d \geq 0$ besitzt Knoten auf den Ebenen 0 bis d. Ein vollständiger k-ärer Baum hat k^d Knoten auf seiner Blattebene d. Für die Zahl seiner Knoten n gilt also $n \geq k^d$. Daraus folgt $d \leq \log_k n$. Der *Durchmesser* D eines Graphen ist das Maximum der Längen aller kürzesten Wege zwischen je zwei Knoten im Graphen. Offenbar ist der Durchmesser eines vollständigen k-ären Baumes gleich dem Zweifachen seiner Tiefe d. Daraus folgt $D = O(\log_k m)$, es gilt sogar $D = \Theta(\log_k m)$.

Als Nächstes zeigen wir, dass es in diesen Bäumen für jedes Matching M mit m^* weniger Kanten als Maximummatchings mindestens einen verbessernden Pfad gibt, der kürzer als $L := 2\log_k(2m/m^*)$ ist.

Lemma 5.8. *Für $k \geq 2$ sei ein vollständiger k-ärer Baum mit m Kanten gegeben. Es sei M^* ein Maximummatching und M ein Matching, das $m^* \geq 1$ Kanten weniger besitzt als M^*. Es gibt einen verbessernden Pfad bez. M, dessen Länge weniger als $L := 2\log_k(2m/m^*)$ beträgt.*

Beweis. Aus Lemma 4.2 wissen wir, dass man m^* knotendisjunkte verbessernde Pfade bez. M auswählen kann. Wir betrachten so eine Auswahl verbessernder Pfade und bezeichnen mit ℓ die (ungerade) Länge eines kürzesten verbessernden Pfades darin. Es sei $d \leq \log_k m$ die Tiefe des Baumes. Jeder (einfache) Pfad kann höchstens zwei Knoten auf jeder Ebene $0, \ldots, d$ des Baumes enthalten. Das bedeutet, dass jeder verbessernde Pfad, dessen ungerade Länge mindestens $\ell < 2d$ beträgt, mindestens einen Knoten auf einer Ebene hat, deren Nummer nicht größer als $d' \leq d - \lceil \ell/2 \rceil$ ist. Also ist die Zahl m^* der knotendisjunkten verbessernden

Pfade durch die Zahl der Knoten auf den Ebenen $0, \ldots, d-\lceil \ell/2 \rceil$ beschränkt, d. h.

$$m^* \leq k^0 + k^1 + \cdots + k^{d-\lceil \ell/2 \rceil} \leq \frac{k^{d-\lceil \ell/2 \rceil+1} - 1}{k-1} < \frac{k}{k-1} k^{d-\lceil \ell/2 \rceil} \leq 2mk^{-\lceil \ell/2 \rceil}.$$

Indem man nun $m^* < 2mk^{-\lceil \ell/2 \rceil}$ nach ℓ auflöst, erhält man die behauptete Schranke $\ell < L$. □

Mit Blick auf Theorem 5.5 über Pfadgraphen besagt das folgende Theorem, dass – zumindest für die RLS – bei vollständigen k-ären Bäumen der Nachteil des höheren Knotengrades für $k \geq 2$ durch den Vorteil logarithmisch längenbeschränkter verbessernder Pfade mehr als aufgewogen wird.

Theorem 5.9. *Unabhängig vom Startpunkt der Suche ist für $k \geq 2$ die erwartete Optimierzeit der RLS für vollständige k-äre Bäume $O(m^{7/2})$.*

Beweis. Aus Lemma 4.1 folgt, dass die erwartete Zeit, bis ein Matching gefunden ist, klein genug ist. Danach wählen wir wie bei der Analyse der Pfadgraphen in jedem Schritt einen kürzesten verbessernden Pfad aus, den wir P nennen. Dann haben relevante Schritte, nämlich Schritte, die P verändern, in Situationen ohne wählbare Kante immer Wahrscheinlichkeit $\Omega(1/m^2)$. In Situationen mit wählbarer Kante nehmen wir für die Definition relevanter Schritte keinen Bezug zu P. Es sind alle Schritte relevant, die entweder das Matching verbessern oder das Matching nicht verbessern und dazu führen, dass die letzte wählbare Kante verschwindet. In dieser Situation haben relevante Schritte sogar Wahrscheinlichkeit $\Omega(1/m)$. Im Folgenden sprechen wir einfach nur von relevanten Schritten, wobei aus der betrachteten Situation heraus klar sein wird, welche Art Schritte damit gemeint sind. Entscheidend ist, dass die erwartete Wartezeit auf den nächsten relevanten Schritt in jeder Situation nur $O(m^2)$ beträgt. Unter Berücksichtigung dieses Laufzeitfaktors genügt es also zu zeigen, dass für die höchstens $O(m)$ erforderlichen Matchingverbesserungen die erwartete Zahl relevanter Schritte insgesamt $O(m^{3/2})$ ist. In der Analyse unterscheiden wir, ob die Güte des aktuellen Matchings schon nahe bei der Güte eines Maximummatchings liegt oder noch weit davon entfernt ist.

Solange der aktuelle Suchpunkt ein Matching repräsentiert, das um mindestens $2\sqrt{m}$ Kanten kleiner ist als Maximummatchings, wissen wir aus Lemma 5.8, dass die Länge von P kleiner als $L := \log_k m$ ist. Da der Grad jedes Knotens höchstens $k+1$ beträgt, folgt aus Lemma 5.7, dass in Situationen ohne wählbare Kante ein relevanter Schritt den Pfad P mit einer Wahrscheinlichkeit von mindestens $p := 1/(k+1)$ verkürzt und sonst um höchstens 2 Kanten verlängert. Wenn wir Verkürzungen pessimistisch durch Verkürzungen um genau 2 Kanten ersetzen, können wir die aktuelle Pfadlänge j durch den Zustand $\lceil j/2 \rceil$ der Markoffkette M aus Lemma 5.2 mit Zustandsmenge $\{0, \ldots, \lceil L/2 \rceil\}$ repräsentieren. In Situationen mit einer wählbaren Kante, d. h., wenn die Pfadlänge 1 ist und die Markoffkette in Zustand 1 ist, folgt aus Lemma 5.6, dass der nächste relevante Schritt mit einer Wahrscheinlichkeit von mindestens 1/2 das Matching verbessert, d. h., P erreicht Länge 0 und die Markoffkette Zustand 0. Andernfalls hat danach der kürzeste verbessernde Pfad Länge 3, was in der Markoffkette dem Zustand 2 entspricht. Wir dürfen daher (pessimistisch) $r := 1/2$ für die Markoffkette aus Lemma 5.2 wählen. Wenn wir (ebenfalls pessimistisch) annehmen, dass P zu Beginn eine Länge von L hat, dann ist die erwartete Zahl relevanter Schritte bis zur nächsten Matchingverbesserung nach Lemma 5.2 höchstens

$$\frac{k+1}{k-1}\left(\frac{k^{\lceil L/2\rceil}-1}{k-1}+k^{\lceil L/2\rceil-1}-(\lceil L/2\rceil+1)\right)+2 \;=\; O(k^{L/2}) \;=\; O(m^{1/2}).$$

Da wir diese Schranke höchstens $O(m)$-mal anwenden, ist nach erwartet $O(m^{3/2})$ relevanten Schritten ein Matching erreicht, das um weniger als $2\sqrt{m}$ Kanten schlechter als Maximummatchings ist.

Für die verbleibenden Matchingverbesserungen verfahren wir wie zuvor, verwenden jedoch für die Länge von P anstatt L die triviale obere Schranke $2\log_k m$ für den Durchmesser des Baumes. Damit erhalten wir eine erwartete Zahl von $O(m)$ relevanten Schritten für eine Matchingverbesserung. In der Summe benötigen die verbleibenden Matchingverbesserungen also ebenfalls nur eine erwartete Zahl von $O(m^{3/2})$ relevanten Schritten. □

5.3 Uneingeschränkte Bäume

Pfadgraphen haben sich für die RLS als die schwierigsten vollständigen k-ären Bäume herausgestellt. Man kann vermuten, dass Pfadgraphen wegen ihrer potentiell $\Omega(m)$ Kanten langen verbessernden Pfade zu den schwierigsten Bäumen für unsere Heuristiken zählen. Die experimentellen Ergebnisse der Projektgruppe 427 für den (1+1)-EA unterstützen diese Vermutung. Die Gruppe hat die Optimierzeit des (1+1)-EA für Pfadgraphen mit der Optimierzeit zufälliger Bäume mit gleich vielen Kanten verglichen. Die Experimente der Gruppe haben hohe Signifikanz für die Hypothese ergeben, dass zufällige Bäume schneller optimiert werden können als Pfadgraphen (siehe Briest, Brockhoff, Degener, Englert, Gunia, Heering, Jansen, Leifhelm, Plociennik, Röglin, Schweer, Sudholt, Tannenbaum und Wegener, 2004a,b). Leider gelingt es noch nicht, für alle Bäume die obere Schranke $O(m^4)$ nachzuweisen. Da die Graphstruktur weniger regelmäßig ist, sind gute obere Schranken hier schwieriger zu zeigen.

Die im Folgenden vorgestellte Schranke von $O(D^2m^4)$ für die RLS hängt auch vom Durchmesser D des Baumes ab. Wir stellen zunächst nur die Beweisidee zu dieser Laufzeitschranke vor.

Beweisübersicht. Wieder betrachten wir nur relevante Schritte, die je nach Situation unterschiedlich definiert sind, jedoch in jedem Fall eine Wahrscheinlichkeit von $\Omega(1/m^2)$ haben. Dies macht in der Laufzeitschranke höchstens einen Faktor von m^2 aus. Einen weiteren Faktor von höchstens m macht die Zahl der erforderlichen Matchingverbesserungen aus. Für die angekündigte Schranke genügt es also zu zeigen, dass die erwartete Zahl relevanter Schritte für eine Matchingverbesserung $O(D^2m)$ ist.

Dazu ist das zentrale Stück, wie schon in den vorhergehenden Beweisen, die Analyse der Zahl relevanter Schritte, bis eine Situation mit einer wählbaren Kante entsteht. Dazu muss man eigentlich die Bewegung *aller* freien Knoten im Auge behalten. Wie zuvor beschränken wir uns darauf, nur *einen* verbessernden Pfad P zu verfolgen, der zu Beginn zwei freie Knoten verbindet. Anders als zuvor gelingt es uns hier nicht unmittelbar, die Länge von P zu betrachten. Wir führen stattdessen ein neues, aber weniger exaktes „Potential“ ein, das, wenn es groß ist, impliziert, dass P kurz ist. Die umgekehrte Implikation gilt jedoch nicht. Im Folgenden nehmen wir an, dass wir im Baum bereits eine Wurzel w gewählt haben. Dann bezeichne r die Wurzel des kleinsten Unterbaumes, der beide Endpunkte von P beinhaltet. P ist dann vollständig in diesem Unterbaum mit Wurzel r enthalten. Es sei $d(r)$ die Tiefe von r, d. h. der Abstand von r zu w. In jedem Unterbaum, der nur zwei Ebenen umfasst, hat der längste Pfad ungerader Länge die Länge 1. Also ist $d(r) \geq D - 1$ eine hinreichende Bedingung dafür, dass eine wählbare Kante existiert, denn $d(r)$ ist durch D beschränkt.

Die Beweisidee ist nun die, dass nach erwartet $O(m)$ relevanten Schritten ein Entscheidungsschritt folgt, der bestimmt, ob $d(r)$ wächst oder sinkt. Diese Wartezeiten auf Entscheidungsschritte machen den Faktor m in der erwarteten Zahl von $O(D^2m)$ relevanten Schritten für eine Matchingverbesserung aus. Die Chancen für das Wachsen von $d(r)$ sind in einem Entscheidungsschritt mindestens so gut wie für das Sinken. Wir zeigen, dass die Beträge, um die $d(r)$ wächst oder sinkt, sogar zum Vorteil von Wachsen ausfallen. Damit gelingt es, das Gambler's-Ruin-Problem erneut als Analysewerkzeug einzusetzen, und zwar für den $d(r)$-Wert. Dieser ist zu Beginn nicht kleiner als 0 und soll $D-1$ erreichen. Dies führt zu dem verbleibenden Faktor von D^2.

Wir bereiten nun den skizzierten Beweis der angekündigten oberen Schranke durch einige Behauptungen vor, die zunächst zu beweisen sind. Das Ziel ist es zu zeigen, dass eine Phase, die $\Theta(D^2m)$ relevante Schritte umfasst, mit einer Wahrscheinlichkeit von $\Omega(1)$ eine wählbare Kante erzeugt.

Zu Beginn einer solchen Phase nehmen wir nichts über den Suchpunkt an, außer, dass er ein Matching repräsentiert, das kein Maximummatching ist und keine wählbare Kante besitzt. Daher sind wir auch frei, zu Beginn jeder neuen Phase beliebig eine Wurzel w und einen verbessernden Pfad P auszuwählen. Im Unterschied zu den vorhergehenden Beweisen wählen wir diesen Pfad aber nur zu Beginn jeder Phase aus. Das heißt, wir dürfen unser Augenmerk nicht mehr beliebig auf einen anderen, kürzeren verbessernden Pfad verlegen. Gedanklich betrachten wir eine Phase als (vorzeitig) beendet, sobald eine wählbare Kante hergestellt ist – unabhängig davon, ob P oder ein anderer verbessernder Pfad dafür verantwortlich ist. An dieser Stelle erinnern wir uns, dass Lemma 5.7 versichert, dass der betrachtete Pfad bis zum Ende der Phase nicht plötzlich verschwinden kann und auch das Matching nicht innerhalb der Phase verbessert werden kann.

Mit u und v bezeichnen wir stets die Knoten, welche die Endpunkte des jeweils aktuellen Pfades P sind, d. h., u und v sind keine fest gewählten Knoten. Ein Schritt heißt u- oder v-relevant, wenn er den Endpunkt u bzw. v bewegt. Offenbar haben u- und v-relevante Schritte Wahrscheinlichkeit $\Omega(1/m^2)$. Für einen beliebigen Knoten r bezeichnet $T(r)$ den Unterbaum mit Wurzelknoten r. Nachdem wir die Wurzel w des Gesamtbaumes fixiert haben, ist $T(r)$ eindeutig bestimmt. Mit der Größe von $T(r)$ (in Zeichen $|T(r)|$) bezeichnen wir die Zahl der Knoten in $T(r)$.

Behauptung 5.10. *Angenommen, es gibt keine wählbare Kante, $r \neq w$ ist der Knoten, an dem der Endpunkt v liegt, und u liegt nicht in $T(r)$.*

Die erwartete Zahl v-relevanter Schritte der RLS, bis eines der folgenden drei Ereignisse eintritt, ist $O(|T(r)|)$. Die Ereignisse sind:

- *v verlässt $T(r)$,*
- *u betritt $T(r)$,*
- *eine wählbare Kante wird erzeugt.*

Beweis. Die Wartezeit auf die Vereinigung der drei Ereignisse kann nur kürzer sein als die Wartezeit auf das erste Ereignis. Wir schätzen die erwartete Zahl v-relevanter Schritte bis zum Eintreten des ersten Ereignisses nach oben ab. Wir nehmen also an, dass die beiden anderen Ereignisse nicht eintreten, d. h., u dringt nicht in den Unterbaum $T(r)$ ein und es entsteht keine wählbare Kante. Dadurch können wir den Blick allein auf die Irrfahrt von v im Unterbaum $T(r)$ richten. Bevor eines der drei Ereignisse eingetreten ist, sind wir in einer Situation, in der Lemma 5.7 gilt.

Solange u außerhalb von $T(r)$ bleibt, kann v nur Knoten des Unterbaumes $T(r)$ besuchen, die eine gerade Distanz von dessen Wurzel r einhalten. Das bedeutet, v besucht nur Ebenen von $T(r)$ mit gerader Nummer. Man kann dies folgendermaßen einsehen. Es ist klar, dass der Pfad P zu jeder Zeit die Wurzel r von $T(r)$ passieren muss, denn in Bäumen gibt es zwischen je zwei Knoten nur *einen* (einfachen) Pfad. Zu Beginn hält sich v bei r auf (siehe Bild 5.7). Die Kante, die r mit seinem Elterknoten verbindet, liegt auf P und ist eine freie Kante. Da jeder akzeptierte Schritt nur *einen* Endpunkt bewegt, kann sich dies nur ändern, wenn v den Unterbaum $T(r)$ verlässt oder u den Unterbaum $T(r)$ betritt. Dann ist das Spiel sofort beendet. Also weist zu jeder Zeit die freie Kante des Pfades P, die mit r

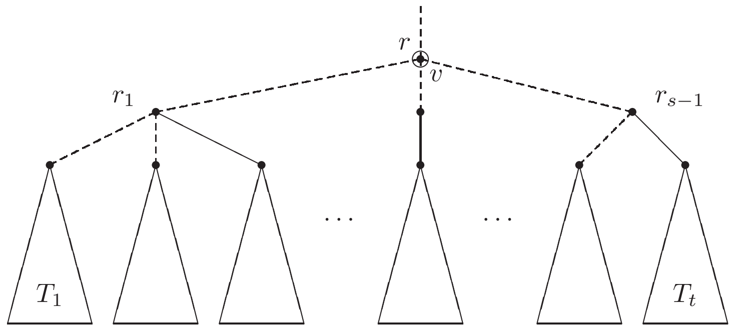

Bild 5.7: Der Unterbaum $T(r)$. Zu Beginn ist v an der Wurzel r von $T(r)$.

inzidiert, zu u. Da P ein verbessernder Pfad ist, endet der Pfadabschnitt zwischen r und u bei u mit einer freien Kante und hat ungerade Länge. Also kann der andere Abschnitt des Pfades zwischen r und v nur gerade Länge haben. Daher hält v stets einen geraden Abstand zu r ein.

Nun können wir die Behauptung per Induktion über die Größe von $T(r)$ zeigen. Falls $|T(r)| = 1$ ist, dann besteht $T(r)$ nur aus einem Blatt, das v im nächsten v-relevanten Schritt nur verlassen kann; die erwartete Zahl v-relevanter Schritte ist also 1.

Nun sei $|T(r)|$ mindestens 2 und s bezeichne den Grad von r. Also gibt es Knoten $r_1, \ldots, r_{s-1}$ in $T(r)$ mit Abstand 1 zu r (siehe Bild 5.7). Immer wenn v bei r ist, also auch zu Beginn, sind die Kanten zwischen r und jedem Knoten r_i freie Kanten; die Kante zwischen r und seinem Elterknoten ist immer frei. Es seien $T_1, \ldots, T_t$ die Unterbäume von $T(r)$, deren Wurzeln Abstand 2 zu r haben. Da es nach Voraussetzung keine wählbare Kante geben darf, ist jeder der Knoten $r_1, \ldots, r_{s-1}$ mit genau einem dieser Unterbäume über eine Matchingkante verbunden. Das bedeutet, wenn v bei r steht und sich nun bewegt, dann sind innerhalb von $T(r)$ nur die Wurzeln von irgend $s-1$ Unterbäumen aus $\{T_1, \ldots, T_t\}$ erreichbar (vgl. Lemma 5.7). Außerhalb von $T(r)$ ist mindestens ein Knoten erreichbar, der auf P liegt. Darüber hinaus kann es Schritte geben, die P um mehr als 2 Kanten verkürzen und v an einen weiter entfernten Knoten auf dem Pfad P verschieben. Das Gleiche gilt, wenn v zuerst einen Unterbaum T_i betritt und diesen irgendwann später verlässt. Beim Verlassen von T_i kann v nur zu r zurückkehren oder gleich zu einem Knoten auf dem Pfad P gehen, der außerhalb von $T(r)$ liegt.

Abweichend von der üblichen Notation bezeichne $E(T)$ im Folgenden die erwartete Zahl v-relevanter Schritte, bis v den Unterbaum T verlässt, wenn v an der Wurzel von T startet. Ohne Beschränkung der Allgemeinheit seien $T_1, \ldots, T_{s-1}$ die $s-1$ größten Unterbäume aus $\{T_1, \ldots, T_t\}$. Nach dem Satz von der totalen Wahrscheinlichkeit gilt

$$E(T(r)) \leq \frac{1}{s} \cdot 1 + \frac{1}{s}\big(1 + E(T_1) + E(T(r))\big) + \cdots + \frac{1}{s}\big(1 + E(T_{s-1}) + E(T(r))\big).$$

Durch Auflösen nach $E(T(r))$ und Anwenden der Induktionsvoraussetzung auf die Erwartungswerte für die Unterbäume von $T(r)$ erhält man

$$E(T(r)) \leq s + |T_1| + \cdots + |T_{s-1}| \leq |T(r)|.$$

Die letzte Ungleichung gilt, da $T(r)$ mindestens alle Unterbäume $T_1, \ldots, T_{s-1}$, die Knoten $r_1, \ldots, r_{s-1}$ und die Wurzel r umfasst. □

Behauptung 5.11. *Angenommen, es gibt keine wählbare Kante, r ist der Knoten, an dem der Endpunkt v liegt, und u liegt in $T(r)$.*

Die erwartete Zahl v-relevanter Schritte der RLS, bis eines der folgenden drei Ereignisse eintritt, ist $O(|T(r)|)$. Die Ereignisse sind:

– *v verlässt $T(r)$ und u bleibt in $T(r)$ zurück,*

– *u und v befinden sich im selben echten Unterbaum von $T(r)$,*

– *eine wählbare Kante wird erzeugt.*

Beweis. Eine entscheidende Beobachtung ist, dass der Abstand zwischen u und r zu jeder Zeit ungerade ist und daher u den Unterbaum $T(r)$ nicht verlassen kann. Zu Beginn und immer wenn sich v an der Wurzel r befindet, ist der Abstand von u zu r gleich $|P|$, also ungerade (siehe Bild 5.8(a)). In dieser Situation kann u den Teilbaum $T(r)$ nicht verlassen, da sich der Pfad am Endpunkt u nur um zwei oder ein Vielfaches von zwei Kanten verkürzen kann. Der Endpunkt u kann sich also lediglich bis auf eine Kante an v annähern, doch damit träte das dritte Ereignis ein. Wenn nun v die Wurzel r verlässt, kann v entweder $T(r)$ verlassen, in denselben Unterbaum gehen, in dem sich schon u befindet, oder in einen anderen Unterbaum gehen. In den ersten beiden Fällen ist das Spiel beendet.

Für den dritten Fall bezeichnen $r_1, \ldots, r_{s-1}$ wie im Beweis zu Behauptung 5.10 die direkten Nachfahren von r im Baum. Dann sind $T(r_1), \ldots, T(r_{s-1})$ die direkten Unterbäume von r. Im Folgenden bezeichnen T_u und T_v die beiden Unterbäume aus $\{T(r_1), \ldots, T(r_{s-1})\}$, in denen sich u bzw. v jeweils aktuell aufhalten (siehe Bild 5.8(b)). Nachdem sich v von der Wurzel r in seinen Unterbaum T_v bewegt hat, durchläuft der Pfad P die Wurzel r. Da sich P am Endpunkt u nur um zwei oder ein Vielfaches von zwei Kanten verlängern und verkürzen kann, bleibt auch jetzt der Abstand von u zu r ungerade. Das heißt auch, dass der Wurzelknoten r für u nicht erreichbar ist. Wenn also u seinen Unterbaum T_u verlassen will, kann u nur in einem verkürzenden Schritt einen Knoten auf dem Pfadabschnitt von P erreichen, der in T_v liegt. Dann tritt das zweite Ereignis ein und das Spiel ist beendet. Das bedeutet, bevor der betrachtete Zeitraum beendet ist, kann u den Unterbaum T_u von $T(r)$, in dem u sich zu Beginn befindet, nicht verlassen. Ferner hält u stets einen ungeraden Abstand zu r ein, v stets einen geraden Abstand.

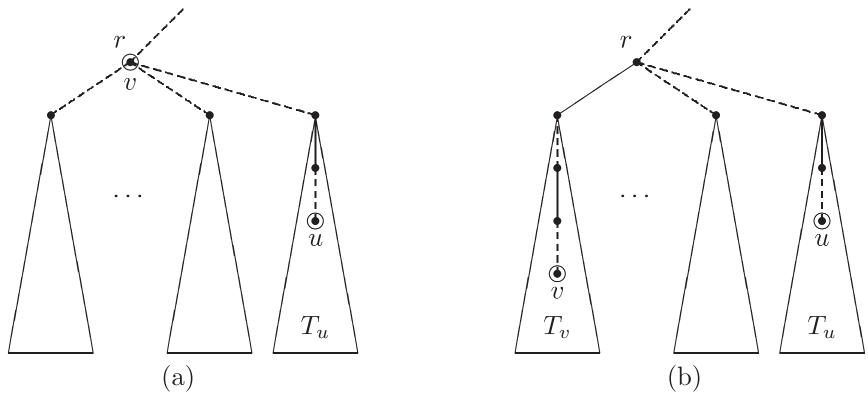

Bild 5.8: (a) Der Unterbaum $T(r)$ in Ausgangssituation und (b) nachdem u sich in einen Unterbaum von $T(r)$ bewegt hat

Wir benutzen nun die Beweisidee von Behauptung 5.10 und Behauptung 5.10 selbst. Dazu betrachten wir zunächst den Fall $r \neq w$. Mit s sei der Grad von r bezeichnet. Wenn v an der Wurzel r steht (Bild 5.8(a)), muss es s Knoten in Abstand 2 geben, die v erreichen kann. Andernfalls gäbe es wählbare Kanten, denn v ist frei, und das Spiel wäre schon beendet. Falls der nächste v-relevante Schritt P um mehr als 2 Kanten verkürzt, endet das Spiel. Wir machen keinen Fehler, wenn wir diese guten Ereignisse in einer oberen Schranke nicht berücksichtigen. Wenn v im nächsten v-relevanten Schritt einen der s Knoten in Abstand 2 besucht, dann ist jeder dieser Knoten gleich wahrscheinlich. Beim Besuch von zwei dieser Knoten tritt das erste oder zweite Ereignis ein. Andernfalls befindet sich v in der Situation, die in Behauptung 5.10 beschrieben ist. Nach einer erwarteten Zahl v-relevanter Schritte, die nicht größer ist als der Unterbaum, an dessen Wurzel v sich nun befindet, verlässt v diesen Unterbaum – oder das zweite oder dritte Ereignis beendet das Spiel zwischenzeitlich. Bei dem Schritt, in dem v seinen Unterbaum verlässt, handelt es sich um einen verkürzenden Schritt. Folglich kann v darin nur zu einem Knoten gelangen, der auf dem Pfad P liegt. Es kommt nur r in Frage oder ein Knoten in T_u.

Nach dem Satz von der totalen Wahrscheinlichkeit gilt

$$E'(T(r)) \;\leq\; \frac{2}{s} \cdot 1 + \frac{1}{s}\big(1 + E(T_1) + E'(T(r))\big) + \cdots + \frac{1}{s}\big(1 + E(T_{s-2}) + E'(T(r))\big).$$

Darin bezeichnen $T_1 \ldots, T_{s-2}$ die $s-2$ größten Unterbäume, deren Wurzeln Abstand 2 zu r haben, $E(T_i)$ die erwartete Zahl v-relevanter Schritte nach Behauptung 5.10 und $E'(T(r))$ die erwartete Zahl v-relevanter Schritte nach Behauptung 5.11. Mithilfe von Behauptung 5.10 können wir nun jedes $E(T_i)$ durch $|T_i|$ abschätzen und erhalten $2 \cdot E'(T(r)) \;\leq\; s + |T_1| + \cdots + |T_{s-2}|$. Daraus folgt auf dieselbe Weise wie im Beweis zu Behauptung 5.11 $E'(T(r)) \leq |T(r)|/2$.

Im Fall $r = w$ ist das erste Ereignis ausgeschlossen. Es gilt die Ungleichung

$$E'(T(r)) \;\leq\; \frac{1}{s} \cdot 1 + \frac{1}{s}\bigl(1 + E(T_1) + E'(T(r))\bigr) + \cdots + \frac{1}{s}\bigl(1 + E(T_{s-1}) + E'(T(r))\bigr),$$

die analog zu der Abschätzung $E'(T(r)) \leq |T(r)|$ führt. □

Im Folgenden bezeichnet r stets die Wurzel des kleinsten Unterbaumes, der u und v einschließt. Das bedeutet, r ist der Knoten, an dem sich der Pfad von u zu der gewählten Wurzel w und der Pfad von v zu w vereinigen. Damit ist klar, dass ganz P in $T(r)$ enthalten ist und r durchläuft. Mit $d(r)$ bezeichnen wir die Tiefe von r gemessen im Gesamtbaum, d. h., $d(r)$ ist die Distanz zwischen r und w oder einfach die Ebene (das Level) von r. Wir sind in der Analyse daran interessiert, dass $d(r)$ wächst.

Die in der Bedingung von Behauptung 5.11 beschriebene und in Bild 5.8(a) dargestellte Situation ist eine besondere Situation, die immer wiederkehrt und für die Analyse eine besondere Rolle spielt: Es gibt keine wählbare Kante, ein Endpunkt von P befindet sich an der Wurzel von $T(r)$, der andere Endpunkt befindet sich in einem der Unterbäume von $T(r)$. Wir geben dieser Situation den Namen *Ausgangssituation*. Wir legen fest, dass der Name des Endpunktes, der in einer Ausgangssituation bei r liegt, v heißt und der Endpunkt mit ungeradem Abstand zu r den Namen u bekommt. Das bedeutet, die Namen von u und v vertauschen sich eventuell, wenn wir eine neue Ausgangssituation betrachten.

In der folgenden Behauptung ist die Idee der „Entscheidungsschritte“ aus der Beweisübersicht formalisiert.

Behauptung 5.12. *Immer wenn der aktuelle Suchpunkt eine Ausgangssituation beschreibt, kann man vier disjunkte Ereignisse $E_1, \ldots, E_4$ definieren, sodass*

- *E_1 das Ereignis ist, dass $d(r)$ sinkt, und zwar um höchstens 2,*
- *E_2 impliziert, dass $d(r)$ um mindestens 2 wächst,*
- *E_3 impliziert, dass $d(r)$ um mindestens 1 wächst, und*
- *E_4 das Ereignis ist, dass eine wählbare Kante erzeugt wird.*

Die erwartete Zahl v-relevanter Schritte der RLS bis zum ersten Eintreten eines dieser Ereignisse ist $O(m)$. Die Wahrscheinlichkeit, dass E_2 zuerst eintritt, ist mindestens so groß wie die Wahrscheinlichkeit, dass E_1 zuerst eintritt. Nachdem eines der Ereignisse $E_1, \ldots, E_3$ eingetreten ist, ist die erwartete Zahl v-relevanter Schritte $O(m)$, bis wieder eine Ausgangssituation vorliegt oder E_4 eintritt; in dieser Zeit sinkt $d(r)$ nicht.

Die Tiefe $d(r)$ kann offenbar nur in Schritten sinken, die P verlängern. Weil Verlängerungen nur um genau zwei Kanten möglich sind (Lemma 5.7), kann $d(r)$ in einem Schritt um höchstens 2 sinken. Deswegen ist das Ereignis E_1 das Ereignis „$d(r)$ sinkt". Der Zusatz „um höchstens 2" ist keine Einschränkung.

Beweis zu Behauptung 5.12. Die Anfangssituation ist eine Ausgangssituation. Zunächst stellen wir fest, dass $T(r)$ mindestens 4 Ebenen umfasst, da P in der Anfangssituation mindestens die Länge 3 haben muss.

Nach Behauptung 5.11 tritt nach einer erwarteten Zahl von $O(m)$ v-relevanten Schritten eines der drei Ereignisse aus Behauptung 5.11 ein. Aus dem Beweis zu Behauptung 5.11 wissen wir, dass nur v den Unterbaum $T(r)$ zuerst verlassen kann (vgl. Bild 5.8(a)). Deswegen ist das erste Ereignis aus Behauptung 5.11 gleich dem Ereignis E_1. Das zweite Ereignis aus Behauptung 5.11 führt zu E_2 oder E_3. Das letzte Ereignis aus Behauptung 5.11 entspricht genau dem Ereignis E_4 und braucht keine weitere Behandlung. Wir kümmern uns im Folgenden nur noch um E_1, E_2 und E_3.

Dazu benutzen wir die Bezeichnungen T_u und T_v in derselben Weise wie im Beweis zu Behauptung 5.11. Wir wissen, dass bis zum Eintreten eines der Ereignisse aus Behauptung 5.11 u stets einen ungeraden Abstand zu r einhält und v stets einen geraden. Der Endpunkt u kann die Wurzel r nicht erreichen und er kann seinen anfänglichen Unterbaum T_u nur verlassen, wenn ein verkürzender relevanter Schritt u zu einem Pfadknoten in den Unterbaum T_v bewegt. In diesem Fall erreicht u mindestens die Wurzel von T_v, die einen Abstand von 1 zu r hat. In diesem Fall wächst $d(r)$ in dem Moment um mindestens 1 und die neue Situation ist wieder eine Ausgangssituation. Damit haben wir das Ereignis E_3 beschrieben.

Andernfalls muss v eines der Ereignisse aus Behauptung 5.11 auslösen. Wir betrachten die letzte Bewegung von v, die dazu führt. Falls sich P in diesem Schritt um mehr als 2 Kanten verkürzt, ist es möglich, dass v zu einem Knoten auf dem Pfad im Unterbaum T_u gelangt und somit $d(r)$ um mindestens 2 erhöht. Auch dann ist die neue Situation eine Ausgangssituation. Diese Möglichkeit führt zu Ereignis E_2. Wir machen jedoch keinen Fehler, wenn wir diese günstige Möglichkeit für die Wahrscheinlichkeit von Ereignis E_2 unberücksichtigt lassen.

Um $T(r)$ verlassen zu können, muss v vor dem letzten Schritt bei r stehen. Nur so kann ein verlängernder Schritt v aus $T(r)$ herausbewegen. Gegeben, dass v bei r steht und der nächste Schritt das Spiel beendet, gibt es jedoch zwei Möglichkeiten. Es gibt mindestens eine 2-Bit-Mutation, die v um mindestens zwei Kanten in Richtung u bewegt, d. h. in den Unterbaum T_u hinein. Danach befinden sich u und v in einer Ausgangssituation und $d(r)$ ist um mindestens 2 gewachsen. (Dasselbe gilt, wenn dieser Schritt um mehr als zwei verkürzt.) Diese Möglichkeit führt zu Ereignis E_2. Die andere Möglichkeit beschreibt das Ereignis E_1, nämlich dass v den Unterbaum $T(r)$ verlässt. Dazu muss die freie Kante kippen, die r mit seinem Vorfahren im Baum verbindet, und die Matchingkante, die mit dem Vorfahren inzidiert.

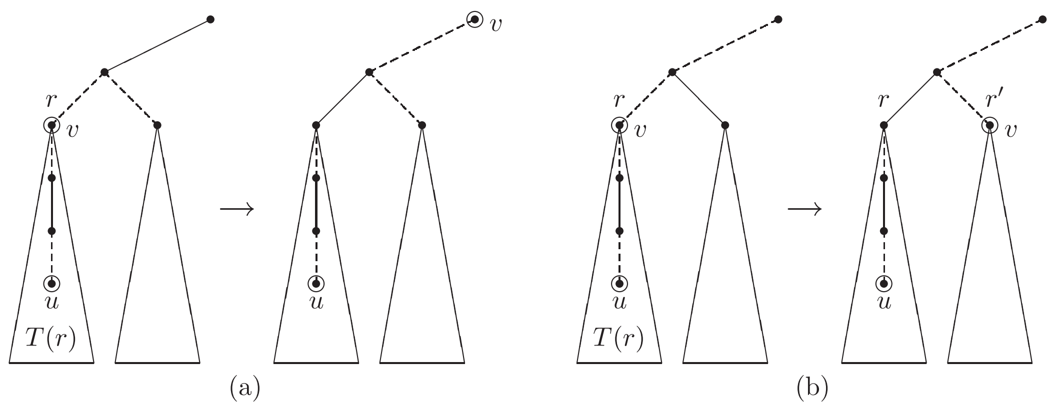

Bild 5.9: Wenn $d(r)$ sinkt, bewegt v sich (a) zum Vorvorfahren von r im Baum oder (b) zu einem Geschwisterknoten

Für dieses Ereignis gibt es genau eine 2-Bit-Mutation. Daher ist die Wahrscheinlichkeit, dass E_1 zuerst eintritt, nicht größer als die Wahrscheinlichkeit, dass E_2 zuerst eintritt.

Falls E_1 eingetreten ist, können danach zwei verschiedene Situationen vorliegen. In der ersten ist v zu dem Vorvorfahren im Baum gewandert (siehe Bild 5.9(a)). Dann ist $d(r)$ um 2 gesunken und es liegt wieder eine Ausgangssituation vor. Andernfalls ist v zu einem Geschwisterknoten r' von r gewandert (siehe Bild 5.9(b)). In diesem Moment ist der gemeinsame Vorgängerknoten von r und r' zur Wurzel des kleinsten Teilbaumes geworden, der u und v enthält, sodass der neue $d(r)$-Wert um 1 gesunken ist. Es liegt jedoch keine Ausgangssituation vor. Gemäß Behauptung 5.10 genügt eine erwartete Zahl von $O(|T(r')|) = O(m)$ v-relevanten Schritten, bis v den Unterbaum $T(r')$ verlassen hat oder u ihm in $T(r')$ gefolgt ist. Sollte sich zu diesem Zeitpunkt u noch immer in $T(r)$ befinden, dann muss v beim Verlassen von $T(r')$ zu einem Knoten auf dem Pfad P gehen. Das ist entweder r oder ein Knoten, der noch tiefer in $T(r)$ liegt. Falls u den Unterbaum $T(r)$ verlässt, bevor v den Unterbaum $T(r')$ verlässt, kann u in dem Moment ebenfalls nur zu einem Knoten wechseln, der auf P liegt. Das kann der gemeinsame Vaterknoten von r und r' sein oder ein Knoten in $T(r')$. In beiden Fällen wird eine neue Ausgangssituation erreicht.

Wir fassen den letzten Absatz zusammen. Nach dem Eintreten von E_1 wird unmittelbar oder nach einer zusätzlichen Wartezeit von erwartet $O(m)$ v-relevanten Schritten eine neue Ausgangssituation erreicht. In dieser zusätzlichen Wartezeit sinkt der $d(v)$-Wert nicht, er kann aber steigen. □

Behauptung 5.13. *Die RLS habe einen Suchpunkt erreicht, der ein Matching beschreibt, das keine wählbare Kante besitzt, jedoch einen verbessernden Pfad P mit Endpunkten u und v. Die erwartete Zahl v-relevanter Schritte, bis eine wählbare Kante hergestellt ist, ist $O(D^2m)$.*

Beweis. Zu Beginn wählen wir einen Knoten des Baums aus, der die Wurzel sein soll. Wir wählen $w = v$. Dadurch ist sichergestellt, dass die Anfangssituation eine Ausgangssituation ist, und es gilt $d(r) = 0$.

Schon in der Beweisübersicht haben wir die Idee skizziert, das Kriterium $d(r) \geq D-1$ als Indikator für das Vorhandensein einer wählbaren Kante zu nutzen. Leider ist Behauptung 5.12 zum letzten Mal für $d(r) = D - 3$ anwendbar. Das liegt daran, dass in einer Ausgangssituation keine wählbare Kante existiert und somit der Unterbaum $T(r)$ mindestens vier Ebenen umfasst. Daher werden wir die ursprüngliche Idee jetzt verfeinern.

Dazu behaupten wir, dass in jeder Situation, in der $T(r)$ höchstens drei (d. h. zwei oder drei) Ebenen umfasst, der verbessernde Pfad P eine wählbare Kante ist oder andernfalls der nächste u- oder v-relevante Schritt eine wählbare Kante erzeugt. Falls $T(r)$ nur zwei Ebenen umfasst, kann P nur noch aus einer wählbaren Kante bestehen, da P ungerade Länge hat. Falls $T(r)$ drei Ebenen umfasst, ist die Pfadlänge von P wegen der Baumeigenschaft durch 4 beschränkt. Weil P ungerade Länge hat, kann dann P maximal Länge 3 haben. Wenn u oder v sich an der Wurzel r aufhalten, dann ist wegen der Baumeigenschaft die Pfadlänge durch 2 beschränkt, sodass P eine wählbare Kante sein muss. Andernfalls halten sich u und v auf der mittleren oder tiefsten Ebene von $T(r)$ auf und P durchläuft die Wurzel r. Daher sind in dieser Situation keine Schritte möglich, die P um wenigstens 2 Kanten an u oder v verlängern. Nach Lemma 5.7 kann der nächste Schritt, der P verändert, P nur um 2 Kanten verkürzen und muss eine wählbare Kante erzeugen.

Also ist die erwartete Zahl v-relevanter Schritte, bis eine wählbare Kante entsteht, höchstens um 1 größer als die erwartete Zahl v-relevanter Schritte, bis $d(r) \geq D-2$ gilt. Da nach Behauptung 5.12 nach einer erwarteten Zahl von $O(m)$ v-relevanten Schritten eines der Ereignisse $E_1, \ldots, E_4$ eintritt, genügt es zu zeigen, dass die erwartete Zahl der Ereignisse E_1, E_2 und E_3 $O(D^2)$ ist, bis $d(r) \geq D - 2$ eintritt.

Wir betrachten nun den stochastischen Prozess, der die Veränderung der Tiefe $d(r)$ in den Schritten der RLS beschreibt, die $d(r)$ tatsächlich verändern. Jedoch „verlangsamen" wir den Prozess zu unseren Ungunsten. Wir nehmen an, dass das Ereignis E_1 den $d(r)$-Wert stets um 2 statt höchstens 2 sinken lässt, E_2 den $d(r)$-Wert stets nur um 2 statt mindestens 2 wachsen lässt und die Wahrscheinlichkeiten, dass E_1 oder E_2 eintreten, jeweils gleich groß sind. Die zuletzt genannten Wahrscheinlichkeiten kennen wir nicht. Es kann sein, dass vornehmlich die Ereignisse E_1 und E_2 eintreten, vornehmlich das Ereignis E_3 oder dass sich alle drei Ereignisse gut mischen. Wir zeigen, dass eine Phase aus Schritten der RLS, die $\lceil D^2/2 \rceil + 2D$ der Ereignisse E_1, E_2 und E_3 beinhaltet, mit einer Wahrscheinlichkeit von mindestens $1/2$ erfolgreich ist. Daraus folgt die behauptete Aussage über den Erwartungswert, denn im Fehlerfall ist die nächste solche Phase mit mindestens derselben Wahrscheinlichkeit erfolgreich, da für jede neue Phase die Wurzel w neu gewählt werden darf.

Angenommen, die Anzahl der E_3-Ereignisse in der Phase ist mindestens $D-2$. Dann erhöhen E_3-Ereignisse den $d(r)$-Wert um $D-2$. Da E_1- und E_2-Ereignisse gleich wahrscheinlich sind, gibt es mit Wahrscheinlichkeit $1/2$ mindestens so viele E_2-Ereignisse wie E_1-Ereignisse, sodass E_1- und E_2-Ereignisse in der Summe den $d(r)$-Wert nicht senken.

Wenn die Anzahl der E_3-Ereignisse weniger als $D-2$ beträgt, gibt es mindestens $D^2/2+D$ E_1- und E_2-Ereignisse in der Phase. In diesem Fall brauchen wir die Hilfe der E_3-Ereignisse nicht und ignorieren diese. Die E_1- und E_2-Ereignisse beschreiben nun eine beinahe symmetrische Irrfahrt, wie wir sie in Lemma 5.2 analysiert haben. Der Startzustand $\ell = \lceil (D-2)/2 \rceil$ steht für $d(r) = 0$, der absorbierende Zustand 0 für $d(r) \geq D-2$. Die erwartete Zahl an E_1- und E_2-Ereignissen, bis $d(r) \geq D-2$ erreicht ist, ist weniger als $D^2/4 + D/2$. Nach der Markoffungleichung ist die doppelte Zahl, nämlich $D^2/2 + D$, mit einer Wahrscheinlichkeit von mindestens $1/2$ erfolgreich. $\square$

Nun haben wir alle Argumente beisammen, um die obere Schranke von $O(D^2m^4)$ zu beweisen. Nach Lemma 4.1 ist die erwartete Zeit, bis ein Matching gefunden wird, klein genug. Das Herstellen einer wählbaren Kante benötigt nach Behauptung 5.13 erwartet $O(D^2m)$ u-relevante Schritte, also erwartet $O(D^2m^3)$ Schritte. Nach Lemma 5.6 fällt nach einer erwarteten Zeit von $O(m)$ die Entscheidung, ob der Versuch gelingt, das Matching zu verbessern. Mit Wahrscheinlichkeit $1/2$ ist der Versuch erfolgreich. Daraus folgt, dass die erwartete Zeit für eine Matchingverbesserung ebenfalls $O(D^2m^3)$ ist. Höchstens m solcher Matchingverbesserungen sind ausreichend, um ein Maximummatching zu konstruieren. Insgesamt haben wir folgendes Theorem bewiesen.

Theorem 5.14. *Unabhängig vom Startpunkt der Suche ist für Bäume mit Durchmesser D die erwartete Optimierzeit der RLS $O(D^2m^4)$.*

6 Graphen mit exponentieller erwarteter Optimierzeit

Es gibt Graphen, die eindeutig bestimmte Maximummatchings besitzen. Wir stellen in diesem Kapitel eine solche Graphklasse vor und zeigen, dass die erwartete Optimierzeit der RLS und des (1+1)-EA für Graphen dieser Graphklasse exponentiell ist, sofern der initiale Suchpunkt nicht das Maximummatching ist. Bei zufälliger Wahl des ersten Suchpunktes oder Wahl des leeren Matchings mit Wahrscheinlichkeit 1 impliziert das Resultat eine exponentielle erwartete Optimierzeit. Hinsichtlich des Startpunktes ist dies das stärkste Resultat für die erwartete Optimierzeit, das überhaupt möglich ist. Bei Wahl des ersten Suchpunktes gemäß einer beliebigen Verteilung kann grundsätzlich nicht ausgeschlossen werden, dass der erste Suchpunkt das Optimum trifft.

Abschnitt 6.1 führt zunächst Beweistechniken für exponentielle untere Schranken ein. Abschnitt 6.2 stellt die Graphen vor, für die in Abschnitt 6.3 exponentielle erwartete Optimierzeit nachgewiesen wird.

6.1 Beweistechniken für exponentielle untere Schranken

In diesem Abschnitt werden zwei zentrale Theoreme eingeführt, die wir für den Nachweis exponentieller unterer Schranken in diesem und dem folgenden Kapitel wiederholt einsetzen werden.

Das Gambler's-Ruin-Problem

Der Hintergrund des Gambler's-Ruin-Problem ist ein zeithomogener, stochastischer Prozess, der in jedem Schritt mit Wahrscheinlichkeit p zu einem Zustand mit einer um 1 größeren Nummer wechselt und mit der Gegenwahrscheinlichkeit zu einem Zustand mit einer um 1 kleineren Nummer geht. Dieses Szenario ist uns schon in Lemma 5.2 begegnet. Dort haben wir uns dafür interessiert, wie lange es dauert, den Zustand mit der kleinsten Nummer zu erreichen. In den Anwendungen sind wir pessimistisch davon ausgegangen, im Zustand mit der größten Nummer zu starten. Beim Gambler's-Ruin-Problem interessiert man sich dafür, mit welcher Wahrscheinlichkeit man zuerst den Zustand mit kleinster Nummer erreicht, bevor man den Zustand mit größter Nummer erreicht, wenn man in einem der übrigen Zustände startet.

Theorem 6.1 (Gambler's-Ruin-Problem). *Ein Kartenspieler spielt gegen einen Gegner und gewinnt jede Runde des Kartenspiels mit Wahrscheinlichkeit $p \in]0,1[$. Gewinnt der Kartenspieler eine Runde, erhält er von seinem Gegner einen Euro, andernfalls muss er seinem Gegner einen Euro zahlen. Das Anfangskapital des Kartenspielers sei $a \in \mathbb{N}_0$ und das Anfangskapital seines Gegners sei $b \in \mathbb{N}_0$. Das Gesamtkapital ist $z := a+b$. Das Kartenspiel wird so lange fortgesetzt, bis einer der Spieler bankrott ist. Derjenige Spieler, der am Ende das gesamte Kapital z besitzt, hat das Kartenspiel gewonnen.*

Für $p \neq 1/2$ und $t := (1-p)/p$ ist die Wahrscheinlichkeit, dass der Spieler das Kartenspiel gewinnt,

$$\frac{1-t^a}{1-t^z}.$$

Das Gambler's-Ruin-Problem ist aus der Literatur gut bekannt. Es wird ausführlich in der Monographie von Feller (1968) besprochen und wird auch in Lehrbüchern wie dem von Schickinger und Steger (2001) behandelt. Der folgende Beweis folgt der Darstellung in Chung (1974).

Beweis zu Theorem 6.1. Für $0 \le a \le z$ sei p_a die Wahrscheinlichkeit, dass der Kartenspieler das Spiel gewinnt, wenn sein Anfangskapital a beträgt. Dann gilt offenbar übereinstimmend mit der Aussage im Theorem $p_0 = 0$ und $p_z = 1$. Für $1 \le a \le z-1$ gilt nach dem Satz von der totalen Wahrscheinlichkeit

$$p_a \;=\; p \cdot p_{a+1} + (1-p) \cdot p_{a-1}.$$

Dies ist äquivalent zu

$$p \cdot (p_{a+1} - p_a) \;=\; (1-p) \cdot (p_a - p_{a-1}).$$

Nun setzen wir $d_a := p_{a+1} - p_a$. Dann ergibt sich aus der letzten Gleichung

$$d_a \;=\; t \cdot d_{a-1}.$$

Das Expandieren der rechten Seite liefert $d_a = t^a \cdot d_0$. Durch Bilden der „Teleskopsumme" aller d_a-Werte erhalten wir

$$1 \;=\; p_z - p_0 \;=\; \sum_{a=0}^{z-1} d_a \;=\; \sum_{a=0}^{z-1} t^a \cdot d_0 \;=\; d_0 \cdot \frac{1-t^z}{1-t}. \qquad (*)$$

Analog verfahren wir für p_a und erhalten

$$p_a \;=\; p_a - p_0 \;=\; \sum_{i=0}^{a-1} d_i \;=\; \sum_{i=0}^{a-1} t^i \cdot d_0 \;=\; d_0 \cdot \frac{1-t^a}{1-t}. \qquad (**)$$

Indem man Gleichung $(**)$ durch Gleichung $(*)$ teilt, erhält man

$$p_a = \frac{1-t^a}{1-t^z},$$

die gesuchte Wahrscheinlichkeit, dass der Spieler das Kartenspiel gewinnt. $\square$

Wir werden Theorem 6.1 in Situationen einsetzen, in denen die Gewinnchancen in jeder Runde zugunsten des Spielers stehen, d. h., es gilt $p > 1/2$. Dann liegt t im Intervall $[0, 1[$ und die Wahrscheinlichkeit, dass der Spieler das Kartenspiel gewinnt, ist $(1 - t^a)/(1 - t^z) \geq 1 - t^a$.

Korollar 6.2. *Für $p > 1/2$ ist beim Gambler's-Ruin-Problem die Wahrscheinlichkeit, dass der Spieler das Kartenspiel gewinnt, mindestens $1 - t^a$.*

Wenn die Wahrscheinlichkeit p nur um mindestens eine kleine Konstante größer als 1/2 ist, d. h. $p = 1/2 + \Omega(1)$, führt dies offenbar zu einer Gewinnwahrscheinlichkeit, die bez. des Parameters a exponentiell nahe an 1 liegt, d. h., sie ist $1 - \mathrm{e}^{-\Omega(a)}$. Daraus folgt, dass die erwartete Zahl unabhängiger Wiederholungen des Kartenspiels, bis der Spieler zum ersten Mal verliert, exponentiell in a ist.

Das Drifttheorem

Mit dem Gambler's-Ruin-Problem können wir stochastische Prozesse erfassen, die in jedem Schritt zu einem von zwei möglichen Nachbarzuständen gehen. Für die Analyse der RLS, die nur kleine, lokale Veränderungen des aktuellen Suchpunktes vornimmt, ist dieses Szenario geeignet. Für den (1+1)-EA ist es jedoch nicht immer ausreichend, da dieser manchmal auch große Veränderungen des Suchpunktes vornimmt, die entscheidend sein können. Dies kommt in unserer bisherigen Sichtweise dem Überspringen von Zuständen gleich.

Die Drifttheorie gestattet es in solchen Situationen, nur die erwartete Veränderung des aktuellen Zustandes (oder einer seiner Eigenschaften) in einem Schritt zu analysieren und so zu Aussagen über die erwartete Zeit bis zum Erreichen eines Zielzustandes (oder eines Zustandes mit bestimmten Eigenschaften) zu kommen. Das folgende Drifttheorem geht auf Hajek (1982) zurück. He und Yao (2001) haben die zugrunde liegenden Beweisideen aufgegriffen und geben Bedingungen an, für die evolutionäre Algorithmen höchstens polynomielle oder mindestens exponentielle erwartete Optimierzeit haben. Das im Folgenden formulierte Drifttheorem ist eine Verschärfung der Formulierung von He und Yao für den exponentiellen Fall, die ursprünglich nur eine untere Schranke für den Erwartungswert angibt. Implizit ist die Aussage des folgenden Theorems aber schon in ihren und Hajeks Beweisen enthalten.

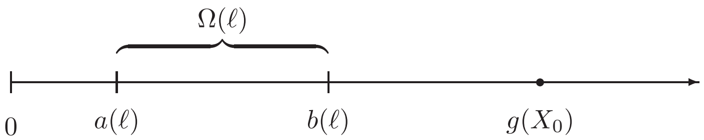

Bild 6.1: Ausgangssituation für das Drifttheorem

Theorem 6.3 (Drifttheorem). *Es seien $X_0, X_1, X_2, \ldots$ Zufallsgrößen, die einen Markoffprozess mit Zustandsmenge S beschreiben, und $g\colon S \to \mathbb{R}_0^+$ sei eine Funktion, die jedem Zustand eine nicht negative reelle Zahl zuordnet. In Abhängigkeit von dem Parameter $\ell \in \mathbb{R}^+$ seien zwei reelle Zahlen $a(\ell)$ und $b(\ell)$ gewählt, sodass $0 \le a(\ell) < b(\ell)$ gilt. Die Zufallsvariable T sei der früheste Zeitpunkt $t \ge 0$, zu dem $g(X_t) \le a(\ell)$ erfüllt ist.*

Für beliebige Konstanten $\lambda > 0$ und $D \ge 1$ und ein beliebiges Polynom $p(\ell)$, das nur positive Werte annimmt, gilt unter den vier Bedingungen

1. $\mathrm{Prob}\big(g(X_0) \ge b(\ell)\big) \;=\; 1,$
2. $b(\ell) - a(\ell) \;=\; \Omega(\ell),$
3. $\forall t \ge 0\colon\; E\Big(\mathrm{e}^{-\lambda(g(X_{t+1})-g(X_t))} \;\Big|\; X_t,\; a(\ell) < g(X_t) < b(\ell)\Big) \;\le\; 1 - 1/p(\ell)$ *und*
4. $\forall t \ge 0\colon\; E\Big(\mathrm{e}^{-\lambda(g(X_{t+1})-b(\ell))} \;\Big|\; X_t,\; b(\ell) \le g(X_t)\Big) \;\le\; D$

für die Zufallsvariable T für alle Zeitschranken $B \ge 0$

$$\mathrm{Prob}(T \le B) \;\le\; \mathrm{e}^{\lambda(a(\ell)-b(\ell))} \cdot B \cdot D \cdot p(\ell).$$

Für $B = \mathrm{e}^{c\cdot\ell}$, wobei die Konstante c genügend klein zu wählen ist, wird diese Schranke exponentiell klein, denn es ist $\lambda(a(\ell) - b(\ell)) = -\Omega(\ell)$ und $D \cdot p(\ell)$ ein Polynom.

Die Bedingungen des Drifttheorems können wie folgt gedeutet werden. Bezüglich des g-Wertes soll das Intervall zwischen $a(\ell)$ und $b(\ell)$ von rechts kommend schwierig zu überwinden sein (siehe Bild 6.1). Die erste Bedingung sichert, dass der Startpunkt nicht links von $b(\ell)$ liegt. Die zweite Bedingung besagt, dass das Intervall mindestens linear im Parameter ℓ wächst. In der dritten Bedingung kommt zum Ausdruck, dass innerhalb des Intervalls eine sehr starke Drift nach rechts weist. Die letzte Bedingung für den Bereich rechts von $b(\ell)$, in dem auch der Startpunkt liegt, ist schwächer. Für $D > 1$ erlaubt sie sogar, dass die Drift dort nach links weist. Da D eine Konstante ist, bleibt es dennoch unwahrscheinlich genug, dass das Intervall in einem Schritt übersprungen werden kann.

Der folgende Beweis ist eine modifizierte und ausgearbeitete Version der Darstellung in He und Yao (2001).

Beweis zu Theorem 6.3. Zur übersichtlicheren Notation sei im Folgenden $a := a(\ell)$, $b := b(\ell)$ und $\rho := 1 - 1/p(\ell)$.

Wir behaupten, dass für jede Realisierung der Zufallsgröße X_0, die den Startzustand angibt und die erste Bedingung $g(X_0) \geq b$ einhält, die Ungleichung

$$E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \bigm| X_0,\ T \geq t\big) \ \leq\ \rho^t \cdot \mathrm{e}^{-\lambda \cdot g(X_0)} + \frac{1-\rho^t}{1-\rho} \cdot D \cdot \mathrm{e}^{-\lambda \cdot b} \tag{$*$}$$

erfüllt ist. Diese Behauptung beweisen wir zunächst per vollständiger Induktion über t. Man kann leicht einsehen, dass sich für $t = 0$ beide Seiten der Ungleichung $(*)$ zu $\mathrm{e}^{-\lambda \cdot g(X_0)}$ vereinfachen lassen. Für den Induktionsschritt muss

$$E\big(\mathrm{e}^{-\lambda \cdot g(X_{t+1})} \bigm| X_0,\ T \geq t+1\big) \ =\ E\big(E(\mathrm{e}^{-\lambda \cdot g(X_{t+1})} \mid X_t,\ T \geq t+1) \bigm| X_0,\ T \geq t+1\big)$$

abgeschätzt werden. Da $T \geq t+1$ gilt, ist $a < g(X_t)$ bekannt. Daher ist der innere Erwartungswert höchstens das Maximum von

$$E\big(\mathrm{e}^{-\lambda \cdot g(X_{t+1})} \bigm| X_t,\ b \leq g(X_t)\big) \ =\ \mathrm{e}^{-\lambda \cdot b} \cdot E\big(\mathrm{e}^{\lambda(b - g(X_{t+1}))} \bigm| X_t,\ b \leq g(X_t)\big) \ \leq\ D \cdot \mathrm{e}^{-\lambda b}$$

und

$$\begin{aligned} &E\big(\mathrm{e}^{-\lambda \cdot g(X_{t+1})} \bigm| X_t,\ a < g(X_t) < b\big) \\ &\qquad = \ \mathrm{e}^{-\lambda \cdot g(X_t)} \cdot E\big(\mathrm{e}^{\lambda(g(X_t) - g(X_{t+1}))} \bigm| X_t,\ a < g(X_t) < b\big) \ \leq\ \rho \cdot \mathrm{e}^{-\lambda \cdot g(X_t)}. \end{aligned}$$

Für die Abschätzungen haben wir die dritte und vierte Bedingung benutzt. Dieses Maximum ist seinerseits durch die Summe der beiden Erwartungswerte nach oben beschränkt. Also erhalten wir

$$E\big(\mathrm{e}^{-\lambda \cdot g(X_{t+1})} \bigm| X_0,\ T \geq t+1\big) \ \leq\ D \cdot \mathrm{e}^{-\lambda \cdot b} + \rho \cdot E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t+1\big). \tag{$**$}$$

Mithilfe von $\mathrm{Prob}(T = t \mid T \geq t) + \mathrm{Prob}(T \geq t+1 \mid T \geq t) = 1$ und

$$E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t+1\big) \ \leq\ \mathrm{e}^{-\lambda \cdot a} \ \leq\ E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T = t\big)$$

kann der Erwartungswert auf der rechten Seite der Ungleichung $(**)$ nun wie folgt abgeschätzt werden:

$$\begin{aligned} &E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t+1\big) \\ &\qquad = \ E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t+1\big) \cdot \mathrm{Prob}(T = t \mid T \geq t) \ + \\ &\qquad\quad\ \ E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t+1\big) \cdot \mathrm{Prob}(T \geq t+1 \mid T \geq t) \\ &\qquad \leq \ E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T = t\big) \cdot \mathrm{Prob}(T = t \mid T \geq t) \ + \\ &\qquad\quad\ \ E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t+1\big) \cdot \mathrm{Prob}(T \geq t+1 \mid T \geq t). \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad = \ E\big(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t\big). \end{aligned}$$

Damit erhalten wir

$$E\left(\mathrm{e}^{-\lambda \cdot g(X_{t+1})} \;\middle|\; X_0,\ T \geq t+1\right) \;\leq\; D \cdot \mathrm{e}^{-\lambda \cdot b} + \rho \cdot E\left(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t\right).$$

Auf den Erwartungswert auf der rechten Seite lässt sich nun die Induktionsannahme anwenden, wodurch wir Ungleichung $(*)$ für den Zeitpunkt $t+1$ erhalten. Damit ist der Induktionsbeweis beendet.

Aus der Markoffungleichung, der Tatsache, dass a, b und λ nicht negativ sind, und dem zuletzt gezeigten Resultat folgt nun

$$\begin{aligned}
\mathrm{Prob}(g(X_t) \leq a \mid X_0,\ T \geq t) &= \mathrm{Prob}\left(\mathrm{e}^{-\lambda \cdot g(X_t)} \geq \mathrm{e}^{-\lambda \cdot a} \;\middle|\; X_0,\ T \geq t\right) \\
&\leq \frac{E\left(\mathrm{e}^{-\lambda \cdot g(X_t)} \mid X_0,\ T \geq t\right)}{\mathrm{e}^{-\lambda \cdot a}} \\
&\leq \rho^t \cdot \mathrm{e}^{\lambda(a-g(X_0))} + \frac{1-\rho^t}{1-\rho} \cdot D \cdot \mathrm{e}^{\lambda(a-b)}.
\end{aligned}$$

Da $D \geq 1$, $b \leq g(X_0)$ und $\lambda > 0$, ist der erste Summand höchstens $\rho^t \cdot D \cdot \mathrm{e}^{\lambda(a-b)}$. Daher gilt

$$\begin{aligned}
\mathrm{Prob}(g(X_t) \leq a \mid X_0,\ T \geq t) &\leq \left(\rho^t + \frac{1-\rho^t}{1-\rho}\right) \cdot D \cdot \mathrm{e}^{\lambda(a-b)} \\
&\leq D \cdot \mathrm{e}^{\lambda(a-b)} \cdot \frac{1}{1-\rho} \\
&= D \cdot \mathrm{e}^{\lambda(a-b)} \cdot p(\ell).
\end{aligned}$$

Nun gilt für alle Realisierungen von X_0, für die nach Voraussetzung $g(X_0) \geq b$ erfüllt ist,

$$\begin{aligned}
\mathrm{Prob}(T \leq B \mid X_0) &= \sum_{1 \leq t \leq B} \mathrm{Prob}(T = t \mid X_0) \\
&= \sum_{1 \leq t \leq B} \mathrm{Prob}(g(X_t) \leq a \text{ und } T \geq t \mid X_0) \\
&\leq \sum_{1 \leq t \leq B} \mathrm{Prob}(g(X_t) \leq a \mid X_0,\ T \geq t) \\
&\leq \sum_{1 \leq t \leq B} D \cdot \mathrm{e}^{\lambda(a-b)} \cdot p(\ell) \\
&= \mathrm{e}^{\lambda(a-b)} \cdot B \cdot D \cdot p(\ell).
\end{aligned}$$

Damit ist der Beweis des Drifttheorems beendet. $\square$

6.2 Ein schwieriger Graph für randomisierte Suchheuristiken

Wie könnten Graphen aussehen, die für randomisierte Suchheuristiken schwierig sind? In Bäumen gehen von einem Endpunkt u eines verbessernden Pfades P höchstens $n-1 \leq m$ Pfade zu anderen Knoten aus, die sich alle bis auf einen als „Irrweg“ herausstellen können, wenn es darum geht, den verbessernden Pfad P zu finden. Wenn wir in einen Graphen möglichst viele Kreise einbauen, in denen sich die von u ausgehenden Pfade immer wieder vereinigen und verzweigen, und die Zahl der Verzweigungsmöglichkeiten groß ist, dann lassen sich mit polynomiell vielen Knoten und Kanten exponentiell viele „Irrwege“ aufbauen. Für eine Situation mit nur $O(1)$ verbessernden Pfaden kann man sich nun vorstellen, dass eine randomisierte Suchheuristik, die in typischen Schritten nur eine zufällige, lokale Änderung an den Enden verbessernder Pfade vornimmt, lange Zeit braucht, um in exponentiell vielen Pfaden die wenigen verbessernden Pfade zu finden. Das ist die Idee, die der im Folgenden vorgestellten Graphklasse zugrunde liegt.

Im Rest diese Kapitels bezeichnen h und ℓ Parameter, die für die Höhe bez. Länge der Graphen in der Graphklasse $G_{h,\ell}$ stehen, die bereits von Sasaki und Hajek (1988) für die Analyse von Simulated Annealing eingeführt wurde. Die Länge $\ell \geq 3$ ist stets ungerade, sodass $\ell = 2\ell' + 1$ gilt. Die Graphen $G_{h,\ell}$ besitzen $n := h(\ell + 1)$ Knoten, die man sich auf einem Raster der Höhe h und Breite $\ell + 1$ angeordnet vorstellen kann. Die Knotenmenge ist dann $V = \{(i, j) \mid 1 \leq i \leq h,\ 0 \leq j \leq \ell\}$. Zwischen den Knoten in Spalte j, j gerade, und Spalte $j + 1$ verlaufen nur „waagerechte“ Kanten, d. h. genau die Kanten zwischen (i, j) und $(i, j + 1)$. Die Knoten in Spalte j, j ungerade, und Spalte $j + 1$ bilden mit den dazwischen liegenden Kanten den vollständigen bipartiten Graphen $K_{h,h}$, d. h., zwischen Spalte j und $j + 1$ existieren alle Kanten mit einem Endpunkt in Spalte j und dem anderen Endpunkt in Spalte $j + 1$. Bild 6.2 zeigt als Beispiel den Graphen $G_{3,11}$ mit Höhe 3 und Länge 11.

Ein Matching heißt *perfekt* für einen gegebenen Graphen, wenn die Kanten des Matchings jeden Knoten des Graphen überdecken. Perfekte Matchings haben also

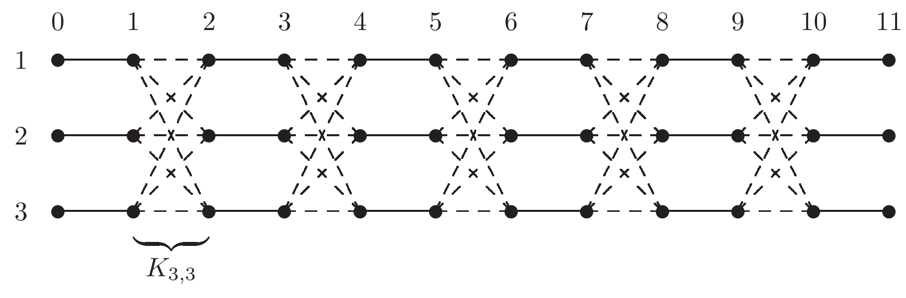

Bild 6.2: Der Graph $G_{3,11}$ mit seinem perfekten Matching

Kardinalität $n/2$ und sind daher Maximummatchings. Jeder Graph $G_{h,\ell}$ besitzt ein eindeutig bestimmtes Maximummatching, und zwar ein perfektes Matching (vgl. Bild 6.2).

Proposition 6.4. *Jeder Graph $G_{h,\ell}$ besitzt ein eindeutig bestimmtes perfektes Matching M^*, das aus den (waagerechten) Kanten zwischen allen Spalten j und $j+1$ für gerade $j \in \{0, \ldots, \ell-1\}$ besteht.*

Beweis. Offensichtlich ist die beschriebene Kantenauswahl ein perfektes Matching der Größe $n/2$, da jeder Knoten von genau einer Matchingkante überdeckt ist. Jedes perfekte Matching muss die Knoten in Spalte 0 überdecken und daher alle Kanten zwischen Spalte 0 und Spalte 1 auswählen. Damit sind alle Kanten zwischen Spalte 1 und Spalte 2 nicht wählbar. Um die Knoten in Spalte 2 zu überdecken, müssen alle Kanten zwischen Spalte 2 und Spalte 3 gewählt werden. Das Argument lässt sich nun induktiv bis zu den Kanten zwischen Spalte $\ell - 1$ und Spalte ℓ fortsetzen, die ebenfalls gewählt werden müssen, weil nur so die Knoten in Spalte ℓ überdeckt werden können. □

Im Folgenden bezeichnen wir die Kanten des perfekten Matchings stets als M^*-Kanten und die übrigen Kanten als die $\overline{M}^*$-Kanten. Für die Zahl der Kanten des Graphen gilt dann

$$m \;=\; |M^*| + |\overline{M}^*| \;=\; (\ell'+1)h + \ell' h^2 \;=\; \Theta(\ell h^2).$$

Proposition 6.5. *Es sei M ein Matching bez. $G_{h,\ell}$.*

– *$M \oplus M^*$ besteht ausschließlich aus $|M^*| - |M|$ knotendisjunkten Pfaden, die in G verbessernde Pfade bez. M sind. Diese heißen im Folgenden* spezielle Pfade *bez. M.*

– *Alle Pfade in $M \oplus M^*$ verlaufen „von links nach rechts“, genauer, jeder Pfad enthält höchstens einen Knoten aus jeder Spalte.*

Beweis. Die erste Behauptung gilt sogar für alle Graphen G mit eindeutigem Maximummatching M^*. Wie im Beweis zu Lemma 4.2 sei $G' = (V, E')$ der Graph mit Kantenmenge $E' := M \oplus M^*$. Der Graph G' besteht im Wesentlichen aus knotendisjunkten Pfaden und Kreisen, die die Zusammenhangskomponenten von G' bilden. Im Beweis zu Lemma 4.2 haben wir gesehen, dass es in G' nur drei Arten von Zusammenhangskomponenten geben kann: Isolierte Knoten, Kreise und Pfade gerader Länge, entlang denen sich M- und M^*-Kanten abwechseln, und ebensolche Pfade ungerader Länge mit M^*-Kanten an den Enden. Da sich die Anzahl der M- und M^*-Kanten in den Komponenten der ersten beiden Typen nicht unterscheidet, ist nur der dritte Typ verbessernd bez. M. Das bedeutet, es muss genau $|M^*| - |M|$ Zusammenhangskomponenten vom dritten Typ geben, die verbessernden Pfaden

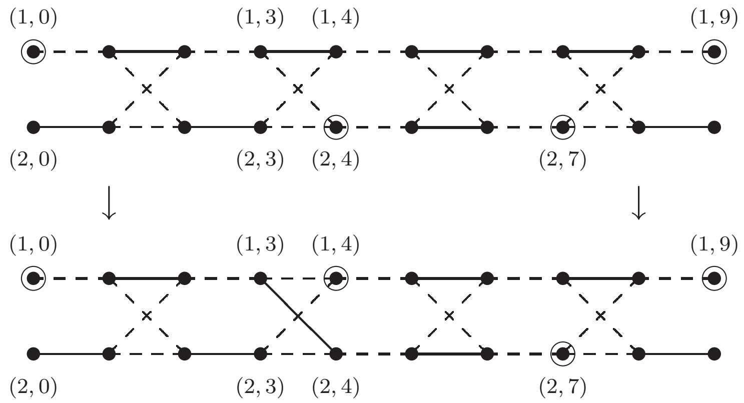

Bild 6.3: Spezielle Pfade in $G_{2,9}$ vor und nach einer 2-Bit-Mutation

bez. M entsprechen. Da es kein weiteres Maximummatching neben M^* gibt, kann es auch keine andere Auswahl von $|M^*| - |M|$ knotendisjunkten verbessernden Pfaden geben. Die $|M^*|-|M|$ verbessernden Pfade bez. M sind also eindeutig bestimmt. Weil M^* eindeutig ist, kann es daher auch keine Zusammenhangskomponenten vom zweiten Typ geben, denn diese erlaubten verschiedene Maximummatchings.

