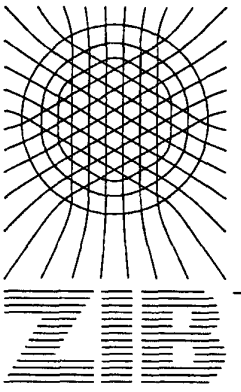
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Straße 10, D-1000 Berlin 31



Robert Weismantel

Plazieren von Zellen: Theorie und Lösung eines quadratischen 0/1-Optimierungsproblems

Technical Report TR 92-3 (Juli 1992)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Heilbronner Str. 10
1000 Berlin 31
Verantwortlich: Dr. Klaus André
Umschlagsatz und Druck: Rabe KG Buch- und Offsetdruck Berlin
D83
ISSN 0933-789X

meinen Eltern
gewidmet

Vorwort

Diese Dissertation wurde an der Universität Augsburg begonnen und am Konrad-Zuse-Zentrum für Informationstechnik Berlin fertiggestellt. Mein ganz besonderer Dank gilt dabei Prof. Dr. Martin Grötschel, der mich das Gebiet der polyedrischen Kombinatorik lehrte und mir stets mit Rat und Tat zur Seite stand. Bedanken möchte ich mich auch ganz herzlich bei Herrn Alexander Martin, der bei der Entwicklung des Verfahrens unentbehrlich war. Weiterer Dank gilt Herrn Carlos Ferreira, der mich bei bei den Untersuchungen des Clusteringpolyeders unterstützt hat. Dank gilt ebenfalls den Herren Professoren Dr. Michael Jünger und Dr. Gerhard Reinelt, da sie mit einem ersten Entwurf zur Lösung des Plazierungsproblems die Grundlagen für diese Arbeit gelegt haben. Für die wertvollen Anregungen möchte ich noch Herrn Professor Dr. Bixby danken. Ferner danke ich meinem Vater, Gundolf Weismantel, für die Korrektur dieser Arbeit. Mein Dank richtet sich auch an die Siemens AG, München Neu-Perlach und insbesondere an die Abteilung von Herrn Dr. Grüter für die finanzielle Zuwendung und die Testdaten. Schließlich möchte ich noch meiner Frau Simone für ihr Verständnis und die Unterstützung danken.

Beim Konrad-Zuse-Zentrum bedanke ich mich für den Druck dieser Dissertation.

Berlin, April 1992 Robert Weismantel

Inhaltsverzeichnis

Einleitung

Teil I Problemstellung und Hintergrund

Kapitel 1 Das Plazierungsproblem beim Entwurf elektronischer Schaltungen . . 2
1.1 Der Entwurf elektronischer Schaltungen 2
1.2 Das Plazierungsproblem 7
Kapitel 2 Plazierungsansätze 9
2.1 Schwierigkeiten bei der Modellierung des Plazierungsproblems 9
2.2 Die Min-Cut Heuristik 10
2.3 Kräftemodelle 11
2.4 Simulated Annealing 13
Kapitel 3 Das quadratische 0/1 Modell (P) 15
3.1 Das Modell (P) 16
3.2 Eine graphentheoretische Formulierung von (P) 23
3.3 Quadratische 0/1-Optimierung 24
Kapitel 4 Komplexitätsaussagen bezüglich (P) 31
4.1 Das Problem (P) und seine Varianten 31
4.2 Einige $\mathcal{NP}$-Vollständigkeitsresultate 32
4.3 Über die Existenz von polynomialen, ϵ-approximativen Algorithmen . . 34

Teil II Heuristische Lösung des Plazierungsproblems

Kapitel 5 Dekomposition des quadratischen 0/1 Modells 44
5.1 Ein Dekompositionsverfahren mit 16 Positionen für jede Zelle 46
5.2 Ein Dekompositionsverfahren mit 9 Positionen für jede Zelle 50
5.3 Dynamische Dekomposition 58
Kapitel 6 Heuristische Lösung der Probleme (P_m^+) 63
6.1 Heuristische Grundideen 63
6.2 Datenstrukturen 66

6.3 Implementierung der Heuristiken 71

Kapitel 7 Tests zur Parametereinstellung 79

7.1 Vergleich der Heuristiken 79
7.2 Einstellung des Strafparameters für die Überlappungen 82
7.3 Berücksichtigung der Verdrahtbarkeitsbedingung 83
7.4 Vergleich der Dekompositionsverfahren 89

Kapitel 8 Vergleich des Plazierungsverfahrens 95

Kapitel 9 Lösung einer linearen Relaxierung von (P) 100

Teil III Das Clusteringproblem

Kapitel 10 Das Clustering Problem und seine Varianten 112

10.1 Motivation 112
10.2 Problemstellung 113
10.3 Modellierung des r-Clusteringproblems 115
10.4 Modellierung des Kardinalitäts-Clusteringproblems 116

Kapitel 11 Heuristische Lösung des Kardinalitäts-Clusteringproblems 118

Kapitel 12 Das r-Clusteringpolytop 127

Kapitel 13 Grundlegende Facetten des r-Clusteringpolytops 131

Kapitel 14 Einige allgemeine Resultate 135

Kapitel 15 Kombinierte Knoten-Kanten-Facetten 142

15.1 Ungleichungen aus einer Überdeckung und disjunkten Sternen . . . 143
15.2 Ungleichungen aus einer Überdeckung und einem Baum 146
15.3 Ungleichungen aus einer $(1, d)$-Konfiguration und einem Stern . . . 148
15.4 Ungleichungen aus zwei Überdeckungen und einem Baum 152
15.5 Ungleichungen aus einer Überdeckung und einer Kante 156
15.6 Eine kleine Klasse von kombinierten Knoten-Kanten-Ungleichungen . 157

Ausblick 165

Anhang I

Testserie zu Kapitelabschnitt 7.1 166

Anhang II

Grundlagen 213

Literaturverzeichnis 215

Index 220

Einleitung

Die vorliegende Arbeit beschäftigt sich mit der Lösung des sogenannten Plazierungsproblems, welches beim Entwurf elektronischer Schaltungen auftritt. Wir skizzieren zunächst kurz die Probleme, die beim Entwurf elektronischer Schaltungen entstehen, und gehen dann auf die eigentliche Fragestellung ein.

Der enorm zunehmenden Leistungsfähigkeit elektronischer Schaltungen ist ein wesentlicher Teil des technologischen Fortschritts zu verdanken. So ermöglichte die Entwicklung auf diesem Gebiet beispielsweise, daß Computer inzwischen mehrere Millionen Instruktionen pro Sekunde ausführen können.
Eine elektronische Schaltung (auch Chip genannt) stellt eine komplexe Verflechtung von Halbleiterbauelementen (sogenannten Transistoren) dar. Diese Verflechtung ist die physikalische Umsetzung einer logischen Funktion, die sich aus vielen elementaren Operationen zusammensetzt. Elektronische Schaltungen der heutigen Generation integrieren mehrere Millionen Transistoren auf wenigen Quadratzentimetern. Aus diesem Grund spricht man auch von hoch integrierten Schaltungen (Very Large Scale Integration).
Der Entwurf elektronischer Schaltungen läßt sich in zwei Teile gliedern. Der erste Teil ist der sogenannte logische Entwurf. Wie bereits oben erwähnt, stellt eine elektronische Schaltung eine logische Funktion dar, die aus vielen elementaren Operationen besteht. In der Praxis haben Chip-Designer Programmbibliotheken aufgebaut, in denen mehrere elementare logische Operationen zu einer logischen Funktionseinheit (einer sogenannten Zelle) zusammengefasst werden. Im logischen Entwurf wird einerseits festgelegt, welche Zellen aus der entsprechenden Programmbibliothek für die konkrete Schaltung verwendet werden. Andererseits wird definiert, welche Zellen an welchen Kontaktpunkten miteinander verbunden werden müssen (Kontaktpunkte werden auch Pins genannt, eine Menge von zu verbindenden Pins wird als Netz bezeichnet). Zellen, Netze und Pins bilden die Eingabe für den zweiten Teil des Entwurfs (physikalischer Entwurf). Die Aufgabenstellung des physikalischen Entwurfs, und nur mit diesem werden wir uns im weiteren beschäftigen, läßt sich als ein kombinatorisches Optimierungsproblem formulieren:

> *Gegeben ist eine ebene Grundfläche, eine Menge von Netzen sowie eine Menge von Zellen. Jede Zelle ist charakterisiert durch ihre Breite, Höhe und Pins.*
> *Gesucht ist eine Anordnung der Zellen auf der Grundfläche. Desweiteren soll festgelegt werden, wie die Netze physikalisch auf dem Chip verlaufen. Dabei sind bestimmte Nebenbedingungen und Optimalitätskriterien zu berücksichtigen.*

Nebenbedingungen und Optimalitätskriterien hängen von der zugrundeliegenden Technologie ab. So dürfen beispielsweise Zellen nur gewissen a-priori definierten Einbauplätzen zugeordnet werden, oder es müssen bestimmte Zellen gewisse Mindestabstände einhalten. Andere Nebenbedingungen fordern, daß nur solche Lösungen zulässig sind, in denen manche Netze eine bestimmte Länge nicht über- oder unterschreiten und dergleichen mehr. Die

zu optimierende Zielfunktion für das Entwurfsproblem berücksichtigt beispielsweise die Gesamtfläche der Schaltung, oder aber es ist das Ziel, die Gesamtlänge der Verbindungen zu minimieren.

Selbst einfache Versionen des Entwurfsproblems sind $\mathcal{NP}$-schwer [GJ79]. Zusätzlich liegt die Größenordnung praktischer Entwurfsprobleme im Bereich mehrerer tausend Zellen und Netze. Von daher scheint es unmöglich, praxisrelevante Varianten dieses Problems in naher Zukunft algorithmisch optimal zu lösen.

Da selbst die heuristische Behandlungen des Gesamtproblems extrem schwierig ist [M90], behilft man sich im allgemeinen damit, das Entwurfsproblem in Unterprobleme aufzuteilen. Das üblicherweise erste Unterproblem ist bei dieser Vorgehensweise das Plazieren der Zellen. Dieses läßt sich folgendermaßen formulieren:

> *Finde eine Anordnung der Zellen auf der ebenen Grundfläche, so daß je zwei Zellen nicht überlappen, die technologischen Nebenbedingungen erfüllt sind und alle Netze physikalisch verbunden werden können.*

Wie bereits bei der Definition des Entwurfsproblems erwähnt, hängen die Optimalitätskriterien von der zugrundeliegenden Technologie ab. So ist beispielsweise im sogenannten "sea of cells"-Entwurfsstil vorgegeben, daß die linke untere Ecke einer Zelle nur mit bestimmten Punkten der Grundfläche zur Deckung gebracht werden darf.

