UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

Constraint Programming versus Logik
Vergleich zweier Ansätze zur Aufstellungsplanung von Chemieanlagen

Hanna Köpcke und Andreas Schröder

No. CI-183/04

Technical Report ISSN 1433-3325 October 2004

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence," at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

Constraint Programming versus Logik
Vergleich zweier Ansätze zur Aufstellungsplanung von Chemieanlagen

Hanna Köpcke und Andreas Schröder

Universität Dortmund
Fachbereich Informatik, Lehrstuhl für künstliche Intelligenz
E-Mail: {koepcke,schroede}@ls8.cs.uni-dortmund.de

Zusammenfassung

Die Aufstellungsplanung von Chemieanlagen ist eine komplexe Aufgabe. Es handelt um einen Spezialfall des *quadratic assignment problem*, welches sich als NP-hart erwiesen hat. Zur Lösung dieser Aufgabe wurde am Lehrstuhl für Anlagentechnik ein Werkzeug entwickelt, das mit Hilfe eines Simulated Annealing Verfahrens einen sowohl funktionellen als auch möglichst kostengünstigen Platzierungsvorschlag generiert. Hierbei hat sich allerdings gezeigt, dass die Abkühlrate der Simulated Annealing-Methode durch empirische Parameter gesteuert werden muss, um Lösungen in vertretbarer Zeit zu erhalten. In dieser Arbeit werden zwei alternative Ansätze zur Lösung des Aufstellungsproblems untersucht. Der eine Ansatz beruht auf Techniken des Constraint Programming, der andere Ansatz verwendet Logik und Backtracking.

1 Einleitung

Aufgabe der Aufstellungsplanung ist die Plazierung sämtlicher für eine Chemieanlage relevanten, verfahrenstechnischen Equipments innerhalb und im Nahbereich außerhalb einer Stahlbaukonstruktion [12]. Dabei müssen Anforderungen an die Lage der Equipments im Stahlbau und zueinander berücksichtigt werden. Diese Anforderungen sind bestimmt durch eine Vielzahl von Vorgaben der einzelnen Equipments, die sich u.a. durch die Betriebsweise des Equipments, technische Spezifikationen und Vorgaben des Anlagenbetreibers ergeben. Anforderungen an die Positionierung von Equipments sind immer geprägt durch eine Reihe von konkurrierenden und widersprüchlichen Bedingungen, die abgewogen und gelöst werden müssen.

Zur Lösung dieser Problemstellung wurde am Lehrstuhl für Anlagentechnik der Universität Dortmund ein System zur computerunterstützten Aufstellungsplanung entwickelt. Für ein konkretes Planungsszenario (vorgegebenes Baufeld inklusive des Anlagengerüstes und der Anlagenwege sowie eine Anzahl zu platzierender Equipmentmodelle) werden die zu erfüllenden Anforderungen unter Verwendung einer in einer Datenbank hinterlegten Regelbasis abgeleitet. Die inferierten Anforderungen, d.h. die zu berücksichtigenden Abhängigkeiten zwi-

schen einzelnen Equipments bzw. zwischen einzelnen Equipments und dem Baufeld oder Stahlbau, werden in einem zweiten Schritt in eine numerische Form übertragen. Auf der Basis dieser numerischen Anforderungsrepräsentation wird nun ein heuristisches Suchverfahren (ein *Simulated Annealing Algorithmus* (SA)) zur Minimierung der Regelverstöße bzw. zur Maximierung der erfüllten Anforderungen verwendet. Die bisherigen Arbeiten haben gezeigt, dass eine Platzierung von Equipments im Anlagengebäude mit dem beschriebenen Verfahren möglich ist. Allerdings ist diese Methode für große Anlagen zu aufwändig und erfordert viel Erfahrung bei der Festlegung der SA-Strategieparameter, insbesondere der Abkühlvorschriften.

Aus diesem Grunde werden in dieser Arbeit zwei alternative Lösungsstrategien untersucht. Bei der ersten Lösungsstrategie wird das Problem der Aufstellungsplanung als Konfigurationsproblem formalisiert und Verfahren des Constraint Logic Programming (CLP) angewendet. Die zweite Strategie beruht auf Hornlogik und einem Backtracking-Algorithmus.

2 Problembeschreibung

Die Aufstellungsplanung hat die Aufgabe, die benötigten Equipments einer Chemieanlage unter Berücksichtigung verschiedenster konstruktiver und verfahrenstechnischer Randbedingungen im Anlagengerüst zu positionieren [12]. Zusätzliche Bedingungen ergeben sich aus der Sicht der Montage, des Anlagenbetriebes sowie aufgrund sicherheitstechnischer Anforderungen. Die Entwicklung von Aufstellungsentwürfen ist daher eine sehr komplexe Aufgabe und erfordert ein großes Maß an Fachwissen aus unterschiedlichsten Fachrichtungen.

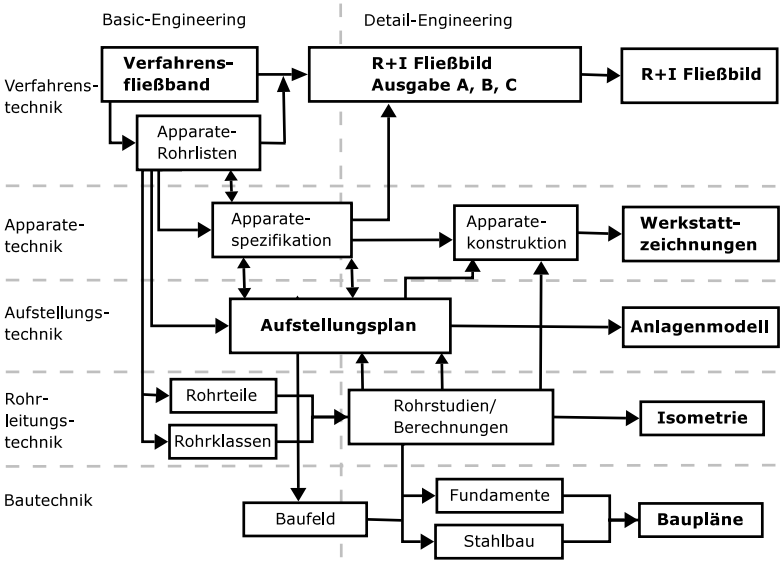


Abbildung 1: Einordnung der Aufstellungsplanung in den Planungsprozess für chemische Anlagen.

Abbildung 1 zeigt welche Stellung die Aufstellungsplanung im gesamten Pla-

nungsprozess einnimmt. Im *basic engineering* wird das chemische Verfahren und die benötigten Ausrüstungen ausgewählt und optimiert. Im anschließenden *extended basic engineering* sind die wesentlichen Equipments mit ihren Hauptabmessungen bekannt. Im Rahmen der Aufstellungsplanung werden die Equipments dann in einem vorgegebenen oder zu entwickelnden Stahlbau möglichst optimal platziert. Für die folgenden Phasen des *detail engineering* liegt damit das Layout der Anlage fest und kann aus Kosten- und Termingründen auch nicht mehr verändert werden. Dies macht deutlich, wie wichtig es ist, den Aufstellungsentwurf z.B. aufgrund von Variantenkonstruktionen zu optimieren, um die Konsequenzen auf nachfolgende Arbeiten, wie z.B. die Rohrleitungskonstruktion, möglichst genau ermitteln zu können.