Die zweite Behauptung beweisen wir, indem wir die gegenteilige Annahme zum Widerspruch führen. Angenommen, ein Pfad aus $M \oplus M^*$ enthält zwei Knoten in derselben Spalte j. Da keine Kante zwei Endpunkte in derselben Spalte besitzt, muss es dann zwei benachbarte Kanten auf dem Pfad geben, die beide zwischen denselben Spalten j' und $j'+1$ liegen, d. h., die Kanten besitzen einen gemeinsamen Endpunkt in Spalte j oder $j+1$. Weil sich auf allen Pfaden in $M \oplus M^*$ M- und M^*-Kanten abwechseln, gehört eine der beiden betrachteten Kanten zu M^* und die andere nicht. Dies widerspricht der Tatsache, dass entweder alle oder keine Kante zwischen Spalte j und $j+1$ zu M^* gehört. □

Spezielle Pfade haben eine besondere Bedeutung. Sie markieren die Abschnitte von G, auf denen M von M^* abweicht. Da wir nun an unteren Schranken interessiert sind, ist eine erste Idee zur Analyse, den längsten speziellen Pfad im Auge zu behalten. Dies lässt sich jedoch kaum bewerkstelligen, da schon 2-Bit-Mutationen einen speziellen Pfad in einen verbessernden Pfad umwandeln können, der kein spezieller Pfad mehr ist.

Dazu betrachten wir das Beispiel in Bild 6.3 für den Graphen $G_{2,9}$ und ein Matching M, das nur um zwei Kanten kleiner als das perfekte Matching ist. Also gibt es zwei spezielle Pfade. In der Situation oben im Bild betrachten wir die folgenden verbessernden Pfade.

$(1,0),\ldots,(1,3),(1,4),\ldots,(1,9)$	erster spezieller Pfad
$(2,4),\ldots,(2,7)$	zweiter spezieller Pfad
$(2,4),(1,3),(1,4),\ldots,(1,9)$	verbessernder Pfad, nicht speziell

Neben dem zuletzt genannten, nicht speziellen verbessernden Pfad, gibt es noch weitere dieser Art, die hier nicht aufgeführt sind.

Nach einer 2-Bit-Mutationen, die zwei Kanten zwischen Spalte 3 und Spalte 4 betrifft, entsteht die Situation unten im Bild, in der wir wieder drei Pfade betrachten.

$(1,0),\ldots,(1,3),(2,4),\ldots,(2,7)$	erster spezieller Pfad
$(1,4),\ldots,(1,9)$	zweiter spezieller Pfad
$(1,4),(1,3),(2,4),\ldots,(2,7)$	verbessernder Pfad, nicht speziell

Diese Mutation verlängert den zweiten speziellen Pfad zwischen $(2,4)$ und $(2,7)$ im Bild oben um zwei Kanten, jedoch ist das Ergebnis unten im Bild kein spezieller Pfad mehr.

Daher betrachten wir in den Analysen eine andere Größe, und zwar die Zahl $g = g(M)$ der $\overline{M}^*$-Kanten, die von dem Matching M gewählt werden. Das bedeutet, g gibt die Zahl der Kanten an, die in M „falsch“ gewählt sind und noch durch M^*-Kanten ersetzt werden müssen. Der g-Wert ist eng verknüpft mit der Gesamtlänge aller speziellen Pfade. Wenn wir von wählbaren Kanten absehen, d. h., wenn alle speziellen Pfade mindestens Länge 3 haben, besitzen spezielle Pfade der Länge $2\ell+1$ eine Anzahl von ℓ $\overline{M}^*$-Kanten, die von dem Matching gewählt sind.

Wir fassen diese Größe g als ein Potential des aktuellen Suchpunktes auf. Um zum perfekten Matching zu gelangen, muss das Potential g schließlich den Wert 0 erreichen. Dies ist offensichtlich nur eine notwendige Bedingung, denn z. B. das leere Matching hat ebenfalls Potential 0. Unser Ziel ist es daher zu zeigen, dass das Potential g zu Beginn des Optimierungsprozesses zunächst groß wird und danach ein langer Zeitraum erforderlich ist, um es auf 0 abzusenken.

Mit dem Begriff *zweitbeste Matchings* bezeichnen wir Matchings, die nur um eine Kante kleiner sind als Maximummatchings, in diesem Fall das perfekte Matching. Bei der Untersuchung der erwarteten Optimierzeit für $G_{h,\ell}$ ist das Kernstück der Analyse die Abschätzung der erwarteten Zeit für den letzten Optimierungsschritt, nämlich die Verbesserung eines zweitbesten Matchings. Allein dafür wollen wir eine exponentielle untere Schranke beweisen.

Die Situation eines zweitbesten Matchings erleichtert die Analyse erheblich. In dieser Situation kann es nur einen verbessernden Pfad P geben und dieser ist ein spezieller Pfad. Da wir im Fall $|M^*| - |M| = 1$ sind, folgt dies aus dem ersten Teil von Proposition 6.5. (Weitere verbessernde Pfade, die vielleicht nicht knotendisjunkt zu P sind, kann es allein deshalb nicht geben, weil das perfekte Matching eindeutig bestimmt ist.) Bild 6.4 veranschaulicht diese Situation. Der g-Wert ist für zweitbeste Matchings folglich proportional zur Länge von P, d. h. $|P| = 2 \cdot g + 1$. Offenbar ist g durch $\ell' = (\ell-1)/2$ beschränkt, weil die Länge von P durch ℓ beschränkt ist (vgl. Proposition 6.5).

Anhand des Beispiels können wir intuitiv verstehen, warum der letzte Optimierungsschritt in den Graphen $G_{h,\ell}$ für randomisierte Suchheuristiken schwierig ist.

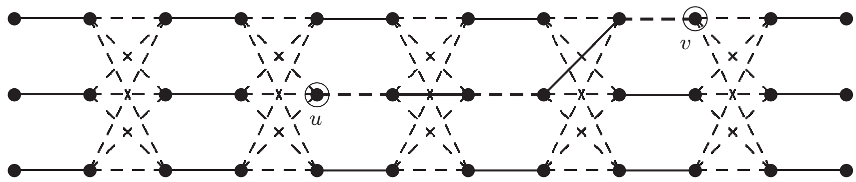

Bild 6.4: Der spezielle Pfad in $G_{3,11}$ bez. eines zweitbesten Matchings

Dazu beschränken wir uns auf 2-Bit-Mutationen, die in dieser Situation nur akzeptiert werden können, wenn sie eine freie Kante und eine benachbarte Matchingkante kippen, wobei die freie Kante einen der beiden Endpunkte u oder v des verbessernden Pfades P überdecken muss. An jedem Endpunkt des verbessernden Pfades P, der nicht in der ersten oder der letzten Spalte liegt, gibt es h Möglichkeiten, P zu verlängern. Solange P sich nicht von der ersten bis zur letzten Spalte erstreckt, d. h., mindestens einer der Endpunkte liegt nicht in diesen Spalten, gibt es mindestens h Verlängerungsmöglichkeiten, denen insgesamt nur 2 Verkürzungsmöglichkeiten entgegenstehen. Bei einer Verlängerung steigt der g-Wert um 1, bei einer Verkürzung sinkt er um 1.

Randomisierte Suchheuristiken können nicht erkennen, welche Möglichkeiten die „richtigen" sind und wählen zufällig. Solange h mindestens 3 ist, beschreiben die Möglichkeiten durch 2-Bit-Mutationen ein unfaires Spiel nach Art des Gambler's-Ruin-Problems. Dabei stellen wir uns den g-Wert als das Kapital des Spielers vor und den maximal möglichen g-Wert von $\ell' = (\ell - 1)/2$ als das Gesamtkapital. Aus unseren Vorüberlegungen zum Gambler's-Ruin-Problem folgt, dass man dieses Spiel nur mit exponentiell kleiner Wahrscheinlichkeit bezüglich des anfänglichen g-Wertes verliert. Daher sind im Erwartungswert exponentiell viele Wiederholungen des Spiels erforderlich, um einen kleinen g-Wert zu erreichen. Das ist die Grundidee des im folgenden Abschnitt vorgestellten Beweises.

6.3 Die Analyse

Beweisübersicht. Wir konzentrieren uns zunächst auf das Kernstück des Beweises und nehmen an, dass die Suche bereits ein zweitbestes Matching erreicht hat. Wir zeigen, dass der verbessernde Pfad sich mit einer Wahrscheinlichkeit von $\Omega(h/m)$ auf seine maximal mögliche Länge ℓ ausdehnt und so einen g-Wert von $\Omega(\ell)$ erzeugt. Danach beweisen wir mithilfe des Gambler's-Ruin-Problems (im Fall des (1+1)-EA mithilfe des Drifttheorems), dass dann die erwartete Zeit, bis der g-Wert auf 1 oder darunter fällt, exponentiell in ℓ ist. Schließlich weisen wir nach, dass die angenommene Situation eines zweitbesten Matchings mit genügend großer Wahrscheinlichkeit $\Omega(1/h)$ erreicht wird, wenn wir mit einem beliebigen Startpunkt beginnen, der nicht das perfekte Matching ist.

Exponentielle untere Schranken bez. der Problemgröße m für die erwartete Optimierzeit erhalten wir auf diese Weise, wenn ℓ polynomiell in m ist. Daher interessieren uns besonders die Graphen $G_{h,\ell}$ mit $2 \leq h \leq \ell$, denn für diese gilt $\ell = \Omega(m^{1/3})$.

Der Fall $h = 2$ erfordert häufig zusätzliche Argumente, da in diesem Fall die Gewinnchance in dem angesprochenen Gambler's-Ruin-Problem nicht mehr grundsätzlich zu Gunsten des Kartenspielers ausfällt. Wir betrachten diesen Fall hier auch nur für die RLS. Der Fall $h = 2$ ist als Grenzfall besonders interessant. Die Graphen $G_{1,\ell}$ sind Pfadgraphen, von denen wir wissen, dass die erwartete Optimierzeit polynomiell ist. Ab $h = 3$ sind die Graphen $G_{h,\ell}$ nicht mehr planar, denn die enthaltenen Teilgraphen $K_{3,3}$ sind nach dem Satz von Kuratowski nicht plättbar. Die Graphen $G_{2,\ell}$ hingegen sind planar und haben einen Knotengrad, der durch die sehr kleine Konstante 3 beschränkt ist. (Zusammenhängende Graphen mit Knotengrad höchstens 2 sind Pfadgraphen.) Wir zeigen also, dass Graphen mit kleinem Knotengrad für die RLS und den (1+1)-EA i. Allg. nicht einfach sind, sondern schwierige Instanzen beinhalten. Für die RLS zeigen wir sogar, dass planare Graphen schwierig sein können, und die Vermutung ist, dass dies auch für den (1+1)-EA gilt.

Lemma 6.6. *Wenn die RLS ein zweitbestes Matching mit g-Wert g_0 erreicht hat, ist die Wahrscheinlichkeit, dass später ein zweitbestes Matching mit dem maximal möglichen g-Wert ℓ' erreicht wird, mindestens*

$$1 - (2/h)^{g_0}, \text{ falls } h \geq 3 \text{ und } g_0 \geq 1, \text{ und}$$
$$1 - (5/6)^{\lfloor g_0/2 \rfloor}, \text{ falls } h = 2 \text{ und } g_0 \geq 2.$$

Beweis. Nachdem ein zweitbestes Matching erreicht ist, werden nur noch zweitbeste Matchings und das perfekte Matching akzeptiert. Solange der g-Wert mindestens 1 ist, hat der verbessernde Pfad mindestens die Länge 3, sodass 1-Bit-Mutationen nicht akzeptiert werden können und 2-Bit-Mutationen nur akzeptiert werden können, wenn sie eine freie Kante e und eine Matchingkante e' kippen. Da es keine wählbaren Kanten gibt, ist jede freie Kante Nachbar einer Matchingkante. Deshalb muss e mit einem der einzigen beiden freien Knoten inzidieren, nämlich den Endpunkten u und v des verbessernden Pfades. In ihrem zweiten Endpunkt muss die Kante e an e' grenzen. Da P ein spezieller Pfad ist, liegt sein linker Endpunkt u stets in einer Spalte mit gerader Nummer und sein rechter Endpunkt v stets in einer Spalte mit ungerader Nummer (vgl. Bild 6.4). Wenn also u nicht in Spalte 0 liegt, gibt es h Möglichkeiten für 2-Bit-Mutationen, um P an u um zwei Kanten zu verlängern, und eine Möglichkeit, um P an u um zwei Kanten zu verkürzen. Dies verändert den g-Wert um 1 bzw. -1. Wenn u in Spalte 0 liegt, entfallen nur die Verlängerungsmöglichkeiten. Gleiches gilt für den anderen Endpunkt v und Spalte ℓ. Immer wenn g nicht seinen maximalen Wert ℓ' hat, liegt mindestens ein Endpunkt u oder v nicht in einer der äußeren Spalten, sodass die Wahrscheinlichkeit einer

Erhöhung des g-Wertes in der nächsten akzeptierten 2-Bit-Mutation mindestens $h/(h+2)$ ist.

Die Wirkung akzeptierter 2-Bit-Mutationen auf den g-Wert kann nun durch das Gambler's-Ruin-Problem modelliert werden. Das Kapital des Kartenspielers ist der g-Wert und das Gesamtkapital ist ℓ'. Ein relevanter Schritt, der den g-Wert um 1 erhöht, entspricht dem Ereignis, dass der Spieler eine Runde des Kartenspiels gewinnt. Obwohl $g = 0$ nur eine notwendige Bedingung für das perfekte Matching ist, deuten wir $g = 0$ zu unseren Ungunsten als das Ereignis, dass das perfekte Matching erzeugt wurde.

Zunächst betrachten wir den Fall $h \geq 3$ und nehmen pessimistisch an, dass die Gewinnwahrscheinlichkeit p des Kartenspielers für eine Runde immer nur $p(h) := h/(h+2) \geq 3/5$ beträgt. Damit wird der Parameter t aus Theorem 6.1 zu $t(h) = 2/h$. Aus Korollar 6.2 folgt, dass die Wahrscheinlichkeit, einen g-Wert von ℓ' zu erreichen, bevor der Wert auf 0 sinkt, mindestens $1 - (2/h)^{g_0}$ beträgt.

Im Fall $h = 2$ gilt es die besondere Situation zu berücksichtigen, dass einer der beiden Endpunkte von P am Rand, d. h. in der ersten oder letzten Spalte, liegt. Wir nennen dies eine *Randsituation*. In einer Randsituation ist die Wahrscheinlichkeit für eine Verlängerung bei der nächsten akzeptierten 2-Bit-Mutation nur $1/2$, sonst ist sie $2/3$. Daher bündeln wir akzeptierte 2-Bit-Mutationen mit der jeweils nächsten akzeptierten 2-Bit-Mutation. Wenn zwei gebündelte 2-Bit-Mutationen in ihrer Wirkung auf den g-Wert gegensätzlich ausfallen, dann heben sich die Wirkungen gerade auf, sodass wir solche Bündel nicht zu berücksichtigen brauchen. Andernfalls ändert sich der g-Wert insgesamt um 2 oder -2.

Unabhängig davon, ob eine Randsituation vorliegt, ist die Wahrscheinlichkeit, dass beide 2-Bit-Mutationen in einem Bündel den g-Wert erhöhen, mindestens $1/4$. Die Wahrscheinlichkeit, dass beide 2-Bit-Mutationen in einem Bündel den g-Wert senken, kann wie folgt nach oben beschränkt werden. Für die erste 2-Bit-Mutation ist die Wahrscheinlichkeit für Absenken höchstens $1/2$. Wenn diese Mutation absenkend war und vorher tatsächlich eine Randsituation vorlag, dann wurde mit Wahrscheinlichkeit $1/2$ der Endpunkt in der äußeren Spalte bewegt, sodass danach keine Randsituation mehr vorliegt. In jedem Fall liegt nach dem ersten absenkenden Schritt nur noch mit einer Wahrscheinlichkeit von höchstens $1/2$ eine Randsituation vor. Daher ist die Wahrscheinlichkeit für ein Bündel aus zwei absenkenden 2-Bit-Mutation höchstens

$$\frac{1}{2} \cdot \left(\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{3}\right) \;=\; \frac{5}{24}.$$

Unter der Bedingung, dass ein Bündel nicht neutral bez. seiner Wirkung auf den g-Wert ist, ist die Wahrscheinlichkeit für eine Erhöhung um 2 mindestens

$$\frac{1/4}{1/4 + 5/24} \;=\; 6/11 \;=:\; p(2).$$

Für die Modellierung mithilfe des Gambler's-Ruin-Problems ist das Startkapital des Spielers nun mindestens $\lfloor g_0/2 \rfloor$ und $t(2) \leq (1-p(2))/p(2)$ ist höchstens 5/6. Die Gewinnwahrscheinlichkeit ist mindestens $1-(5/6)^{\lfloor g_0/2 \rfloor}$. Die bisher noch nicht motivierte Bedingung $g_0 \geq 2$ im Fall $h = 2$ in Lemma 6.6 sichert, dass die Bündelungstechnik ab der Startsituation anwendbar ist, denn dazu ist eine Pfadlänge von mindestens 5 erforderlich. □

Um ein ähnliches Resultat für den (1+1)-EA zu beweisen, müssen wir Mutationsschritte berücksichtigen, in denen viele Bits auf einmal kippen. Im Beweis für die RLS haben wir uns nicht darum gekümmert, wie lange es dauert, bis der g-Wert sein Maximum erreicht. Die Beweisidee für den (1+1)-EA ist, dass der g-Wert mit hoher Wahrscheinlichkeit innerhalb von nur $O(m^3)$ Schritten zum ersten Mal seinen maximal möglichen Wert erreicht. Für diesen kurzen Zeitraum ist es unwahrscheinlich, dass ein Mutationsschritt viele Bits kippt und akzeptiert wird. Dadurch verschlechtert sich das Resultat für den (1+1)-EA gegenüber der RLS nur geringfügig.

Lemma 6.7. *Wenn der (1+1)-EA ein zweitbestes Matching mit g-Wert g_0 erreicht hat, ist die Wahrscheinlichkeit, dass später ein zweitbestes Matching mit dem maximal möglichen g-Wert ℓ' erreicht wird, mindestens*

$$1 - O(1/m) - \big(2/h + O(1/m)\big)^{g_0}, \quad \textit{falls } h \geq 3 \textit{ und } g_0 \geq 1.$$

Beweis. Ein Schritt heißt *relevant*, wenn er akzeptiert wird und den verbessernden Pfad P verändert. Wir teilen die relevanten Schritte in *saubere* und *unsaubere* Schritte ein.

Jede einzelne der relevanten 2-Bit-Mutationen, die wir schon für die RLS betrachtet haben, ist sauber. Alle weiteren relevanten Schritte, die den g-Wert beliebig erhöhen, sind auch sauber. Jedoch ignorieren wir diese, da sie dem Kartenspieler nur helfen würden. Nur in den Situationen mit $g = 1$ gibt es die Möglichkeit, mit einer 3-Bit-Mutation genau die Kanten von P zu kippen. Dieser Schritt ist auch sauber.

Relevante Schritte, die g um mindestens 2 senken, sind unsauber. Sie müssen mindestens die ersten 4 Kanten von P, mindestens die letzten 4 Kanten von P, oder mindestens die zwei ersten und die zwei letzten Kanten von P kippen. Dasselbe gilt für relevante Schritte, die zwar g effektiv nur um 1 senken, aber keine 2-Bit-Mutation sind. (Ausgenommen ist die spezielle 3-Bit-Mutation, die genau die Kanten von P kippt.) Alle diese unsauberen Schritte haben zusammen eine Wahrscheinlichkeit von $O(1/m^4)$.

Die Wahrscheinlichkeit, dass eine Phase aus $O(m^3)$ Schritten einen unsauberen Schritt enthält, ist $O(1/m)$. Für jede Konstante c ist die erwartete Anzahl der 2-Bit-Mutationen, die wir schon für die RLS betrachtet haben, in einer Phase aus cm^3 Schritten $\Theta(m)$. Für jede Konstante $c' < c$ folgt aus Chernoffschranken (siehe Anhang A), dass nur mit exponentiell kleiner Wahrscheinlichkeit $e^{-\Omega(m)}$ weniger als $c'm$ solcher 2-Bit-Mutationen darin enthalten sind. Wir betrachten das Spiel, das durch die 2-Bit-Mutationen beschrieben wird. Die Wahrscheinlichkeit $p(h)$ ist mindestens $1/2 + \Omega(1)$ (siehe den Beweis zu Lemma 6.6). Ebenfalls aus Chernoffschranken folgt für genügend große Konstanten c und c': Die Wahrscheinlichkeit, dass der Kartenspieler in $c'm$ Runden weniger als $\ell' = O(m)$ Runden *mehr* als sein Gegner gewinnt, ist $\mathrm{e}^{-\Omega(m)}$. Damit ist die Wahrscheinlichkeit, dass das Kartenspiel nicht innerhalb der Phase beendet werden kann, durch die Summe der drei „Fehlerwahrscheinlichkeiten“ $O(1/m) + \mathrm{e}^{-\Omega(m)} + \mathrm{e}^{-\Omega(m)} = O(1/m)$ beschränkt und kommt in Lemma 6.7 in dem Summanden „$-O(1/m)$“ zum Ausdruck.

Die Wahrscheinlichkeit, dass der jeweils nächste Schritt ein sauberer Schritt ist, ist $\Theta(h/m^2)$. Nur in Situationen mit einem g-Wert von 1 besteht die Möglichkeit der speziellen 3-Bit-Mutation, die genau die Kanten von P kippt und so g um 1 senkt. Dieser Mutationsschritt hat nur eine Wahrscheinlichkeit von $O(1/m^3)$ und unter der Bedingung, dass ein Schritt sauber ist, ist die bedingte Wahrscheinlichkeit für diese 3-Bit-Mutation $O(1/m)$. Die Wahrscheinlichkeit, dass der g-Wert in einem sauberen Schritt um mindestens 1 wächst, ist also mindestens $p(h) := h/(h+2) - O(1/m)$. Dies führt zu $t(h) = 2/h + O(1/m)$. Die Wahrscheinlichkeit, dass der g-Wert seinen maximalen Wert erreicht, bevor er 0 erreicht, ist $1 - (2/h + O(1/m))^{g_0}$. Die Wahrscheinlichkeit, dass $O(m^3)$ Schritte nicht genügen oder darin unsaubere Schritte auftreten, verringert diese Erfolgswahrscheinlichkeit gemäß unserer Abschätzung im vorhergehenden Absatz insgesamt nur um $O(1/m)$. □

Lemma 6.6 und Lemma 6.7 sind nicht anwendbar, wenn der g-Wert beim Erreichen eines zweitbesten Matchings 0 ist (im Fall $h = 2$ kleiner als 2 ist). Es ist dann nicht unwahrscheinlich, dass der erste relevante Schritt (bzw. das erste Bündel) das Matching verbessert. Daher behandelt Lemma 6.8 diesen Fall gesondert.

Lemma 6.8. *Wenn die RLS oder der (1+1)-EA ein zweitbestes Matching mit einem g-Wert von 0 oder 1 erreicht hat, ist die Wahrscheinlichkeit $\Omega(h/m)$, dass ein zweitbestes Matching mit einem g-Wert von mindestens 2 erreicht wird.*

Beweis. Für zweitbeste Matchings impliziert ein g-Wert von 0, dass der verbessernde Pfad P nur aus einer wählbaren Kante e besteht. Um im nächsten Schritt das Matching zu verbessern, ist es notwendig und hinreichend, nur e zu kippen. Die Wahrscheinlichkeit, dass der nächste Schritt das Matching verbessert, ist daher für die RLS und den (1+1)-EA $\Theta(1/m)$.

Solange der g-Wert 0 ist, kann P nur aus einer wählbaren Kante bestehen, die eine M^*-Kante sein muss. Deswegen liegt mindestens einer der Endpunkte von P nicht in der ersten und nicht in der letzten Spalte. Um g zu erhöhen, muss eines der mindestens h und höchstens $2h$ Kantenpaare kippen, die P verlängern. Daher ist für die RLS die Wahrscheinlichkeit $\Theta(h/m^2)$, im nächsten Schritt eine Situation mit $g \geq 1$ zu erreichen. Die untere Schranke von $\Omega(h/m^2)$ erhält man für den

(1+1)-EA auf dieselbe Weise. Für die obere Schranke muss man noch die Möglichkeit in Betracht ziehen, dass auch e kippt. Damit die Matchinggröße nicht wächst, muss dann anderswo eine Matchingkante e' kippen. Wie zuvor muss dann noch mindestens eines von maximal $2h$ an e' grenzenden Kantenpaaren kippen. Diese zusätzlichen Möglichkeiten haben jedoch insgesamt keine größere Wahrscheinlichkeit als $O(h/m^2)$. Indem man die bedingte Wahrscheinlichkeit für die Erhöhung des g-Wertes betrachtet, erhält man eine Wahrscheinlichkeit von $\Theta(h/m)$, dass ein g-Wert von 1 erreicht wird.

Nun genügt es zu zeigen, dass in einer Situation mit $g = 1$ der nächste Schritt, der den g-Wert ändert, den g-Wert mit einer Wahrscheinlichkeit von $\Omega(1)$ erhöht. Zunächst betrachten wir wieder die RLS. Aus dem Beweis zu Lemma 6.6 wissen wir, dass 1-Bit-Mutationen nicht akzeptiert werden können, genau zwei 2-Bit-Mutationen den g-Wert senken und mindestens h 2-Bit-Mutationen ihn erhöhen. Die bedingte Wahrscheinlichkeit für eine Erhöhung ist also mindestens $h/(h+2) \geq 1/2$. Für den (1+1)-EA gibt es mindestens dieselben 2-Bit-Mutationen, mit denen auch die RLS den g-Wert erhöht. Diese haben zusammen eine Wahrscheinlichkeit von $h \cdot (1/m)^2 \cdot (1-1/m)^{m-2} \geq h/(\mathrm{e}m^2)$. Alle akzeptierten Schritte, die den g-Wert senken, beinhalten das Ereignis, dass die beiden ersten oder die beiden letzten Kanten von P kippen. Dieses Ereignis hat nur eine Wahrscheinlichkeit von $2/m^2$. Die bedingte Wahrscheinlichkeit für eine Erhöhung ist also $\Omega(1)$. □

Wir fassen die bisherigen Ergebnisse aus Lemma 6.6, Lemma 6.7 und Lemma 6.8 in folgendem Korollar zusammen.

Korollar 6.9. *Wenn die RLS (für $h \geq 2$) oder der (1+1)-EA (für $h \geq 3$) ein zweitbestes Matching erreicht hat, ist die Wahrscheinlichkeit $\Omega(h/m)$, dass später ein zweitbestes Matching mit dem maximal möglichen g-Wert ℓ' erreicht wird.*

Das eigentliche Kernstück des Gesamtbeweises für die RLS bildet folgendes Lemma. Der Beweis des Lemmas fällt nun leicht, da im Beweis zu Lemma 6.6 bereits alle wesentlichen Ideen ausgearbeitet sind.

Lemma 6.10. *Wenn die RLS ein zweitbestes Matching mit g-Wert ℓ' erreicht hat, ist für eine geeignet gewählte Konstante $c > 0$ die Wahrscheinlichkeit, dass das perfekte Matching in den nächsten $\lceil \mathrm{e}^{c \cdot \ell} \rceil$ Schritten gefunden wird, durch $\mathrm{e}^{-\Omega(\ell)}$ nach oben beschränkt.*

Beweis. Ein g-Wert von 0 ist eine notwendige Bedingung, um das perfekte Matching zu erreichen. Aufgrund des lokal arbeitenden Suchoperators der RLS muss der Optimierungsprozess alle g-Werte dazwischen auf dem Plateau der zweitbesten Matchings durchlaufen.

Für $h \geq 3$ ist der anfängliche g-Wert gleich ℓ'. Die Entwicklung des g-Wertes kann als Kartenspiel nach Art des Gambler's-Ruin-Problems mit Parameter $t(h) = 2/h$ aufgefasst werden (vgl. dazu den Beweis zu Lemma 6.6). Das Gesamtkapital ist ℓ'.

Wir warten den ersten für den g-Wert relevanten Schritt ab. Dieser kann den g-Wert nur um 1 verringern. Danach beginnt das Kartenspiel. Die Wahrscheinlichkeit, dass der Kartenspieler das Spiel verliert, d. h., der g-Wert erreicht 0, ist nach Korollar 6.2 höchstens $(2/h)^{\ell'-1} \leq \mathrm{e}^{-d \cdot \ell}$ für eine genügend klein gewählte Konstante $d > 0$. Andernfalls gewinnt der Kartenspieler, d. h., der g-Wert erreicht seinen Ausgangswert ℓ', und das nächste Spiel beginnt, sobald wieder der g-Wert $\ell' - 1$ erreicht ist. Für eine beliebige Wahl der Konstanten d' mit $0 < d' < d$ beträgt die Wahrscheinlichkeit, dass der Kartenspieler wenigstens einmal in $\lfloor \mathrm{e}^{d' \cdot \ell} \rfloor$ Spielen Bankrott geht, höchstens $\mathrm{e}^{d' \cdot \ell} \cdot \mathrm{e}^{-d \cdot \ell} = \mathrm{e}^{(d'-d)\ell} = \mathrm{e}^{-\Omega(\ell)}$. Weil jedes dieser Kartenspiele wenigstens einen Schritt des RLS beinhaltet, folgt die Behauptung des Lemmas.

Im Fall $h = 2$ verfahren wir analog. Um wieder die Bündelungstechnik anwenden zu können, betrachten wir den Kartenspieler schon dann als ruiniert, wenn der g-Wert unter 2 fällt. Das Spiel beginnt, sobald das erste nicht neutrale Bündel den g-Wert auf $\ell' - 2$ gesenkt hat. Nach diesen Anpassungen ist das Startkapital des Spielers immer noch von derselben Größenordnung, nämlich $\Omega(\ell)$. Mit entsprechender Wahl der Konstanten d und d' kommen wir so zum gleichen Ergebnis. □

Der Suchoperator des (1+1)-EA erlaubt auch große Veränderungen des aktuellen Suchpunktes. Zwar wird die Wahrscheinlichkeit für einen große Veränderung mit zunehmendem Hammingabstand kleiner, doch in exponentiell langen Zeiträumen können Schritte, die einen großen Hammingabstand überbrücken, nicht mehr mit genügend großer Wahrscheinlichkeit ausgeschlossen werden. Daher trägt die Technik des Gambler's-Ruin-Problems allein nicht mehr für ein ähnliches Resultat wie Lemma 6.10 für den (1+1)-EA.

In dieser Situation greifen wir auf das Drifttheorem zurück, das Bedingungen an die Veränderung des g-Wertes in einem Schritt stellt. Um zu zeigen, dass der g-Wert lange Zeit groß bleibt, genügt es nicht, dass die erwartete Veränderung $E(g_{t+1} - g_t)$ des g-Wertes, vom Zeitpunkt t zum Zeitpunkt $t + 1$, positiv ist. Bei genauer Betrachtung stellt man fest, dass die Driftbedingungen sehr streng sind. Die dritte Bedingung in Theorem 6.3 ist an den Erwartungswert der Zufallsvariablen $\mathrm{e}^{-\lambda(g_{t+1}-g_t)}$ gerichtet, der für ein positives λ kleiner als 1 bleiben muss. Dadurch werden g vergrößernde Schritte und g verkleinernde Schritte nicht mehr gleich behandelt. Schritte, die den g-Wert um j senken, werden nun mit einem exponentiell in j wachsenden Gewicht größer als 1 versehen. Schritte, die den g-Wert erhöhen, erhalten ein exponentiell in j fallendes Gewicht kleiner als 1. Dies führt dazu, dass die Wahrscheinlichkeiten für erhöhende Schritte nun entschieden größer ausfallen müssen als für verkleinernde Schritte.

Lemma 6.11. *Wenn der (1+1)-EA für $h \geq 3$ ein zweitbestes Matching mit g-Wert ℓ' erreicht hat, ist für eine geeignet gewählte Konstante $c > 0$ die Wahrscheinlichkeit, dass das perfekte Matching in den nächsten $\lceil \mathrm{e}^{c \cdot \ell} \rceil$ Schritten gefunden wird, durch $\mathrm{e}^{-\Omega(\ell)}$ nach oben beschränkt.*

Beweis. Der (1+1)-EA beschreibt in dieser Situation einen zeithomogenen Markoffprozess, dessen Zustandsmenge die Menge der Suchpunkte $S \subseteq \{0,1\}^m$ ist, die zweitbeste Matchings oder das Maximummatching sind. Die Funktion $g\colon S \to \mathbb{N}_0$ ordnet jedem dieser Suchpunkte ein Potential zu, den g-Wert. Der Parameter ℓ ist die Länge des Graphen $G_{h,\ell}$. Für Theorem 6.3 wählen wir $b(\ell) := \ell'$ und $a(\ell) = 1$. Damit sind offenbar die ersten beiden Bedingungen des Drifttheorems erfüllt. Das Polynom $p(\ell)$ und die Werte $\lambda > 0$ und $D \geq 1$ werden wir später geeignet wählen.

Für die dritte Bedingung müssen wir für alle Zeitpunkte $t \geq 0$ und für alle Realisierungen $x \in S$ der Zufallsgröße X_t, die den Zustand zum Zeitpunkt t angibt, den Erwartungswert

$$E\left(\mathrm{e}^{-\lambda(g(X_{t+1})-g(X_t))} \;\middle|\; X_t = x,\ 1 < g(X_t) < \ell'\right)$$

untersuchen. Da der Markoffprozess zeithomogen ist, sind sämtliche Übergangswahrscheinlichkeiten von t unabhängig. Der obige Erwartungswert kann daher für einen Suchpunkt x zum Zeitpunkt t mit $2 \leq g(x) < \ell'$ als

$$\sum_{-g(x) \leq j \leq \ell' - g(x)} \mathrm{e}^{-\lambda \cdot j} \cdot p_j(x)$$

geschrieben werden, wobei $p_j(x)$ die Wahrscheinlichkeit für $g(X_{t+1}) - g(X_t) = j$ ist.

Um die letzte Summe nach oben abzuschätzen, ist folgende Beobachtung hilfreich. Wenn wir die Wahrscheinlichkeit $p_j(x)$ oder einen Teil davon auf eine Wahrscheinlich $p_i(x)$ mit $i < j$ umverteilen, dann vergrößert sich die Summe nur. Diese Beobachtung nutzen wir wie folgt. Wir schlagen alle Wahrscheinlichkeiten $p_j(x)$ mit $j \geq 2$ der Wahrscheinlichkeit $p_0(x)$ zu und ersetzen $p_j(x)$ durch die triviale untere Schranke 0. Die Wahrscheinlichkeit $p_1(x)$ ersetzen wir durch eine untere Schranke und schlagen die Differenz zwischen der unteren Schranke und $p_1(x)$ ebenfalls $p_0(x)$ zu. Die Wahrscheinlichkeiten $p_j(x)$ mit $j < 0$ ersetzen wir durch obere Schranken. Effektiv ziehen wir dadurch die Differenz zwischen der oberen Schranke und $p_j(x)$ von der Wahrscheinlichkeit $p_0(x)$ ab und schlagen sie $p_j(x)$ zu.

Unsere Abschätzungen für $p_j(x)$ werden nur für $j \geq g(x)$ von x abhängen, und zwar lediglich von $g(x)$. Daher notieren wir obere und untere Schranken für $p_j(x)$ mit $g(x) = g$ als $p_j(g)$. Für den oben beschriebenen Ersetzungsschritt genügt es nun zu zeigen, dass

$$\begin{aligned} p_j(g) &:= 0 \quad \text{für } j \geq 2 \text{ und} \\ p_1(g) &:= h \cdot (1/m)^2 (1 - 1/m)^{m-2} \end{aligned}$$

untere Schranken für die entsprechenden Wahrscheinlichkeiten $p_j(x)$ mit $g(x) = g$ sind und dass

$$\begin{aligned} p_{-1}(g) &:= 2 \cdot (1/m^2)(1 - 1/m)^{m-2} + 3 \cdot (1/m)^4, \\ p_{-j}(g) &:= (j+1)(1/m)^{2j} \quad \text{für } 1 < j < g \text{ und} \\ p_{-g}(g) &:= (1/m)^{2g+1} \end{aligned}$$

entsprechende obere Schranken sind. Dann ergibt sich als „restliche Wahrscheinlichkeit“

$$p_0(g) \;:=\; 1 - \sum_{j \neq 0} p_j(g) \;>\; 0.$$

Für $p_j(g)$ mit $j \geq 2$ ist nichts zu zeigen. Da es immer mindestens h 2-Bit-Mutationen gibt, die den g-Wert erhöhen, folgt die untere Schranke $p_1(g)$.

Um g um 1 zu senken, gibt es genau zwei Möglichkeiten, die nur 2 Kanten kippen. Da $g \geq 2$ gilt, gibt es keine akzeptierten 3-Bit-Mutationen. In allen weiteren Möglichkeiten müssen für ein $k \in \{0, 1, 2\}$ mindestens die ersten $2k$ Kanten von P und dazu die letzten $4 - 2k$ Kanten von P kippen. Daraus folgt die obere Schranke $p_{-1}(g)$. Um g um j, $1 < j < g$, zu senken, müssen für ein $k \in \{0, \ldots, j\}$ mindestens die ersten $2k$ Kanten von P und dazu die letzten $2(j - k)$ Kanten von P kippen. Daraus ergibt sich die obere Schranke $p_{-j}(g)$ für $1 < j < g$. Schließlich müssen alle $2g + 1$ Kanten des verbessernden Pfades P kippen, um den g-Wert um den Betrag g auf 0 zu senken. Das begründet die obere Schranke $p_{-g}(g)$.

Die Wahrscheinlichkeit $p_0(g)$ steht dafür, dass der Prozess in einem Schritt den g-Wert nicht verändert. Wir eliminieren diese Schritte, indem wir die Übergangswahrscheinlichkeiten durch bedingte Übergangswahrscheinlichkeiten ersetzen, und zwar für die Bedingung, dass sich der g-Wert ändert. Diese bedingten Übergangswahrscheinlichkeiten sind

$$q_0(g) \;:=\; 0 \qquad \text{und} \qquad q_j(g) \;:=\; \frac{p_j(g)}{1 - p_0(g)}.$$

Für den Nenner gilt

$$1 - p_0(g) \;=\; (h + 2) \cdot (1/m^2) \cdot (1 - 1/m)^{m-2} + O(1/m^3) \;=\; \Omega(1/m^2).$$

Die Wahrscheinlichkeit, dass der g-Wert in $\lceil \mathrm{e}^{-\Omega(\ell)} \rceil$ Schritten, die diesen verändern, einmal unter 2 fällt, ist nur größer als in derselben Zahl an beliebigen Schritten. Es genügt also, die dritte Bedingung für diesen „beschleunigten“ Prozess nachzuweisen; die ersten beiden Bedingungen sind offenbar auch für den neuen Prozess gültig. Dazu gilt es nun folgende Summe abzuschätzen.

$$\mathrm{e}^{-\lambda} q_1(g) + \mathrm{e}^{\lambda} q_{-1}(g) + \sum_{2 \leq j \leq g} \mathrm{e}^{j \cdot \lambda} q_{-j}(g). \tag{$*$}$$

Dazu behaupten wir, dass für noch zu wählende Konstanten $\lambda > 0$ und $\delta > 0$

$$\mathrm{e}^{-\lambda} q_1(g) + \mathrm{e}^{\lambda} q_{-1}(g) \;\leq\; 1 - \delta \qquad \text{und} \qquad \sum_{2 \leq j \leq g} \mathrm{e}^{j \cdot \lambda} q_{-j}(g) \;=\; O(1/m)$$

gilt. Daraus folgt, dass die dritte Bedingung des Drifttheorems erfüllt ist. Der „Abstand“ zu 1 auf der rechten Seite der dritten Bedingung ist dann sogar eine kleine

Konstante δ' und nicht nur höchstens polynomiell klein, wie es bereits ausreichend ist.

Wir weisen zuerst die zweite Behauptung nach. Aus der Definition von $q_j(g)$ folgt $q_{-j}(g) = O(j/m^{2j-2})$ für $1 < j < g$ und $q_{-g}(g) = O(1/m^{2g-1})$. Also gilt

$$\sum_{2 \le j \le g} \mathrm{e}^{j \cdot \lambda} q_{-j}(g) \;=\; O\left(\frac{\mathrm{e}^{g \cdot \lambda}}{m^{2g-1}} + \sum_{2 \le j < g} \frac{j \cdot \mathrm{e}^{j \cdot \lambda}}{m^{2j-2}} \right).$$

Für den ersten Summanden gilt

$$\frac{\mathrm{e}^{g\lambda}}{m^{2g-1}} \;=\; \frac{(\mathrm{e}^{\lambda/2})^{2g}}{m^{2g-1}} \;=\; \frac{(\mathrm{e}^{\lambda/2})^2 \cdot (\mathrm{e}^{\lambda/2})^{2g-2}}{m \cdot m^{2g-2}} \;=\; \frac{\mathrm{e}^{\lambda}}{m} \cdot \left(\frac{\mathrm{e}^{\lambda/2}}{m} \right)^{2g-2} \;=\; O(1/m),$$

weil λ eine Konstante ist und $\mathrm{e}^{\lambda/2}/m < 1$ für genügend große m gilt. Für die übrigen Summanden

$$\sum_{2 \le j < g} \frac{j \cdot \mathrm{e}^{j \cdot \lambda}}{m^{2j-2}} \;=\; O\left(\frac{1}{m^2} \cdot \sum_{j \ge 2} \frac{j \cdot \mathrm{e}^{j \cdot \lambda}}{m^{2j-4}} \right)$$

genügt es nun zu zeigen, dass die Summe im letzten O-Term höchstens eine Konstante ist. Die Summanden für $j = 2, \ldots, 4$ liefern offenbar höchstens einen konstanten Beitrag, sodass wir nur noch die gleiche Summe für $j \ge 5$ abzuschätzen brauchen. Für diese gilt

$$\sum_{j \ge 5} \frac{j \cdot \mathrm{e}^{j \cdot \lambda}}{m^{2j-4}} \;=\; \sum_{j \ge 5} \frac{j}{m^{j-4}} \cdot \left(\frac{\mathrm{e}^{\lambda}}{m} \right)^{j} \;=\; O\left(\sum_{j \ge 5} \frac{1}{m^{j-5}} \right) \;=\; O(1).$$

Die erste Behauptung zeigen wir mithilfe des Taylorpolynoms der Exponentialfunktion für den Entwicklungspunkt 0. Wenn wir das Taylorpolynom nach dem Grad 1 abbrechen, ist das Restglied in jedem Fall positiv und wir erhalten für eine Konstante $\alpha > 0$ die Abschätzungen

$$\mathrm{e}^{-\lambda} \;\le\; 1 - \lambda + \alpha\lambda^2 \qquad \text{und} \qquad \mathrm{e}^{\lambda} \;\le\; 1 + \lambda + \alpha\lambda^2.$$

Daraus folgt

$$\begin{aligned} \mathrm{e}^{-\lambda} q_1(g) + \mathrm{e}^{\lambda} q_{-1}(g) \;&\le\; (1 - \lambda + \alpha \cdot \lambda^2) q_1(g) + (1 + \lambda + \alpha \cdot \lambda^2) q_{-1}(g) \\ &\le\; \big(q_1(g) + q_{-1}(g)\big) - \lambda\big(q_1(g) - q_{-1}(g)\big) \\ &\quad\; + \alpha \cdot \lambda^2 \big(q_1(g) + q_{-1}(g)\big) \\ &\le\; 1 - \lambda\big(q_1(g) - q_{-1}(g)\big) + \alpha \cdot \lambda^2. \end{aligned}$$

Für $h \ge 3$ gilt

$$q_1(g) - q_{-1}(g) \;=\; \frac{(h-2) \cdot (1/m^2) \cdot (1 - 1/m)^{m-2} - 3 \cdot (1/m^4)}{(h+2) \cdot (1/m^2) \cdot (1 - 1/m)^{m-2} + O(1/m^3)} \;\ge\; \beta \qquad (**)$$

für eine geeignet gewählte Konstante $\beta > 0$ und m groß genug. Es ist hierbei wichtig, dass die Konstante β so gewählt wird, dass sie für alle $h \geq 3$ geeignet ist und somit unabhängig von h ist. Also genügt es nun zu zeigen, dass

$$1 - \lambda \cdot \beta + \alpha \cdot \lambda^2 \ \leq \ 1 - \delta$$

gilt. Dies ist äquivalent zu

$$\beta \ \geq \ \frac{\delta}{\lambda} + \alpha \cdot \lambda.$$

Wir wählen nun ein λ mit $0 < \lambda \leq \beta/(2\alpha)$ und bestimmen damit $\delta := \beta \cdot \lambda/2 > 0$. Damit ist die Ungleichung für genügend große m erfüllt.

In einer Situation, die der letzten Bedingung des Drifttheorems entspricht, ist der g-Wert für unsere Wahl von $b(\ell) = \ell'$ schon maximal. Daher ist hier $p_j(g) := 0$ für alle $j \geq 1$ die einzig korrekte untere Schranke für die Wahrscheinlichkeit für die Erhöhung des g-Wertes um j. Unsere oberen Schranken $p_{-j}(g)$ bleiben weiterhin gültig. Es ergibt sich nun

$$1 - p_0(g) \ = \ 2 \cdot (1/m)^2 \cdot (1 - 1/m)^{m-2} + O(1/m^3) \ = \ \Omega(1/m^2).$$

Dadurch verändern sich auch die Wahrscheinlichkeiten $q_{-j}(g)$ für den „beschleunigten“ Prozess entsprechend; die Wahrscheinlichkeiten $q_j(g)$ für $j > 0$ sind 0. In der Summe $(*)$ verschwindet daher der erste Summand, sodass nun die Summe

$$\mathrm{e}^{\lambda} q_{-1}(g) + \sum_{2 \leq j \leq g} \mathrm{e}^{j \cdot \lambda} q_{-j}(g)$$

abzuschätzen ist. Darin ist der erste Summand durch die Konstante e^{λ} beschränkt. Die Restsumme kann auf genau dieselbe Weise wie zuvor durch $O(1/m)$ abgeschätzt werden, denn für $q_{-j}(g) = O(j/m^{2j-2})$ mit $1 < j < g$ und $q_{-g}(g) = O(1/m^{2g-1})$ haben wir lediglich $1 - p_0(g) = \Omega(1/m^2)$ benutzt. Für genügend große m ist die Summe daher durch die Konstante $D := \mathrm{e}^{\lambda} + 1$ nach oben beschränkt.

Nun folgt für $B := \lceil \mathrm{e}^{c\ell} \rceil$ aus dem Drifttheorem, dass die Wahrscheinlichkeit, in B Schritten das Optimum zu erreichen, höchstens $\mathrm{e}^{\lambda(a(\ell) - b(\ell))} \cdot B \cdot D \cdot p(\ell)$ ist. Das Polynom $p(\ell)$ ist in unserem Fall sogar eine Konstante. Unabhängig davon ist für ein genügend kleines $c > 0$ diese Wahrscheinlichkeit nur $\mathrm{e}^{-\Omega(\ell)}$, da D und λ Konstanten sind und $b(\ell) - a(\ell) = \Omega(\ell)$ gilt. □

Wir fassen noch einmal unsere Zwischenergebnisse zusammen.

Korollar 6.12. *Wenn die RLS oder der (1+1)-EA ein zweitbestes Matching erreicht hat, ist die Wahrscheinlichkeit höchstens* $1 - \Omega(h/m)$*, dass für eine Konstante* $c \geq 0$ *innerhalb der nächsten* $\lceil \mathrm{e}^{c \cdot \ell} \rceil$ *Schritte das perfekte Matching erreicht wird.*

Wenn der g-Wert beim Erreichen des zweitbesten Matchings g_0 ist, dann ist diese Wahrscheinlichkeit für die RLS durch

$$\begin{aligned} &\mathrm{e}^{-\Omega(\ell)} + (2/h)^{g_0} && \text{für } 3 \le h \le \ell \text{ und } g_0 \ge 1 \text{ sowie} \\ &\mathrm{e}^{-\Omega(\ell)} + (5/6)^{\lfloor g_0/2 \rfloor} && \text{für } h = 2 \text{ und } g_0 \ge 2 \end{aligned}$$

beschränkt und für den (1+1)-EA durch

$$O(1/m) + \big((2/h) + O(1/m)\big)^{g_0} \quad \text{für } 3 \le h \le \ell \text{ und } g_0 \ge 1.$$

Wir kommen nun zu der Frage, mit welcher Wahrscheinlichkeit der Optimierungsprozess der RLS und des (1+1)-EA zweitbeste Matchings erzeugen.

Lemma 6.13. *Wenn der anfängliche Suchpunkt des (1+1)-EA oder der RLS nicht das perfekte Matching ist, wird mit einer Wahrscheinlichkeit von $\Omega(1/h)$ ein zweitbestes Matching erzeugt, bevor das perfekte Matching erzeugt wird.*

Beweis. Es sei M die Kantenauswahl, die durch den aktuellen Suchpunkt beschrieben wird, und es sei $d := |M \oplus M^*|$ der Hammingabstand zum perfekten Matching M^*. Wir untersuchen nur Situationen, in denen M weder ein zweitbestes Matching noch das perfekte Matching ist. Dies schließt alle Situationen mit ein, in denen M (noch) kein Matching ist. In all diesen Situationen wird offenbar jeder Mutationsschritt akzeptiert, der ein zweitbestes Matching oder das perfekte Matching erzeugt.

Der (1+1)-EA erzeugt M^* im nächsten Schritt mit einer Wahrscheinlichkeit von $\Theta(1/m^d)$. Es genügt nun zu zeigen, dass diese Wahrscheinlichkeit höchstens um einen Faktor von $O(h)$ größer ausfällt als die Wahrscheinlichkeit, im nächsten Schritt ein zweitbestes Matching zu erzeugen. Falls $M \oplus M^*$ wenigstens eine M^*-Kante enthält, fehlt diese Kante in M. Da M kein zweitbestes Matching ist, gibt es in $M \oplus M^*$ noch weitere Kanten. Der Mutationsschritt, der alle Kanten aus $M \oplus M^*$ bis auf die ausgesuchte M^*-Kante kippt, erzeugt ein zweitbestes Matching. In diesem Fall ist seine Wahrscheinlichkeit, nämlich $\Theta(1/m^{d-1})$, sogar größer als die des Schrittes, der M^* erzeugt. Andernfalls enthält $M \oplus M^*$ keine M^*-Kanten, weil alle M^*-Kanten bereits zu M gehören. Es gibt dann $|M^*|$ Möglichkeiten, ein zweitbestes Matching zu erzeugen, indem man alle Kanten aus $M \oplus M^*$ kippt und zusätzlich eine M^*-Kante. Die Wahrscheinlichkeit dieser Schritte ist zusammen $\Theta(|M^*|/m^{d+1}) = \Theta(1/(hm^d))$.

Für die RLS ist es notwendig, dass $d \le 2$ gilt, um im nächsten Schritt M^* zu erreichen. Für Situationen mit $d = 1$ argumentieren wir analog zum Fall des (1+1)-EA. Für $d = 2$ zeigen wir, dass die Wahrscheinlichkeit, M^* zu erzeugen, um höchstens einen Faktor von $O(h)$ größer ist als die Wahrscheinlichkeit, ein zweitbestes Matching zu erzeugen, wenn man die nächsten *zwei* Schritte betrachtet.

Im Fall $d = 1$ gibt es nur den Fall zu untersuchen, dass M eine Obermenge von M^* ist, denn andernfalls wäre M bereits ein zweitbestes Matching. Es sei also

$M = M^* \cup \{e\}$, woraus folgt, dass e eine $\overline{M}^*$-Kante ist. Der nächste Schritt erzeugt M^* mit einer Wahrscheinlichkeit von $\Theta(1/m)$. Wenn jedoch e und noch eine weitere Kante von M kippen, entsteht ein zweitbestes Matching. Das passiert im nächsten Schritt mit einer Wahrscheinlichkeit von $\Theta(|M^*|/m^2) = \Theta(1/(hm))$.

Im Fall $d = 2$ muss jede der zwei Kanten in $M \oplus M^*$ mindestens einmal in den nächsten zwei Schritten kippen, damit M^* erzeugt wird. Die Wahrscheinlichkeit ist $\Theta(1/m^2)$. Wenn $M \oplus M^*$ zwei M^*-Kanten enthält, sind beide wählbar. Der erste Schritt erzeugt mit Wahrscheinlichkeit $\Theta(1/m)$ ein zweitbestes Matching, indem er nur eine der beiden Kanten kippt. Wenn $M \oplus M^*$ eine M^*-Kante und eine $\overline{M}^*$-Kante enthält, entfernt der erste Schritt mit Wahrscheinlichkeit $\Theta(1/m)$ die $\overline{M}^*$-Kante aus M, wobei ein zweitbestes Matching entsteht. Wenn $M \oplus M^*$ zwei $\overline{M}^*$-Kanten enthält, gilt $M = M^* \cup \{e_1, e_2\}$ für zwei $\overline{M}^*$-Kanten e_1 und e_2. Jeder erste Schritt, der e_1 und dazu eine beliebige M^*-Kante kippt, wird akzeptiert, da die Summe der Knotenstrafen um mindestens $r \geq m+1$ sinkt. Wenn danach der zweite Schritt e_2 kippt, ist ein zweitbestes Matching erreicht. Die Wahrscheinlichkeit dieser Ereignisfolge ist $\Theta((|M^*|/m^2) \cdot (1/m)) = \Theta(1/(hm^2))$. □

Sasaki und Hajek (1988) haben Simulated Annealing für eine ähnliche Modellierung des Matchingproblems für $G_{h,\ell}$ untersucht. Der Zustandsraum des untersuchten stochastischen Prozesses umfasst nur Matchings und die Suche beginnt mit dem leeren Matching. Sie zeigen, dass die erwartete Optimierzeit exponentiell ist. Ihr Beweis beruht ebenfalls auf einer Variante des Drifttheorems. Das dort betrachte Potential hat Ähnlichkeit mit dem hier benutzten g-Wert. Sasaki (1991) hat den Metropolisalgorithmus für denselben Graphen analysiert, wobei der Zustandsraum des Prozesses ebenfalls nur Matchings enthält. Sasaki zeigt, dass es einen Startzustand gibt, von dem aus die erwartete Optimierzeit exponentiell ist. Seine Analysemethode benennt diesen Startzustand nicht unmittelbar, sodass es sich um eine Existenzaussage handelt.

Wir sind nun in der Lage, für den (1+1)-EA und die RLS bei weitem stärkere Aussagen zu treffen, nämlich dass die erwartete Optimierzeit für jeden Startpunkt exponentiell ist – ausgenommen ist selbstverständlich das Optimum selbst. Wenn die Suche nicht mit dem perfekten Matching beginnt, wird gemäß Lemma 6.13 vor dem Erreichen des perfekten Matchings mit Wahrscheinlichkeit $\Omega(1/h)$ ein zweitbestes Matching besucht. Daher gilt zusammen mit dem ersten Teil von Korollar 6.12, dass die Schranke von $e^{c\ell}$ Schritten mit einer Wahrscheinlichkeit von $\Omega(1/h) \cdot \Omega(h/m) = \Omega(1/m)$ nicht unterschritten wird. Aus der Definition des Erwartungswertes folgt nun unmittelbar, dass die erwartete Optimierzeit für alle Startpunkte bis auf das perfekte Matching $e^{\Omega(\ell)}$ ist. Wenn wir uns auf Instanzen von $G_{h,\ell}$ mit $2 \leq h \leq \ell$ einschränken, folgt aus $m = \Theta(h^2\ell)$ die Eigenschaft $\ell = \Omega(m^{1/3})$. So erhalten wir die in der Problemdimension m exponentielle untere Schranke von $e^{\Omega(m^{1/3})}$ Schritten. Für konstante Höhe h ergibt sich sogar $e^{\Omega(m)}$. Das folgende Theorem fasst das Hauptergebnis diese Kapitels zusammen.

Theorem 6.14. *Für die Graphen $G_{h,\ell}$ ist für jeden Startpunkt außer dem perfekten Matching die erwartete Optimierzeit der RLS für $2 \leq h \leq \ell$ und des (1+1)-EA für $3 \leq h \leq \ell$ durch $\mathrm{e}^{\Omega(\ell)} = \mathrm{e}^{\Omega(m^{1/3})}$ von unten beschränkt. Es gibt eine Konstante $c > 0$, sodass die Optimierzeit mit einer Wahrscheinlichkeit von $\Omega(1/m)$ mindestens $\mathrm{e}^{c \cdot \ell}$ ist.*

Für alle typischen Verteilungen für den Startpunkt der Suche ist die Wahrscheinlichkeit, nicht das Optimum zu treffen, überwältigend. Bei Wahl des leeren Matchings ist sie 1, bei gleichverteilter Auswahl eines zufälligen Startpunktes ist sie $1 - 2^{-m}$. Sogar bei gleichverteilter Auswahl eines zufälligen Matchings ist sie $1 - 2^{-\Omega(h\ell)}$, weil mindestens alle Teilmengen des perfekten Matchings Matchings sind. Für diese Verteilungen für den Startpunkt gilt die Aussage des Theorem 6.14 also auch, wenn der Zusatz „für jeden Startpunkt außer dem perfekten Matching“ gestrichen wird.

7 Exponentielle Optimierzeit mit überwältigender Wahrscheinlichkeit

In diesem Kapitel zeigen wir, dass die Optimierzeit der RLS und des (1+1)-EA für einige Instanzen der in Kapitel 6 untersuchten Graphen $G_{h,\ell}$ sogar mit überwältigender Wahrscheinlichkeit exponentiell ist. Zuvor motivieren wir, warum wir an diesen Verschärfungen der Ergebnisse aus Kapitel 6 interessiert sind.

7.1 Motivation

Im Allgemeinen werden randomisierte Suchheuristiken als effizient für ein Problem angesehen, wenn ihre erwartete Optimierzeit $E(T)$ polynomiell beschränkt ist. Nach der Markoffungleichung ist dann die Wahrscheinlichkeit, dass die Optimierzeit mindestens $k \cdot E(T)$ ist, höchstens $1/k$.

Man darf daraus jedoch nicht den Schluss ziehen, dass eine randomisierte Suchheuristik für ein Problem versagt, wenn ihre erwartete Optimierzeit erwiesenermaßen exponentiell ist. Die Wahrscheinlichkeit für exponentielle Optimierzeit kann so gering sein, dass sie praktisch unbedeutend wird. In Kapitel 6 konnten wir lediglich nachweisen, dass diese Wahrscheinlichkeit $\Omega(1/m)$ ist (vgl. Theorem 6.14). Zudem ist man nur selten auf einen einzigen Lauf der Suchheuristik angewiesen, sondern führt sequentielle oder parallele unabhängige Läufe durch, die nach einer festgelegten Zeit abgebrochen werden. Man nennt diese Abwandlungen *Multistartvarianten* der ursprünglichen randomisierten Suchheuristik.

Im Folgenden bezeichnet die Zufallsvariable T die Optimierzeit einer randomisierten Suchheuristik für ein Optimierungsproblem. Wenn

$$\mathrm{Prob}\bigl(T \leq p(m)\bigr) \;\geq\; 1/q(m) \qquad (*)$$

für Polynome $p(m)$ und $q(m)$ gilt, ist es immer noch möglich, dass $E(T)$ exponentiell in der Problemgröße m ist. Falls die beiden Polynome bekannt sind, kann ein Lauf nach $p(m)$ Schritten gestoppt werden. Dann ist mindestens einer aus $q(m)^2$ unabhängigen Läufen mit einer überwältigenden Wahrscheinlichkeit von mindestens

$$1 - \bigl(1 - 1/q(m)\bigr)^{q(m)^2} \;\geq\; 1 - \mathrm{e}^{-q(m)}$$

erfolgreich. In dieser einfachen Multistartvariante ist die von allen Läufen zusammen benötigte Zeit nur $p(m) \cdot q(m)^2$.

Selbst wenn $p(m)$ und $q(m)$ nicht bekannt sind, aber existieren, können Multistartvarianten erfolgreich sein. Dazu betrachten wir folgende Multistartvariante, die in Phasen arbeitet (vgl. Wegener, 2005).

Multistartvariante. Die Parameter $r_0 \geq 1$ und $t_0 \geq 1$ seien polynomiell in der Problemgröße m gewählt, z. B. $t_0 = r_0 = m$.

In Phase $i \geq 0$ werden (sequentiell) $r_i = 2^i \cdot r_0$ unabhängige Läufe der randomisierten Suchheuristik durchgeführt und jeder dieser Läufe nach $t_i = 2^i \cdot t_0$ Schritten abgebrochen.

Wir analysieren diese Multistartvariante für den Fall, dass Ungleichung $(*)$ für zwei (unbekannte) Polynome $p(m)$ und $q(m)$ erfüllt ist. Es sei

$$u(m) \ := \ \max\{\lceil p(m)/t_0 \rceil, \lceil q(m)/r_0 \rceil\}.$$

Die Laufzeit der Phase i ist $r_i \cdot t_i = 4^i \cdot r_0 t_0$. Die Laufzeit der frühen Phasen $0, \ldots, k := \lceil \log u(m) \rceil = O(\log m)$, in denen die Läufe eventuell weniger als $p(m)$ Schritte umfassen, ist insgesamt $r_0 t_0 \cdot (4^0 + \cdots + 4^k) = O(u(m)^2 \cdot r_0 t_0)$. Wir nehmen pessimistisch an, dass alle diese frühen Phasen erfolglos bleiben.

In einer Phase $i > k$, gibt es $2^i \cdot r_0 \geq 2^{i-k} \cdot u(m) r_0 \geq 2^{i-k} \cdot q(m)$ Läufe mit je mindestens $p(m)$ Schritten. Die Wahrscheinlichkeit, dass die Phase erfolglos ist, ist höchstens

$$\bigl(1 - 1/q(m)\bigr)^{q(m) \cdot 2^{i-k}} \ = \ \mathrm{e}^{-\Omega(2^{i-k})}.$$

Die Gesamtlaufzeit der Phase i ist

$$2^i \cdot r_0 \cdot 2^i \cdot t_0 \ = \ O\bigl(2^{i-k} \cdot u(m) r_0 \cdot 2^{i-k} \cdot u(m) t_0\bigr) \ = \ O(4^{i-k} \cdot u(m)^2 r_0 t_0).$$

Die Wahrscheinlichkeit, dass spätestens die $2k$-te Phase erfolgreich ist, ist überwältigend, nämlich mindestens

$$1 - \mathrm{e}^{-\Omega(2^{2k-k})} \ = \ 1 - \mathrm{e}^{-\Omega(u(m))} \ = \ 1 - \mathrm{e}^{-\Omega(m^{\varepsilon})}$$

für eine Konstante $\varepsilon > 0$, die von $p(m)$, $q(m)$, t_0 und r_0 abhängt. Die Gesamtlaufzeit der ersten $2k$ Phasen ist offenbar ebenfalls durch ein Polynom beschränkt. Darüber hinaus ist die erwartete Optimierzeit der Multistartvariante

$$O\bigl(u(m)^2 r_0 t_0\bigr) \cdot \Bigl(1 + \sum_{1 \leq j < \infty} 4^j \cdot \mathrm{e}^{-\Omega(2^j)}\Bigr) \ = \ O\bigl(u(m)^2 r_0 t_0\bigr),$$

d. h., sie ist ebenfalls polynomiell.

Vor diesem Hintergrund können wir eine randomisierte Suchheuristik erst dann *erfolglos* nennen, wenn für jedes Polynom $p(m)$ und jede Konstante k ihre Erfolgswahrscheinlichkeit für $p(m)$ Schritte $o(m^{-k})$ ist. Dann ist die Erfolgswahrscheinlichkeit jeder Multistartvariante, in der sich polynomiell viele Schritte beliebig auf verschiedene Läufe verteilen dürfen, superpolynomiell klein.

Um auszuschließen, dass Multistartvarianten der RLS und des (1+1)-EA für das Matchingproblem erfolgreich sein können, wollen wir nun zeigen, dass diese randomisierten Suchheuristiken in diesem strengen Sinne für einige Graphen $G_{h,\ell}$ erfolglos sind. Dazu zeigen wir, dass die Erfolgswahrscheinlichkeit sogar für exponentielle Zeit exponentiell klein sein kann.

7.2 Die Analyse

Für das stärkere Resultat benötigen wir einen nicht zu kleinen Knotengrad. Daher betrachten wir hier nur Graphen $G_{h,\ell}$ mit $h = \omega(\log m)$. Darüber hinaus soll $h \leq \ell - 2$ gelten. Unsere Schranken werden typischerweise exponentiell in h sein. Da h z. B. in derselben Größenordnung wie ℓ gewählt werden kann, beinhaltet dies wegen $m = \Theta(h^2\ell)$ exponentielle Schranken in m.

Im Gegensatz zu Kapitel 6 konzentrieren wir uns nicht mehr allein auf das Plateau der zweitbesten Matchings, sondern beziehen nun auch die früheren Phasen des Optimierungprozesses mit ein, in denen kleinere Matchings vorliegen. Da kleinere Matchings weniger Struktur aufweisen als zweitbeste Matchings, wird es nun schwieriger, die Wahrscheinlichkeiten abzuschätzen, mit denen der g-Wert steigt und fällt. Hier kommt uns nun der Knotengrad von $\omega(\log m)$ zu Hilfe.

Beweisübersicht. Wir wenden uns zunächst wieder dem Kernstück des Beweises zu und nehmen an, dass ein Matching mit einem g-Wert von mindestens $h/2$ vorliegt. Zum Schluss werden wir zeigen, dass ein solcher Suchpunkt mit überwältigender Wahrscheinlichkeit erreicht wird, wenn mit dem leeren Matching oder einem zufälligen Matching begonnen wird.

Die Chance, in einem Schritt den g-Wert zu erhöhen, ist besonders hoch, wenn ein Matching mit einem g-Wert von höchstens $h/2$ vorliegt. Die Analyse beschränkt sich daher auf diese Situationen. Für die RLS zeigen wir wieder mithilfe des Gambler's-Ruin-Problems, dass die Wahrscheinlichkeit, dass der g-Wert auf 0 sinkt, bevor er wieder $h/2$ erreicht, $\mathrm{e}^{-\Omega(h)}$ ist. Für eine genügend kleine Konstante c sind dann selbst e^{ch} solcher Versuche, den g-Wert auf 0 zu senken, mit einer Wahrscheinlichkeit von $\mathrm{e}^{-\Omega(h)}$ erfolglos.

Für den (1+1)-EA verfahren wir analog, setzen aber das Drifttheorem ein. Zusätzlich müssen wir hier noch sicherstellen, dass der g-Wert in dem Moment, in dem er unter $h/2$ fällt, nicht schlagartig zu klein werden kann.

Sowohl für die RLS als auch für den (1+1)-EA gibt es für $g \leq h/2$ spezielle Situationen, in denen die Wahrscheinlichkeit groß ist, dass der g-Wert im nächsten Schritt sinkt. Diese Situationen können wir nicht mit dem Gambler's-Ruin-Problem bzw. dem Drifttheorem abdecken. Daher zeigen wir, dass diese Situationen mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(h)}$ den g-Wert in der Summe nur um so wenig absenken, dass die Analyse der übrigen Schritte nicht gefährdet ist.

Wir beginnen zunächst mit der Analyse der RLS und teilen die akzeptierten Mutationsschritte anhand der darin kippenden Kanten in vier Typen ein.

Typ-1-Schritte kippen eine oder zwei wählbare Kanten. Der g-Wert bleibt unverändert oder wächst um 1 oder 2.

Typ-2-Schritte kippen eine wählbare Kante und eine Matchingkante. Der g-Wert ändert sich höchstens um 1.

Typ-3-Schritte kippen eine Matchingkante und eine angrenzende freie Kante, sodass die Endpunkte der beiden Kanten in drei benachbarten Spalten liegen. Der g-Wert ändert sich höchstens um 1.

Typ-4-Schritte kippen eine Matchingkante und eine angrenzende freie Kante, die zwischen denselben Spalten liegen. Der g-Wert bleibt unverändert.

Behauptung 7.1. *Wenn der Suchpunkt der RLS ein Matching beschreibt, können alle akzeptierten Mutationsschritte genau einem der vier Typen zugeordnet werden. Der g-Wert ändert sich wie angegeben.*

Beweis. Offenbar können 1-Bit-Mutationen nur akzeptiert werden, wenn sie die Matchingeigenschaft erhalten und die Matchinggröße nicht vermindern. Daher sind kippende Matchingkanten ausgeschlossen und freie Kanten müssen wählbar sein. Also sind akzeptierte 1-Bit-Mutationen Typ-1-Schritte.

Aus demselben Grund können 2-Bit-Mutationen nur akzeptiert werden, wenn sie zwei wählbare Kanten kippen (Typ 1) oder eine freie Kante und eine Matchingkante (Typ 2, Typ 3, oder Typ 4). Falls die freie Kante eine wählbare Kante ist, kann die Matchingkante kein Nachbar dieser freien Kante sein. Dies führt zu Typ-2-Schritten. Andernfalls müssen die beiden kippenden Kanten benachbart sein, weil die freie, aber nicht wählbare Kante sonst die Matchingeigenschaft zerstört, wenn sie kippt. Da es keine Kanten zwischen den Knoten einer Spalte gibt, müssen die benachbarten Kanten mindestens zwei Spalten berühren (Typ 4) und können höchstens drei Spalten berühren (Typ 3).