Nach der Durchführung des Plazierungsschritts wird das Verdrahtungsproblem gelöst. Dieses besteht darin, festzulegen, wie die Netze physikalisch auf dem Chip verlaufen sollen. Eine Lösung dieses Problems wird dabei wiederum durch die zugrundeliegende Technologie beeinflußt. Bei einigen Technologien bildet schließlich die Kompaktierung das dritte Problem. In diesem Schritt wird versucht, die Fläche der Schaltung zu reduzieren.

Die Komplexität des Plazierungsproblems und die Größenordnung praktischer Anwendungen motivieren die Entwicklung von Dekompositionsmethoden zu seiner Lösung. Im allgemeinen wird versucht, das Problem in eine Folge von Problemen geringerer Größenordnung zu unterteilen. Die durch Dekomposition enstandenen Probleme werden dann meistens in einer speziellen Reihenfolge gelöst und zu einer Lösung des ursprünglichen Problems zusammengefügt. Ein Ansatz in diesem Zusammenhang ist die Lösung eines sogenannten Clusteringproblems. Letzteres läßt sich folgendermaßen formulieren:

> *Gegeben ist ein Graph, dessen Knoten die Zellen und dessen Kanten die Verbindungen zwischen den Zellen repräsentieren. Ferner sind Knotengewichte gegeben, die beispielsweise der Fläche der zugehörigen Zelle entsprechen. Gesucht ist eine Aufteilung der Knotenmenge, so daß die Summe der Knotengewichte für jedes Element der Aufteilung eine gegebene Schranke nicht überschreitet und die Summe der Kanten, deren Endknoten verschiedenenen Elementen der Aufteilung zugeordnet wurden, minimal ist.*

Anstelle der ursprünglich gegebenen Zellen betrachtet man nun die Elemente der Aufteilung als Zellen (sogenannte Superzellen), d.h. es werden Höhe, Breite und Kontaktpunkte festgelegt. In dem folgenden Schritt werden die Superzellen plaziert. Anschließend löst

man die Superzellen wieder in die ursprünglichen Zellen auf, was im allgemeinen zu einer nicht überlappungsfreien Plazierung der Zellen führt. Davon ausgehend versucht man, eine zulässige Plazierung zu finden.

Ziel dieser Dissertation ist es, im Rahmen des VLSI-Entwurfs einen neuen Ansatz zur Plazierung von Zellen zu entwickeln. Dabei soll insbesondere den drei folgenden Aspekten Rechnung getragen werden.

Zum einen wollen wir eine Modellierung des Plazierungsproblems vorstellen und analysieren. Das Modell soll möglichst robust gegenüber Änderungen technischer Bedingungen sein und sich auf verschiedene Entwurfsstile mit modifizierten Optimalitätskriterien anwenden lassen. In diesem Sinne – so hoffen wir – liefert die vorliegende Arbeit einen Beitrag zur Verwirklichung des in [L90, Seite 223] formulierten Ziels:

> "One of the major research goals in circuit layout is to improve the robustness of layout algorithms and technologies. The layout algorithm should be adaptable to a wide range of technological boundary conditions and allow us to incorporate several cost functions of varying degrees of importance."

Der zweite Aspekt bezieht sich auf die Lösung praktischer Probleme in einem speziellen Entwurfsstil ("sea of cells"). Zur Lösung dieser Probleme wurde ein Prototyp implementiert, welcher mit "state of the art"-Plazierungsverfahren konkurrieren kann.

Schließlich wollen wir uns mit dem Clusteringproblem und Varianten davon beschäftigen. Dabei steht einerseits im Vordergrund, wie diese Probleme heuristisch gelöst werden können und wie die Integration dieses Ansatzes in das Plazierungsprogramm erfolgt. Andererseits soll das Clusteringproblem aus polyedrischer Sicht untersucht werden.

Die Dissertation ist in drei Teile gegliedert und wie folgt aufgebaut.

Teil I. Ziel dieses Teils ist es, eine Modellierung für das Plazierungsproblem vorzustellen und zu analysieren. Teil I gliedert sich in 4 Kapitel.
In Kapitel 1 stellen wir die verschiedenen Entwurfsstile im VLSI-Design vor. Wir werden für diese Entwurfsstile das Plazierungsproblem und die unterschiedlichen Optimalitätskriterien diskutieren.
Anschließend fassen wir die wichtigsten Ansätze zur Lösung des Plazierungsproblems zusammen. In diesem Rahmen werden das Min-Cut-Verfahren, Kräftemodelle und Simulated Annealing behandelt.
Kapitel 3 ist der Modellierung des Plazierungsproblems für den "sea of cells"-Entwurfsstil gewidmet. Dabei soll ein quadratisches 0/1 Optimierungsproblem vorgestellt und hinsichtlich seiner Erweiterbarkeit auf andere Technologien kritisch erörtert werden. Das quadratische Modell (im folgenden mit (P) abgekürzt) und eine graphentheoretische Formulierung hiervon werden die weitere Grundlage für diese Dissertation sein. Einen kurzen Überblick über den Stand der Forschung für quadratische 0/1 Optimierungsprobleme bildet den Abschluß des Kapitels.

Kapitel 4 beschäftigt sich damit, das Plazierungsmodell komplexitätstheoretisch zu analysieren. Hierzu definieren wir zunächst für festes $k \in \mathbb{N}$ Probleme (P_k), (P_k^+) und (P_k^-). Diese Probleme sind wiederum quadratische 0/1 Optimierungsprobleme, die sich so interpretieren lassen, daß es für jede Zelle höchstens k Positionen gibt, auf denen sie plaziert werden kann. Die Notation (P_k^+) bzw. (P_k^-) wird verwendet, um zu kennzeichnen, daß die zugehörigen Matrixeinträge nicht negativ bzw. nicht positiv sind. Für festes $k \geq 2$ wird sich zeigen, daß die Optimierungsproblem (P_k) und (P_k^+) $\mathcal{NP}$-schwer sind. Es wird sich sogar herausstellen, daß keine ϵ-approximativen Algorithmen, welche polynomiale Laufzeit in der Codierungslänge der Eingabedaten haben, zur Lösung der Probleme (P_2) und (P_3^+) existieren. Dies impliziert, daß es auch keine solchen Algorithmen zur Lösung von (P), (P_k) und (P_k^+) für festes $k \geq 3$ gibt. Zur Lösung des Problems (P_m^-) werden wir einen $\frac{1}{m^2}$-approximativen Algorithmus vorstellen. Die Fälle (P_2^+) und (P^-) bleiben offen.

Teil II. Dieser Teil der Arbeit ist der Lösung des in Kapitel 3 eingeführten Plazierungsmodells gewidmet und enthält die Kapitel 5 bis 9.
Zur Lösung des Plazierungsproblem im "sea of cells"-Entwurfsstil werden wir uns in Kapitel 5 mit heuristischen Dekompositionsmethoden beschäftigen. Die Idee dabei ist, das ursprüngliche Problem (P) durch eine Folge von Problemen (P_m) zu ersetzen (für jede Zelle gibt es höchstens m mögliche Positionen, denen sie zugewiesen werden darf). Diesen Verfahren ist gemeinsam, daß Zellen über die Iterationen hinweg nach allen Richtungen wandern können. Abschließend wird ein dynamisches Verfahren zur Dekomposition des Plazierungsproblems behandelt.
In Kapitel 6 beschäftigen wir uns mit der Lösung der dekomponierten Probleme. In diesem Zusammenhang werden wir verschiedene Primal- und Verbesserungsheuristiken vorstellen. Insbesondere soll dabei auf die effiziente Implementierung inklusive der dazu entwickelten Datenstrukturen eingegangen werden.
Diese approximativen Verfahren sollen in dem anschließenden Kapitel im Bezug auf Qualität und Laufzeit an einer Testschaltung verglichen werden. Ferner werden hier die Auswirkungen verschiedener Parameter (Parameter zur Bestraffung von Überlappungen, Parameter zur Berücksichtigung der Verdrahtbarkeitsbedingung) auf die Qualität der Lösung untersucht. Diese Tests sollen es ermöglichen, eine automatische Einstellung der Heuristiken und Parameter für alle Iterationen und Problembeispiele zu finden.
In Kapitel 8 werden wir dann an Hand von 4 Testschaltungen einen implementierten Prototyp unseres Verfahrens mit zwei industriell eingesetzten Verfahren vergleichen.
Um untere Schranken für praktische Instanzen von (P) zu erhalten, lösen wir in Kapitel 9 eine lineare Relaxierung des quadratischen Modells. Da die Anzahl der Variablen im Bereich mehrerer Millionen liegt, greifen wir auf eine Lagrange-Relaxierung zurück.

Teil III. Im letzten Teil der Dissertation beschäftigen wir uns mit dem Clusteringproblem bzw. Varianten davon. Dieser Teil gliedert sich in 6 Kapitel.
Zunächst werden wir in Kapitel 10 das Problem motivieren, definieren und zwei lineare, ganzzahlige Modellierungen für das sogenannte r-Clusteringproblem in Graphen und ei-

nen Spezialfall davon vorstellen. Zur Formulierung der Modelle werden wir Variablen benötigen, die angeben, welcher Knoten welchem Cluster zugeordnet wird. Andererseits repräsentieren wir eine Kante durch eine Variable, die immer dann den Wert 1 annimmt, wenn die zwei Endknoten einer Kante verschiedenen Clustern zugeordnet werden.
Anschließend stellen wir die von uns implementierten Heuristiken zur Lösung des Clusteringproblems vor. Die Einbindung dieser Algorithmen in unser Plazierungsverfahren und die damit erzielten Ergebnisse werden ebenfalls in Kapitel 11 präsentiert.
In den Kapiteln 12 bis 15 studieren wir das r-Clusteringproblem aus polyedrischer Sicht. Zunächst werden wir das Modell so transformieren, daß das zugehörige Polyeder volldimensional ist. Dies erfolgt in Kapitel 12.
Die "einfachen" Facetten des von uns untersuchten Polyeders erhält man bereits aus der ganzzahligen Modellierung des zugehörigen Problems. Dies werden wir in Kapitel 13 beweisen.
In Kapitel 14 wird schließlich das Problem des Liftings untersucht, d.h. wie können Facetten für ein Problembeispiel so modifiziert werden, daß sie Facetten für ein anderes Problembeispiel definieren. Desweiteren sollen zwei Eigenschaften, die für alle facettendefinierenden Ungleichungen des Clusteringpolytops gelten, herausgestellt werden.
Wir werden in Kapitel 15 einige Klassen von facetten-definierenden Ungleichungen beschreiben, die die Relaxierung des ganzzahligen, linearen Programms verschärfen. Diesen Ungleichungen ist gemeinsam, daß sie Konten- und Kantenvariablen kombinieren.
Abschließen werden wir die Dissertation mit einem Ausblick auf zukünftige Arbeiten.
In Anhang I findet sich die Testserie zu Kapitel 7.1. Anhang II enthält einen kurze Einführung in Graphen-, Polyeder- und Komplexitätstheorie sowie das Literaturverzeichnis.