2.1 Größenabschätzung des Suchraums

Es ist offensichtlich, dass es sich bei dem hier betrachteten Konfigurationsproblem um ein NP-hartes Problem handelt. Um für praktisch relevante Problemgrößen abschätzen zu können, ob eine Aufstellung in akzeptabler Zeit berechnet werden kann, soll zunächst untersucht werden, wie groß der Suchraum ist, in dem nach Lösungen gesucht wird, um beurteilen zu können, ob die Suche nach einer Lösung (einer Aufstellung) in diesem Raum effizient gestaltet werden kann.

Idealerweise können Komponenten (Equipments) beliebig plaziert werden, d.h. der Suchraum ist unendlich. Zur Vereinfachung des Problems wird allerdings eine Rasterung des Baufelds vorgenommen. Sei d die Kantenlänge zwischen zwei Rasterpunkten, dann wird das gegebene Baufeld der Fläche A in $R = \frac{A}{d^2}$ Raster aufgeteilt. Ferner sei n die Anzahl der Komponenten. Jeder Komponente wird mindestens ein Raster zugewiesen; es ist aber möglich, dass nicht alle Raster belegt werden. Damit ergibt sich die Anzahl k der möglichen Anlagen als:

$$k \leq \frac{\frac{A}{d^2}!}{(\frac{A}{d^2} - n)!} \qquad \text{mit } \frac{A}{d^2} \geq n$$

Für eine kleine Anlage mit $50m^2$ Fläche, einem Raster von $1cm^2$ und 30 Komponenten ergibt sich als Größe des Suchraums:

$$k \leq \frac{25.000.000!}{(25.000.000 - 30)!} \approx 8,67^{221}$$

3 Formalisierung

Unter dem Anlagenentwurfsproblem versteht man die Positionierung rechteckiger Anlagenteile innerhalb einer gegebenen Fläche [17]. Dabei sollen die durch die Flussmatrix $M = [m_{ij}]_{i,j=1,\ldots,n}$ induzierten Kosten minimiert werden. Diese Matrix definiert die Verbindungskosten zwischen zwei Komponenten i und j. Das Ziel ist eine nichtüberlappende Anordnung aller Komponenten mit minimalen Flusskosten $\sum_{i,j} d_{ij} \cdot m_{ij}$ mit d_{ij} als Distanz zwischen den betreffenden Komponenten. Diese Formalisierung schließt sowohl das Problem des Anlagenentwurfs als auch die Design von Schaltungen (VLSI Design) ein. Es ist ein Spezialfall des *quadratic assignment problem*, welches sich als NP-hart erwiesen hat [7].

Der Aufstellungsentwurf einer Chemieanlage ist eine Variante dieses Problems mit unregelmäßigen Grundflächen, mehreren Dimensionen durch Verwendung eines Stahlbaus, Zonen ohne Bebauung sowie vordefinierten Wegen, die ebenfalls nicht bebaut werden dürfen. Ähnliche Varianten wurden bereits in [11, 14] untersucht. Bewegliche Komponenten, also Maschinen und Anlagenteile, können darüberhinaus um ihr Zentroid rotiert werden. Eine weitere Besonderheit ist die Berücksichtigung von Anforderungen.

4 Anlagenentwurf als Constraint Satisfaction Problem

Der Aufstellungsentwurf einer Chemieanlage kann als Konfigurationsproblem aufgefasst und mit Verfahren des Constraint Logic Programming gelöst werden. Dazu wird das Problem als Constraint Satisfaction Problem formuliert.

Definition 1 *Ein Constraint satisfaction problem (CSP) ist definiert durch:*

- *eine Menge von Variablen $V = \{x_1, \ldots, x_n\}$. Jeder Variablen x_i ist ein endlicher Wertebereich D_i zugeordnet, der die möglichen Werte der Variablen angibt.*
- *eine Menge von Constraints $C = \{C_1, \ldots, C_r\}$. Ein k-stelliger Constraint C ist ein Paar bestehend aus einer Menge von Variablen $X = \{x_1, \ldots, x_k\}$ und einer entscheidbaren Relation R, wobei die $x_i, i = 1, \ldots, k$, Werte aus ihrem gegebenen Wertebereich D_i annehmen können und R eine Teilmenge von $D_1 \times \cdots \times D_k$ ist.*

Die Menge der Variablen ergibt sich beim Anlagenentwurfsproblem aus der zu platzierenden Komponenten. Jeder Komponente ist eine Variable zugeordnet, die ihre Position im Stahlbau repräsentiert. Der Wertebereich einer Variablen ist die Menge der möglichen Positionen auf dem Baufeld. Eine mögliche Position ist ein Tupel (x, y, z, r_x, r_y), dabei gibt z die Etage an, x und y definieren die Koordinaten in bezug auf die Etage. r_x und r_y geben die Rotation der Komponente an, wobei lediglich diskrete 90°Schritte und somit vier mögliche Orientierungen zugelassen werden. Idealerweise sollten Komponenten beliebig plaziert werden können, d.h. der Suchraum ist unendlich. Das Problem wird hier allerdings relaxiert, indem eine Rasterung des Baufeldes vorgenommen wird. Dazu wird das gegebene Baufeld der Fläche A in $R = \frac{A}{d^2}$ Raster aufgeteilt. Bei der Berechnung der möglichen Positionen einer Komponente werden gesonderte Flächen und Räume wie Wege, gesperrte Bereiche und andere Hindernisse, an denen die Komponente nicht plaziert werden darf, von vorneherein ausgeschlossen.

4.1 Umsetzung der Anforderungen in Constraints

Unäre und binäre Constraints werden aus den Regeln für die Lageanforderungen abgeleitet. Unäre Constraints sind Anforderungen an die Lage einer einzelnen Ausrüstung. Folgende Positionsangaben sind definiert: `am Weg`, `am Anlagenrand`, `in Etage` und `außerhalb des Stahlbaus`. Binäre Constraints definieren relative Lageanforderungen zweier Ausrüstungen zueinander. Es wird zwischen drei Beziehungen unterschieden: `neben`, `nahe` und `über`.

Die Anforderungen sind unterschiedlich wichtig. Es gibt Anforderungen, welche für das Funktionieren der Anlage oder aus Sicherheitsgründen eingehalten werden müssen, und solche, welche nur eine untergeordnete Rolle spielen. Die Gewichtung einer Regel wird über die Schlüsselwörter `muss, soll, sollte` ausgedrückt, wobei `muss` die maximale Wichtigkeit ausdrückt und `sollte` eine untergeordnete Bedeutung beschreibt.

Zum Erfassen des Erfüllungsgrades der Constraints bieten sich deshalb Techniken der Fuzzy-Logik an [4, 5]. In Anlehnung an eine linguistische Variable können ein- bzw. zweistellige Funktionen definiert werden, die auf einen Zugehörigkeitsgrad zu einem bestimmten Konzept wie Nachbarschaft oder vertikale Anordnung abbilden.