Typ-1-Schritte können den g-Wert nicht senken, da keine Kante aus dem Matching entfernt wird. In Typ-2-Schritten kann die hinzugewählte Kante den g-Wert um 1 erhöhen und die entfernte Kante ihn um 1 senken. Insgesamt kann der g-Wert dadurch auch unverändert bleiben. Dasselbe gilt für Typ-3-Schritte. Zwischen zwei Spalten liegen entweder nur $\overline{M}^*$-Kanten oder nur M^*-Kanten. Typ-4-Schritte können nur zwei $\overline{M}^*$-Kanten kippen, denn die Kanten des perfekten Matchings M^* haben (per Definition) keine gemeinsamen Endpunkte. Da eine $\overline{M}^*$-Kante hinzugefügt und eine $\overline{M}^*$-Kante entfernt wird, bleibt der g-Wert unverändert. □

Typ-4-Schritte brauchen wir in unserer Analyse nicht zu berücksichtigen und Typ-1-Schritte sind aus unserer Sicht zumindest „nicht schädlich“, da wir daran interessiert sind, dass der g-Wert wächst. Wir werden sehen, dass in Typ-3-Schritten für $g \leq h/2$ die Chancen zugunsten einer Erhöhung des g-Wertes stehen. Situationen, in denen es wählbare Kanten gibt, ermöglichen jedoch Typ-2-Schritte. Da es nicht auszuschließen ist, dass der überwiegende Teil der Matchingkanten des aktuellen Suchpunktes $\overline{M}^*$-Kanten sind, könnte ein Typ-2-Schritt mit hoher Wahrscheinlichkeit den g-Wert senken. Daher versuchen wir zunächst, den „Gesamtschaden" abzuschätzen, den Typ-2-Schritte am g-Wert verursachen. Die Idee ist es, Typ-2-Schritte mit Typ-1-Schritten zu vergleichen. Beide Typen erfordern eine wählbare Kante. In einer Situation mit einer wählbaren Kante ist jedoch ein Typ-1-Schritt, der in jedem Fall das Matching vergrößert, viel wahrscheinlicher als ein Typ-2-Schritt. Da es offenbar höchstens $|M^*|$ Typ-1-Schritte geben kann, gibt es mit hoher Wahrscheinlichkeit nur wenige Situationen, in denen Typ-2-Schritte möglich sind.

Behauptung 7.2. *Es sei $h = \omega(\log m)$. Für den Zeitraum nach dem Erreichen des ersten Matchings gilt für die RLS Folgendes. Die Wahrscheinlichkeit, dass der g-Wert in allen Typ-2-Schritten, die in einer Situation mit einem g-Wert von höchstens $h/2$ stattfinden, in der Summe um höchstens $h/8$ sinkt, ist mindestens $1 - \mathrm{e}^{-\Omega(h)}$.*

Beweis. Typ-1- und Typ-2-Schritte können nur in Situationen auftreten, in denen es mindestens eine wählbare Kante gibt. Spätestens nach $|M^*| = (\ell' + 1)h$ Typ-1-Schritten kann es keine wählbare Kante mehr geben. Wir betrachten nun Situationen mit mindestens einer wählbaren Kante. Die Wahrscheinlichkeit eines Typ-1-Schritts, der nur eine bestimmte wählbare Kante kippt, ist $1/(2m)$. Die Wahrscheinlichkeit eines Typ-2-Schritts, der dieselbe wählbare Kante und dazu eine der g $\overline{M}^*$-Kanten des aktuellen Suchpunktes kippt, ist $g/(m(m-1))$. Gegeben, dass ein Typ-1- oder Typ-2-Schritt stattfindet, ist die bedingte Wahrscheinlichkeit für einen Typ-1-Schritt $\Omega(1)$ und für einen Typ-2-Schritt $O(g/m)$. Mit Chernoffschranken folgt, dass für eine geeignete Wahl einer Konstanten $c > 0$ in $ch\ell$ Schritten, die eine wählbare Kante kippen, mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(h \cdot \ell)}$ mindestens $|M^*|$ Typ-1-Schritte enthalten sind. Mit mindestens derselben Wahrscheinlichkeit gibt es insgesamt höchstens $O(\ell h)$ Typ-1- oder Typ-2-Schritte in Situationen mit einem g-Wert von höchstens $h/2$. Vorausgesetzt, es gibt höchstens so viele Typ-1- oder Typ-2-Schritte in diesen Situationen, ist die erwartete Anzahl an Typ-1-Schritten daran $O(gh\ell/m) = O(1)$. Für ein genügend großes m wird auch h genügend groß, sodass aus Chernoffschranken folgt, dass diese Zufallsvariable mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(h)}$ nicht größer als $h/8$ wird. Insgesamt ist damit die Wahrscheinlichkeit für höchstens $h/8$ Typ-1-Schritte in Situationen mit $g \leq h/2$ mindestens von derselben Größenordnung. □

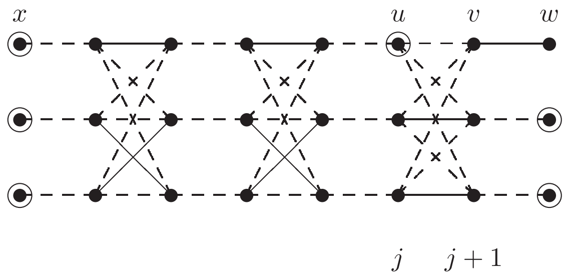

Bild 7.1: Situation mit nur zwei möglichen Typ-3-Schritten bei u

Behauptung 7.3. *Es sei $h \leq \ell - 2$ und der aktuelle Suchpunkt der RLS sei ein Matching mit einem g-Wert von höchstens $h/2$, das nicht das perfekte Matching ist. Für jeden speziellen Pfad gibt es 2 Typ-3-Schritte, die den g-Wert senken, und mindestens $h/2$ Typ-3- oder Typ-1-Schritte, die den g-Wert erhöhen.*

Beweis. Weil das Maximummatching ein perfektes Matching ist, ist jeder freie Knoten Endpunkt eines speziellen Pfades. Typ-3-Schritte benötigen einen freien Knoten u, der mit der kippenden freien Kante inzidiert. Ohne Beschränkung der Allgemeinheit sei u ein freier Knoten in Spalte j und j sei ungerade (siehe Bild 7.1). Wenn in einem Typ-3-Schritt die letzten beiden Kanten des speziellen Pfades von x zu u kippen, sinkt der g-Wert. Da u frei ist, kann es höchstens $\min\{h-1, g\}$ Matchingkanten zwischen den Spalten j und $j+1$ geben. Also hat u einen Nachbarknoten v in Spalte $j+1$, der nicht von einer Matchingkante zwischen Spalte j und $j+1$ überdeckt wird. Wenn nun in einem Typ-3-Schritt $\{u, v\}$ und $\{v, w\}$ kippen, wächst g. Falls v selbst ein freier Knoten ist, dann ist $\{u, v\}$ wählbar und die 1-Bit-Mutation, die nur $\{u, v\}$ kippt, ist sogar noch wahrscheinlicher als der zuvor betrachtete Typ-3-Schritt.

In der in Bild 7.1 gezeigten Situation ist der g-Wert viel größer als $h/2$. Aus unseren Vorüberlegungen folgt, dass es für $g \leq h/2$ mindestens $h/2$ Typ-3-Schritte (oder Typ-1-Schritte) gibt, die g erhöhen, sofern u nicht in der ersten oder letzten Spalte liegt. Wegen $h < \ell - 2$ liegt wenigstens ein Endpunkt jedes speziellen Pfades nicht in der ersten oder letzten Spalte. Da jeder freie Knoten auf einem und nur einem speziellen Pfad liegt, gibt es also pro speziellen Pfad höchstens 2 absenkende Typ-3-Schritte und $h/2$ steigernde Typ-3- oder Typ-1-Schritte. □

Lemma 7.4. *Es sei $h = \omega(\log m)$ und $h \leq \ell - 2$. Wenn die RLS ein Matching mit einem g-Wert von mindestens $h/2$ erreicht hat, ist für eine geeignet gewählte Konstante $c > 0$ die Wahrscheinlichkeit, dass das perfekte Matching in den nächsten $e^{c \cdot h}$ Schritten gefunden wird, durch $e^{-\Omega(h)}$ nach oben beschränkt.*

Beweis. Wir beschränken unsere Analyse auf die Schritte in Situationen mit einem g-Wert von höchstens $h/2$. Dann ist die Gesamtzahl der Schritte nur größer. Für den g-Wert sind unter den akzeptierten Schritten nur Typ-1-, Typ-2- und Typ-3-

Schritte relevant. Nach Behauptung 7.2 sinkt der g-Wert mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(h)}$ durch Typ-2-Schritte insgesamt um höchstens $h/8$. Es genügt nun zu zeigen, dass Typ-1- und Typ-3-Schritte allein den g-Wert mit einer Wahrscheinlichkeit von $1-\mathrm{e}^{-\Omega(h)}$ nicht unter $h/4$ absenken. Der wahre g-Wert, d. h. unter Einbeziehung der Typ-2-Schritte, liegt dann stets bei mindestens $h/8$. Da unsere Vorüberlegungen für alle g-Werte zwischen 1 und $h/2$ gültig sind, können wir diese Trennung der Schritte in der Analyse vornehmen.

Die Typ-3-Schritte und Typ-1-Schritte beschreiben gemäß Behauptung 7.3 ein unfaires Spiel nach Art des Gambler's-Ruin-Problems, wobei die Gewinnwahrscheinlichkeit des Kartenspielers für eine Runde $p = 1-O(1/h)$ ist und $t := t(h) = O(1/h)$ ist. Wir warten den ersten Zeitpunkt ab, zu dem g unter $h/2$ fällt, d. h. $\lceil h/2 \rceil - 1$ erreicht. Das Anfangskapital des Spielers entspricht $(\lceil h/2 \rceil - 1) - \lceil h/4 \rceil = \Omega(h)$. Also ist die Wahrscheinlichkeit, dass der g-Wert $h/2$ wieder überschreitet, bevor er $h/4$ unterschreitet, $1 - \mathrm{e}^{-\Omega(h)}$. Für eine genügend klein gewählte Konstante $c \geq 0$ ist die Wahrscheinlichkeit, dass nach $\mathrm{e}^{c \cdot h}$ Unterschreitungen des Schwellenwertes $h/2$ jedes Mal wieder $h/2$ überschritten wird, bevor h/4 unterschritten wird, von derselben Größenordnung. □

Für den (1+1)-EA gehen wir nun ähnlich vor. Für die RLS haben wir den ungünstigen Einfluss der Typ-2-Schritte durch ihre geringe Anzahl beschränken können. Für das Argument haben wir Typ-1-Schritte benutzt, die ebenfalls auf wählbare Kanten angewiesen sind. Bei der Analyse des (1+1)-EA fassen wir entsprechende Schritte unter *einem* Begriff zusammen: *Riskante* Schritte sind akzeptierte Mutationsschritte, die mindestens eine wählbare M^*-Kante kippen. Sie heißen so, weil für jede kippende wählbare M^*-Kante eine beliebige $\overline{M}^*$-Kante aus dem Matching entfernt werden kann, ohne dass die Fitness sinkt. In einer Situation mit nur einer wählbaren Kante ist die Wahrscheinlichkeit, dass g durch Kippen der wählbaren Kante und einer gewählten $\overline{M}^*$-Kante sinkt, $\Theta(g/m^2)$. Hingegen ist die Wahrscheinlichkeit $\Theta(1/m)$, dass nur die wählbare Kante kippt und so das Matching vergrößert wird.

Behauptung 7.5. *Es sei $h = \omega(\log m)$. Für den Zeitraum nach dem Erreichen des ersten Matchings gilt für den (1+1)-EA Folgendes. Die Wahrscheinlichkeit, dass der g-Wert in allen riskanten Schritten, die in einer Situation mit einem g-Wert von höchstens $h/2$ stattfinden, in der Summe um höchstens $h/8$ sinkt, ist mindestens* $1 - \mathrm{e}^{-\Omega(h)}$.

Beweis. Wir arbeiten unter der Bedingung, dass in den betrachteten Schritten mindestens eine wählbare M^*-Kante kippt, und rechnen mit bedingten Wahrscheinlichkeiten. Wir betrachten nur Situation, in denen $g \leq h/2$ gilt.

Das Matching vergrößert sich um mindestens eine Kante, wenn in dem riskanten Schritt keine weitere Kante kippt. Dies geschieht mit einer Wahrscheinlichkeit von $(1-1/m)^{m-1} \geq 1/\mathrm{e}$. Aus Chernoffschranken folgt, dass $2\mathrm{e}\ell' h$ riskante Schritte das Matching mindestens $|M^*| = (\ell'+1)h$-mal mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(\ell h)}$ anwachsen lassen. Dadurch ist die Anzahl riskanter Schritte begrenzt, aber noch nicht deren Gesamtwirkung auf g. Wenn der g-Wert in einem riskanten Schritt um einen bestimmten Betrag fällt, dann müssen genauso viele gewählte $\overline{M}^*$-Kanten in diesem Schritt kippen. In jedem der betrachteten riskanten Schritte können höchstens $h/2$ gewählte $\overline{M}^*$-Kanten kippen. Die Wahrscheinlichkeit, dass eine bestimmte $\overline{M}^*$-Kante kippt, ist $1/m$. Also ist die erwartete Anzahl kippender $\overline{M}^*$-Kanten in $2\mathrm{e}\ell' h$ riskanten Schritten für $g \leq h/2$ höchstens $2\mathrm{e}\ell' h \cdot (h/2) \cdot (1/m) = O(1)$. Aus Chernoffschranken folgt nun wieder, das diese Zufallsvariable mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(h)}$ durch $h/8$ beschränkt ist, wenn m (und dadurch auch h) nicht zu klein ist. □

Behauptung 7.6. *Für eine beliebige Konstante $c > 0$ enthält eine Folge von $\mathrm{e}^{c \cdot h}$ Mutationsschritten des (1+1)-EA mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(h \log h)}$ keinen Mutationsschritt, der mindestens $h/8$ der $\overline{M}^*$-Kanten kippt, wenn zu Beginn der Folge ein Matching vorliegt.*

Aus der Behauptung folgt, dass mit mindestens derselben Wahrscheinlichkeit in jedem Schritt der Folge der g-Wert um weniger als $h/8$ sinkt.

Beweis zu Behauptung 7.6. In jedem Matching sind höchstens $|M^*|$ der $\overline{M}^*$-Kanten enthalten, weil ein Matching nicht größer sein kann als das perfekte Matching. Die Wahrscheinlichkeit, dass in einem Schritt mindestens $h/8$ der darin enthaltenen $\overline{M}^*$-Kanten kippen, ist höchstens

$$\binom{(\ell'+1)h}{h/8} \frac{1}{m^{h/8}} \leq \left(\frac{(\ell'+1)h}{m}\right)^{h/8} = \mathrm{e}^{-\Omega(h \log h)}.$$

In einer Folge aus $\mathrm{e}^{c \cdot h}$ Schritten ist so ein Schritt nur mit einer Wahrscheinlichkeit von $\mathrm{e}^{-\Omega(h \log h)}$ enthalten. □

Lemma 7.7. *Es sei $h = \omega(\log m)$ und $h \leq \ell - 2$. Wenn der (1+1)-EA ein Matching mit einem g-Wert von mindestens $h/2$ erreicht hat, ist für eine geeignet gewählte Konstante $c > 0$ die Wahrscheinlichkeit, dass das perfekte Matching in den nächsten $\lceil \mathrm{e}^{c \cdot h} \rceil$ Schritten gefunden wird, durch $\mathrm{e}^{-\Omega(h)}$ nach oben beschränkt.*

Beweis. Gemäß Behauptung 7.5 ist die Wahrscheinlichkeit klein genug, dass riskante Schritte den g-Wert insgesamt um mehr als $h/8$ senken, wenn wir uns nur auf Situationen mit $g \leq h/2$ beschränken. Wegen Behauptung 7.6 ist auch die Wahrscheinlichkeit klein genug, dass der g-Wert in den Schritten, in denen er unter $h/2$ fällt, gleich unter $3h/8$ fällt. Wir können also davon ausgehen, dass es zuvor einen Zeitpunkt gibt, zu dem g zwischen $3h/8$ und $h/2$ liegt. Die folgenden Ereignisse treten jeweils mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(h)}$ ein. Wir werden zeigen, dass die übrigen, also nicht riskanten Schritte den g-Wert in Situationen mit $g \leq h/2$ insgesamt ebenfalls nur um $\lfloor h/8 \rfloor$ senken. Diese übrigen Schritte allein würden den g-Wert also insgesamt allenfalls auf $2h/8$ absenken. Wenn wir nun die Wirkung der riskanten Schritte hinzunehmen, dann ist der wahre g-Wert am Ende höchstens um $h/8$ geringer, also noch mindestens $\lfloor h/8 \rfloor$. Dies ist unabhängig davon, wie sich riskante und nicht riskante Schritte miteinander mischen, denn wir werden darauf achten, dass alle unsere Abschätzungen für die nicht riskanten Schritte in jeder Situation mit einem g-Wert zwischen 1 und $h/2$ gültig sind.

Wir betrachten im Folgenden Suchpunkte x, die Matchings mit $1 \leq g(x) \leq h/2$ sind, und schätzen die Wahrscheinlichkeiten $p_j(x)$ für Veränderungen des g-Wertes um j in nicht riskanten Schritten ab. Das bedeutet, wir arbeiten unter der Bedingung, dass der Mutationsoperator keine wählbaren M^*-Kanten kippt. Dazu sei ein *langer* spezieller Pfad ein spezieller Pfad aus mindestens drei Kanten, d. h., lange spezielle Pfade bestehen aus mehr als einer wählbaren M^*-Kante. Unsere Schranken werden von $k(x)$, der Zahl der langen speziellen Pfade im durch x beschriebenen Matching, abhängen. Analog zum Beweis zu Lemma 6.11 notieren wir Schranken für $p_j(x)$, die für alle Suchpunkte mit $1 \leq g(x) \leq h/2$ und $k(x) = k$ gelten, mit $p_j(k)$.

Zunächst behaupten wir, dass

$$\begin{aligned} p_j(k) &:= 0 \qquad \text{für } j \geq 2 \text{ und} \\ p_1(k) &:= kh/(2em^2) \end{aligned}$$

entsprechende untere Schranken und

$$\begin{aligned} p_{-1}(k) &:= (2k+2)/m^2, \\ p_{-2}(k) &:= 1/(m^2 \ell h) \qquad \text{und} \\ p_{-j}(k) &:= 1/m^j \qquad \text{für } j \geq 3 \end{aligned}$$

entsprechende obere Schranken sind.

Für $j \geq 2$ ist nichts zu zeigen. Mit exakt den gleichen Argumenten wie im Beweis zu Behauptung 7.3 folgt, dass es mindestens $h/2$ Mutationen gibt, die den g-Wert erhöhen. Diese kippen entweder zwei nicht wählbare Kanten oder eine wählbare $\overline{M}^*$-Kante. Daher ist $(kh/2) \cdot (1/m^2) \cdot (1 - 1/m)^{m-2}$ eine korrekte untere Schranke für $p_1(x)$, aus der $p_1(k)$ folgt.

Ein notwendiges Ereignis, damit g um 1 sinkt, ist, dass eine gewählte $\overline{M}^*$-Kante e und eine freie M^*-Kante e' kippen. Hier gibt es drei Fälle zu unterscheiden. Falls e und e' benachbarte Kanten sind, können sie die beiden ersten oder letzten Kanten eines langen verbessernden Pfades sein. Offenbar gibt es nur $2k$ solcher Kantenpaare und die Wahrscheinlichkeit, dass eines dieser Paare kippt, ist höchstens $2k/m^2$. Der zweite Fall ist, dass e und e' zwar benachbart, aber nicht die ersten oder letzten beiden Kanten eines langen speziellen Pfades sind. In diesem Fall hat die freie M^*-Kante e' eine weitere gewählte $\overline{M}^*$-Kante e'' als Nachbarn, die nun ebenfalls

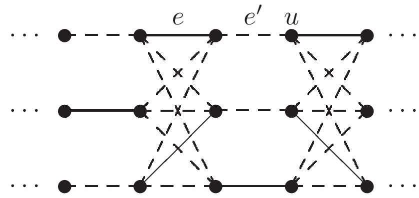

Bild 7.2: Wäre u frei, dann wären e und e' die beiden ersten Kanten auf einem langen speziellen Pfad, der in u beginnt.

kippen muss. Andernfalls wäre ein Endpunkt u von e' ein freier Knoten, sodass e und e' die beiden ersten Kanten auf einem langen verbessernden Pfad wären (siehe Bild 7.2). Wie viele Möglichkeiten gibt es, die freie M^*-Kante e' zwischen zwei gewählten $\overline{M}^*$-Kanten e und e'' zu wählen? Die g $\overline{M}^*$-Kanten des Matchings berühren g verschiedene Knoten in Spalten mit gerader Nummer und ebenso viele Knoten in Spalten mit ungerader Nummer. Also können höchstens g wählbare M^*-Kanten je zwei solche Knoten verbinden. Weil der Suchpunkt ein Matching ist, gibt es folglich höchstens g Wahlmöglichkeiten für das Tripel aus e, e' und e''. Das führt zu einer Wahrscheinlichkeit von höchstens g/m^3 für den zweiten Fall. Der dritte Fall ist, dass die gewählte $\overline{M}^*$-Kante e und die freie M^*-Kante e' nicht benachbart sind. Da wir unter der Bedingung arbeiten, dass keine wählbaren M^*-Kanten kippen, brauchen wir nur den Fall zu betrachten, dass e' nicht wählbar ist. Das bedeutet, e' hat mindestens eine Nachbarkante e'', die zum Matching des aktuellen Suchpunktes gehört. Also gibt es höchstens g Wahlmöglichkeiten für e und höchstens $(\ell'+1)h$ Wahlmöglichkeiten für das Paar aus e' und e'' . Dies führt zu einer Wahrscheinlichkeit von höchstens $g(\ell'+1)h/m^3 \le 1/m^2$ für diese Möglichkeit. Insgesamt ist die Wahrscheinlichkeit, dass g um 1 sinkt, (für nicht zu kleines m) höchstens

$$\frac{2k}{m^2} + \frac{g}{m^3} + \frac{1}{m^2} \le \frac{2k+2}{m^2}.$$

Um g um 2 zu vermindern, ist es notwendig, dass zwei der gewählten $\overline{M}^*$-Kanten e_1 und e_2 kippen. Also gibt es höchstens $\binom{g}{2} \le g^2/2$ Möglichkeiten, e_1 und e_2 zu wählen. Damit der Schritt akzeptiert werden kann, ist es notwendig, dass zwei freie M^*-Kanten e_1' und e_2' ebenfalls kippen. Wir brauchen hier nur den Fall zu betrachten, dass e_1' und e_2' nicht wählbar sind. Dies impliziert, dass sie Nachbarn von gewählten $\overline{M}^*$-Kanten sind. Weil jede $\overline{M}^*$-Kante zwei M^*-Kanten als Nachbarn hat, gibt es höchstens $\binom{2g}{2} \le 2g^2$ Möglichkeiten, e_1' und e_2' als Nachbarn von e_1 und e_2 zu wählen. Da der Mutationsoperator des (1+1)-EA M^*-Kanten und $\overline{M}^*$-Kanten unabhängig kippt, treten die beiden notwendigen Ereignisse höchstens mit einer Wahrscheinlichkeit von

$$\frac{g^2}{2m^2} \cdot \frac{2g^2}{m^2} \le \frac{1}{16m^2\ell'^2} \le \frac{1}{m^2\ell h}$$

ein.

Damit g um $j \geq 3$ sinkt, müssen mindestens j der gewählten $\overline{M}^*$-Kanten kippen und ebenso viele freie M^*-Kanten. Es gibt $\binom{g}{j}$ Wahlmöglichkeiten für die ersten j Kanten und höchstens $\binom{(\ell'+1)h}{j}$ Wahlmöglichkeiten für die zweiten j Kanten. Also ist die Wahrscheinlichkeit für einen solchen Schritt

$$\frac{\binom{g}{j}\binom{(\ell'+1)h}{j}}{m^{2j}} \leq \frac{(g(\ell'+1)h)^j}{m^j \cdot m^j} \leq \frac{1}{m^j}.$$

Damit sind die behaupteten Schranken $p_j(k)$ bewiesen.

Wir wollen nun wie oben angekündigt nachweisen, dass der g-Wert unter Auslassung der riskanten Schritte in Situationen mit $g \leq h/2$ nicht unter $h/4$ sinkt. Der entscheidende Grund, warum uns dies gelingen wird, ist, dass für alle k die Schranke $p_1(k)$ um einen Faktor von $\Theta(h)$ größer als $p_{-1}(k)$ ist. Da h nicht mehr konstant ist, sondern $h = \omega(\log m)$ gilt, ist dies ein entscheidender Vorteil für g vergrößernde Schritte. Hier nutzen wir also den Knotengrad des Graphen aus. Die übrigen Schranken $p_{-j}(k)$ sind so klein, dass sie kaum Einfluss haben werden.

In dem Moment, in dem der g-Wert mindestens $3h/8$ und höchstens $h/4$ wird, beginnen wir den stochastischen Prozess, der durch die Änderung des g-Wertes in nicht riskanten Schritten beschrieben ist, zu betrachten. In diesem Moment sei der g-Wert gleich g_{Start}. Wir zeigen dazu, dass die Wahrscheinlichkeit in e^{ch} Schritten den Zielwert $g_{\text{Start}} - \lfloor h/8 \rfloor \geq h/4$ zu erreichen, höchstens $\mathrm{e}^{-\Omega(h)}$ ist. Es kann dabei immer wieder passieren, dass der g-Wert einen Wert größer als g_{Start} erreicht. Falls der g-Wert in diesem Moment oder danach wieder höchstens $h/2$ wird, interpretieren wir dies als einen Neustart mit neuem Startwert g'_{Start} mit $3h/8 \leq g'_{\text{Start}} \leq h/2$ und Zielwert $g'_{\text{Start}} - \lfloor h/8 \rfloor$. Effektiv bedeutet das, dass wir Schritte, in denen sich der g-Wert um mehr als $\lfloor h/8 \rfloor$ von seinem Zielwert entfernt, durch Schritte ersetzen, in denen die Entfernung zum Zielwert $\lfloor h/8 \rfloor$ wird. Das kann die Wahrscheinlichkeit, den Zielwert in dem abgesteckten Zeitrahmen zu erreichen, nur erhöhen. Daher können wir den unterbrochenen Prozess zu einem Prozess zusammenfassen, bei dem der g-Wert nicht über seinen ursprünglichen Startwert g_{Start} hinaus wachsen kann.

Um das Drifttheorem anzuwenden, ist der Parameter ℓ in Theorem 6.3 durch h zu ersetzen. Wir wählen $b(h) := g_{\text{Start}} \leq h/2$ und $a(\ell) := b(h) - \lfloor h/8 \rfloor \geq h/4$, wobei $b(h)$ auch der größtmögliche g-Wert sein soll. Damit sind die ersten beiden Driftbedingungen erfüllt. Um die dritte Bedingung zu prüfen, müssen wir die Summe

$$\sum_{-g(x) \leq j \leq b(h)-g(x)} \mathrm{e}^{-\lambda \cdot j} \cdot p_j(x)$$

untersuchen. Dazu gehen wir analog zum Beweis zu Lemma 6.11 vor. Dort haben wir uns schon überlegt, dass sich die Summe nur vergrößert, wenn wir die $p_j(x)$ für $j \geq 1$ durch die unteren Schranken $p_j(k)$ und für $j \leq 1$ durch die oberen Schranken $p_j(k)$ ersetzen. Die restliche Wahrscheinlichkeit sei $p_0(k)$.

Indem wir zu den bedingten Wahrscheinlichkeiten $q_j(k) := p_j(k)/(1-p_0(k))$ und $q_0(k) := 0$ übergehen, zählen wir nur diejenigen Schritte, die tatsächlich den g-Wert verändern. Es genügt, diesen „beschleunigten“ Prozess zu untersuchen. Es gilt

$$1 - p_0(k) \;=\; p_1(k) + \sum_{1\le j\le b(h)-g(x)} p_{-j}(k) \;=\; \frac{1}{m^2}\Big(\frac{kh}{2\mathrm{e}} + 2k + 2 + O(1/(\ell h))\Big)$$

und somit für $j = 1$ und $j \le -1$

$$q_j(k) \;=\; \frac{m^2 \cdot p_j(k)}{kh/(2e) + 2k + 2 + O(1/(\ell h))} \;\le\; m^2 \cdot p_j(k).$$

Es genügt also,

$$\mathrm{e}^{-\lambda} q_1(k) + \mathrm{e}^{\lambda} q_{-1}(k) + \sum_{j\ge 2} \mathrm{e}^{j\cdot\lambda} q_{-j}(k)$$

nach oben abzuschätzen. Dazu zeigen wir, dass es Konstanten $\lambda > 0$ und $\delta > 0$ gibt, sodass

$$\mathrm{e}^{-\lambda} q_1(k) + \mathrm{e}^{\lambda} q_{-1}(k) \;\le\; 1 - \delta \qquad \text{und} \qquad \sum_{j\ge 2} \mathrm{e}^{j\cdot\lambda} q_{-j}(k) \;=\; o(1)$$

gelten. Daraus folgt die dritte Bedingung. Die zweite Behauptung ist leicht zu prüfen, denn $q_{-j}(k) \le m^2 p_{-j}(k)$ impliziert $q_{-2}(k) \le 1/(\ell\cdot h)$ sowie $q_{-j}(k) \le 1/m^{j-2}$ für $j \ge 3$. Also folgt

$$\sum_{j\ge 2} \mathrm{e}^{j\cdot\lambda} q_{-j}(k) \;\le\; \frac{\mathrm{e}^{2\cdot\lambda}}{\ell h} + \mathrm{e}^{2\lambda} \sum_{j\ge 3} \left(\frac{\mathrm{e}^{\lambda}}{m}\right)^{j-2} \;=\; O\Big(\frac{1}{\ell h}\Big).$$

Aus dem Beweis zu Lemma 6.11 wissen wir, dass es für die erste Behauptung genügt, eine positive Konstante $\beta > 0$ als untere Schranke für die Differenz $q_1(k) - q_{-1}(k)$ nachzuweisen. Für genügend großes h gilt

$$q_1(k) - q_{-1}(k) \;=\; \frac{kh/(2\mathrm{e}) - (2k+2)}{kh/(2\mathrm{e}) + 2k + 2 + O(1/(\ell h))} \;\ge\; 1/2.$$

Daraus erhalten wir wie im Beweis zu Lemma 6.11 die Konstanten λ und δ.

In einer Situation, die der letzten Bedingung entspricht, gilt $g = b(\ell)$, weil wir nicht gestatten, dass g größer wird. Mit der oberen Schranke $p_1(k) := 0$ gilt dann

$$1 - p_0(k) \;=\; \frac{1}{m^2}\big(2k + 2 + O(1/(\ell h))\big) \;\ge\; \frac{1}{m^2}.$$

Dies führt zu neuen q-Werten für die Summe

$$\mathrm{e}^{\lambda} q_{-1}(k) + \sum_{j\ge 2} \mathrm{e}^{j\cdot\lambda} q_{-j}(k).$$

Der erste Summand ist durch die Konstante e^λ nach oben beschränkt. Die übrigen Summanden können genau wie zuvor durch $O(1/(\ell h))$ abgeschätzt werden, weil $q_j(k) \leq m^2 \cdot p_j(k)$ gültig bleibt. Insgesamt ist die Summe also durch eine Konstante $D > 1$ beschränkt.

Dem Drifttheorem zufolge gibt es also eine Konstante $c \geq 0$, sodass $\lceil e^{c \cdot h} \rceil$ Schritte nur mit einer Wahrscheinlichkeit von $e^{-\Omega(h)}$ den g-Wert auf $h/4$ senken. Die „Fehlerwahrscheinlichkeiten", dass riskante Schritte den g-Wert zusätzlich um mehr als $h/8$ senken und dass ein einzelner Schritt den g-Wert um mindestens $h/8$ senkt, sind von derselben Größenordnung. □

Lemma 7.4 und Lemma 7.7 setzen voraus, dass ein Matching mit $g \geq h/2$ erreicht ist. Um insgesamt exponentielle Optimierzeit mit überwältigender Wahrscheinlichkeit zu erhalten, ist es noch notwendig zu zeigen, dass aus einer Anfangssituation heraus ein entsprechender Suchpunkt mit überwältigender Wahrscheinlichkeit erreicht wird.

Lemma 7.8. *Es sei $h = \omega(1)$. Wenn man gleichverteilt aus allen Matchings ein Matching wählt, hat das gewählte Matching einen g-Wert von mindestens h mit einer Wahrscheinlichkeit von $1 - e^{-\Omega(\ell \cdot h \log h)}$.*

Beweis. Wir beweisen eine obere Schranke von $e^{O(\ell \cdot h)}$ für die Anzahl der Matchings mit geringem g-Wert von höchstens h und eine untere Schranke von $e^{\Omega(\ell \cdot h \log h)}$ für die Gesamtzahl der Matchings. Dann folgt, dass der Anteil der Matchings mit einem g-Wert von höchstens h durch $e^{-\Omega(\ell \cdot h \log h)}$ beschränkt ist.

Wir beginnen mit der oberen Schranke. Für $h \geq 3$ gibt es höchstens

$$\sum_{0 \leq k \leq h} \binom{\ell' h^2}{k} \;\leq\; (h+1) \cdot \frac{(\ell' h^2)^h}{h!} \;\leq\; (\ell' h^2)^h$$

Möglichkeiten, höchstens h der $\overline{M}^*$-Kanten auszuwählen. Jede dieser Möglichkeiten kann mit höchstens jeder der $2^{(\ell'+1)h}$ Teilmengen des perfekten Matchings M^* kombiniert werden. Daraus erhalten wir $(\ell' h^2)^h \cdot 2^{(\ell'+1)h} = 2^{h(\log \ell' + 2 \log h)} \cdot 2^{(\ell'+1)h} = 2^{O(\ell \cdot h)}$ als obere Schranke für die Anzahl der Matchings mit geringem g-Wert.

Die untere Schranke für die Zahl der Matchings erhalten wir, indem wir die Knoten in den Spalten 1, 5, 9, 13 usw. betrachten (siehe Bild 6.2). In einem Matching gibt es für jeden Knoten in diesen Spalten die folgenden $h+2$ Möglichkeiten. Er kann frei sein, von der inzidenten M^*-Kante überdeckt sein oder von einer der h inzidenten $\overline{M}^*$-Kanten überdeckt sein. Da jede der betrachteten Spalten zu ihrer nächsten betrachteten Spalte einen Abstand von drei Kanten einhält, beeinflussen sich diese Spalten nicht gegenseitig. Innerhalb einer Spalte sind diese Wahlen jedoch nicht unabhängig. Daher wählen wir in jeder betrachteten Spalte nur für $\lceil h/2 \rceil$ Knoten eine der $h/2$ Möglichkeiten aus. Wenn wir bestimmen, dass ein Knoten frei oder von der angrenzenden $\overline{M}^*$-Kante überdeckt sein soll, dann berührt diese

Entscheidung keinen der anderen Knoten in derselben Spalte. Wenn wir jedoch entscheiden, dass ein Knoten von einer angrenzenden $\overline{M}^*$-Kante überdeckt sein soll, dann verringert dies die Auswahlmöglichkeiten der anderen Knoten in derselben Spalte um 1. Da wir nur für $\lceil h/2 \rceil$ Knoten eine der Möglichkeiten auswählen, bleibt bei jedem dieser Knoten die Wahl aus mindestens $h+2-\lceil h/2 \rceil \geq h/2$ Möglichkeiten. Also gibt es mindestens $(h/2)^{\Omega(\ell h)} = 2^{\Omega(\ell \cdot h \log h)}$ verschiedene Matchings. □

Da ℓ und h nicht gleichzeitig klein sein können, vermittelt uns Lemma 7.8, dass nur ein verschwindend kleiner Bruchteil der Matchings einen g-Wert kleiner als h hat. Der Optimierungsprozess muss irgendwann auf ein Matching stoßen, um zum Optimum zu gelangen. Es scheint nun sehr wahrscheinlich, dass dieses ein Matching mit genügend großem g-Wert ist. Für den typischen Fall, dass mit dem leeren Matching begonnen wird, zeigen wir, dass dies so ist.

Lemma 7.9. *Es sei $h = \omega(\log m)$ und $h \leq \ell - 2$. Wenn die RLS oder der (1+1)-EA mit dem leeren Matching startet, wird mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(\ell)}$ später ein Matching mit $g = \Omega(h^2)$ erreicht.*

Beweis. Wir zeigen zunächst, dass es unwahrscheinlich ist, dass innerhalb von $\ell' h/4$ Schritten das perfekte Matching M^* erzeugt wird. In einer Phase dieser Länge gibt es $|M^*|\ell' h/4$ Möglichkeiten für den (1+1)-EA, M^*-Kanten zu kippen. Die erwartete Zahl kippender M^*-Kanten ist durch $|M^*|\ell' h/(4m) \leq (\ell' + 1)/4$ beschränkt. Für die RLS ist die Wahrscheinlichkeit $|M^*|/m$, dass die erste Kante, die kippt, eine M^*-Kante ist. Falls die RLS noch eine zweite Kante kippt, ist diese mit einer Wahrscheinlichkeit von höchstens $|M^*|/(m-1)$ eine M^*-Kante. Es gibt also höchstens $\ell' h/2$ Möglichkeiten für die RLS, M^*-Kanten zu kippen, und die erwartete Zahl kippender M^*-Kanten ist durch $|M^*|\ell' h/(2(m-1)) \leq (\ell' + 1)/2$ beschränkt. Mithilfe von Chernoffschranken ist die Wahrscheinlichkeit, dass sie mehr als $\ell' < |M^*|$ verschiedene M^*-Kanten berühren, für beide Suchheuristiken durch $1 - 2^{-\Omega(\ell)}$ nach unten begrenzt. Wir arbeiten im Folgenden unter der Bedingung, dass höchstens so viele M^*-Kanten berührt werden.

Solange wie der aktuelle Suchpunkt noch weniger als $\ell' h/4$ der $\overline{M}^*$-Kanten auswählt, sind wegen der Bedingung höchstens $k := \ell' h/4 + \ell'$ Kanten insgesamt ausgewählt. Jede dieser k Kanten ist mit höchstens $2h$ Kanten benachbart. Also muss es mindestens $\ell' h^2 - \ell' h/4 - 2hk$ wählbare $\overline{M}^*$-Kanten geben. Diese Zahl ist durch

$$\ell' h^2 - \ell' h/4 - 2h(\ell' h/4 + \ell') \;=\; \ell' h^2(1/2 - 9/(4h)) \;\geq\; \ell' h^2/4$$

für genügend großes h nach unten beschränkt. Die Bedingung, unter der wir arbeiten, berührt nicht die Wahrscheinlichkeit von Schritten, die nur eine wählbare $\overline{M}^*$-Kante kippen. Die Wahrscheinlichkeit für einen solchen Schritt ist demnach für beide Suchheuristiken mindestens $(\ell' h^2/4) \cdot (1/m) \cdot (1 - 1/m)^{m-1} = \Omega(1)$. Also folgt aus Chernoffschranken, dass der g-Wert durch diese Schritte in der Phase mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(\ell \cdot h)}$ um insgesamt $\Omega(\ell \cdot h)$ wächst. Unter unserer Bedingung kann der g-Wert um insgesamt höchstens ℓ' gesenkt werden, und zwar durch Schritte, die $\overline{M}^*$-Kanten durch M^*-Kanten ersetzen. Insgesamt wächst g auf $\Omega(\ell h) = \Omega(h^2)$ mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(\ell)}$. $\square$

Das folgende Theorem fasst die Ergebnisse dieses Kapitels zusammen. Es folgt aus Lemma 7.4, Lemma 7.7, Lemma 7.8 und Lemma 7.9.

Theorem 7.10. *Für die Graphen $G_{h,\ell}$ mit $h = \omega(\log m)$ und $h \leq \ell - 2$ gilt für die RLS und den (1+1)-EA Folgendes.*

Wenn die Suche mit einem Matching mit einem g-Wert von mindestens $h/2$, einem zufälligen Matching oder dem leeren Matching beginnt, existiert eine Konstante $c > 0$, sodass die Wahrscheinlichkeit mindestens $1 - \mathrm{e}^{-\Omega(h)}$ ist, dass die Optimierzeit mindestens $\mathrm{e}^{c \cdot h}$ ist.

8 Zusammenfassung

Als Beispiel für die Analyse der Optimierzeit randomisierter Suchheuristiken für kombinatorische Optimierungsprobleme wurden hier die randomisierte lokale Suche (RLS) und der (1+1)-EA für das Matchingproblem untersucht. Das Matchingproblem ist interessant, weil es ein gut bekanntes kombinatorisches Optimierungsproblem ist, das in vielen Situationen keine unmittelbare Verbesserung einer Lösung durch kleine, lokale Veränderungen erlaubt.

Es wurde gezeigt, dass das Matchingproblem in typischen Modellierungen als Black-Box-Problem geringe Black-Box-Komplexität hat. Für die gewählte Modellierung wurde gezeigt, dass die RLS und der (1+1)-EA effizient Maximummatchings approximieren können. Durch Hinzufügen einer Stoppregel und einer geeigneten Implementierung für die Fitnessfunktion werden sie zu polynomiellen, randomisierten Approximationsschemata. Auf Pfadgraphen finden beide Heuristiken in erwartet polynomieller Zeit Maximummatchings. Dasselbe gilt für die RLS für beliebige Bäume. Die Resultate über Pfadgraphen und Bäume sind vermutlich die ersten „Positivresultate" für das Matchingproblem, in denen polynomielle Optimierzeit für eine randomisierte Suchheuristik für eine Graphklasse nachgewiesen wird.

Es gibt jedoch eine Graphklasse, in der auch Graphen mit kleinem Knotengrad exponentielle erwartete Optimierzeit haben. Für Graphen derselben Graphklasse, jedoch mit Knotengrad $\omega(\log m)$, ist die Optimierzeit für viele Startpunkte sogar mit überwältigender Wahrscheinlichkeit exponentiell. Zu diesen Startpunkten gehören alle Matchings, die auch einige wenige Kanten wählen, die nicht zu den Kanten des eindeutig bestimmten Maximummatching gehören. Zu diesen Startpunkten gehört aber auch das leere Matching. Daraus folgt, dass (für diese Startpunkte) selbst Multistartvarianten erfolglos sind. Diese Resultate sind weiter gehend als bekannte Resultate über Simulated Annealing und den Metropolisalgorithmus für das Matchingproblem.

Teil II

Zur Analyse einer grundlegenden randomisierten Suchheuristik für mehrkriterielle Probleme

9 Einleitung

Dieser Teil der Dissertation beschäftigt sich mit der Analyse der Optimierzeit des einfachen evolutionären Algorithmus für mehrkriterielle Optimierungsprobleme SEMO (Simple Evolutionary Multi-objective Optimizer). In diesem Kapitel wird das Szenario der mehrkriteriellen Optimierung vorgestellt und die Zielsetzung der Optimierung festgelegt, d. h. wie eine Lösung (genauer: eine Lösungsmenge) beschaffen sein soll. Danach wird die randomisierte Suchheuristik SEMO beschrieben. Abschließend wird ein kurzer Überblick über Arbeiten in diesem Themengebiet gegeben.

Optimierungsprobleme, wie sie schon im alltäglichen Leben vorkommen, sind selten so beschaffen, dass es nur ein Optimierungsziel gibt. So beinhaltet schon die Anschaffung eines technischen Gerätes, beispielsweise eines PKWs, ein Optimierungsproblem mit vielen verschiedenen Optimierungszielen. Zu den (messbaren) Optimierungszielen gehören z. B. der Preis, die Größe, die Höchstgeschwindigkeit, die Motorleistung, der Kraftstoffverbrauch, das Alter usw. Optimierungsprobleme dieser Art gibt es in den verschiedensten Bereichen, z. B. bei der Planung und dem Betrieb technischer Anlagen oder bei betriebswirtschaftlichen Entscheidungen. Typisch ist, dass viele der Optimierungsziele miteinander unvereinbar sind, sodass eine Lösung nicht bez. aller Optimierungsziele optimal sein kann. Ferner sind viele der Ziele unvergleichbar, d. h., sie können nicht gegeneinander aufgewogen werden.

9.1 Szenario und grundlegende Definitionen

Im Szenario der mehrkriteriellen Optimierung sind m unvergleichbare Optimierungsziele (im Folgenden kurz *Ziele* genannt) einer Lösung gleichzeitig zu maximieren oder zu minimieren. Ohne Beschränkung der Allgemeinheit gehen wir hier stets davon aus, dass alle Ziele zu maximieren sind. Gewöhnlich stehen die Ziele im Konflikt, sodass ein einzelnes Ziel nur auf Kosten der anderen Ziele maximiert werden kann. Die Güte einer Lösung wird mithilfe der Zielfunktion (engl. *objective function*) $f\colon S \to \mathbb{R}^m$ bewertet, die jeder Lösung aus dem Suchraum S insgesamt m Zielwerte zuordnet, die es zu maximieren gilt. Der Zielraum (engl. *objective (function) space*), der Bildbereich von f, ist eingebettet in die Menge der m-dimensionalen reellwertigen Vektoren $\mathbb{R}^m$. Mehrkriterielle Optimierungsprobleme werden in der Literatur häufig wie folgt notiert:

$$\begin{aligned} \text{„Maximiere“} \quad & f(s), \\ \text{sodass} \quad & g_i(s) \;=\; 0 \quad \text{für } i \in \{1,\ldots,p\} \\ \text{und} \quad & u_j(s) \geq\; 0 \quad \text{für } j \in \{1,\ldots,q\}. \end{aligned}$$

Dabei ist $f(s) = (f_1(s),\ldots,f_m(s))$ die mehrkriterielle Zielfunktion und $g_i(s)$ und $u_j(s)$ sind Nebenbedingungen, die an zulässige Lösungen gestellt werden. Die Behandlung von Nebenbedingungen soll hier nicht thematisiert werden. Wir werden nur Probleme ohne Nebenbedingungen betrachten, sodass stets alle Punkte im Suchraum S zulässige Lösungen darstellen.

Offenbar ist ein Vektor $x = (x_1,\ldots,x_m) \in \mathbb{R}^m$ nicht besser als ein zweiter Vektor $y = (y_1,\ldots,y_m)$, wenn jede Komponente von x kleiner oder gleich der zugehörigen Komponente von y ist. Wir notieren diesen Fall als $x \preceq y$ mit der formalen Definition

$$x \preceq y \;\Leftrightarrow\; \forall i \in \{1,\ldots,m\}\colon x_i \leq y_i.$$

Es ist jedoch nicht in jedem Fall möglich, einem von zwei verschiedenen Vektoren den Vorzug zu geben. Folglich können wir auch nicht immer entscheiden, dass eine von zwei Lösungen aus dem Suchraum mit Zielwertvektoren x und y die bessere ist. In dieser Situation ist das Ziel der Optimierung, solche Lösungen im Suchraum zu finden, für die eine Verbesserung eines Ziels nur noch möglich ist, wenn sich der Wert mindestens eines anderen Ziels dadurch verschlechtert.

Um grundlegende Begriffe einzuführen, ist es hilfreich, mehrkriterielle Optimierungsprobleme als Spezialfall von Optimierungsproblemen mit einem (nur) partiell geordneten Zielraum anzusehen. Die folgenden Definitionen lehnen sich z. T. an Rudolph (1998a, 2001a) an.

Definition 9.1 (Präferenzordnung, partielle Ordnung). *Es sei F eine Menge und $\preceq$ eine binäre Relation in F. Die Relation $\preceq$ heißt* Präferenzordnung[1]*, wenn sie*

reflexiv $(\forall\, x \in F\colon x \preceq x)$ *und*

transitiv $(\forall\, x,y,z \in F\colon x \preceq y \wedge y \preceq z \;\Rightarrow\; x \preceq z)$ *ist.*

Die Relation $\preceq$ heißt partielle Ordnung *und das Paar $(F,\preceq)$ heißt* partiell geordnete Menge, *falls $\preceq$ eine*

antisymmetrische $(\forall\, x,y \in F\colon x \preceq y \wedge y \preceq x \;\Rightarrow\; x = y)$

Präferenzordnung ist.

Die oben beschriebene Relation $\preceq$ ist offenbar eine partielle Ordnung für $\mathbb{R}^m$.

[1] Anstelle von „Präferenzordnung“ ist auch „Quasiordnung“ üblich. Jedoch ist die Definition einer Quasiordnung bez. der Eigenschaften Reflexivität und Irreflexivität schwankend.

Definition 9.2 (vergleichbar, unvergleichbar). *Zwei Elemente $x, y \in F$ heißen* vergleichbar *bez. einer Präferenzordnung $\preceq$ in F, wenn $x \preceq y$ oder $y \preceq x$ gilt; insbesondere ist x mit x vergleichbar. Andernfalls heißen verschiedene Elemente x und y* unvergleichbar *(in Zeichen $x \parallel y$).*

Die folgenden Sprechweisen sind für vergleichbare Elemente üblich.

Definition 9.3 (dominieren). *Falls $x \preceq y$ und $x \neq y$ gelten, sagen wir „y* dominiert *x“. Wir notieren dies als $x \prec y$. Falls $x \preceq y$ gilt, sagen wir „y* dominiert *x* schwach“ *(und schließen damit nicht aus, dass y sogar x dominiert).*

Im Zielraum F interessieren uns also Elemente, die nicht von anderen Elementen dominiert werden.

Definition 9.4 (maximale Elemente). *Ein Element $x^* \in F$ heißt* maximales Element *einer partiell geordneten Menge $(F, \preceq)$, wenn es kein Element $x \in F$ gibt, das x^* dominiert. Mit $M(F, \preceq)$ wird die Menge aller maximalen Elemente von $(F, \preceq)$ bezeichnet.*

Man sagt, dass $M(F, \preceq)$ vollständig ist, wenn es für jedes $x \in F$ ein maximales Element $x^* \in M(F, \preceq)$ gibt, sodass $x \preceq x^*$ gilt. Wenn F eine endliche Menge ist, ist $M(F, \preceq)$ stets vollständig (und offenbar auch endlich). Wir werden später nur mit endlichen Zielräumen umgehen, sodass wir es immer mit vollständigen und endlichen maximalen Mengen zu tun haben werden.

Die maximalen Elemente des partiell geordneten Zielraums selbst sind jedoch nicht das Ziel der Suche. Wir suchen nach den „besten Lösungen“. Das Ziel der Suche sind die Urbilder der maximalen Elemente des Zielraums im Suchraum. Falls f keine Bijektion ist, kann es z. B. sein, dass es diese Urbilder nicht gibt oder dass die Urbildmenge viel größer ist als $M(F, \preceq)$.

Die folgende Definition versucht, Eigenschaften der partiellen Ordnung im Zielraum auf den Suchraum zu übertragen.

Definition 9.5 ($\preceq_f$). *Es sei S der Suchraum, $(F, \preceq)$ ein partiell geordneter Zielraum und $f\colon S \to F$ eine Abbildung. Dann definieren*

$$
\begin{aligned}
x \prec_f y \ &:\Leftrightarrow \ f(x) \prec f(y), \\
x =_f y \ &:\Leftrightarrow \ f(x) = f(y) \qquad \textit{und} \\
x \preceq_f y \ &:\Leftrightarrow \ x \prec_f y \ \vee \ x =_f y
\end{aligned}
$$

eine Präferenzordnung $\preceq_f$ in S.

Mithilfe dieser Definition sind Suchpunkte nun vergleichbar. Wir können jetzt sagen, dass ein Suchpunkt y einen zweiten Suchpunkt x dominiert oder schwach dominiert, wenn $x \prec_f y$ bzw. $x \preceq_f y$ gilt. Die Präferenzrelation $\preceq_f$ ist jedoch i. Allg.

keine partielle Ordnung für den Suchraum, da die Antisymmetrie verloren gehen kann: Wenn zwei verschiedene Suchpunkte x und y auf denselben Zielwertvektor abgebildet werden, gilt zwar $x \preceq_f y$ und $y \preceq_f x$, woraus $x =_f y$ folgt, es gilt jedoch nicht $x = y$.

Definition 9.6 (Paretofront, Paretomenge). *Es sei S ein endlicher Suchraum und $F := f(S) \subseteq \mathbb{R}^m$ der Zielraum. Die partielle Ordnung $\preceq$ in F sei durch*

$$(x_1, \ldots, x_m) \ \preceq \ (y_1, \ldots, y_m) \ \Leftrightarrow \ \forall i \in \{1, \ldots, m\}\colon x_i \leq y_i \qquad (*)$$

definiert. Die Menge der maximalen Elemente $F^* := M(F, \preceq)$ im Zielraum heißt Paretofront. *Ein Element $s \in S$ heißt* paretooptimal, *wenn $f(s)$ in der Paretofront F^* liegt. Die Menge aus allen paretooptimalen Elementen $S^* := f^{-1}(F^*)$ heißt* Paretomenge.

Es ist zweckmäßig, den Zielraum auf das Bild der Zielfunktion zu begrenzen, da nur Lösungen im Suchraum S gefunden werden können, die zu Punkten in $f(S)$ gehören. Dann haben wir es mit einer surjektiven Abbildung f zu tun. So ist gewährleistet, dass das Urbild der Paretofront nicht leer ist. Solange der Suchraum endlich ist, ist der Zielraum dann ebenfalls endlich und das Problem nicht vollständiger Mengen maximaler Elemente im Zielraum tritt nicht auf. Daher sei im Folgenden stets $F := f(S)$.

Der Begriff der Paretooptimalität ist streng genommen dem Fall $F \subseteq \mathbb{R}^m$ mit der Relation $\preceq$ aus Definition 9.6 vorbehalten. Gelegentlich werden wir auch von Paretooptimalität sprechen, wenn F ein beliebiger, partiell geordneter Zielraum ist. Es ist klar, dass die Paretofront dann die Menge maximaler Elemente im Zielraum meint und die Paretomenge das Urbild dieser Menge.

Was ist nun das Ziel der Optimierung, was soll berechnet werden? Es soll eine noch genau zu charakterisierende Teilmenge der Paretomenge S^* berechnet werden. Wir werden dabei stets mit endlichen Suchräumen S arbeiten, sodass es im Prinzip möglich ist, die darin enthaltene Paretomenge vollständig aufzuzählen. Die gesamte Paretomenge zu berechnen ist dennoch nicht immer sinnvoll. Es ist möglich, dass die Paretofront klein ist (z. B. polynomiell in der Suchraumdimension n beschränkt), die Paretomenge jedoch groß ist (z. B. exponentiell in n). Ein extremes Beispiel ist ein Problem, bei dem alle Punkte im Suchraum auf denselben Zielwertvektor abgebildet werden, z. B. $f(s) = (0, \ldots, 0)$. Dann sind sämtliche Punkte im Suchraum paretooptimal. Wenn es also Lösungen $s_1, \ldots, s_k \in S$ mit identischen Zielwertvektoren $f(s_1) = \cdots = f(s_k)$ gibt, dann soll nur *eine* Lösung aus $\{s_1, \ldots, s_k\}$ in der berechneten Lösungsmenge enthalten sein. Das führt zu folgender Definition.

Definition 9.7 (Repräsentantenmenge). *Eine* Repräsentantenmenge *für eine Menge $A' \subseteq F$ ist eine Menge $A \subseteq f^{-1}(A')$, sodass $f(A) = A'$ und $|A| = |A'|$ gilt.*

Jetzt können wir die Zielsetzung der Optimierung von Funktionen mit partiell geordneten Zielräumen definieren, wie sie im Folgenden verstanden wird: Es soll eine Repräsentantenmenge für die Paretofront gefunden werden. Diese Definition der Zielsetzung erzeugt keine Widersprüche zu den in der Literatur vorhandenen Laufzeitanalysen für mehrkriterielle evolutionäre Algorithmen, die auf dem diskreten, endlichen Suchraum $X^n := \{0,1\}^n$ arbeiten (siehe Abschnitt 9.3). Dieser Suchraum eignet sich für kombinatorische Optimierungsprobleme. Für kontinuierliche Suchräume ist die hier gesteckte Zielsetzung in der Regel nicht vernünftig, da der Zielraum und die Paretofront dann meistens überabzählbare Mengen sind. Zumindest explizite Darstellungen von Repräsentantenmengen überabzählbarer Paretofronten verbieten sich.

Auch für den Fall endlicher Suchräume müssen wir uns darüber im Klaren sein, dass eine Repräsentantenmenge der Paretofront den gesamten Suchraum umfassen kann. Dazu betrachten wir als Beispiel die folgende Zielfunktion $f\colon \{0,1\}^n \to \mathbb{R}\times\mathbb{R}$. Es sei $\mathrm{BV}(x) := \sum_{i=0}^{n-1} 2^i \cdot x_i$ die Funktion, die den Bitstring x als Binärzahl (engl. *binary value*) interpretiert, und $f(x) := (\mathrm{BV}(x), -\mathrm{BV}(x))$. Beide Ziele sind zu maximieren (oder beide zu minimieren). Wenn einer der beiden Zielwerte wächst, muss zwangsläufig der andere Zielwert sinken. Da jeder Suchpunkt einen anderen Zielwertvektor hat, der nicht dominiert wird, gibt es für jeden Punkt der Paretofront nur einen möglichen Repräsentanten, sodass S die einzige Repräsentantenmenge ist. Es ist klar, dass jeder Algorithmus, der die Repräsentantenmenge S aufzählt, exponentielle Laufzeit in n haben muss.

Letztendlich geht es bei mehrkriteriellen Optimierungsproblemen darum, eine Lösung oder wenige Lösungen aus allen paretooptimalen Lösungen auszuwählen, um diese zu realisieren. Dazu kann es sinnvoll sein, das Gebiet zulässiger Lösungen so einzuschränken, dass nur der wirklich interessierende Teil der Paretofront von einer Lösungsmenge abgedeckt wird. So versucht man zu vermeiden, unnötig große Lösungsmengen zu berechnen. Es ist jedoch nicht leicht zu entscheiden, welches Gebiet der Paretofront interessant ist, wenn im Vorhinein nicht bekannt ist, welche Zielwerte überhaupt erreichbar sind und welche Trade-offs man in Kauf nehmen muss. Das Problem der Auswahl einer Lösung behandeln wir hier nicht, bemerken aber, dass die Auswahl leichter fällt, wenn die Paretofront (zumindest approximativ) bekannt ist.

Da es bei mehrkriteriellen Optimierungsproblemen fast immer um die Berechnung einer Lösungsmenge anstatt nur einer Lösung geht, sind populationsbasierte evolutionäre Algorithmen geeignete randomisierte Heuristiken. Der Grundgedanke solcher mehrkriterieller evolutionärer Algorithmen (engl. *multi-objective evolutionary algorithms, MOEAs*) ist, eine Population P von Suchpunkten (Lösungen) zu verwalten, deren Bild im Zielraum die Paretofront im Laufe der Zeit immer besser approximiert. Den Begriff „approximieren“ kann man auf unterschiedliche Weise konkretisieren. Wir wollen hier das Konzept der $(1+\varepsilon)$-Dominanz für $\varepsilon > 0$ betrachten (vgl. Laumanns, Thiele, Deb und Zitzler, 2002a). Man definiert, dass $x \preceq^{(1+\varepsilon)} y$

gilt, wenn $x_i \leq (1+\varepsilon) \cdot y_i$ für alle $i \in \{1, \dots, m\}$ gilt. Das bedeutet, der Punkt $y \in \mathbb{R}^m$ ist eine $(1+\varepsilon)$-Approximation von $x \in \mathbb{R}^m$ bez. sämtlicher Ziele. Dann wird das Bild der Population $f(P) \subseteq \mathbb{R}^m$ eine $(1+\varepsilon)$-Approximationsmenge der Paretofront genannt, wenn für jeden Punkt x der Paretofront in $f(P)$ mindestens ein Punkt y enthalten ist, der x $(1+\varepsilon)$-dominiert. Zusätzlich kann man von solchen $(1+\varepsilon)$-Approximationsmengen noch verlangen, dass sie keine Punkte enthalten, die sich gegenseitig bez. $\preceq$ dominieren. Dominierte Punkte können entfernt werden, ohne dass sich die Approximationsgüte verschlechtert. Da $(1+\varepsilon)$-Dominanz nur in dem speziellen Zielraum $\mathbb{R}^m$ definiert ist und wir im Folgenden ganz ohne Approximationsgüten auskommen werden, definieren wir den Begriff der Approximationsmenge für beliebige Zielräume wie folgt.

Definition 9.8 (Approximationsmenge). *Eine Menge $A' \subseteq F$ heißt* Approximationsmenge, *wenn kein Element aus A' von einem anderen Element aus A' bez. $\preceq$ schwach dominiert wird, d. h., verschiedene Elemente von A' sind unvergleichbar.*

Die Menge der Suchpunkte, welche die im nächsten Abschnitt vorgestellte Suchheuristik verwaltet, wird stets so beschaffen sein, dass ihr Bild im Zielraum eine Approximationsmenge ist.

9.2 Die Suchheuristik SEMO für partiell geordnete Zielräume

Der Name der randomisierten Suchheuristik SEMO (Simple Evolutionary Multi-objective Optimizer) geht auf die Arbeit von Laumanns, Thiele, Zitzler, Welzl und Deb (2002b) zurück, in der die Heuristik als grundlegender evolutionärer Algorithmus für mehrkriterielle Optimierungsprobleme eingeführt wird. Da über den Zielraum F nichts weiter angenommen wird, als dass auf F eine partielle Ordnung definiert ist, ist das Anwendungsgebiet des SEMO nicht allein auf die mehrkriterielle Optimierung beschränkt. Er kann als Ausprägung des Basisalgorithmus PR aus der Arbeit von Rudolph und Agapie (2000) aufgefasst werden, der in dem allgemeineren Szenario der Optimierung in partiell geordneten Mengen arbeitet. Der Einfachheit halber werden wir im Folgenden trotzdem häufig die Begriffe der mehrkriteriellen Optimierung benutzen, wenn wir über den SEMO sprechen.

Rudolph (2001b) stellt weitere Szenarien vor, in denen die Suche nach maximalen Elementen in partiell geordneten Zielräumen Anwendungen findet. Dazu gehören intervallwertige Fitnessfunktionen, die anstatt eines reellwertigen Fitnesswertes ein Intervall liefern. Auf der Menge der Intervalle kann dann mittels $([r_1, r_2] \preceq [s_1, s_2]) :\Leftrightarrow ((r_2 \leq s_1) \vee ([r_1, r_2] = [s_1, s_2]))$ eine partielle Ordnung definiert werden, die Intervalle mit nicht leerem Schnitt als unvergleichbar einstuft, es sei denn, die Intervalle sind gleich. Eine ähnliche Anwendung sind verrauschte

Fitnessfunktionen. Dazu betrachtet man zu einer Fitnessfunktion $f\colon S \to \mathbb{R}$ die verrauschte Fitnessfunktion $\tilde{f}(s) := f(s)+Z$, die durch Auswerten von $f(s)$ unter Rauschen entsteht. Der Fehler beim Auswerten („Messen“) von f wird durch den zufälligen additiven Fehlerterm Z modelliert, der eine Zufallsvariable mit Wertebereich $[-a, a]$ ist. Aufgrund so gestörter Auswertungen zweier Suchpunkte s und s' kann nur mit Sicherheit gesagt werden, dass $f(s) \leq f(s')$ gilt, wenn $\tilde{f}(s) + a \leq \tilde{f}(s') - a$ gilt. Dies führt zu der partiellen Ordnung $\tilde{f}(s) \preceq \tilde{f}(s') :\Leftrightarrow \tilde{f}(s) + a \leq \tilde{f}(s') - a$ im Zielraum.

Der SEMO ist ein populationsbasierter evolutionärer Algorithmus, dessen einziger Suchoperator die Mutation ist. Die Grundidee des Algorithmus ist, dass die Population P (aus Suchpunkten) zu jeder Zeit eine Repräsentantenmenge ihres Bildes $f(P)$ ist und $f(P)$ eine Approximationsmenge ist. Die Hoffnung ist, dass $f(P)$ im Laufe der Zeit immer besser die Paretofront F^* approximiert. Die Größe der Population P kann sich in jedem Schritt ändern und ist i. Allg. nur durch die Größe des Suchraums S zu beschränken.

Definition 9.9 (SEMO). *Der SEMO (Simple Evolutionary Multi-objective Optimizer) läuft für eine Funktion $f\colon S \to F$ mit einem partiell geordneten Zielraum $(F, \preceq)$ wie folgt ab.*

1. *Setze $t := 1$.*
   Wähle $s_1 \in S$ gemäß einer Wahrscheinlichkeitsverteilung.
   Bestimme $f(s_1)$.
   Setze $P_1 := \{s_1\}$. *(Initialisierung)*

2. *Wiederhole:*

   Setze $t := t + 1$.

   Wähle $s \in P_{t-1}$ zufällig gleichverteilt.
   Erzeuge s_t aus s durch Mutation.
   Bestimme $f(s_t)$. *(Suche)*

   Falls für alle $s' \in P_{t-1}$ die Bedingung $s_t \not\prec_f s'$ erfüllt ist,
   dann setze $P_t := \{s' \in P_{t-1} \mid s' \not\preceq_f s_t\} \cup \{s_t\}$,
   sonst setze $P_t := P_{t-1}$. *(Selektion)*

Zu jedem Suchpunkt $s \in P_t$ ist zusätzlich $f(s)$ gespeichert, damit Suchpunkte der Population bei der nächsten Selektion nicht erneut ausgewertet zu werden brauchen.

Nach der Initialisierung ist $P_1 = \{s_1\}$ offenbar eine Repräsentantenmenge für die Approximationsmenge $\{f(s_1)\}$. Im t-ten Schritt wählt der SEMO einen zufälligen Suchpunkt aus der Population aus und erzeugt daraus den Mutanten s_t. Der neue Punkt s_t wird in die Population aufgenommen, wenn es darin kein Individuum gibt, das den Mutanten dominiert. In dem Moment, in dem s_t der Population hinzugefügt wird, werden gleichzeitig alle Elemente aus der Population entfernt, die von s_t

schwach dominiert werden. Daher sind alle Punkte in der neuen Population bez. $\preceq_f$ wieder unvergleichbar. Also ist P_t zu jedem Zeitpunkt t eine Repräsentantenmenge für $f(P_t)$ und $f(P_t)$ ist eine Approximationsmenge.

In Definition 9.9 sind der Mutationsoperator und die Wahrscheinlichkeitsverteilung, gemäß der der erste Suchpunkt ausgewählt wird, nicht näher spezifiziert. Wir leiten nun zwei Varianten des SEMO für die Suche im diskreten Suchraum $X^n := \{0,1\}^n$ ab. Immer wenn wir über Varianten für den n-dimensionalen booleschen Suchraum sprechen, bezeichnen wir den Suchraum nicht wie im allgemeinen Fall mit S, sondern mit X^n.

Definition 9.10 (lokaler und globaler SEMO). *Der lokal suchende SEMO (der* lokale SEMO*) und der global suchende SEMO (der* globale SEMO*) zur Optimierung von Funktionen $X^n \to F$ sind Ausprägungen des SEMO.*

Der Initialisierungsschritt wählt einen Suchpunkt zufällig gleichverteilt aus X^n aus.

Angewendet auf einen Suchpunkt $x \in X^n$ kippt der Mutationsoperator des lokalen SEMO ein *gleichverteilt ausgewähltes Bit von x. Der globale SEMO kippt jedes Bit von x unabhängig mit einer Wahrscheinlichkeit von $1/n$.*

Damit erhalten wir Gegenstücke zur randomisierten lokalen Suche (RLS), wenn deren Suchoperator ebenfalls nur ein Bit kippt, und zum (1+1)-EA für den mehrkriteriellen Fall. In der Tat wird der lokale SEMO zur RLS und der globale SEMO zum (1+1)-EA, wenn wir sie auf einkriterielle Optimierungsprobleme $f\colon \{0,1\}^n \to \mathbb{R}$ anwenden. In diesem Fall kann die Population nicht auf mehr als einen Punkt anwachsen, da je zwei Suchpunkte bez. $\preceq_f$ vergleichbar sind. Der Mutant x_t ersetzt den einzigen Punkt in der Population, wenn x_t nicht schlechter ist als dieser.

Die Definition des SEMO in seiner ursprünglichen Fassung in Laumanns, Thiele, Zitzler, Welzl und Deb (2002b) und Laumanns, Thiele und Zitzler (2004c) entspricht bis auf ein Detail der Heuristik, die wir hier den lokalen SEMO genannt haben. Der manchmal nicht unbedeutende Unterschied besteht in der Selektion. Ein wichtiger Aspekt unserer Analysen der randomisierten lokalen Suche und des (1+1)-EA für das Matchingproblem ist die Fähigkeit dieser Heuristiken, Plateaus aus Punkten gleicher Fitness abzusuchen. Das erfordert, dass neue Suchpunkte auch dann akzeptiert werden, wenn sie nur gleich gut wie bereits zuvor gefundene Punkte aus der Population sind. Um auch in dieser Hinsicht die Analogie zum (1+1)-EA und zur RLS aufrechtzuerhalten, wurde der Selektionsoperator des SEMO in Definition 9.9 bereits entsprechend angepasst. Um die ursprüngliche Fassung zu erhalten, ist bei der Selektion $\not\prec$ durch $\not\preceq$ zu ersetzen. In allen hier vorgestellten Analysen des SEMO bleibt dieser Unterschied jedoch ohne Auswirkung.

In Abschnitt 9.1 haben wir uns die Berechnung einer Repräsentantenmenge der Paretofront zum Ziel der Optimierung gesetzt. Mit Blick auf Definition 3.1 ist der Gedanke nun, die Optimierzeit einer randomisierten Suchheuristik für ein mehrkriterielles Black-Box-Problem als die Zahl der Zielfunktionsauswertungen zu definieren, bis zum ersten Mal für jeden Punkt der Paretofront ein Urbildpunkt ausgewertet wurde.

Definition 9.11 (Optimierzeit des SEMO). *Die Optimierzeit des SEMO für eine Funktion $f: S \to F$ mit einem partiell geordneten Zielraum F ist die Zufallsvariable $T_f \in \mathbb{N}$, welche die kleinste Zahl t angibt, sodass am Ende der Schleife des SEMO $f(P_t) = F^*$ gilt. Die erwartete Optimierzeit ist der Erwartungswert $E(T_f)$.*

Nachdem die Population eine Repräsentantenmenge der Paretofront geworden ist, erhält der SEMO diese Eigenschaft aufrecht, da neue Punkte die Individuen der Population nicht dominieren können. Der SEMO kann also bedenkenlos über seine Optimierzeit hinaus weiterlaufen, ohne die Lösung zu zerstören. Es kann lediglich passieren, dass Repräsentanten gegen gleich gute Punkte aus dem Suchraum eingetauscht werden.

Die Annahme hinter der Definition der Optimierzeit ist hier wieder, dass Auswertungen der Black Box teure Operationen sind, deren Kosten die Kosten aller weiteren Operationen in der Schleife dominieren. Diese Annahme ist hier nicht mehr so leicht zu rechtfertigen. Sie mag sogar ungerechtfertigt erscheinen, da die Population zur Größe von $\Omega(|F|)$ anwachsen kann. Sie kann also exponentiell groß in der Suchraumdimension n werden. (Es ist nicht klar, ob der Selektionsschritt im Worst Case effizienter als in Zeit $\Theta(|P_t|)$ realisiert werden kann.) Definition 9.11 ist jedoch im Einklang mit den in der Literatur vorhandenen Laufzeitanalysen, sodass die im Folgenden vorgestellten Ergebnisse damit vergleichbar sind.