Teil I

Problemstellung und Hintergrund

In dem ersten Teil dieser Dissertation werden wir uns mit dem Plazieringsproblem beschäftigen. Zur seiner Modellierung werden wir ein quadratisches 0/1 Optimierungsproblem unter linearen Nebenbedingungen vorstellen und kritisch erörtern. Die komplexitätstheoretische Analyse des Modells wird diesen ersten Teil abschließen. Im Detail gliedert sich Teil I folgendermaßen:
Da das Plazierungsproblem ein Teilproblem beim Entwurf elektronischer Schaltungen darstellt, werden wir zunächst das Entwurfsproblem und seine Nebenbedingungen für unterschiedliche Technologien vorstellen. Dies ist der Inhalt von Kapitel 1. Einen Überblick über bisherige Plazierungsverfahren mit ihren Vor- und Nachteilen werden wir in Kapitel 2 geben. Um bestimmte Nachteile herkömlicher Plazierungsalgorithmen zu vermeiden, wird in Kapitel 3 eine Modellierung des Plazierungsproblems vorgeschlagen und anschließend kritisch erörtert. Das von uns entwickelte Modell führt zu einem quadratischen 0/1 Optimierungsproblem unter linearen Nebenbedingungen. In Kapitel 4 werden wir letzteres Optimierungsproblem komplexitätstheoretisch untersuchen. Dabei wird uns vornehmlich die Frage nach der Existenz beweisbar guter, polynomialer Algorithmen beschäftigen.

Kapitel 1 Das Plazierungsproblem beim Entwurf elektronischer Schaltungen

Das Plazierungsproblem im VLSI-Design ist ein Teilproblem beim Entwurf elektronischer Schaltungen. Abhängig von Designregeln und Technologie ergeben sich dabei verschiedene Varianten des Entwurfsproblems und damit auch des Plazierungsproblems. Diese unterscheiden sich in Nebenbedingungen und Optimalitätskriterien. In Abschnitt 1.1 wird das allgemeine Entwurfsproblem vorgestellt. Dem Plazierungsproblem werden wir uns in dem darauffolgenden Abschnitt zuwenden.

1.1 Der Entwurf elektronischer Schaltungen

Eine elektronische Schaltung (auch **Chip** genannt) stellt eine komplexe Verflechtung von Halbleiterbauelementen (sogenannten **Transistoren**) dar. Diese Verflechtung ist die physikalische Umsetzung einer logischen Funktion, die sich aus vielen elementaren Operationen zusammensetzt. Der Entwurf elektronischer Schaltungen läßt sich in zwei Teile gliedern. Im ersten Teil des Entwurfs (**logischer Entwurf**) wird einerseits festgelegt, welche elementaren logischen Operationen zu einer logischen Funktionseinheit zusammengesetzt werden. Andererseits wird definiert, welche logischen Funktionseinheiten an welchen Kontaktpunkten miteinander verbunden werden müssen. Der zweite Teil des Entwurfs ist der **physikalische Entwurf**. In dieser Arbeit werden wir uns ausschließlich mit dem physikalischen Entwurf beschäftigen. Letzteren werden wir kurz den Entwurf nennen.
Bevor auf das Entwurfsproblem näher eingegangen wird, wollen wir zunächst einige grundlegende Begriffe einführen.

Eine **Zelle** stellt eine logische Funktionseinheit dar. Jede Zelle hat aufgrund technischer Bedingungen eine endliche Menge von Realisierungen. Die **Realisierung** einer Zelle ist durch eine Breiten- und Höhenangabe sowie durch die Lage spezieller Kontaktpunkte charakterisiert. Kontaktpunkte werden häufig auch **Pins** genannt. An diese können Drähte angeschlossen werden. Wenn aus dem Zusammenhang ersichtlich ist, daß eine Zelle nur eine Realisierung hat, werden wir von Breite und Höhe der Zelle sprechen. Ein **Netz** ist eine Menge von Tupeln, wobei jedes Tupel ein Pin und die zugehörige Zelle spezifiziert. Letztere Definition führt zu einem aufwendigen Formalismus. Für unsere Zwecke ist es ausreichend, ein Netz als Teilmenge der Zellen zu betrachten und dabei implizit vorauszusetzen, daß die entsprechenden Pins der Zelle gemeint sind. Die Realisierung eines Netzes besteht darin, die Zellen des Netzes an den entsprechenden Kontaktpunkten miteinander durch einen Draht zu verbinden. Unter einem **Master** verstehen wir eine ebene Grundfläche. Der Randbereich des Masters ist die Vereinigung von vier rechteckigen ebenen Flächen, welche den Master umschließen. Zellen werden in **logische Zellen** und **Randzellen** eingeteilt. Logische Zellen müssen innerhalb des Masters plaziert werden, wohingegen Randzellen auf dem Randbereich angeordnet sind. Abbildung 1.1.1 stellt einen Chip (ohne Verdrahtung) vereinfacht dar. Die logischen Zellen sind grau gezeichnet. Das Rechteck, auf welchem die

logischen Zellen plaziert sind, entspricht dem Master. Letzterer ist von den Randzellen umgeben.

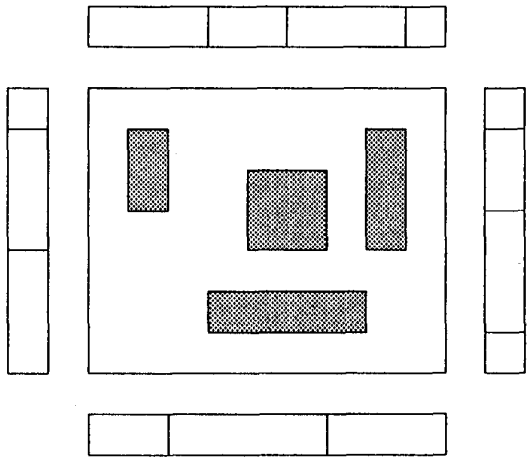


Abbildung 1.1.1

Mit Hilfe dieser Terminologie läßt sich das allgemeine Entwurfsproblem wie folgt formulieren:

Gegeben ist eine ebene Grundfläche (Master und eventuell der Randbereich). Desweiteren sei B eine endliche Menge von Zellen, die wiederum eine endliche Menge von rechteckigen Realisierungen in der Ebene aufweisen. Jede dieser möglichen Realsierungen einer Zelle ist, wie bereits oben erwähnt, unter anderem durch ihre Breite und Höhe, durch ihre elektrischen Anschlüsse (Pins) und ihr elektrisches Verhalten (z.B. Schaltgeschwindigkeit) charakterisiert. Gegeben ist ferner eine Netzliste $N \subseteq 2^B$. Jedes Element $t \in N$ definiert ein Netz.

Gesucht ist eine Zuweisung von genau einer Realisierung für jede Zelle auf der Grundfläche und die Verbindung all der Netze unter bestimmten Nebenbedingungen und unter verschiedensten Optimalitätskriterien. Nebenbedingungen und Optimalitätskriterien hängen dabei von dem jeweiligen Entwurfsstil ab.
In der Praxis haben sich folgende Entwurfsstile durchgesetzt.

"Gate-Arrays"

Ein Gate-Array ist ein vorgefertigter Master, dessen Oberfläche partiell mit Transistoren belegt ist. Zellen dürfen auf der von Transistoren belegten Fläche des Masters plaziert werden. Dies bedeutet physikalisch, daß die logische Funktion, welche eine Zelle definiert, realisiert wird, indem die zugehörigen Transistoren in der geeigneten Weise verbunden (verdrahtet) werden. Die von Transistoren nicht überdeckte Fläche des Masters dient dazu, die Drähte zwischen verschiedenen Zellen zu realisieren. Wesentlicher Vorteil dieses Entwurfsstils ist, daß sich aufgrund des vorgefertigten Masters spezielle Chips lediglich in der

Verdrahtung unterscheiden. Somit ist ein schnellerer Entwurf möglich. Das Verdrahtungsgebiet umschließt die von Transistoren belegte Fläche entweder vollständig (Insel-Typ) oder in horizontaler Richtung (Zeilen-Typ). Ziel der Plazierung ist es, die Verdrahtbarkeit des Chips zu garantieren und die Verdrahtungslänge zu minimieren. Die Abbildungen 1.1.2 bzw. 1.1.3 stellen einen Gate-Array des Insel-Typs bzw. Zeilen-Typs vereinfacht dar.

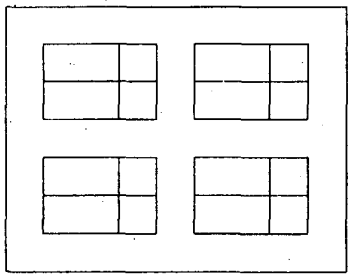


Abbildung 1.1.2

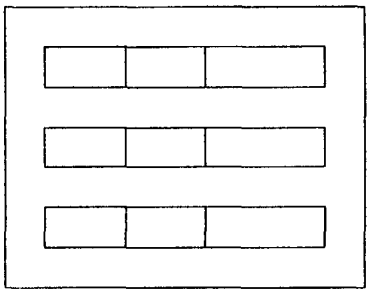


Abbildung 1.1.3

"Sea of cells"

Im Unterschied zu Gate-Arrays ist in diesem Entwurfsstil die Oberfläche des Masters vollständig mit Transistoren belegt. Es existiert somit kein a-priori definiertes Verdrahtungsgebiet. Die von den Zellen nach Plazierung nicht belegte Fläche des Masters dient der Verdrahtung der Netze. Die Zielfunktion für die Plazierung im "Sea of cells"-Entwurfsstil entspricht der für Gate-Arrays. Häufig ist der Master im "Sea of cells"-Entwurfsstil von vier horizontalen und vertikalen Reihen bzw. Spalten umgeben. Sie dienen der Plazierung von Randzellen, welche Eingabe bzw. Ausgabe-Zellen darstellen. Diese Zellen sind in der Regel vorplaziert (vergleiche Abbildung 1.1.4).