Diese Abbildung kann definiert werden als:

$$f : \prod_{x_i \in var(C)} D_i \mapsto [0,1]$$

D_i ist hierbei der Wertebereich der Variablen x_i, d.h. die Menge der möglichen Positionen für die Komponente k_i. Diese *Fuzzy-Relation* kann dann als Erfüllungsgrad von Constraints verwendet werden. Darüberhinaus ist eine Gewichtung durch Skalierung mit dem Regelgewicht möglich. Der Erfüllungsgrad e eines Constraints c berechnet sich für eine Regel r mit Gewicht $w_r \in \mathbb{R}$ als:

$$e = w_r \cdot f_c$$

Im Folgenden wird die Definition der Fuzzy-Constraints im Detail beschrieben. Für die Erfüllung der Anforderung `über` kommt es nicht auf eine genau senkrechte Ausrichtung an. Das Fuzzy-Constraint `über` bildet daher in Abhängigkeit vom von den Komponenten eingeschlossenen Winkel auf das Interval $[0,1]$ ab. Sei h_{ij} der Höhenunterschied zwischen Komponente i und Komponente j. Dieser Wert ist negativ, falls Komponente i unter Komponente j liegt. Dann errechnet sich der Erfüllungsgrad des Fuzzy-Constraints `über`:

$$ueber(i,j) = \begin{cases} 0 & \text{falls } h_{ij} < 0 \\ \max\left(0, 1 - \frac{\arctan \frac{d_{ij}}{h_{ij}}}{\alpha_{max}}\right) & \text{sonst} \end{cases}$$

Für alle Winkel größer α_{max} liefert die Funktion also ebenfalls 0.

Für eine geforderte Nebeneinanderanordnung `neben` zweier Komponenten kann der Abstand d_{ij} der Komponenten mit der Flächendiagonale d_2 der Grundfläche normiert werden. Die Komponenten müssen sich hierzu in der gleichen Ebene befinden:

$$neben(i,j) = \begin{cases} 0 & \text{falls } h_{ij} < 0 \\ \frac{d_{ij}}{d_2} & \text{sonst} \end{cases}$$

In gleicher Weise liefert die Definition von `nahe` die Ausprägung der Nachbarschaft in allen drei Raumdimensionen für die Raumdiagonale d_3:

$$nahe(i,j) = \frac{d_{ij}}{d_3}$$

Um die Anforderung `innerhalb des Stahlbaus` zu realisieren, wurden die folgenden Fuzzy-Constraints konstruiert:

$$innerhalb(i,j) = o(i,j) / \min(a(i), a(j))$$

$$ausserhalb(i,j) = 1 - innerhalb(i,j)$$

Dabei berechnet $a(i)$ die Fläche der Komponente i und $o(i,j)$ die Größe der überlappenden Flächen der Komponente mit dem Stahlbau j. Das Ergebnis liegt zwischen 0 und dem Minimum der Flächen der beiden Körper.

Für einige Komponenten existieren Regeln, die eine Anordnung in einer bestimmten Ebene j verlangen. Mit Hilfe der Höhe der Anlage h kann das folgende Fuzzy-Constraint `in_etage` definiert werden:

$$in\_etage(i,j) = 1 - \frac{h_{ij}}{h}$$

4.2 Forward Checking Algorithmus zur Lösung des CSP

Das definierte CSP ist überbestimmt. Es existieren verschiedene Ansätze zur Lösung von überbestimmten Constraint-Problemen [15]. Eine Möglichkeit besteht darin, das Problem zu relaxieren, indem weniger wichtige Constraints fallen gelassen werden. Dieses Vorgehen wird auch als partielle Constrainterfüllung bezeichnet. In [6] wird eine Übersicht über Methoden zur partiellen Constrainterfüllung gegeben.

Zur Berechnung einer Platzierung wurde ein Forward Checking Algorithmus [6] adaptiert. Der Algorithmus sortiert zunächst die Wertebereiche der Variablen nach dem Erfüllungsgrad der unären Constraints und initialisiert danach schrittweise die Variablen. Bei jeder Instanziierung einer Variablen werden Verletzungen binärer Constraints mit bereits belegten Variablen untersucht und die Auswirkungen dieser Wertzuweisung auf die Wertebereiche der zukünftigen Variablen propagiert. Dadurch wird versucht die Zahl der Knoten des Suchbaumes zu verringern, indem die Auswirkungen vorgenommener Wertzuweisungen auf die zukünftigen Definitionsbereiche schon bei der Wertzuweisung berücksichtigt werden.

4.2.1 Variablenauswahl

Einen wichtigen Einfluss auf die Suche nach einer Lösung hat die Reihenfolge, in der die Variablen belegt werden. Die Variablen werden im vorliegenden Fall nach der *minimal remaining values (MRV)* Heuristik [1, 2], auch first fail principle genannt, initialisiert. Sie ist ein Beispiel für eine dynamische Heuristik zur Anordnung der Variablen im Suchbaum, bei der die Variablen nach der Größe ihrer Domänen ausgewählt werden. Es wird jeweils die Komponente mit den wenigsten in Frage kommenden Orten plaziert.

4.3 Experimente und Ergebnisse

Bei der im Folgenden betrachteten Anlage handelt es sich um eine Anlage zur Gasbehandlung. Tabelle 1 fasst die Kennzahlen der Anlage zusammen. Die Anlage umfasst 28 Ausrüstungen. Die Größe des Baufeldes beträgt $50m \times 50$ m. Die Stahlbaukonstruktion weist im Erdgeschoss und in der ersten Etage 4×1, in der zweiten, dritten und vierten Etage 3×1 Stahlraster auf. Das Standardrastermaß beträgt $6 \times 6m$. Nach Ableitung der Regeln ergeben sich 100 Constraints, jeweils 50 unäre und binäre Constraints. Von den 50 unären Constraints sind 39 `muss`, 10 `soll` und 1 `sollte`. Von den 50 binären Constraints sind 45 `muss` und 5 `soll`.

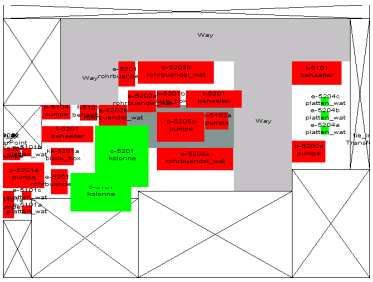
Anzahl der Komponenten	28
vorgegebene Platzierungen	5
freie Platzierungen	23
Anzahl der Verbindungen	108
Ebenen	4
Stahlbau	ja
verbotene Zonen	ja
Wege	ja
Regeln insgesamt	66
abgeleitete Constraints	100

Tabelle 1: Die Kennzahlen der Anlage, die mit dem hier vorgestellten Verfahren geplant wurde.

Constraints	Original	Variante 1	Variante 2
unäre Constraints			
muss	79,6%	95,2%	86,5%
soll	54,2%	83,3%	67,5%
sollte	100%	100%	100%
binäre Constraints			
muss	82,8%	86,4%	87,3%
soll	15,5%	57,4%	87,5%

Tabelle 2: Vergleich der Erfüllungsgrade der Constraints der beiden erzeugten Platzierungsvarianten mit der realen Platzierung