So wie wir es schon für den (1+1)-EA und die RLS gewohnt sind, bezeichnen wir einen Durchlauf der Schleife des SEMO als einen *Schritt*. Der Initialisierungsschritt ist der erste Schritt. Immer wenn der SEMO das Ende des Schleifenrumpfs erreicht hat, ist also der Wert der Variablen t gleich der Zahl der ausgeführten Schritte und gleich der Zahl der Zielfunktionsauswertungen.

Einen Suchpunkt x in der Population bezeichnen wir im Folgenden manchmal auch als *Individuum*, wenn die Tatsache zum Ausdruck kommen soll, dass x zur Population gehört. Formal soll hier jedoch keine Unterscheidung zwischen den Begriffen *Suchpunkt* und *Individuum* eingeführt werden.

9.3 Literaturüberblick über Laufzeitanalysen für MOEAs

Das Wissen über den Einsatz und Entwurf von MOEAs (Multi-objective Evolutionary Algorithms) als Vertreter randomisierter Suchheuristiken für mehrkriterielle Optimierungsprobleme ist in den letzten Jahren immens angewachsen. Die verschiedensten Varianten von MOEAs werden erfolgreich für kombinatorische Optimierungsprobleme, besonders auch für Optimierungsprobleme mit reellwertigen

Suchräumen, angewendet. Ihr Erfolg und das zunehmende Interesse an diesen Heuristiken wird z. T. damit begründet, dass sie im Gegensatz zu klassischen Verfahren in der Lage sind, mit nur einem Lauf eine Menge von paretooptimalen oder zumindest nahe paretooptimalen Suchpunkten zu finden. Ausführliche Darstellungen des Themas der mehrkriteriellen Optimierung mit MOEAs, die verschiedenen Varianten von MOEAs und Anwendungsbeispiele findet man in den Monographien von Deb (2001) sowie Coello Coello, Van Veldhuizen und Lamont (2002) .

Die theoretische Analyse von MOEAs hat sich zunächst auf das Konvergenzverhalten konzentriert. Es wurde die Frage untersucht, unter welchen Bedingungen MOEAs überhaupt das gesteckte Optimierungsziel erreichen, wenn die Laufzeit gegen unendlich strebt. Hier sind besonders die Arbeiten von Rudolph (1998a,b, 2001a) und Rudolph und Agapie (2000) zu nennen. Ehrgott (2000) beschäftigt sich dagegen mit der Komplexität mehrkriterieller Optimierungsprobleme.

Die Analyse der Optimierzeit von MOEAs ist ein Thema, das erst seit jüngster Zeit bearbeitet wird. In der Regel werden kombinatorische Optimierungsprobleme betrachtet. Als Vorbild dient die schon viel weiter entwickelte Analyse grundlegender einkriterieller evolutionärer Algorithmen wie die des (1+1)-EA und der randomisierten lokalen Suche. Um scharfe Laufzeitschranken beweisen zu können, ist es notwendig, Eigenschaften des Optimierungsproblems in die Analyse einfließen zu lassen. Wenn man die Optimierzeit für zu große (d. h. zu allgemein gefasste) Problemklassen analysiert, erhält man häufig Schranken, die für viele der Probleme in der Klasse wenig Aussagekraft besitzen. Daher müssen sich Analysen mit aussagekräftigen Ergebnissen meist auf eng umrissene Problemklassen konzentrieren, häufig nur auf *ein* Problem.

Die vermutlich erste Laufzeitanalyse für ein mehrkriterielles Problem ist Scharnow, Tinnefeld und Wegener (2002) für eine mehrkriterielle Variante des (1+1)-EA und eine mehrkriterielle Formulierung des Kürzeste-Wege-Problems (single source shortest path) gelungen. Für Graphen mit n Knoten ist die erwartete Optimierzeit $O(n^2)$. Dieses Problem ist jedoch kein „echt“ mehrkriterielles Problem, da die Optimierungsziele nicht im Konflikt miteinander stehen, sodass es nur eine einzige paretooptimale Lösung gibt. Das Bemerkenswerte an dem Resultat ist, dass die vergleichbare einkriterielle Formulierung des Problems für einige Graphen, nämlich Bäume, zu einem Nadel-im-Heuhaufen-Szenario führt, das für randomisierte Suchheuristiken nur sehr schwer zu bewältigen ist.

Die Arbeit von Laumanns, Thiele, Zitzler, Welzl und Deb (2002b), in der der SEMO eingeführt wird, ist vermutlich die erste Arbeit, in der die Optimierzeit eines MOEAs für ein wirklich mehrkriterielles Problem analysiert wird. Die Arbeit untersucht die Optimierzeit des lokalen SEMO und eine Variante des lokalen SEMO für das Problem LOTZ (Leading Ones Trailing Zeroes). In der eigenen Arbeit Giel (2003) wird die (erwartete) Optimierzeit des globalen SEMO analysiert und insbesondere der Worst Case untersucht. Die Ergebnisse werden hier in Kapitel 10, 11 und 12 vorgestellt. Die Arbeit von Thierens (2003) versucht eine Analyse des lokalen und globalen SEMO für das Problem MOCO (Multi-objective Counting Ones). Kapitel 13 dieser Dissertation setzt sich eingehend mit diesem Problem auseinander. Die Arbeit von Laumanns, Thiele und Zitzler (2004c) analysiert weitere Varianten des SEMO für das LOTZ-Problem und das Problem COCZ (Count Ones Count Zeroes). Die Beweistechnik der fitnessbasierten Partitionierung des Suchraums für einkriterielle randomisierte Suchheuristiken wird hier auf den mehrkriteriellen Fall erweitert. Damit gelingt es in vielen Situationen, die Zeit abzuschätzen, bis das erste paretooptimale Individuum erzeugt wird. Darüber hinaus wird eine Multistartvariante des (1+1)-EA für mehrkriterielle Probleme untersucht.

Laumanns, Thiele und Zitzler (2004b) analysieren den globalen SEMO für eine spezielle Instanz des Rucksackproblems. In Laumanns, Thiele und Zitzler (2004a) wird eine Heuristik untersucht, die auf der sogenannten *ε-constrained method* beruht. Kumar und Banerjee (2005) analysieren REMO, eine weitere Variante des lokalen SEMO, unter anderem auch für sehr spezielle Instanzen des Rucksackproblems. Neumann (2004) untersucht die Optimierzeit des globalen SEMO für das k-kriterielle Spannbaumproblem für k Gewichte pro Kante eines Graphen. Neumann und Wegener (2005) vergleichen die randomisierte lokale Suche und den (1+1)-EA mit dem lokalen und globalen SEMO für das (einkriterielle) Spannbaumproblem. Die RLS und der (1+1)-EA arbeiten dabei mit einer einkriteriellen Formulierung, der globale und lokale SEMO hingegen mit einer zweikriteriellen Formulierung des (einkriteriellen) Spannbaumproblems. Die beiden Optimierungsziele sind die Zahl der Zusammenhangskomponenten und das Gesamtgewicht aller ausgewählten Kanten einer Lösung. Sie zeigen, dass der globale und lokale SEMO in dichten Graphen eine kleinere erwartete Optimierzeit haben als die randomisierte lokale Suche und der (1+1)-EA.

10 Die erwartete Optimierzeit im Worst Case

In diesem Kapitel untersuchen wir die erwartete Optimierzeit des globalen SEMO im Worst Case. Das bedeutet, wir untersuchen die erwartete Laufzeit für ein Problem, das dem globalen SEMO die größtmögliche erwartete Optimierzeit aufzwingt.

Für den lokalen SEMO kann man leicht einsehen, dass dieser i. Allg. keine endliche erwartete Optimierzeit haben kann, d. h., der Erwartungswert $E(T_f)$ existiert für viele Funktionen f nicht. Sofern es im Suchraum eine Menge aus nicht paretooptimalen Suchpunkten $\widehat{X} \subseteq X$ gibt, sodass alle Hammingnachbarn dieser Punkte, die nicht selbst zu $\widehat{X}$ gehören, von den Punkten in $\widehat{X}$ dominiert werden, kann die Population P des lokalen SEMO die Teilmenge $\widehat{X}$ nicht mehr verlassen, sobald sie einmal vollständig in $\widehat{X}$ enthalten ist.

Solche Funktionen f kann man sich leicht ausdenken. Dazu betrachten wir das folgende (einkriterielle) Problem. Es sei $|x|$ die Zahl der 1-Bits von $x \in X^n$. Für $|x| \leq n-2$ sei $f(x) = 1$, für $|x| = n-1$ sei $f(x) = 0$ und für $|x| = n$ sei $f(x) = 2$. Offenbar ist 1^n, der Suchpunkt mit n 1-Bits, der einzige Punkt in der Paretomenge X^*. Angenommen, die Population des lokalen SEMO liegt in $\widehat{X} := \{x \in X^n \mid |x| \leq n-2\}$. Dann kann sie $\widehat{X}$ nicht verlassen, denn der einzige Punkt, der einen Punkt aus $\widehat{X}$ mindestens schwach dominiert, nämlich 1^n, hält mindestens einen Hammingabstand von 2 zu jedem Punkt in $\widehat{X}$ ein. Aus Chernoffschranken folgt, dass der initiale Suchpunkt x_1 mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ in $\widehat{X}$ liegt. Daher gilt $\mathrm{Prob}(T_f \geq t) = 1 - \mathrm{e}^{-\Omega(n)}$ für alle $t \geq 1$. Die Optimierzeit ist also mit überwältigender Wahrscheinlichkeit beliebig groß. Aus $E(T_f) = \sum_{t\geq 1} \mathrm{Prob}(T_f \geq t)$ folgt nun, dass der Erwartungswert $E(T_f)$ nicht endlich ist. Auch Multistartvarianten des lokalen SEMO sind erfolglos, da offenbar auch polynomiell viele Wiederholungen mit überwältigender Wahrscheinlichkeit keinmal das Paretooptimum finden (vgl. Abschnitt 7.1).

Wir schließen daraus, dass lokale Suchstrategien wie der lokale SEMO nur auf Probleme angewendet werden sollten, für die die Wahrscheinlichkeit gering ist, in lokalen Optima stecken zu bleiben. Es ist also Vorwissen über das Problem erforderlich, das häufig in den Anwendungsszenarien randomisierter Suchheuristiken gerade *nicht* verfügbar ist. Das Ziel in diesem Teil der Dissertation ist daher, zur Analyse des global suchenden SEMO beizutragen. Jedoch ist die Analyse lokal suchender randomisierter Suchheuristiken häufig der Ausgangspunkt für die meist kompliziertere Analyse global suchender randomisierter Suchheuristiken. In Kapitel 13 werden wir Beweisideen zuerst anhand des lokalen SEMO vorführen, um so die schwierigere Analyse des globalen SEMO vorzubereiten.

Es ist bekannt, dass die erwartete Optimierzeit des (1+1)-EA im Worst Case $\Theta(n^n)$ ist, wenn der Startpunkt der Suche wie beim globalen SEMO gleichverteilt gewählt wird (Droste, Jansen und Wegener, 2002). Da der globale SEMO für einkriterielle Probleme zum (1+1)-EA wird, gilt die untere Schranke von $\Omega(n^n)$ auch für den globalen SEMO. Der Beweis der unteren Schranke in Droste, Jansen und Wegener (2002) mithilfe der Funktionen DISTANCE oder TRAP ist jedoch vergleichsweise kompliziert. In einem Szenario mit zwei Optimierungszielen wird ein entsprechender Beweis für den globalen SEMO nahezu trivial.

Theorem 10.1. *Es gibt eine Funktion $f\colon X^n \to \mathbb{R}^2$, für die die erwartete Optimierzeit des globalen SEMO mindestens n^n ist. Die Optimierzeit ist mindestens $n^{n/2}$ mit einer überwältigenden Wahrscheinlichkeit von $1 - n^{-\Omega(n)}$.*

Beweis. Es sei $f(0^n) = (0, 1)$, $f(1^n) = (1, 0)$ und für alle anderen Suchpunkte x sei $f(x) = (0, 0)$. Beide Ziele sind zu maximieren. Offenbar bilden 0^n und 1^n die Paretomenge. Es sei a der paretooptimale Suchpunkt, der zuerst erzeugt wird, und dies geschehe zum Zeitpunkt t. Da a sämtliche Suchpunkte dominiert, die zu allen vorhergehenden Zeitpunkten $t' < t$ erzeugt wurden, gilt $P_t = \{a\}$. Der Hammingabstand zu dem zweiten Paretooptimum $\overline{a}$ im Suchraum beträgt n. Daher ist die erwartete Wartezeit n^n, bis $\overline{a}$ erzeugt wird. Die Wahrscheinlichkeit, dass $\overline{a}$ in $n^{n/2}$ Schritten erzeugt wird, ist höchstens $n^{-n} \cdot n^{n/2} = n^{-\Omega(n)}$. □

Die obere Schranke von $O(n^n)$ für den (1+1)-EA folgt daraus, dass die Wahrscheinlichkeit, im nächsten Schritt einen optimalen Suchpunkt zu erzeugen, mindestens $1/n^n$ ist. Für den globalen SEMO können wir daraus ableiten, dass dieser in jedem Fall eine endliche erwartete Optimierzeit hat. Die Wahrscheinlichkeit, dass im nächsten Schritt ein Repräsentant für einen bisher noch nicht repräsentierten Punkt der Paretofront gefunden wird, ist ebenfalls mindestens $1/n^n$. Der Suchraum $\{0, 1\}^n$ enthält 2^n Punkte, die im ungünstigsten Fall alle zur Repräsentantenmenge der Paretofront gehören. Es folgt nun mithilfe der Linearität des Erwartungswertes eine obere Schranke von $O(2^n \cdot n^n)$ für die Optimierzeit für beliebige Funktionen $f\colon X^n \to F$.

Die letzte Überlegung liefert allerdings nur eine sehr grobe obere Schranke für den globalen SEMO. Wir wollen hier zeigen, dass die Optimierzeit im Worst Case von derselben Größenordnung ist wie die des (1+1)-EA. Das Hauptergebnis in diesem Kapitel ist das folgende Theorem.

Theorem 10.2. *Für jede Funktion $f\colon X^n \to F$ ist die erwartete Optimierzeit des globalen SEMO höchstens $(1 + o(1))n^n$ und weniger als $5n^n$.*

Für den Beweis benötigen wir folgendes Lemma.

Lemma 10.3. *Es sei $n \in \mathbb{N}$ mindestens so groß, dass $n^{2\lceil \log n\rceil} < 2^n$ gilt. Gegeben sei eine Menge P aus mindestens $n^{2\lceil \log n\rceil}$ verschiedenen Punkten aus X^n und ein Punkt $x \in X^n \backslash P$. Für mindestens die Hälfte der Punkte in P beträgt der Hammingabstand zu x höchstens $n - \lceil \log n\rceil$.*

Beweis. Zwei Punkte mit Hammingabstand k unterscheiden sich in genau k Bitpositionen. Also ist die Anzahl der Punkte $y \in \{0,1\}^n$ mit Hammingabstand $H(y,x) = k$ zu x genau $\binom{n}{k}$. Die Anzahl der Punkte mit einem Hammingabstand zu x von mindestens $n - \lceil \log n\rceil + 1$ ist

$$\sum_{n-\lceil \log n\rceil+1 \leq k \leq n} \binom{n}{k} \;\leq\; \lceil \log n\rceil \binom{n}{\lceil \log n\rceil} \;\leq\; n^{\lceil \log n\rceil}.$$

Folglich enthält die Menge P mindestens $n^{2\lceil \log n\rceil} - n^{\lceil \log n\rceil} \geq (1/2)n^{2\lceil \log n\rceil}$ Punkte y mit $H(x,y) \leq n - \lceil \log n\rceil$. □

Beweis zu Theorem 10.2. Für $n = 1$ ist die Optimierzeit höchstens 2, da der im zweiten Schritt erzeugte Mutant sich mit Wahrscheinlichkeit 1 von dem initialen Suchpunkt unterscheidet.

Im Folgenden sei $n \geq 2$. Es sei $P_0 := \emptyset$ die Population vor dem Initialisierungsschritt. Der Zeitpunkt $t = 1$ sei der Zeitpunkt unmittelbar nach dem Initialisierungsschritt und der Zeitpunkt $t \geq 2$ sei der Zeitpunkt nach der t-ten Zielfunktionsauswertung am Ende der Schleife des SEMO. Es sei $Y_t \subseteq F^*$ die Menge der Punkte in der Paretofront, die zum Zeitpunkt t (noch) nicht von einem Punkt in $f(P_t)$ schwach dominiert werden. Mit anderen Worten ist Y_t die Menge der Paretooptima im Zielraum, für die P_t (noch) keinen Repräsentanten (Urbildpunkt) enthält. Es sei $X_t \subseteq X^*$ das Urbild von Y_t, d. h. $X_t := X^* \setminus f^{-1}(f(P_t))$. Anders gesagt ist X_t der Teil der Paretomenge, der dem noch nicht dominierten Teil der Paretofront im Zielraum entspricht. Falls also der im $(t+1)$-ten Schritt erzeugte Mutant x_{t+1} zu X_t gehört, wird x_{t+1} in die Population aufgenommen. Später kann x_{t+1} nur noch durch gleich gute Punkte ausgetauscht werden, da x_{t+1} als Punkt der Paretomenge von keinem anderen Punkt dominiert wird.

Wir betrachten nun die zufällige Folge der Mengen $(X_t)_{t\geq 0}$, die bei einem Lauf des globalen SEMO entsteht. Offenbar gilt $X_0 = X^*$ und für $t \geq 0$ gilt $X_t \supseteq X_{t+1}$. Immer dann, wenn der $(t+1)$-te Schritt einen Mutanten $x'_{t+1} \in X_t$ erzeugt, ist X_{t+1} eine echte Teilmenge von X_t, es gilt also $X_t \supset X_{t+1}$. Wenn schließlich nach T_f Schritten für jeden Punkt der Paretofront ein Repräsentant gefunden ist, gilt $X_t = \emptyset$ für $t \geq T_f$. Die Optimierzeit T_f ist also gleich der Anzahl nicht leerer Mengen in der Folge $(X_t)_{t\geq 0}$. Es bezeichne $k := |F^*| \leq 2^n$ die Kardinalität der Paretofront. Es gibt genau k Zeitpunkte t, für die $X_t \supset X_{t+1}$ gilt. Diese Zeitpunkte teilen die Sequenz $(X_t)_{t\geq 0}$ in Abschnitte ein, sodass die Mengen innerhalb eines Abschnitts gleich sind. Es bezeichne T_j, $1 \leq j \leq k$, die Länge des j-ten Abschnitts, d. h. die

Zahl der enthaltenen Mengen X_t. Bis zum Zeitpunkt T_f könnte die Einteilung z. B. wie folgt aussehen:

$$\underbrace{X_0 = X_1 = X_2}_{T_1} \supset \underbrace{X_3 = X_4 = X_5}_{T_2} \supset \cdots \supset \underbrace{\cdots = X_{T-1}}_{T_k} \supset X_{T_f} = \emptyset.$$

Jede Menge X_t mit $1 \leq t \leq T_f$ kann nun mit dem t-ten Schritt des globalen SEMO verknüpft werden. Da die Menge X_0 mit keinem Schritt verknüpft ist und andererseits X_{T_f} zu keinem Abschnitt gehört, gilt

$$E(T_f) \;=\; \sum_{1 \leq j \leq k} E(T_j).$$

Die Aufgabe ist es nun, die Erwartungswerte $E(T_j)$ unabhängig von f abzuschätzen.

Der Initialisierungsschritt ($t = 1$) ist eine besondere Situation. Die Wahrscheinlichkeit, dass x_1 in $X_0 = X^*$ liegt, ist

$$\frac{|X^*|}{2^n} \;\geq\; (|X_0| - 1) \cdot \frac{1}{2^n} \;\geq\; (|X_0| - 1) \cdot \frac{n-1}{n^n}.$$

Für $t \geq 2$ ist diese Wahrscheinlichkeit durch den Hammingabstand des zur Mutation ausgewählten Punktes x zu allen Punkten in X_{t-1} bestimmt. Die Wahrscheinlichkeit, dass der t-te Schritt $y \in X_{t-1}$ erzeugt, ist

$$\left(\frac{1}{n}\right)^{H(x,y)} \cdot \left(1 - \frac{1}{n}\right)^{n-H(x,y)}.$$

Diese Wahrscheinlichkeit fällt monoton mit wachsendem Hammingabstand $H(x, y)$. Ausgenommen wenn y gerade das Komplement von x ist, ist sonst $H(x, y)$ höchstens $n - 1$. Daraus folgt, dass

$$(|X_{t-1}| - 1) \cdot \left(\frac{1}{n}\right)^{n-1} \cdot \left(1 - \frac{1}{n}\right) \;=\; (|X_{t-1}| - 1) \cdot \frac{n-1}{n^n}$$

eine untere Schranke für die Wahrscheinlichkeit ist, dass im t-ten Schritt irgendein Punkt aus X_{t-1} erzeugt wird. Die Kardinalität von X_0 im ersten Abschnitt ist $|X^*| \geq |F^*| = k$. Die Kardinalität einer Menge X_t, die im j-ten Abschnitt liegt, ist daher mindestens $k - j + 1$. Also gilt

$$E(T_j) \;\leq\; \frac{1}{k-j} \cdot \frac{n^n}{n-1} \qquad \text{für } 1 \leq j < k.$$

(Für $j = k$ ist diese Schranke nicht anwendbar.) Wir erhalten

$$E(T) \;\leq\; \left(\sum_{1 \leq j \leq k-1} \frac{n^n}{(k-j)(n-1)}\right) + E(T_k) \;=\; \frac{H_{k-1}}{n-1} \cdot n^n + E(T_k),$$

wobei H_{k-1} für die $(k-1)$-te harmonische Zahl steht (siehe Anhang A).

Für die weitere Abschätzung unterscheiden wir, ob die Paretofront „groß“ oder „klein“ ist. Falls $k \leq n^{2\lceil \log n \rceil}$ gilt, benutzen wir für $E(T_k)$ die triviale obere Schranke n^n. (Die Wahrscheinlichkeit, im nächsten Schritt den letzten fehlenden Punkt zu erzeugen, ist mindestens $1/n^n$.) In diesem Fall erhalten wir mit der Abschätzung für harmonische Zahlen (siehe Anhang A)

$$E(T) \leq \frac{H_{n^{2\lceil \log n \rceil}-1}}{n-1} \cdot n^n + n^n \leq \left(1 + \frac{(2\lceil \log n \rceil \ln n) + 1}{n-1}\right) n^n = \bigl(1 + o(1)\bigr) n^n.$$

Man kann nachrechnen, dass der Vorfaktor auch für kleine $n \geq 2$ kleiner als 5 ist.

Es bleibt noch der Fall $n^{2\lceil \log n \rceil} < k \leq 2^n$. Im k-ten Abschnitt gibt es nur noch einen Punkt y in der Paretofront, für den noch ein Repräsentant in X gefunden werden muss. Es gibt mindestens einen Repräsentanten $x \in X$ für y. Da die Population nun schon mindestens $n^{2\lceil \log n \rceil}$ Repräsentanten für die übrigen Punkte der Paretofront beinhaltet, können nicht alle Punkte der Population einen großen Hammingabstand zu x haben. Wegen Lemma 10.3 ist die Wahrscheinlichkeit mindestens $1/2$, dass der globale SEMO einen Punkt mutiert, dessen Hammingabstand zu x höchstens $n - \lceil \log n \rceil$ beträgt. Die Wahrscheinlichkeit, dass x generiert wird, ist also mindestens

$$\frac{1}{2} \cdot \left(\frac{1}{n}\right)^{n-\lceil \log n \rceil} \cdot \left(1 - \frac{1}{n}\right)^{\lceil \log n \rceil} \geq \frac{1}{2\mathrm{e}n^{n-\lceil \log n \rceil}}.$$

Es folgt

$$E(T) \leq \frac{H_{2^n-1}}{n-1} \cdot n^n + 2\mathrm{e}n^{n-\lceil \log n \rceil} \leq \left(\frac{(\ln 2^n) + 1}{n-1} + \frac{2\mathrm{e}}{n^{\lceil \log n \rceil}}\right) n^n = \bigl(1 - \Omega(1)\bigr) n^n$$

für n groß genug. Auch hier kann man nachrechnen, dass der Vorfaktor selbst für kleine $n \geq 2$ kleiner als 5 ist. □

Die obere Schranke für den Worst Case (d. h. $k \leq n^{2\lceil \log n \rceil}$) strebt also mit wachsendem n gegen die untere Schranke n^n. Selbst für kleine Werte für n liegt zwischen den Schranken höchstens ein Faktor von 5.

Korollar 10.4. *Für Funktionen $f\colon X^n \to F$ ist die erwartete Optimierzeit des globalen SEMO im Worst Case $\Theta(n^n)$.*

Einkriterielle Probleme (d. h. $m = 1$) erlauben nur Paretofronten aus einem Punkt. Interessant ist, dass die Laufzeitschranken im Beweis zu Theorem 10.2 für Probleme mit großen Paretofronten kleiner ausfallen als für Probleme mit kleinen Paretofronten. Die untere Schranke von $\Omega(n^n)$ erhalten wir bereits für $m = 1$, wenn wir auf Ergebnisse über den (1+1)-EA zurückgreifen, oder für $m = 2$ mithilfe von Theorem 10.1. Wenn wir die erwartete Optimierzeit betrachten und wie es üblich ist von den Konstanten in der O-Notation absehen, gehören also einige ein- und zweikriterielle Probleme zu den schwierigsten Problemen für den globalen SEMO. Kein hochdimensionales Problem (d. h. mit großer Zielraumdimension m) ist für den globalen SEMO schwieriger. Dies steht im Widerspruch zu der häufig anzutreffenden Vermutung, dass gerade die hohe Zielraumdimension Ursache für die Schwierigkeiten sind, die MOEAs mit mehrkriteriellen Problemen haben können. Zumindest für den globalen SEMO ist dies unzutreffend, wenn wir die erwartete Optimierzeit als das Kriterium für die Schwierigkeit eines Problems für den SEMO ansehen.

Wir wollen uns noch kurz überlegen, ob Nebenbedingungen die erwartete Optimierzeit im Worst Case verändern können. Zunächst ist der globale SEMO nicht dafür entworfen worden, Nebenbedingungen zu berücksichtigen. Eine sehr einfache Methode, mit Nebenbedingungen umzugehen, ist folgende. Man transformiert ein Optimierungsproblem f mit Nebenbedingungen in ein Problem f' ohne Nebenbedingungen, ohne dabei die Menge der zulässigen paretooptimalen Punkte zu verändern. Dazu fügt man dem partiell geordneten Zielraum einen neuen Punkt hinzu, der per Definition von jedem Punkt des ursprünglichen Zielraums dominiert wird. Dieses neue Minimum im Zielraum weist man nun allen Suchpunkten zu, die eine Nebenbedingung verletzen. So sind sämtliche Punkte, die eine Nebenbedingung verletzen, von allen zulässigen Punkten dominiert. Offenbar bleiben alle zulässigen paretooptimalen Punkte für f auch unter f' paretooptimal. Diese Modifikation der Zielfunktion kann der globale SEMO selbst übernehmen, indem er vor jeder Auswertung der Zielfunktion die Zulässigkeit des Punktes überprüft. In der Wirkung kommt ein Lauf dieses modifizierten globalen SEMO für ein Problem f mit Nebenbedingungen einem Lauf des unveränderten globalen SEMO für ein modifiziertes Problem f' ohne Nebenbedingungen gleich. Wir sehen also, dass die Anwesenheit von Nebenbedingungen für die erwartete Laufzeit im Worst Case kein entscheidender Aspekt ist.

11 Ein Problem mit einstellbarem Schwierigkeitsgrad

Nachdem wir im vorhergehenden Kapitel gesehen haben, dass ein- und zweikriterielle Probleme Worst-Case-Probleme für den globalen SEMO sein können, soll dieses Ergebnis in diesem Kapitel erweitert werden. Wir zeigen, dass zweikriterielle Probleme eine weite Spanne möglicher erwarteter Optimierzeiten abdecken. Dazu stellen wir ein Problem mit einstellbarem Schwierigkeitsgrad vor, für das die erwartete Optimierzeit $\Theta(n^2), \Theta(n^3), \ldots, \Theta(n^n)$ sein kann. Eine Variation des Problems liefert eine erwartete Optimierzeit von $O(n \log n)$.

Das Problem ist von einem beliebten Testproblem für mehrkriterielle evolutionäre Algorithmen inspiriert. Die Funktion $f(x) := (x^2, (x-2)^2)$ ist auch unter dem Namen *Schaffers Funktion* bekannt und gilt als das einfachste Testproblem für mehrkriterielle evolutionäre Algorithmen für den kontinuierlichen Suchraum $\mathbb{R}^n$. (Siehe z. B. Coello Coello, Van Veldhuizen und Lamont (2002) für eine Übersicht über Testprobleme.) Bild 11.1(a) zeigt die beiden Zielfunktionen in Abhängigkeit vom Suchpunkt $x \in \mathbb{R}$. Es gilt hier, beide Ziele zu minimieren. (Die Präferenzordnung aus Definition 9.6 ist für Minimierungsprobleme durch $y \preceq z \Leftrightarrow \ \forall i\colon y_i \geq z_i$ zu ersetzen.)

Schaffers Funktion ist zunächst für den globalen und den lokalen SEMO nicht zugänglich, da diese nicht für den Suchraum $\mathbb{R}$ konzipiert sind. Die Idee ist es nun, den diskreten Suchraum $X^n := \{0,1\}^n$ zunächst auf $\mathbb{N}_0 \subset \mathbb{R}$ abzubilden. Dies leistet die Funktion *binary value*, die als

$$\mathrm{BV}(x) := \sum_{0 \leq i \leq n-1} 2^i \cdot x_i$$

definiert ist. Die so erhaltenen Zahlen können nun mit f abgebildet werden. Um verschieden schwierige Varianten des Problems zu erzeugen, führen wir zwei Parameter a und b ein, die die Nullstellen der beiden quadratischen Funktionen von f angeben. Dies führt zu folgender Definition.

Definition 11.1. *Die Funktion* $f_{a,b}\colon X^n \to \mathbb{N}_0 \times \mathbb{N}_0$ *ist für* $0 \leq a < b \leq 2^n - 1$ *durch*

$$f_{a,b}(x) \ := \ \big((\mathrm{BV}(x) - a)^2, (\mathrm{BV}(x) - b)^2\big)$$

definiert. Beide Ziele sind zu minimieren.

Bild 11.1(b) skizziert die beiden Ziele von $f_{a,b}$ in Abhängigkeit von $\mathrm{BV}(x)$.

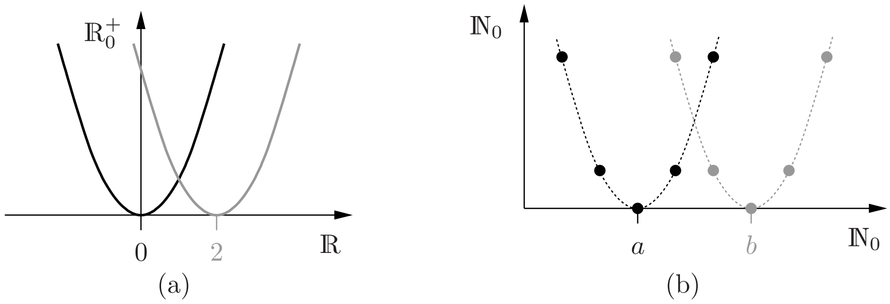

Bild 11.1: (a) Schaffers Funktion und (b) $f_{a,b}(x)$ in Abhängigkeit von $\mathrm{BV}(x)$

Proposition 11.2. *Die Paretomenge und die Paretofront von $f_{a,b}$ sind*

$$X^* = \{x \mid a \leq \mathrm{BV}(x) \leq b\} \quad \textit{bzw.} \quad F^* = \left\{\left(i^2, (b-a-i)^2\right) \,\middle|\, 0 \leq i \leq b-a\right\}$$

und daher gilt $|X^*| = |F^*|$.

Beweis. Jeder Punkt x mit $\mathrm{BV}(x) < a$ ist nicht paretooptimal, weil der Punkt mit dem um 1 größeren Binärwert bezüglich beider Ziele kleinere Zielfunktionswerte hat. Jeder Punkt x mit $\mathrm{BV}(x) > b$ ist nicht paretooptimal, weil der Punkt mit dem um 1 kleineren Binärwert bezüglich beider Ziele kleinere Zielfunktionswerte hat. Die verbleibenden Punkte z, $a \leq \mathrm{BV}(z) \leq b$, sind paretooptimal. Dazu zeigen wir, dass es keinen Punkt w gibt, der einen dieser Punkte z dominiert. Falls $\mathrm{BV}(z) = \mathrm{BV}(w)$ gilt, folgt $z =_{f_{a,b}} w$, sogar $z = w$. Falls $\mathrm{BV}(z) < \mathrm{BV}(w)$ gilt, gilt auch $(\mathrm{BV}(z)-a)^2 < (\mathrm{BV}(w)-a)^2$ und es folgt $z \not\preceq_{f_{a,b}} w$, d. h., w dominiert z *nicht*. Falls $\mathrm{BV}(z) > \mathrm{BV}(w)$ gilt, gilt auch $(\mathrm{BV}(z)-b)^2 < (\mathrm{BV}(w)-b)^2$ und es folgt ebenfalls $z \not\preceq_{f_{a,b}} w$. Der zum Punkt $z \in X^*$ mit $\mathrm{BV}(z) = a+i$ gehörende Punkt der Paretofront ist $(i^2, (b-a-i)^2)$. □

Um eine erwartete Laufzeit von $\Theta(n^k)$ zu erreichen, ist die zugrunde liegende Idee folgende. Man legt zwei Punkte mit Hammingabstand k so in den Suchraum, dass der Suchprozess einmal vom einen zum anderen Punkt „springen“ muss. Für den lokalen SEMO ist die erwartete Wartezeit auf eine Mutation, die k ausgewählte Bits kippt, $\Theta(n^k)$. Nun muss noch sichergestellt werden, dass diese Wartezeit den überwiegenden Anteil an der Optimierzeit ausmacht.

Dieses Konzept verwirklichen wir mithilfe von $f_{a,b}$. Wenn wir die Werte a und b so wählen, dass b eine Zweierpotenz ist und a um 1 kleiner als b ist, unterscheiden sich die Binärdarstellungen bei richtiger Wahl der Zweierpotenz genau in den niederwertigsten k Bitpositionen. Die Minima der beiden Ziele von $f_{a,b}$ sind Nachbarn auf der Zahlengerade. Nach Proposition 11.2 besitzt jeder der beiden Punkte der Paretofront nur einen Repräsentanten in der Paretomenge. Zudem dominiert bei dieser Wahl von a und b jedes Paretooptimum alle Punkte mit Ausnahme des anderen Paretooptimums. Eines der beiden Paretooptima im Suchraum wird zuerst gefunden. Folglich schrumpft in diesem Moment die Population bis auf diesen Punkt zusammen. Also muss der zweite paretooptimale Suchpunkt durch Mutation von k ausgewählten Bits aus dem ersten entstehen.

Theorem 11.3. *Für $k \in \{2, \ldots, n\}$ seien $a := 2^{k-1} - 1$ und $b := 2^{k-1}$. Die erwartete Optimierzeit des globalen SEMO für $f_{a,b}$ ist $\Theta(n^k)$. Für jede Konstante c mit $0 < c < 1$ ist die Optimierzeit mit einer Wahrscheinlichkeit von mindestens $1 - n^{(c-1)k}$ mindestens $n^{c \cdot k}$. Die Optimierzeit ist höchstens n^{k+1} mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$.*

Beweis. Für den Beweis ist es hilfreich, die Binärdarstellung von a und b vor Augen zu haben (siehe Tabelle 11.1). Gemäß Proposition 11.2 ist die Paretomenge X^* die zweielementige Menge aus diesen beiden Bitstrings.

Wir teilen den Lauf in zwei Phasen ein. Die erste dauert, bis der erste Punkt aus X^* erzeugt ist. Die zweite umfasst die verbleibenden Schritte, bis auch der andere Punkt aus X^* erzeugt ist. Wir zeigen zuerst, dass die erste Phase mit einer überwältigenden Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ nach $O(n^2)$ Schritten abgeschlossen ist.

Für $z < a$ sind $(z-a)^2$ und $(z-b)^2$ streng monoton fallend in z. Also sind zwei Punkte $x, y \in X^n$ mit $\mathrm{BV}(x) < a$ und $\mathrm{BV}(y) < a$ stets vergleichbar. Ebenso sind x und y vergleichbar, wenn $b < \mathrm{BV}(x)$ und $b < \mathrm{BV}(y)$ gilt. Folglich kann es zu jeder Zeit höchstens einen Suchpunkt x^{links} mit $\mathrm{BV}(x^{\text{links}}) < a$ und höchstens einen Suchpunkt x^{rechts} mit $b < \mathrm{BV}(x^{\text{rechts}})$ in der Population geben.

Wir unterteilen die erste Phase weiter in zwei Subphasen. Die erste Subphase endet, sobald ein Suchpunkt x mit $\mathrm{BV}(x) < 2^k$ erzeugt wurde. Sollte der initiale Suchpunkt schon diese Eigenschaft aufweisen, so entfällt die erste Subphase. Da alle Suchpunkte mit einem Binärwert von mindestens 2^k miteinander vergleichbar sind, kann die Population in der ersten Subphase höchstens aus einem Individuum bestehen. Im Folgenden nehmen wir o. B. d. A. an, dass die Wertigkeit der Bits in der Binärdarstellung wie in Tabelle 11.1 von links nach rechts abnimmt, d. h., das höchstwertigste Bit mit dem Gewicht 2^{n-1} ist das linkeste Bit. Wir nennen einen Schritt einen *Erfolg in der ersten Subphase*, wenn nur das höchstwertigste der vorhandenen 1-Bits kippt. Dies geschieht mit einer Wahrscheinlichkeit von $\Omega(1/n)$. Da dieses Ereignis in der ersten Subphase den Binärwert und beide Zielfunktionswerte senkt, ersetzt der Mutant das Individuum in der Population. In der ersten Subphase genügen $n-k$ Erfolge, um diese Subphase zu beenden, denn diese erzeugen ein Individuum mit einem Präfix aus mindestens $n-k$ 0-Bits. Dieses Individuum dominiert sämtliche Suchpunkte mit einem 1-Bit in den $n-k$ Präfixbits.

In der zweiten Subphase kann die Population höchstens aus x^{rechts} und x^{links} bestehen. Es kann aber immer wieder vorkommen, dass ein Mutant beide Individuen dominiert, sodass nach einem Schritt nur noch ein neues Individuum x^{rechts} oder ein neues Individuum x^{links} existiert. Wir definieren nun die Distanz der Individuen

	2^{n-1}	$\dots$	2^k	2^{k-1}	2^{k-2}	2^{k-3}	$\dots$	2^1	2^0
a	0	$\dots$	0	0	1	1	$\dots$	1	1
b	0	$\dots$	0	1	0	0	$\dots$	0	0
	x_{n-1}	$\dots$	x_k	x_{k-1}	x_{k-2}	x_{k-3}	$\dots$	x_1	x_0

Tabelle 11.1: Binärdarstellung von $a := 2^{k-1} - 1$ und $b := 2^{k-1}$

zur Paretofront als $d^{\text{links}} := a - \text{BV}(x^{\text{links}})$ und $d^{\text{rechts}} := \text{BV}(x^{\text{rechts}}) - b$. Zu jeder Zeit in der zweiten Subphase bezeichne x dasjenige Individuum, das die kleinere Distanz hat (bei gleichen Distanzen beliebig). Wenn es nur ein Individuum gibt, sei dieses x. Es bezeichne d die Distanz von x. Wenn ein Schritt dazu führt, dass x aus der Population entfernt wird, muss im selben Schritt ein neues Individuum x' in die Population eingefügt werden, das x dominiert. Falls x' in der Paretomenge liegt, ist die Phase beendet. Andernfalls sei o. B. d. A. $x = x^{\text{links}}$. Da schwach dominierende Punkte x' in keinem der Ziele schlechter sind als x, muss $\text{BV}(x) \leq \text{BV}(x')$ gelten. Falls $\text{BV}(x')$ zwischen $\text{BV}(x)$ und a liegt, ist die Distanz von x' nicht größer als die von x. Falls $\text{BV}(x') > b$ gilt, kann $\text{BV}(x')$ nicht weiter von a entfernt sein als $\text{BV}(x)$, da x' den Punkt x dominiert. Da $\text{BV}(x')$ dann sogar näher an b als an a liegt, folgt, dass auch in diesem Fall die Distanz von x' nicht größer als die von x ist. Analog argumentiert man im Fall $x = x^{\text{rechts}}$. Daraus schließen wir, dass d bis zum Ende der zweiten Subphase in keinem Schritt wächst.

Wenn in der zweiten Subphase das Individuum x mit kleinerer Distanz rechts von b liegt, d. h. $\text{BV}(x) > b$, muss offenbar $x_{k-1} = 1$ gelten, da nach dem Ende der ersten Subphase die $n-k$ Präfixbits $x_{n-1}, \dots, x_k$ jedes Individuums 0-Bits sind. Es gilt dann $d(x) = \text{BV}(x) - b = \sum_{0 \leq i \leq k-2} 2^i \cdot x_i$. Wenn lediglich das linkeste 1-Bit im Suffix $x_{k-2}, \dots, x_0$ von x kippt, entsteht ein Mutant, der x in der Population ersetzt und den Distanzwert d mindestens halbiert. Wenn andererseits das Individuum mit kleinerem Distanzwert links von a liegt, d. h. $\text{BV}(x) < a$, gilt $x_{n-1} = \cdots = x_{k-1} = 0$ und $d(x) = a - \text{BV}(x) = \sum_{0 \leq i \leq k-2} 2^i \cdot (1 - x_i)$. Kippen des linkesten 0-Bits im Suffix $x_{k-2}, \dots, x_0$ führt nun ebenfalls dazu, dass x ersetzt und der Distanzwert d mindestens halbiert wird. Wir nennen eine solche Halbierung des Distanzwertes einen *Erfolg in der zweiten Subphase*. Wegen $|P| \leq 2$ ist die Wahrscheinlichkeit eines Erfolges in der zweiten Subphase ebenfalls $\Omega(1/n)$. Um die zweite Subphase zu beenden, sind höchstens $k - 1$ Erfolge ausreichend.

Insgesamt genügen weniger als n Erfolge in der gesamten ersten Phase. Da in beiden Subphasen die Wahrscheinlichkeit für einen Erfolg $\Omega(1/n)$ ist, folgt aus Chernoffschranken, dass für eine genügend groß gewählte Konstante $d > 0$ in $\lceil dn^2 \rceil$ Schritten mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ mindestens n Erfolge enthalten sind. Falls die ersten $\lceil dn^2 \rceil$ Schritte die erste Phase nicht beenden, gelingt dies den nächsten $\lceil dn^2 \rceil$ Schritten mit mindestens derselben Wahrscheinlichkeit. Die erwartete Zahl der Wiederholungen von $\lceil dn^2 \rceil$ Schritten ist also weniger als 2. Dies impliziert, dass die erwartete Länge der ersten Phase $O(n^2)$ ist.

Wenn die zweite Phase beginnt, enthält P entweder den Suchpunkt $0^{n-k+1}1^{k-1}$ oder $0^{n-k}10^{k-1}$. Diese Punkte haben die Zielwertvektoren $(0,1)$ und $(1,0)$. Offenbar dominiert jedes dieser Paretooptima jeden anderen Punkt mit Ausnahme des jeweils anderen Paretooptimums. Daher besteht die Population nur aus demjenigen Paretooptimum $x^* \in X^*$, das zuerst generiert wird. Da jeder Punkt der Paretofront nur einen Urbildpunkt als möglichen Repräsentanten besitzt, wird x^* nie mehr aus der Population entfernt. Um nun den anderen paretooptimalen Suchpunkt zu generieren, müssen genau die k niederwertigsten Bits von x^* kippen. In einem Schritt geschieht dies höchstens mit einer Wahrscheinlichkeit von $1/n^k$ und mindestens mit einer Wahrscheinlichkeit von $1/(\mathrm{e}n^k)$. Also ist die erwartete Wartezeit $\Theta(n^k)$. Da $k \geq 2$ vorausgesetzt wurde und die erste Phase nur eine erwartete Zeit von $O(n^2)$ benötigt, ist damit die Behauptung über die erwartete Optimierzeit in Theorem 11.3 bewiesen.

Die Wahrscheinlichkeit, dass die ersten $n^{c\cdot k}$ Schritte in der zweiten Phase erfolglos bleiben, ist mindestens

$$1 - n^{c\cdot k} \cdot \frac{1}{n^k} \;=\; 1 - n^{(c-1)k}.$$

Daraus folgt, dass die Optimierzeit mit mindestens dieser Wahrscheinlichkeit wenigstens $n^{c\cdot k}$ ist. Für die behauptete obere Schranke ist das Resultat für die erste Phase gut genug. Die ersten $n^{k+1} - O(n^2) = \Theta(n^{k+1})$ Schritte in der zweiten Phase sind erfolgreich mit einer Wahrscheinlichkeit von mindestens

$$1 - \left(1 - \frac{1}{\mathrm{e}n^k}\right)^{\Omega(n^{k+1})} \;=\; 1 - \mathrm{e}^{-\Omega(n)}.$$

Insgesamt sind damit n^{k+1} Schritte in beiden Phasen mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ erfolgreich. □

Aus Theorem 11.3 folgt, dass für $k = \Theta(n)$ die Optimierzeit mit überwältigender Wahrscheinlichkeit $n^{\Theta(n)}$ ist. Darf man daraus wie bei einkriteriellen Problemen schließen, dass auch Multistartvarianten für dieses Problem erfolglos sind (vgl. Abschnitt 7.1)? Das ist i. Allg. für mehrkriterielle Probleme nicht der Fall. Wir haben nämlich gesehen, dass eines der beiden Paretooptima bei dem Problem aus Theorem 11.3 sehr schnell gefunden wird. Es gibt keinen Anhaltspunkt, dass es in jedem Lauf dasselbe ist. Die einfache Multistartstrategie, die aus verschiedenen Läufen die (vermeintlichen) Paretooptima sammelt und am Ende aus dieser Menge schwach dominierte Punkte entfernt, könnte durchaus mit hoher Wahrscheinlichkeit die Paretomenge finden.

Lineare Optimierzeiten sind für randomisierte Suchheuristiken untypisch. Bei der $1/n$-Standardmutation ist schon allein die durchschnittliche Wartezeit auf das

Kippen eines bestimmten Bits linear in n. Für sogenannte lineare Funktionen, das sind Linearkombinationen boolescher Variablen mit reellwertigen Koeffizienten, ist die Optimierzeit des (1+1)-EA von der Größenordnung $n \log n$ (Droste, Jansen und Wegener, 2002). Ein zweikriterielles Problem mit erwarteter Optimierzeit $O(n \log n)$ erhalten wir auf die gleiche Weise wie $f_{a,b}$. Anstatt Bitstrings als Binärzahlen zu interpretieren, zählen wir nun die Zahl der 1-Bits. Dies ist mit einer unären Kodierung vergleichbar. Im Folgenden sei

$$|x| := \sum_{0 \le i \le n-1} x_i.$$

Diese Funktion wird auch *CountingOnes* genannt.

Definition 11.4. *Die Funktion* $g_{a,b}\colon X^n \to \mathbb{N}_0 \times \mathbb{N}_0$ *ist für* $0 \le a < b \le n$ *durch*

$$g_{a,b}(x) := \big((|x|-a)^2, (|x|-b)^2\big)$$

definiert. Beide Ziele sind zu minimieren.

Folgende Proposition kann analog zum Beweis zu Proposition 11.2 bewiesen werden.

Proposition 11.5. *Die Paretomenge und die Paretofront von* $g_{a,b}$ *sind*

$$X^* = \{x \mid a \le |x| \le b\} \qquad \textit{bzw.} \qquad F^* = \big\{\big(i^2, (b-a-i)^2\big) \;\big|\; 0 \le i \le b-a\big\}.$$

Theorem 11.6. *Die erwartete Optimierzeit des globalen SEMO für* $g_{a,b}$ *ist durch* $O(n \log n + n(b-a)\log(b-a))$ *beschränkt.*

Die Kardinalität der Paretofront von $g_{a,b}$ ist $b-a+1 \le n+1$. Für Konstanten a und b, wie es in Schaffers Funktion der Fall ist, ergibt sich eine Paretofront konstanter Größe und eine obere Schranke von $O(n \log n)$ für die erwartete Optimierzeit. Wenn die Paretofront für $a = 0$ und $b = n$ maximal groß wird, ist die erwartete Optimierzeit durch $O(n^2 \log n)$ beschränkt.

Beweis zu Theorem 11.6. Wie im Beweis zu Theorem 11.3 teilen wir die Optimierzeit in zwei Phasen ein, wobei die erste Phase endet, sobald der erste paretooptimale Suchpunkt erzeugt ist. In der ersten Phase kann es nun wieder höchstens zwei Individuen in der Population geben, weil zwei Punkte $x, y \in X^n$ mit $|x| < a$ und $|y| < a$ oder $b < |x|$ und $b < |y|$ stets vergleichbar sind. Also gibt es auch hier höchstens ein Individuum x^{links} mit $|x^{\text{links}}| < a$ und höchstens ein Individuum x^{rechts} mit $b < |x^{\text{rechts}}|$ in der Population P. Zu einem Zeitpunkt in der ersten Phase braucht jedoch nur eines von beiden zu existieren.

Es sei zu jeder Zeit in der ersten Phase $d := \min\{||y|-a| \mid y \in P\}$ und es sei x ein Individuum mit $||x|-a| = d$. Das bedeutet, x ist dasjenige Individuum von

x^{links} und x^{rechts}, dessen Zahl der 1-Bits am wenigsten von a abweicht. Offenbar ist d^2 gleich dem Wert des ersten Zieles von $g_{a,b}(x)$. Wenn x aus P entfernt wird, dann muss im selben Schritt ein (schwach) dominierender Punkt eingefügt werden, der folglich bezüglich des ersten Zieles keinen größeren Wert hat. Somit kann sich der d-Wert nie erhöhen. Falls $|x| > b$ gilt, gibt d die Zahl der 1-Bits von x an, die noch kippen müssen, damit der Wert von d null wird. Falls $|x| < a$ gilt, gibt d die Zahl der 0-Bits von x an, die noch kippen müssen, damit der Wert von d null wird. Wenn im ersten Fall ein 1-Bit und im zweiten Fall ein 0-Bit kippt, sinkt der Wert von d um 1. Da x nicht paretooptimal ist, sinkt auch der Wert des zweiten Ziels, sodass der neue Punkt das Individuum x ersetzt. Also ist die Wahrscheinlichkeit für einen Schritt, der d um 1 verringert, mindestens

$$\frac{1}{2} \cdot d \cdot \left(\frac{1}{n}\right) \cdot \left(1 - \frac{1}{n}\right)^{n-1} \;\geq\; \frac{d}{2en}.$$

Für $d = 0$ gilt $|x| = a$. Nach Proposition 11.5 gehört x dann zur Paretomenge. Es folgt, dass die erwartete Zeit für die erste Phase durch

$$\sum_{n \geq d \geq 1} \frac{2en}{d} \;=\; 2enH_n \;=\; O(n \log n)$$

beschränkt ist.

Wir behaupten, dass die Größe der Population vor dem Ende der zweiten Phase durch $b - a$ beschränkt ist. Wenn wir zeigen, dass jede Menge P aus mindestens $b - a + 1 = |F^*|$ unvergleichbaren Suchpunkten eine Repräsentantenmenge für die Paretofront F^* ist, folgt daraus die Behauptung. Wenn es in P zwei Punkte mit höchstens $a + 1$ 1-Bits gibt, dann sind dies je ein Punkt mit a und $a + 1$ 1-Bits. Je zwei andere Punkte mit dieser Eigenschaft sind vergleichbar. Aus demselben Grund kann es nur zwei Punkte mit mindestens $b - 1$ 1-Bits in P geben, nämlich einen mit $b - 1$ und einen mit b 1-Bits. Von den Punkten mit $a + 2, \ldots, b - 2$ 1-Bits kann es je höchstens einen in P geben. Das sind weitere $b - a - 3$ Punkte. Offenbar gehören alle $b - a + 1$ Punkte, die wir nicht ausgeschlossen haben und verschieden viele 1-Bits haben, zur Paretomenge X^*. Daher sind sie paarweise unvergleichbar.

Die zweite Phase beginnt mit einer Population, die ein Individuum mit mindestens a und höchstens b 1-Bits besitzt. Wir schätzen die Wahrscheinlichkeit p_i ab, dass im nächsten Schritt ein Suchpunkt mit i 1-Bits erzeugt wird, wenn es mindestens ein Individuum x in der Population gibt, das $i - 1$ oder $i + 1$ 1-Bits hat. Das Individuum x wird mit mindestens einer Wahrscheinlichkeit von $1/(b - a)$ für den Mutationsschritt ausgewählt. Wenn x genau $i - 1$ 1-Bits hat, ist die Wahrscheinlichkeit, dass der Mutationsschritt einen neuen Suchpunkt mit genau i 1-Bits erzeugt, mindestens $(n - i + 1)(1/n)(1 - 1/n)^{n-1}$. Wenn x genau $i + 1$ 1-Bits hat, ist die Wahrscheinlichkeit, dass der Mutationsschritt einen neuen Suchpunkt mit genau i 1-Bits erzeugt, mindestens $(i+1)(1/n)(1-1/n)^{n-1}$. Wir unterschätzen die Wahrscheinlichkeit p_i nur, wenn wir im Folgenden die Abschätzung

$$p_i \geq \begin{cases} \frac{n-i+1}{b-a} \cdot \frac{1}{n} \cdot \left(1-\frac{1}{n}\right)^{n-1} \geq \frac{n-i+1}{(b-a)ne} & \text{für } i > n/2 \text{ und} \\ \frac{i+1}{b-a} \cdot \frac{1}{n} \cdot \left(1-\frac{1}{n}\right)^{n-1} \geq \frac{i+1}{(b-a)ne} & \text{für } i \leq n/2 \end{cases}$$

verwenden. Es sei T_i die Zufallsvariable, die die Wartezeit auf ein Individuum mit i 1-Bits angibt, nachdem ein Individuum mit $i-1$ oder $i+1$ 1-Bits in die Population aufgenommen worden ist. Für $i \in \{a, \ldots, b\}$ sind Suchpunkte mit i 1-Bits Elemente der Paretomenge. Daher ist für diese i der Erwartungswert $E(T_i)$ höchstens $1/p_i$. Die erwartete Zeit für die zweite Phase ist nun durch die Summe der $|F^*|-1 = b-a$ größten Schranken für $E(T_i)$ und $0 \leq i \leq n$ beschränkt. Die Schranken für $E(T_0)$ und $E(T_n)$ sind am größten, nämlich $(b-a)ne$. Für $E(T_1)$ und $E(T_{n-1})$ sind sie $(b-a)ne/2$, für $E(T_2)$ und $E(T_{n-2})$ sind sie $(b-a)ne/3$ usw. Also ist die erwartete Länge der zweiten Phase höchstens

$$2 \cdot (b-a)ne \sum_{1 \leq i \leq \lceil (b-a)/2 \rceil} \frac{1}{i} = O\bigl(n(b-a)\log(b-a)+n\bigr).$$

Der letzte Summand "$+n$" auf der rechten Seite ist für den Fall $b-a=1$ erforderlich. Zusammen mit der ersten Phase ergibt sich nun eine erwartete Optimierzeit von $O(n \log n + n(b-a)\log(b-a))$. □

Die im letzten Beweis vorgestellte Analyse des globalen SEMO beruht ausschließlich auf 1-Bit-Mutationen. Schritte, in denen der globale SEMO mehr als ein Bit kippt, haben keine Auswirkungen auf die Korrektheit der oberen Schranke. Für den Mutationsoperator des lokalen SEMO ist die Wahrscheinlichkeit, ein bestimmtes Bit zu kippen, nicht kleiner als die Wahrscheinlichkeit, dass in $1/n$-Standardmutationen nur ein bestimmtes Bit kippt. Daraus folgt, dass die Abschätzungen im Beweis zu Theorem 11.6 auch für den lokalen SEMO gültig sind.

Korollar 11.7. *Die erwartete Optimierzeit des lokalen SEMO für $g_{a,b}$ ist durch $O(n \log n + n(b-a)\log(b-a))$ beschränkt.*

Beim lokalen SEMO breitet sich die Population bei der Optimierung von $g_{a,b}$ in der zweiten Phase um den zuerst gefundenen paretooptimalen Suchpunkt aus. Die Population bildet in diesem Gebiet eine Kette aus Hammingnachbarn. Das Gleiche geschieht eventuell später nochmal für den zweiten Punkt aus der ersten Phase, wenn dieser die Paretomenge erreicht. Irgendwann verschmelzen die Ketten zu einer Kette. Der lokal arbeitende Suchoperator des lokalen SEMO kann jedoch nur dann neue Punkte der Paretomenge erzeugen, wenn ein Suchpunkt von einem Ende dieser Ketten mutiert wird. Diese zeichnen sich dadurch aus, dass sie nur einen Hammingnachbarn in P besitzen, während Individuen aus der Mitte einer Kette zwei Hammingnachbarn in P haben.

Diese Beobachtung machen sich Kumar und Banerjee (2005) zu Nutze und entwerfen eine Variante des lokalen SEMO. Die Variante REMO (Restricted Evolutionary Multi-objective Optimizer) teilt ihre Population P in einen „aktiven" Teil P' aus nur zwei Individuen und ein Archiv A ein. Diese Einteilung wird regelmäßig neu festgelegt. Für P' werden bevorzugt diejenigen Individuen gewählt, die am wenigsten Hammingnachbarn in P besitzen. Zur Mutation wird ein Individuum aus P' ausgewählt. So ist in der zweiten Phase gewährleistet, dass die Wahrscheinlichkeit, einen Randpunkt einer Kette zu mutieren, $\Omega(1)$ ist. Kumar und Banerjee (2005) nehmen sich den Beweis zu Theorem 11.6 zum Vorbild und zeigen für REMO eine Schranke von $O(n \log n)$. Die Analyse der ersten Phase bleibt unberührt, da die Population hier höchstens aus zwei Individuen besteht. In der zweiten Phase vergrößern sich die unteren Schranken für p_i um den Faktor $(b-a)$, da die Selektion nun mit Wahrscheinlichkeit $\Omega(1)$ anstatt $\Omega(1/(b-a))$ das „richtige" Individuum zur Mutation auswählt. Dadurch ergibt sich eine erwartete Zeit von $O(n \log(b-a) + n) = O(n \log n)$ auch für die zweite Phase.

Im Gegensatz zum globalen SEMO, der möglichst allgemein gehalten ist, handelt es sich bei dem REMO um eine spezialisierte Heuristik, die Problemwissen einsetzt. Der Selektionsoperator ist auf Paretomengen zugeschnitten, die bez. der Hammingnachbarschaft zusammenhängend sind. Für das Problem $g_{a,b}$ und z. B. $b-a = \Omega(\log n)$ ist die Schranke für REMO kleiner als die für den globalen und für den lokalen SEMO. Es liegt nahe, dass der REMO dann tatsächlich schneller ist.

12 Das Leading-Ones-Trailing-Zeroes-Problem

Die ersten Laufzeitanalysen für den lokalen SEMO wurden in der Arbeit von Laumanns, Thiele, Zitzler, Welzl und Deb (2002b) vorgestellt. Das dort behandelte zweikriterielle Problem LOTZ (Leading Ones Trailing Zeroes) wird in diesem Kapitel für den globalen SEMO untersucht.

Definition 12.1. *Für einen Bitstring $x = x_1, \ldots, x_n$ seien*

$$\begin{aligned} \mathrm{LO}(x) &:= \sum_{i=1}^{n} \prod_{j=1}^{i} x_j \qquad \textit{und} \\ \mathrm{TZ}(x) &:= \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j) \end{aligned}$$

die Länge seines Präfixes aus 1-Bits (engl. leading ones) bzw. die Länge seines Suffixes aus 0-Bits (engl. trailing zeroes).

Die Funktion $\mathrm{LOTZ}\colon \{0,1\}^n \to \{(i,j) \mid 0 \le i + j \le n,\ i + j \ne n - 1\}$ *ist durch*

$$\mathrm{LOTZ}(x) \;:=\; \big(\mathrm{LO}(x), \mathrm{TZ}(x)\big)$$

definiert. Beide Ziele sind zu maximieren.

Proposition 12.2. *Die Paretofront und die Paretomenge von LOTZ sind*

$$F^* \;=\; \{(i, n-i) \mid 0 \le i \le n\} \qquad \textit{bzw.} \qquad X^* \;=\; \{1^i 0^{n-i} \mid 0 \le i \le n\}.$$

Die Paretomenge ist die einzige Repräsentantenmenge für die Paretofront.

Beweis. Offenbar wird jeder Suchpunkt mit i Präfixeinsen und j Suffixnullen für $i + j \le n - 2$ von dem Suchpunkt $1^i 0^{n-i}$ dominiert. Suchpunkte, für die die Anzahl der Präfixeinsen und Suffixnullen in der Summe $n - 1$ ergibt, kann es nicht geben, da die verbleibende Bitposition entweder zum Präfix oder zum Suffix gehört. Alle Suchpunkte der Form $1^i 0^{n-i}$ sind paretooptimal, da eine Verlängerung des Präfixes aus 1-Bits eine Verkürzung des Suffixes aus 0-Bits zur Folge hat. Ebenso hat eine Verlängerung des Suffixes aus 0-Bits eine Verkürzung des Präfixes aus 1-Bits zur Folge. Weil es nur *einen* Bitstring der Länge n aus i Präfixeinsen und $n - i$ Suffixnullen gibt, folgt auch die zweite Aussage. □

In der oben angesprochenen Arbeit von Laumanns, Thiele, Zitzler, Welzl und Deb sowie in der später erschienenen, erweiterten Zeitschriftenfassung (Laumanns, Thiele und Zitzler, 2004c) wird für den lokalen SEMO eine erwartete Optimierzeit von $\Theta(n^3)$ für LOTZ nachgewiesen. Hier soll untersucht werden, ob der Einsatz von $1/n$-Standardmutationen anstelle des lokalen Suchoperators zu einer Verschlechterung der Optimierzeit führt. Wir gehen ähnlich vor wie Laumanns, Thiele, Zitzler, Welzl und Deb und teilen die Optimierzeit in zwei Phasen ein, wobei die erste Phase mit dem Erzeugen eines paretooptimalen Suchpunktes beendet ist. Das LOTZ-Problem hat die besondere Eigenschaft, dass 1-Bit-Mutationen nicht gleichzeitig das Präfix aus 1-Bits und das Suffix aus 0-Bits verlängern können. Das hat zur Folge, dass der Mutant in der ersten Phase stets mit seinem Elter vergleichbar ist. Daher kann die Population des lokalen SEMO in der ersten Phase nicht mehr als *ein* Individuum umfassen. Diese Eigenschaft überträgt sich jedoch nicht auf den globalen SEMO. Um die Größe der Population des globalen SEMO abzuschätzen, benutzen wir folgendes Lemma.

Lemma 12.3. *Es sei $A \subseteq X^n$ eine Repräsentantenmenge für eine Approximationsmenge $A' \neq F^*$. Die Kardinalität von A ist höchstens n.*

Beweis. Weil A eine Repräsentantenmenge ist, gilt $|A| = |A'|$ und es genügt zu zeigen, dass $|A'| \leq n$ gilt. Die charakteristische Funktion, die angibt, ob ein Element der Menge $\{0, \dots, n\}^2$ zum Zielraum $F \subseteq \{0, \dots, n\}^2$ gehört, kann man sich als 0-1-Dreiecksmatrix mit 1-Einträgen an den Positionen (i, j) mit $0 \leq i + j \leq n$ und $i + j \neq n - 1$ vorstellen. Der Zeilenindex i gibt die Anzahl der Präfixeinsen an, der Spaltenindex j die Anzahl der Suffixnullen. Da A' eine Approximationsmenge ist und deswegen kein Element von A' von einem anderen Element von A' (schwach) dominiert wird, kann es aus jeder der $n + 1$ Spalten höchstens ein Element in A' geben. Das Gleiche gilt für die $n + 1$ Zeilen. Also folgt $|A'| \leq n + 1$.

Angenommen, es gilt $|A'| = n + 1$. Dann gibt es aus jeder Zeile und jeder Spalte ein Element in A'. Offenbar gibt es nur eine Möglichkeit, die Elemente von A' zu wählen, nämlich die Elemente auf der Diagonalen $(i, n - i)$ für $0 \leq i \leq n$. Für $|A'| = n + 1$ folgt also $A' = F^*$. Nach der Voraussetzung im Lemma gilt jedoch $A' \neq F^*$. □

Theorem 12.4. *Die erwartete Optimierzeit des globalen SEMO für das Problem LOTZ ist $O(n^3)$. Die Optimierzeit ist mit einer überwältigenden Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ durch $O(n^3)$ beschränkt.*

Beweis. Wir teilen die Optimierzeit in zwei Phasen auf, sodass die erste Phase beendet ist, sobald der erste paretooptimale Suchpunkt erzeugt wurde.

Jeder Suchpunkt x, der nicht paretooptimal ist, hat eine „Lücke" der Breite $n - \mathrm{LO}(x) - \mathrm{TZ}(x)$ zwischen seinem Präfix aus 1-Bits und seinem Suffix aus 0-Bits. Im Folgenden bezeichne x stets ein Individuum der Population mit kleinster Lücke.

Wegen Lemma 12.3 wird x in jedem Schritt mindestens mit einer Wahrscheinlichkeit von $1/n$ zur Mutation ausgewählt. Mit einer Wahrscheinlichkeit von $\Theta(1/n)$ kippt der Mutationsoperator nur das am weitesten vorn stehende 0-Bit oder nur das am weitesten hinten stehende 1-Bit. Wenn das geschieht, dominiert der Mutant seinen Elter und wird in die Population aufgenommen. Also ist die Wahrscheinlichkeit, dass ein Schritt des globalen SEMO die minimale Lückenbreite in der Population um mindestens 1 senkt, $\Omega(1/n^2)$. Da die Lückenbreite 2 nicht möglich ist, genügt es in der ersten Phase, die minimale Lückenbreite höchstens $(n-1)$-mal zu verkleinern. Für eine genügend groß gewählte Konstante $c \geq 0$ folgt aus Chernoffschranken, dass in cn^3 Schritten mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ mindestens $n-1$ verkleinernde Schritte enthalten sind.

Die Paretomenge X^* kann man als eine Kette von Suchpunkten auffassen, die von 0^n zu 1^n reicht und dazwischen die Punkte 1^i0^{n-i} $(1 \leq i \leq n-1)$ eingliedert. Offenbar hat jedes Mitglied der Kette mindestens einen Hammingnachbarn, der ebenfalls zur Kette gehört und sein Vorgänger oder Nachfolger in der Kette ist. Solange wie in der zweiten Phase $P \neq X^*$ gilt, gibt es mindestens ein Individuum $x \in P$ mit einem Hammingnachbarn $x' \in X^* \setminus P$. Der nächste Schritt erzeugt x' mit einer Wahrscheinlichkeit von $\Omega(1/n^2)$. Aus Chernoffschranken folgt, dass für eine genügend groß gewählte Konstante $c' > 0$ in $c'n^3$ Schritten in der zweiten Phase alle n verbleibenden Punkte der Paretomenge mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ gefunden werden.

Damit ist die zweite Aussage des Theorems bewiesen. Die Aussage über die erwartete Optimierzeit folgt, weil die nächsten $(c + c')n^3$ Schritte mit mindestens derselben Wahrscheinlichkeit erfolgreich sind, sodass die erwartete Zahl an erforderlichen Wiederholungen kleiner als 2 ist. □

Die erwartete Laufzeit des globalen SEMO für LOTZ ist abgesehen von den Konstanten, die sich in der O-Notation verbergen, nicht größer als die des lokalen SEMO. Das ist das Hauptergebnis dieses Kapitels.

Im letzten Beweis haben wir zur Abschätzung der Laufzeit wieder nur auf 1-Bit-Mutationen gesetzt. Die Wahrscheinlichkeit dieser 1-Bit-Mutationen ist für den lokalen SEMO nur größer, sodass die Aussagen in Theorem 12.4 auch für den lokalen SEMO gelten. Laumanns, Thiele, Zitzler, Welzl und Deb haben nicht nur die erwartete Optimierzeit des lokalen SEMO untersucht, sondern auch bewiesen, dass die Optimierzeit des lokalen SEMO mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ mindestens $\Omega(n^3)$ ist. Indem wir beide Resultate kombinieren, erhalten wir folgendes Korollar.

Korollar 12.5. *Die erwartete Optimierzeit des lokalen SEMO für LOTZ ist* $\Theta(n^3)$. *Die Optimierzeit ist* $\Theta(n^3)$ *mit einer Wahrscheinlichkeit von* $1 - \mathrm{e}^{-\Omega(n)}$.

Die Wahrscheinlichkeitsverteilung der Optimierzeit des lokalen SEMO ist also scharf um den Erwartungswert konzentriert. Große Abweichungen vom Erwartungswert sind unwahrscheinlich.

In der Arbeit von Laumanns, Thiele, Zitzler, Welzl und Deb (2002b) wird noch eine andere Variante des lokalen SEMO untersucht, die FEMO (Fair Evolutionary Multi-objective Optimizer) heißt. Der Grundgedanke dieser Variante ist es, dass alle Individuen der Population zu jeder Zeit möglichst gleich oft („fair") zur Mutation ausgewählt worden sein sollen. Neu in die Population aufgenommene Individuen werden daher bevorzugt ausgewählt, damit sie gegenüber den älteren Individuen aufholen können. Dazu erhält jedes Individuum einen Zähler, der erhöht wird, immer wenn das Individuum ausgewählt wird. FEMO wählt stets ein Individuum mit kleinstem Zählerstand aus. Die Hoffnung hinter dieser Strategie ist folgende. Wenn ein neuer Punkt in die Population aufgenommen wird, bezeugt dies, dass in dem Gebiet dieses Punktes im Suchraum bisher nicht dominierte Punkte gefunden wurden (FEMO verwirft Punkte, die von der Population schwach dominiert werden). Daher soll sich die Suche auf dieses Gebiet konzentrieren. Wenn der Erfolg aber ausbleibt, bekommen auch ältere Punkte bald wieder mehr Chancen. Die Autoren zeigen, dass die erwartete Optimierzeit des FEMO für LOTZ nur $\Theta(n^2 \log n)$ ist.