"Standardzellen"

Im Standardzellen-Entwurf besitzt der Master keine vorgefertigte Struktur. Vielmehr existieren Reihen, welche horizontal angeordnet sind und welche ausschließlich für die Plazierung der Zellen zur Verfügung stehen. Zwischen zwei benachbarten horizontalen Reihen verläuft ein sogenannter Verdrahtungskanal. Dieser dient, wie im Gate-Array-Entwurfsstil, der Realisierung der Verdrahtung. Die Zellen sind in der Regel von gleicher Höhe. Sie entspricht der Höhe der Reihen. Das primäre Ziel im Standardzellen-Entwurfsstil ist die Minimierung der von der Plazierung und Verdrahtung benötigten Fläche. Daneben bleibt die Verdrahtbarkeitsbedingung des Chips weiterhin als Nebenbedingung bestehen. Abbildung 1.1.5 stellt einen Master im Standardzellen-Entwurfsstil vereinfacht dar.

"Entwurfsstil für allgemeine Zellen"

Bei diesem Entwurfsstil unterliegen die Zellen keinen Einschränkungen. Ferner ist der Master ohne vordefinierte Struktur. Insbesondere gibt es keine Unterteilung der Grundfläche in einen Verdrahtungs- und einen Plazierungsbereich. In diesem Entwurfsstil geht es vorrangig darum, die Fläche, die von Plazierung und Verdrahtung benötigt wird, zu minimieren. Danaben muß die Verdrahtbarkeit der Plazierung garantiert sein.

Abbildung 1.1.4 zeigt einen abstrahierten Chip (ohne Verdrahtung) im "Sea of cells"- Entwurfsstil. Die Abstraktion besteht darin, daß der zugehörige Master nicht auf Transistor-Ebene dargestellt ist, sondern vielmehr eine Unterteilung in kleine, gleich große Quadrate aufweist. Diese werden als **Basiszellen** bezeichnet. Aus physikalischer Sicht bestehen sie selbstverständlich aus vielen Transistoren. Das große Quadrat in Abbildung 1.1.4 stellt den Master dar. Auf diesem sind die logischen Zellen plaziert. Die nicht von Zellen überdeckten Basiszellen stehen für die Verdrahtung zur Verfügung. Der Master ist von vier Rechtecken

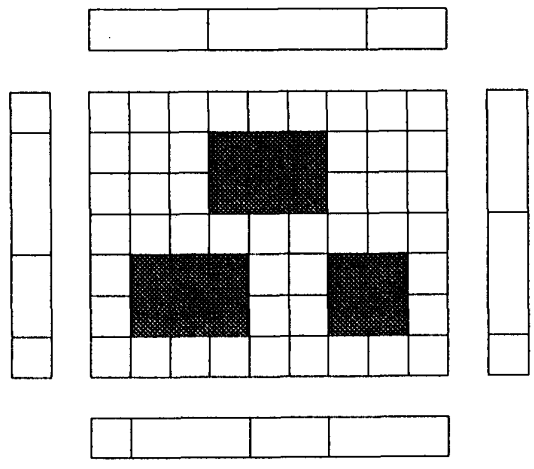


Abbildung 1.1.4

eingeschlossen, auf welchen die Randzellen plaziert sind. Der Master ist in Abbildung 1.1.4 vergrößert dargestellt, damit die Basiszellen besser erkennbar werden. In der Realität sind die Randzellen wesentlich größer als die logischen Zellen.

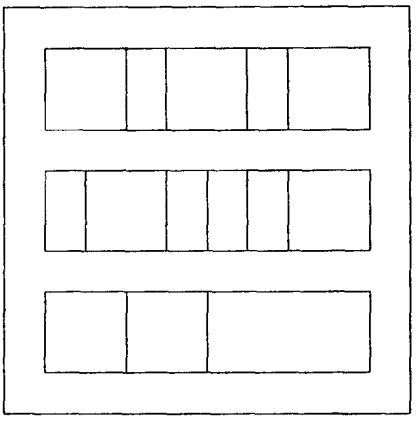


Abbildung 1.1.5

Als Kostenfunktion für das Entwurfsproblem elektronischer Schaltungen werden nach [L90] in der Praxis beispielsweise folgende Größen verwendet:

(i) die von der Plazierung und Verdrahtung benötigte Fläche,

(ii) die Verdrahtungslänge,

(iii) die Anzahl von Punkten, in welchen die Drähte die Verdrahtungsebene wechseln (sogenannte Vias).

Selbst einfache Versionen des Entwurfsproblems sind $\mathcal{NP}$-schwer [GJ79]. Zusätzlich bewirkte die rasante Entwicklung bei der Chip-Herstellung, daß heutzutage elektronische Schaltungen mit mehreren 100 000 Transistoren keine Ausnahme mehr bilden. Von daher scheint es unmöglich, praxisrelevante Varianten dieses Problems in naher Zukunft algorithmisch optimal zu lösen.

Da selbst die heuristische Behandlungen des Gesamtproblems extrem schwierig ist [M90], behilft man sich üblicherweise, das Entwurfsproblem in Unterprobleme aufzuteilen. Ein erstes Problem, welches durch Dekomposition entsteht, ist das Plazieren der Zellen. Dieser Schritt endet damit, daß einer jeden Zelle eine zulässige Position auf der Grundfläche zugeordnet ist. Anschließend wird das Verdrahtungsproblem gelöst. Dieses realisiert die Netze auf dem Chip, wobei die Fabrikationstechnologie des Chips die Netzführung beeinflußt. Bei verschiedenen Technologien bildet schließlich die Kompaktierung des Chips das dritte Problem. In diesem Schritt wird versucht, die Fläche des plazierten und verdrahteten Chips zu reduzieren. Für bestimmte Anwendungen existieren zusätzliche Nebenbedingungen, die von den Algorithmen zur Plazierung, Verdrahtung oder Kompaktierung nicht ausreichend berücksichtigt wurden. Falls der Anwender mit dem Entwurf nicht zufrieden ist, so werden

mit einer anderen Einstellung der Parameter oder auch mit anderen Algorithmen erneut die Schritte Plazierung, Verdrahtung und eventuell Kompaktierung durchlaufen.

1.2 Das Plazierungsproblem

Geht man davon aus, daß jede Zelle nur genau eine Realisierung auf dem Chip hat, d.h. jede Zelle ist gegeben durch ihre Breite und Höhe sowie ihre Pinkoordinaten, so läßt sich das Plazierungsproblem folgendermaßen beschreiben:

> Finde eine überlappungsfreie Zuweisung der linken unteren Ecken der Zellen zu Punkten auf der ebenen Grundfläche, so daß die technologischen Nebenbedingungen erfüllt sind und alle Netze realisiert werden können.
> Wie bereits bei der Definition des Entwurfsproblems erwähnt, hängen die Optimalitätskriterien von dem zugrundeliegenden Entwurfsstil ab. Desweiteren bestimmt der Entwurfsstil, welche Punkte der Ebene für welche Zellen als mögliche Einbauplätze zulässig sind.

Im "Sea of cells"-Entwurfsstil müssen beispielsweise logische Zellen so plaziert werden, daß ihre linken unteren Ecken mit der linken unteren Ecke einer Basiszelle zur Deckung gebracht werden. Ferner sind Randzellen in der Regel vorplaziert.

Es ist die eigentliche Aufgabe einer Plazierung in allen Entwurfsarten, daß die Verdrahtbarkeit des Chips garantiert wird. Das Problem, ein Netz mit minimaler Länge zu verdrahten, ist äquivalent zu dem Problem, einen Steinerbaum minimaler Länge zu finden. Dieses Optimierungsproblem ist $\mathcal{NP}$-schwer [GJ77], [K72]. Das Verdrahtungsproblem im VLSI-Design ist die Aufgabe, Steinerbäume für mehrere Netze simultan zu finden. Kramer und van Leuwen [KL84] beziehungsweise Korte et al. [KPS90] konnten zeigen, daß dieses Entscheidungsproblem bereits in Spezialfällen $\mathcal{NP}$-vollständig ist. Sucht man nach einer Verdrahtung minimaler Länge, so werden in der Regel nicht mehr alle einzelnen Steinerbäume mit minimaler Länge gelegt werden können, da sich die Bäume eventuell gegenseitig behindern. Vielmehr treten zusätzliche Schwierigkeiten in dem Sinne auf, daß zu entscheiden ist, welche der Steinerbäume welche Umwege machen müssen und welche der Bäume mit minimaler Länge gelegt werden können.

Obige Ausführungen machen deutlich, daß es $\mathcal{NP}$-vollständig ist, zu entscheiden, ob eine Plazierung verdrahtbar ist. Aus diesem Grund wird bei den bisher bekannten Plazierungsverfahren die Verdrahtbarkeitsbedingung ersetzt durch Berücksichtigung der Verdrahtungslänge in der Zielfunktion. Heuristisch betrachtet ist es für eine Plazierung mit geringerer Verdrahtungslänge wahrscheinlicher, daß der so plazierte Chip verdrahtbar ist, als bei einer Plazierung mit größerer Verdrahtungslänge. Dies ist plausibel, da die für die Verdrahtung benötigte Fläche zur Verdrahtungslänge proportional ist. Um die exakte Verdrahtungslänge eines Chips zu bestimmen, muß das Verdrahtungsproblem gelöst

werden. Setzt man $\mathcal{P} \neq \mathcal{NP}$ voraus, so ist diese Aufgabe im allgemeinen in polynomialer Laufzeit nicht lösbar. In der Praxis behilft man sich deshalb, eine Abschätzung der Verdrahtungslänge als Zielfunktion zu verwenden.