Zunächst wurde eine Rastergröße von 1 m gewählt. Die Constraints wurden mit den beim Simulted Annealing Ansatz verwendeten Gewichten bewertet. Entsprechend erhielten `muss`-Constraints ein Gewicht von 5, `soll`-Constraints ein Gewicht von 3 und `sollte`-Constraints ein Gewicht von 1. Abbildung 2 zeigt die generierte Aufstellung in einer zweidimensionalen Darstellung, Abbildung 3 zeigt den gleichen Konstruktionsvorschlag in einer 3D-Darstellung. In Tabelle 2 werden die Erfüllungsgrade der Constraints für den erzeugten Konstruktionsvorschlag mit den Erfüllungsgraden für die real gebaute Anlage verglichen. Der Vergleich zeigt, dass der Konstruktionsvorschlag die Originalanlage in Bezug auf die Erfüllungsgrade der Constraints dominiert. Die Belegungdichten der Etagen unterscheiden sich stark. Die meisten Ausrüstungen wurden im Erdgeschoss plaziert, dagegen sind die erste, zweite und dritte Etage überhaupt nicht belegt. Dies könnte unter anderem daran liegen, dass die binären Constraints durch die vorgenommene Gewichtung schlechter erfüllt wurden als die unären Constraints. Deshalb wurden in einem zweiten Versuch alle binären Constraints mit einem Gewicht von 1000 versehen. Der resultierende Platzierungsvorschlag ist in Abbildung 4 in 2D und in Abbildung 3 in 3D Darstellung zu sehen. Die Erfüllungsgrade haben sich zugunsten der binären Constraints verschoben. Die Etagen sind nun gleichmäßiger ausgelastet. Auch diese Variante dominiert die originale Aufstellung in Bezug auf die Erfüllungsgrade der Constraints. Nun wurde versucht die Rastergröße zu verringern, allerdings musste schon bei einer Rastergröße von 10 cm der Konstruktionversuch abgebrochen werden, da selbst nach 3 Tagen noch keine Aufstellung gefunden wurde.



(a) Ebene 0

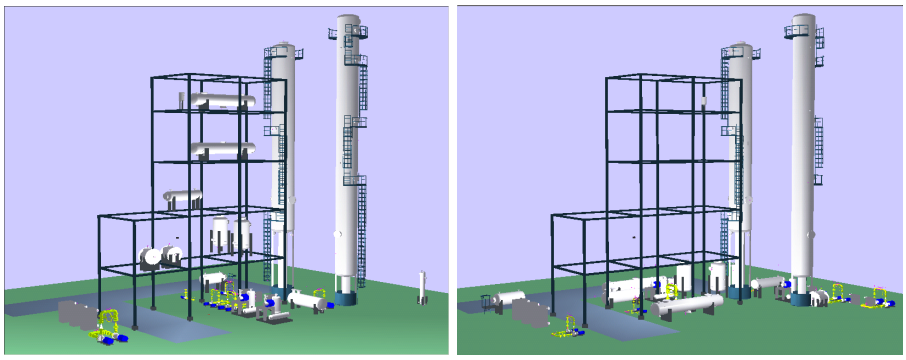
(b) Ebene 1

(c) Ebene 2

(d) Ebene 3

(e) Ebene 4

Abbildung 2: Zweidimensionale Darstellung der Ebenen der Variante 1.

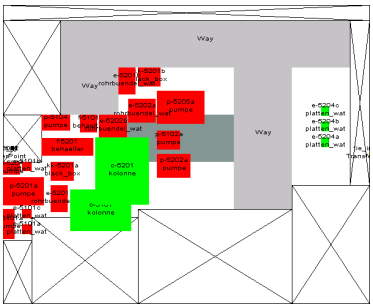


(a) Anlage original

(b) Variante 1

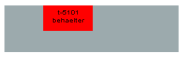
(c) Variante 2

Abbildung 3: Dreidimensionale Darstellung der tatsächlichen Aufstellung der Anlage und der zwei mit CP erzeugten Aufstellungen.



(a) Ebene 0

(b) Ebene 1



(c) Ebene 2

(d) Ebene 3

(e) Ebene 4

Abbildung 4: Zweidimensionale Darstellung der Ebenen der Variante 2.

4.4 Fazit

Die Rastergröße kann aufgrund der Komplexität des CSP nicht beliebig gewählt werden. Die Komplexität des CSP ist exponentiell in der Anzahl n der Variablen und linear in der Anzahl m der Constraints. Nimmt man an, dass alle Variablen die selbe Domänengröße L haben, beträgt die Komplexität $\mathcal{O}(L^n \times m)$. Bereits bei einer Rastergröße von 10 cm wird keine Aufstellung in akzeptabler Zeit gefunden. Es müsste untersucht werden, inwieweit Techniken zur Dekomposition eines CSP [9, 8, 3] die Rechenzeit minimieren.

Die Rohrleitungslänge ist im Augenblick nicht unmittelbar Gegenstand der Optimierung. Die Längen einiger Rohrleitungen werden aber indirekt durch die Erfüllung binärer Constraints minimiert. Es wäre jedoch auch denkbar die Verbindungskosten aller Rohrleitungen zu minimieren, indem mögliche Positionen für Komponenten zusätzlich zum Erfüllungsgrad der Constraints auch nach der Auswirkung auf die Rohrlänge bewertet werden.

5 Berechnung einer Aufstellung mit Hilfe von Hornlogik und Backtracking

Der hier verfolgte Weg zur Berechnung einer möglichen Aufstellung einer Chemieanlage ist ein klassischer Ansatz, im Gegensatz zu den heute vielfach verwendeten randomisierten oder evolutionären Algorithmen (Simulated Annealing, genetische Algorithmen) zur Lösung von Problemen dieses Typs (siehe z. B. [10, 13]). Bei dem vorgestellten Verfahren wird zunächst versucht, die Menge in Frage kommender Standorte der Anlagen-Komponenten mit Hilfe eines Inferenz-Prozesses auf Basis von Horn-Logik möglichst stark zu beschränken, um dann in einem konventionellen Backtracking-Verfahren eine vollständige Aufstellung zu berechnen. Zur Demonstration der Funktionsweise und experimentellen Untersuchungen wurde das Verfahren dazu in Java implementiert.

5.1 Formalisierung

Es ist zunächst Ziel dieses Ansatzes, für jede Komponente die Menge der in Frage kommenden Plätze zu verkleinern. Es wird also mit jeder Komponente eine Menge assoziiert, in der alle Plätze enthalten sind, an denen die Komponente platziert werden kann. Es muss dabei aber unterschieden werden zwischen Anforderungen, die eine Komponente an die Eigenschaften ihres Standortes stellt (unäre Anforderungen) und Anforderungen, die eine Komponente an Komponenten in ihrer Nachbarschaft stellt (binäre Anforderungen).

Es sei L die Menge der Orte, an denen eine Komponente platziert werden kann und K die Menge der zu platzierenden Komponenten. Dann sei A_k^u die Menge der unären Anforderungen und A_{k,k_2}^b die Menge der binären Anforderungen einer Komponente k. Jede Anforderung $a_k^u \in A_k^u$ ist eine Funktion

$$a_k^u : L \mapsto \{0, 1\}$$

welche 1 wird, falls die entsprechende Anforderung am Ort $l \in L$ erfüllt ist, sonst 0. Eine solche unäre Anforderung könnte zum Beispiel sein, dass eine Komponente an einem Ort stehen muss, der im Erdgeschoss des Stahlbaus liegt.

Demgegenüber beziehen sich die binären Anforderungen immer auf ein Paar von Komponenten. Es sei also $a^b_{k,k_2} \in A^b_k$ als eine Funktion

$$a^b_{k,k_2} : L \times L \mapsto \{0,1\}$$

definiert. Eine solche Anforderung wäre zum Beispiel, dass die Komponente k neben der Komponente k_2 stehen muss.