Die am Ende von Kapitel 11 erwähnte Variante REMO, die auf Probleme mit einer bez. der Hammingnachbarschaft zusammenhängenden Paretomenge spezialisiert ist, hat eine erwartete Laufzeit von nur $O(n^2)$ für LOTZ (Kumar und Banerjee, 2005). Für die erste Phase bis zum Erreichen der Paretomenge ist die erwartete Zeit $O(n^2)$. Hier überträgt sich die Analyse des lokalen SEMO unmittelbar auf REMO. In der zweiten Phase, in der die Paretomenge erschlossen wird, bildet die Population eine Kette aus Hammingnachbarn in der Paretomenge. In jedem Schritt wird mit einer Wahrscheinlichkeit von $\Omega(1)$ ein Suchpunkt von einem Ende dieser Kette gewählt. Der nachfolgende Mutationsschritt erzeugt dann mit Wahrscheinlichkeit $1/n$ das nächste Glied der Kette. Die erwartete Zeit, bis sämtliche Punkte der Paretomenge aufgereiht sind, ist daher $O(n^2)$.

13 Das Multi-Objective-Counting-Ones-Problem

In diesem Kapitel analysieren wir eingehend die Optimierzeit des lokalen SEMO und des globalen SEMO für das zweikriterielle Problem MOCO (Multi-objective Counting Ones).

13.1 Das Problem

Die Idee der Zielfunktion MOCO ist, zunächst die Zahl der 1-Bits eines Bitstrings linear auf einen Winkel φ im Intervall $[0, 2\pi]$ abzubilden. Der Sinus und der Kosinus dieses Winkels φ sind die beiden Ziele, die es zu maximieren gilt.

Definition 13.1. *Für* $n := 4k$ *ist die Funktion* $\text{MOCO}\colon \{0,1\}^n \to [-1,1]^2$ *durch*

$$\text{MOCO}(x) \;:=\; \big(\cos\varphi(x), \sin\varphi(x)\big) \qquad \textit{und} \qquad \varphi(x) := 2\pi \cdot \frac{|x|}{n}$$

definiert. Beide Ziele sind zu maximieren.

Die Zielfunktion MOCO wird in der Arbeit von Thierens (2003) eingeführt und die Optimierzeit des lokalen und des globalen SEMO sowohl theoretisch als auch experimentell untersucht. Bevor wir motivieren, warum es notwendig ist, die Optimierzeit beider randomisierter Suchheuristiken für MOCO erneut zu analysieren, betrachten wir die folgende Einteilung des Suchraums in vier Regionen (vgl. Bild 13.1). Diese liegt Thierens' theoretischer Analyse zugrunde.

Region I ($0 \leq |x| < n/4$). Beide Ziele sind in dieser Region nicht negativ. Für jeden Suchpunkt y in den Regionen II–IV ist mindestens ein Ziel nicht positiv. Also wird kein Punkt dieser Region von einem Punkt aus den Regionen II–IV dominiert. Das zweite Ziel (Sinus) wächst streng monoton und das erste Ziel (Kosinus) fällt streng monoton in der Zahl der 1-Bits. Also gilt für zwei Punkte x und y dieser Region entweder $x =_{\text{MOCO}} y$ oder $x \parallel_{\text{MOCO}} y$. Darüber hinaus sind sämtliche Suchpunkte dieser Region paretooptimal.

Region II ($n/4 \leq |x| < n/2$). In dieser Region wird jeder Punkt mit i 1-Bits von jedem anderen Punkt dieser Region mit weniger als i 1-Bits dominiert. Also ist jeder Punkt mit (genau) $n/4$ 1-Bits ein maximales Element innerhalb dieser Region. Zudem ist jeder dieser Punkte mit $n/4$ 1-Bits sogar ein Paretooptimum.

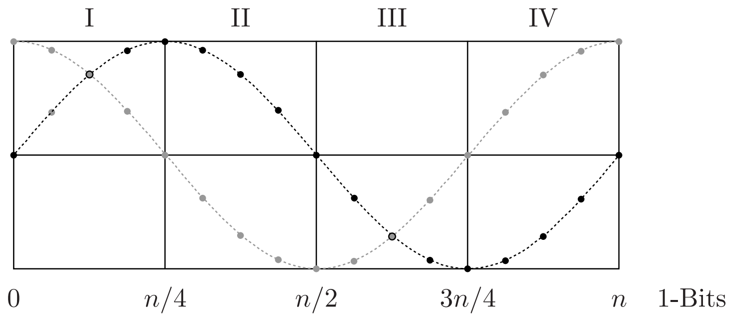

Bild 13.1: Die Ziele der Zielfunktion MOCO in Abhängigkeit von der Zahl der 1-Bits für $n = 16$

Region III ($n/2 \leq |x| < 3n/4$). Jeder Punkt dieser Region wird von jedem Punkt aus Region I dominiert. Also gibt es in dieser Region keine Paretooptima.

Region IV ($3n/4 \leq |x| \leq n$). In dieser Region wird jeder Punkt mit i 1-Bits von jedem anderen Punkt dieser Region mit mehr als i 1-Bits dominiert. Also ist der Punkt 1^n das maximale Element innerhalb dieser Region. Da der Punkt 1^n und der Punkt 0^n aus Region I Repräsentanten desselben Punktes der Paretofront sind, ist 1^n sogar ein Paretooptimum.

Wir bemerken, dass die Funktionen Sinus und Kosinus in der Definition von MOCO ebenso gut innerhalb jeder der vier Regionen durch lineare Funktionen ersetzt werden können, sodass sich in Bild 13.1 „Sägezahnkurven“ ergeben. Das würde die partielle Ordnung im Zielraum nicht verändern. Die trigonometrischen Funktionen sind also kein wesentlicher Bestandteil von MOCO. Das Ergebnis der Diskussion der vier Regionen halten wir in folgender Proposition fest.

Proposition 13.2. *Die Paretomenge von MOCO ist die Menge*

$$X^* = \{x \mid (0 \leq |x| \leq n/4) \vee (|x| = n)\}.$$

Jede Repräsentantenmenge der Paretofront enthält für jedes $i \in \{1, \ldots, n/4\}$ genau einen Punkt mit i 1-Bits und dazu entweder den Punkt 0^n oder den Punkt 1^n.

Man kann nun jede der vier Regionen als ein eigenes zweikriterielles Optimierungsproblem auffassen und die Optimierzeit für jede Region getrennt analysieren. Die theoretische Analyse von Thierens (2003) stützt sich auf ein Modell (*„convergence model“*), das auf folgender Annahme beruht. Da seine Analyse der Region I die größte Optimierzeit ergeben hat und Startpunkte mit wesentlich mehr als $n/2$ 1-Bits unwahrscheinlich sind, wird angenommen, dass die Optimierzeit die Summe der Optimierzeiten von Region I und Region II ist. Seine Analyse und die experimentelle Untersuchung kommt zu dem Ergebnis, dass die erwartete Optimierzeit des lokalen und des globalen SEMO für MOCO vermutlich von der Größenordnung $n^2 \log n$ ist.

13.2 Die erwartete Optimierzeit

Wir zeigen zunächst, dass der Erwartungswert der Optimierzeit des lokalen SEMO für MOCO nicht endlich ist (d. h., der Erwartungswert existiert nicht) und dass die erwartete Optimierzeit des globalen SEMO sehr groß ist, nämlich $n^{\Omega(n)}$. Dazu benötigen wir das folgende Lemma.

Lemma 13.3. *Für eine beliebige Konstante $\varepsilon \in]0,1]$ und jedes beliebige Polynom $p(n) \geq 0$ gilt Folgendes. Die Wahrscheinlichkeit, dass in $p(n)$ $1/n$-Standardmutationen mindestens einmal n^{ε} Bits kippen, ist $n^{-\Omega(n^{\varepsilon})}$.*

Beweis. Die Wahrscheinlichkeit, dass in einer $1/n$-Standardmutation mindestens $k \geq 0$ Bits kippen, ist höchstens

$$\binom{n}{k}\left(\frac{1}{n}\right)^{k} \leq \frac{1}{k!} \leq \left(\frac{\mathrm{e}}{k}\right)^{k} = \mathrm{e}^{-k\ln k + k}.$$

(Für Abschätzungen der Fakultätsfunktion und von Binomialkoeffizienten siehe Anhang A.) Für jedes Polynom gibt es Konstanten $c \geq 0$ und $d \geq 0$, sodass $p(n) \leq cn^d$ gilt. Die Wahrscheinlichkeit, dass in $p(n) \leq cn^d$ vielen $1/n$-Standardmutationen wenigstens eine Mutation mindestens k Bits kippt, ist durch die Summe der entsprechenden Wahrscheinlichkeiten für jede einzelne Mutation beschränkt. Diese Summe ist höchstens $\mathrm{e}^{-k\ln k+k} \cdot cn^d$. Für $k = n^{\varepsilon}$ ist diese Summe höchstens $\mathrm{e}^{-\Omega(n^{\varepsilon}\ln n)} = n^{-\Omega(n^{\varepsilon})}$. □

Theorem 13.4. *Der lokale SEMO hat für MOCO keine endliche erwartete Optimierzeit. Die erwartete Optimierzeit des globalen SEMO für MOCO ist $n^{\Omega(n)}$.*

Beweis. Mit einer positiven Wahrscheinlichkeit von 2^{-n} ist der Startpunkt der Suche der Punkt 1^n. Dieser Punkt dominiert jeden Punkt mit mindestens $n/2$ 1-Bits. Falls die Suche dort startet, erzeugt jeder Mutationsschritt, der höchstens $n/2$ Bits kippt, einen Suchpunkt, der vom Startpunkt dominiert wird. Der lokale SEMO ist dann nicht in der Lage, neue Punkte zu erzeugen, die in die Population aufgenommen werden können. Für alle $t \geq 1$ ist also die Wahrscheinlichkeit, dass die Optimierzeit T_{lokal} des lokalen SEMO mindestens t ist, mindestens so groß wie die Wahrscheinlichkeit, dass der Startpunkt der Punkt 1^n ist. Daraus folgt, dass

$$E(T_{\text{lokal}}) = \sum_{t\geq 1} \text{Prob}(T_{\text{lokal}} \geq t) \geq \sum_{t\geq 1} 2^{-n}$$

divergiert.

Im Beweis zu Lemma 13.3 haben wir gezeigt, dass die Wahrscheinlichkeit, dass der Mutationsoperator des globalen SEMO mehr als $n/2$ Bits kippt, höchstens $\mathrm{e}^{-(n/2)\ln(n/2)+(n/2)} = n^{-\Omega(n)}$ ist. Es bezeichne T_{global} die Optimierzeit des globalen

SEMO und A das Ereignis, dass der Startpunkt der Suche der Punkt 1^n ist. Es folgt nun mit dem Satz von der totalen Wahrscheinlichkeit (siehe Anhang A)

$$E(T) \ \geq \ E(T \mid A) \cdot \text{Prob}(A) \ = \ n^{\Omega(n)} \cdot 2^{-n} \ = \ n^{\Omega(n)}.$$

Damit ist der Beweis beider Aussagen beendet. Wir bemerken, dass die Tatsache, dass 1^n ein Paretooptimum ist, für den Beweis keine Rolle spielt. Es ist auch nicht entscheidend, dass der Startpunkt genau der Punkt 1^n ist. Für den lokalen SEMO genügt es offenbar, dass der Startpunkt in Region IV liegt. Das gilt im Grunde auch für den globalen SEMO, es sind aber zusätzliche Argumente erforderlich. $\square$

Bezüglich der erwarteten Optimierzeit ist MOCO für den lokalen SEMO offenbar ein Worst-Case-Beispiel und für den globalen SEMO ist die erwartete Optimierzeit exponentiell. Wie ist es da möglich, dass Thierens' experimentelle Untersuchungen seine theoretisch begründete Vermutung bestätigen konnten? Die erwartete Optimierzeit ist in diesem Fall kein aussagekräftiges Maß. In typischen Läufen ist die Optimierzeit des lokalen und globalen SEMO nicht annähernd so groß wie ihr Erwartungswert.

13.3 Eine obere Schranke für die Optimierzeit

In dieser Situation suchen wir nach Resultaten, die beweisen, dass die Optimierzeit ein bestimmtes Maß mit möglichst hoher Wahrscheinlichkeit nicht überschreitet. Das Ziel dieses Kapitels ist es daher, die folgende Vermutung zu beweisen: Mit einer Wahrscheinlichkeit von mindestens $1 - o(1)$ ist die Optimierzeit des lokalen und des globalen SEMO $\Theta(n^2 \log n)$. In diesem Abschnitt beginnen wir mit einer entsprechenden oberen Schranke von $O(n^2 \log n)$.

Die Vermutung stützt sich auf folgende Beobachtung. Bei der Optimierung besteht die Gefahr, dass sich die Population zu Beginn in Region III und Region IV versammelt, um dann zum Paretooptimum 1^n zu streben. Aus dem Beweis zu Theorem 13.4 wissen wir, dass dann selbst für den globalen SEMO die Optimierzeit groß wird. Eine „sichere“ Situation hingegen ist erreicht, sobald einmal ein Individuum mit weniger als $n/2$ 1-Bits existiert. Da Punkte mit weniger als $n/2$ 1-Bits nicht von Punkten mit mindestens $n/2$ 1-Bits dominiert werden, besteht die beschriebene Gefahr dann nicht mehr. Zudem ist die erwartete Anzahl an 1-Bits im Startpunkt der Suche mit hoher Wahrscheinlichkeit nicht viel größer als der Erwartungswert $n/2$. Es besteht also eine gute Chance, dass ein Punkt mit weniger als $n/2$ 1-Bits erzeugt wird, bevor ein Punkt mit vielen 1-Bits erzeugt wird, der den Startpunkt dominieren würde.

Für die im Folgenden vorgestellten Beweise der oben formulierten Vermutung führen wir eine Notation für Individuen der Population ein. Weil die Population des lokalen und des globalen SEMO stets eine Repräsentantenmenge ist, kann es in der Population zu jeder Zeit höchstens ein Individuum mit i 1-Bits geben. Für $i \in \{0, \ldots, n\}$ bezeichnen wir dieses Individuum mit x^i, falls es existiert. Analog zum Beweis zu Theorem 11.6 bezeichnen wir mit x^{links} das Individuum einer Population mit der geringsten Zahl an 1-Bits und mit x^{rechts} dasjenige mit der größten.

Weil die Definition der Zielfunktion MOCO auf der Anzahl der 1-Bits eines Punktes beruht, ist es klar, dass die Populationsgröße $|P|$ zu jeder Zeit durch $n+1$ beschränkt ist. Da $\text{MOCO}(0^n) = \text{MOCO}(1^n)$ gilt, ist die Populationsgröße höchstens n. (Tatsächlich gilt sogar $|P| \leq n/4 + 1$.) Wir merken uns nur, dass die Wahrscheinlichkeit, dass ein bestimmtes Individuum der Population zur Mutation ausgewählt wird, stets mindestens $1/n$ ist.

Die im restlichen Teil dieses Kapitels ermittelten Schranken vom Typ $1 - \mathrm{e}^{-\Omega(n)}$ für Wahrscheinlichkeiten folgen stets aus Anwendung von Chernoffschranken (siehe Anhang A). Da dies an so zahlreichen Stellen der Fall ist, wird im Folgenden darauf verzichtet, dies zu erwähnen.

Theorem 13.5. *Für jedes Polynom $p(n) > 0$ und n genügend groß für $p(n)$ ist mit einer Wahrscheinlichkeit von mindestens $1 - 1/p(n)$ die Optimierzeit des lokalen und des globalen SEMO für MOCO $O(n^2 \log n)$.*

Beweis. Wir beschreiben einen typischen Lauf der Suchheuristiken, der sich in drei aufeinander folgende Phasen einteilt. Wir werden zeigen, dass die ersten beiden Phasen mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ nach $O(n^2)$ Schritten beendet sind. Für die letzte Phase werden wir zeigen, dass diese für eine beliebige Konstante d mit einer Wahrscheinlichkeit von mindestens $1 - 1/n^d$ nach $O(n^2 \log n)$ Schritten beendet ist, wenn n genügend groß für das jeweilige d ist. Dann folgt, dass es für jedes Polynom $p(n) > 0$ eine genügend groß zu wählende Konstante d gibt, sodass die Summe der „Fehlerwahrscheinlichkeiten“ höchstens $1 - 1/p(n)$ ist. Mit „Fehlerwahrscheinlichkeiten“ sind die Wahrscheinlichkeiten gemeint, dass eine Phase länger als angenommen dauert.

Wir beweisen nun die aufgestellten Behauptungen für den lokalen und den globalen SEMO gemeinsam. Für den lokalen SEMO lassen sich die Beweisschritte an einigen Stellen vereinfachen und z. T. auch etwas bessere Schranken nachweisen. Darauf verzichten wir hier, erhalten dafür jedoch einen Beweis für beide Heuristiken. Die im Folgenden vorkommenden Konstanten sind in gewissen Grenzen willkürlich gewählt.

Die erste Phase dauert, bis ein Punkt mit weniger als $n/2$ 1-Bits generiert wurde. Der Startpunkt der Suche hat mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ weniger als $(17/32)n$ 1-Bits. Wir betrachten also nur den Fall, dass der Startpunkt wenigstens $n/2$ und weniger als $(17/32)n$ 1-Bits besitzt. Bei weniger als $n/2$ 1-Bits ist für die erste Phase nichts zu zeigen und der Fall mit mehr als $(17/32)n$ ist durch die „Fehlerwahrscheinlichkeit“ abgedeckt. Ein Schritt heiße *relevant*, wenn x^{links} oder x^{rechts} zur Mutation ausgewählt wird. In einer genügend großen Zahl von $O(n^2)$ Schritten sind mit einer Wahrscheinlichkeit von mindestens $1 - \mathrm{e}^{-\Omega(n)}$ mindestens n relevante Schritte enthalten. Im Folgenden betrachten wir nur die ersten n relevanten Schritte in einem Lauf. In jedem relevanten Schritt wird x^{links} mindestens mit einer Wahrscheinlichkeit von $1/2$ gewählt. Also wählen mindestens $(3/8)n$ der ersten n relevanten Schritte mit einer Wahrscheinlichkeit von $1-\mathrm{e}^{-\Omega(n)}$ das Individuum x^{links} aus und folglich höchstens $(5/8)n$ das Individuum x^{rechts}. Da nach Voraussetzung $|x^{\text{links}}| \geq n/2$ gilt, verringern Schritte, die x^{links} wählen, $|x^{\text{links}}|$ mindestens mit einer Wahrscheinlichkeit von $1/2$ im Falle des lokalen SEMO. Für den globalen SEMO ist die letzte Wahrscheinlichkeit mindestens $(n/2) \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/6$. Das bedeutet, $|x^{\text{links}}|$ sinkt mindestens mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ um mindestens $(1/32)n$ durch relevante Schritte. Die behauptete Aussage über die erste Phase folgt, wenn wir zeigen können, dass mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ kein Schritt der ersten Phase $|x^{\text{links}}|$ erhöht.

Es wächst $|x^{\text{links}}|$ nur, wenn x^{links} aus der Population entfernt wird, weil im selben Schritt ein x^{links} dominierender Punkt mit mehr 1-Bits eingefügt wird, d. h. ein Punkt mit mindestens $(31/32)n$ 1-Bits aus Region IV. Wegen Lemma 13.3 können wir mit einer Wahrscheinlichkeit von $1 - n^{-\Omega(n)}$ ausschließen, dass in unserer Phase aus $O(n^2)$ Schritten irgendein Schritt mindestens $(1/64)n$ Bits kippt. Also ist es mit mindestens derselben Wahrscheinlichkeit ausgeschlossen, dass $|x^{\text{rechts}}|$ in irgendeinem Schritt der Phase um mindestens $(1/64)n$ wächst. Für den lokalen SEMO gilt dies offenbar sogar mit Wahrscheinlichkeit 1. Wir können also im Folgenden davon ausgehen, dass es einen Zeitpunkt mit einem Individuum mit wenigstens $(3/4 + 1/64)n$ 1-Bits gibt, bevor ein Individuum mit mindestens $(3/4 + 1/32)n$ 1-Bits entsteht. In Abschnitt 13.1 haben wir uns schon überlegt, dass es zu jeder Zeit höchstens ein Individuum in Region IV geben kann. Daher brauchen wir uns nur auf relevante Schritte zu konzentrieren, die x^{rechts} wählen, wenn wir pessimistisch annehmen, dass x^{rechts} in Region IV mit weniger als $(3/4 + 1/32)n$ 1-Bits startet. (In Wirklichkeit ist x^{rechts} zu Beginn gleich dem Startpunkt der Suche.) Da x^{rechts} weniger als $n/4$ 0-Bits besitzt, gibt es in den höchstens $(5/8)n$ relevanten Schritten, die x^{rechts} wählen, insgesamt höchstens $(n/4)\cdot(5/8)n$ Möglichkeiten für den globalen SEMO, ein 0-Bit in ein 1-Bit zu verwandeln. In jeder dieser Möglichkeiten geschieht dies nur mit Wahrscheinlichkeit $1/n$, sodass die erwartete Vergrößerung von $|x^{\text{rechts}}|$ höchstens $(5/32)n$ ist. Für den lokalen SEMO gibt es nur $(5/8)n$ Möglichkeiten (in ebenso vielen Schritten), in denen jeweils höchstens mit einer Wahrscheinlichkeit von $1/4$ ein 0-Bit durch ein 1-Bit ersetzt wird. Also ist die erwartete Vergrößerung von $|x^{\text{rechts}}|$ auch im Fall des lokalen SEMO höchstens $(5/32)n$. Für beide Heuristiken folgt, dass die Vergrößerung mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ höchstens $(6/32)n$ ist. Also wird mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ kein Individuum mit wenigstens $(31/32)n$ 1-Bits erzeugt.

Die zweite Phase dauert, bis ein paretooptimales Individuum mit höchstens $n/4$ 1-Bits erzeugt ist. Aus unserer Vorüberlegung in Abschnitt 13.1 folgt, dass in der zweiten Phase x^{links} das einzige Individuum aus Region II ist. Da $|x^{\text{links}}| \geq n/4$

gilt, senken beide Heuristiken $|x^{\text{links}}|$ im jeweils nächsten Schritt mit einer Wahrscheinlichkeit von $\Omega(1)$. Eine ausreichend lang bemessene Phase aus $O(n)$ Schritten erzeugt also mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ ein Individuum mit höchstens $n/4$ 1-Bits.

Die dritte und letzte Phase erzeugt die noch verbleibenden Paretooptima aus Region I des Suchraums. Dabei kann es sein, dass das Individuum $x^0 = 0^n$ nicht erzeugt zu werden braucht, weil schon das Individuum $x^n = 1^n$ existiert. Es sei $\ell \leq n/4$ die Anzahl der 1-Bits des ersten Individuums mit höchstens $n/4$ 1-Bits, das erzeugt wird. Wenn es schon ein Individuum x^i aus Region I gibt, ist die Wahrscheinlichkeit, dass im nächsten Schritt x^{i+1} erzeugt wird, für beide Heuristiken $\Omega(1/n)$, da x^i mindestens $(3/4)n$ 0-Bits hat. Also werden die Individuen $x^{\ell+1}, \ldots, x^{n/4}$ in einem genügend groß gewählten Zeitraum aus $O(n^2)$ Schritten mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ erzeugt.

Für die noch fehlenden Individuen $x^{\ell-1}, \ldots, x^0$ betrachten wir zuerst den lokalen SEMO. Der lokale SEMO kann diese Individuen nur in absteigender Reihenfolge bezüglich der Zahl der 1-Bits erzeugen. Vereinfacht gesagt sind die Individuen $x^{\ell-1}, \ldots, x^0$ Kopien von x^ℓ, in der mehr und mehr 1-Bits durch 0-Bits ersetzt sind. Die letzte Aussage ist nicht (ganz) richtig, weil das Individuum x^i einer Population nicht identisch mit dem Individuum x^i der nächsten Population zu sein braucht. Es kann nämlich sein, dass in dem Schritt ein neuer Punkt mit i 1-Bits erzeugt wird, der x^i in der Population ersetzt. Wir können jedoch die Bitpositionen von x^i in der ersten Population bijektiv auf die Bitpositionen von x^i in der nächsten Population abbilden, sodass 1-Bits auf 1-Bits und 0-Bits auf 0-Bits abgebildet werden. So können wir jede Bitposition des (ursprünglichen) x^ℓ mit einer Bitposition in jedem x^i mit $i \leq \ell$ identifizieren und umgekehrt. Somit ist für den lokalen SEMO das Ereignis, dass $x^{\ell-1}, \ldots, x^0$ erzeugt worden sind, gleich dem Ereignis, dass in den Schritten, die x^{links} aus der jeweils aktuellen Population zur Mutation ausgewählt haben, das Bit an jeder Bitposition, die mit der Bitposition eines 1-Bits von x^ℓ identifiziert wird, mindestens einmal gekippt wurde. Dieses Szenario beschreibt das Sammelbilderproblem (vgl. Anhang A). Demzufolge ist der Erwartungswert der erforderlichen Zahl an Schritten, die x^{links} wählen, $O(n \log n)$ und der Erwartungswert der Gesamtzahl an Schritten $O(n^2 \log n)$. Wir müssen nun zeigen, dass die Zahl an Schritten mit hoher Wahrscheinlichkeit von derselben Größenordnung wie der Erwartungswert ist.

Es sei $p = \Omega(1/n^2)$ eine untere Schranke für die Wahrscheinlichkeit, dass x^{links} zur Mutation ausgewählt wird und ein Bit an einer bestimmten Bitposition kippt. Die Wahrscheinlichkeit, dass eine bestimmte Bitposition nicht wenigstens einmal in $s := (d'/p) \ln n = O(n^2 \log n)$ Schritten kippt, ist für eine beliebige positive Konstante d' höchstens

$$(1-p)^s \;\leq\; \mathrm{e}^{-p \cdot s} \;=\; \left(\frac{1}{n}\right)^{d'} .$$

Die Wahrscheinlichkeit der Vereinigung dieser Ereignisse für alle Bitpositionen ist höchstens $(1/n)^{d'-1}$. Wenn n und $d' > d+1$ genügend groß sind, ist die Summe der „Fehlerwahrscheinlichkeiten" in der dritten Phase für den lokalen SEMO klein genug, nämlich höchstens $1/n^d$. Damit ist der Beweis für den lokalen SEMO beendet.

Für den globalen SEMO führen wir ein Potential j ein, das zu Beginn der Phase den Wert von $\ell = |x^{\text{links}}|$ bekommt. Der Wert von j sinkt nur langsamer als der Wert von $|x^{\text{links}}|$, d. h., er ist nie kleiner. Wir betrachten nur diejenigen Schritte, die x^j zur Mutation auswählen. Immer wenn darin nur ein 1-Bit kippt, wird j um 1 verringert. Es ist dann sichergestellt, dass alle Individuen $x^\ell, \ldots, x^{j-1}$ in der neuen Population existieren. Wie im Fall des lokalen SEMO identifizieren wir die Bitpositionen jedes neuen Individuums x^i mit den Bitpositionen des ursprünglichen Individuums x^ℓ zu Beginn der dritten Phase. Es ist nun klar, dass jedes 1-Bit des ursprünglichen Individuums x^ℓ den Wert von j höchstens einmal um 1 senken kann und dass das Bit an jeder mit einem 1-Bit identifizierten Bitposition mindestens einmal kippen muss. Offenbar gibt es auch für den globalen SEMO eine untere Schranke $p = \Omega(1/n^2)$ für die Wahrscheinlichkeit, dass in einem Schritt x^{links} ausgewählt wird und eine bestimmte Bitposition kippt. Damit erhalten wir das gleiche Resultat für die dritte Phase wie im Fall des lokalen SEMO. □

13.4 Eine untere Schranke für die Optimierzeit

Die Idee der unteren Schranke ist zu zeigen, dass die dritte Phase im Beweis zu Theorem 13.5 mit hoher Wahrscheinlichkeit $\Omega(n^2 \log n)$ Schritte benötigt. Wir wissen aus dem Beweis, dass die Individuen mit mehr als i 1-Bits in Region I mit überwältigender Wahrscheinlichkeit in $O(n^2)$ Schritten erzeugt werden, wenn ein Individuum mit i 1-Bits in Region I existiert. Die dritte Phase kann also nur $\omega(n^2)$ Schritte mit hoher Wahrscheinlichkeit benötigen, wenn Suchpunkte mit wenigen 1-Bits mit hoher Wahrscheinlichkeit erst spät gefunden werden.

Das erste Ziel ist zu zeigen, dass die Population mit hoher Wahrscheinlichkeit auf eine Größe von $\Omega(n)$ anwächst, noch bevor Suchpunkte mit wenigen, d. h. mit weniger als $\Omega(n^\varepsilon)$ 1-Bits erreicht werden. Für den lokalen SEMO ist das beinahe offensichtlich, doch für den globalen SEMO werden wir hierfür eine Reihe von Argumenten benötigen. Danach befinden wir uns für die verbleibenden Schritte wieder im Szenario des Sammelbilderproblems, aus dem sich die untere Schranke herleiten lässt. Für den lokalen SEMO ist die Wahrscheinlichkeit nun hoch, dass $\Omega(n \log n)$ Schritte erforderlich sind, die x^{links} wählen, um jedes der verbleibenden $\Omega(n^\varepsilon)$ 1-Bits mindestens einmal zu kippen. Da die Population dann bereits eine Größe von $\Omega(n)$ hat, ist die Wartezeit auf so einen Schritt $\Omega(n)$.

Wir beginnen mit der Analyse des lokalen SEMO, um das skizzierte Beweiskonzept näher vorzustellen. Danach wenden wir uns dem globalen SEMO zu.

Theorem 13.6. *Die Optimierzeit des lokalen SEMO ist für jede Konstante* $\varepsilon < 1$ *mit einer Wahrscheinlichkeit von* $1 - \mathrm{e}^{-\Omega(n^\varepsilon)}$ *mindestens* $\Omega(n^2 \log n)$.

Beweis. Die Suche beginnt mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ bei einem Suchpunkt mit mehr als $n/4$ 1-Bits. Es gibt dann sogar eine geringe Chance, dass die Population zum Punkt 1^n strebt und nie das Optimierungsziel erreicht. Dieses „gute" Ereignis berücksichtigen wir nicht, sondern nehmen an, dass es irgendwann einmal ein erstes Individuum mit höchstens $n/4$ 1-Bits gibt. Da jeder Schritt nur *ein* Bit kippt, kann dieses nur das Individuum $x^{n/4}$ sein. Sobald $x^{\lfloor n/8 \rfloor}$ erzeugt ist, enthält die Population mindestens $n/8$ Individuen $x^{\lfloor n/8 \rfloor}, \ldots, x^{n/4}$. Da es sich um Paretooptima handelt, wird die Populationsgröße dann nie mehr kleiner als $\Omega(n)$. Also ist die Wahrscheinlichkeit, dass der Selektionsoperator ein bestimmtes Individuum wählt, von da an höchstens $8/n = O(1/n)$. Wir behaupten nun, dass für eine genügend klein gewählte Konstante $c > 0$, die von ε abhängt, die nächsten $cn^2 \log n$ Schritte die Individuen $P' = \{x^1, \ldots, x^{\lfloor n/8 \rfloor - 1}\}$ höchstens mit einer Wahrscheinlichkeit von $\mathrm{e}^{-\Omega(n^\varepsilon)}$ erzeugen. Es ist wichtig, dass x^0 nicht zu P' gehört, da wir nicht wissen können, ob x^n einmal erzeugt wird. Aus der Behauptung folgt, dass $\Omega(n^2 \log n)$ Schritte (inklusive des Initialisierungsschritts) mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n^\varepsilon)}$ nicht erfolgreich sind.

Ohne Beschränkung der Allgemeinheit sei $I = \{1, \ldots, \lfloor n/8 \rfloor\}$ die Menge der Bitpositionen der 1-Bits von demjenigen Individuum $x^{\lfloor n/8 \rfloor}$, das zuerst erzeugt wird. Gemeint ist das Individuum $x^{\lfloor n/8 \rfloor}$ aus der frühesten Population, die ein solches Individuum enthält. Diese Bitpositionen identifizieren wir auf dieselbe Weise wie im Beweis zu Theorem 13.5 mit Bitpositionen des jeweils aktuellen Individuums x^{links}.

Ein Schritt heißt *relevant*, wenn er x^{links} zur Mutation auswählt. Ein neues Individuum aus P' kann nur entstehen, wenn ein relevanter Schritt ein Bit an einer *neuen* Bitposition aus I kippt, d. h., deren Bit zuvor noch in keinem relevanten Schritt gekippt wurde. Es sei E das Ereignis, dass eine Phase aus $\ell := \lfloor (1-\varepsilon)(n^2/8-1) \ln n \rfloor = \Omega(n^2 \log n)$ Schritten erfolgreich ist, d. h., dass die Phase alle Individuen aus P' erzeugt. Wir wollen die Wahrscheinlichkeit von E abschätzen. Dazu sei A_i das Ereignis, dass das Bit an Position $i \in I$ wenigstens einmal in einem relevanten Schritt der Phase kippt. Damit E eintritt, ist es notwendig, dass alle bis auf eines der Ereignisse $A_1, \ldots, A_{\lfloor n/8 \rfloor}$ eintreten. Also gilt

$$\mathrm{Prob}(E) \le \mathrm{Prob}\big((A_2 \cap \cdots \cap A_{\lfloor n/8 \rfloor}) \cup (A_1 \cap A_3 \cap \cdots \cap A_{\lfloor n/8 \rfloor}) \cup \cdots \cup (A_1 \cap \cdots \cap A_{\lfloor n/8 \rfloor - 1})\big).$$

Da alle Ereignisse A_i gleiche Wahrscheinlichkeit haben, folgt mit dem Multiplikationssatz (vgl. Anhang A)

$$\begin{aligned} \mathrm{Prob}(E) &\le \lfloor n/8 \rfloor \cdot \mathrm{Prob}(A_1 \cap \cdots \cap A_{\lfloor n/8 \rfloor - 1}) \\ &= \lfloor n/8 \rfloor \cdot \prod_{i=1}^{\lfloor n/8 \rfloor - 1} \mathrm{Prob}(A_i \mid A_1 \cap \cdots \cap A_{i-1}). \qquad (*) \end{aligned}$$

Die Bedingung $A_1 \cap \dots \cap A_{i-1}$ bedeutet, dass es in der Phase mindestens $i-1$ relevante Schritte gibt, in denen das Bit an Position i *nicht* gekippt wird. Also gilt $\text{Prob}(A_i \mid A_1 \cap \dots \cap A_{i-1}) \le \text{Prob}(A_i)$. Die Wahrscheinlichkeit, dass x^{links} ausgewählt wird, ist in jedem Schritt höchstens $8/n$ und die Wahrscheinlichkeit, dass das Bit an Position i kippt, ist $1/n$. Daraus folgt

$$\text{Prob}(A_i) \;\le\; 1 - \left(1 - \frac{8}{n^2}\right)^{\ell} \;\le\; 1 - \frac{1}{n^{1-\varepsilon}}.$$

Schließlich erhalten wir daraus

$$\text{Prob}(E) \;\le\; \frac{n}{8} \cdot \left(1 - \frac{1}{n^{1-\varepsilon}}\right)^{\lfloor n/8 \rfloor - 1} \;=\; \mathrm{e}^{-\Omega(n^{\varepsilon})},$$

womit die Behauptung bewiesen ist. □

In der Analyse des globalen SEMO müssen wir nun den Einfluss von Schritten berücksichtigen, in denen viele Bits kippen. Das verkompliziert die Analyse. Es gelingt nur noch eine etwas schwächere untere Schranke.

Theorem 13.7. *Die Optimierzeit des globalen SEMO für MOCO ist mit einer Wahrscheinlichkeit von $1 - O(1/\sqrt{n})$ mindestens $\Omega(n^2 \log n)$.*

Auch im folgenden Beweis sind einige der vorkommenden Konstanten in gewissen Grenzen willkürlich gewählt. Bereits das Polynom $1/\sqrt{n} = n^{-1/2}$ in Theorem 13.7 wurde wegen seiner Einfachheit gewählt. Tatsächlich beweisen wir sogar eine Wahrscheinlichkeit von $1 - O(n^{-5/8} \log n)$.

Beweis zu Theorem 13.7. Wir behaupten, dass folgende Eigenschaft mindestens mit einer Wahrscheinlichkeit von $1 - O(1/\sqrt{n})$ für einen Lauf des globalen SEMO gilt: Es gibt einen Zeitpunkt, zu dem die Anzahl der paretooptimalen Individuen $\Omega(n)$ ist und $n^{1/8} \le |x^{\text{links}}|$ gilt, oder die Optimierzeit ist ohnehin $\Omega(n^2 \log n)$.

Unser erstes Ziel ist es, diese Behauptung zu beweisen. Der Startpunkt hat mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ mindestens $n/4$ 1-Bits. Wenn dies so eintritt, warten wir auf den ersten Zeitpunkt, zu dem ein Individuum mit höchstens $3n^{1/8}$ 1-Bits entsteht. Falls die Wartezeit mindestens $n^2 \log n$ Schritte umfasst, ist für die Behauptung nichts mehr zu zeigen. Andernfalls betrachten wir nach dem Ende der Wartezeit eine Phase aus $O(n^2)$ Schritten. Zu jedem Zeitpunkt innerhalb dieser Phase bezeichne j die kleinste Zahl aus der Menge $\{|x^{\text{links}}|, \dots, n/4\}$, sodass $x^{j-1} \in P$ und $x^j \notin P$ gilt. Das bedeutet anschaulich, dass j die Bitposition des linken Randes der am weitesten links liegenden Lücke innerhalb der bereits erzeugten Individuen aus $\{x^{\text{links}}, \dots, x^{n/4}\}$ angibt. Offenbar ist j nicht definiert, wenn es keine Lücke gibt. Wenn dies der Fall ist, enthält die Population mindestens $n/4 - 3n^{1/8} = \Omega(n)$ paretooptimale Individuen und wir können die Phase als vorzeitig (erfolgreich) beendet ansehen.

Innerhalb der Phase heißt nun ein Schritt *relevant*, wenn er x^{j-1} oder eines der beiden Individuen mit den wenigsten 1-Bits aus Region I zur Mutation auswählt. (Zu den beiden Individuen mit den wenigsten 1-Bits kann selbstverständlich auch x^{j-1} gehören.) Diese Definition relevanter Schritte ist an dieser Stelle schwer zu motivieren. Wir werden gleich sehen, dass mit hoher Wahrscheinlichkeit nur die relevanten Schritte $|x^{\text{links}}|$ verringern. Die Wahrscheinlichkeit für einen relevanten Schritt ist $\Omega(1/n)$. Eine genügend lang gewählte Phase aus $O(n^2)$ Schritten enthält also mit einer Wahrscheinlichkeit von $1-\mathrm{e}^{-\Omega(n)}$ mindestens $n/28$ relevante Schritte. Die Phase kann spätestens nach dem $\lfloor n/28 \rfloor$-ten relevanten Schritt als beendet angesehen werden. Jeder relevante Schritt darin wählt mit einer Wahrscheinlichkeit von mindestens $1/3$ das Individuum x^{j-1} aus. Also erzeugt jeder relevante Schritt mit einer Wahrscheinlichkeit von $\Omega(1)$ ein neues Individuum x^j und verkleinert somit die Lücke. Daraus folgt, dass mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n)}$ zu dem Zeitpunkt, an dem die Phase endet, in der Population $\Omega(n)$ paretooptimale Individuen existieren.

Wir müssen noch beweisen, dass es unwahrscheinlich ist, dass bis zum Ende der Phase ein Individuum mit weniger als $n^{1/8}$ 1-Bits erzeugt wird – weder in den $O(n^2)$ Schritten in der Phase noch in den $O(n^2 \log n)$ Schritten vor dem Beginn der Phase. Dazu werden wir zeigen, dass mit hoher Wahrscheinlichkeit sämtliche Individuen mit weniger als $2n^{1/8}$ 1-Bits aus Mutationen der beiden Individuen mit den beiden wenigsten 1-Bits entstehen und in den entsprechenden Schritten $|x^{\text{links}}|$ jeweils um nicht mehr als 2 sinkt.

Ein Schritt heißt *unsauber*, wenn er

– ein Individuum mit höchstens $3n^{1/8}$ 1-Bits wählt und bei der Mutation die Zahl der 1-Bits um mindestens 3 senkt oder

– ein Individuum mit mehr als $3n^{1/8}$ 1-Bits wählt und wenigstens $n^{1/8}$ Bits kippt.

Sonst heißt ein Schritt *sauber*. Aus Lemma 13.3 folgt, dass die Wahrscheinlichkeit $n^{-\Omega(n^{1/8})}$ ist, dass das Ereignis, das der zweiten Möglichkeit entspricht, bis zum Ende der Phase wenigstens einmal vorkommt. Das Ereignis, das zu der ersten Möglichkeit gehört, kann offenbar erst in der Phase aus $O(n^2)$ Schritten eintreten. Wir schätzen nun für jedes konstante $\alpha > 0$ und jedes $\varepsilon \in \,]0,1[$ die Wahrscheinlichkeit ab, dass ein einzelner Mutationsschritt die Zahl der 1-Bits um wenigstens i senkt, wenn das gewählte Individuum höchstens $m \le \alpha n^{\varepsilon}$ 1-Bits hat. Diese ist höchstens

$$\sum_{i\le k\le m} \binom{m}{k}\left(\frac{1}{n}\right)^k \;\le\; \sum_{i\le k\le m}\left(\frac{m}{n}\right)^k \;\le\; \left(\frac{m}{n}\right)^i \cdot \sum_{k\ge 0}\left(\frac{m}{n}\right)^k$$
$$= \left(\frac{\alpha}{n^{1-\varepsilon}}\right)^i \cdot \sum_{k\ge 0}\left(\frac{\alpha}{n^{1-\varepsilon}}\right)^k \;=\; O\big(\alpha^i n^{-(1-\varepsilon)i}\big). \quad (*)$$

(In der Abschätzung steht k für die Anzahl kippender 1-Bits.) Also senkt ein Schritt, der ein Individuum mit höchstens $3n^{1/8}$ 1-Bits wählt, die Anzahl der 1-Bits mit einer Wahrscheinlichkeit von nur $O(n^{-21/8})$ um mindestens 3. Dieses Ereignis kommt in der Phase nur mit einer Wahrscheinlichkeit von $O(n^{-21/8}) \cdot O(n^2) = O(n^{-5/8})$ vor. Jetzt ist es klar, dass mit hoher Wahrscheinlichkeit nur die relevanten Schritte den Wert $|x^{\text{links}}|$ senken.

Im Folgenden arbeiten wir unter der Bedingung, dass alle Schritte bis zum Ende der Phase sauber sind. Die Bedingung impliziert, dass Individuen mit höchstens $2n^{1/8}$ 1-Bits von Individuen mit höchstens $3n^{1/8}$ 1-Bits abstammen. Außerdem sinkt $|x^{\text{links}}|$ um maximal 2, wenn Individuen mit höchstens $3n^{1/8}$ 1-Bits ausgewählt werden. Folglich sind alle Individuen mit höchstens $2n^{1/8}$ 1-Bits in relevanten Schritten entstanden, die ein Individuum mit höchstens $3n^{1/8}$ 1-Bits ausgewählt haben. Für genügend große n gilt $\alpha/(n^{1-\varepsilon}) \leq 1/2$, sodass die rechte Seite von Ungleichung $(*)$ durch $2\alpha^i n^{-(1-\varepsilon)}$ beschränkt ist. Also senkt ein Schritt, der ein Individuum mit höchstens $3n^{1/8}$ 1-Bits auswählt, die Anzahl der 1-Bits um weniger als 3 mit einer Wahrscheinlichkeit von mindestens $1 - 54n^{-21/8}$. So ein Schritt senkt die Anzahl der 1-Bits jedoch um mindestens 1 mit einer Wahrscheinlichkeit von höchstens $6n^{-7/8}$. Demnach ist für einen Schritt, der ein Individuum mit höchstens $3n^{1/8}$ 1-Bits auswählt, die (bedingte) Wahrscheinlichkeit, dass die Anzahl der 1-Bits sinkt, gegeben, dass er sauber ist, für genügend große n höchstens $6n^{-7/8}/(1 - 54n^{-21/8}) \leq 7n^{-7/8}$. Nun können höchstens alle $\lfloor n/28 \rfloor$ relevanten Schritte der Phase ein Individuum mit höchstens $3n^{1/8}$ 1-Bits auswählen. Mit einer Wahrscheinlichkeit von $1 - \mathrm{e}^{-\Omega(n^{1/8})}$ senken alle sauberen relevanten Schritte $|x^{\text{links}}|$ höchstens $\lfloor (1/2)n^{1/8} \rfloor$-mal um höchstens 2. Das bedeutet, dass $|x^{\text{links}}|$ am Ende der Phase noch mindestens einen Wert von $n^{1/8}$ hat. Die größte „Fehlerwahrscheinlichkeit“, die uns bisher begegnet ist, ist $O(n^{-5/8})$. Damit ist die zu Beginn formulierte Behauptung bewiesen.

Für den restlichen Teil dieses Beweises nehmen wir an, dass bereits eine Situation gemäß der Behauptung zu Beginn vorliegt. Das soll heißen, dass $n^{1/8} \leq |x^{\text{links}}|$ gilt und dass für eine gewisse Konstante c die Anzahl der Individuen aus Region I durch cn nach unten beschränkt ist. Wir können zusätzlich auch annehmen, dass schon $|x^{\text{links}}| \leq 2n^{1/8}$ gilt. Der Grund ist, dass Schritte, die mindestens $n^{1/8}$ Bits kippen, mit überwältigender Wahrscheinlichkeit ausgeschlossen werden können (siehe Lemma 13.3), sodass die angenommene Situation sonst später eintritt.

Unser zweites Ziel ist zu zeigen, dass eine (neue) Phase bestehend aus $\ell := \lfloor (1/16)(cn^2 - 1)\ln n \rfloor = \Theta(n^2 \log n)$ Schritten nur mit einer Wahrscheinlichkeit von $O(1/\sqrt{n})$ das Individuum x^1 erzeugt. Dazu behaupten wir, dass die Phase mit hoher Wahrscheinlichkeit die folgenden Eigenschaften hat:

1. Aus keinem Individuum mit mehr als $3n^{1/8}$ 1-Bits wird ein Nachkomme mit weniger als $2n^{1/8}$ 1-Bits erzeugt.

2. Kein Schritt, der $x \neq x^{\text{links}}$ auswählt, senkt $|x^{\text{links}}|$.

Die erste Behauptung folgt aus Lemma 13.3. Für die zweite Behauptung argumentieren wir wie folgt. Es kann nur $O(n^{1/8})$ Individuen mit höchstens $n^{1/8}$ 1-Bits in der Population geben, sodass irgendeines davon nur mit einer Wahrscheinlichkeit von $O(n^{1/8})/\Omega(n) = O(n^{-7/8})$ ausgewählt werden kann. Für ein Individuum $x \neq x^{\text{links}}$ müssen mindestens zwei 1-Bits verschwinden, damit $|x^{\text{links}}|$ sinkt. Aus Ungleichung $(*)$ folgt, dass die Wahrscheinlichkeit für eine solche Mutation höchstens $O(n^{-14/8})$ ist. Also verletzt ein einzelner Schritt die zweite Bedingung höchstens mit einer Wahrscheinlichkeit von $O(n^{-21/8})$. Dieses Ereignis tritt in der Phase also höchstens mit einer Wahrscheinlichkeit von $O(n^{-5/8} \log n) = O(1/\sqrt{n})$ auf. Insgesamt hält die Phase beide Bedingungen mit einer Wahrscheinlichkeit von mindestens $1 - O(1/\sqrt{n})$ ein. Das bedeutet, dass dann jeder Schritt, der $|x^{\text{links}}|$ senkt, das bisherige Individuum x^{links} mutiert.

Wir arbeiten nun unter der Bedingung, dass die Phase beide Eigenschaften aufweist. Man beachte, dass diese Bedingung nicht die Wahrscheinlichkeit beeinflusst, dass x^{links} ausgewählt wird. Es wird dadurch auch keine Bedingung an die Mutation in Schritten gestellt, die x^{links} auswählen. Wir verfahren nun so wie am Ende des Beweises zu Theorem 13.6. Zu Beginn der Phase hat x^{links} mindestens $n^{1/8}$ 1-Bits. Wir wählen nun beliebig $x^{\lfloor n^{1/8} \rfloor}$ Bitpositionen von 1-Bits von x^{links} aus. Diese Bitpositionen seien o. B. d. A. $I := \{1, \dots, \lfloor n^{1/8} \rfloor\}$. Wegen der zweiten Eigenschaft kann jedes neue x^{links} mit weniger 1-Bits nur aus x^{links} entstehen. Auf die gewohnte Weise können wir die Bitpositionen des jeweils aktuellen Individuums mit den Bitpositionen des ursprünglichen Individuums x^{links} identifizieren. Es sei $P' := \{x^1, \dots, x^{\lfloor n^{1/8} \rfloor}\}$. Mindestens die in P' enthaltenen Individuen fehlen zu Beginn der Phase noch. Nun sei A_i das Ereignis, dass das Bit an Position $i \in I$ wenigstens einmal in den Schritten der Phase gekippt wird, die x^{links} auswählen. Da $1/n$-Standardmutationen jedes Bit unabhängig kippen, folgt

$$\text{Prob}(A_i \mid A_1 \cap \cdots \cap A_{i-1}) \;=\; \text{Prob}(A_i).$$

Es gilt

$$\text{Prob}(A_i) \;\leq\; 1 - \left(1 - \frac{1}{cn^2}\right)^{\ell} \;\leq\; 1 - n^{-1/16}.$$

Die Wahrscheinlichkeit $\text{Prob}(E)$, dass in der Phase alle Individuen aus P' erzeugt werden, ist damit höchstens (vgl. Ungleichung $(*)$ im Beweis zu Theorem 13.6)

$$\text{Prob}(E) \;\leq\; n^{1/8} \cdot \left(1 - \frac{1}{n^{1/16}}\right)^{\lfloor n^{1/8} \rfloor - 1} \;=\; \mathrm{e}^{-\Omega(n^{1/16})}.$$

Das Individuum x^1 wird in der zweiten Phase also höchstens mit einer Wahrscheinlichkeit von $O(1/\sqrt{n})$ erzeugt. Zusammen mit der Behauptung vom Beginn des Beweises folgt nun die Aussage des Theorems. $\square$

Das folgende Korollar fasst abschließend das Ergebnis dieses Kapitels zusammen, das zu Beginn lediglich eine Vermutung war. Es folgt aus Theorem 13.5, Theorem 13.6 und Theorem 13.7.

Korollar 13.8. *Mit einer Wahrscheinlichkeit von* $1 - o(1)$ *ist die Optimierzeit des lokalen und des globalen SEMO für MOCO* $\Theta(n^2 \log n)$.

14 Zusammenfassung

Es wurde die erwartete Optimierzeit des einfachen, aber grundlegenden mehrkriteriellen evolutionären Algorithmus SEMO (Simple Evolutionary Multi-objective Optimizer) untersucht. Der lokal suchende SEMO und der global suchende SEMO können als Gegenstücke für mehrkriterielle Probleme zur randomisierten lokalen Suche bzw. zum (1+1)-EA aufgefasst werden.

Die erwartete Optimierzeit des globalen SEMO ist im Worst Case nicht größer als die des (1+1)-EA. Zudem gehören einige ein- und zweikriterielle Probleme mit zu den schwierigsten Problemen für den globalen SEMO, wenn als Maßstab die erwartete Optimierzeit verwendet wird. Probleme mit großer Zielraumdimension sind für diese randomisierte Suchheuristik nicht schwieriger. Es wurde ein zweikriterielles Problem vorgestellt, mit dem erwartete Optimierzeiten von $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(n^4)$ usw. bis zum Worst Case von $\Theta(n^n)$ erreicht werden können.

Für das Problem LOTZ wurde gezeigt, dass $1/n$-Standardmutationen die Optimierzeit gegenüber dem lokalen Suchoperator, der nur *ein* Bit kippt, nicht verschlechtern. Die Untersuchung des Problems MOCO hat ergeben, dass die in der Literatur geäußerte Vermutung einer erwarteten Optimierzeit von $\Theta(n^2 \log n)$ sowohl für den lokalen wie auch globalen SEMO unzutreffend ist. Die erwartete Optimierzeit ist exponentiell groß bzw. nicht endlich. Es konnte aber auch gezeigt werden, dass ein Lauf mit hoher Wahrscheinlichkeit eine Optimierzeit von $\Theta(n^2 \log n)$ benötigt.

Teil III

Über die Fehlerrate deterministischer Seitenwechselstrategien

15 Problemstellung, Motivation und Modellbildung

In den ersten beiden Abschnitten dieses Kapitels werden zunächst das Seitenwechselproblem und übliche Seitenwechselstrategien vorgestellt. Danach diskutieren wir bekannte Resultate der kompetitiven Analyse zum Seitenwechselproblem, aus denen sich die Motivation für die Analyse aus einem anderen Blickwinkel ergibt. Dazu werden im letzten Abschnitt dieses Kapitels zwei Modelle eingeführt, in denen das Seitenwechselproblem später analysiert werden soll.

15.1 Das Seitenwechselproblem

Problembeschreibung

Das Seitenwechselproblem (engl. *paging problem*) wird in einem Szenario formuliert, das ein vereinfachtes Modell der Vorgänge in einem Rechner ist, der das Konzept des virtuellen Speichers umsetzt. Wir betrachten zunächst, wie die Speicherzugriffe eines Prozesses in so einem Rechner bearbeitet werden. Dabei gehen wir von einem Rechner aus, dessen realer Speicher in zwei Speicherebenen eingeteilt ist. Die erste Ebene ist ein großer, aber langsamer Hintergrundspeicher (z. B. ein Plattenspeicher), die zweite Ebene ist ein schneller, aber kleiner Speicher (z. B. der Arbeitsspeicher des Rechners). Beide Speicherebenen sind in Blöcke gleicher Größe eingeteilt, die auch Kacheln (engl. *frames*) genannt werden.

Der virtuelle Speicher eines Prozesses ist ebenfalls in Blöcke eingeteilt, die Seiten (engl. *pages*) heißen. In der Regel haben Seiten und Kacheln dieselbe Größe, sodass eine Seite des virtuellen Speichers in einer Kachel des realen Speicher gespeichert werden kann. Ein Prozess erzeugt bei seiner Abarbeitung durch den Rechner eine Folge von Speicherzugriffen, die auf die verschiedenen Seiten des virtuellen Speichers des Prozesses lesend oder schreibend zugreifen. Wir nennen einen Zugriff auf eine Adresse innerhalb einer Seite des virtuellen Speichers eine *Anfrage* an diese Seite. Um eine Anfrage bedienen zu können, bildet der Rechner die Seiten des virtuellen Speichers auf Kacheln im realen Speicher ab. Da ein Prozess meistens nur einen kleinen Teil seines Adressraumes nutzt, wird eine Seite in der Regel erst in einer Kachel des realen Speichers vorgehalten, nachdem sie zum ersten Mal angefragt wurde.

Man beobachtet bei realen Prozessen, dass diese meist nicht unstrukturiert oder zufällig auf Seiten zugreifen, sondern dass die Anfragefolge ein hohes Maß an *Anfragelokalität* aufweist. Damit meint man die Eigenschaft, dass n aufeinander folgende Anfragen wesentlich weniger als n *verschiedene* Seiten anfragen. Man erklärt dies folgendermaßen. Die Anfragen eines Prozesses entstehen zum einen durch lesende Zugriffe im virtuellen Speicher auf die Maschinenbefehle des ausgeführten Programms und zum anderen durch lesende und schreibende Zugriffe auf die in den Befehlen verarbeiteten Daten. Aufeinander folgende Befehle eines Programms sind im Speicher meistens hintereinander abgelegt, sodass ganze Programmfragmente in derselben Seite liegen. Daher fordern die Anfragen, die durch das Laden des nächsten auszuführenden Maschinenbefehls entstehen, sehr häufig dieselbe Seite an wie schon im vorhergegangenen Programmschritt. Weil Programme typischerweise Schleifen enthalten und bei wiederholter Ausführung desselben Schleifenrumpfs häufig dieselben Daten verarbeiten, kommt es auch hier in kurzen Zeitabständen zu Anfragen an dieselbe Seite. Ein weiterer Grund ist sicher auch die Art und Weise, wie Datenstrukturen im virtuellen Speicher abgelegt werden. So werden etwa benachbarte Einträge eines Arrays meistens auf benachbarte Adressbereiche abgebildet. Falls mehrere Arrayeinträge auf eine Seite passen, dann erzeugt schon ein sequentieller Lauf durch ein Array mehrmals dieselbe Anfrage in Folge.

Um von der beschriebenen Anfragelokalität zu profitieren, werden Anfragen nicht unmittelbar aus dem Hintergrundspeicher bedient, sondern immer aus dem schnellen Speicher. Zwar sind alle (jemals zugegriffenen) Seiten eines Prozesses in den Kacheln des Hintergrundspeichers abgelegt, im schnellen Speicher werden jedoch zusätzlich Kopien ausgewählter Seiten vorgehalten. Bei einer Anfrage an eine Seite wird zunächst geprüft, ob eine Kopie der Seite im schnellen Speicher verfügbar ist. Wenn dies der Fall ist, kann die Schreib- oder Leseoperation sofort auf dieser Kopie im schnellen Speicher ausgeführt werden. Andernfalls spricht man von einem *Seitenfehler*. Es muss dann zunächst die geforderte Seite aus dem Hintergrundspeicher in den schnellen Speicher kopiert werden. Dieser Vorgang ist im Vergleich zum Zugriff auf den schnellen Speicher erheblich zeitaufwendiger. (Arbeitsspeicherzugriffe liegen heute etwa im Bereich weniger Nanosekunden (10^{-9} Sekunden), während Plattenzugriffe im Bereich weniger Millisekunden (10^{-3} Sekunden) liegen. Zudem kann die Zugriffszeit im Fall eines Plattenspeichers als Hintergrundspeicher auch stark schwankend sein.) Sobald die Kapazität des schnellen Speichers erschöpft ist, muss entschieden werden, der Inhalt welcher Kachel des schnellen Speichers durch die angeforderte Seite ersetzt werden soll. Gegebenenfalls muss die bisher dort vorgehaltene Seite zuvor noch in den Hintergrundspeicher kopiert werden, weil ihr Inhalt inzwischen verändert wurde. Wir können also vom Ersetzen einer Seite im schnellen Speicher durch eine Seite aus dem Hintergrundspeicher oder auch vom Wechseln der Seiten zwischen den beiden Speicherebenen sprechen.

Das *Seitenwechselproblem* besteht nun darin, zu entscheiden, zu welchen Zeitpunkten welche Seiten zwischen den beiden Speicherebenen ausgetauscht werden sollen. Eine *Seitenwechselstrategie* (auch *Seitenersetzungsstrategie,* engl. *paging algorithm*) trifft diese Entscheidungen.

Das formale Modell des Problems

Beim Seitenwechselproblem, wie wir es im Folgenden verstehen wollen, vereinfachen wir das beschriebene Szenario. Wir unterscheiden nicht länger zwischen dem virtuellen Speicher des Prozesses und dem realen Hintergrundspeicher des Rechners, sondern setzen den virtuellen Speicher mit dem Hintergrundspeicher gleich und sprechen einfach nur noch von Seiten $p_1, \ldots, p_N$. Diese verweilen entweder im schnellen Speicher oder im Hintergrundspeicher. Der schnelle Speicher, im Folgenden auch *Cache* genannt, soll k Seiten aufnehmen können. Die Anfragesequenz $\sigma = \sigma_1, \ldots, \sigma_n$ eines Prozesses stellen wir uns als eine endliche Folge von Indizes aus $\{1, \ldots, N\}$ vor, sodass im Schritt $t \in \{1, \ldots, n\}$ die Anfrage σ_t an eine Seite p_{σ_t} geht. Eine Seitenwechselstrategie darf in jedem Schritt t als Reaktion auf die Anfrage σ_t das Ersetzen höchstens einer Seite veranlassen. Das bedeutet, eine Seite aus dem Cache wird in den Hintergrundspeicher bewegt und die angeforderte Seite wird aus dem Hintergrundspeicher in den Cache bewegt, wo sie den frei gewordenen Platz einnimmt. Diese Operation wird innerhalb des t-ten Schritts ausgeführt. Wenn jedoch zu Beginn des t-ten Schritts die in σ_t angefragte Seite nicht im schnellen Speicher vorgehalten wird, verursacht dies einen Seitenfehler. Das dann erforderliche Ersetzen einer Seite im Cache durch die angefragte Seite ist mit Kosten belegt. Diese Kosten modellieren die Wartezeit, die dem Prozess entstünde.

Das grundsätzliche Ziel einer Seitenwechselstrategie ist es, die Gesamtkosten zum Abarbeiten einer Sequenz σ zu minimieren. Dazu werden verschiedene Kostenmodelle betrachtet. Die beiden wichtigsten Kostenmodelle sind das Seitenfehlermodell (engl. *page fault model*) und das Zugriffskostenmodell (engl. *full access cost model*). Die Aufgabe besteht nun zum einen darin, geeignete Seitenwechselstrategien zu entwerfen, und zum anderen, diese zu analysieren. In diesem Teil verschreiben wir uns nur der zweiten Aufgabe.

Im *Seitenfehlermodell* wird jeder Seitenwechsel mit Kosten 1 belegt unabhängig vom Zeitpunkt, zu dem der Wechsel veranlasst wird. Man kann leicht einsehen, dass in diesem Kostenmodell jede Seitenwechselstrategie so abgeändert werden kann, dass sie nur bei Seitenfehlern einen Seitenwechsel veranlasst, ohne dass sich dabei für irgendeine Anfragesequenz die Kosten erhöhen. Solche Strategien, die nur bei Bedarf, d. h. bei einem Seitenfehler, einen Seitenwechsel vornehmen, heißen Demand-Paging-Strategien (engl. *demand paging algorithms*). Wenn wir Demand-Paging-Strategien betrachten, erklärt sich auch der Name „Seitenfehlermodell“, denn nur die Anfragen, die Seitenfehler verursachen, verursachen Kosten. Das Seitenfehlermodell ist für Demand-Paging-Strategien eng verwandt mit der *Fehlerrate*, dem Quotienten aus der Zahl der Seitenfehler und der Länge n der Anfragesequenz. Aus der Fehlerrate können auf die offensichtliche Weise die Kosten errechnet werden, wenn n bekannt ist. Wir werden später nur noch Demand-Paging-Strategien und deren Fehlerrate untersuchen.

Im *Zugriffskostenmodell* wird dem Umstand Rechnung getragen, dass Zugriffe auf den schnellen Speicher in Wirklichkeit auch etwas Zeit benötigen und daher nicht völlig kostenfrei sein sollten. Man veranschlagt in diesem Modell Kosten 1 für jeden Seitenfehler und Kosten $1/p$ für jede Anfrage, die aus dem schnellen Speicher bedient werden kann. Für p gegen unendlich erhält man dann im Grenzfall wieder das Seitenfehlermodell. Offenbar können auch in diesem Modell aus der Fehlerrate einer Demand-Paging-Strategie die Kosten bestimmt werden, wenn n bekannt ist.

15.2 Seitenwechselstrategien

Man kann Seitenwechselstrategien danach unterscheiden, wie sie die Anfragesequenz verarbeiten. Von einer *Online*-Strategie verlangen wir, dass sie für ihre Entscheidungen bei der Bearbeitung der Anfrage σ_t nur die Kenntnis der Anfragen $\sigma_1, \ldots, \sigma_t$ einfließen lässt. Einer Online-Strategie bleiben zum Zeitpunkt t die zukünftigen Anfragen $\sigma_{t+1}, \ldots, \sigma_n$ verborgen, d. h., die Anfragesequenz wird ihr nur schrittweise eröffnet. In jedem Schritt werden ihr Entscheidungen abverlangt, die sie im Nachhinein nicht mehr revidieren kann. Wenn einer Strategie für ihre Entscheidungen zum Zeitpunkt t auch die Kenntnis der zukünftigen Anfragen $\sigma_{t+1}, \ldots, \sigma_n$ zur Verfügung steht, sprechen wir von einer *Offline*-Strategie.

Im Offline-Szenario kann das Seitenwechselproblem als gelöst angesehen werden, da eine einfache Strategie bekannt ist, die eine kostenminimale Lösung berechnet (siehe unten). Weitaus interessanter ist das Online-Szenario. Es spiegelt die Situation beim Abarbeiten eines Prozesses wider, denn die zukünftigen Anfragen sind nur schwer vorhersehbar. Dazu müsste man den Prozess, der die Anfragen erzeugt, simulieren und würde so dessen Rechnung vorwegnehmen. Aus diesem Grund arbeiten Seitenwechselstrategien im Online-Szenario heuristisch. Die heuristisch getroffenen Entscheidungen können sich im Nachhinein als ungünstig erweisen. Die Hoffnung ist, dass sie bei typischen Anfragesequenzen nicht zu viele „vermeidbare“ Seitenfehler erzeugen. Wir können solche Strategien als *Online-Heuristiken* bezeichnen. Im Folgenden bleiben wir jedoch bei dem Begriff *Online-Strategie* (oder einfach nur *Strategie*). Es ist üblich und erscheint angemessen, diese häufig sehr einfachen Algorithmen, die oft nur wenig mehr als eine Regel sind, als Strategien zu bezeichnen.

Normalerweise wird Online-Strategien nur wenig Speicherplatz zur Verfügung gestellt, sodass sich keinesfalls alle bisherigen Anfragen $\sigma_1, \ldots, \sigma_t$ gemerkt werden können. Der Gesamtplatzbedarf praktisch relevanter Online-Strategien ist meistens linear in der Cachegröße k. Obwohl die Laufzeit zum Bearbeiten einer Anfrage nur eine untergeordnete Rolle spielt – die Wartezeiten durch Seitenfehler sind viel entscheidender –, benötigen diese in der Regel nur Zeit $O(k)$. Diese Zeit gestattet es, den Cache vollständig zu inspizieren. Nicht wenige Strategien benötigen lediglich Zeit $O(1)$ pro Anfrage. Dies führt zu sehr einfachen Online-Strategien.

Des Weiteren kann man *deterministische* von *randomisierten* Strategien unterscheiden. In dieser Arbeit untersuchen wir ausschließlich deterministische Seitenwechselstrategien. Das sind Strategien, die ihre Entscheidung, zu einem Zeitpunkt eine Seite im Cache zu ersetzen, nur von der Anfragesequenz abhängig machen dürfen. Im Gegensatz zu randomisierten Strategien ist es deterministischen Strategien nicht gestattet, Zufallsbits auszuwerten. Die Unterscheidung von randomisierten und deterministischen Strategien ist allerdings nur im Online-Szenario gut zu motivieren, denn es ist eine einfache deterministische Offline-Strategie bekannt, die eine Lösung mit geringstmöglichen Kosten berechnet. Im Online-Szenario hingegen kann man versuchen, sich durch zufällige Entscheidungen gegen Worst-Case-Anfragesequenzen zu wappnen.

Die folgenden deterministischen Strategien werden in dieser Arbeit untersucht. Darunter sind die vermutlich bekanntesten Strategien LRU und FIFO. Wir machen hier keine Annahmen über den Zustand des Caches zu Beginn. Man kann sich den Cache zu Beginn leer vorstellen oder auch mit beliebigen Seiten gefüllt.

FIFO (First In First Out) ersetzt bei einem Seitenfehler diejenige Seite im Cache, deren letzter Einlagerungszeitpunkt am weitesten zurückliegt. Solange es im Cache Seiten gibt, die noch nie angefragt wurden, werden diese zuerst ersetzt.

Markierungsstrategien (engl. *marking algorithms*) sind eine Klasse von Seitenwechselstrategien, die wie folgt beschrieben werden kann. Die Strategien dieser Klasse arbeiten in Runden. Jede Seite im Cache kann markiert oder unmarkiert sein. Zu Beginn einer Runde werden alle Markierungen gelöscht. Bei einem Seitenfehler wird eine unmarkierte Seite durch die angeforderte Seite ersetzt und markiert; ist die angeforderte Seite schon im Cache vorrätig, dann wird sie nur markiert. Sobald alle Seiten im Cache markiert sind, können Seitenfehler nicht mehr auf diese Weise behandelt werden. Dann beginnt diejenige Anfrage, die den nächsten Seitenfehler verursacht, eine neue Runde. Das bedeutet, bevor der Seitenfehler behandelt wird, werden zunächst alle Markierungen gelöscht.

Die beiden folgenden Strategien sind spezielle Markierungsstrategien.

LRU (Least Recently Used) ersetzt bei einem Seitenfehler diejenige Seite durch die angeforderte Seite, deren letztes Vorkommen in der Anfragesequenz am weitesten zurückliegt. Solange es im Cache Seiten gibt, die noch nie angefragt wurden, werden diese zuerst ersetzt.

Man kann zeigen, dass LRU eine deterministische Markierungsstrategie ist (siehe z. B. Borodin und El-Yaniv (1998)).

FWF (Flush When Full) ist eine vergleichsweise primitive deterministische Strategie. Zu Beginn einer Runde wird der Cache geleert („flush“), d. h., alle

Seiten werden aus dem Cache entfernt und ggf. in den Hintergrundspeicher kopiert. Solange im Cache noch Platz für neue Seiten ist, werden Seitenfehler durch Einlagern der verlangten Seite behandelt. Verursacht eine Anfrage bei gefülltem Cache einen Seitenfehler, dann beginnt diese Anfrage eine neue Runde.

Offensichtlich passt sich FWF nicht in unserer Modell des Seitenwechselproblems ein, da zu Beginn einer Runde bis zu k Seiten auf einmal in den Hintergrundspeicher kopiert werden müssen. Wenn wir aber anstatt den Cache zu leeren nur Markierungen löschen, dann können wir das Zurückschreiben einer Seite in den Hintergrundspeicher bis zum tatsächlichen Überschreiben der Seite im Cache verzögern und wir erhalten eine Markierungsstrategie. Wir nennen die Strategien, die nach dieser Idee vorgehen, die *Markierungsvarianten von FWF*. Die einzelnen Markierungsvarianten von FWF unterscheiden sich untereinander darin, wie sie bei einem Seitenfehler eine unmarkierte Seite auswählen.

LFD (Longest Forward Distance) ersetzt bei einem Seitenfehler diejenige Seite, deren nächstes Vorkommen in der Anfragesequenz am weitesten vorausliegt. Falls es im Cache Seiten gibt, die in der restlichen Anfragesequenz nicht mehr vorkommen, werden diese zuerst ersetzt.

Bei der Strategie LFD, die auf Belady (1966) zurückgeht, handelt es sich offensichtlich um eine Offline-Strategie. Man kann beweisen, dass LFD eine optimale Offline-Strategie ist (z. B. in Borodin und El-Yaniv, 1998). Das bedeutet, LFD berechnet für die gegebene Anfragesequenz eine Lösung mit minimalen Kosten.

So wie sie hier vorgestellt wurden, sind einige der Strategien genau genommen nicht eindeutig beschrieben. Es ist z. B. nicht festgelegt, *welche* Seite LFD bei einem Seitenfehler auswählt, wenn es im Cache Seiten gibt, die später nie mehr angefragt werden. Auf diese Details wird es im Folgenden nicht ankommen. Wir können uns diese „Lücken“ in der Beschreibung durch deterministische Entscheidungen ausgefüllt vorstellen.