Um die Länge eines Netzes abzuschätzen, werden in der Literatur unter anderem folgende Methoden aufgeführt:

(i) Der halbe Umfang des Rechtecks, das die Pins des Netzes minimal einschließt, liefert eine schnell zu berechnende untere Schranke für die Netzlänge. Zur Berechnung der Distanz verwendet man üblicherweise die L_1-Norm.

(ii) Eine weitere Approximation für die Länge eines Netzes erhält man, indem das Netz in seine zwei-Punkt-Verbindungen unterteilt wird. Der Abstand zweier Pins des Netzes wird beispielsweise in der L_1-Norm berechnet. Die Summe all der Abstände der Zwei-Punkt-Verbindungen schätzt die Länge des Netzes ab. Da durch dieses Vorgehen Netze der Kardinalität größer 2 überbewertet werden, ist eine Gewichtung der zwei-Punkt-Abstände mit einem Faktor (beispielsweise $\frac{1}{n-1}$) angebracht, wenn n die Kardinalität des Netzes t ist. Diese Methode wird als das **vollständige Graphenmodell** zur Abschätzung der Netzlängen bezeichnet.

(iii) In der Literatur findet man eine Abschätzung für die Länge eines Netzes durch Berechnung des minimal aufspannenden Baumes. Der zugrundeliegende Graph ist der vollständige Graph auf den durch die Pins des Netzes definierten Knoten. Das Gewicht einer Kante bestimmt sich als Abstand, gemessen in der L_1- oder L_2-Norm, zwischen den zu der Kante inzidenten Knoten.

Somit sind die im folgenden benötigten Details bezüglich des Plazierungsproblems und des übergeordneten Entwurfsproblems dargestellt.

Kapitel 2 Plazierungsansätze

In diesem Kapitel werden wir zunächst die Schwierigkeiten bei der Modellierung des Plazierungsproblems aufzeigen. Desweiteren geben wir einen Überblick über die wichtigsten Plazierungsverfahren beim Entwurf elektronischer Schaltungen. Dabei werden wir uns auf den "Sea of cells"-Entwurfsstil beschränken. Diese Einschränkung ist notwendig, da eine Vielzahl von Plazierungsalgorithmen existiert, welche für die unterschiedlichen Entwurfsstile konzipiert wurden und die auch nur für einen Entwurfsstil effizient anwendbar sind. Wir beginnen mit der Beschreibung der Min-Cut-Heuristik. Anschließend werden wir die Idee von Kräftemodellen darstellen und im besonderen das Plazierungsverfahren "Gordian" genauer erklären. Eine kurze Zusammenfassung der "Simulated Annealing"-Methode wird dieses Kapitel abschließen. Für eine exzellente und detailierte Beschreibung der verschiedenen Plazierungsverfahren verweisen wir auf das Buch [L91].

2.1 Schwierigkeiten bei der Modellierung des Plazierungsproblems

Unter der Modellierung eines praktischen Problems versteht man die Extraktion der wesentlichen Eigenschaften und Zielsetzungen der Problemstellung und deren Abbildung auf eine mathematische Aufgabe. Solch eine mathematische Aufgabe läßt sich häufig durch Einführen von Zustandsvariablen und deren Verkettung ableiten. Bei der Modellierung ist es insbesondere wichtig, daß das praktische Problem zunächst präzise nachgebildet wird. Versucht man diese Leitlinie auf das Plazierungsproblem anzuwenden, so ergeben sich einige Schwierigkeiten.

Wie wir bereits in Kapitel 1 gesehen haben, muß eine Plazierung in allen Entwurfsarten die Verdrahtbarkeit des Chips garantieren. Das zugehörige Entscheidungsproblem ist jedoch bereits in Spezialfällen $\mathcal{NP}$-vollständig. Aus diesem Grund wird bei den Plazierungsverfahren die Verdrahtbarkeitsbedingung dadurch ersetzt, daß in der Zielfunktion die Verdrahtungslänge berücksichtigt wird. Um die exakte Verdrahtungslänge eines Chips zu bestimmen, muß das Verdrahtungsproblem gelöst werden. Setzt man $\mathcal{P} \neq \mathcal{NP}$ voraus, so ist diese Aufgabe im allgemeinen in polynomialer Laufzeit nicht lösbar. Dies führt dazu, daß die Verdrahtungslänge abgeschätzt wird.

Die Aufgabe, eine präzise Modellierung des allgemeinen Plazierungsproblems zu finden, wird zusätzlich dadurch erschwert, daß verschiedene Entwurfsverfahren mit unterschiedlichen Optimalitätskriterien vorhanden sind. Ferner existiert eine Vielzahl technischer Randbedingungen, welche von Anwendung zu Anwendung stark differieren. So dürfen beispielsweise bei bestimmten Anwendungen Netze eine vorgegebene Länge nicht überschreiten. Umgekehrt dürfen andere Netze eine Mindestlänge nicht unterschreiten. Unter Umständen ist es auch erforderlich, unter verschiedenen Netzen unterschiedliche Mindestdistanzen einzuhalten. Bedingungen dieser Art sind in gewissem Rahmen bereits beim Plazieren der Zellen zu berücksichtigen.

Diese Fülle von Problemen bei der Modellierung des Plazierungsproblems lassen es zunächst nur schwer möglich erscheinen, ein präzises Modell für das allgemeine Plazierungsproblem zu finden. Nicht zuletzt aus diesem Grund existiert eine Vielzahl von Plazierungsalgorithmen, welche für die unterschiedlichen Entwurfsstile konzipiert wurden und auch nur für einen Entwurfsstil effizient anwendbar sind.

In den Abschnitten 2.2 bis 2.4 dieses Kapitels werden wir drei Plazierungsansätze vorstellen, welche für den "Sea of cells"-Entwurfsstil entwickelt wurden. In dem folgenden Kapitel 3 dieser Dissertation werden wir schließlich unser Modell zur Lösung des Plazierungsproblems vorstellen. Dieses soll ebenfalls für den Entwurf von Schaltungen in der "Sea of cells"-Technologie anwendbar sein. Wir werden das von uns gewählte Modell hinsichtlich der Erweiterbarkeit auf andere Entwurfsstile bzw. hinsichtlich der Integration zusätzlicher Bedingungen in dieses Modell jedoch kritisch beurteilen.

2.2 Die Min-Cut-Heuristik

Gegeben ist ein Hypergraph $G = (V, E)$, dessen Knoten die Zellen repräsentieren. Jede Hyperkante $e \in E$ entspricht einem Netz der Schaltung. Ferner ist im "Sea of cells"-Entwurfsstil der Master der Schaltung (Grundfläche) gegeben.

Alle Varianten der Min-Cut-Heuristik beruhen auf der Idee, das Problem rekursiv in kleinere Probleme zu zerlegen, bis die kleinsten dabei auftretenden Teilprobleme trivial lösbar sind. Das kleinste Teilproblem besteht darin, eine konstante Anzahl von Zellen auf einer gegebenen rechteckigen Teilfläche des Masters zu plazieren. Durch entsprechendes Zusammenfügen der Teilprobleme läßt sich schließlich eine Lösung des Ausgangsproblems finden (vergleiche hierzu [SK72] [B77] [L79]).
Diese Vorgehensweise kann schematisch in einem binären Baum dargestellt werden. Jeder Knoten des Baums repräsentiert einen Teilgraphen des Hypergraphen und eine rechteckige Teilfläche des Masters, auf welcher der Teilgraph zu plazieren ist. Die Wurzel des Baums repräsentiert den gesamten Hypergraphen G und den Master. Die beiden "Kinder" eines inneren Knotens v werden generiert, indem die zugehörige Grundfläche in zwei Rechtecke aufgeteilt wird. Ferner wird der Hypergraph, welcher durch v repräsentiert wird, in zwei Teilgraphen zerlegt.

Das Abarbeiten des Baumes erfolgt entweder über Tiefen- oder Breitensuche. Der Unterschied unterschiedlicher Varianten von Min-Cut-Verfahren besteht im wesentlichen darin,
(a) wie das Aufteilen der Grundflächen organisiert ist und
(b) nach welchen Optimalitätskriterien und mit welchen Verfahren das Problem gelöst wird, den Hypergraphen in zwei Teilgraphen zu zerlegen.

Zu (a):

Beim "Sea of cells"-Entwurf definiert die Einteilung des Masters in Basiszellen eine Menge von horizontalen und vertikalen Schnittlinien. Folglich lassen sich die Probleme, rechteckige Grundflächen in zwei rechteckige Teilflächen zu unterteilen, darauf zurückführen,

eine Reihenfolge der Schnittlinien festzulegen. In der Praxis haben sich zwei Vorgehensweisen durchgesetzt. Die erste Methode wendet abwechselnd einen vertikalen und horizontalen Schnitt an. Das bedeutet zum Beispiel, daß iterativ eine Fläche zunächst durch einen vertikalen Schnitt halbiert wird. Anschließend werden die beiden Teile durch je einen horizontalen Schnitt halbiert.
Bei der anderen Methode wird zunächst eine Schnittrichtung festgelegt (z. B. vertikal). Dann werden von dem Master von links nach rechts gehend Scheiben (slice cuts) in vertikaler Richtung abgeschnitten. Nachdem die Menge der vertikalen Schnitte abgearbeitet ist, wird dieselbe Prozedur in horizontaler Richtung angewendet.

Zu (b):

Wie bereits in Kapitel 1 ausgeführt, ist beim "Sea of cells"-Entwurfstil der Packungsaspekt (Minimierung der für die Plazierung benötigten Fläche) irrelevant. Dagegen steht die Aufgabe im Vordergrund, die gesamte Verdrahtungslänge zu minimieren. Als Heuristik ist es daher plausibel, die Anzahl der Hyperkanten, welche eine Schnittlinie kreuzen, zu minimieren. Um eine gleichmäßige Zuordnung der Zellen zu den beiden Teilflächen zu gewährleisten, ist dabei eine Flächenrestriktion einzuhalten. Seien R_1 und R_2 die beiden Teilrechtecke mit Flächen $F(R_1)$ bzw. $F(R_2)$, dann sollen die Gesamtflächen der Zellen, welche R_1 zugeordnet werden und die, welche R_2 zugeordnet werden, in dem gleichen Verhältnis stehen, wie $F(R_1)$ und $F(R_2)$. Dieses Problem ist $\mathcal{NP}$-schwer [GJ 79]. Zu seiner Lösung sind eine Vielzahl von deterministischen und nicht-deterministischen Heuristiken entwickelt worden. Die bekannteste hiervon ist die Kernighan-Lin-Heuristik [KL70], welche sich auch auf andere Optimierungsprobleme anwenden läßt. Das Verfahren beruht im wesentlichen darauf, eine Anfangslösung iterativ zu verbessern. In jedem Schritt wird dabei der beste Austausch durchgeführt, auch wenn — und das ist entscheidend — dadurch momentan eine Verschlechterung in der Zielfunkton erreicht wird. Zu Ende des Verfahrens wird ausgehend von der Anfangslösung die Sequenz des Austauschens bis zu dem Schritt durchlaufen, in welchem die Zielfunktion ihr Minimum (unter allen Schritten) annimmt. Eine schnelle Variante der Kernighan-Lin Heuristik ist von Fiduccia und Matheyses [FM82] entwickelt worden. Die sorgfältige Analyse der Aufgabenstellung, zusammen mit der Konzeption effizienter Datenstrukturen, haben diese Heuristik bekannt werden lassen.