Für eine Komponente k sei nun L_k die Menge der Orte, an denen die Anforderungen der Komponente an ihren Aufstellungsplatz, also die unären Anforderungen, erfüllt sind, definiert als

$$L_k = \{l \in L | \forall a^u_k \in A^u_k : a^u_k(l) = 1\}$$

Die gesuchte Lösung – eine Aufstellung der Komponenten – kann nun dargestellt werden als eine Funktion $P : K \mapsto L$, die jeder Komponente genau einen Ort zuordnet. Dabei muss gelten:

- Die unären Anforderungen jeder Komponente müssen erfüllt sein:

$$\forall k \in K : \forall a^u_k \in A^u_k : a^u_k(P(k)) = 1$$

- Die binären Anforderungen jeder Komponente müssen erfüllt sein:

$$\forall k, k_2 \in K, k \neq k_2 : \forall a^b_{k,k_2} \in A^b_{k,k_2} : a^b_{k,k_2}(P(k), P(k_2)) = 1$$

5.2 Verlauf der Berechnung

Die Berechnung einer gültigen Platzierung P verläuft in mehreren aufeinanderfolgenden Schritten. Im ersten Schritt wird für jede Komponente k die Menge L_k der Orte bestimmt, an denen die unären Anforderungen A^u_k erfüllt sind. Dieser Schritt wird auf Basis eines hornlogischen Modells in einer Inferenz-Maschine durchgeführt.

Nun werden sowohl die Orte als auch die Komponenten gerastert, d.h. in kleinere Einheiten unterteilt, um eine exaktere Positionierung und platzsparendere Anordnung erreichen zu können. Die Größe des Rasters der Eingabedaten entspricht der durch den Stahlbau vorgegebenen Größe. Das nun aufgebaute Raster ist um einen konstanten Faktor r feiner aufgeteilt. Der Faktor r wird mit den Eingabedaten angegeben. Zu beachten ist hierbei, dass der Berechnungsaufwand durch ein feineres Raster extrem ansteigt. Der Benutzer muss also abwägen zwischen einer schnellen Berechnung und einer exakteren Platzierung – und damit einer unter Umständen besseren Raumausnutzung – und der schnelleren Generierung einer Lösung.

Im darauffolgenden und letzten Schritt wird dann in einem konventionellen Backtracking-Verfahren eine Aufstellung berechnet, wobei nun nur noch die binären Anforderungen betrachtet werden. Diese können im ersten Schritt (Hornlogik) noch nicht berücksichtigt werden, da zu diesem Zeitpunkt noch keine (Teil-)Aufstellung bekannt ist und somit noch nicht entschieden werden kann, ob eine bestimmte relative Anordnung zweier Komponenten vorhanden ist oder nicht.

5.3 Hornlogik

Im ersten Teil der Berechnungen wird ein eingeschränktes hornlogisches Verfahren angewendet, um aus den Eingabedaten zunächst die zur Berechnung einer Platzierung benötigen Anforderungen der Komponenten an ihren Stellplatz und die Eigenschaften des Stahlbaus zu berechnen. Damit wird dann die Menge in Frage kommender Stellplätze einer Komponente eingeschränkt.

Bei den Eingabedaten für diesen ersten Teil der Berechnung handelt es sich um die Eingabe für eine Inferenz-Maschine. Die Daten bestehen zum einen aus der Beschreibung der realen Gegebenheiten der Anlage (Größe und Eigenschaften des Stahlbaus bzw. einzelner Stellplätze, zu platzierende Komponenten, deren technische Daten, etc.) und andererseits aus Regeln, die zur Ableitung der Anforderungen der Komponenten verwendet werden sollen (z. B. "Komponenten mit einer besonders teuren Rohrverbindung müssen nebeneinander stehen", "Komponenten, die gewartet oder gereinigt werden, müssen im Erdgeschoss und an einem Weg stehen", etc.). Die Regeln zur Ableitung der Anforderungen sind also nicht Teil des Platzierungs-Verfahrens, sondern Teil der Eingabedaten und können somit auf einfachem Weg erweitert bzw. an neue Gegebenheiten angepasst werden.

Bei der hier implementierten Inferenz-Maschine handelt es sich um eine in ihrer Funktion beschränkte Komponente, die aus einem Satz von xml-Dateien eine Menge von Fakten und eine Menge von Regeln in Horn-Klausel-Form

$$not(P_1) \vee \cdots \vee not(P_n) \vee not(Co_1) \vee \cdots \vee not(Co_n) \vee C$$

einliest und darauf Inferenz betreibt.

Um die Geschwindigkeit der Inferenz zu optimieren, arbeitet die Engine in zwei Schritten. Im ersten Schritt werden die Fakten gesucht, die den Prämissen P_i entsprechen, sodass rekursiv eine Tabelle aufgebaut werden kann, in der die Argumente der Fakten an die Variablen der Prämissen gebunden werden. Neben den Prämissen, die durch eingegebene oder bereits inferierte Fakten erfüllt werden, gibt es die `unknown`-Prämisse, die erfüllt ist, wenn kein Fakt des angegebenen Prädikates bekannt ist und die `count`-Prämisse, die die Anzahl von Fakten eines bestimmten Prädikates zählen und das Ergebnis an eine Variable binden kann.

Nun wird jede Variablenbelegung mit den in einer Regel enthaltenen Constraints C_i verglichen. Dabei handelt es sich um Beschränkungen der Variablenbelegungen. Es werden hier also keine weiteren Variablen gebunden, sondern nur noch geprüft, ob eine Variablenbelegung bestimmte Kriterien erfüllt. Dabei sind Vergleiche der Variablen wie $\geq$, $>$, $=$ aber auch die Berechnung von Summen und Differenzen zwischen Variablen möglich.

Sind alle Prämissen belegt und erfüllt die Variablenbelegung sämtliche Constraints, so kann die Konklusion C anhand der Variablenbelegung inferiert und der Faktmenge hinzugefügt werden. Die Engine ist aber in einigen Punkten gegenüber einer "ausgewachsenen" Engine, wie zum Beispiel der in Mobal verwendeten (siehe [16]), stark eingeschränkt.

Es gibt keine explizite Darstellung von negierten Fakten. (Schlüsselwort *not* in Mobal). Somit gibt es für einen Fakt nur die Möglichkeit, wahr zu sein (dann ist er in der Faktmenge enthalten) oder unbekannt zu sein (dann ist er in der Faktmenge nicht enthalten). Soll ein negierter Fakt eingegeben werden, kann dies durch Voranstellen eines Prefix im Prädikat-Namen o.ä. gekennzeichnet werden.

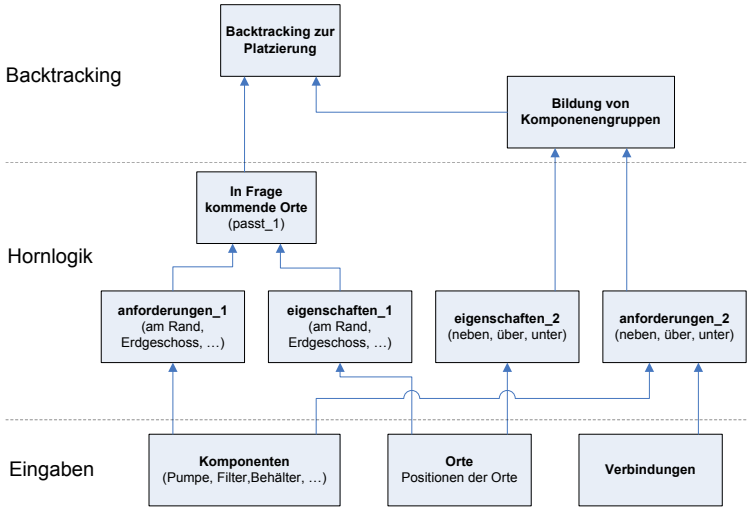


Abbildung 5: Inferenzpfad

Die Inferenzengine kann somit keine Widersprüche entdecken, die Eingabe eines positiven und negierten Faktes ist also möglich. Trotz dieser Einschränkungen reichen die Fähigkeiten der Maschine aus, um die für diese Problemstellungen nötigen Berechnungen durchzuführen.