Wir fassen zusammen. FIFO und alle Markierungsstrategien sind Online-Seitenwechselstrategien, während LFD eine Offline-Seitenwechselstrategie ist. Alle diese Strategien sind deterministische Demand-Paging-Strategien.

15.3 Ergebnisse der kompetitiven Analyse

Die von Sleator und Tarjan eingeführte kompetitive Analyse ist vielleicht die erfolgreichste Technik, um Online-Algorithmen zu analysieren. Das Seitenwechselproblem wiederum ist von Anfang an und intensiv im Rahmen der kompetitiven Analyse untersucht worden. Heute dient es in Lehrbüchern häufig als wichtigstes Beispiel für kompetitive Analysen (siehe z. B. Motwani und Raghavan (1995) und Borodin und El-Yaniv (1998)). Allein daher müssen diese Resultate hier Erwähnung finden. Nicht zuletzt aber auch, weil sie ein Teil der Motivation zu den später durchgeführten Analysen sind, deren Ergebnisse mit denen der kompetitiven Analyse verglichen werden sollen.

Beim Seitenwechselproblem kann man jeder deterministischen Strategie $\mathcal{A}$ bei jeder Anfrage einen Seitenfehler aufzwingen. Dazu braucht man in jedem Schritt nur eine Anfrage an eine Seite zu stellen, die zu der Zeit nicht im Cache vorgehalten wird. Im Seitenfehlermodell bedeutet das, dass die Strategie im schlimmsten Fall Kosten $|\sigma|$ hat, also die maximal möglichen Kosten. (Dabei bezeichnet $|\sigma|$ die Länge der Anfragesequenz σ.) Dieses Ergebnis ist jedoch wenig hilfreich, da es keinen Hinweis gibt, welche deterministischen Strategien besser sind als andere. Die Idee der kompetitiven Analyse ist nun folgende. Anstatt die Kosten von $\mathcal{A}$ für σ zu bestimmen, vergleicht man diese mit den Kosten $\mathrm{OPT}(\sigma)$ einer kostenminimalen (Offline-)Lösung. Dabei interessiert man sich nun nur noch dafür, um welchen Faktor die Kosten der Lösung von $\mathcal{A}$, im Folgenden mit $\mathcal{A}(\sigma)$ bezeichnet, größer sind als die Kosten $\mathrm{OPT}(\sigma)$ einer kostenminimalen Lösung. Hier wie auch im Folgenden ist immer das Seitenfehlermodell als Kostenmodell zugrunde gelegt.

Definition 15.1 (c-kompetitiv). *Eine Strategie $\mathcal{A}$ heißt c-kompetitiv, wenn es eine Konstante α gibt, sodass für alle endlichen Eingabesequenzen σ gilt*

$$\mathcal{A}(\sigma) \;\leq\; c \cdot \mathrm{OPT}(\sigma) + \alpha.$$

Normalerweise wächst sowohl $\mathcal{A}(\sigma)$ als auch $\mathrm{OPT}(\sigma)$ mit wachsender Sequenzlänge $|\sigma|$, sodass der Einfluss der Konstanten α mit wachsender Sequenzlänge verschwindet. Der Grund für die Konstante α liegt darin, dass z. B. beim Seitenwechselproblem die Anfangsbelegung des Caches Einfluss auf die entstehenden Kosten hat. Bei „vernünftigen“ Strategien sollte dieser Einfluss aber maximal eine additive Konstante sein, in dem diese Anfangseffekte zum Ausdruck kommen. Die Konstante α darf daher nicht von der initialen Cachebelegung oder der Eingabe σ abhängen (insbesondere auch nicht von der Länge von σ). Beim Seitenwechselproblem wird die Cachegröße k als konstant angesehen und in der Regel wird α hier von k abhängen.

In anderen Szenarien, z. B. wenn die Kosten $\mathcal{A}(\sigma)$ von vornherein durch Konstanten beschränkt sind, kann es sinnvoller sein, zum Begriff *streng c-kompetitiv* (engl. *strictly c-competitive*) überzugehen. Dann verlangt man, dass α nicht positiv sein darf. In diesem Fall ist c nichts anderes als eine garantierte Approximationsgüte für die Lösung, die $\mathcal{A}$ berechnet. Aber auch im Fall $\alpha > 0$ ist die Vorstellung hilfreich, dass c im Wesentlichen eine Approximationsgüte angibt.

Schließlich kann man einer Strategie $\mathcal{A}$ einen *Kompetitivitätsfaktor*[1] (engl. *competitive ratio*) zuweisen, indem man das Infimum über alle c bildet, für die $\mathcal{A}$ c-kompetitiv ist. Man bezeichnet eine Strategie als *kompetitiv*, wenn ihr Kompetitivitätsfaktor höchstens eine Konstante ist. Die kompetitive Analyse versucht, diesen Kompetitivitätsfaktor zu bestimmen – oder zumindest durch obere und untere Schranken einzugrenzen. Für viele wichtige Seitenwechselstrategien und für alle in Abschnitt 15.2 vorgestellten Strategien ist die genaue Bestimmung des Kompetitivitätsfaktors gelungen:

Alle deterministischen Online-Seitenwechselstrategien erreichen bestenfalls einen Kompetitivitätsfaktor von k (Sleator und Tarjan, 1985). Alle in Abschnitt 15.2 vorgestellten deterministischen Online-Strategien erreichen den Kompetitivitätsfaktor k; das sind FIFO und LRU (Sleator und Tarjan, 1985) und alle deterministischen Markierungsstrategien (Torng, 1998). Einige Online-Strategien haben sich als nicht kompetitiv erwiesen, d. h., ihr Kompetitivitätsfaktor ist durch keine Konstante beschränkt. Dazu gehören z. B. *Least Frequently Used* und *Last In First Out* (Sleator und Tarjan, 1985). Eine weiter gehende Darstellung der Ergebnisse findet man z. B. in Borodin und El-Yaniv (1998).

Gegen die kompetitive Analyse kann folgende berechtigte Kritik vorgetragen werden:

– Die kompetitive Analyse charakterisiert zwar einige Strategien als nicht kompetitiv und damit den kompetitiven Strategien unterlegen, innerhalb der Menge der kompetitiven Strategien führen sie jedoch nicht zu einer aussagekräftigen Bewertung. Zu viele, nämlich die meisten der gängigen Strategien wie FIFO, LRU und alle Markierungsstrategien, sind im Sinne der reinen kompetitiven Analyse optimale deterministische Online-Strategien, denn sie erreichen den kleinstmöglichen Kompetitivitätsfaktor k.

– Diese Ergebnisse entsprechen nicht den Erfahrungen aus der Praxis, wo zumindest LRU als sehr gut gilt, während das für beliebige Markierungsstrategien nicht zutrifft, z. B. FWF. Das liegt daran, dass die kompetitive Analyse beliebige Anfragesequenzen betrachtet, also auch solche, die in der Praxis nie vorkommen. Die Tatsache, dass Anfragesequenzen Anfragelokalität beinhalten, wird nicht berücksichtigt. In der Dissertation von Young (1991) (Seite 26) gibt es z. B. eine Studie, die nahelegt, dass die „empirische Kompetitivität“ vieler Strategien wohl häufig viel kleiner als k ist und in typischen Fällen vielleicht sogar nach oben durch eine kleine Konstante unabhängig von k beschränkt werden kann. So liegt in dem dort gegebenen Beispiel die gemessene Kompetitivität von LRU immer unterhalb von 2,5, während die von FIFO

[1] Im Deutschen gibt es kein allgemein übliches Äquivalent zu *competitive ratio*. Mit Blick auf Definition 15.1 scheint es zumindest genauso angemessen, von einem *Faktor* anstatt von einem *Quotienten* zu sprechen.

zwar immer größer als die von LRU ist, aber durch 4 beschränkt ist. Dies entspricht auch der allgemeinen Vermutung, dass LRU nicht schlechter (sondern meistens besser) als FIFO ist (z. B. in Borodin, Irani, Raghavan und Schieber (1995) sowie in Irani, Karlin und Phillips (1996)). Es gibt also große Unterschiede zwischen praktischen Erfahrungen und theoretischen Ergebnissen.

– Die kompetitive Analyse liefert nur eine relative Bewertung der Strategien, nämlich bezüglich einer optimalen, aber praktisch nicht erreichbaren Offline-Lösung. Über die eigentlich interessierenden Kosten oder die Fehlerrate erfährt man unmittelbar nichts.

Das Zugriffsgraphenmodell (Borodin, Irani, Raghavan und Schieber, 1995) begegnet den ersten beiden Kritikpunkten, indem es Anfragelokalität berücksichtigt. In dem Modell wird jede Seite des virtuellen Speichers durch einen Knoten eines Graphen repräsentiert. Die Anfragelokalität der Anfragesequenzen wird durch die Kanten dieses Zugriffsgraphen G modelliert: Genau dann, wenn es zulässig ist, dass auf eine Anfrage an eine Seite p eine Anfrage an eine Seite p' folgt, gibt es in G eine Kante zwischen den Knoten von p und p'. Das bedeutet, eine zulässige Anfragesequenz beschreibt einen Pfad durch den Graphen. Die Struktur des von einem Prozess ausgeführten Programms bestimmt die Kanten des Graphen und legt somit fest, welche Pfade zulässig sind. Die dem Prozess zugeführte Eingabe bestimmt, welcher Pfad gewählt wird. Man unterscheidet im Zugriffsgraphenmodell zwischen gerichteten und ungerichteten Zugriffsgraphen. Wenn die Anfragen eines Prozesses im Wesentlichen durch den Kontrollfluss des ausgeführten Programms bestimmt sind, ist ein Zugriffsgraph mit gerichteten Kanten angemessen. Dieser kann vom Übersetzer bestimmt werden und so der Seitenwechselstrategie zur Verfügung gestellt werden. Für Prozesse, deren Anfragen stark durch die Zugriffe auf Datenstrukturen bestimmt sind, mögen auch ungerichtete Zugriffsgraphen angemessen sein, die dann im Wesentlichen die Datenstruktur widerspiegeln. Hier kann man sich vorstellen, dass ein Tiefendurchlauf durch einen Baum einen Zugriffsgraphen erzeugt, der im Wesentlichen ein ungerichteter Baum ist.

Borodin, Irani, Raghavan und Schieber (1995) bezeichnen mit $c_{\mathcal{A},k}(G)$ den Kompetitivitätsfaktor der Strategie $\mathcal{A}$ bei Cachegröße k für Sequenzen, die für den Zugriffsgraphen G zulässig sind. In ihren Analysen beschränken sie $c_{\mathcal{A},k}(G)$ mithilfe der Eigenschaften von G. Unter anderem zeigen sie für ungerichtete Graphen G, dass $c_{\mathrm{LRU},k}(G) \leq 2 \cdot c_{\mathrm{FIFO},k}(G)$ gilt. Chrobak und Noga (1998) zeigen, dass der Faktor 2 entfallen kann, d. h., dass LRU in diesem Modell keinesfalls schlechter ist als FIFO. Borodin, Irani, Raghavan und Schieber (1995) entwerfen auch eine einfache Strategie FAR, die mithilfe des Graphen G entscheidet, welche Seite bei einem Seitenfehler ersetzt werden soll. FAR geht dabei nach Art einer Markierungsstrategie vor und wählt eine unmarkierte Seite, die in G den maximalen Abstand zu allen markierten Seiten hat. Damit versucht FAR das Verhalten von LFD nachzuahmen. Die Autoren zeigen auch, dass der Kompetitivitätsfaktor von FAR im Wesentlichen nur um einen Faktor $O(\log k)$ größer ist als der Kompetitivitätsfaktor $c_k(G)$ der besten Online-Strategie für G. (Das bedeutet *nicht*, dass FAR $O(\log k)$-kompetitiv ist.) Irani, Karlin und Phillips (1996) verbessern diese Schranke und zeigen, dass FAR stark kompetitiv (engl. *strongly competitive*) ist, d. h. $c_{\mathrm{FAR},k} = O(c_k(G))$. Fiat und Karlin (1995) untersuchen randomisierte Strategien und den Fall, dass mehrere Prozesse gleichzeitig auf demselben Zugriffsgraphen arbeiten. Fiat und Mendel (1997) präsentieren stark kompetitive Algorithmen, die kein Vorabwissen über den Zugriffsgraphen benötigen, die also wirklich online (engl. *truly online*) arbeiten.

Verwandt mit dem Zugriffsgraphenmodell ist das Markoffkettenmodell (engl. *Markov paging*) von Karlin, Phillips und Raghavan (2000). Darin wird eine zufällige Anfragesequenz von einer Markoffkette erzeugt, deren Zustände den Seiten des virtuellen Speichers entsprechen. Für jede Markoffkette ist die bestmögliche Online-Strategie in diesem Modell $\Theta(\log k)$-kompetitiv. Die Autoren stellen eine Strategie vor, die diesen Faktor für beliebige Markoffketten erreicht.

Die Idee von Young (1991) ist es, die Bedingungen für den Begriff c-kompetitiv aufzuweichen. Er führt den Begriff der *loose competitiveness* ein, der variierende Cachegrößen k betrachtet. Der Kompetitivitätsfaktor wird dann als Funktion $c(k)$ aufgefasst. Für $N \in \mathbb{N}$ müssen die Bedingungen nur noch für fast alle Cachegrößen $k \in \{1, \dots, N\}$ erfüllt sein, d. h., für $o(N)$ der N Werte für k dürfen die Kosten der Online-Strategie auch mehr als das $c(k)$-fache der Kosten der optimalen Offline-Strategie betragen. Darüber hinaus werden die Bedingungen für den Fall abgeschwächt, dass die Kosten der Anfragesequenz unbedeutend klein und daher nicht relevant sind. In diesem Modell erreicht Young deutlich kleinere Kompetitivitätsfaktoren. Für die oben diskutierten Online-Strategien rückt der Kompetitivitätsfaktor $c(k)$ in die Nähe von $\ln k$. Jedoch können FIFO und LRU nicht in ihrer Güte unterschieden werden.

Torng (1998) analysiert das Seitenwechselproblem im Zugriffskostenmodell und nimmt an, dass eine Anfrage an eine Seite im Cache Kosten 1 verursacht, während ein Seitenfehler Kosten $p + 1$ verursacht. Die kompetitive Analyse vergleicht hier weiterhin die Kosten von Online-Algorithmen mit den Kosten der kostenminimalen Offline-Lösung. In diesem Modell ist der bestmögliche Kompetitivitätsfaktor $k(p + 1)/(k + p)$ und alle Markierungsstrategien, also auch LRU, erreichen diesen Faktor. Um Anfragelokalität zu modellieren, teilt Torng Anfragesequenzen von vorne beginnend in Phasen maximaler Länge ein, sodass jede der entstehenden Phasen k verschiedene Seiten anfragt. Die Arbeitsmenge in jeder dieser Phasen besteht dann aus k Seiten. Wenn die durchschnittliche Länge dieser Phasen größer als k ist, dann muss es offenbar Anfragelokalität geben. Mithilfe des Parameters $\alpha \geq 1$ kann man nun das Mindestmaß an Anfragelokalität vorgeben, indem man nur Sequenzen mit durchschnittlicher Phasenlänge von mindestens αk betrachtet. In diesem Fall ist die Kompetitivität von Markierungsstrategien durch $1 + p/\alpha$ beschränkt. Dieses Ergebnis kann man für ein konstantes α als eine Konstante auffassen. Allerdings ist p sehr groß. Größenordnungen von bis zu 10^6 scheinen für p noch realistisch (siehe Abschnitt 15.1). Die im Folgenden vorgestellten Modelle haben Ähnlichkeit mit Torngs Ansatz.

15.4 Maximums- und Durchschnittsmodell

In dieser Arbeit sollen zwei Modelle zur Modellierung der Anfragelokalität untersucht werden. Das Ziel ist es, Modelle zu finden, die den Aspekt der Anfragelokalität möglichst einfach modellieren und unmittelbar die Analyse der Fehlerrate ermöglichen. Der letzte Punkt ist im Rahmen der kompetitiven Analyse nicht möglich. Die Modelle gestatten es, die Menge der zulässigen Anfragesequenzen so zu beschränken, dass diese Anfragelokalität aufweisen. Beide Modelle sind sehr eng miteinander verwandt und basieren auf dem Arbeitsmengenmodell (engl. *working-set model*) von Denning (1968). Unter der Arbeitsmenge (engl. *working set*) versteht man vereinfacht gesagt die Menge der Seiten, die innerhalb eines kurzen Zeitraums von einem Prozess angefragt werden und daher im Cache vorgehalten werden sollten. Formal kann man den Begriff der Arbeitsmenge zum Zeitpunkt t und für die Fenstergröße ℓ so fassen, dass man die Arbeitsmenge $W_t(\ell)$ als die Menge der Seiten definiert, die in der Teilsequenz (dem Fenster) $\sigma_t(\ell)$ aus den Anfragen $\sigma_t, \ldots, \sigma_{t+\ell-1}$ angefragt werden. Dabei modelliert die Anfragesequenz σ den betrachteten Prozess.

Wenn man zu einem beliebigen, aber festen Zeitpunkt t die Fenstergröße ℓ variiert, erhält man für diesen Zeitpunkt t die Größe der Arbeitsmenge als Funktion $N_t(\ell) = |W_t(\ell)|$. Bild 15.1 stellt den qualitativen Verlauf dieser Funktion dar, den man typischerweise beobachten kann. Die Funktion $N_t(\ell)$ ist auf jeden Fall monoton wachsend, da ein vergrößertes Fenster immer nur gleich viele oder mehr Seiten enthalten kann. Für große Fenstergrößen ℓ ist die Zunahme nur noch gering. Die Kurve nähert sich schließlich der Prozessgröße, also der Zahl der insgesamt in σ angefragten Seiten, an. Denning betrachtet das Fenster $\sigma_{t-\ell}(\ell)$ als geeignete „Vorhersage“ für $\sigma_t(\ell)$ und folgert unter der Annahme, dass sich „$N_{t-\ell}(\ell)$ im Durchschnitt genauso verhält wie $N_t(\ell)$“, dass der generelle Verlauf von $N_t(\ell)$ als monoton wachsend und konkav beschrieben werden kann. (Anschaulich heißt eine Funktion *konkav*, wenn ihre Kurve niemals oberhalb einer ihrer Tangenten liegt.) In einer „vergröberten“ Sicht wird man in der Regel den Eindruck einer konkaven Funktion haben; in einer Detailansicht mag diese Eigenschaft jedoch verletzt sein. Für echte Anfragesequenzen kommt der Verlauf von $N_t(\ell)$ in der Regel dem Verlauf einer monoton wachsenden, konkaven Funktion f zumindest nahe, und zwar so, dass $f(\ell)$ in jedem Punkt eine obere Schranke für $N_t(\ell)$ ist.

Das ursprüngliche Arbeitsmengenmodell von Denning betrachtet die Größe der Arbeitsmengen also eher lokal, in dem Sinne, dass Fenster in der Nähe eines Zeitpunktes t betrachtet werden. In den hier eingeführten Modellen hingegen begrenzt die Funktion f die maximale oder durchschnittliche Arbeitsmengengröße global, d. h. für alle Fenster, die in der Sequenz liegen. Daher nennen wir die Modelle

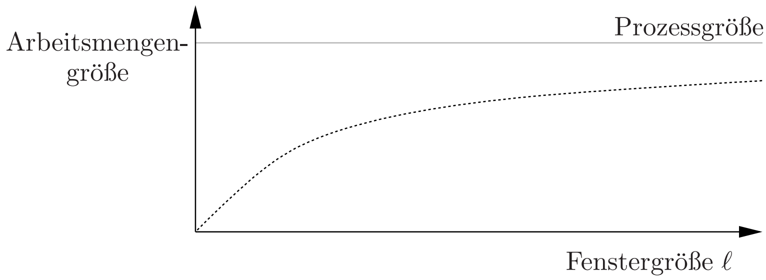

Bild 15.1: Arbeitsmengengröße in Abhängigkeit von der Fenstergröße

Maximums- und *Durchschnittsmodell*. In beiden Modellen gehen wir davon aus, dass ein Programm durch eine Funktion $f\colon \mathbb{N} \to \mathbb{R}^{\geq 1}$ charakterisiert werden kann, wenn die bei seiner Ausführung erzeugte Anfragesequenz $\sigma_1, \ldots, \sigma_n$ der Funktion f genügt. (Das Symbol $\mathbb{R}^{\geq 1}$ steht hier für die Menge der reellen Zahlen, die größer oder gleich 1 sind.) Im Maximumsmodell *genügt* eine Anfragesequenz $f(\ell)$, wenn für alle $\ell \leq n$ in jedem Fenster der Länge ℓ die maximale Zahl angefragter (verschiedener) Seiten höchstens $f(\ell)$ ist. Im Durchschnittsmodell ist die Forderung weniger strikt. Dort *genügt* eine Anfragesequenz bereits f, wenn für alle $\ell \leq n$ im Durchschnitt über alle Fenster der Länge ℓ die Zahl angefragter (verschiedener) Seiten höchstens $f(\ell)$ ist. Nur diese Funktion f wird in beiden Modellen als bekannt vorausgesetzt, sodass die Analyseergebnisse für alle Sequenzen gelten, die f genügen.

Unter der Fehlerrate $F_{\mathcal{A}}(\sigma)$ einer Strategie $\mathcal{A}$ auf einer Sequenz σ versteht man den Quotienten aus der Zahl der Seitenfehler $\mathcal{A}(\sigma)$, die σ bei Anwendung von $\mathcal{A}$ verursacht, und der Sequenzlänge $|\sigma|$.

Definition 15.2 (Fehlerrate). *Die Fehlerrate einer Seitenwechselstrategie $\mathcal{A}$ auf einer Sequenz σ ist $F_{\mathcal{A}}(\sigma) := \mathcal{A}(\sigma)/|\sigma|$.*

Selbstverständlich hängt die Fehlerrate auch von der Cachegröße k ab. Dieser Parameter soll aber aus Gründen der Übersichtlichkeit nicht in unsere Notation eingehen. Wir wollen im Folgenden die Fehlerrate nicht nur bez. einer einzigen Sequenz σ betrachten, sondern bez. der Menge aller Sequenzen, die einer Funktion f genügen. Im Wesentlichen soll die Fehlerrate bez. f die größte Fehlerrate sein, die für Sequenzen in dieser Menge vorkommen kann. In diesem Sinne ist die Fehlerrate bez. f ein Worst-Case-Maß. Damit aber Anfangseffekte keinen merklichen Einfluss auf die Fehlerrate haben, betrachten wir den Grenzübergang, bei dem die Sequenzlänge gegen unendlich strebt. Dies führt zu folgender Definition.

Definition 15.3 (Fehlerrate bez. f). *Die Fehlerrate einer Seitenwechselstrategie $\mathcal{A}$ bezüglich einer Funktion f ist*

$$F_{\mathcal{A}}(f) \;:=\; \lim_{n\to\infty} \max\{F_{\mathcal{A}}(\sigma) \mid \sigma \text{ genügt } f \text{ und } |\sigma| \geq n\}.$$

Der Limes in der vorhergehenden Definition existiert. Er wird über eine monoton fallende Folge gebildet, die nach unten durch 0 beschränkt ist. Zur Klarheit sei noch bemerkt, dass die Definition sich auf den Begriff „genügt f“ abstützt, der in beiden Modellen verschieden – wie oben beschrieben – zu interpretieren ist. $F_{\mathcal{A}}(f)$ hängt also auch vom betrachteten Modell ab, das ebenfalls aus Gründen der Übersichtlichkeit nicht noch zusätzlich in die Notation einfließen soll.

Welche Eigenschaften können wir von f annehmen? Wenn wir für eine bestimmte Strategie die Fehlerrate für alle Sequenzen bestimmen wollen, die einer gegebenen Funktion f genügen, steht uns nur f zur Verfügung. Daher sollen die wesentlichen Eigenschaften von Funktionen, die die maximale und durchschnittliche Zahl angefragter Seiten in Fenstern der Länge ℓ beschreiben, auch für $f(\ell)$ vorausgesetzt werden. Sicherlich ist es vernünftig, anzunehmen, dass $f(1) = 1$ ist, da ein Fenster der Länge 1 genau eine Anfrage enthält. Das gilt sowohl im Durchschnitt als auch für das Maximum über alle Fenster der Länge 1. Im Maximumsmodell ist es klar, dass f monoton wachsend sein sollte, da in größeren Fenstern nicht weniger Seiten angefragt werden können. Die Monotonie gilt aber auch im Durchschnitt, wovon wir uns überzeugen wollen. Mit

$$N(\ell) := \sum_{t=1}^{|\sigma|-\ell+1} N_t(\ell) \qquad \text{und} \qquad \mathrm{Av}(\ell) := \frac{N(\ell)}{|\sigma| - \ell + 1}$$

sei im Folgenden die Summe der Arbeitsmengengrößen über alle möglichen Fenster der Länge ℓ in einer Sequenz σ bzw. die durchschnittliche Arbeitsmengengröße über alle diese Fenster bezeichnet. Es ist also zu zeigen, dass $\mathrm{Av}(\ell) \leq \mathrm{Av}(\ell + 1)$ für $1 \leq \ell \leq n - 1$ gilt, wobei $n = |\sigma|$ ist. Dazu betrachten wir zunächst alle Werte $N_1(\ell), \ldots, N_{n-\ell+1}(\ell)$ und wählen einen Index j, sodass der $N_j(\ell)$-Wert höchstens durchschnittlich ist, d. h. $N_j(\ell) \leq (N_1(\ell) + \cdots + N_{n-\ell+1}(\ell))/(n - \ell + 1)$. Daraus folgt

$$N_j(\ell) \leq \frac{N_1(\ell) + \cdots + N_{j-1}(\ell) + N_{j+1}(\ell) + \cdots + N_{n-\ell+1}(\ell)}{n - \ell}. \qquad (*)$$

Nun stellen wir $\mathrm{Av}(\ell)$ gemäß der obigen Definition als

$$\frac{n-\ell}{n-\ell+1} \cdot \frac{N_1(\ell) + \cdots + N_{j-1}(\ell) + N_{j+1}(\ell) + \cdots + N_{n-\ell+1}(\ell)}{n-\ell} + \frac{1}{n-\ell+1} \cdot N_j(\ell)$$

dar. Da die Koeffizienten $(n - \ell)/(n - \ell + 1)$ und $1/(n - \ell + 1)$ in der Summe 1 sind, d. h., sie beschreiben eine Linearkombination, erhalten wir mithilfe von Ungleichung $(*)$

$$\mathrm{Av}(\ell) \leq \frac{N_1(\ell) + \cdots + N_{j-1}(\ell) + N_{j+1}(\ell) + \cdots + N_{n-\ell+1}(\ell)}{n-\ell}.$$

Offensichtlich gilt $N_i(\ell) \leq N_i(\ell+1)$ für $1 \leq i < j$ und $N_i(\ell) \leq N_{i-1}(\ell+1)$ für $j < i \leq n-\ell+1$. Somit ergibt sich schließlich das gewünschte Resultat

$$\mathrm{Av}(\ell) \;\leq\; \frac{N_1(\ell+1) + \cdots + N_{n-\ell}(\ell+1)}{n-\ell} \;=\; \mathrm{Av}(\ell+1).$$

Motiviert durch Dennings Beobachtung interessieren uns hier besonders konkave Funktionen f. Für monoton wachsende Funktionen $f\colon \mathbb{N} \to \mathbb{R}^{\geq 1}$ wollen wir unter dem Begriff „konkav“ die Eigenschaft $f(n+1)-f(n) \geq f(n+2)-f(n+1)$ für alle $n \in \mathbb{N}$ verstehen. Diese Eigenschaft ist jedoch in beiden Modellen eine Annahme für f. Sie ist nicht mathematisch aus Anfragesequenzen herleitbar; es gibt Gegenbeispiele. Dennoch können die maximale und durchschnittliche Zahl angefragter Seiten in der Regel gut durch konkave Funktionen f beschränkt werden. (Dazu genügt es, für f Punkte auf der konvexen Hülle der monoton wachsenden Maximums- bzw. Durchschnittsfunktion zu wählen.) In der folgenden Definition fassen wir die drei bisher diskutierten Eigenschaften ($f(1)=1$, Monotonie und Konkavität) unter dem Begriff „konkav*“ zusammen.

Definition 15.4 (konkav*). *Eine Funktion $f\colon \mathbb{N} \to \mathbb{R}^{\geq 1}$ heißt* konkav*, *falls*

1. $f(1) = 1$ *und*
2. $\forall n \in \mathbb{N}\colon f(n+1)-f(n) \;\geq\; f(n+2)-f(n+1) \;\geq\; 0$

gilt.

Im Durchschnittsmodell werden wir die Fehlerrate bez. Funktionen f analysieren, die konkav* sind.

Nur im Fall des Maximumsmodells werden wir noch zwei zusätzliche Eigenschaften hinzufügen. Die erste Eigenschaft ist, dass wir $f(2) = 2$ fordern. Diese Einschränkung ist leicht zu motivieren, sie ist nahezu zwingend. Keine Sequenz kann in einem Fenster der Länge 2 mehr als 2 Seiten anfragen und $f(2) < 2$ erlaubte im Maximumsmodell nur Sequenzen, die insgesamt lediglich eine Seite anfragen. Jede (vernünftige) Seitenwechselstrategie sollte bez. Funktionen f mit $f(2) < 2$ Fehlerrate 0 erreichen; für Demand-Paging-Strategien gilt dies ohnehin. Funktionen f mit $f(2) < 2$ sind also im Maximumsmodell von vornherein uninteressant.

Die zweite zusätzliche Eigenschaft ist wie folgt motiviert. Wenn in jedem Fenster der Länge ℓ höchstens $g(\ell) \in \mathbb{N}$ Seiten angefragt werden, dann können es in jedem Fenster der Länge $\ell+1$ nur $g(\ell+1) \in \{g(\ell), g(\ell)+1\}$ Seiten sein. Wenn wir also die maximale Arbeitsmengengröße in Abhängigkeit von der Fenstergröße an einer beliebigen Anfragesequenz bestimmen, dann erhalten wir eine Funktion g, die ausschließlich natürliche Zahlen und alle natürliche Zahlen zwischen 1 und einem Maximalwert M annimmt. Naheliegend wäre es, auch für f nur solche Funktionen $f\colon \mathbb{N} \to \mathbb{N}$ zuzulassen. Wir bleiben aber bei Funktionen $f\colon \mathbb{N} \to \mathbb{R}^{\geq 1}$, denn die

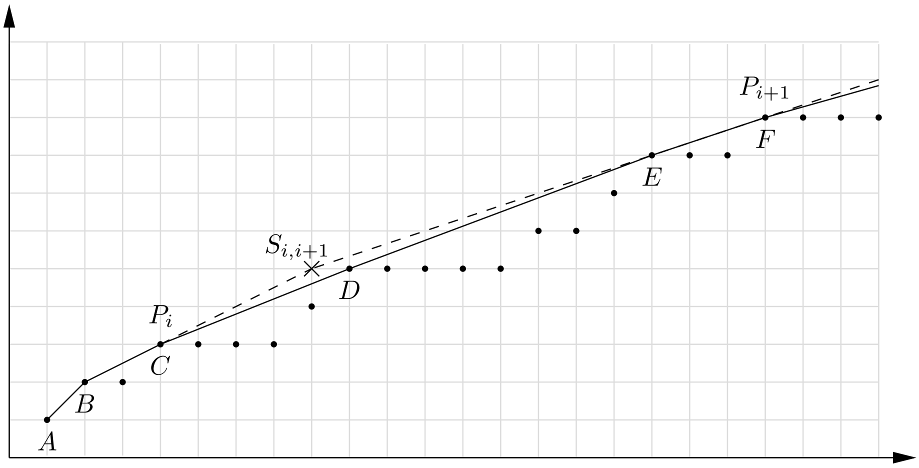

Bild 15.2: Konstruktion von f

wesentliche Eigenschaft von g ist, dass alle natürlichen Zahlen ein Urbild besitzen. Nur diese übernehmen wir in die Funktion f. Auf diese Weise wird diese wichtige Eigenschaft der Funktion g Analysen zugänglich gemacht, in denen die obere Schranke f, aber nicht g selbst als bekannt vorausgesetzt wird.

Im Folgenden bezeichnet $\mathrm{Bild}(f)$ für eine Funktion $f\colon X \to Y$ die Menge $\{f(x) \mid x \in X\}$.

Definition 15.5 (konkav).** *Eine Funktion $f\colon \mathbb{N} \to \mathbb{R}^{\geq 1}$ heißt konkav**, falls*

1. *sie konkav* ist,*
2. $f(2) = 2$ *gilt und*
3. *entweder* $\mathbb{N} \subseteq \mathrm{Bild}(f)$ *oder für die größte natürliche Zahl* $M \in \mathrm{Bild}(f)$ *gilt, dass alle natürlichen Zahlen* $m \leq M$ *ebenfalls in* $\mathrm{Bild}(f)$ *enthalten sind.*

Wir wollen uns kurz davon überzeugen, dass für jede monoton wachsende und surjektive Funktion $g\colon \mathbb{N} \to \{1, \ldots, M\}$ mit $g(i) \leq i$, $g(1) = 1$ und $g(2) = 2$ leicht eine Funktion $f\colon \mathbb{N} \to \mathbb{R}^{\geq 1}$ konstruiert werden kann, sodass g der Funktion f genügt und f konkav** ist. Natürlich soll f dabei möglichst dicht bei g liegen. Bild 15.2 veranschaulicht die Konstruktion, auf die wir später in Kapitel 19 zurückgreifen werden. Sie beruht darauf, ausgehend von der konvexen Hülle von g einen Polygonzug zu konstruieren, der die Funktion f mit den gewünschten Eigenschaften definiert.

Als erste Näherung bestimmen wir zu g (dargestellt durch Punkte) die konvexe Hülle von g (dargestellt durch ununterbrochene Linien). Wegen der Konvexität der Hülle von g nimmt die Steigung der Segmente der konvexen Hülle von links nach rechts niemals zu. Wenn wir mit $h : \mathbb{N} \to \mathbb{R}^{\geq 1}$ die Funktion bezeichnen, die durch die konvexe Hülle von g definiert ist, dann ist h wegen der Monotonie von g konkav*, aber nicht unbedingt konkav**, weil Bild(h) möglicherweise einige Zahlen aus $\{1, \ldots, M\}$ auslässt. Daher ersetzen wir die Segmente der konvexen Hülle ggf. durch Segmente mit Steigung 1/1, 1/2, 1/3, 1/4 usw.

Dazu bestimmen wir zunächst besondere Eckpunkte P_i der konvexen Hülle. P_i ist der Eckpunkt, sodass das Segment links an P_i eine Steigung aus $[1/i, 1/(i-1)[$ hat und das Segment rechts eine Steigung von weniger als $1/i$ hat. In Bild 15.2 sind dies die Punkte $P_1 = B$, $P_2 = C$ und $P_3 = F$. Hat die konvexe Hülle zwischen zwei Punkten P_i und P_{i+1} überall die Steigung $1/(i+1)$ (im Beispiel zwischen B und C), haben P_i und P_{i+1} auf der Abszisse einen Abstand, der ein Vielfaches von $i+1$ ist, weil sie als Eckpunkte der konvexen Hülle ganzzahlige Koordinaten haben. Daher lässt h zwischen diesen Punkten keinen ganzzahligen Ordinatenwert aus. In diesen Bereichen übernehmen wir also die Segmente der konvexen Hülle. Andernfalls ersetzten wir die konvexe Hülle zwischen P_i und P_{i+1} durch zwei neue Segmente, wobei das erste in P_i beginnt und Steigung $1/i$ hat und das zweite in P_{i+1} endet und Steigung $1/(i+1)$ hat. Daraus ergibt sich der gemeinsame Endpunkt $S_{i,i+1}$ als Schnitt der Geraden, auf denen die neuen Segmente liegen. Offenbar liegen die neuen Segmente oberhalb der Segmente der konvexen Hülle. Mit den Bezeichnungen $P_i = (x_i, y_i)$ und $P_{i+1} = (x_{i+1}, y_{i+1})$ wird jeder ganzzahlige Ordinatenwert zwischen y_i und der Ordinate von $S_{i,i+1}$ an den Stellen $x_i + i$, $x_i + 2i$ usw. angenommen, jeder ganzzahlige Ordinatenwert zwischen der Ordinate von $S_{i,i+1}$ und y_{i+1} an den Stellen $x_{i+1} - (i+1)$, $x_{i+1} - 2(i+1)$ usw. Es bleibt noch der Fall, dass es zwar einen Punkt P_i, aber keinen Punkt P_{i+1} gibt, weil g plötzlich abflacht. Dann übernimmt der „nächste“ benannte Punkt P_j, $j > i$, die Rolle von P_{i+1}.

Der so aus der konvexen Hülle gewonnene Polygonzug definiert nun f. Offensichtlich gilt $f(1) = 1$ und $f(2) = 2$, weil f hier mit g übereinstimmt. Insgesamt nimmt die Steigung der Segmente von links nach rechts nur ab, sodass f konkav* ist. Da nun jedem ganzzahligen Ordinatenwert zwischen 1 und M ein ganzzahliges Urbild gegenübersteht, ist f konkav**. Weil der Polygonzug nirgends unterhalb der konvexen Hülle von g liegt, genügt g der Funktion f.

16 Analyse der Fehlerrate im Maximumsmodell

In diesem und dem folgenden Kapitel werden die theoretischen Analysen der Fehlerraten der in Abschnitt 15.2 vorgestellten Strategien für das Seitenwechselproblem präsentiert. Wir beginnen in diesem Kapitel mit dem Maximumsmodell und untersuchen danach in Kapitel 17 das Durchschnittsmodell. In den beiden Kapiteln gehen wir so vor, dass wir zuerst untere Schranken für die Fehlerraten beweisen, die für *alle* Online-Strategien im jeweiligen Modell gelten. Dann wenden wir uns der konkreten Online-Strategie LRU zu, die sich in beiden Modellen als optimal erweisen wird, und studieren dann Markierungsstrategien im Allgemeinen. Danach untersuchen wir FIFO. Für alle diese Online-Strategien werden wir in beiden Modellen scharfe oder zumindest beinahe scharfe Schranken erhalten. Zum Schluss untersuchen wir jeweils noch LFD, weil diese optimale Offline-Strategie als Bezugspunkt der kompetitiven Analyse besonders interessant im Vergleich zu den Online-Strategien ist. In Kapitel 18 fassen wir die theoretischen Ergebnisse aus beiden Modellen übersichtlich zusammen und stellen die sich entsprechenden Ergebnisse der beiden Modelle einander gegenüber.

Damit unsere Analysen überhaupt Sinn ergeben, vereinbaren wir, dass die Cachegröße k mindestens 2 ist. Andernfalls, also für $k = 1$, verhalten sich alle Seitenwechselstrategien gleich, da es keine echten Entscheidungen zu treffen gibt. Des Weiteren betrachten wir k als eine frei wählbare, aber konstante Größe. Der entscheidende Punkt dabei ist, dass k nicht von der Sequenzlänge $n := |\sigma|$ (also der Eingabelänge im Offline-Szenario) abhängen soll.

16.1 Vorbereitungen

Bevor wir mit den eigentlichen Analysen beginnen können, müssen wir uns noch etwas „technisches Rüstzeug“ zurechtlegen. Zunächst müssen wir zu einer Zahl m bestimmen können, welches die kleinste Fensterlänge ist, sodass in einem Fenster dieser Länge m verschiedene Seiten vorkommen dürfen. Dies leistet die Funktion f^{-1}, die man vereinfacht gesagt als die inverse Funktion zu f ansehen kann. Im Folgenden bezeichnet $\mathbb{N}_{\mathrm{Bild}(f)}$ die Menge der natürlichen Zahlen, die im Bild von f enthalten sind, d. h. $\mathbb{N}_{\mathrm{Bild}(f)} := \mathbb{N} \cap \mathrm{Bild}(f)$.

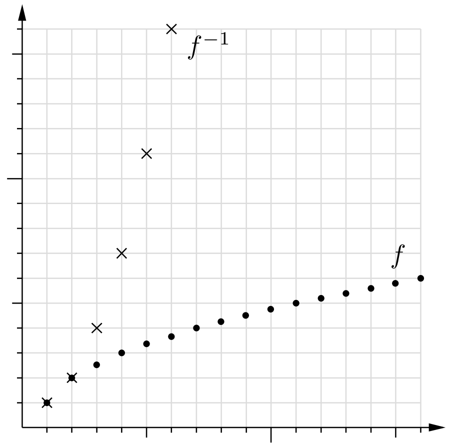

Bild 16.1: Beispiel für eine Funktionen f, die konkav** ist, und die zugehörige Funktion f^{-1}

Definition 16.1 (f^{-1}). *Es sei $f\colon \mathbb{N} \to \mathbb{R}$ eine reellwertige Funktion. Dann ist $f^{-1}\colon \mathbb{N}_{\mathrm{Bild}(f)} \to \mathbb{N}$ definiert als*

$$f^{-1}(m) := \min\{n \in \mathbb{N} \mid f(n) \geq m\}.$$

Im Maximumsmodell untersuchen wir die Fehlerrate bez. Funktionen, die konkav** sind. Wir treffen folgende Vereinbarung.

Im Rest dieses Kapitels bezeichnet f eine beliebige Funktion, die konkav** ist.

Bild 16.1 illustriert eine solche Funktion f und die zugehörige Funktion f^{-1}. Die folgende Aussage beinhaltet anschaulich, dass die Funktion f^{-1} konvex ist.

Proposition 16.2. *Wenn f konkav** ist, ist f^{-1} streng monoton wachsend und für alle $m \geq 3$, die in $\mathbb{N}_{\mathrm{Bild}(f)}$ enthalten sind, gilt*

$$f^{-1}(m) - f^{-1}(m-1) \geq f^{-1}(m-1) - f^{-1}(m-2).$$

Beweis. Da f konkav** ist, gilt $f(1) = 1$, $f(2) = 2$ und $f^{-1}(2) - f^{-1}(1) = 1$. Falls die behauptete Ungleichung gilt, dann folgt sofort, dass f^{-1} streng monoton wachsend ist.

Wegen der dritten Eigenschaft in Definition 15.5 gibt es natürliche Zahlen n_{m-2}, n_{m-1} und n_m, sodass $f(n_{m-2}) = m-2$, $f(n_{m-1}) = m-1$ und $f(n_m) = m$. Daraus folgt $f^{-1}(m) = n_m$, $f^{-1}(m-1) = n_{m-1}$ und $f^{-1}(m-2) = n_{m-2}$. Es gilt

$$1 = f(n_m) - f(n_{m-1}) = \sum_{i=n_{m-1}}^{n_m - 1} \big(f(i+1) - f(i)\big) \quad \text{und}$$

$$1 = f(n_{m-1}) - f(n_{m-2}) = \sum_{i=n_{m-2}}^{n_{m-1} - 1} \big(f(i+1) - f(i)\big).$$

Wegen der zweiten Eigenschaft von Definition 15.4 ist jeder Summand der unteren Summe mindestens so groß wie jeder Summand der oberen Summe. Also hat die obere Summe mindestens so viele Summanden wie die untere und es folgt daraus $(n_m - 1) - n_{m-1} \geq (n_{m-1} - 1) - n_{m-2}$. Daher gilt die Ungleichung

$$f^{-1}(m) - f^{-1}(m-1) \;=\; n_m - n_{m-1} \;\geq\; n_{m-1} - n_{m-2} \;=\; f^{-1}(m-1) - f^{-1}(m-2).$$

□

16.2 Eine größtmögliche untere Schranke

Wir zeigen hier eine untere Schranke, die für jede beliebige deterministische Online-Strategie im Maximumsmodell gilt. Im darauf folgenden Abschnitt über Markierungsstrategien werden wir sehen, dass diese Schranke i. Allg. nicht größer sein kann.

In Analysen nimmt man häufig an, dass der Cache zu Beginn einer Anfragesequenz leer ist. Wenn wir davon absehen, dass Prozesse sich Speicher teilen können (engl. *shared memory*) oder dass Seiten aus früheren Läufen desselben Programms noch im Cache liegen, entspricht das auch durchaus der Realität. Wir werden hier für untere Schranken grundsätzlich ohne diese Annahme auskommen, d. h., wir nehmen im Gegenteil an, dass bei Beginn irgendwelche k Seiten des Prozesses bereits im Cache liegen. Das kann den Beweis möglichst großer unterer Schranken nur erschweren. Dies ist allerdings kein entscheidender Punkt, denn wir haben die Definition der Fehlerrate bez. f (Definition 15.3) so gewählt, dass Änderungen der Fehlerzahl um additive Konstanten die Fehlerrate nicht verändert. Dieses Vorgehen findet man auch bei der kompetitiven Analyse wieder, damit sich Anfangseffekte nicht auswirken.

Die Idee für untere Schranken im Maximumsmodell lernen wir im Beweis des folgenden Theorems kennen. Wir konstruieren Anfragesequenzen, die sich aus sich beliebig oft wiederholenden Phasen zusammensetzen. Die Fehlerrate ergibt sich dann aus der Zahl der Fehler pro Phase und der Länge einer Phase. Die Phasen sind meistens aus Blöcken zunehmender Länge aufgebaut und innerhalb eines Blocks wird nur eine Seite mehrmals hintereinander angefragt. In der Regel ist es der aufwendigere Teil, zu zeigen, dass die konstruierte Sequenz zulässig ist, d. h., dass sie der Funktion f genügt.

Theorem 16.3. *Es sei $\mathcal{A}$ eine deterministische Online-Strategie. Dann gilt im Maximumsmodell*

$$F_{\mathcal{A}}(f) \;\geq\; \frac{k-1}{f^{-1}(k+1)-2}.$$

Beweis. Wir konstruieren für jede beliebige Online-Strategie $\mathcal{A}$ Anfragesequenzen $\sigma = \sigma_1, \ldots, \sigma_n$, deren Länge n beliebig groß werden kann. Dazu benutzen wir $k+1$

verschiedene Seiten $p_1, \ldots, p_{k+1}$. Eine Anfragesequenz ist in Phasen eingeteilt und kann durch Anhängen weiterer Phasen beliebig verlängert werden. Jede Phase hat die Länge $f^{-1}(k+1) - 2$ und besteht aus $k-1$ Blöcken. Ein Block ist eine Teilsequenz aus Anfragen an dieselbe Seite, und zwar jeweils an diejenige Seite, die zum Ende des vorhergehenden Blocks nicht in $\mathcal{A}$s Cache war. Diese Seite ist eindeutig bestimmt, weil $\mathcal{A}$ deterministisch arbeitet. Also hat $\mathcal{A}$ Kosten 1, d. h. einen Seitenfehler, pro Block und somit Kosten $k-1$ pro Phase. Daraus ergibt sich die behauptete Fehlerrate.

In jeder Phase beginnt der erste Block mit Anfrage Nummer 1 in der Phase. Im Allgemeinen beginnt der j-te Block, $1 \leq j \leq k-1$, stets mit der Anfrage mit Nummer $f^{-1}(j+1) - 1$. Diese Einteilung in Blöcke ist wohldefiniert: Der j-te Block hat die Länge $(f^{-1}(j+2) - 1) - (f^{-1}(j+1) - 1) = f^{-1}(j+2) - f^{-1}(j+1)$. Wegen Proposition 16.2 sind die Blocklängen innerhalb eines Blocks nicht fallend. Da f^{-1} streng monoton wachsend und $f^{-1}(2) = 2$ gilt, sind die Blöcke auch nicht leer und die gesamte Sequenz ist wohldefiniert.

Nun müssen wir nur noch nachweisen, dass die konstruierten Sequenzen auch f genügen. Dazu zeigen wir, dass jede Teilsequenz, die j verschiedene Seiten anfragt, mindestens Länge $f^{-1}(j)$ hat. Für $1 \leq j \leq 2$ ist die Aussage offensichtlich richtig, da $f^{-1}(1) = 1$ und $f^{-1}(2) = 2$ gilt.

Wir betrachten nun den Fall $3 \leq j \leq k$. Jede Teilsequenz mit j verschiedenen Seiten muss j aufeinander folgende Blöcke (zumindest teilweise) überdecken. Da alle Blöcke bez. der darin angefragten Seite homogen sind, überdeckt eine Teilsequenz minimaler Länge mit j verschiedenen Seiten nur die letzte Anfrage des frühesten Blocks, den sie (teilweise) überdeckt, und die erste Anfrage des letzten Blocks, den sie (teilweise) überdeckt. Würden wir die Teilsequenz weiter in diese Randblöcke ausdehnen, könnten wir keine neuen Seiten hinzugewinnen. Wie oben gezeigt sind die Blocklängen innerhalb einer Phase nicht fallend. Daher überdeckt eine Sequenz mit j verschiedenen Seiten minimaler Länge die ersten $j-2$ Blöcke einer Phase vollständig. Diese haben zusammen Länge $f^{-1}(j) - 2$. Hinzu kommt die letzte Anfrage der Vorgängerphase und die erste Anfrage von Block $j-1$. Die Gesamtlänge ist also mindestens $f^{-1}(j)$.

Es bleibt noch der Fall $j = k+1$. Jede Teilsequenz mit $k+1$ verschiedenen Anfragen muss mindestens $k+1$ Blöcke überdecken, wobei die Randblöcke nur zum Teil überdeckt zu werden brauchen. Die $k-1$ inneren Blöcke einer solchen Teilsequenz bestehen aus $k-1$ aufeinander folgenden Blöcken und sind daher mindestens so lang wie eine Phase. In den Randblöcken muss noch jeweils mindestens eine Anfrage überdeckt werden. Daher hat die Teilsequenz mindestens die Länge $(f^{-1}(k+1) - 2) + 2 = f^{-1}(k+1)$. □

16.3 LRU und Markierungsstrategien

Für obere Schranken kehren wir das Beweiskonzept der unteren Schranken gewissermaßen um. Wir benutzten die Seitenfehler der untersuchten Strategie, um die Sequenz in Phasen einzuteilen. Wenn wir z. B. jeweils nach einer festen Zahl an Seitenfehlern eine Phasengrenze ziehen, dann lassen sich häufig mithilfe der Eigenschaften der untersuchten Strategie Aussagen über die Zahl der in einer Phase angefragten Seiten gewinnen. Daraus können wir dann mithilfe von f^{-1} Abschätzungen für die Phasenlängen und damit auch die Fehlerrate erhalten. Beim Beweis oberer Schranken nutzen wir die Anfangsbelegung des Caches nicht aus. Man darf sich eine beliebige Anfangsbelegung vorstellen, sogar dass der Cache zu Beginn leer ist.

Theorem 16.4. *Für die Fehlerrate von LRU bez. f gilt im Maximumsmodell*

$$F_{\mathrm{LRU}}(f) \leq \frac{k-1}{f^{-1}(k+1)-2}.$$

Beweis. Es sei σ eine beliebige Anfragesequenz, die f genügt. Wir teilen σ in Phasen ein, sodass für LRU in jeder Phase (evtl. mit Ausnahme der letzten Phase) genau $k-1$ Seitenfehler auftreten. Dabei gehen wir wie folgt vor. Die erste Phase beginnt mit der ersten Anfrage und endet unmittelbar vor der Anfrage, die den k-ten Fehler verursacht. Im Allgemeinen beginnt die i-te Phase, $i \geq 2$, mit dem $((i-1)(k-1)+1)$-ten Fehler und endet unmittelbar vor dem $(i(k-1)+1)$-ten Fehler. Die letzte Phase mag dabei unvollständig sein. Auf jeden Fall hat LRU höchstens Kosten $k-1$ pro Phase. Wir zeigen, dass mit Ausnahme der ersten und letzten Phase jede Phase mindestens Länge $f^{-1}(k+1)-2$ hat.

Wir betrachten eine beliebige Phase P, die nicht die erste oder letzte Phase sei. Wir zeigen, dass die Teilsequenz von σ, die mit der letzten Anfrage vor P beginnt und nach der ersten Anfrage nach P beendet ist, $k+1$ verschiedene Seiten anfragt. Dann hat P mindestens die Länge $f^{-1}(k+1)-2$.

Es sei x die Seite, die in der letzten Anfrage vor P angefragt wird. Daher ist die Seite x zu Beginn der Phase im Cache. Die Phase P und die erste Anfrage nach P umfassen k Seitenfehler. Falls diese k Seitenfehler von Anfragen an k paarweise verschiedene Seiten verursacht sind und diese zudem noch alle verschieden von x sind, ist nichts mehr zu zeigen. Falls einer dieser k Fehler durch eine Anfrage an x verursacht ist, dann wurde x irgendwann in der Phase bei einer Anfrage an eine Seite $y \neq x$ aus dem Cache entfernt. Zu diesem Zeitpunkt muss es seit dem Beginn der Phase k Anfragen an k verschiedene Seiten (inklusive y) gegeben haben, die alle verschieden von x sind. Andernfalls wäre x nicht die Seite, deren letzte Anfrage am weitesten zurückliegt. Damit haben wir $k+1$ verschiedene Seiten in der betrachteten Teilsequenz gefunden. Dasselbe Argument greift, falls LRU in dieser Teilsequenz zweimal einen Seitenfehler bei einer Anfrage an eine Seite $z \neq x$ hat.

Wir haben σ also höchstens in

$$1 + \left\lceil \frac{|\sigma| - \ell_1}{f^{-1}(k+1) - 2} \right\rceil \le \frac{|\sigma|}{f^{-1}(k+1) - 2} + 2$$

Phasen eingeteilt, wobei ℓ_1 die Länge der ersten Phase bezeichnet. In jeder Phase gibt es höchstens $k-1$ Seitenfehler. Somit ist die Fehlerrate für jede Sequenz σ, die f genügt, höchstens

$$F_{\mathrm{LRU}}(\sigma) \le \frac{k-1}{f^{-1}(k+1) - 2} + \frac{2k-2}{|\sigma|}.$$

Der zweite Summand wird mit wachsender Sequenzlänge $|\sigma|$ immer kleiner. Daraus folgt die behauptete Schranke für $F_{\mathrm{LRU}}(f)$. □

Aus der unteren Schranke aus Theorem 16.3 und der oberen Schranke aus Theorem 16.4 erhalten wir das folgende Korollar, welches auch zeigt, dass die untere Schranke für Online-Strategien nicht größer sein kann.

Korollar 16.5. *Für die Fehlerrate von LRU bez. f gilt im Maximumsmodell*

$$F_{\mathrm{LRU}}(f) = \frac{k-1}{f^{-1}(k+1) - 2}.$$

Wir kennen also nun die exakte Fehlerrate der Markierungsstrategie LRU bez. f im Maximumsmodell und wissen, dass LRU eine optimale Strategie in diesem Modell ist. Innerhalb der Klasse der Markierungsstrategien sind aber nicht alle Strategien gleich gut. Das wollen wir im Folgenden zeigen. Wir beginnen mit einer oberen Schranke für Markierungsstrategien.

Theorem 16.6. *Es sei $\mathcal{M}$ eine beliebige Markierungsstrategie. Für die Fehlerrate von $\mathcal{M}$ bez. f gilt im Maximumsmodell*

$$F_{\mathcal{M}}(f) \le \frac{k}{f^{-1}(k+1) - 1}.$$

Beweis. Die Markierungsstrategie $\mathcal{M}$ teilt die Sequenz σ in Phasen ein, sodass jede Phase (evtl. mit Ausnahme der letzten Phase) Anfragen an k verschiedene Seiten enthält und darüber hinaus die erste Anfrage in jeder Sequenz einen Seitenfehler erzeugt (evtl. mit Ausnahme der ersten Phase). Jede Teilsequenz, die mit der ersten Anfrage einer Phase beginnt und bis zur ersten Anfrage der nächsten Phase reicht (und diese einschließt) hat mindestens Länge $f^{-1}(k+1)$. Der Grund ist, dass innerhalb der Phase k verschiedene Seiten angefragt werden und die erste Anfrage der folgenden Phase die Phase beendet, weil sie die $(k+1)$-ste neue Seite anfragt. Also haben alle bis auf die letzte Phase mindestens die Länge $f^{-1}(k+1) - 1$. (Die letzte Aussage gilt offensichtlich auch für die erste Phase.) Die Strategie $\mathcal{M}$ teilt σ in höchstens

$$\left\lceil \frac{|\sigma|}{f^{-1}(k+1)-1} \right\rceil \leq \frac{|\sigma|}{f^{-1}(k+1)-1} + 1$$

Phasen ein, wobei jede Phase höchstens k Seitenfehler enthält. Daher ist die Fehlerrate von $\mathcal{M}$ für jede Sequenz σ, die f genügt, durch

$$\frac{k}{|\sigma|}\left(\frac{|\sigma|}{f^{-1}(k+1)-1} + 1\right) \leq \frac{k}{f^{-1}(k+1)-1} + \frac{k}{|\sigma|}$$

beschränkt. Daraus folgt das Resultat für $F_{\mathcal{M}}(f)$. □

Im nächsten Schritt zeigen wir, dass diese obere Schranke scharf ist, d. h., es gibt Markierungsstrategien, deren Fehlerrate tatsächlich nicht besser ist.

Theorem 16.7. *Es gibt Markierungsstrategien $\mathcal{M}^*$, für deren Fehlerrate bez. f im Maximumsmodell gilt:*

$$F_{\mathcal{M}^*}(f) = \frac{k}{f^{-1}(k+1)-1}.$$

Eine Markierungsvariante von Flush When Full gehört zu dieser Klasse von Markierungsstrategien.

Beweis. Die obere Schranke folgt aus Theorem 16.6. Für die untere Schranke konstruieren wir eine Folge von Anfragesequenzen wachsender Länge n und beschreiben gleichzeitig, wie sich eine Strategie $\mathcal{M}^*$ auf einer Sequenz $\sigma = \sigma_1, \ldots, \sigma_n$ der Folge verhält. Die Sequenz σ unterteilen wir in Phasen, sodass je eine Runde der Markierungsstrategie $\mathcal{M}^*$ genau einer Phase entspricht. Jede Phase besteht aus k homogenen Blöcken, d. h., innerhalb eines Blocks wird immer dieselbe Seite angefragt. Innerhalb einer Phase hat deren j-ter Block, $1 \leq j \leq k$, die Länge $f^{-1}(j+1)-f^{-1}(j)$. Unsere Blocklängen sind wohldefiniert, da der erste Block einer Phase eine positive Länge hat und Proposition 16.2 sichert, dass innerhalb einer Phase der jeweils nächste Block nicht kürzer als sein Vorgänger ist. Eine Phase umfasst also $f^{-1}(k+1)-1$ Anfragen.

Nachdem wir die Blocklängen festgelegt haben, konstruieren wir nun die Anfragen, mit denen die Blöcke auszufüllen sind. Gleichzeitig beschreiben wir das Verhalten von $\mathcal{M}^*$. Wie gewohnt fragen wir lediglich $k+1$ verschiedene Seiten $p_1, \ldots, p_{k+1}$ an. Generell stellt der jeweils nächste Block Anfragen an diejenige Seite, die im vorhergehenden Block nicht im Cache war; im Falle des ersten Blocks der ersten Phase an die Seite, die zu Beginn nicht im Cache liegt. So ist sichergestellt, dass jeder Block mit seiner ersten Anfrage einen Seitenfehler verursacht, und wir erhalten die behauptete untere Schranke für die Fehlerrate.

Für Strategien $\mathcal{M}^*$ genügt es, festzulegen, wie sie sich bei dem Seitenfehler verhält, der durch die erste Anfrage im ersten Block einer Phase verursacht wird. Hier legen wir fest, dass diejenige Seite aus dem Cache entfernt wird, die im letzten Block der vorhergehenden Phase angefragt wurde; im Falle des ersten Blocks der ersten Phase eine beliebige Seite aus dem Cache. Bei allen weiteren Fehlern in der Phase darf $\mathcal{M}^*$ eine beliebige unmarkierte Seite aus dem Cache auswählen. Da es sich bei $\mathcal{M}^*$ um eine Markierungsstrategie handelt, ist gesichert, dass alle k Seitenfehler einer Runde von verschiedenen Seiten verursacht werden. Daraus ergibt sich, dass alle k Blöcke einer Phase ebenso viele verschiedene Seiten anfragen. Es ist klar, dass FWF die Bedingung, die wir an $\mathcal{M}^*$ gestellt haben, erfüllt, da FWF zu Beginn jeder Phase eigentlich sogar den Cache leert. In Abschnitt 15.2 haben wir uns schon überlegt, dass wir uns FWF auch als Markierungsstrategie vorstellen können. Offensichtlich gibt es auch eine Markierungsvariante von FWF, die die Bedingung an $\mathcal{M}^*$ erfüllt. In der oben konstruierten Sequenz gibt es keine Anfragen an unmarkierte Seiten im Cache, da jeder neue Block stets nach derjenigen Seite fragt, die nicht im Cache liegt. Daher erzeugt die Markierungsvariante von FWF an denselben Stellen Seitenfehler wie FWF.

Nun müssen wir noch zeigen, dass die konstruierte Anfragesequenz für alle Markierungsstrategien, die die obige Bedingung erfüllen, f genügt. Wir zeigen, dass jede Teilsequenz mit Anfragen an j verschiedene Seiten, $1 \leq j \leq k+1$, eine Länge von mindestens $f^{-1}(j)$ hat. Für $j \in \{1, 2\}$ ist dies wieder offensichtlich. Für $3 \leq j \leq k+1$ gilt, dass jede Teilsequenz mit j verschiedenen Seiten j aufeinander folgende Blöcke zumindest teilweise überdecken muss, d. h., die Randblöcke müssen nicht vollständig überdeckt sein. Da die Blocklängen innerhalb einer Phase nicht fallend sind und die im zweiten Block angefragte Seite gleich der im letzten Block der Vorgängerphase ist, beginnt eine Teilsequenz minimaler Länge mit j verschiedenen Seiten am Anfang einer Phase und endet unmittelbar nach der ersten Anfrage im j-ten Block derselben Phase. Die Länge einer solchen Teilsequenz ist daher mindestens $f^{-1}(j)$. Im letzten Fall, $j = k+1$, müssen wir zwei aufeinander folgende Phasen im Blick haben. Falls die betrachtete Teilsequenz eine Phase vollständig umfasst, beträgt ihre Länge die Länge einer Phase zuzüglich mindestens einer weiteren Anfrage. Insgesamt kommen wir dann auf eine Länge von mindestens $f^{-1}(k+1)$. Andernfalls überdeckt die Teilsequenz zwei aufeinander folgende Phasen i und $i+1$ jeweils nur teilweise. Die Teilsequenz berührt dann in jedem Fall den letzten Block der Phase i, aber nicht den ersten Block der Phase i, da dieser nur aus einer Anfrage besteht und wir sonst im zuvor betrachteten Fall wären. Da in Phase i daher noch höchstens Anfragen an $k-1$ verschiedene Anfragen überdeckt werden können, müssen in Phase $i+1$ noch Anfragen an mindestens zwei neue Seiten überdeckt werden. Dazu müssen mindestens die ersten drei Blöcke von Phase $i+1$ zumindest teilweise überdeckt werden, weil die Seite, die im letzten Block von Phase i angefragt wird, dieselbe ist, die im zweiten Block von Phase $i+1$ angefragt wird. Insgesamt muss die Teilsequenz also $k+2$ aufeinander folgende Blöcke berühren, wobei der erste Block nicht der erste Block einer Phase ist. Da k aufeinander folgende Blöcke genauso lang sind wie eine Phase, hat die Teilsequenz sogar mindestens Länge $(f^{-1}(k+1)-1)+2 \geq f^{-1}(k+1)$. □

Was haben wir nun erreicht? Wir haben Markierungsstrategien $\mathcal{M}^*$ gefunden, deren Fehlerrate bez. f genau $k/(f^{-1}(k+1)-1)$ ist. Es ist vielleicht nicht unmittelbar klar, ob diese Fehlerrate größer als die Fehlerrate $(k-1)/(f^{-1}(k+1)-2)$ der Markierungsstrategie LRU ist. Falls $f^{-1}(k+1) = k+1$ ist, sind beide Ausdrücke gleich. Das kann nicht überraschen, denn das bedeutet, dass es in der Sequenz in Fenstern der Länge $k+1$ keine Anfragelokalität zu geben braucht. Mit nur $k+1$ Seiten kann man sowohl für LRU als auch für alle anderen Markierungsstrategien mit jeder neuen Anfrage einen Seitenfehler erzeugen, d. h. Fehlerrate 1 erzwingen. Wenn aber $f^{-1}(k+1) > k+1$ gilt, dann gibt es in der Sequenz Anfragelokalität in Fenstern der Länge $k+1$. Für $k \geq 2$ kann man leicht nachrechnen, dass in diesem Fall die Markierungsstrategien $\mathcal{M}^*$ schlechter sein können als LRU:

$$\frac{F_{\mathcal{M}^*}(f)}{F_{\mathrm{LRU}}(f)} = \frac{f^{-1}(k+1)-2}{f^{-1}(k+1)-1} \cdot \frac{k}{k-1} > \frac{k+1-2}{k+1-1} \cdot \frac{k}{k-1} = 1.$$

16.4 FIFO

Als nächste Strategie untersuchen wir FIFO, die sich als etwas schlechter als LRU erweisen wird.

Theorem 16.8. *Für die Fehlerrate von FIFO bez. f gilt im Maximumsmodell*

$$F_{\mathrm{FIFO}}(f) \leq \frac{k}{f^{-1}(k+1)-1}.$$

Beweis. Wenn wir $k+1$ aufeinander folgende Seitenfehler von FIFO betrachten, dann sind diese Fehler alle durch Anfragen an verschiedene Seiten entstanden. Wir teilen die Sequenz so in Phasen ein, dass jede Phase genau k Seitenfehler enthält und mit einem Seitenfehler beginnt; die erste Phase beginnt mit der ersten Anfrage der Sequenz. Jetzt betrachten wir eine Teilsequenz, die aus einer Phase und zusätzlich der ersten Anfrage der folgenden Phase besteht. Diese Teilsequenz überdeckt also $k+1$ aufeinander folgende Seitenfehler und deswegen Anfragen an mindestens ebenso viele verschiedene Seiten. Daher hat eine Phase mindestens die Länge $f^{-1}(k+1)-1$. □

Die Fehlerrate bez. f ist für FIFO also im Maximumsmodell nicht schlechter als für einige Markierungsstrategien. Jedoch ist FIFO i. Allg. nicht optimal.

Theorem 16.9. *Falls $f^{-1}(4)-f^{-1}(3) > f^{-1}(3)-f^{-1}(2)$, dann gilt für die Fehlerrate bez. f für FIFO im Maximumsmodell*

$$F_{\text{FIFO}}(f) \geq \frac{k-1/k}{f^{-1}(k+1)-1}.$$

Wir behaupten also, dass diese untere Schranke (unter der im Theorem 16.9 formulierten Bedingung) größer als die Fehlerrate von LRU bez. f sein kann. Zunächst einmal stellen wir fest, dass die Schranke nicht kleiner als die Fehlerrate von LRU bez. f ist. Dazu überprüft man, dass

$$\frac{k-1}{f^{-1}(k+1)-2} \leq \frac{k-1/k}{f^{-1}(k+1)-1} \tag{$*$}$$

gilt. Diese Ungleichung lässt sich zu $f^{-1}(k+1) \geq k+2$ umformen. Aus der Bedingung in Theorem 16.9 folgt, dass $f^{-1}(4) \geq 5$ gilt. Daraus und aus dem streng monotonen Wachstum von f^{-1} (Proposition 16.2) folgt, dass $f^{-1}(k+1) \geq k+2$ gilt und somit Ungleichung ($*$). Für Funktionen f, für die sogar $f^{-1}(k+1) > k+2$ gilt, erhalten wir dann untere Schranken für FIFO, die zeigen, dass FIFO im Maximumsmodell echt höhere Fehlerraten bez. f als LRU haben kann.

Beweis zu Theorem 16.9. Wieder benutzen wir Anfragen an $k+1$ verschiedene Seiten, die mit $p_0, \ldots, p_k$ bezeichnet werden sollen. Die Seite p_k spielt eine besondere Rolle. Die erste Anfrage geht an diese Seite p_k und ist keiner Phase zugeordnet. Danach folgen Phasen, die beliebig oft wiederholt werden können. Wir betrachten hier nur solche Sequenzen, bei denen die Zahl der Phasen ein Vielfaches von k ist. Jede Phase besteht aus $k-1$ Blöcken, die diesmal nicht homogen sind; siehe Bild 16.2 für die erste Phase. Jeder Block beginnt mit einer Anfrage an eine Seite p_i, $0 \leq i \leq k-1$, gefolgt von mindestens einer Anfrage an die Seite p_k, d. h., jeder Block wird nach der ersten Anfrage mit Anfragen an p_k aufgefüllt. In der Folge der Blöcke werden für die jeweils erste Anfrage die Seiten $p_0, \ldots, p_{k-1}$ zyklisch durchlaufen. Im j-ten Block der Anfragesequenz, $j \geq 1$, geht die erste Anfrage an die Seite $p_{(j-1) \bmod k}$. Die Blocklängen sind wie folgt definiert. Der jeweils erste Block einer Phase hat die Länge $f^{-1}(3) - f^{-1}(2) + 1 \geq 2$ und der j-te Block, $2 \leq j \leq k-1$, hat die Länge $f^{-1}(j+2) - f^{-1}(j+1)$. Wegen Proposition 16.2 und der in Theorem 16.9 geforderten Eigenschaften ist der jeweils folgende Block in einer Phase nicht kürzer als sein Vorgänger. Jeder Block enthält also mindestens eine Anfrage an p_k. Die Länge einer Phase ist $f^{-1}(k+1) - f^{-1}(2) + 1 = f^{-1}(k+1) - 1$.

Ohne Beschränkung der Allgemeinheit dürfen wir annehmen, dass die Seite p_k diejenige Seite ist, die zu Beginn nicht im Cache liegt, sodass die erste Anfrage einen Seitenfehler verursacht. Im Rest des Beweises zeigen wir, dass die eben konstruierte Sequenz f genügt und dass in k aufeinander folgenden Phasen – das nennen wir

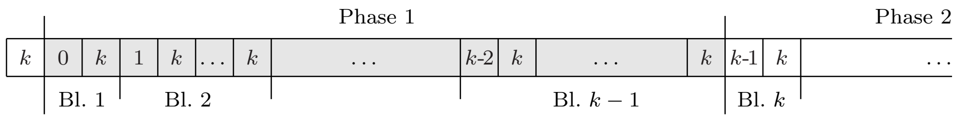

Bild 16.2: Die erste Phase der Anfragesequenz aus dem Beweis zu Theorem 16.9

eine *Superphase* – FIFO mindestens $(k-1)(k+1)$ Seitenfehler hat. Daraus folgt unmittelbar die untere Schranke für die Fehlerrate

$$\frac{(k-1)(k+1)}{k(f^{-1}(k+1)-1)} = \frac{k-1/k}{f^{-1}(k+1)-1}.$$

Wir zeigen, dass jede Teilsequenz mit j verschiedenen Anfragen eine Länge von mindestens $f^{-1}(j)$ hat. Der Fall $j \in \{1, 2\}$ ist wieder offensichtlich. Es sei nun j aus $\{3, \dots, k\}$. Jede Teilsequenz mit Anfragen an j verschiedene Seiten umfasst mehr als einen Block. Eine solche Teilsequenz minimaler Länge beginnt nicht mit einem Präfix aus Anfragen an p_k, weil diese Seite ohnehin im nächsten Block enthalten ist. Sie beginnt also mit der ersten Anfrage in einem Block und erstreckt sich bis mindestens zur ersten Anfrage des $(j-2)$-ten Blocks nach diesem Block. Weil die Blocklängen innerhalb einer Phase nicht fallend sind, hat die Teilsequenz also mindestens die Länge der ersten $j-2$ Blöcke einer Phase zuzüglich der ersten Anfrage im folgenden Block. Das führt mindestens zur Länge $(f^{-1}(j) - f^{-1}(2) + 1) + 1 = f^{-1}(j)$. Schlussendlich muss jede Teilsequenz mit Anfragen an $j = k+1$ verschiedene Seiten mehr als eine Phase umfassen und hat somit mindestens die Länge $f^{-1}(k+1)$.