2.3 Kräftemodelle

Der wesentliche Nachteil von Min-Cut-Verfahren ist sicherlich, daß durch das rekursive Aufteilen des Problems in kleinere Unterprobleme die globale Sicht verloren geht. Um dies zu vermeiden, hat sich eine Klasse von Verfahren herausgebildet, die das Plazierungsproblem so modelliert, daß eine Relaxierung des Modells optimal gelöst werden kann. Diese Relaxierung ist ein kontinuierliches quadratisches Optimierungsproblem mit oder ohne Nebenbedingungen. Die Zielfunktion dieser Relaxierung entspricht in der Regel einer

quadratischen Abschätzung der gesamten Verdrahtungslänge. Die Nebenbedingungen modellieren beispielsweise, daß die triviale Plazierung (alle Zellen werden derselben Position zugewiesen) ausgeschlossen wird [B85]. Andere Kräftemodellverfahren lösen eine Folge von quadratischen Optimierungsproblemen. In diesem Fall steuern die Nebenbedingungen, daß Information aus den vorigen Iterationen in der momentanen Berücksichtigung findet.
Die kontinuierlichen quadratischen Optimierungsprobleme werden mit Eigenvektorverfahren oder Versionen der Gradientenmethode gelöst. Oft funktionieren die Ansätze nur, wenn bestimmte Zellen vorplaziert sind. Sonst wäre die triviale Plazierung eine Optimallösung des Problems.
In einer zweiten Phase versuchen dann Kräftemodelle die optimale Lösung der Relaxierung zu modifizieren, um eine zulässige Lösung für das Plazierungsproblem zu erhalten.

In dem folgenden Teil des Kapitels beschreiben wir kurz das Plazierungsverfahren "Gordian" als einen Vertreter der Klasse von Kräftemodellen (vergleiche hierzu [JJA83] [JK85] [KSJ88] [KSFA91]).

Das Plazierungsverfahren "Gordian" besteht aus dem iterativen Lösen eines kontinuierlichen quadratischen Optimierungsproblems und einem Aufteilungsschritt. Zur Formulierung des quadratischen Optimierungsproblems werden zunächst für die horizontalen und vertikalen Koordianten des "Schwerpunktes" eines Netzes Variablen eingeführt. Die Koordinaten des Mittelpunkts einer Zelle werden ebenfalls durch Variablen repräsentiert. Die Zielfunktion der Optimierungsaufgabe ist eine Abschätzung der Verdrahtungslänge. Diese bestimmt sich als die Summe über alle Netze der Abstände der Netzpositionen und der zu dem Netz gehörigen Zellpositionen. Als Distanz zwischen einem Netz und den zu dem Netz gehörigen Zellen wird der euklidsche Abstand gewählt. Die Nebenbedingungen des Optimierungsproblems sind eine Anforderung an die Verteilung der Zellenpositionen auf die Teilflächen der gegenwärtigen Iteration. Stochastisch lassen sie sich so interpretieren, daß der Erwartungswert einer gewichteten Summe der x (y)-Koordinaten aller Zellen einer Teilfläche dem Mittelpunkt der Teilfläche gleich sein muß. Nebenbedingungen und Zielfunktion des quadratischen Optimierungsproblem sind konvex. Daher definiert die Lösung des Modells ein globales Optimum. In dem Aufteilungsschritt werden, wie in Kapitel 2.1 beschrieben, alle Teilflächen der gegenwärtigen Iteration durch vertikale oder horizontale Schnitte unterteilt, sofern mehr als eine Zelle zu der Teilfläche gehört. Die Zellen einer Teilfläche werden anschließend den beiden unterteilten Gebieten zugewiesen, wobei die Zuweisung das Flächenverhältnis der unterteilten Gebiete berücksichtigen muß. Das Verfahren endet schließlich, wenn in jeder Teilfläche genau eine Zelle ist.
Diese Methode ist meines Wissens nach das einzige Kräftemodellverfahren, welches beim Entwurf hoch integrierter Schaltungen in der Praxis eingesetzt wird. Dies liegt wohl daran, daß Kräftemodellverfahren in der Regel das große Problem mit sich bringen, die Lösung der kontinuierlichen Relaxierung zu einer Lösung des eigentlich diskreten Problems der Plazierung zu modifizieren. Aus diesem Grund greift das Verfahren "Gordian" auf den Aufteilungsschritt des Min-Cut-Verfahrens zurück. Jedoch funktioniert "Gordian" nur dann, wenn die Menge der vorplazierten Zellen nicht leer ist. Falls für den Entwurf einer

Schaltung alle Zellen frei zu plazieren sind, ist die Optimallösung des ersten quadratischen Problems die triviale Plazierung.

2.4 Simulated Annealing

Die Methode des Simulated Annealing (simulierte Abkühlung) ist ein nicht-deterministisches Verfahren zur Lösung allgemeiner kombinatorischer Optimierungsprobleme. Die Idee hierzu kommt aus der Physik und läßt sich im wesentlichen so beschreiben:
Ein Gebiet der statistischen Mechanik ist das Studium des Verhaltens komplexer Systeme. Diese bestehen bei fester Temperatur aus einer großen Anzahl von in Wechselwirkung stehenden Atomen in thermischem Gleichgewicht. Die Wahrscheinlichkeit, daß sich das System in einem bestimmten Zustand befindet, ist von der Energie eben dieses Zustands abhängig. Speziell sind Zustände mit geringstem Energiepotential am wahrscheinlichsten. Um Materie in den Zustand geringer Energie zu bringen, verwendet man in der Physik den Prozeß des sorgfältigen Abkühlens. Hierzu erhitzt man das System zunächst, bis der Schmelzpunkt erreicht wird. Anschließend kühlt man es langsam ab [M53].
Die Methode des Simulated Annealing besteht nun darin, ein Minimierungsproblem auf den physikalischen Prozeß des Suchens nach einem Zustand minimaler Energie abzubilden (vergleiche hierzu [K83], [O84]). Eine zulässige Lösung des Optimierungsproblems simuliert einen physikalischen Zustand, die Zielfunktion entspricht dem Energiepotential, und der Übergang eines physikalischen Zustands in einen benachbarten entspricht dem Generieren einer benachbarten Lösung über einen Austauschschritt.
Das Verfahren beginnt damit, eine Anfangslösung für das Optimierungsproblem zu generieren (zum Beispiel zufällig). Ferner wird eine Zahl, welche der Temperatur bei Schmelzung des physikalischen Systems entsprechen soll, vorgegeben. Diese Zahl wird im folgenden als Temperartur bezeichnet. Bei fester Temperatur werden sukzessive benachbarte Lösungen des Optimierungsproblems über Austauschschritte zufällig erzeugt. Wenn die Zielfunktion nach Durchführung des Austauschschritts verbessert werden kann, akzeptiert man den Austausch. Führt der Austausch zu einer Verschlechterung in der Zielfunktion, so wird er mit einer bestimmten Wahrscheinlichkeit akzeptiert. Dieser Prozeß wird bis zur Erfüllung eines Stopkriteriums iteriert. Anschließend verringert man die Temperatur und wiederholt obige Vorgehensweise. Die Akzeptanzwahrscheinlichkeit hängt dabei von der Temperatur qualitativ so ab, daß bei höherer Temperatur Verschlechterungen mit größerer Wahrscheinlichkeit akzeptiert werden als bei niedrigerer Temperatur. Dadurch soll der Abkühlungsprozess simuliert werden. Eine Interpretation für das Akzeptieren von Lösungen mit höherem Zielfunktionswert ist die, daß man es vermeiden möchte, aus bezüglich der Austauschschritte lokalen Optima nicht herauszufinden.

Simulated Annealing ist ein generelles Verfahren zur Lösung kombinatorischer Optimierungsprobleme. In der Regel liefert es, sofern ausreichend Rechenzeit zur Verfügung steht, recht gute Ergebnisse. Entscheidender Nachteil der Methode ist jedoch, daß einerseits

eine gute Einstellung (Automatisierung) der vielen Parameter wie etwa Anfangstemperatur, Anfangslösung, Akzeptanzwahrscheinlichkeiten, Austauscharten, Strafparameter und mehr äußerst schwer ist. Andererseits benötigt das Verfahren wesentlich mehr Rechenzeit als deterministische Algorithmen.

Kapitel 3 Das quadratische 0/1 Modell (P)

Es existiert eine Vielzahl von Verfahren, welche zur Lösung des Plazierungsproblems entwickelt wurden. Diese sind jedoch zum Teil nur auf einen Entwurfstil oder für eine spezielle Zielfunktion effizient anwendbar. Insbesondere können zusätzliche Nebenbedingungen bewirken, daß eine völlig neue Implementierung notwendig wird.
Wir werden das Plazierungsproblem als ein quadratisches ganzzahliges Optimierungsproblem unter linearen Nebenbedingungen modellieren. Die Modellierung wird zunächst für den "Sea of cells"-Entwurf zugeschnitten sein. Erweiterungen des Modells, insbesondere im Hinblick auf Berücksichtigung der von der Plazierung benötigten Fläche, frei plazierbarer Randzellen, Spiegelung der Zellen, Floorplaning und verschiedener zusätzlicher Nebenbedingungen werden anschließend diskutiert. In dem folgenden Abschnitt werden wir einen Graphen einführen. Dieser wird es uns erlauben, das quadratische Modell als ein graphentheoretisches Problem zu formulieren. Den Abschluß dieses Kapitels wird ein Einblick in die Literatur über quadratische 0/1 Optimierung ohne Nebenbedingungen bilden. Dieses Problem wurde intensiv studiert. Wir interessieren uns dafür insbesondere, da (P) und das quadratische 0/1 Optimierungproblem ohne Nebenbedingungen bis auf Addition einer Konstanten äquivalent sind.