5.3.1 Inferenzpfad

Damit eine Lösung, also eine Platzierung der Komponenten, gefunden werden kann, muss zunächst einmal aus den Eingabedaten eine Repräsentation der Eigenschaften des Stahlbaus und der Anforderungen der Komponenten abgeleitet werden. Dann wird für jede Koponente die Menge der Orte gebildet, an denen sämtliche Anforderungen der Komponente an den Standort erfüllt werden können. Der komplette Verlauf der Berechnung ist in Abbildung 5 dargestellt.

5.3.2 Beschreibung der Orte

Bei den Eigenschaften der Orte lässt sich unterscheiden zwischen jenen, die sich auf einen Ort beziehen (am Rand des Stahlbaus, an einem Weg, im Erdgeschoss, ...) und solchen, die sich auf eine Relation zwischen zwei Orten beziehen (neben, über). Die Eingabedaten setzen sich zusammen aus einer Beschreibung des Stahlbaus und den technischen Daten der Komponenten. Der Stahlbau wird durch die Position der Orte innerhalb und ausserhalb des Stahlbaus in Form von Koordinatenangaben `ortsposition/2: <ort>, <x>, <y>, <z>` und durch die Beschreibung der Position von Wegen und Rändern des Stahlbaus durch `eigenschaft_1/2: <ort>, <eigenschaft>` beschrieben. Bei `eigenschaft` kann es sich dabei also um die Werte

- `am_weg`
- `am_rand`

- `ausserhalb_stahlbau`
- `im_erdgeschoss`
- `oberste_etage`

handeln.

Aus den Positionsangaben von je zwei Stellplätzen können nun deren relationale Beziehungen `neben` oder `über` hergeleitet werden[1]. Zur Zeit ist `ueber` vorhanden, falls ein Ort direkt oberhalb eines anderen liegt, also sich nur die z-Koordinate der beiden Orte unterscheidet. `neben` ist erfüllt, wenn sich entweder die x- oder die y-Koordinaten der Orte um höchstens 1 unterscheiden, während die z-Koordinate gleich sein muss.

5.3.3 Beschreibung der Komponenten

Die zu platzierenden Komponenten werden durch einen eindeutigen Bezeichner, ihre Art (z. B. Pumpe, Behälter, Filter, etc.) und ggf. durch weitere technische Daten beschrieben. Aus diesen Komponenten-Daten werden nun die Anforderungen der Komponenten abgeleitet. Dabei kann es sich sowohl um unäre als auch binäre Anforderungen handeln. Die unären Anforderungen entsprechen den Werten, die die Variable `eigenschaft` des Prädikats `eigenschaft_1` annehmen kann (s. o.). Die binären Anforderungen, die sich auf die relative Position zweier Komponenten beziehen, sind – ebenso wie bei den Orten – `neben` und `ueber`. Diese Anforderungen werden anhand der Art und der technischen Daten der Komponenten mithilfe der in den Eingabedaten enthaltenen eingegebenen Regeln abgeleitet.

5.3.4 Signatur des Logikmodells

Im folgenden werden wesentliche Prädikate des verwendeten logischen Modells aufgeführt und erläutert. Siehe hierzu auch Abbildung 5.

`art_komponente/1: <komponente>` Gibt an, dass ein Objekt eine Komponente darstellt.

`art_ort/1: <ort>` Gibt an, dass ein Objekt einen Ort darstellt.

`art_hilfsmittel/1: <hilfsmittel>` Beschreibung, dass ein Objekt ein Hilfsmittel darstellt.

`ortsposition/4: <ort>, <x>, <y>, <z>` Einem Ort (1. Argument) wird eine Position innerhalb des Stahlbau-Rasters in Form von Koordinaten (2., 3. und 4. Argument) zugewiesen.

`eigenschaft_1/2: <ort>, <eigenschaft>` Einem Ort wird eine Eigenschaft zugewiesen, die nur diesen einen Ort betrifft. Mögliche Eigenschaften sind zum Beispiel `am_rand` (am äußeren Rand des Stahlbaus) oder `neben_weg` (neben einem Weg durch den Stahlbau).

[1] Eine weitere Relation `oberhalb` könnte (z.B. für für die Anordnung von Behältern oberhalb von Pumpen) nützlich bzw. notwendig werden

`eigenschaft_2/3: <ort>, <ort>, <eigenschaft>` Zwei Orten wird eine relationale Eigenschaft zugewiesen. Mögliche Eigenschaften sind `neben`, `ueber`, `unter`.

`anforderung_1/2: <komponente>, <anforderung>` Mit diesem Prädikat wird festgelegt, dass eine Komponente eine bestimmte Anforderung an einen Ort hat. Die möglichen Anforderungen entsprechen den Eigenschaften, die mit `eigenschaft_1` verwendet werden.

`anforderung_2/3: <komponente>, <komponente>, <anforderung>` Es wird festgelegt, dass zwischen zwei Komponenten eine „ verbindende“, also binäre Anforderung besteht. Mögliche Anforderungen sind die unter `eigenschaft_2` aufgeführten.

`passt_1/3: <komponente>, <ort>, <anforderung>` An einem Ort ist eine Anforderung einer Komponente erfüllt. Das bedeutet aber nicht, dass die Komponente hier platziert werden kann, da andere Anforderungen unerfüllt sein könnten.

`anforderungen_1_erfuellt/2: <komponente>, <ort>` Es sind alle Anforderungen einer Komponente an diesen Ort erfüllt. Die Komponente kann also bezüglich der mit `anforderung_1` angegebenen Anforderungen an diesem Ort platziert werden. Binäre Anforderungen zu anderen Komponenten, die durch `anforderung_2` angegeben sind, werden hier nicht berücksichtigt.

`platziert/2: <komponente>, <ort>` Eine Komponente ist einem Ort zugeordnet worden. Dieses Prädikat dient dazu, eine Anlage vollständig zu beschreiben, um zum Beispiel mithilfe eines maschinellen Lernverfahrens aus den gültigen Platzierungen der Komponenten neue Erkenntnisse und Inforationen über eine effizientere Aufstellungs-Berechnung zu erlangen. Es wird zur Zeit noch nicht verwendet.

5.4 Rasterung

Bei der Rasterung handelt es sich um eine Aufteilung der Komponenten und Orte in mehrere quadratische Subkomponenten und Suborte. Da die Größe des verwendeten Rasters ein durch den Faktor r festgeleter Bruchteil der Größe des Rasters des Stahlbaus ist, können Orte ohne Rundungsfehler gerastert werden. Die Komponenten werden ebenso gerastert, wobei es vorkommen kann, dass eine Komponente nur Teile eines Rasters belegt. Dann wird entsprechend aufgerundet, damit jede Komponente nur ganze Rasterflächen belegt.