Es bleibt noch die Zahl der Seitenfehler in einer Superphase zu ermitteln. Ohne Beschränkung der Allgemeinheit sei zu Beginn p_0 die älteste Seite im Cache, p_1 die zweitälteste usw. bis p_{k-1}. FIFO hat also Seitenfehler bei der ersten Anfrage, nämlich an p_k, dann bei der ersten Anfrage im ersten Block, nämlich an p_0, und danach bei der ersten Anfrage im zweiten Block, nämlich an p_1. Wir zeigen induktiv, dass unsere Vermutung gilt, dass FIFO bei jeder ersten Anfrage eines Blocks einen Seitenfehler hat. Für die ersten k Blöcke ist dies leicht einzusehen. Wenn man nur die Anfragen mit Seitenfehlern betrachtet, gilt für FIFO, dass sich in k aufeinander folgenden Anfragen keine Seite wiederholt. Angenommen, FIFO hat einen Seitenfehler bei der ersten Anfrage in Block j, $j \geq k$. Wenn die Seite p_i mit $i < k-1$ angefragt wird, dann ist die Seite p_{i+1} die älteste und wird aus dem Cache entfernt. Diese wird jedoch zu Beginn von Block $j+1$ angefragt. Falls die Seite p_{k-1} angefragt wird, dann wird p_k entfernt, jedoch wird p_k noch im selben Block erneut angefragt. Dies führt zu einem neuen Seitenfehler, bei dem p_0 entfernt wird. Die „Fehlerreihenfolge" ist also $p_k, p_0, p_1, \dots, p_{k-1}$ und dann zyklisch immer so weiter. Wir haben sogar gezeigt, dass FIFO in jedem Block, in dem p_{k-1} angefragt wird, zwei Seitenfehler hat, nämlich bei der Anfrage an p_{k-1} und bei der ersten Anfrage an p_k. In k aufeinander folgenden Phasen tritt ein solcher Block in $k-1$ Phasen je einmal auf. Daher ist in jeder Superphase die Seitenfehlerzahl $k(k-1) + k - 1 = (k+1)(k-1)$. $\square$

16.5 Die optimale Offline-Strategie LFD

Für LFD hängen unsere Schranken auch von der Zahl N der in der Sequenz angefragten Seiten ab. (Die Zahl N ist nicht zu verwechseln mit der Sequenzlänge $n = |\sigma|$.) Falls f es nur gestattet, eine endliche Zahl von Seiten anzufragen, bezeichnen wir den größtmöglichen Wert von N mit M. Die Fälle mit $k \geq M$ sind nicht interessant, da hier die Fehlerrate bez. f für Demand-Paging-Strategien ohnehin immer 0 ist.

Theorem 16.10. *Falls* $|\operatorname{Bild}(f)|$ *endlich ist, sei* M *die größte natürliche Zahl in* $\operatorname{Bild}(f)$. *Für die Fehlerrate von LFD bez.* f *gilt dann im Maximumsmodell für* $k < M$

$$F_{\mathrm{LFD}}(f) \geq \max_{\substack{m \in \mathbb{N} \\ k+m \leq M}} \left\{ \frac{m}{f^{-1}(k+m+1)-2} \right\}.$$

Andernfalls entfällt die Bedingung $k + m \leq M$ *bei der Maximumsbildung.*

Beweis. Wir betrachten ein festes $m \in \mathbb{N}$ und benutzen $N := k + m$ Seiten $p_0, \ldots, p_{N-1}$. Im dem Fall, dass M existiert, betrachten wir nur solche m, für die $k + m \leq M$ gilt. Wie üblich konstruieren wir Anfragesequenzen, die aus Phasen zusammengesetzt sind. Jede Phase hat eine Länge von $f^{-1}(N+1) - 2$ und besteht aus $N - 1$ Blöcken, in denen jeweils nur eine Seite angefragt wird. Der j-te Block einer Phase, $j = 1, \ldots, N-1$, hat die Länge $f^{-1}(j+2) - f^{-1}(j+1)$. In den Blöcken der Sequenz werden die Seiten $p_0, \ldots, p_{N-1}$ zyklisch angefragt, d. h., der j-te Block der Sequenz, $j \geq 1$, enthält nur Anfragen an die Seite $p_{(j-1) \bmod N}$.

Wir wissen, dass LFD am Ende einer Phase zumindest die Seite im Cache vorhält, die im letzten Block der Phase angefragt wurde. Wegen der zyklischen Anfragereihenfolge wird diese Seite in der folgenden Phase jedoch nicht angefragt. Von den $N-1$ verschiedenen Anfragen der Folgephase kann LFD also höchstens $k-1$ ohne Seitenfehler bedienen und hat daher mindestens $(N-1)-(k-1) = m$ Seitenfehler pro Phase. Das gilt auch für die erste Phase, denn o. B. d. A. dürfen wir annehmen, dass p_N zu Beginn im Cache ist. (Wenn wir annehmen wollen, dass der Cache zu Beginn leer ist, dann verursacht sogar jeder Block der ersten Phase einen Seitenfehler.) Daraus folgt die behauptete untere Schranke für die Fehlerrate. Schließlich folgt auf die gleiche Art wie im Beweis zu der unteren Schranke für Online-Strategien (Theorem 16.3), dass die konstruierte Anfragesequenz f genügt. $\square$

Wir beweisen im Folgenden eine obere Schranke für LFD, die im Wesentlichen um den Faktor 2 über unserer unteren Schranke liegt. Im Beweis werden wir die folgende Aussage benutzen, die vielleicht sogar intuitiv klar ist, wenn wir uns erinnern, dass f^{-1} konvex ist. Die Aussage folgt aus der Konvexität von f^{-1}.

Proposition 16.11. *Es sei f eine Funktion, die konkav*** *ist. Für natürliche Zahlen $m_1, \ldots, m_n$ und deren arithmetisches Mittel $\overline{m}$ gilt*

$$f^{-1}(m_1) + \cdots + f^{-1}(m_n) \;\geq\; n \cdot f^{-1}\left(\lfloor \overline{m} \rfloor\right).$$

Beweis. Wir manipulieren schrittweise die linke Seite der behaupteten Ungleichung, wobei diese in jedem Schritt durch einen Ausdruck ersetzt wird, der nicht größer ist als zuvor. Zu jeder Zeit haben wir eine Summe aus n Summanden $f^{-1}(\widetilde{m}_1) + \cdots + f^{-1}(\widetilde{m}_n)$, wobei die Argumente $\widetilde{m}_i$ natürliche Zahlen sind. Zu Beginn gilt $\widetilde{m}_i = m_i$ für $1 \leq i \leq n$. Nun betrachten wir einen Ersetzungsschritt. Es sei $m' = \min\{\widetilde{m}_1, \ldots, \widetilde{m}_n\}$ und $m'' = \max\{\widetilde{m}_1, \ldots, \widetilde{m}_n\}$. Falls $m'' - m' \geq 2$ ist, ersetzen wir zwei Summanden $f^{-1}(m')$ und $f^{-1}(m'')$ durch $f^{-1}(m'+1)$ und $f^{-1}(m''-1)$. Wegen Proposition 16.2 gilt

$$\begin{aligned} f^{-1}(m') + f^{-1}(m'') &= f^{-1}(m'+1) - \big(f^{-1}(m'+1) - f^{-1}(m')\big) + \\ &\quad\; f^{-1}(m''-1) + \big(f^{-1}(m'') - f^{-1}(m''-1)\big) \\ &\geq f^{-1}(m'+1) + f^{-1}(m''-1), \end{aligned}$$

sodass die Summe der $f^{-1}(\widetilde{m}_i)$ bei der Ersetzung nicht wächst. Ferner bleibt die Summe der Argumente $\widetilde{m}_i$ unverändert, sodass immer $\widetilde{m}_1 + \cdots + \widetilde{m}_n = n \cdot \overline{m}$ gilt. Sobald $m'' - m' \leq 1$ erfüllt ist, stoppt das Verfahren. Dann lässt sich die Summe der Argumente $\widetilde{m}_i$ für ein k aus $\{1, \ldots, n\}$ als $k \cdot \lfloor \overline{m} \rfloor + (n-k) \cdot (\lfloor \overline{m} \rfloor + 1)$ darstellen und die Summe der $f^{-1}(\widetilde{m}_i)$ als

$$k \cdot f^{-1}(\lfloor \overline{m} \rfloor) + (n-k) \cdot f^{-1}(\lfloor \overline{m} \rfloor + 1).$$

Aus $f^{-1}(\lfloor \overline{m} \rfloor + 1) \geq f^{-1}(\lfloor \overline{m} \rfloor)$ folgt nun die behauptete Ungleichung. □

Theorem 16.12. *Falls* $|\operatorname{Bild}(f)|$ *endlich ist, sei M die größte natürliche Zahl in* $\operatorname{Bild}(f)$*. Für die Fehlerrate von LFD bez. f gilt dann im Maximumsmodell*

$$F_{\mathrm{LFD}}(f) \;\leq\; 2 \max_{\substack{1 \leq m \leq k \\ k+m \leq M}} \left\{ \frac{m+1}{f^{-1}(k+m)} \right\}.$$

Andernfalls entfällt die Bedingung $k + m \leq M$ bei der Maximumsbildung.

Beweis. Wir teilen die Sequenz von vorne beginnend so in Phasen ein, dass jede Phase (bis auf eventuell die letzte Phase) Anfragen an k verschiedene Seiten enthält und die erste Anfrage der jeweils nächsten Phase verschieden von diesen k Anfragen ist. Die letzte Eigenschaft stellen wir dadurch sicher, dass wir jede Phase so lang wie möglich wählen. Die Zahl der so entstehenden Phasen sei p; die Phasen nummerieren wir mit 1 bis p durch. Für jede Phase $i \geq 2$ sei $m_i \leq k$ die Anzahl der Anfragen, die an Seiten gerichtet sind, die nicht schon in Phase $i-1$ angefragt wurden. Diese

Seiten nennen wir die *neuen* Seiten in Phase i. Für die erste Phase setzen wir $m_1 := k$.

Nun betrachten wir folgende Offline-Strategie. Bei einem Seitenfehler wird eine beliebige Seite aus dem Cache entfernt, die nicht in derselben Phase angefragt wird. So eine Seite existiert, da in jeder Phase nur k verschiedene Seiten angefragt werden. Diese Strategie hat offensichtlich in Phase $i \geq 2$ genau m_i Seitenfehler und in Phase 1 höchstens m_1 Seitenfehler. Auf der gesamten Sequenz hat die optimale Offline-Strategie LFD höchstens so viele Seitenfehler wie die gerade betrachtete Strategie. Die Zahl der Seitenfehler von LFD ist also durch $m_1 + \cdots + m_p = k + (p-1)\overline{m}$ beschränkt, wobei $\overline{m} = (m_2 + \cdots + m_p)/(p-1)$ die durchschnittliche Anzahl neuer Seiten pro Phase (ohne die erste Phase) ist.

Je zwei aufeinander folgende Phasen $i-1$ und i enthalten Anfragen an $k + m_i$ verschiedene Seiten und haben daher eine Länge von $|P_{i-1}| + |P_i| \geq f^{-1}(k + m_i)$, wobei $|P_i|$ die Länge der i-ten Phase bezeichnet. Hierbei ist klar, dass f^{-1} an der Stelle $k + m_i$ definiert ist, denn die betrachtete Sequenz genügt f. Auf jeden Fall gilt $k+m_i \leq 2k$ und, falls M existiert, gilt zusätzlich $k+m_i \leq M$. Die Gesamtlänge von σ lässt sich nun wie folgt abschätzen:

$$|\sigma| \;=\; \sum_{i=1}^{p} |P_i| \;=\; \frac{1}{2}\sum_{i=2}^{p}(|P_{i-1}| + |P_i|) + \frac{1}{2}(|P_1| + |P_p|) \;>\; \frac{1}{2}\sum_{i=2}^{p} f^{-1}(k + m_i).$$

Mithilfe von Proposition 16.11 erhalten wir

$$|\sigma| \;>\; \frac{p-1}{2} f^{-1}\left(\left\lfloor \frac{(k+m_2) + \cdots + (k+m_p)}{p-1} \right\rfloor\right) \;=\; \frac{p-1}{2} f^{-1}(k + \lfloor \overline{m} \rfloor).$$

Die Fehlerrate von LFD für eine Sequenz σ, die f genügt, ist also durch

$$F_{\mathrm{LFD}}(\sigma) \;\leq\; \frac{k + (p-1)\overline{m}}{|\sigma|} \;\leq\; \frac{2\overline{m}}{f^{-1}(k + \lfloor \overline{m} \rfloor)} + \frac{k}{|\sigma|}$$

beschränkt. Der Summand $k/|\sigma|$ verschwindet mit wachsender Sequenzlänge und wir erhalten

$$F_{\mathrm{LFD}}(f) \;\leq\; \frac{2\overline{m}}{f^{-1}(k + \lfloor \overline{m} \rfloor)} \;\leq\; 2\frac{\lfloor \overline{m} \rfloor + 1}{f^{-1}(k + \lfloor \overline{m} \rfloor)} \;\leq\; 2 \max_{\substack{1 \leq m \leq k \\ k+m \leq M}} \left\{ \frac{m+1}{f^{-1}(k+m)} \right\}.$$

Da wir $\lfloor \overline{m} \rfloor$ nicht kennen, können wir auf die rechte Seite der letzten Ungleichung nur den größtmöglichen Wert schreiben, den der Bruch auf der linken Seite für alle zulässigen Werte von $\lfloor \overline{m} \rfloor$ annimmt. Aus der Definition einer Phase folgt, dass der Wert von $\lfloor \overline{m} \rfloor$ zwischen 1 und k liegt. □

17 Analyse der Fehlerrate im Durchschnittsmodell

Im Durchschnittsmodell untersuchen wir die Fehlerrate bez. Funktionen, die konkav* sind. Wir wollen uns kurz die schon in Abschnitt 15.4 eingeführte Notation in Erinnerung rufen. Wie zuvor bezeichnet σ_t die t-te Anfrage der Anfragesequenz σ. Mit $\sigma_t(\ell)$ bezeichnen wir das Fenster aus den ℓ Anfragen $\sigma_t, \ldots, \sigma_{t+\ell-1}$. Die Arbeitsmenge $W_t(\ell)$ ist die Menge der in $\sigma_t(\ell)$ angefragten Seiten und $N_t(\ell) := |W_t(\ell)|$ die Zahl der in $W_t(\ell)$ enthaltenen Seiten. Mit

$$N(\ell) \;:=\; \sum_{t=1}^{|\sigma|-\ell+1} N_t(\ell) \qquad \text{und} \qquad \mathrm{Av}(\ell) \;:=\; \frac{N(\ell)}{|\sigma|-\ell+1}$$

wird die Summe der Arbeitsmengengrößen über alle möglichen Fenster der Länge ℓ in der Sequenz bzw. die durchschnittliche Arbeitsmengengröße über alle diese Fenster bezeichnet. Eine Sequenz σ genügt einer Funktion f, wenn für alle ℓ, $1 \le \ell \le |\sigma|$, $\mathrm{Av}(\ell) \le f(\ell)$ gilt. Wir treffen folgende Vereinbarung.

In diesem Kapitel bezeichnet f eine beliebige Funktion, die konkav* ist.

17.1 Eine größtmögliche untere Schranke

Das Ziel ist es, eine untere Schranke von $(f(k+1)-1)/k$ für jede deterministische Online-Strategie zu zeigen. Dazu konstruieren wir wieder eine geeignete Folge von Anfragesequenzen aus Sequenzen wachsender Länge n. Das Studium der folgenden Sequenz, die nur $k+1$ verschiedene Seiten anfragt, wird uns dabei als Ausgangspunkt dienen:

$$\sigma(i,j) \;:=\; (1,2,3,\ldots,k,k+1)^i(1)^j, \quad i \ge k+2,\; j \ge k+1.$$

Dabei bedeutet $(\ldots)^i$ die i-fache Aneinanderreihung der geklammerten Teilsequenz. Diese Sequenz besteht also aus einem Präfix, in dem alle $k+1$ Seiten zyklisch angefragt werden, und einem Suffix, in dem nur noch die Seite p_1 angefragt wird. Das Präfix dient dazu, Seitenfehler zu erzeugen, während mit der Suffixlänge die durchschnittliche Zahl angefragter Seiten gesteuert werden kann.

Ein wesentlicher Bestandteil des Beweises der unteren Schranke ist es, zu zeigen, dass man für jede gegebene Funktion f, die konkav* ist, durch geeignete Wahl der

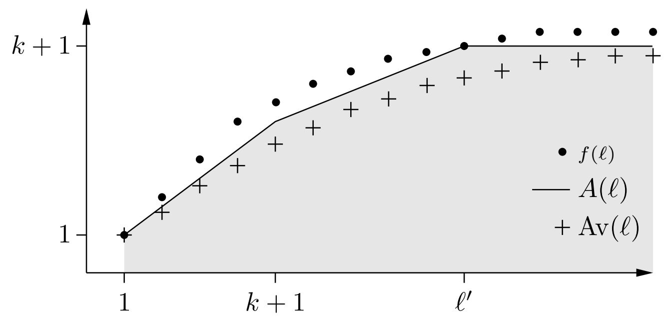

Bild 17.1: Für geeignet gewählte Parameter i und j liegt die stückweise lineare Funktion $A(\ell)$ zwischen $f(\ell)$ und $\mathrm{Av}(\ell)$

Parameter i und j Sequenzen $\sigma(i,j)$ erhält, die dieser Funktion f genügen. Dieser Aufgabe wenden wir uns zunächst zu und gehen dabei in zwei Schritten vor. Im ersten Schritt zeigen wir, dass in $\sigma(i,j)$ (für genügend groß gewählte Parameter i und j) die durchschnittliche Zahl angefragter Seiten $\mathrm{Av}(\ell)$ in Fenstern der Länge ℓ durch eine stückweise lineare Funktion $A(\ell)$ nach oben beschränkt ist (Lemma 17.1). Im zweiten Schritt zeigen wir dann, dass man für jedes gegebene f die Parameter i und j so einstellen kann, dass die sich ergebende Funktion $A(\ell)$ der Funktion f genügt (Lemma 17.2). Daraus ergibt sich unmittelbar, dass dann auch $\sigma(i,j)$ der Funktion f genügt. In Bild 17.1 ist diese Situation veranschaulicht.

Lemma 17.1. *Für fest gewählte Konstanten $i \geq k+2$, $j \geq k+1$ und $\ell' \geq k+2$ aus $\mathbb{N}$ sei die stückweise lineare Funktion $A : \mathbb{R}^{\geq 1} \to \mathbb{R}^{\geq 1}$ wie folgt definiert:*

$$A(\ell) \;:=\; \begin{cases} 1 + d_1(\ell - 1), & \textit{falls } 1 \leq \ell \leq k+1, \\ (1 + d_1 k) + d_2(\ell - (k+1)), & \textit{falls } k+1 \leq \ell \leq \ell', \\ k+1, & \textit{falls } \ell \geq \ell', \end{cases}$$

wobei die Steigungen der ersten beiden Abschnitte durch

$$d_1 \;:=\; 1 - \frac{j-k}{i(k+1)+j} \qquad \textit{und} \qquad d_2 \;:=\; \frac{k - d_1 k}{\ell' - (k+1)}$$

gegeben sind.

Für jede Wahl von $\ell' \geq k+2$ und jede Konstante $c > 0$ gibt es ein $i_0 \in \mathbb{N}$, sodass für alle $i \geq i_0$ und $j \geq c \cdot i$ für die Sequenz $\sigma(i,j)$ gilt:

$$\mathrm{Av}(\ell) \leq A(\ell) \qquad \textit{für alle } \ell \in \{1, \ldots, |\sigma|\}.$$

Beweis. Es gilt

$$
\begin{aligned}
N(\ell+1) - N(\ell) &= \sum_{t=1}^{|\sigma|-\ell} N_t(\ell+1) - \sum_{t=1}^{|\sigma|-\ell+1} N_t(\ell) \\
&= \sum_{t=1}^{|\sigma|-\ell} \big(N_t(\ell+1) - N_t(\ell)\big) - N_{|\sigma|-\ell+1}(\ell) \qquad (*)
\end{aligned}
$$

und

$$
\begin{aligned}
\mathrm{Av}(\ell+1) - \mathrm{Av}(\ell) &= \frac{N(\ell+1)}{|\sigma|-\ell} - \frac{N(\ell)}{|\sigma|-\ell+1} = \frac{N(\ell+1) - N(\ell)\frac{|\sigma|-\ell+1-1}{|\sigma|-\ell+1}}{|\sigma|-\ell} \\
&= \frac{N(\ell+1) - N(\ell) + \mathrm{Av}(\ell)}{|\sigma|-\ell}. \qquad (**)
\end{aligned}
$$

Für $1 \leq \ell \leq i(k+1)$ sei $r(\ell) := i(k+1)-\ell+1$ die Position der ersten Anfrage im letzten Fenster der Länge ℓ, welches noch vollständig im Präfix $(p_1, p_2, \ldots, p_{k+1})^i$ zu liegen kommt. Dieses Fenster $\sigma_{r(\ell)}(\ell)$ liegt also möglichst weit rechts im Präfix. Im Rest des Beweises unterscheiden wir drei Fälle für ℓ, die den drei linearen Abschnitten von $A(\ell)$ entsprechen.

Fall $1 \leq \ell \leq k+1$. Es gilt offensichtlich $\mathrm{Av}(1) = A(1) = 1$ und wir brauchen nur $\mathrm{Av}(\ell+1) - \mathrm{Av}(\ell) \leq d_1$ für $1 \leq \ell \leq k$ nachzuweisen. Dazu bestimmen wir $N(\ell+1) - N(\ell)$ und benutzen dann Gleichung $(**)$. Wir betrachten also nun Fensterlängen ℓ zwischen 1 und k. Offenbar können in keinem dieser Fenster alle $k+1$ Seiten angefragt werden. Für $1 \leq t \leq r(\ell)$ gilt $N_t(\ell+1) - N_t(\ell) = 1$, da in diesem Bereich $k+1$ Seiten zyklisch angefragt werden. Das gilt auch im Grenzfall für $t = r(\ell)$, da wegen $\ell \leq k$ keine Anfrage an p_1 in $\sigma_{r(\ell)}(\ell)$ enthalten ist, wohl aber in $\sigma_{r(\ell)}(\ell+1)$, da nun auch die erste Anfrage aus dem Suffix überdeckt wird. Für $t > r(\ell)$ gilt $N_t(\ell+1) = N_t(\ell)$, da bei der Verlängerung des Fensters nur noch weitere Anfragen an p_1 hinzukommen. Daher gilt $\sum_{t=1}^{|\sigma|-\ell} \big(N_t(\ell+1) - N_t(\ell)\big) = r(\ell)$. Das letzte Fenster der Länge ℓ ist vollständig im Suffix enthalten, weil schon in der Definition von $\sigma(i,j)$ festgelegt ist, dass das Suffix mindestens die Länge $k+1$ hat. Es gilt also $N_{|\sigma|-\ell+1}(\ell) = 1$. Mithilfe von Gleichung $(*)$ erhalten wir $N(\ell+1) - N(\ell) = r(\ell) - 1 = i(k+1) - \ell$. Aus Gleichung $(**)$ und $\mathrm{Av}(\ell) \leq k$ (da $\ell \leq k$) folgt jetzt

$$
\begin{aligned}
\mathrm{Av}(\ell+1) - \mathrm{Av}(\ell) &\leq \frac{i(k+1)-\ell+k}{i(k+1)-\ell+j} = 1 - \frac{j-k}{i(k+1)-\ell+j} \\
&< 1 - \frac{j-k}{i(k+1)+j} = d_1.
\end{aligned}
$$

Fall $k+1 \leq \ell \leq \ell'$. Für $\ell = k+1$ überlappt sich dieser Fall mit dem vorhergehenden. Der Funktionswert $A(k+1)$ ist für beide Fälle gleich und wir wissen aus der

Betrachtung des vorhergehenden Falles, dass $\mathrm{Av}(k+1) \leq A(k+1) = 1+d_1k$ gilt. Es genügt also $\mathrm{Av}(\ell+1)-\mathrm{Av}(\ell) \leq d_2$ für $k+1 \leq \ell \leq \ell'-1$ nachzuweisen. Da $\ell \geq k+1$, gilt für die Fenster im Präfix, d. h. für $1 \leq t \leq r(\ell)$, $N_t(\ell) = N_t(\ell+1) = k+1$. Für $t > r(\ell)$ gilt $N_t(\ell) = N_t(\ell+1)$, da eine Anfrage an p_1 bereits in $\sigma_t(\ell)$ enthalten ist. Also gilt $N_t(\ell+1) - N_t(\ell) = 0$ für alle t und aus Gleichung $(*)$ folgt $N(\ell+1) - N(\ell) = 0 - N_{|\sigma|-\ell+1}(\ell) \leq -1$. Weil ℓ' und k konstant sind, erhalten wir mithilfe von Gleichung $(**)$

$$\mathrm{Av}(\ell+1) - \mathrm{Av}(\ell) \;\leq\; \frac{-1+(k+1)}{i(k+1)+j-\ell} \;\leq\; \frac{k}{i(k+1)+j-\ell'} \;=\; O(1/i).$$

Um zu zeigen, dass es ein i_0 gibt, sodass für $i \geq i_0$ die Ungleichung $\mathrm{Av}(\ell+1) - \mathrm{Av}(\ell) \leq d_2$ erfüllt ist, genügt es nun, $d_2 = \omega(1/i)$ nachzuweisen. Aus der Definition von d_2 und $i \leq j/c$ ergibt sich

$$d_2 \;=\; \frac{k \cdot j - k^2}{(\ell' - (k+1))(i(k+1)+j)} \;\geq\; \frac{k \cdot j - k^2}{(\ell' - (k+1))((k+1)/c+1) \cdot j}.$$

Für gewisse positive Konstanten a und b gilt also $d_2 \geq (a \cdot j - a^2)/(b \cdot j) = \Omega(1)$.

Fall $\ell' \leq \ell$. Da die Sequenz nur $k+1$ Seiten anfragt, gilt $\mathrm{Av}(\ell) \leq k+1$ sogar für alle ℓ. □

Nun kommen wir zum zweiten Schritt. Im Wesentlichen schätzen wir hier die Suffixlänge ab, die ausreichend ist, damit eine Sequenz $\sigma(i,j)$ einer gegebenen Funktion genügen kann.

Lemma 17.2. *Für jede Funktion f, die konkav* ist und für die $f^{-1}(k+1)$ existiert, gibt es ein $i_0 \in \mathbb{N}$, sodass die Sequenzen $\sigma(i,j)$ der Funktion f genügen, sofern*

$$i \;\geq\; i_0 \quad \textit{und} \quad j \;\geq\; \frac{(k+1-f(k+1))(k+1)}{f(k+1)-1} \cdot i + \frac{k^2}{f(k+1)-1} =: u(i).$$

Beweis. Zuerst betrachten wir den trivialen Fall, dass f bis zur maximalen Seitenzahl $k+1$ sehr stark wächst, nämlich wenn $f(k+1) \geq k+1$. Dann gilt $f(\ell) \geq \ell$ für $1 \leq \ell \leq k+1$, da f konkav* ist. Wegen $\mathrm{Av}(\ell) \leq \ell$ gilt in diesem Fall $\mathrm{Av}(\ell) \leq f(\ell)$ für alle ℓ.

Andernfalls gilt $k+1-f(k+1) > 0$. Da j die Bedingung aus Lemma 17.1 erfüllt, ist dieses Lemma anwendbar und wir wählen $\ell' = f^{-1}(k+1) \geq k+2$. Für $i \geq i_0$, i_0 aus Lemma 17.1, gilt also $\mathrm{Av}(\ell) \leq A(\ell)$ für alle ℓ und insbesondere $\mathrm{Av}(1) = A(1) = f(1)$ und $A(\ell') = k+1$ (siehe Bild 17.1). Da $A(\ell)$ stückweise linear ist und $f(\ell)$ konkav ist, genügt es, die Stelle $k+1$ zu untersuchen. Wir zeigen durch Äquivalenzumformung, dass $A(k+1) \leq f(k+1)$ gilt:

$$\begin{aligned} A(k+1) \;\leq\; f(k+1) \quad &\Leftrightarrow \quad k+1-\frac{jk-k^2}{i(k+1)+j} \;\leq\; f(k+1) \\ &\Leftrightarrow \quad j \;\geq\; \frac{(k+1-f(k+1))(k+1)\cdot i + k^2}{f(k+1)-1}. \end{aligned}$$

Wir finden hier genau die Forderung an j, die im Lemma formuliert ist. Diese besagt im Wesentlichen, dass j für festes f und k mindestens linear in i wachsen muss. □

Wir sind nun in der Lage, Sequenzen $\sigma(i,j)$ zu konstruieren, die „unter" jeder beliebigen Funktion f bleiben, sofern diese konkav* ist. Wie schon im Maximumsmodell müssen wir für untere Schranken Anfragesequenzen konstruieren, die auf die verschiedenen Strategien zugeschnitten sind. Die Sequenzen vom Typ $\sigma(i,j)$ sind dazu noch zu unflexibel, da das Präfix immer gleich aussieht. Mit den $\sigma(i,j)$-Sequenzen haben wir jedoch schon den schlimmsten Fall untersucht, was die Zahl verschiedener Anfragen in Fenstern im Präfix betrifft. Wir gehen nun zu $S(i,j)$-Sequenzen über, die sich von den bisher untersuchten $\sigma(i,j)$-Sequenzen nur darin unterscheiden, dass im Präfix die Seiten $p_1,\ldots,p_{k+1}$ beliebig angefragt werden dürfen. Für festes i und festes j handelt es sich bei $S(i,j)$ also um eine Menge von Sequenzen gleicher Länge und gleichem Suffix, zu der auch $\sigma(i,j)$ gehört. Es ist nun vielleicht schon intuitiv klar, dass in einer $S(i,j)$-Sequenz die durchschnittliche Zahl angefragter Seiten in Fenstern einer festen Länge nicht größer sein kann als in $\sigma(i,j)$. Das ist die Aussage des folgenden Lemmas.

Lemma 17.3. *Für festes i und j sei $\sigma' \in S(i,j)$, $\sigma := \sigma(i,j)$ und n sei die gemeinsame Länge dieser Sequenzen. Für $1 \le \ell \le n$ gilt $\mathrm{Av}'(\ell) \le \mathrm{Av}(\ell)$.*

Beweis. Da σ' und σ gleich lang sind, genügt es zu zeigen, dass in jedem Fenster der Sequenz σ' nicht mehr Seiten angefragt werden als im entsprechenden Fenster von σ.

Falls $1 \le t \le (k+1)i - k$, ist $N_t(\ell) = \min\{\ell, k+1\}$. Das ist die größtmögliche Zahl in Fenstern der Länge ℓ und $N'_t(\ell)$ kann nicht größer als $N_t(\ell)$ sein. Die $k+1$ Anfragen $\sigma_{(k+1)i-k+1},\ldots,\sigma_{(k+1)i+1}$ gehen alle an verschiedene Seiten. Daher ist für Fenster, die an einer Position t im Bereich $(k+1)i - k + 1 \le t \le (k+1)i$ beginnen, die Zahl $N_t(\ell)$ nicht kleiner als $N'_t(\ell)$. Für $t \ge (k+1)i+1$ sind σ_t und σ'_t Anfragen an dieselbe Seite. Sich entsprechende Fenster, die in diesem Bereich beginnen, haben in beiden Sequenzen den gleichen Inhalt. □

Nach dieser längeren Vorbereitung fällt der Beweis der angekündigten unteren Schranke nun nicht mehr schwer.

Theorem 17.4. *Es sei $\mathcal{A}$ eine deterministische Online-Strategie. Sofern $f^{-1}(k+1)$ existiert, gilt für die Fehlerrate bez. f im Durchschnittsmodell*

$$F_{\mathcal{A}}(f) \ \ge\ \frac{f(k+1)-1}{k}.$$

Beweis. Für j monoton wachsend in i, d. h. $j = j(i)$, konstruieren wir eine Folge von Anfragesequenzen $\big(\sigma'(i,j)\big)_{i\ge i_0}$ wachsender Länge, sodass jedes Glied der Folge eine Sequenz $\sigma'(i,j) \in S(i,j)$ ist. Weil es zu jeder Zeit eine Seite aus $\{p_1,\ldots,p_{k+1}\}$

gibt, die nicht im Cache vorgehalten werden kann, können wir für jede deterministische Strategie $\mathcal{A}$ die Seiten im Präfix der Länge $i(k+1)$ von $\sigma'(i,j)$ so wählen, dass $\mathcal{A}$ auf jeder Anfrage im Präfix einen Seitenfehler hat. Nun wählen wir die Suffixlänge j gerade so groß, dass Lemma 17.2 anwendbar ist, nämlich $j := \lceil u(i) \rceil$ (siehe Lemma 17.2 für die Definition von $u(i)$). Aus Lemma 17.2 und Lemma 17.3 zusammen folgt nun, dass es ein i_0 gibt, sodass für alle $i \geq i_0$ alle Sequenzen $\sigma'(i,j)$ aus der obigen Folge f genügen. Die Fehlerrate von $\mathcal{A}$ für eine Sequenz aus der Folge berechnet sich nun wir folgt:

$$\begin{aligned} F_{\mathcal{A}}\bigl(\sigma'(n,m)\bigr) \;&\geq\; \frac{i(k+1)}{i(k+1)+u(i)+1} \\ &=\; \frac{1}{1+\frac{k+1-f(k+1)}{f(k+1)-1}+\Theta(1/i)} \;=\; \frac{f(k+1)-1}{k+\Theta(1/i)}. \end{aligned}$$

Der Term $\Theta(1/i)$ verschwindet mit wachsendem i und es folgt die behauptete untere Schranke für $F_{\mathcal{A}}(f)$. □

Im folgenden Abschnitt stellen wir der eben gezeigten unteren Schranke eine passende obere Schranke für LRU gegenüber.

17.2 LRU und Markierungsstrategien

Im Maximumsmodell haben wir vornehmlich ausgehend von Fenstern (Phasen) über die Zahl der darin angefragten Seiten argumentiert und konnten so Aussagen über die Länge des Fensters treffen. Für obere Schranken im Durchschnittsmodell kehren wir die Argumentationsrichtung um. Ausgehend von einer einzelnen Anfrage σ_t betrachten wir für eine feste Fensterlänge ℓ die Fenster, in denen σ_t vorkommt. So schätzen wir für jede Anfrage σ_t ihren Beitrag zu $N(\ell)$ ab. Dann können wir häufig folgendes Lemma anwenden.

Lemma 17.5. *Falls es für eine Strategie $\mathcal{A}$ Konstanten $a, b, c \geq 0$ und eine feste Fensterlänge ℓ gibt, sodass für alle Funktionen f, die konkav* sind, und alle Anfragesequenzen σ, die f genügen,*

$$N(\ell) \;\geq\; a \cdot \mathcal{A}(\sigma) + b \cdot |\sigma| - c$$

gilt, dann folgt im Durchschnittsmodell

$$F_{\mathcal{A}}(f) \;\leq\; \frac{f(\ell)-b}{a}.$$

Beweis. Weil σ der Funktion f genügt, gilt

$$f(\ell) \;\geq\; \mathrm{Av}(\ell) \;=\; \frac{N(\ell)}{|\sigma|-\ell+1} \;\geq\; \frac{a \cdot \mathcal{A}(\sigma) + b \cdot |\sigma| - c}{|\sigma|} \;=\; a \cdot F_{\mathcal{A}}(\sigma) + b - \frac{c}{|\sigma|}.$$

Umstellen ergibt

$$F_{\mathcal{A}}(\sigma) \;\leq\; \frac{f(\ell) - b}{a} + \frac{c}{a \cdot |\sigma|}.$$

Da der letzte Term mit wachsender Sequenzlänge verschwindet, folgt die behauptete obere Schranke für die Fehlerrate bez. f. □

Anfragen σ_t, die keinen Seitenfehler verursachen, bezeichnen wir im Folgenden als *kostenlose* Anfragen. Um zu zeigen, dass LRU optimal ist, zeigen wir, dass für LRU jeder Seitenfehler einen Beitrag von $k + 1$ zu $N(k + 1)$ leistet und jede kostenlose Anfrage mindestens den Beitrag 1 leistet.

Theorem 17.6. *Für die Fehlerrate von LRU bez. f gilt im Durchschnittsmodell*

$$F_{\mathrm{LRU}}(f) \;=\; \frac{f(k+1) - 1}{k}.$$

Beweis. Die behauptete untere Schranke ist in der Aussage von Theorem 17.4 enthalten. Wir beweisen im Folgenden die passende obere Schranke.

Es sei σ eine Anfragesequenz, die f genügt. Von der folgenden Betrachtung sind die ersten und die letzten k Anfragen der Sequenz ausgenommen. Nachdem eine Seite p angefragt wurde, sind die unmittelbar folgenden k Anfragen frei von Seitenfehlern durch Anfragen an p. Also sind auch die k Anfragen vor einem Seitenfehler auf p keine Anfragen an p. Somit ist jeder Seitenfehler auf einer Seite p in $k + 1$ Fenstern der Länge $k + 1$ enthalten, in denen es keine weiteren Seitenfehler auf p geben kann. Ferner beginnt mit jeder kostenlosen Anfrage an p ein Fenster der Länge $k+1$, das keine Seitenfehler für p enthält. Also trägt jeder Seitenfehler $k+1$ und jede kostenlose Anfrage mindestens 1 zu $N(k + 1)$ bei. Wenn wir die Beiträge der ersten und letzten k Anfragen pessimistisch mit $c := 2k(k + 1)$ nach oben abschätzen, erhalten wir

$$N(k+1) \;\geq\; (k+1) \cdot \mathrm{LRU}(\sigma) + (|\sigma| - \mathrm{LRU}(\sigma)) - c \;=\; k \cdot \mathrm{LRU}(\sigma) + |\sigma| - c.$$

Die behauptete obere Schranke folgt nun mit Lemma 17.5. □

Als Nächstes wollen wir wieder eine möglichst kleine obere Schranke nachweisen, die für alle Markierungsstrategien gilt. Diese ist etwas kleiner als $(4/3)(f(k)/k)$ und hängt auch von der Parität von k ab. Da wir unseren oberen Schranken immer möglichst genau passende untere Schranken gegenüberstellen wollen, können wir uns hier nicht mit der Näherung $(4/3)(f(k)/k)$ zufrieden geben, sondern müssen mit der exakten Schranke arbeiten.

Theorem 17.7. *Es sei $\mathcal{M}$ eine beliebige Markierungsstrategie. Für die Fehlerrate von $\mathcal{M}$ bez. f gilt im Durchschnittsmodell*

$$F_{\mathcal{M}}(f) \;\leq\; \begin{cases} \frac{4k}{3k+2} \cdot \frac{f(k)}{k}, & \text{falls } k \text{ gerade,} \\ \frac{4k}{3k+2-1/k} \cdot \frac{f(k)}{k}, & \text{falls } k \text{ ungerade.} \end{cases}$$

Beweis. Es sei σ eine Sequenz, die f genügt. Wegen Lemma 17.5 genügt es zu zeigen, dass für gerades und ungerades k der Wert von $N(k)$ mindestens

$$\frac{3k+2}{4}\mathcal{M}(\sigma) - O(1) \quad \text{bzw.} \quad \frac{3k+2-1/k}{4}\mathcal{M}(\sigma) - O(1)$$

ist.

Wir teilen σ von vorne beginnend in Phasen ein, sodass jede der Phasen Anfragen an genau k verschiedene Seiten enthält und die letzte Phase höchstens k verschiedene Seiten. Indem wir jede Phase nicht kürzer als notwendig wählen, erreichen wir, dass die k Seiten in jeder Phase verschieden von der Seite sind, die von der ersten Anfrage in der folgenden Phase angefragt wird. Sei m die Zahl der so entstehenden Phasen. Die k Seiten in Phase i bezeichnen wir mit $p_1^i, p_2^i, \ldots, p_k^i$, und zwar gemäß ihrem ersten Vorkommen in Phase i. Das bedeutet, die erste Seite in Phase i ist p_1^i, die nächste Seite verschieden von p_1^i bekommt den Namen p_2^i und so weiter. Die Phasen entsprechen den Runden der Markierungsstrategien, sodass jede Seite einer Phase höchstens einen Seitenfehler in ihrer Phase verursachen kann. Es sei t_i der Zeitpunkt (Index) der ersten Anfrage in Phase i. Die Anfrage σ_{t_i} fragt also p_1^i an.

Nun betrachten wir die Fenster

$$\sigma_{t_i - \lceil k/2 \rceil + 1}(k), \ldots, \sigma_{t_i}(k), \ldots\ldots\ldots\ldots, \sigma_{t_{i+1} - \lceil k/2 \rceil}(k),$$

die wir gedanklich der Phase i zuordnen, und definieren

$$N^i(k) := \sum_{t_i - \lceil k/2 \rceil + 1 \le t \le t_{i+1} - \lceil k/2 \rceil} N_t(k)$$

als den Beitrag von Phase i zu $N(k)$. Für $2 \le i \le m-2$ ist dies eine wohldefinierte Einteilung der Fenster aus allen Phasen i: Weil jede dieser Phasen und auch die Phase $m-1$ mindestens Länge k hat, liegen die Fenster aller dieser Phasen i vollständig in σ und jedes Fenster ist einer und nur einer Phase zugeordnet. Wir wollen zeigen, dass für $2 \le i \le m-2$ und k gerade die Fenster von Phase i mindestens $(3k^2+2k)/4$ zu $N^i(k)$ beitragen. Dann ist $N(k)$ wie gewünscht durch $((3k^2+2k)/4)(m-3) = ((3k+2)/4)(km - O(1)) = ((3k+2)/4)\mathcal{M}(\sigma) - O(1)$ nach unten beschränkt. Für k ungerade wollen wir zeigen, dass der Beitrag der Fenster von Phase i mindestens $(3k^2+2k-1)/4$ ist. Dann ist $N(k)$ mindestens $((3k+2-1/k)/4)\mathcal{M}(\sigma) - O(1)$.

Zunächst sei k gerade. Wie viele Fenster, die wir zu Phase i rechnen und Länge k haben, überdecken die erste Anfrage r an die Seite p_j^i in Phase i? Für $1 \le j \le k/2$ gibt es mindestens j Startzeitpunkte solcher Fenster, die in Phase i liegen, nämlich der Zeitpunkt der Anfrage r und die $j-1$ Zeitpunkte unmittelbar davor. Weitere $k/2-1$ solche Startpunkte liegen noch weiter davor, zum Teil oder sogar alle in Phase $i-1$. Die von p_j^i angefragte Seite trägt also in mindestens

$k/2-1+j$ Fenstern zu $N^i(k)$ bei. Für $k/2+1 \leq j \leq k$ steht die erste Anfrage an p_i^j spätestens an Position $t_{i+1}-(k-j+1)$. Das letzte Fenster, das noch zu Phase i gehört, steht per Definition an Position $t_{i+1}-k/2$. Es gibt also mindestens $k-((t_{i+1}-k+j-1)-(t_{i+1}-k/2)) = 3k/2-j+1$ Fenster von Phase i, in denen die von p_i^j angefragte Seite zu $N^i(k)$ beiträgt. Es folgt

$$N^i(k) \;\geq\; \sum_{1\leq j\leq k/2}\Big(\frac{k}{2}-1+j\Big)+\sum_{k/2+1\leq j\leq k}\Big(\frac{3k}{2}+1-j\Big) \;=\; \frac{3k^2}{4}.$$

Da wir bisher nur Beiträge von verschiedenen Seiten aus verschiedenen Anfragen in Phase i zu $N^i(k)$ gezählt haben, haben wir keinen Beitrag mehrfach eingerechnet. Die Seite p_j^{i+1}, die von der ersten Anfrage von Phase $i+1$ angefragt wird, ist nach Konstruktion der Phasen verschieden von allen in Phase i angefragten Seiten. Diese Anfrage wird jedoch von den letzten $k/2$ Fenstern von Phase i mit den Startpunkten $t_{i+1}-k+1,\ldots,t_{i+1}-k/2$ überdeckt. Daher leistet die erste Anfrage von Phase $i+1$ einen zusätzlichen Beitrag von $k/2$ zu $N^i(k)$. Damit ist der Fall „k gerade“ abgeschlossen.

Nun sei k ungerade. Mit analogen Überlegungen folgt für $1 \leq j \leq (k-1)/2$, dass es mindestens $j+(k-1)/2$ Fenster von Phase i gibt, in denen p_j^i zu $N^i(k)$ beiträgt. Für $(k+1)/2 \leq j \leq k$ gibt es mindestens $k-((t_{i+1}-k+j-1)-(t_{i+1}-(k+1)/2)) = (3/2)k+1/2-j$ solche Fenster. Daraus erhalten wir die erste Abschätzung

$$N^i(k) \;\geq\; \sum_{1\leq j\leq \frac{k-1}{2}}\Big(\frac{k-1}{2}+j\Big)+\sum_{\frac{k+1}{2}\leq j\leq k}\Big(\frac{3k+1}{2}-j\Big) \;=\; \frac{3k^2+1}{4}.$$

Diese untere Schranke können wir diesmal noch um $(k-1)/2$ erhöhen, weil die erste Anfrage in Phase $i+1$ von den letzten $(k-1)/2$ Fenstern aus Phase i, die an den Positionen $t_{i+1}-k+1,\ldots,t_{i+1}-(k+1)/2$ beginnen, überdeckt wird. □

Für die untere Schranke verwenden wir Sequenzen, in denen die Nummer der angefragten Seite „sägezahnartig“ wiederholt an- und wieder absteigt. Für eine Höhe h sei $\textsc{Auf}_h$ die Anfragesequenz $1,2,\ldots,h-1$ und $\textsc{Ab}_h$ sei die Anfragesequenz $h,h-1,\ldots,2$. Dann sei $\textsc{AufAb}_h := \textsc{Auf}_h\textsc{Ab}_h$ die Konkatenation dieser beiden Sequenzen. Für $m \in \mathbb{N}_0$ sei $\textsc{AufAb}_h^m := (\textsc{AufAb}_h)^m$ die m-fache Konkatenation von $\textsc{AufAb}_h$.

Im Folgenden betrachten wir Sequenzen σ von diesem Typ und kennzeichnen unsere Notation für $N(\ell)$ und $\mathrm{Av}(\ell)$ mit der Zahl der Wiederholungen m der Grundsequenz $\textsc{AufAb}_h$. So steht $N^m(\ell)$ und $\mathrm{Av}^m(\ell)$ für $N(\ell)$ bzw. $\mathrm{Av}(\ell)$, wobei $\sigma = \textsc{AufAb}_h^m$ als vereinbart gelten soll und h eine fest gewählte Konstante ist. Zunächst soll $\mathrm{Av}^m(\ell)$ hinreichend genau nach oben beschränkt werden.

Lemma 17.8. *Für $m \geq 2$ und $\sigma := \textsc{AufAb}_h^m$ gilt*

$$\mathrm{Av}^m(\ell) \;\leq\; f^m(\ell) \;:=\; \begin{cases} \ell - \frac{(\ell-1)^2}{4(h-1)} + \frac{\ell}{m-1}, & \text{falls } 1 \leq \ell \leq 2h-3 \text{ und } \ell \text{ ungerade,} \\ \ell - \frac{(\ell-1)^2-1}{4(h-1)} + \frac{\ell}{m-1}, & \text{falls } 2 \leq \ell \leq 2h-3 \text{ und } \ell \text{ gerade,} \\ h, & \text{falls } \ell \geq 2h-2. \end{cases}$$

Die Funktion $f^m(\ell)$ ist konkav.*

Für $m \to \infty$ verschwinden in Lemma 17.8 die Terme $\ell/(m-1)$, sodass $f^m(\ell)$ und $\mathrm{Av}^m(\ell)$ im Grenzwert tatsächlich zusammenfallen. Die Abweichungen für endliche Sequenzlängen ergeben sich daraus, dass in der folgenden Analyse der Beitrag der letzten Auf- und der letzten Ab-Sequenz von $\textsc{AufAb}_h^m$ zu $N^m(\ell)$ überschätzt wird. Der Einfluss der letzten $\textsc{AufAb}_h$-Sequenz schwindet jedoch mit wachsender Sequenzlänge.

Beweis zu Lemma 17.8. Für $\ell = 1$ gilt $\mathrm{Av}^m(\ell) = 1 \leq 1 + 1/(m-1) = f^m(1)$ und für $\ell \geq 2h-2$ kann keine Seite ausgelassen werden, sodass $\mathrm{Av}^m(\ell) = h = f^m(\ell)$ gilt. Wenn wir im Folgenden mit dem Beitrag einer Auf- oder Ab-Sequenz den Gesamtbeitrag der Fenster zu $N^m(\ell)$ bezeichnen, die ihren Startpunkt in der betrachteten Auf- oder Ab-Sequenz haben, dann gilt offensichtlich Folgendes. Für $2 \leq \ell \leq 2h-3$ leisten alle Auf- und Ab-Sequenzen bis auf die jeweils letzte Auf- und die letzte Ab-Sequenz den gleichen Beitrag. Den Beitrag der letzten Auf- und der letzten Ab-Sequenz können wir jeweils durch $(h-1)\ell$ nach oben abschätzen. Die Kernaufgabe besteht nun darin, den Beitrag einer Auf- oder Ab-Sequenz zu bestimmen, die nicht die letzte Auf- oder Ab-Sequenz in der Gesamtsequenz ist. Ohne Beschränkung der Allgemeinheit wählen wir die erste Auf-Sequenz aus und bestimmen deren Beitrag

$$N^{\textsc{Auf}}(\ell) := \sum_{1 \leq t \leq h-1} N_t(\ell)$$

für $2 \leq \ell \leq 2h-3$ (siehe Bild 17.2).

Für Fenster der Länge ℓ ist der Zeitpunkt $t^*(\ell) := h - \lfloor \ell/2 \rfloor$ der letzte Startzeitpunkt, sodass alle Seiten, die in dem Fenster angefragt werden, bereits zwischen $t^*(\ell)$ und h (jeweils einschließlich) angefragt werden. Für $1 \leq t \leq t^*(\ell)$ können wir uns also bei der Bestimmung von $N_t(\ell)$ auf die ersten h Anfragen der Auf-Sequenz konzentrieren. Für Startzeitpunkte $t^*(\ell) < t \leq h-1$ verhält es sich gerade so, dass in den Anfragen ab dem Zeitpunkt h alle in dem Fenster angefragten Seiten noch mindestens einmal angefragt werden. Hier konzentrieren wir uns also auf die Anfragen vom Zeitpunkt h an p_h bis zum Zeitpunkt $2h-1$ an p_1.

Wir betrachten nun zuerst den Fall $2 \leq \ell \leq h-1$. Für die Startzeitpunkte $t = 1, \ldots, h-\ell+1$ gilt offenbar $N_t(\ell) = \ell$, da alle Anfragen bis zum Zeitpunkt h

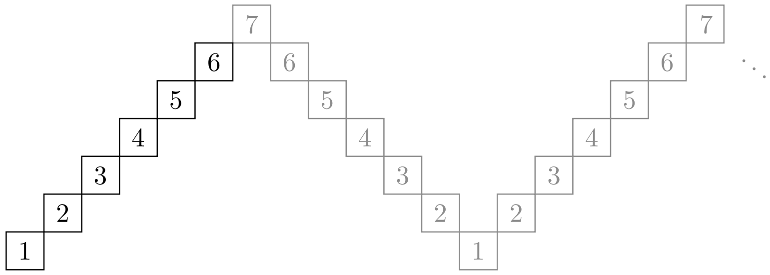

Bild 17.2: Die erste AUF_h-Sequenz in AUFAB_h^m für $h = 7$

an verschiedene Seiten gehen. Für $t = h - \ell + 2, \ldots, t^*(\ell)$ liegt das Ende des Fensters hinter der Position h. Mit unserer Vorüberlegung folgt, dass für diese Fenster $N_t(\ell) = h - t + 1$ gilt. Für die übrigen Startpositionen $t = t^*(\ell) + 1, \ldots, h - 1$ sind alle Anfragen ab dem Zeitpunkt h an verschiedene Seiten gerichtet. Es ergibt sich $N_t(\ell) = (t + \ell - 1) - h + 1 = t + \ell - h$. In der Summe erhalten wir

$$
\begin{aligned}
N^{\text{AUF}}(\ell) &= h\ell - \ell^2 + \ell + \big((\lfloor \ell/2 \rfloor + 1) + \cdots + (\ell - 1)\big) \\
&\quad + \big((\lceil \ell/2 \rceil + 1) + \cdots + (\ell - 1)\big) \\
&= \begin{cases} h\ell - \frac{(\ell+1)^2 - 1}{4}, & \text{falls } \ell \text{ gerade,} \\ h\ell - \frac{(\ell+1)^2}{4}, & \text{falls } \ell \text{ ungerade.} \end{cases}
\end{aligned}
$$

Im Fall $h \leq \ell \leq 2h - 3$ errechnen sich die $N_t(\ell)$-Werte wie folgt. Für Startzeitpunkte $t = 1, \ldots, t^*(\ell)$ reicht jedes Fenster mindestens bis zur Anfrage zum Zeitpunkt h, sodass hier $N_t(\ell) = h - t + 1$ gilt. Für $t = t^*(\ell) + 1, \ldots, 2h - \ell$ reichen die Fenster bis maximal zur Anfrage zum Zeitpunkt $2h - \ell - 1$ an die Seite p_1. Das Fenster $\sigma_t(\ell)$ reicht bis zur Anfrage $t+\ell-1$. Daher gilt $N_t(\ell) = (t+\ell-1)-h+1 = t+\ell-h$. Fenster, die zu den Zeitpunkten $t = 2h - \ell + 1, \ldots, h - 1$ beginnen, umfassen noch mindestens die Anfrage zum Zeitpunkt $2h - 1$, d. h., sie umfassen die gesamte AB-Sequenz und noch die Anfrage an p_1. Hier gilt also $N_t(\ell) = h$. In der Summe erhalten wir

$$
\begin{aligned}
N^{\text{AUF}}(\ell) &= h(\ell - 1) - h^2 + \big((\lfloor \ell/2 \rfloor + 1) + \cdots + h\big) + \big((\lceil \ell/2 \rceil + 1) + \cdots + h\big) \\
&= \begin{cases} h\ell - \frac{(\ell+1)^2 - 1}{4}, & \text{falls } \ell \text{ gerade,} \\ h\ell - \frac{(\ell+1)^2}{4}, & \text{falls } \ell \text{ ungerade.} \end{cases}
\end{aligned}
$$

Beide Fälle führen also zum selben $N^{\text{AUF}}(\ell)$-Wert, sodass wir für beide Fälle die Abschätzung

$$\begin{aligned}\mathrm{Av}^m(\ell) \;&\le\; \frac{2(m-1)N^{\text{AUF}}(\ell)+2(h-1)\ell}{2m(h-1)-\ell+1} \;\le\; \frac{2(m-1)N^{\text{AUF}}(\ell)+2(h-1)\ell}{2(m-1)(h-1)}\\ &=\; \frac{N^{\text{AUF}}(\ell)}{(h-1)}+\frac{\ell}{m-1} \;=\; \begin{cases}\ell-\frac{(\ell-1)^2}{4(h-1)}+\frac{\ell}{m-1}, & \text{falls } \ell \text{ ungerade,}\\ \ell-\frac{(\ell-1)^2-1}{4(h-1)}+\frac{\ell}{m-1}, & \text{falls } \ell \text{ gerade.}\end{cases}\end{aligned}$$

erhalten.

Es bleibt zu zeigen, dass $f^m(\ell)$ konkav* ist. Die erste Bedingung $f^m(1)=1$ ist offensichtlich erfüllt. Nun ist noch für alle $\ell \ge 2$ nachzuprüfen, dass

$$f^m(\ell)-f^m(\ell-1) \;\ge\; f^m(\ell+1)-f^m(\ell) \;\ge\; 0$$

eingehalten wird. Für $1 \le \ell \le 2h-3$ gilt unabhängig von der Parität von ℓ

$$f^m(\ell+1)-f^m(\ell) \;=\; 1-\frac{\lfloor \ell/2 \rfloor}{(h-1)}+\frac{1}{m-1}.$$

Offensichtlich ist die rechte Seite der letzten Gleichung positiv und monoton fallend in ℓ. Schließlich ist

$$f^m(2h-2)-f^m(2h-3) \;=\; \frac{1}{h-1}+\frac{1}{m-1} \;\ge\; f^m(2h-1)-f^m(2h-2) \;=\; 0$$

und $f^m(\ell+1)-f^m(\ell)=0$ für $\ell \ge 2h-2$. □

Im Maximumsmodell ist es uns an dieser Stelle gelungen, eine Markierungsstrategie auszumachen, die für alle Funktionen die größtmögliche Fehlerrate für Markierungsstrategien erreicht. Dies ist aus folgendem Grund hier nicht möglich. Eine genau passende untere Schranke zu Theorem 17.7, die für alle Funktionen f gilt, würde z. B. im Fall $f(k)=k$ eine Fehlerrate größer als 1 behaupten. Im Durchschnittsmodell kann unsere untere Schranke also nicht von derselben Qualität sein.

Zwar gelingt es auch hier, eine besonders schlechte Markierungsstrategie auszumachen, jedoch ist die Funktion f nicht wie sonst eine in weiten Grenzen wählbare Funktion, sondern eine ganz spezielle Funktion. Zudem ist diese Funktion von der Sequenzlänge und der Cachegröße k abhängig. Was wir erreichen, ist, für jede Cachegröße und Sequenzlänge (Letztere in gewissen Stufen) eine Anfragesequenz und eine Funktion zu konstruieren, sodass die erzielte Fehlerrate mit zunehmender Sequenzlänge (aber festem k) von unten gegen die obere Schranke aus Theorem 17.7 konvergiert.

Theorem 17.9. *Es gibt eine Markierungsvariante $\mathcal{M}^*$ von FWF, für die Folgendes gilt: Für jede Cachegröße k, jede initiale Cachebelegung und jede Sequenzlänge $n := 2km$ gibt es eine Funktion f, die konkav* ist, und eine Anfragesequenz σ der Länge n, die f genügt, sodass für die Fehlerrate von $\mathcal{M}^*$ gilt:*

$$F_{\mathcal{M}^*}(\sigma) \geq \begin{cases} \frac{4k}{3k+2+O(1/m)} \cdot \frac{f(k)}{k} - O(1/m), & \text{falls } k \text{ gerade,} \\ \frac{4k}{3k+2-1/k+O(1/m)} \cdot \frac{f(k)}{k} - O(1/m), & \text{falls } k \text{ ungerade.} \end{cases}$$

Beweis. Wir betrachten die Anfragesequenz $\sigma := \textsc{AufAb}_{k+1}^m$. Darin entsprechen die Auf- und Ab-Sequenzen je einer Runde einer Markierungsstrategie. Für $\mathcal{M}^*$ wählen wir die Strategie, die bei einem Seitenfehler diejenige unmarkierte Seite auswählt, die zuletzt eingelagert wurde. Diese Strategie können wir auch als Markierungsvariante von FWF auffassen (siehe Abschnitt 15.2). Ohne Beschränkung der Allgemeinheit dürfen wir annehmen, dass die Seiten $p_1, \ldots, p_k$ zu Beginn im Cache sind und in dieser Reihenfolge eingelagert wurden, d. h., p_k ist die „jüngste" Seite im Cache. Die Strategie $\mathcal{M}^*$ hat in der ersten Auf-Sequenz keine Seitenfehler. Die erste Anfrage der ersten Ab-Sequenz erzeugt den ersten Seitenfehler und bewirkt das Löschen aller Markierungen. Von nun an wird jeweils gerade die im nächsten Schritt angefragte Seite aus dem Cache entfernt. Die Strategie $\mathcal{M}^*$ hat also auf der Sequenz σ mit Länge $2km$ insgesamt $2km - k$ Seitenfehler.

Als Funktion wählen wir $f := f^m$ aus Lemma 17.8. Damit wissen wir auch, dass σ der gewählten Funktion f genügt. Die Fehlerrate von $\mathcal{M}^*$ auf σ ist

$$F_{\mathcal{M}^*}(\sigma) = \frac{2km - k}{2km} = 1 - \frac{2}{2m} = \frac{k}{f(k)} \cdot \frac{f(k)}{k} - O(1/m).$$

Wir brauchen nun nur noch den Bruch $k/f(k)$ nach unten zu beschränken. Die Höhe unserer Sequenz $\sigma = \textsc{AufAb}_{k+1}^m$ ist $h = k + 1$. Da wir $f = f^m$ an der Stelle $k \leq 2h - 3$ auswerten, sind wir auf jeden Fall in einem der beiden ersten Fälle der Definition von f^m in Lemma 17.8. Für k ungerade ist der Nenner des Bruchs

$$k - \frac{(k-1)^2}{4k} + \frac{k}{m-1} = \frac{1}{4}\big(3k + 2 - 1/k + O(1/m)\big),$$

für k gerade ist der Nenner

$$k - \frac{(k-1)^2 - 1}{4k} + \frac{k}{m-1} = \frac{1}{4}\big(3k + 2 + O(1/m)\big).$$

Damit ist der Bruch $k/f(k)$ so wie gewünscht beschränkt. □

Damit ist gezeigt, dass es für jede Cachegröße k eine Folge von Funktionen (f^m) gibt, sodass die einzelnen Funktionen konkav* sind und jeder Funktion eine Anfragesequenz genügt, die der Markierungsstrategie $\mathcal{M}^*$ eine Fehlerrate aufzwingt, die sich mit wachsendem m immer mehr der oberen Schranke für Markierungsstrategien annähert. Daher kann die obere Schranke aus Theorem 17.7 für beliebige Funktionen f, die konkav* sind, nicht kleiner sein.

17.3 FIFO

Für eine obere Schranke für FIFO gelingt es nicht unmittelbar, genügend viele Anfragen zu finden, denen man einen genügend großen Beitrag zu $N(\ell)$ zuweisen könnte. Das folgende Lemma schafft uns einen Zugang zu einer anderen Beweistechnik.

Lemma 17.10 (Einfügelemma). *Fügt man in eine Anfragesequenz an einer beliebigen Stelle eine beliebige Anfrage ein, dann wächst $N(\ell)$ für alle Fensterlängen ℓ um mindestens 1. Genauer: Die Anfragesequenz σ' sei aus der Anfragesequenz σ durch Einfügen einer Anfrage entstanden. Dann gilt für alle ℓ mit $1 \le \ell \le |\sigma|$: $N'(\ell) - N(\ell) \ge 1$.*

Beweis. Es sei σ' die Sequenz, die entsteht, wenn man die neue Anfrage zwischen σ_{i-1} und σ_i eingefügt, d. h. $p_{\sigma'_t} = p_{\sigma_t}$ für $1 \le t \le i-1$ und $p_{\sigma'_{t+1}} = p_{\sigma_t}$ für $i \le t \le |\sigma|$. Für $1 \le t \le i - \ell$ haben die Fenster $\sigma_t(\ell)$ und $\sigma'_t(\ell)$ gleichen Inhalt und ebenso die Fenster $\sigma_t(\ell)$ und $\sigma'_{t+1}(\ell)$ für $i \le t \le |\sigma| - \ell + 1$. Wir brauchen also lediglich die jeweils dazwischen liegenden Fenster

$$\sigma_{t_{\min}}(\ell), \ldots, \sigma_{t_{\max}}(\ell) \qquad \text{und} \qquad \sigma'_{t_{\min}}(\ell), \ldots, \sigma'_{t_{\max}+1}(\ell)$$

zu vergleichen (siehe Bild 17.3), wobei

$$t_{\min} := \max\{i - \ell + 1, 1\} \qquad \text{und} \qquad t_{\max} := \min\{i - 1, |\sigma| - \ell + 1\}.$$

Alle Fenster der Länge ℓ in σ, die an Positionen zwischen (und einschließlich) $t_{\min}$ und $t_{\max}$ beginnen, liegen vollständig in σ und für σ' gilt die entsprechende Aussage mit der oberen Grenze $t_{\max} + 1$. Um zu zeigen, dass $N'(\ell) \ge N(\ell) + 1$ gilt, genügt es also nachzuweisen, dass folgende Ungleichung gilt:

$$\sum_{t_{\min} \le t \le t_{\max}} \big(N'_t(\ell) - N_t(\ell)\big) + N'_{t_{\max}+1}(\ell) \ \ge\ 1. \tag{$*$}$$

Jedes Fenster $\sigma'_t(\ell)$, $t_{\min} \le t \le t_{\max}$, enthält die neue Anfrage σ'_i, aber die letzte Anfrage in $\sigma_t(\ell)$ fehlt in $\sigma'_t(\ell)$. Falls $N'_t(\ell) < N_t(\ell)$, gilt $N_t(\ell) - N'_t(\ell) = 1$, andernfalls ist die Differenz nicht positiv, nämlich 0 oder -1.

Für $N'_t(\ell) < N_t(\ell)$ ist es notwendig, dass die letzte Anfrage $\sigma_{t+\ell-1}$ in $\sigma_t(\ell)$ an eine Seite geht, die weder von einer Anfrage in $\sigma_t(\ell-1)$ noch von der neuen Anfrage σ'_i angefragt wird. Für die σ'-Sequenz folgt daraus, dass die von $\sigma'_{t+\ell}$ angefragte Seite in dem unmittelbar davor liegenden Fenster $\sigma'_t(\ell)$ nicht angefragt werden kann. Wir halten fest:

$\forall t,\ t_{\min} \le t \le t_{\max}\colon N'_t(\ell) < N_t(\ell) \Rightarrow p_{\sigma'_{t+\ell}}$ wird von $\sigma'_t, \ldots, \sigma'_{t+\ell-1}$ nicht angefragt.

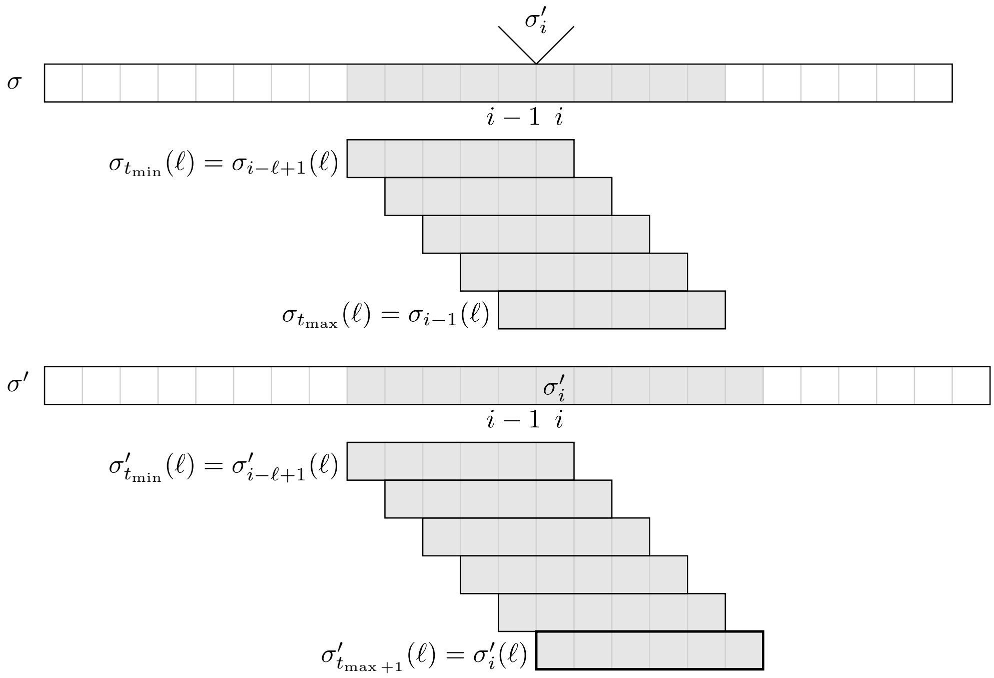

Bild 17.3: Es genügt, höchstens $\ell-1$ Fenster aus σ, die die Einfügestelle überdecken, mit ℓ Fenstern aus σ' zu vergleichen. Hier für den Fall dargestellt, dass keines dieser Fenster über die Ränder hinausragt.

Nun schätzen wir im Fenster $\sigma'_{t_{\max}+1}(\ell) = \sigma'_{t_{\max}+1}, \ldots, \sigma'_{t_{\max}+\ell}$ die Zahl der angefragten Seiten ab. Dabei ist die untere Fenstergrenze $t_{\max}+1$ offensichtlich höchstens i. Mithilfe obiger Implikation bezeugt jedes t zwischen (und einschließlich) $t_{\min}$ und $t_{\max}$ mit $N'_t(\ell) < N_t(\ell)$, dass die Seite an Position

$$(t+\ell) \in \{t_{\min}+\ell, \ldots, t_{\max}+\ell\} \subseteq \{i+1, \ldots, t_{\max}+\ell\}$$

im Fenster $\sigma'_{t_{\max}+1}(\ell)$ zuvor noch nicht angefragt wurde. Wenn wir diese Stellen t zählen und 1 für die Anfrage an Position i hinzuzählen, erhalten wir eine untere Schranke für $N'_{t_{\max}+1}(\ell)$. Mit obiger Überlegung über den Wert von $N_t(\ell) - N'_t(\ell)$ folgt nun:

$$N'_{t_{\max}+1}(\ell) \;\geq\; 1 + \sum_{t_{\min} \leq t \leq t_{\max}} \big(N_t(\ell) - N'_t(\ell)\big).$$

Durch Umstellen erhält man Ungleichung $(*)$. □

Mit dem Einfügelemma können wir Anfragesequenzen nun gedanklich anders auffassen. Wir stellen uns zunächst eine Sequenz vor, die nur aus den Anfragen mit Seitenfehlern besteht und berechnen den $N(\ell)$-Wert. Die dazwischenliegenden, kostenlosen Anfragen stellen wir uns nachträglich eingefügt vor. Die Änderung des

$N(\ell)$-Wertes können wir zwar nicht genau bestimmen, aber das Einfügelemma liefert uns zumindest eine Abschätzung. Im Falle von FIFO wird sich diese Abschätzung für eine kleinstmögliche obere Schranke als ausreichend erweisen.