Zur Formulierung des Plazierungsmodells führen wir noch folgende Notation ein:
Wir gehen davon aus, daß alle zu plazierenden Zellen (logische Zellen) von $1,\ldots,n$ numeriert sind, und R die Menge der vorplazierten Randzellen ist. Sei $N = \{1,\ldots,n\}$. Die Breite bzw. Höhe einer logischen Zelle i werde mit w_i bzw. h_i abgekürzt. Ferner bezeichne z die Anzahl der Netze. Das Netz mit der Nummer t habe die Kardinalität α_t. A_t bezeichne die Menge der Zellen (logische Zellen und Randzellen), welche Netz t repräsentiert. $L_t \subseteq A_t$ ist die Menge der logischen Zellen des Netzes t und $R_t \subseteq A_t$ die Menge der vorplazierten Randzellen. Somit gilt $A_t = L_t \cup R_t$. Zur Vereinfachung der Notation werden wir für die Menge A_t häufig die abkürzende Schreibweise t verwenden. Für eine Randzelle $r \in R$ mit Breite $w(r)$ und Höhe $h(r)$ bezeichne $x(r)$ und $y(r)$ die horizontale und vertikale Koordinate des ihr (a-priori) zugewiesenen Einbauplatzes $e(r)$. Die Breite und Höhe des Masters werde mit a und b bezeichnet. Als Koordinatenursprung wählen wir die linke untere Ecke des Masters.
m sei die Anzahl der Basiszellen des a-priori gegebenen Masters. Ferner bezeichne $Z(i) \subseteq \{1,\ldots,m\}$ die Menge der zulässigen Basiszellen für Zelle i oder kurz die Zulässigkeitsmenge. Eine Basiszelle ist zulässig für eine Zelle genau dann, wenn die Positionierung der linken unteren Ecke der Zelle auf eben dieser Basiszelle auf dem vorgegebenen Master realisiert werden kann.
Um auszudrücken, daß eine Zelle i einer Basiszelle $k \in Z(i)$ zugewiesen wird, führen wir zunächst das Tupel (i,k) ein. Wenn aus dem Zusammenhang keine Verwirrung entstehen kann, so schreiben wir statt (i,k) auch kurz ik.

3.1 Das Modell (P)

Eine Instanz des Plazierungsproblems für den "Sea of cells"-Entwurfsstil ist charakterisiert durch folgende Eingabedaten:

die Anzahl von Zellen;
Breite und Höhe jeder Zelle;
die Zulässigkeitsmenge jeder Zelle;
die Anzahl von Netzen und die Netzliste;
Breite und Höhe des Masters;

Variablen

Zur Modellierung des Problems führen wir für $i = 1, \ldots, n$ und für $k \in Z(i)$ Variablen ein. Diese lassen sich wie folgt interpretieren.

$$x_{ik} = \begin{cases} 1, & \text{wenn die linke untere Ecke der Zelle } i \text{ der} \\ & \text{Basiszelle } k \text{ zugewiesen wird.} \\ 0, & \text{sonst.} \end{cases}$$

Nebenbedingungen

(1) Jede Zelle muß genau einmal plaziert werden. Diese Nebenbedingung kann formal folgendermaßen beschrieben werden:

$$\sum_{k \in Z(i)} x_{ik} = 1 \text{ für alle } i = 1, \ldots, n.$$

(2) Alle Variablen müssen entweder den Wert 0 oder den Wert 1 annehmen:

$$x_{ik} \in \{0, 1\} \text{ für alle } i = 1, \ldots, n, \; k \in Z(i).$$

An dieser Stelle verzichten wir auf eine Modellierung der Überlappungsfreiheit. Diese Nebenbedingung wird in der Zielfunktion berücksichtigt.

Zielfunktion

Die Zielfunktion besteht aus einer gewichteten Summe. Der erste Summand entspricht einer Abschätzung der Gesamtverdrahtungslänge. Der zweite Summand zählt die Anzahl von Basiszellen, welche von mehr als einer Zelle überdeckt werden.

Zur Abschätzung der Gesamtverdrahtungslänge gehen wir folgendermaßen vor:
Zunächst verwenden wir das in Kapitel 1 vorgestellte Modell des vollständigen Graphen in leicht abgeänderter Form, um die Länge eines Netzes abzuschätzen. Den Abstand zwischen zwei Pins des Netzes bestimmen wir nicht nach der L_1 Norm. Stattdessen berechnen wir die "shortest Manhattan"-Distanz zwischen den beiden Zellen der Pins. Die **"shortest Manhattan"-Distanz** zwischen zwei Zellen ist definiert als der minimal Abstand, gemessen in der L_1-Norm, zwischen allen Paaren von Punkten auf den Rändern der zwei Zellen. Wie in Kapitel 1 dargestellt, werden die Zellabstände mit dem Faktor $\frac{1}{\alpha_t - 1}$ gewichtet, damit Netze der Kardinalität größer zwei nicht allzusehr überbewertet werden (die Zahl α_t wurde zu Anfang des Kapitels eingeführt und bezeichnet die Kardinalität des Netzes t).
Ferner führen wir **Affinitätskoeffizienten** ein. Diese sind zwischen allen Paaren von Zellen i und j, $i \neq j$ definiert und bestimmen sich folgendermaßen:

$$c_{ij} = \begin{cases} 0, & \text{wenn kein Netz existiert,} \\ & \text{welches Zellen } i \text{ und } j \text{ beide enthält.} \\ \sum_{t=1 \mid i \in A_t, j \in A_t}^{z} \frac{1}{\alpha_t - 1}, & \text{sonst.} \end{cases}$$

Gegeben sei eine zulässige Lösung des Plazierungsproblems und es bezeichne k_i die Basiszelle, welcher eine logische Zelle oder Randzelle i zugewiesen wurde. Ferner kürzen wir den "shortest Manhattan"-Abstand, wenn Zelle i auf Platz k und Zelle j auf Platz l ist, mit $d(i,k,j,l)$ ab.
Dann entspricht der Ausdruck

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} c_{ij} d(i,k,j,l) x_{ik} x_{jl} + \sum_{i=1}^{n} \sum_{k \in Z(i)} \sum_{r \in R} c_{ir} d(i,k,r,e(r)) x_{ik}$$

der nach dem Modell des vollständigen Graphen abgeschätzten Gesamtverdrahtungslänge. Denn es gilt:

$$\begin{aligned}
&\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} c_{ij} d(i,k,j,l) x_{ik} x_{jl} + \sum_{i=1}^{n} \sum_{k \in Z(i)} \sum_{r \in R} c_{ir} d(i,k,r,e(r)) x_{ik} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} d(i,k_i,j,k_j) + \sum_{i=1}^{n} \sum_{r \in R} c_{ir} d(i,k_i,r,k_r) \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{t=1 \mid \{i,j\} \subseteq A_t}^{z} \frac{1}{\alpha_t - 1} d(i,k_i,j,k_j) + \sum_{i=1}^{n} \sum_{r \in R} \sum_{t=1 \mid \{i,r\} \subseteq A_t}^{z} \frac{1}{\alpha_t - 1} d(i,k_i,r,k_r) \\
&= \sum_{t=1}^{z} \sum_{\{i,j\} \subseteq A_t} \frac{1}{\alpha_t - 1} d(i,k_i,j,k_j).
\end{aligned}$$

Der zweite Teil der Zielfunktion bewertet die Anzahl von Überlappungen in der Plazierung. Hierzu bestimmen wir Koeffizienten $o(i,k,j,l)$ für $i = 1,\ldots,n$, $j = 1,\ldots,n$, $k \in Z(i)$, $l \in$

$Z(j)$. Diese zählen die Anzahl von Basiszellen, die gemeinsam von den Zellen i, j überdeckt werden, wenn i Basiszelle k und j Basiszelle l zugewiesen wird. Da eine zulässige Plazierung überlappungsfrei sein muß, wird der zweite Summand mit einem Bestrafungsparameter $\lambda_o \in \mathbb{R}$ gewichtet.

Zusammenfassend stellt sich die Zielfunktion dar als:

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} c_{ij} d(i,k,j,l) x_{ik} x_{jl} + \sum_{i=1}^{n} \sum_{k \in Z(i)} \sum_{r \in R} c_{ir} d(i,k,r,e(r)) x_{ik}$$

$$+ \lambda_o \Big(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} o(i,k,j,l) x_{ik} x_{jl} \Big).$$

Das vollständige Modell läßt sich damit folgendermaßen angeben:

$$\begin{aligned} \min & \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} c_{ij} d(i,k,j,l) x_{ik} x_{jl} \\ & + \sum_{i=1}^{n} \sum_{k \in Z(i)} \sum_{r \in R} c_{ir} d(i,k,r,e(r)) x_{ik} \\ & + \lambda_o \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} o(i,k,j,l) x_{ik} x_{jl}, \\ \text{so daß } & \sum_{k \in Z(i)} x_{ik} = 1, \text{ für alle } i = 1, \ldots n. \\ & x_{ik} \in \{0,1\}, \text{ für alle } i = 1, \ldots n, \ k \in Z(i). \end{aligned}$$

Im folgenden werden wir dieses quadratische 0/1 Optimierungsproblem mit (P) abkürzen. Formal ist dann eine Instanz von (P) folgendermaßen charakterisiert (man beachte, daß aufgrund der Ganzzahligkeitsbedingungen $x_{ik} = x_{ik}^2$ gilt):

Gegeben natürliche Zahlen n und m. Ferner seien die Zulässigkeitsmengen $Z(i)$ mit $|Z(i)| \leq m$ für alle $i = 1, \ldots, n$ und eine Matrix $Q \in \mathbb{Q}^{\sum_{i=1}^{n} |Z(i)| \times \sum_{i=1}^{n} |Z(i)|}$ gegeben.

$$\begin{aligned} & \min x^T Q x, \\ \text{so daß } & \sum_{k \in Z(i)} x_{ik} = 1, \text{ für alle } i = 1, \ldots n. \\ & x_{ik} \in \{0,1\}, \text{ für alle } i = 1, \ldots n, \ k \in Z(i). \end{aligned}$$

Im folgenden werden wir jedoch häufig das Problem (P) ohne konkreten Bezug zum VLSI-Design betrachten. In diesem Fall gehen wir zur Vereinfachung der Darstellung davon aus, daß alle Zulässigkeitsmengen der Menge $\{1, \ldots, m\}$ entsprechen. Input einer Instanz sind dann die Zahlen n, m und eine Matrix $Q \in \mathbb{Q}^{nm \times nm}$.