Um die Steigerung der Komplexität des Platzierungs-Verfahrens durch das Rastern möglichst wenig zu steigern, werden die Eigenschaften der Orte einfach auf die gerasterten Suborte übertragen. Das bedeutet zum Beispiel, dass wenn ein Ort die Eigenschaft hat, am Rand des Stahlbaus zu liegen, dass dann auch alle aus diesem Ort hervorgegangenen Suborte diese Eigenschaft übernehmen, auch wenn zwischen ihnen und dem Rand des Stahlbaus nun noch andere Suborte liegen können.

5.5 Backtracking

Bei dem nun folgenden Backtracking wird die eigentliche Platzierungs-Arbeit verrichtet. Dazu werden die Komponenten entsprechend der Anzahl der binären Anforderungen an benachbarte Komponenten in absteigender Reihenfolge sortiert. Damit soll erreicht werden, dass Komponenten mit vielen Anforderungen früh platziert werden, in der Hoffnung, dass frühzeitig Konflikte auftreten und damit große Teile des Suchraums nicht durchsucht werden müssen. Falls es aber möglich ist, die Komponenten mit vielen Anforderungen zu platzieren, können zum Schluss die Komponenten mit wenigen Anforderungen leichter platziert und damit zu einer vollständigen Platzierung zusammengesetzt werden.

Zunächst wurde in diesem Schritt versucht, erst alle gültigen Platzierungen für jede durch die binären Anforderungen definierten Zusammenhangskomponenten der Komponenten einzeln zu berechnen, um dann in einem weiteren Schritt diese Zusammenhangskomponenten zu einer Gesamtplatzierung zusammenzusetzen. Dabei wurde jedoch kein Geschwindigkeitsvorteil erzielt und größere Anlagen konnten aufgrund des hohen Bedarfs an Hauptspeicher nicht berechnet werden.

Während des Backtrackings werden nur die binären Anforderungen berücksichtigt. Die unären Anforderungen werden dadurch automatisch erfüllt, dass jede Komponente mit der Menge der für die Komponente k in Frage kommenden Orte L_k assoziiert ist und nur auf diesen Orten eine Platzierung versucht wird. Bei der testweisen Platzierung einer Komponente werden nun jeweils die folgenden Punkte geprüft.

1. Ist der vorgesehene Ort für die Komponente noch nicht durch eine andere Komponente belegt?
2. Ragen Teile der Komponente aus dem Anlagen-Bereich heraus oder sind diese Orte bereits belegt?
3. Wird von der Komponente bei der Platzierung keine Stütze des Stahlbaus überdeckt?
4. Sind in der Nachbarschaft des vorgesehenen Ortes noch genug nicht-belegte Orte, um die noch nicht erfüllten Nachbarschaftsanforderungen zu erfüllen?
5. Sind alle binären Anforderung zwischen der aktuellen und schon platzierten Komponenten erfüllt?

Sobald einer der Punkte nicht erfüllt werden kann, wird die Platzierung dieser Komponente aufgegeben und die Platzierung der im vorhergehenden Rekursionsschritt betrachteten Komponente fortgesetzt.

5.6 Ergebnisse

Bei den bisher durchgeführten Tests wurde offensichtlich, dass das Verfahren zur Zeit noch nicht in der Lage ist, Anlagen realistischer Größe zu platzieren. Daher wurde für alle weiteren Versuche die in Abbildung 6 dargestellte Anlage verwendet. Da es sich um keine real existierende Anlage handelt, sind die einzelnen Komponenten nicht weiter spezifiziert; die dargestellten Anforderungen (Doppelpfeile symbolisieren binäre Anforderungen) sind explizit vorgegeben worden.

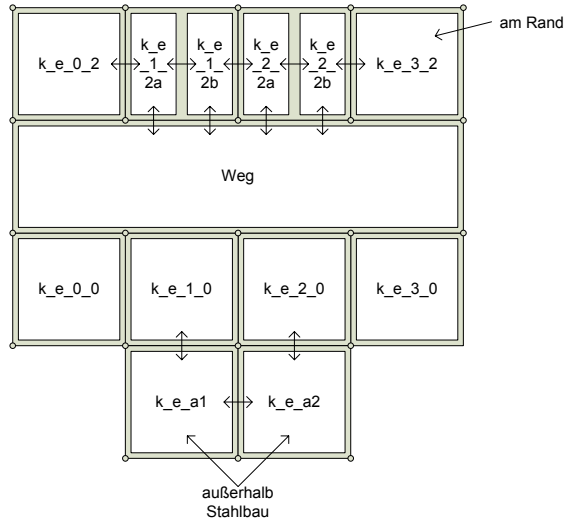


Abbildung 6: Beispiel-Anlage für die Durchführung erster Tests

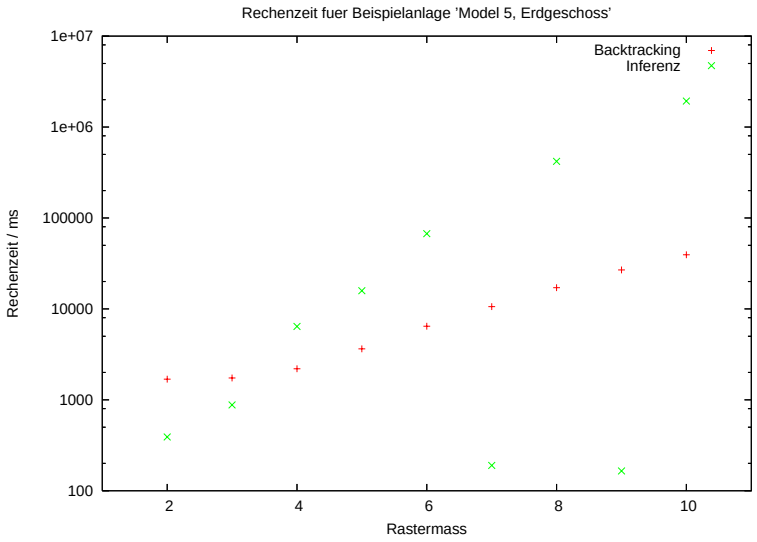


Abbildung 7: Diagramm Rechenzeit

Mit der zur Zeit aktuellen Implementierung des Verfahrens ergeben sich die in dem Diagramm in Abbildung 7 dargestellten Rechenzeiten, um eine erschöpfende Suche aller möglichen Aufstellungen für diese Anlage durchzuführen, aufgespalten in die Zeit für die Inferenz und die Zeit für das Backtracking. Der exponentielle Zusammenhang zwischen Rastergröße und Rechenzeit ist deutlich erkennbar. Die Abweichungen von dieser Tendenz bei den Rastermaßen 7 und 9 sind dadurch zu erklären, dass aufgrund der Aufrundung der Abmessungen der Komponenten nach der Rasterung keine Platzierung mehr möglich ist. Dies kann relativ schnell festgestellt werden. Bei allen anderen Rastermaßen waren 180 verschiedene Aufstellungen möglich. Die Anzahl möglicher Aufstellungen ist bei dieser Anlage nicht vom Rastermaß abhängig, da in der Beispielanlage alle Flächen von Komponenten belegt sind und damit durch ein feineres Raster kein Raum für alternative Aufstellungen gewonnen werden kann.

5.7 Fazit

Die Menge der in Frage kommenden Standorte einer Komponente kann bisher nicht weit genug eingeschränkt werden, um eine akzeptable Rechenzeit zu erreichen. In den verwendeten Beispiel-Anlagen gibt es wesentlich mehr binäre als unäre Anforderungen, deren vielfache Evaluierung während des Backtracking zu viel Rechenzeit kostet.