Theorem 17.11. *Für die Fehlerrate von FIFO bez. f gilt im Durchschnittsmodell*

$$F_{\mathrm{FIFO}}(f) \;=\; \frac{f(k+1)-1}{k}.$$

Beweis. Die untere Schranke folgt aus Theorem 17.4. Wir zeigen im Folgenden die passende obere Schranke.

Es sei σ eine Anfragesequenz, die f genügt. Wir betrachten die Sequenz σ', die aus σ durch Streichen aller Anfragen entsteht, auf denen FIFO keinen Seitenfehler hat. Zwischen zwei Seitenfehlern auf derselben Seite müssen mindestens k andere Seiten Fehler verursachen. Daher hat in σ' jedes Fenster der Länge $k+1$ Anfragen an $k+1$ verschiedene Seiten:

$$N'(k+1) \;=\; (k+1)(|\sigma'|-k) \;\geq\; (k+1)\cdot \mathrm{FIFO}(\sigma) - k(k+1).$$

Mithilfe von Lemma 17.10 folgt nun

$$N(k+1) \;\geq\; N'(k+1) + \big(|\sigma| - \mathrm{FIFO}(\sigma)\big) \;\geq\; k\cdot \mathrm{FIFO}(\sigma) + |\sigma| - \Theta(1).$$

Durch Anwenden von Lemma 17.5 erhält man die behauptete obere Schranke für die Fehlerrate bez. f. □

Wir bemerken, dass – anders als im Maximumsmodell – FIFO im Durchschnittsmodell wie LRU eine optimale Strategie ist.

17.4 Die optimale Offline-Strategie LFD

Unser Ziel ist es, für LFD obere und untere Schranken zu zeigen, die etwa bei

$$\frac{4(M-k)}{4M-k}\cdot\frac{f(k+1)}{k+1}$$

liegen. Dabei bezeichnet M wieder die maximale Zahl der Seiten, die f anzufragen gestattet. Falls M kaum größer als k ist, dann liegt der Vorfaktor nahe bei null und wir erwarten auch eine kleine Fehlerrate für LFD. Falls M im Verhältnis zu k sehr groß ist, kann auch LFD bei nahezu jeder Anfrage an eine neue Seite einen Seitenfehler haben. Die Schranke liegt dann bei etwa $f(k+1)/(k+1)$, d. h., in Fenstern der Länge $k+1$ werden im Durchschnitt fast die maximal möglichen $f(k+1)$ Seitenfehler erreicht.

Für die obere Schranke teilen wir die gegebene Anfragesequenz in Phasen ein, die wie folgt definiert sind. Die erste Phase beginnt mit der ersten Anfrage der Sequenz.

Für $i \geq 2$ beginnt die i-te Phase mit der Anfrage, die eine Seite anfragt, welche in Phase $i-1$ von LFD aus dem Cache entfernt wurde. Jede Phase beginnt also mit einer Anfrage, die einen Seitenfehler verursacht. Es sei m die Zahl der Phasen, die bei der Einteilung einer betrachteten Sequenz entsteht, und s_i, $1 \leq i \leq m$, sei der Startzeitpunkt (Index), bei dem die i-te Phase beginnt. Da LFD niemals Seiten entfernt, die innerhalb der nächsten $k-1$ Anfragen nach einer Anfrage mit Seitenfehler angefragt werden, hat jede bis auf die letzte Phase mindestens Länge k.

Wie in den vorhergehenden Beweisen versuchen wir, die Beiträge von Seitenfehlern und kostenlosen Anfragen zu $N(k+1)$ abzuschätzen. Offenbar können bei LFD in einem Fenster der Länge $k+1$ keine zwei Anfragen enthalten sein, die Seitenfehler durch Anfragen an dieselbe Seite verursachen. Daher trägt jeder Seitenfehler $k+1$ zu $N(k+1)$ bei. Bei den kostenlosen Anfragen müssen wir sorgsam darauf achten, keinen Beitrag mehrfach zu verrechnen. Für eine kostenlose Anfrage an eine Seite p berücksichtigen wir nur solche Fenster, die keinen Seitenfehler durch eine andere Anfrage an p enthalten. Ferner muss die kostenlose Anfrage die erste Anfrage in dem Fenster sein, die p anfragt.

Lemma 17.12. *Wir betrachten eine Anfragesequenz σ und LFD. Für eine kostenlose Anfrage σ_t sei $\#W_t(k+1)$ die Anzahl der Fenster der Länge $k+1$, die*

– *die kostenlose Anfrage σ_t an die Seite $p := p_{\sigma_t}$ beinhalten,*

– *keine Anfrage an p beinhalten, die einen Seitenfehler verursacht, und*

– *keine Anfrage $\sigma_{t'}$ mit $t' < t$ beinhalten, die ebenfalls p anfragt.*

In jeder Phase i, $2 \leq i \leq m-2$, gibt es mindestens $k-1$ kostenlose Anfragen $\sigma_{t_1}, \ldots, \sigma_{t_{k-1}}$ an ebenso viele verschiedene Seiten, sodass in der Summe für Phase i gilt

$$\#W^i(k+1) := \sum_{j=1}^{k-1} \#W_{t_j}(k+1) \ \geq \ \begin{cases} \frac{3}{4}k^2 - \frac{3}{4}, & \text{falls } k \text{ ungerade,} \\ \frac{3}{4}k^2 - 1, & \text{falls } k \text{ gerade.} \end{cases}$$

Beweis. Es sei p die Seite, die von der ersten Anfrage in Phase $i+1$ angefragt wird. Es sei σ_t die Anfrage in Phase i, bei deren Bearbeitung p von LFD aus dem Cache entfernt wird. Ohne Beschränkung der Allgemeinheit seien $p_1, \ldots, p_{k-1}$ die Seiten, die zu diesem Zeitpunkt t ebenfalls im Cache sind, jedoch von LFD nicht gewählt werden. Da LFD p wählt, werden alle Seiten $p_1, \ldots, p_{k-1}$ noch mindestens einmal vor dem Zeitpunkt s_{i+1} angefragt. Dies geschehe für die Seite p_j zum ersten Mal zum Zeitpunkt t_j, $t < t_j < s_{i+1}$. Wir behaupten, dass die Anfragen $\sigma_{t_1}, \ldots, \sigma_{t_{k-1}}$ kostenlos sind. Angenommen, es gäbe eine Anfrage $\sigma_{t'}$ zwischen σ_t und $\sigma_{s_{i+1}}$, die eine Seite $p' \in \{p_1, \ldots, p_{k-1}\}$ anfragt und einen Seitenfehler verursacht. Dann müsste diese Seite p' zwischen σ_t und $\sigma_{t'}$ aus dem Cache entfernt werden. Doch dann würde die Phase $i+1$ bereits mit der Anfrage $\sigma_{t'}$ beginnen. Wir haben also gesehen, dass

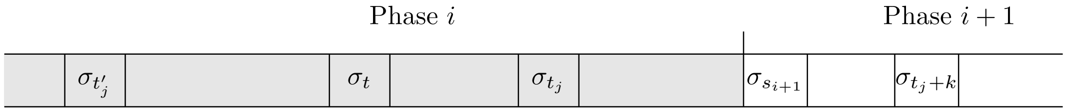

Bild 17.4: Zum Zeitpunkt t entfernt LFD die Seite p aus dem Cache. Die Seite p_j werde zum Zeitpunkt t_j zum ersten Mal nach und zum Zeitpunkt t'_j zum letzten Mal vor dem Zeitpunkt t angefragt. Die Anfragen σ_t bis σ_{t_j+k} sind frei von Seitenfehlern durch Anfragen an die Seite p_j.

es zwischen σ_t und $\sigma_{s_{i+1}}$ keine Anfragen an $p_1, \ldots, p_{k-1}$ geben kann, die Seitenfehler verursachen. Damit haben wir in Phase i die gesuchten $k-1$ kostenlosen Anfragen $\sigma_{t_1}, \ldots, \sigma_{t_{k-1}}$ an $k-1$ verschiedene Seiten $p_1, \ldots, p_{k-1}$ gefunden.

Wir betrachten nun so eine kostenlose Anfrage σ_{t_j} und den zugehörigen Wert $\#W_{t_j}$. Ein Fenster der Länge $k+1$, das σ_{t_j} überdeckt, reicht maximal bis zur Anfrage σ_{t_j+k}. Darin mag es rechts von σ_{t_j} weitere Anfragen an p_j geben, diese sind aber mit Sicherheit frei von Seitenfehlern, selbst wenn diese Anfragen rechts von $\sigma_{s_{i+1}}$ liegen sollten. Der Grund ist folgender. Bei der Bearbeitung von σ_{t_j} bleibt p_j im Cache, da die Anfrage kostenfrei ist. Frühestens bei Bearbeitung von σ_{t_j+1} kann p_j aus dem Cache fallen. Diese Entscheidung würde LFD aber nur treffen können, wenn es keine weitere Anfrage an p_j in den darauf folgenden $k-1$ Anfragen gibt. Daher sind die Anfragen $\sigma_t, \ldots, \sigma_{t_j+k}$ frei von Seitenfehlern durch Anfragen an p_j und von σ_t bis σ_{t_j-1} wird p_j gar nicht angefragt.

Ob ein Fenster, das σ_{t_j} überdeckt, die Kriterien für $\#W_{t_j}$ erfüllt, entscheidet sich also allein daran, ob es noch vor σ_{t_j} eine Anfrage an p_j enthält. Dazu sei t'_j der letzte Zeitpunkt vor t, zu dem die Seite p_j angefragt wird. Falls p_j zuvor nicht angefragt wurde, sondern von Beginn an im Cache war, dann sei $t'_j = 0$. Für den Wert $\#W_{t_j}$ sind also nur die Fenster interessant, die an den $t_j - t'_j$ Positionen (Zeitpunkten) $t'_j + 1, \ldots, t_j$ beginnen, denn für diese ist σ_{t_j} die erste Anfrage an p_{t_j} im Fenster, falls sie überhaupt σ_{t_j} überdecken. Also gilt

$$\sum_{j=1}^{k-1} \#W_{t_j}(k+1) \;=\; \sum_{j=1}^{k-1} \min\{k+1, t_j - t'_j\}.$$

Es sei $I \subseteq \{1, \ldots, k-1\}$ die Menge der Indizes j, für die $t_j - t'_j \leq k+1$ gilt. Dann können wir die letzte Summe wie folgt abschätzen:

$$\begin{aligned}
\sum_{j=1}^{k-1} \#W_{t_j}(k+1) \;&=\; (k-1-|I|)(k+1) + \sum_{j \in I} t_j - \sum_{j \in I} t'_j \\
&\geq\; (k-1-|I|)(k+1) + \sum_{1 \leq j \leq |I|} (t+j) - \sum_{1 \leq j \leq |I|} (t-j)
\end{aligned}$$

$$\begin{aligned} &= (k-1-|I|)(k+1) + 2 \sum_{1 \le j \le |I|} j \\ &= |I|^2 - k|I| + (k^2-1). \end{aligned}$$

Für ein gerades k wird diese untere Schranke für $|I| = k/2$ minimal. Falls k ungerade ist, dann wird das Minimum für $|I| = (k-1)/2$ angenommen. Daraus folgt die untere Schranke für die Summe der $\#W_{t_j}(k+1)$, die im Lemma angegeben ist. □

Nun sind wir gerüstet, die folgende obere Schranke für LFD zu zeigen. Natürlich ist auch hier der Fall $k \ge M$ uninteressant.

Theorem 17.13. *Falls* $|\operatorname{Bild}(f)|$ *endlich ist, sei* M *die größte natürliche Zahl in* $\operatorname{Bild}(f)$. *Für die Fehlerrate von LFD bez.* f *gilt dann im Durchschnittsmodell für* $k < M$

$$F_{\mathrm{LFD}}(f) \le \begin{cases} \frac{4(M-k)}{4M-k-3} \cdot \frac{f(k+1)}{k+1}, & \text{falls } k \text{ ungerade,} \\ \frac{4(M-k)}{4M-k-3-\frac{1}{k+1}} \cdot \frac{f(k+1)}{k+1}, & \text{falls } k \text{ gerade.} \end{cases}$$

Andernfalls gilt $F_{\mathrm{LFD}}(f) \le f(k+1)/(k+1)$.

Beweis. Es sei σ eine beliebige Anfragesequenz, die f genügt. Gemäß der weiter oben beschriebenen Einteilung in Phasen 1 bis m betrachten wir zunächst nur die Phasen 2 bis $m-2$. Durch die Mindestlänge k jeder Phase ist sichergestellt, dass alle Fenster der Länge $k+1$, die eine Anfrage aus diesen Phasen überdecken, vollständig in σ liegen, d. h. nicht über die Ränder von σ hinausragen. Zwischen zwei Anfragen an dieselbe Seite, die jeweils Seitenfehler verursachen, müssen k Anfragen an andere Seiten liegen. Pro Seite überdeckt also jedes Fenster der Länge $k+1$ höchstens eine Anfrage, die diese Seite anfragt und einen Seitenfehler verursacht. Damit trägt jeder Seitenfehler $k+1$ zu $N(k+1)$ bei. Lemma 17.12 garantiert uns nun, dass es in jeder Phase i mindestens $k-1$ kostenlose Anfragen gibt, die insgesamt mindestens $W^i(k+1)$ zu $N(k+1)$ beitragen.

Für jede Phase i, $1 \le i \le m$, sei F^i die Zahl der Seitenfehler in dieser Phase. Offenbar kann es maximal einen Seitenfehler pro Seite geben, aber die Seiten, die zu Beginn einer Phase im Cache sind, können keinen Seitenfehler in der Phase verursachen, da sie dazu in der Phase aus dem Cache entfernt werden müssten. Das widerspricht der Phaseneinteilung. Wir betrachten den Fall, dass M existiert. Dann enthält jede Phase höchstens $M-k$ Anfragen, die Seitenfehler verursachen. Es sei $N^i(k+1)$, $2 \le i \le m-2$, der Gesamtbeitrag aller Anfragen in Phase i zu $N(k+1)$. Dann ist

$$\frac{N^i(k+1)}{F^i} \ge \frac{(k+1)F^i + W^i(k+1)}{F^i} \ge \frac{(k+1)(M-k) + W^i(k+1)}{M-k}.$$

Somit erhalten wir

$$\begin{aligned}\mathrm{LFD}(\sigma) &= \sum_{i=2}^{m-2} \frac{F^i}{N^i(k+1)} \cdot N^i(k+1) + (F^1 + F^{m-1} + F^m) \\ &\le \frac{M-k}{(k+1)(M-k) + W^i(k+1)} \cdot \sum_{i=2}^{m-2} N^i(k+1) + 3(M-k) \\ &\le \frac{M-k}{(k+1)(M-k) + W^i(k+1)} \cdot N(k+1) + 3(M-k)\end{aligned}$$

und können nun die Fehlerrate wie folgt abschätzen:

$$\begin{aligned}F_{\mathrm{LFD}}(\sigma) &\le \frac{M-k}{(k+1)(M-k) + W^i(k+1)} \mathrm{Av}(k+1) + \frac{3(M-k)}{|\sigma|} \\ &\le \frac{M-k}{(k+1)(M-k) + W^i(k+1)} f(k+1) + \frac{3(M-k)}{|\sigma|}.\end{aligned}$$

Mithilfe der Abschätzung für $W^i(k+1)$ aus Lemma 17.12 und $(3/4)k^2 - (3/4) = (3/4)(k-1)(k+1)$ gelangen wir schließlich zu

$$F_{\mathrm{LFD}}(\sigma) \le \begin{cases} \frac{4(M-k)}{4M-k-3} \cdot \frac{f(k+1)}{k+1} + \frac{3(M-k)}{|\sigma|}, & \text{falls } k \text{ ungerade,} \\ \frac{4(M-k)}{4M-k-3-\frac{1}{k+1}} \cdot \frac{f(k+1)}{k+1} + \frac{3(M-k)}{|\sigma|}, & \text{falls } k \text{ gerade.} \end{cases}$$

Für jede gewählte Funktion f mit endlichem Bildbereich ist M eine Konstante. Also verschwindet der Summand $3(M-k)/|\sigma|$ mit zunehmender Sequenzlänge.

Falls Bild(f) nicht endlich ist, argumentieren wir wie folgt. Die Anzahl der Anfragen, die Seitenfehler verursachen, ist LFD(σ). Jede dieser Anfragen, bis auf eventuell die ersten k und die letzten k, trägt $k+1$ zu $N(k+1)$ bei. Also gilt dann $N(k+1) \ge (k+1)(\mathrm{LFD}(\sigma) - 2k)$ und somit $\mathrm{LFD}(\sigma) \le N(k+1)/(k+1) + 2k$. Wir erhalten dann

$$F_{\mathrm{LFD}}(\sigma) \le \frac{N(k+1)}{(k+1)\cdot|\sigma|} + \frac{2k}{|\sigma|} \le \frac{\mathrm{Av}(k+1)}{k+1} + \frac{2k}{|\sigma|} \le \frac{f(k+1)}{k+1} + \frac{2k}{|\sigma|}.$$

Daraus folgt das behauptete Resultat. □

Die zugehörige untere Schranke ist für LFD von der gleichen Art wie schon für Markierungsstrategien (vgl. Theorem 17.9).

Theorem 17.14. *Für jede Cachegröße k, jede initiale Cachebelegung und jede Wahl der Zahlen $N > k$ sowie $n := 2km$ gibt es eine Funktion f, die konkav* ist, und eine Anfragesequenz σ der Länge n, die f genügt, sodass für die Fehlerrate von LFD gilt:*

$$F_{\mathrm{LFD}}(f) \ge \begin{cases} \frac{4(N-k)}{4N-k-3+O(1/m)} \cdot \frac{f(k+1)}{k+1}, & \textit{falls } k \textit{ ungerade,} \\ \frac{4(N-k)}{4N-k-3-\frac{1}{k+1}+O(1/m)} \cdot \frac{f(k+1)}{k+1}, & \textit{falls } k \textit{ gerade.} \end{cases}$$

Beweis. Wir wählen die Sequenz $\textsc{AufAb}_N^m$. Diese hat die gewünschte Länge. Zuerst überlegen wir, bei welchen Anfragen LFD Seitenfehler hat. Ohne Beschränkung der Allgemeinheit seien zu Beginn die Seiten $p_2, \ldots, p_{k+1}$ im Cache. Dann hat LFD bei der ersten Anfrage der $\textsc{Auf}$-Sequenz an p_1 einen Seitenfehler, bei dem die Seite p_{k+1} aus dem Cache entfernt wird. Daher verursachen die letzten $N-k-1$ Anfragen der $\textsc{Auf}$-Sequenz an $p_{k+1}, \ldots, p_{N-1}$ ebenfalls Seitenfehler. Nachdem die letzte Anfrage der $\textsc{Auf}$-Sequenz bearbeitet ist, hat es insgesamt $N-k$ Seitenfehler gegeben und die Seiten $p_{N-k}, \ldots, p_{N-1}$ befinden sich im Cache. In der folgenden $\textsc{Ab}$-Sequenz verursacht ebenfalls die erste Anfrage, an p_M, einen Seitenfehler, bei dem diesmal p_{N-k} entfernt wird. Daher verursachen wieder die letzten $N-k-1$ Anfragen der $\textsc{Ab}$-Sequenz, an $p_{N-k}, \ldots, p_2$, Seitenfehler. Nachdem die letzte Anfrage der $\textsc{Ab}$-Sequenz bearbeitet ist, sind die Seiten $p_2, \ldots, p_{k+1}$ im Cache. Mit Beginn der nächsten $\textsc{Auf}$-Sequenz ist also genau die Anfangsbelegung des Caches wiederhergestellt. Da in jeder $\textsc{Auf}$- und in jeder $\textsc{Ab}$-Sequenz dieselbe Zahl an Seitenfehlern vorkommt, ist die Fehlerrate $(N-k)/(N-1)$. Diese müssen wir nur noch umformulieren.

Wir betrachten im Folgenden den Fall, dass k gerade ist. Unsere Sequenz genügt der Funktion $f^m(\ell)$ aus Lemma 17.8 für $h = N$. Da $k+1$ ungerade ist, erhalten wir

$$
\begin{aligned}
f^m(k+1) &= (k+1) - \frac{k^2}{4(N-1)} + \frac{k+1}{m-1} \\
&= \frac{4(N-1)(k+1) - \big((k-1)(k+1)+1\big) + 4(N-1)(k+1)/(m-1)}{4(N-1)} \\
&= \frac{\big(4N-k-3-1/(k+1)+O(1/m)\big)}{4(N-1)}(k+1).
\end{aligned}
$$

Im Fall, dass k ungerade ist, fällt im letzten Bruch im Zähler der Term $-1/(k-1)$ weg. Nun können wir die Fehlerrate folgendermaßen abschätzen:

$$
\begin{aligned}
F_{\mathrm{LFD}}(\textsc{AufAb}_N^m) &\geq \frac{N-k}{N-1} \cdot \frac{f^m(k+1)}{f^m(k+1)} \\
&= \begin{cases} \frac{4(N-k)}{4N-k-3+O(1/m)} \cdot \frac{f(k+1)}{k+1}, & \text{falls } k \text{ ungerade,} \\ \frac{4(N-k)}{4N-k-3-\frac{1}{k+1}+O(1/m)} \cdot \frac{f(k+1)}{k+1}, & \text{falls } k \text{ gerade.} \end{cases}
\end{aligned}
$$

□

Bei Wahl von $N = M$ wird klar, dass man die obere Schranke aus Theorem 17.13 nur unwesentlich kleiner machen kann; lediglich die O-Terme gibt es dort nicht. Auch für Funktionen, für die M nicht existiert, nähert sich die untere Schranke mit wachsendem N beliebig nahe der dann geltenden oberen Schranke $f(k+1)/(k+1)$ an.

18 Die theoretischen Ergebnisse im Überblick

In diesem Kapitel sind die theoretischen Ergebnisse aus Kapitel 16 und Kapitel 17 gesammelt dargestellt.

Die Fehlerrate bez. f haben wir in beiden Modellen für einige gängige deterministische Seitenwechselstrategien exakt oder beinahe exakt bestimmt. Tabelle 18.1 zeigt eine Zusammenschau der erzielten Ergebnisse.

LRU und FIFO. Im Maximumsmodell ist $(k-1)/(f^{-1}(k+1)-2)$ eine untere Schranke, sodass keine Online-Strategie eine kleinere Fehlerrate bez. f haben kann. LRU hat höchstens diese Fehlerrate, ist also eine optimale Online-Strategie im Maximumsmodell. Für FIFO können wir nur eine obere Schranke von $k/(f^{-1}(k+1)-1)$ und eine untere von $(k-1/k)/(f^{-1}(k+1)-2)$ angeben. Für den Fall, dass $f^{-1}(4)-f^{-1}(3) > f^{-1}(3)-f^{-1}(2)$ gilt, haben wir als untere Schranke $(k-1/k)/(f^{-1}(k+1)-1)$ bewiesen. Diese Schranke ist größer als die Fehlerrate von LRU, falls noch $f^{-1}(k+1) > k+2$ gilt. Das bedeutet, FIFO schneidet in diesem Fall etwas schlechter als LRU ab und ist daher keine optimale Online-Strategie.

Im Maximumsmodell finden wir also die vermutete Überlegenheit von LRU gegenüber FIFO bestätigt. Im Durchschnittsmodell hingegen sind FIFO und LRU optimale Online-Strategien. Beide erreichen genau die bestmögliche Fehlerrate $(f(k+1)-1)/k$ für Online-Strategien. Genau wie bei der kompetitiven Analyse können die Resultate im Durchschnittsmodell nicht entscheiden, ob eine dieser beiden Strategien besser ist als die andere.

Markierungsstrategien. In beiden Modellen sind Markierungsstrategien i. Allg. nicht so gut wie optimale Online-Strategien. Im Maximumsmodell kann die Fehlerrate aller Markierungsstrategien nur durch $k/(f^{-1}(k+1)-1)$ nach oben beschränkt werden; für $f^{-1}(k+1) > k+1$ wird dieser Ausdruck größer als die Fehlerrate von LRU. Wir haben gezeigt, dass es eine Markierungsstrategie gibt, nämlich eine Markierungsvariante von FWF, die auch keine kleinere Fehlerrate erreicht.

Im Durchschnittsmodell ist für gerades k die Fehlerrate aller Markierungsstrategien durch $4k/(3k+2) \cdot f(k)/k$ beschränkt, für ungerades k bekommt der Nenner des ersten Bruchs noch zusätzlich den additiven Term $-1/k$. Hier konnten wir ebenfalls anhand einer Markierungsvariante von FWF zeigen, dass diese obere Schranke im Wesentlichen nicht verbessert werden kann. Obwohl Markierungsstrategien i. Allg.

	Maximumsmodell	Durchschnittsmodell
Online	$\geq \frac{k-1}{f^{-1}(k+1)-2}$	$\geq \frac{f(k+1)-1}{k}$
FIFO	$\geq \frac{k-1/k}{f^{-1}(k+1)-1}, \quad \leq \frac{k}{f^{-1}(k+1)-1}$	$= \frac{f(k+1)-1}{k}$
Mark.-Str.	$\leq \frac{k}{f^{-1}(k+1)-1}$	$\lesssim \frac{4}{3}\frac{f(k)}{k}$
FWF	$= \frac{k}{f^{-1}(k+1)-1}$	$\approx \frac{4}{3}\frac{f(k)}{k}$
LRU	$= \frac{k-1}{f^{-1}(k+1)-2}$	$= \frac{f(k+1)-1}{k}$
LFD	$\geq \max\limits_{\substack{m\in\mathbb{N}\\ k+m\leq M}} \left\{\frac{m}{f^{-1}(k+m+1)-2}\right\}, \leq 2 \max\limits_{\substack{1\leq m\leq k\\ k+m\leq M}} \left\{\frac{m+1}{f^{-1}(k+m)}\right\}$	$\approx \frac{4(M-k)}{4M-k}\frac{f(k+1)}{k+1}$

Tabelle 18.1: Zuoberst untere Schranken für die Fehlerrate bez. f für alle Online-Strategien im Maximums- und Durchschnittsmodell. Darunter obere und untere Schranken für die untersuchten Strategien. Die Zeichen „$\lesssim$“ und „$\approx$“ weisen darauf hin, dass zur besseren Übersicht nicht die korrekten, sondern vereinfachte Ausdrücke eingetragen sind. Für LFD ist nur der Fall berücksichtigt, dass M existiert.

nicht optimal sind, so beinhalten sie doch die Strategie LRU, die sich in beiden Modellen als optimal erwiesen hat.

LFD. Für die Offlinestrategie LFD kann die Fehlerrate von der größten natürlichen Zahl M abhängen, die in Bild(f) enthalten ist. Die Fehlerrate von LFD ist im Maximumsmodell mindestens das Maximum aller $m/(f^{-1}(k+m+1)-2)$, wobei das Maximum über alle $m \in \mathbb{N}$ gebildet wird. Falls M existiert, brauchen (und können) nur diejenigen m betrachtet zu werden, für die $k+m \leq M$ gilt. Für die obere Schranke, das Maximum aller $2(m+1)/(f^{-1}(k+m))$, gilt dasselbe, jedoch ist m zusätzlich durch k beschränkt. Im Wesentlichen ist diese obere Schranke um einen Faktor 2 größer als die untere Schranke.

Im Durchschnittsmodell ist die Fehlerrate von Markierungsstrategien durch $(4(M-k)/(4M-k-3)) \cdot (f(k+1)/(k+1))$ nach oben beschränkt, für gerades k bekommt der Nenner des ersten Bruchs noch zusätzlich den additiven Term $-1/(k+1)$. Falls M nicht existiert, ist die Fehlerrate durch $f(k+1)/(k+1)$ beschränkt. Auch hier haben wir gezeigt, dass diese Schranken nicht wesentlich verbessert werden können.

19 Vergleich mit gemessenen Fehlerraten

Das Maximums- und das Durchschnittsmodell sind dadurch motiviert worden, dass die maximale bzw. durchschnittliche Zahl angefragter Seiten in Sequenzen gut durch Funktionen beschrieben werden kann, die konkav** bzw. konkav* sind. Diese Annahme soll hier nicht rigoros belegt werden. Vielmehr soll eine kleine Zahl von Beispielen untersucht werden, und zwar Anfragesequenzen, die aus echten Prozessen gewonnen wurden. Anhand dieser Beispielsequenzen werden die oberen Schranken aus der vorhergehenden theoretischen Untersuchung mit den tatsächlichen Fehlerraten verglichen, die FIFO, LRU und LFD auf diesen Sequenzen im Modell des Seitenwechselproblems erreichen, sowie mit den Schranken, die sich aus der kompetitiven Analyse ergeben. Aufgrund der geringen Zahl an Sequenzen kann hier lediglich ein erster Eindruck entstehen. Der Frage, ob die Funktionen geeignet sind, das Verhalten typischer Prozesse zu repräsentieren, soll hier nicht nachgegangen werden.

Anfragesequenzen an einem echten Prozessor aufzuzeichnen, erfordert eingehende Kenntnis der Hardware und ist nicht leicht zu bewerkstelligen. Solche Anfragesequenzen werden z. B. beim Entwurf eines Cachesystems benutzt, um mithilfe von Simulationen die Güte des Entwurfs prognostizieren und einschätzen zu können. Wir greifen hier auf frei zur Verfügung stehende, sogenannte (Processor-)Traces zurück. Solche können von der Homepage des Performance and Architecture Research Lab der New Mexico State University (http://tracebase.nmsu.edu/tracebase.html) heruntergeladen werden.

Aus den zur Verfügung stehenden Traces untersuchen wir exemplarisch Traces von VAX- und SPARC-Rechnern. Diese Prozessorarchitekturen sind geeignet, weil sie mit nur *einem* virtuellen Adressraum für jeden Prozess arbeiten, d. h., die Speicherverwaltung kennt keine Segmente. Segmente haben jeweils eigene virtuelle Adressräume, die wir sonst unterscheiden müssten. Insgesamt stehen 13 VAX-Traces und 9 SPARC-Traces zur Verfügung. Diese sind endliche Folgen virtueller Speicheradressen, die alle mit einer von drei möglichen Markierungen versehen sind. Die Markierung gibt an, ob es sich um einen lesenden Zugriff auf Daten (data read), schreibenden Zugriff auf Daten (data write) oder um einen Zugriff handelt, der durch das Lesen des nächsten Maschinenbefehls entstanden ist (instruction fetch). Bild 19.1 zeigt ein Beispiel. In unserem Modell des Seitenwechselproblems wird nicht zwischen lesenden und schreibenden Zugriffen unterschieden. Daher berücksichtigen wir im Folgenden die Markierungen der Anfragen nicht.

Für jede der beiden Architekturen sind hier nur je zwei Traces herausgegriffen (siehe Tabelle 19.1). Die SPARC-Traces sind Aufzeichnungen der Anfragesequenzen,

2	2020	2	2040	2	2064	1	e4e0ffb3	2	513c
2	2024	2	2044	2	2068	2	5128	2	5140
0	ffffe100	0	2208	2	5110	1	e4e0ffaf	2	5144
2	2028	2	2048	2	5114	2	512c	2	5148
2	202c	2	204c	2	5118	0	ffffe0e8	2	6280
2	2030	2	2050	2	511c	2	5130	2	6284
2	2034	2	205c	1	e4e0ffff	0	ffffe104	1	75dfffae
2	2038	2	2060	2	5120	2	5134	2	6288
2	203c	2	6a90	1	e4e10003	2	5138	2	628c
1	200100	2	6a94	2	5124	1	b1da00af	2	6290

Bild 19.1: Die ersten 50 Anfragen aus dem Trace „085.gcc" von einem SUN-SPARC-Rechner. Jede Adresse ist mit einer Markierung versehen, die den Typ der Anfrage angibt (data read (0), data write (1), instruction fetch (2)).

die beim Ausführen der SPEC-CINT92-Benchmarks entstehen und haben die Namen „085.gcc" und „026.compress". Der Benchmark „085.gcc" übersetzt ein C-Quellprogramm mit dem GNU-C-Compiler in SUN-3-Assemblersprache. Der Benchmark „026.compress" komprimiert Daten mit adaptiver Lempel-Ziv-Kodierung. Sämtliche SPARC-Traces haben eine Länge von 10 Millionen Anfragen, d. h., die Sequenzen wurden nach 10 Millionen Anfragen abgeschnitten. Bei den VAX-Traces ist nicht bekannt, um welche Anwendung es sich handelt. Die VAX-Traces haben unterschiedliche Längen. Die beiden Traces „spic" und „pasc" wurden hauptsächlich ausgewählt, weil sie mit Längen von etwa 400.000 Anfragen mit zu den längsten in dieser Gruppe gehören.

Im Folgenden wird angenommen, dass bei der SPARC-Architektur die Seitengröße 2048 Byte beträgt. (Auf moderneren SPARC-Maschinen sind auch größere Seitengrößen von z. B. 4096 Byte oder 8192 Byte üblich.) Die feste Seitengröße der VAX-Architektur wird (z. B. in Silberschatz und Galvin, 1994) mit 512 Byte angegeben. Um aus den Anfragen an Adressen im virtuellen Speicher Seitennummern zu

Name	Architektur	Länge	Seiten
026.compress	SPARC	10 Mio.	229
085.gcc	SPARC	10 Mio.	276
pasc	VAX	ca. 400 Tsd.	499
spic	VAX	ca. 400 Tsd.	384

Tabelle 19.1: Exemplarisch untersuchte Traces. Die Zahl der angefragten Seiten resultiert aus den angenommenen Seitengrößen von 2048 Byte und 512 Byte für die SPARC- bzw. VAX-Architektur.

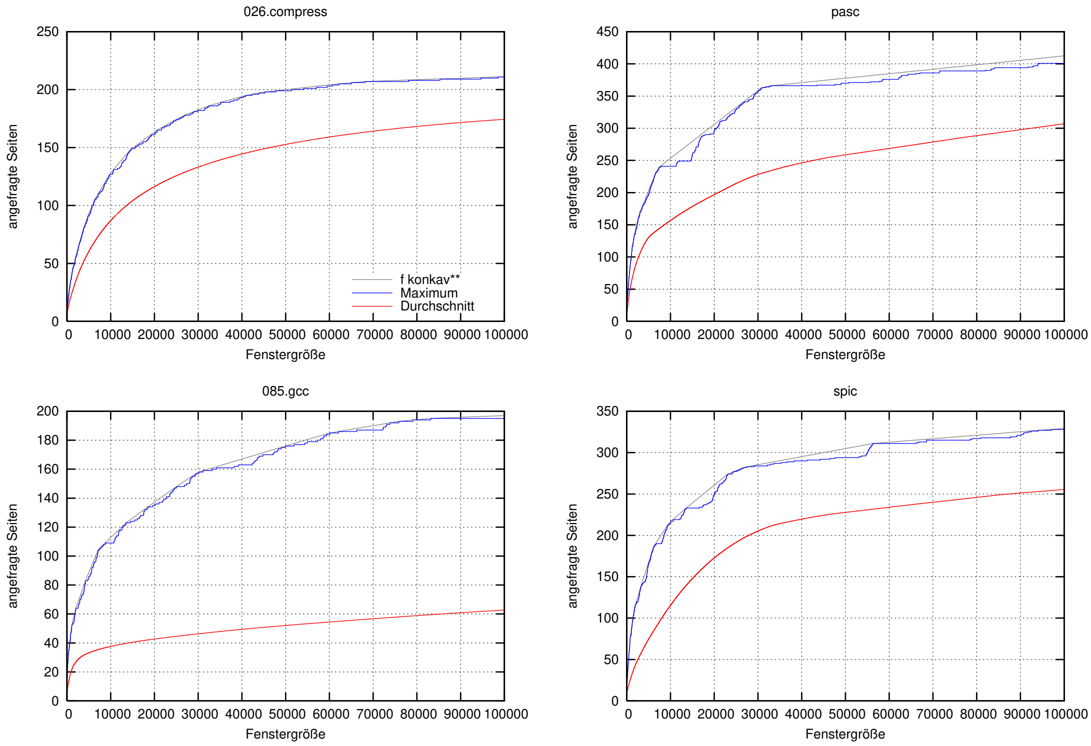

Bild 19.2: Maximale und durchschnittliche Zahl angefragter Seiten in Fenstern bis zur Länge von 100.000 Anfragen. Für das Maximumsmodell zeigen die Diagramme jeweils eine geeignete Funktion f, die konkav** ist, sodass die Funktion, die durch die Maximumswerte beschrieben wird, f genügt.

erzeugen, werden also die niederwertigsten 11 bzw. 9 Bits der angefragten Adressen gestrichen.

Zunächst betrachten wir die maximale und durchschnittliche Arbeitsmengengröße über alle Fenster einer Länge ℓ in diesen Sequenzen. Bild 19.2 zeigt die Ergebnisse für die vier ausgewählten Sequenzen. Der Parameter ℓ variiert zwischen 1 und 100.000 Anfragen. Im Falle der VAX-Traces entspricht die obere Grenze etwa einem Viertel der Sequenzlänge.

Für die durchschnittliche Arbeitsmengengröße ergibt sich für alle vier Sequenzen in dem abgebildeten Bereich nicht nur der monotone Anstieg der Kurven, sondern auch die mit 1 beginnende, aber dann immer weiter abnehmende Steigung der Kurven. In diesem Bereich sind die Durchschnittskurven konkav*. In den Beispielen finden wir das Modell von Denning (1968) auch global im Durchschnitt über alle Fenster der Sequenz bestätigt. Für sehr große Fenstergrößen, größer als die halbe Sequenzlänge, knicken die Durchschnittskurven in der Regel nach oben ab, schlagen also von konkav nach konvex um. Im Durchschnittsmodell ist dieser Bereich aber überhaupt nicht mehr relevant, da wir f nur für sinnvolle Cachegrößen k auswerten.

Diese liegen in allen vier Beispielen bei weniger als 500 Seiten (siehe Tabelle 19.1), denn der Cache braucht nicht größer als die Zahl der insgesamt angefragten Seiten zu werden.

Wie erwartet sind die Kurven für die maximale Arbeitsmengengröße zwar monoton wachsend, jedoch nicht in allen Punkten konkav. Sie können jedoch leicht nach oben durch Funktionen f beschränkt werden, die konkav** sind. Die in den Diagrammen gezeigten Funktionen f wurden mit dem am Ende von Abschnitt 15.4 beschriebenen Verfahren konstruiert. Man beobachtet, dass gemessen an der Sequenzlänge sowohl die durchschnittliche als auch die maximale Zahl angefragter Seiten für alle Fenstergrößen sehr klein ist.

Im Folgenden vergleichen wir die tatsächlichen Fehlerraten von LRU, FIFO und LFD mit den theoretischen Schranken. Dabei muss man im Auge behalten, dass die Schranken z. T. nur asymptotisch für $n \to \infty$ korrekt sind und sich auf den Worst Case beziehen. Da wir nicht glauben, dass typische Eingabesequenzen, die einer Funktion f genügen, Worst-Case-Sequenzen für f sind, erwarten wir auch nicht, dass die Schranken sehr dicht bei den tatsächlichen Fehlerraten liegen. Um die tatsächlichen Fehlerraten zu bestimmen, wurden die Seitenwechselstrategien für alle Cachegrößen zwischen 1 und der Zahl angefragter Seiten auf jede der Sequenzen angewendet.

Im Durchschnittsmodell fallen die folgenden vier Schranken zusammen: obere und untere Schranke für FIFO und LRU (vgl. Tabelle 18.1). Bei der Berechnung der Schranken im Durchschnittsmodell wurden als Funktion f die durchschnittliche Zahl angefragter Seiten selbst verwendet. In den Diagrammen in Bild 19.3 für FIFO und LFD ist die Schranke durch die oberste Kurve dargestellt. Im Maximumsmodell wurden die in Bild 19.2 gezeigten, konstruierten Funktionen f, die konkav** sind, verwendet. In den Diagrammen für FIFO und LRU zeigen die beiden mittleren Kurven die Schranken für FIFO und LRU im Maximumsmodell. Für LRU ist die obere Schranke gleich der unteren Schranke; für FIFO ist nur die obere eingetragen, denn die untere weicht nur gering von ihr ab. Nicht nur die Schranken für FIFO und LRU liegen dicht beieinander, auch die hier ermittelten Fehlerraten von FIFO und LRU (die untersten Kurven). In den untersuchten Sequenzen ist LRU jedoch nie schlechter als FIFO. Das deckt sich mit der Analyse des Maximumsmodells, die sich jedoch nur auf den Worst Case stützt. Für FIFO und LFD liefert das Durchschnittsmodell für kleine Cachegrößen die besseren Schranken. Für LFD ist der Vorteil des Durchschnittsmodells nicht mehr klar, zumal wir auch nicht wissen, ob die untere oder die obere Schranke des Maximumsmodells näher am Worst Case liegt, auf den sie sich ja beziehen.

Es wurde schon in Abschnitt 15.3 erwähnt, dass die „empirische Kompetitivität“ von FIFO und LRU nicht k ist, sondern eher durch eine kleine Konstante beschränkt ist (Young, 1991, Borodin und El-Yaniv, 1998). Dieselbe Beobachtung kann man auch an den hier untersuchten Sequenzen machen und diese Konstante mit etwa 5 beziffern. Bild 19.4 zeigt unter anderem die Fehlerraten von LRU und

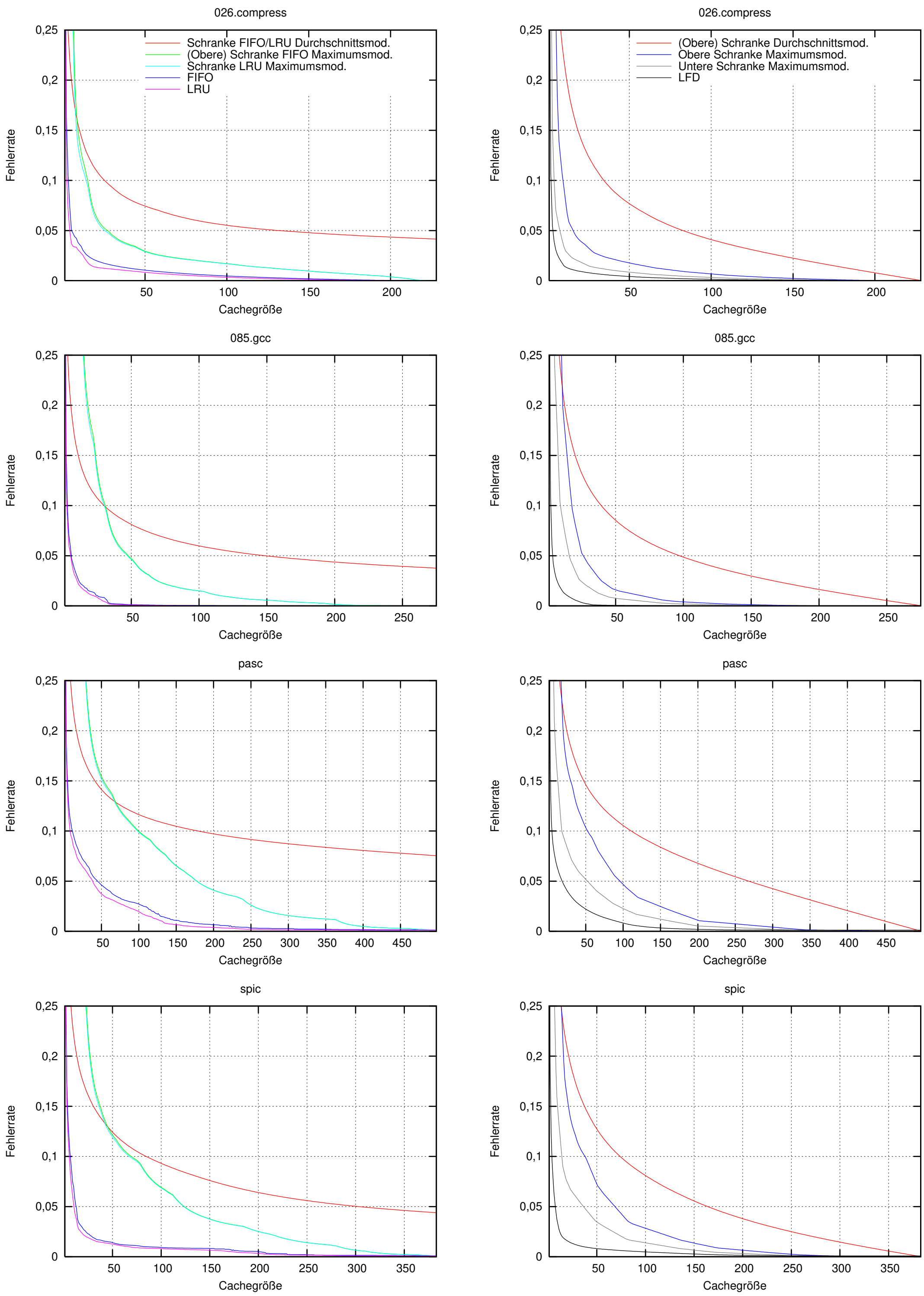

Bild 19.3: Links die Fehlerraten von LRU und FIFO und die Schranken des Maximums- und Durchschnittsmodells. Rechts Fehlerraten und Schranken für LFD.

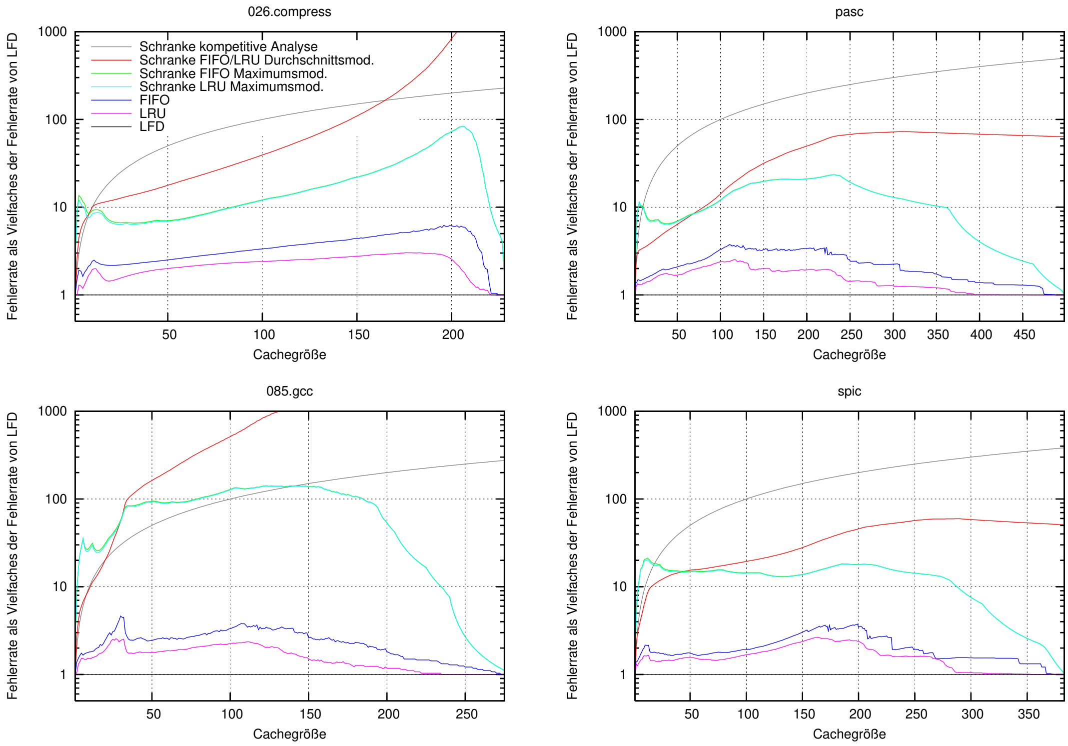

Bild 19.4: Fehlerraten und Schranken aus dem Maximums-, dem Durchschnittsmodell und aus der kompetitiven Analyse für LRU und FIFO. Die Ordinaten sind logarithmisch verkürzt.

FIFO gemessen als Vielfache der Fehlerrate der optimalen Offline-Strategie LFD. Die Kurven geben also unmittelbar die „gemessene Kompetitivität" für die verschiedenen Cachegrößen an. Die Ergebnisse der reinen kompetitiven Analyse, dass FIFO und LRU k-kompetitiv sind, weichen fast überall mindestens um einen Faktor von etwa $k/5$ von der beobachteten Kompetitivität ab. Die Diskrepanz wird also mit wachsender Cachegröße immer größer und wächst (als Faktor ausgedrückt) linear mit der Cachegröße.

Ebenfalls in Bild 19.4 eingetragen sind die (oberen) Schranken für die Fehlerraten aus dem Maximums- und Durchschnittsmodell. Auch hier beobachtet man, dass das Durchschnittsmodell im Vergleich mit dem Maximumsmodell für FIFO und LRU bei kleinen Cachegrößen – vielleicht den realistischen Cachegrößen – die besseren Schranken liefert. Bei größeren Cachegrößen kann die Schranke aber irgendwann schlechter als die aus der kompetitiven Analyse werden (026.compress und 085.gcc). Beim Maximumsmodell verhält es sich eher umgekehrt. Für kleine Cachegrößen zeigen die Beispiele Schranken, die schlechter als die Schranken aus dem Durchschnittsmodell sind und zum Teil sogar schlechter als die der kompetitiven Analyse sind. Anders als im Durchschnittsmodell folgen die Schranken im Maximumsmodell jedoch dem Verhalten von LRU und FIFO: Wenn die Fehlerraten von FIFO oder LRU für wachsendes k gemessen an der Fehlerrate von LFD steigen oder sinken, dann gilt dies im Wesentlichen auch für die zugehörigen Schranken. Einen linearen Anstieg des Quotienten aus den Schranken und der Fehlerrate von LFD (wie bei der kompetitiven Analyse) kann man beim Maximumsmodell aber nicht beobachten. Für Cachegrößen nahe bei der Zahl angefragter Seiten nähert sich der Quotient wieder dem kleinstmöglichen Wert 1 an.

Das bessere Abschneiden der Schranken des Durchschnittsmodells bei kleinen Cachegrößen kann damit erklärt werden, dass die Schranken im Durchschnittsmodell die Funktion f für kleine Fenstergrößen auswertet, denn k überschreitet sinnvollerweise ja nicht die Zahl angefragter Seiten. Information über die Anfragesequenz wird also aus vergleichsweise kurzen Fenstern gewonnen. Für kleine Cachegrößen erscheint uns diese Information intuitiv relevanter als Information über sehr lange Fenster, denn ein hohes Maß an Anfragelokalität in langen Fenstern kann auf kurze Sicht trotzdem Anfragen an viele verschiedene Seiten bedeuten.

Die Schranken im Maximumsmodell werten hingegen f^{-1} aus. Wir haben am Anfang der Untersuchung gesehen, dass dies bei den Beispielsequenzen für nicht allzu kleine Werten von k Information über Fenster liefert, die ganz erheblich länger sind als k. Die Schranken im Maximumsmodell nutzen also Information über vergleichsweise lange Fenster. Es entspricht unserer Intuition, dass für große Cachegrößen das längerfristige Verhalten der Anfragesequenz relevant ist und deshalb die Schranken im Maximumsmodell hier den Schranken des Durchschnittsmodells überlegen sind.

20 Zusammenfassung

Es wurden zwei Modelle zur Analyse des Seitenwechselproblems eingeführt, die Anfragelokalität mittels der durchschnittlichen oder maximalen Zahl Seiten modellieren, die in allen Fenstern einer festen Länge angefragt werden. Die Modelle ermöglichen unmittelbar die Analyse der Fehlerraten gängiger Seitenwechselstrategien, was im Rahmen der kompetitiven Analyse nicht möglich ist.

In beiden Modellen wurden exakte untere Schranken für alle Online-Strategien und exakte oder beinahe exakte obere und untere Schranken für LRU, FIFO und die Klasse der Markierungsstrategien hergeleitet. Im Durchschnittsmodell ist dies auch für LFD gelungen. Im Maximumsmodell liegen die obere und die untere Schranke für LFD etwa um einen Faktor von 2 auseinander. Im Durchschnittsmodell sind FIFO und LRU gleich gut, im Maximumsmodell ist LRU der Strategie FIFO überlegen. Aufgrund der Untersuchung einiger Beispielsequenzen scheint es so, dass das Durchschnittsmodell für kleine Cachegrößen und das Maximumsmodell für große Cachegrößen die besseren Schranken liefert.

Anhang

A Mathematische Hilfsmittel

Die folgenden mathematischen Sätze und Abschätzungen werden an zahlreichen Stellen in dieser Arbeit angewendet. Man findet sie in ähnlicher Form z. B. in Schickinger und Steger (2001), Motwani und Raghavan (1995), Bronstein und Semendjajew (1985) oder Forster (1992).

Multiplikationssatz. Es seien die Ereignisse $A_1, \ldots, A_n$ gegeben. Falls die Wahrscheinlichkeit $\mathrm{Prob}(A_1 \cap \cdots \cap A_n) > 0$ ist, gilt

$$\begin{aligned} &\mathrm{Prob}(A_1 \cap \cdots \cap A_n) \;=\\ &\quad \mathrm{Prob}(A_1) \cdot \mathrm{Prob}(A_2 \mid A_1) \cdot \mathrm{Prob}(A_3 \mid A_1 \cap A_2) \cdots \mathrm{Prob}(A_n \mid A_1 \cap \cdots \cap A_{n-1}). \end{aligned}$$

Satz von der totalen Wahrscheinlichkeit. Die Ereignisse $A_1, \ldots, A_n$ seien paarweise disjunkt und es gelte $B \subseteq A_1 \cup \cdots \cup A_n$ sowie $\mathrm{Prob}(A_1), \ldots, \mathrm{Prob}(A_n) > 0$. Dann folgt

$$\mathrm{Prob}(B) \;=\; \sum_{i=1}^{n} \mathrm{Prob}(B \mid A_i) \cdot \mathrm{Prob}(A_i).$$

Wenn zusätzlich $A_1 \cup \cdots \cup A_n = \Omega$ ist, gilt für Zufallsvariablen $Z\colon \Omega \to \mathbb{R}$

$$E(Z) \;=\; \sum_{i=1}^{n} E(Z \mid A_i) \cdot \mathrm{Prob}(A_i),$$

sofern die Erwartungswerte auf der rechten Seite existieren.

Markoffungleichung. Es sei Z eine Zufallsvariable mit Erwartungswert $E(Z)$, die nur nicht negative Werte annimmt. Für $t \in \mathbb{R}^+$ gilt dann

$$\mathrm{Prob}\big(Z \geq t \cdot E(Z)\big) \;\leq\; \frac{1}{t}$$

und äquivalent dazu

$$\mathrm{Prob}(Z \geq t) \;\leq\; \frac{E(Z)}{t}.$$

Chernoffschranken. Es seien $X_1, \ldots, X_n \in \{0,1\}$ unabhängige, bernoulliverteilte Zufallsvariablen mit $\mathrm{Prob}(X_i = 1) = p_i$ und $\mathrm{Prob}(X_i = 0) = 1 - p_i$ für $1 \le i \le n$. Für $X := \sum_{i=1}^n X_i$ und $\mu := E(X) = \sum_{i=1}^n p_i$ gelten folgende Ungleichungen:

$$\begin{aligned}
\mathrm{Prob}\bigl(X \ge (1+\delta)\mu\bigr) \;&\le\; \mathrm{e}^{-\mu\delta^2/3} && \text{für } 0 < \delta < 1,\\
\mathrm{Prob}\bigl(X \le (1-\delta)\mu\bigr) \;&\le\; \mathrm{e}^{-\mu\delta^2/2} && \text{für } 0 < \delta < 1 \text{ und}\\
\mathrm{Prob}(X \ge t) \;&\le\; 2^{-t} && \text{für } t \ge 2\mathrm{e}\mu.
\end{aligned}$$

Sammelbilderproblem. Jeder Packung eines Produktes ist ein Sammelbild beigelegt. Insgesamt gibt es n verschiedene Typen von Sammelbildern. Es sei Z die Zahl der Packungen, die man kaufen muss, um von jedem Typ mindestens ein Sammelbild zu besitzen. Wenn das Sammelbild, das man mit jeder gekauften Packung erhält, dem unabhängigen Ziehen eines Sammelbildtyps entspricht, bei dem jeder Typ Wahrscheinlichkeit $1/n$ hat, gilt

$$E(Z) \;=\; n \cdot H_n \;=\; n \ln n + O(n).$$

Abschätzung harmonischer Zahlen. Für die n-te harmonische Zahl

$$H_n \;:=\; \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

gilt für alle $n \in \mathbb{N}$

$$\ln n \;<\; H_n \;\le\; (\ln n) + 1.$$

Abschätzungen für Binomialkoeffizienten.

$$\binom{n}{k} \;\le\; \frac{n^k}{k!} \qquad \text{für } n \ge k \ge 0,$$

$$\left(\frac{n}{k}\right)^k \;\le\; \binom{n}{k} \;\le\; \left(\frac{n\mathrm{e}}{k}\right)^k \qquad \text{für } n \ge k \ge 0.$$

Abschätzungen für Reihen.

$$\sum_{i=0}^{\infty} \frac{1}{2^i} \;=\; 2, \qquad \sum_{i=0}^{\infty} \frac{i}{2^i} \;=\; 2,$$

$$\sum_{i=1}^{\infty} \frac{1}{i^k} \;\le\; 2 \qquad \text{für } k \ge 2,$$

$$\sum_{i=0}^{n} q^i = \frac{1-q^{n+1}}{1-q} \qquad \text{für } q \neq 1,$$

$$\sum_{i=0}^{\infty} q^i = \frac{1}{1-q} \qquad \text{für } |q| < 1.$$

Abschätzungen mit e**.**

$$\mathrm{e}^x \geq 1 + x \qquad \text{für } x \in \mathbb{R},$$

$$\left(1 - \frac{1}{x}\right)^x \leq \mathrm{e}^{-1} \leq \left(1 - \frac{1}{x}\right)^{x-1} \qquad \text{für } x \in \mathbb{R} \text{ und } x > 1.$$

Abschätzung für die Fakultätsfunktion.

$$n! > \left(\frac{n}{\mathrm{e}}\right)^n \qquad \text{für } n \in \mathbb{N}.$$

B Symbolverzeichnis

$\preceq$, $\preceq_f$ .... Präferenzordnungen, siehe Definition 9.1 und Definition 9.5

$]a, b[$, $[a, b]$ ...... offenes und abgeschlossenes Intervall mit Grenzen a und b

$\mathrm{Bild}(f)$ .... bezeichnet für $f\colon X \to Y$ die Menge $\{f(x) \mid x \in X\}$.

e .... eulersche Zahl 2,71...

$E(Z)$ .... Erwartungswert der Zufallsvariablen Z

$E(Z \mid B)$ .... bedingter Erwartungswert von Z, gegeben das Ereignis B

f^{-1} .... inverse Funktion zu f, für Teil III siehe Definition 16.1

F, F^* .... in Teil II ein partiell geordneter Zielraum F und die Paretofront $F^* \subseteq F$

H_n .... die n-te harmonische Zahl $1/1 + 1/2 + 1/3 + \cdots + 1/n$

$H(x, y)$ .... Hammingabstand $\sum_{i=1}^{n} |x_i - y_i|$ zwischen $x, y \in \{0, 1\}^n$

ln, log, $\log_k$ .... Logarithmus zur Basis e, zur Basis 2 und zur Basis k

$|M|$ .... Kardinalität der Menge M

M^n .... n-faches kartesisches Produkt $M \times \cdots \times M$ der Menge M

$\mathbb{N}$, $\mathbb{N}_0$ .... Menge der natürlichen Zahlen $\{1, 2, 3, \ldots\}$ und Menge der natürlichen Zahlen vereinigt mit der Menge $\{0\}$

$\mathrm{Prob}(A)$ .... Wahrscheinlichkeit des Ereignisses A

$\mathrm{Prob}(A \mid B)$ .... bedingte Wahrscheinlichkeit von A, gegeben das Ereignis B

$\mathbb{R}$, $\mathbb{R}^+$, $\mathbb{R}_0^+$, $\mathbb{R}^{\geq 1}$ Menge der reellen Zahlen, Menge der positiven reellen Zahlen, Menge der nicht negativen reellen Zahlen und Menge der reellen Zahlen größer oder gleich 1

S, S^* .... in Teil II ein beliebiger Suchraum S und die Paretomenge $S^* \subseteq S$

$|x|$ .... Anzahl der 1-Bits von $x \in \{0, 1\}^n$

X^n, X^* .... in Teil II der Suchraum $X^n := \{0, 1\}^n$ und die Paretomenge $X^* \subseteq X^n$

$\mathbb{Z}$ .... Menge der ganzen Zahlen

C Literaturverzeichnis

Albers, S., Favrholdt, L. M. und Giel, O. (2002). On paging with locality of reference. In *Proceedings of the 34th Annual Symposium on Theory of Computing (STOC '02)*, ACM, S. 258–267.

Albers, S., Favrholdt, L. M. und Giel, O. (2005). On paging with locality of reference. *Journal of Computer and System Sciences*, Band 70(2), S. 145–175.

Belady, L. A. (1966). A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, Band 5(2), S. 78–101.

Berge, C. (1957). Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, Band 43, S. 842–844.

Borodin, A. und El-Yaniv, R. (1998). *Online Computation and Competitive Analysis*. Cambridge University Press.

Borodin, A., Irani, S., Raghavan, P. und Schieber, B. (1995). Competitive paging with locality of reference. *Journal of Computer and System Sciences*, Band 50(2), S. 244–258.

Briest, P., Brockhoff, D., Degener, B., Englert, M., Gunia, C., Heering, O., Jansen, T., Leifhelm, M., Plociennik, K., Röglin, H., Schweer, A., Sudholt, D., Tannenbaum, S. und Wegener, I. (2004a). *Evolutionäre Algorithmen zwischen experimenteller und theoretischer Analyse*. Fachbereich Informatik, Universität Dortmund, Endbericht der Projekgruppe 427.

Briest, P., Brockhoff, D., Degener, B., Englert, M., Gunia, C., Heering, O., Jansen, T., Leifhelm, M., Plociennik, K., Röglin, H., Schweer, A., Sudholt, D., Tannenbaum, S. und Wegener, I. (2004b). Experimental supplements to the theoretical analysis of EAs on problems from combinatorial optimization. In *Proceedings of the 8th Conference on Parallel Problem Solving from Nature (PPSN '04)*, Band 3242 von *Lecture Notes in Computer Science*, Springer, S. 21–30.

Bronstein, I. N. und Semendjajew, K. A. (1985). *Taschenbuch der Mathematik*. Harri Deutsch, Thun und Frankfurt/Main, 22. Auflage.

Bäck, T., Fogel, D. B. und Michalewicz, Z., Herausgeber (1997). *Handbook of Evolutionary Computation*. Institute of Physics Publishing.

Chrobak, M. und Noga, J. (1998). LRU is better than FIFO. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM/SIAM, S. 78–81.

Chung, K. L. (1974). *Elementary Probability Theory with Stochastic Processes*. Springer.

Coello Coello, C. A., Van Veldhuizen, D. A. und Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems*. Kluwer Academic Publishers.

Coppersmith, D., Tetali, P. und Winkler, P. (1993). Collisions among random walks on a graph. *SIAM Journal on Discrete Mathematics*, Band 6, S. 363–374.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley.

Denning, P. J. (1968). The working set model for program behaviour. *Communications of the ACM*, Band 11(5), S. 323–333.

Droste, S., Jansen, T., Tinnefeld, K. und Wegener, I. (2003). A new framework for the valuation of algorithms for black-box optimization. In *Foundations of Genetic Algorithms 7 (FOGA '02)*, Morgan Kaufmann, S. 253–270.

Droste, S., Jansen, T. und Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, Band 276, S. 51–81.

Droste, S., Jansen, T. und Wegener, I. (2005). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, erscheint.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics*, Band 17, S. 449–467.

Ehrgott, M. (2000). Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, Band 7, S. 5–31.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications*, Band 1. Wiley.

Fiat, A. und Karlin, A. R. (1995). Randomized and multipointer paging with locality of reference. In *Proceedings of the 27th Annual Symposium on Theory of Computing (STOC '95)*, ACM, S. 626–634.

Fiat, A. und Mendel, M. (1997). Truly online paging with locality of reference. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, IEEE, S. 326–335.

Forster, O. (1992). *Analysis 1*. Vieweg, Braunschweig/Wiesbaden, 4. Auflage.

Giel, O. (2003). Expected runtimes of a simple multi-objective evolutionary algorithm. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC '03)*, Band 3, IEEE, S. 1918–1925.

Giel, O. und Wegener, I. (2003). Evolutionary algorithms and the maximum matching problem. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS '03)*, Band 2607 von *Lecture Notes in Computer Science*, Springer, S. 415–426.

Giel, O. und Wegener, I. (2004). Searching randomly for maximum matchings. Technischer Bericht, Electronic Colloquium on Computational Complexity (ECCC), Report Nummer 76(2004).

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley.

Hajek, B. (1982). Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, Band 14, S. 502–525.

Harary, F. (1969). *Graph Theory*. Perseus Books, Reading, Massachusetts.

He, J. und Yao, X. (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, Band 127, S. 57–85.

Hopcroft, J. E. und Karp, R. M. (1973). An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, Band 2(4), S. 225–231.

Irani, S., Karlin, A. R. und Phillips, S. (1996). Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, Band 25(3), S. 477–497.

Jansen, T. und Wegener, I. (2001). Evolutionary algorithms – how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, Band 5, S. 589–599.

Jerrum, M. (1992). Large cliques elude the Metropolis process. *Random Structures and Algorithms*, Band 3(4), S. 347–359.

Jerrum, M. und Sinclair, A. (1989). Approximating the permanent. *SIAM Journal on Computing*, Band 18(6), S. 1149–1178.

Jerrum, M. und Sorkin, G. B. (1998). The Metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, Band 82, S. 155–175.

Karlin, A. R., Phillips, S. J. und Raghavan, P. (2000). Markov paging. *SIAM Journal on Computing*, Band 30(3), S. 906–922.

Kirkpatrick, S., Gelatt, C. D. und Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, Band 220, S. 1087–1092.

Kumar, R. und Banerjee, N. (2005). Runtime analysis of a multiobjective evolutionary algorithm on simple and hard problems. In *Foundations of Genetic Algorithms 8 (FOGA '05)*, erscheint.

Laumanns, M., Thiele, L., Deb, K. und Zitzler, E. (2002a). Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, Band 10(3), S. 263–282.

Laumanns, M., Thiele, L. und Zitzler, E. (2004a). An adaptive scheme to generate the Pareto front based on the epsilon-constraint method. Technischer Bericht, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), TIK-Report No. 199.

Laumanns, M., Thiele, L. und Zitzler, E. (2004b). Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Natural Computing*, Band 3, S. 37–51.

Laumanns, M., Thiele, L. und Zitzler, E. (2004c). Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. *IEEE Transactions on Evolutionary Computation*, Band 8(2), S. 170–182.

Laumanns, M., Thiele, L., Zitzler, E., Welzl, E. und Deb, K. (2002b). Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Proceedings of the 7th International Conference on Parallel Problem Solving From Nature (PPSN '02)*, Band 2439 von *Lecture Notes in Computer Science*, Springer, S. 44–53.

Lehot, P. (1974). An optimal algorithm to detect a line graph and output its root graph. *Journal of the Association for Computing Machinery*, Band 21(4), S. 569–575.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. und Teller, E. (1953). Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, Band 21, S. 1087–1092.

Micali, S. und Vazirani, V. V. (1980). An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science (FOCS '80)*, IEEE, S. 17–27.

Motwani, R. und Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

Neumann, F. (2004). Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. In *Proceedings of the 8th Conference on Parallel Problem Solving from Nature (PPSN '04)*, Band 3242 von *Lecture Notes in Computer Science*, Springer, S. 80–89.

Neumann, F. und Wegener, I. (2004). Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In *Proceedings of the 6th Genetic and Evolutionary Computation Conference (GECCO '04)*, Band 3102 von *Lecture Notes in Computer Science*, Springer, S. 713–724.

Neumann, F. und Wegener, I. (2005). Minimum spanning trees made easier via multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, erscheint.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart.

Rechenberg, P. und Pomberger, G., Herausgeber (1997). *Informatik-Handbuch*. Carl Hanser Verlag, München/Wien.

Rudolph, G. (1998a). Evolutionary search for minimal elements in partially ordered finite sets. In *Proceedings of the 7th Annual Conference on Evolutionary Programming*, Band 1447 von *Lecture Notes in Computer Science*, Springer, S. 345–353.

Rudolph, G. (1998b). On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, IEEE, S. 511–516.

Rudolph, G. (2001a). Evolutionary search under partially ordered fitness sets. In *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 01)*, ICSC Academic Press, S. 818–822.

Rudolph, G. (2001b). Some theoretical properties of evolutionary algorithms under partially ordered fitness values. In *Proceedings of the Evolutionary Algorithms Workshop (EAW '01)*, Bukarest, S. 9–22.

Rudolph, G. und Agapie, A. (2000). Convergence properties of some multi-objective evolutionary algorithms. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC '00)*, IEEE, S. 1010–1016.

Sasaki, G. (1991). The effect of the density of states on the Metropolis algorithm. *Information Processing Letters*, Band 37(3), S. 159–163.

Sasaki, G. H. und Hajek, B. (1988). The time complexity of maximum matching by simulated annealing. *Journal of the ACM*, Band 35, S. 387–403.

Scharnow, J., Tinnefeld, K. und Wegener, I. (2002). Fitness landscapes based on sorting and shortest paths problems. In *Proceedings of the 7th Conference on Parallel Problem Solving from Nature (PPSN '02)*, Band 2439 von *Lecture Notes in Computer Science*, Springer, S. 54–63.

Schickinger, T. und Steger, A. (2001). *Diskrete Strukturen*, Band 2. Springer.

Schwefel, H. P. (1995). *Evolution and Optimum Seeking*. Wiley.

Silberschatz, A. und Galvin, P. (1994). *Operating System Concepts*. Addison-Wesley, 4. Auflage.

Sleator, D. D. und Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM*, Band 28, S. 202–208.

Thierens, D. (2003). Convergence time analysis for the multi-objective counting ones problem. In *Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization (EMO '03)*, Band 2632 von *Lecture Notes in Computer Science*, Springer, S. 355–364.

Torng, E. (1998). A unified analysis of paging and caching. *Algorithmica*, Band 20, S. 175–200.

Wegener, I. (2003). *Komplexitätstheorie – Grenzen der Effizienz von Algorithmen*. Springer.

Wegener, I. (2005). Simulated annealing beats Metropolis in combinatorial optimization. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP '05)*, erscheint.

Wegener, I. und Witt, C. (2005). On the optimization of monotone polynomials by simple randomized search heuristics. *Combinatorics, Probability, and Computing*, Band 14, S. 225–247.

Witt, C. (2004). *Über die Analyse randomisierter Suchheuristiken und den Entwurf spezialisierter Algorithmen im Bereich der kombinatorischen Optimierung*. Dissertation, Universität Dortmund.

Young, N. (1991). *Competitive Paging and Dual-Guided On-Line Weighted Caching and Matching Algorithms*. Dissertation, Princeton University, Princeton University Computer Science Technical Reports, TR-348-91.