Eine Verringerung der Rechenzeit wäre unter Umständen auch möglich, indem zunächst auf Basis eines relativ groben Rasters eine Aufstellung berechnet wird, die dann in einzelnen, durch eine geeignete Heuristik zu findenden Bereichen, mit Hilfe eines feineren Rasters weiter verbessert wird.

Bisher ist es mit diesem Ansatz nicht möglich, einzelne Anforderungen zu gewichten. Es werden bei diesem Verfahren zur Zeit immer sämtliche Anforderungen berücksichtigt. Sollten Widersprüche in den Anforderungen enthalten sein, (z. B. eine Komponente muss im Erdgeschoß und in der 2. Etage stehen), so kann keine Lösung gefunden werden.

Es ist somit nicht möglich, Platzierungen zu berechnen, die nicht alle Anforderungen bzw. manche Anforderungen nur zum Teil erfüllen. Ansonsten wäre es möglich, dem Benutzer zunächst teilweise korrekte Platzierungen zu präsentieren und ihn entscheiden zu lassen, welche Qualität der Platzierungen er erreichen möchte bzw. ob er weitere, ggf. bessere Platzierungsvorschläge berechnen möchte. Ebenso ist bei diesem Verfahren bisher keine automatische Bewertung von Platzierungen vorgesehen. Eine solche Bewertung könnte einerseits zur Unterstützung des Benutzers bei der Auswahl einer günstigen Platzierung verwendet werden. Ebenso wäre es aber möglich, bereits während des Backtrackings einen Teil des Suchraums auszuschliessen, der eine durch den Benutzer vorzugebende Mindestbewertung unterschreiten würde.

In diesem Bereich wäre es unter Umständen interessant, anstelle einer booleschen Logik eine mehrwertige Logik einzusetzen, und so im gesamten Berechnungsverlauf unscharfe bzw. teilweise Erfüllung der Anforderungen zuzulassen und zu verarbeiten. So könnten zum Beispiel Anforderungen gewichtet werden, und in der Berechnung zunächst darauf hingearbeitet werden, nur besonders wichtige Anforderungen zu erfüllen oder es könnte eine Teilerfüllung einzelner Anforderungen zugelassen werden.

6 Vergleich

Die beiden vorgestellten Ansätze zur Aufstellungsplanung von Chemieanlagen wurden unabhängig voneinander entwickelt. Interessant ist, dass beide Ansätze ähnliche Konzepte zur Berechnung einer Platzierung verwenden. Beide Ansätze setzen die Platzierungsregeln in Form von unären und binären Constraints um. Die unären Constraints dienen bei beiden Ansätzen zur Auswahl der in Frage kommenden Orte eines Equipments.

Unterschiede zwischen den beiden Ansätzen bestehen zum einen in der Berücksichtigung der Regelgewichtung. Der CSP Ansatz berücksichtigt die Regelgewichtung durch unscharfe Constrainterfüllung. Außerdem werden beim CSP Ansatz in jedem Rekursionschritt zunächst die vielversprechensten Orte zur Platzierung getestet. Demgegenüber wird beim logikbasierten Ansatz nur zu Beginn der Berechnung eine Sortierung der Orte vorgenommen. Dies führt bei diesem Ansatz zu einer großen Rekursionstiefe, da Konflikte oft erst spät erkannt werden.

7 Zusammenfassung und Ausblick

Für die Aufstellungsplanung von Chemieanlagen wurden zwei Ansätze vorgestellt. Der erste Ansatz beruht auf Techniken des Constraint Programming, der zweite verwendet Hornlogik und ein konventionelles Backtracking. Es zeigte sich, dass bei dem zweiten Ansatz die Berechnungskomplexität bisher nicht weit genug reduziert werden kann, um in akzeptabler Zeit Anlagen realistischer Größe zu platzieren. Demgegenüber erwies sich die Relaxierung des Problems durch unscharfe Auswertung der Anforderungen als hilfreich, um die Berechnung einer Aufstellung zu beschleunigen. Es ist bisher nicht absehbar, ob ein Verfahren dem anderen überlegen ist.

Ein Vergleich der beiden Methoden mit dem bisher verwendeten Simulated Annealing Algorithmus steht noch aus. Eine genaue Inspizierung und Bewertung der erzeugten Aufstellungen durch Chemietechniker und Anlagenplaner könnte Hinweise auf weitere Verbesserungsmöglichkeiten geben. Für weiterführende Auswertungen bzw. Beurteilungen sind Experimente mit weiteren Anlagen erforderlich.

8 Danksagungen

Diese Arbeit wurde unterstützt durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen des Sonderforschungsbereichs 531 „Design und Management komplexer technischer Prozesse und Systeme mit Methoden der Computational Intelligence“.

Literatur

[1] Fahiem Bacchus and Paul van Run. Dynamic variable ordering in CSPs. In Ugo Montanari and Francesca Rossi, editors, *Proceedings First International Conference on Constraint Programming*, pages 258–275. Springer, 1995.

[2] Christian Bessiere and Jean-Charles Regin. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. In *Proceedings of the 2nd Int. Conf. on Principles and Practice of Constraint Programming*, pages 61–75, 1996.

[3] Rina Dechter and Judea Pearl. Tree clustering for constraint networks (research note). *Artif. Intell.*, 38(3):353–366, 1989.

[4] D. Dubois, H. Fargier, and H. Prade. The calculus of fuzzy restrictions as a basis of flexible constraint satisfaction. In *Proc. IEEE International Conference on Fuzzy Systems*, pages 1131–1136, 1993.

[5] D. Dubois, H. Fargier, and H. Prade. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 6(4):287–309, 1996.

[6] Eugene C. Freuder. Partial constraint satisfaction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89, Detroit, Michigan, USA*, pages 278–283, 1989.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, 1979.

[8] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural csp decomposition methods. *Artif. Intell.*, 124(2):243–282, 2000.

[9] Marc Gyssens, Peter G. Jeavons, and David A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artif. Intell.*, 66(1):57–89, 1994.

[10] Wee Sng Khoo, P. Saratchandran, and N. Sundararajan. *Constrained two dimensional bin packing using a genetic algorithm.* Physica-Verlag GmbH, 2003.

[11] A. Kuziak and S. Heragu. The facility layout problem. *Europ. J. of Operational Research*, 29:229–251, 1987.

[12] Peter Leuders. *Rechnergestützte Optimierung der Layoutplanung von Chemieanlagen.* Shaker Verlag, 2002.

[13] Thelma D. Mavridou and Panos M. Pardalos. Simulated annealing and genetic algorithms for the facility layout problem: A survey. *Comput. Optim. Appl.*, 7(1):111–126, 1997.

[14] R. D. Meller and K.-Y. Gau. The facility layout problem: Recent trends and perspectives. *Journal of Manafacturing Systems*, 15(5):351–366, 1996.

[15] P. Meseguer, N. Bouhmala, T. Bouzoubaa, M. Irgens, and M. Sanchez. Current Approaches for Solving Over-Constrained Problems. *Constraints*, 8(1):9–39, 2003.

[16] K. Morik, S. Wrobel, J-U. Kietz, and W. Emde. *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications.* Academic Press, 1993.

[17] Volker Schnecke and Oliver Vornberger. Hybrid Genetic Algorithms for Constrained Placement Problems. *IEEE Transactions on Evolutionary Computation*, 1(4):266–277, 1997.