Zusätzlich zu dem allgemeinen Problem (P) werden wir eine Variante (P_m) betrachten, wobei m eine konstante Zahl ist, d.h. m gehört nicht mehr zum Input. Diese Variante wird insbesondere bei dem Dekompositionsansatz, welchen wir in Teil II dieser Arbeit vorstellen, eine wichtige Rolle spielen.

Diskussion des Modells

(a) Definiert man $\sum_{r\in\emptyset} c_{ij}x_{ik}d(i,k,r,e(r)) := 0$, so läßt sich Modell (P) analog anwenden, wenn die Menge R der vorplazierten Zellen leer ist.

(b) Falls eine Teilmenge T der Randzellen nicht vorplaziert, sondern vielmehr frei zu plazieren ist, so läßt sich das sehr einfach in das Modell aufnehmen. Man definiert entsprechend Zulässigkeitsmengen $Z(t)$ für alle $t \in T$ und betrachtet die Menge T als Teilmenge der logischen Zellen.

(c) Durch entsprechende Einschränkung der Mengen $Z(.)$ läßt sich sicherstellen, daß bestimmte Positionen für bestimmte Zellen ausgeschlossen sind, wenn dies eine technische Nebenbedingung ist.

(d) In manchen Entwurfsverfahren ist vorgesehen, daß Zellen vertikal oder horizontal gespiegelt werden können. In diesen Fällen hat eine Zelle bis zu 4 verschiedene Realisierungen, nämlich ungespiegelt, nur vertikal gespiegelt, nur horizontal gespiegelt, horizontal und vertikal gespiegelt. Führt man Variablen x^a_{ik} für Zelle i auf Position $k \in Z(i)$ und in Realisierung (Ausprägung) $a \in A(i)$ ein, so stellt sich das erweiterte quadratische Modell dar als:

$$\begin{aligned}
\min & \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in Z(i)}\sum_{l\in Z(j)}\sum_{a\in A(i)}\sum_{c\in A(j)} c_{ij}d(i,k,a,j,l,c)x^a_{ik}x^c_{jl} \\
& + \sum_{i=1}^{n}\sum_{k\in Z(i)}\sum_{a\in A(i)}\sum_{r\in R} c_{ir}x^a_{ik}d(i,k,a,r,e(r),a(r)) \\
& + \lambda_o\Big(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k\in Z(i)}\sum_{l\in Z(j)}\sum_{a\in A(i)}\sum_{c\in A(j)} o'(i,k,a,j,l,c)x^a_{ik}x^c_{jl}\Big), \\
\text{so daß } & \sum_{k\in Z(i)}\sum_{a\in A(i)} x^a_{ik} = 1, \text{ für alle } i = 1,\dots n. \\
& x^a_{ik} \in \{0,1\}, \text{ für alle } i = 1,\dots n,\ k \in Z(i),\ a \in A(i).
\end{aligned}$$

$a(r)$ bezeichnet die Ausprägung der Randzelle $r \in R$. Die Koeffizienten $d(i,k,a,j,l,c)$ bezeichnen die "shortest Manhattan"-Distanz zwischen Zellen i beziehungsweise j auf Positionen k beziehungsweise l und in Ausprägungen a beziehungsweise c. Analog zählen die Koeffizienten $o'(i,k,a,j,l,c)$ die Anzahl von überlappenden Basiszellen, wenn i, j in Ausprägung a, c Basiszellen k, l zugewiesen werden.

(e) Das **Floorplaning-Problem** stellt eine Verallgemeinerung des Plazierungsproblems dar. Insbesondere steht für die Zellen mehr als eine Realisierung zur Verfügung. Desweiteren ist beim Floorplaning-Problem der Master nicht fest vorgegeben. Als weiteres Optimalitätskriterium bleibt deshalb die für den Entwurf benötigte Fläche zu berücksichtigen. In Kapitel 1 dieser Arbeit haben wir bereits darauf hingewiesen, daß der Packungsaspekt (Fläche der Plazierung) bei anderen Entwurfsstilen ebenfalls von Bedeutung ist. Wir wollen daher in diesem Unterpunkt zeigen, wie unser Modell modifiziert werden muß, um die von der Plazierung benötigten Fläche in unseren Ansatz zu integrieren.

Zunächst berechnen wir untere und obere Schranken für die Maße des Masters.
Wenn — wie beispielsweise beim Standardzellenentwurf — das Ausmaß des Masters in einer Richtung bestimmt ist (o.B.d.A. die Höhe), so teilen wir die Gesamtfläche der Zellen durch diesen Wert. Das Resultat ist offensichtlich eine untere Schranke für die Größe des Masters in der anderen Richtung. Sind sowohl Breite als auch Höhe des Masters nicht vorgegeben, so ist sicherlich die Gesamtfläche der Zellen untere Schranke für die von der Plazierung benötigten Fläche. In diesem Fall gehen wir davon aus, daß unser Plazierungsverfahren keine Lösung liefert, so daß Breite und Höhe der Plazierungsfläche kleiner als die Kantenlängen des Quadrats mit obigem Flächeninhalt sind.
Zur Berechnung einer (trivialen) oberen Schranke für die Breite (Höhe) des Masters gehen wir folgendermaßen vor:
Wir bestimmen für jede Zelle die maximale Höhe (Breite) unter all ihren Realisierungen. Die Gesamtsumme der soeben berechneten Höhen- (Breiten-) Werte liefert eine obere Schranke für die Höhe (Breite) des Masters. Bezeichne $w_{max}(w_{min})$ bzw. $h_{max}(h_{min})$ die maximale (minimale) Breite bzw. Höhe des Masters. Den Master mit maximalen Ausmaßen werden wir im folgenden stets als maximalen Master bezeichnen.

Zunächst gehen wir davon aus, daß der maximale Master in Basiszellen eingeteilt ist. Ferner wird eine Instanz des Plazierungsproblems (Floorplaning-Problems), wie zu Beginn des Abschnitts (3.1) beschrieben, charakterisiert. Die Zulässigkeitsmengen $Z(.)$ beziehen sich dabei auf den maximalen Master. Dadurch ist sichergestellt, daß die Zahlen $w_{max} - w_{min}$ und $h_{max} - h_{min}$ in der Kodierungslänge des Inputs polynomiale Zahlen darstellen.

Bezeichne $A(i)$ die Menge der Realisierungen der Zelle $i \in N$. Wie bereits in den vorigen Unterpunkten dargestellt, führen wir für alle $i \in N$, $k \in Z(i)$, $a \in A(i)$ Entscheidungsvariablen $x_{ik}^{a} \in \{0,1\}$ ein. Diese geben an, in welcher Realisierung eine Zelle welcher Basiszelle zugewiesen wird. Ferner bezeichnen wiederum die Koeffizienten $d(i,k,a,j,l,c)$ bzw. $o'(i,k,a,j,l,c)$ die "shortest Manhattan"-Distanz bzw. die Anzahl Überlappungen, wenn Zellen i und j in Ausprägungen a und c Basiszellen k und l zugewiesen werden.

Wir definieren nun Variablen v^s und h^u für $s = 1, 2, \ldots, w_{max} - w_{min} + 1$ und $u = 1, 2, \ldots, h_{max} - h_{min} + 1$. Die Variablenzuweisung $v^s = 1$ ($h^u = 1$) ist so zu interpretieren, daß die Breite (Höhe) des Masters gleich $s + w_{min} - 1$ ($u + h_{min} - 1$) ist. Dies impliziert, daß für die neu definierten Variablen die folgenden Nebenbedingungen gelten:

$$\sum_{s=1}^{w_{max}-w_{min}+1} v^s = 1,$$

$$\sum_{u=1}^{h_{max}-h_{min}+1} h^u = 1,$$

$$v^s \in \{0,1\}, \text{ für alle } s = 1, \ldots, w_{max} - w_{min} + 1$$

$$h^u \in \{0,1\}, \text{ für alle } u = 1, \ldots, h_{max} - h_{min} + 1.$$

Die Summe der Produkte $\sum_{s=1}^{w_{max}-w_{min}+1} \sum_{u=1}^{h_{max}-h_{min}+1} (s + w_{min} - 1)(u + h_{min} - 1) v^s h^u$ entspricht dann unter Berücksichtigung der Nebenbedingungen der Fläche des Masters.
Es bleibt schließlich noch, eine Beziehung zwischen den x_{ik}^a-Variablen und den v^s- bzw. h^u-Variablen herzustellen. Nehmen wir an, Zelle i wird in Realisierung a Basiszelle b zugewiesen, und bezeichne b_x (b_y) die horizontale (vertikale) Koordinate des Einbauplatzes b. Falls die Beziehung $b_x + w_i > s + w_{min} - 1$ gilt, so muß die Variable v^s den Wert 0 annehmen. Analog impliziert $x_{ik}^a = 1$, daß die Variable h^u den Wert 0 annehmen muß , sofern $k_y + h_i > u + h_{min} - 1$ ist.
Um diese logische Verknüpfung der Variablen zu modellieren, definieren wir eine Zahl M gemäß

$$\begin{aligned} M := & \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} \sum_{a \in A(i)} \sum_{c \in A(j)} |c_{ij} d(i,k,a,j,l,c) + \lambda_o o'(i,k,a,j,l,c)| \\ & + \sum_{i=1}^{n} \sum_{k \in Z(i)} \sum_{a \in A(i)} \sum_{r \in R} |c_{ij} d(i,k,a,r,e(r),a(r))| \\ & + \sum_{u=1}^{h_{max}-h_{min}+1} \sum_{s=1}^{w_{max}-w_{min}+1} (s + w_{min} - 1)(u + h_{min} - 1). \end{aligned}$$

Für eine Instanz des quadratischen 0/1 Problems definieren wir Koeffizienten $g^v(s,i,k,a)$ und $g^h(u,i,k,a)$ ($i \in N$, $k \in Z(i)$, $a \in A(i)$, $s = 1, \ldots, w_{max} - w_{min} + 1$, $u = 1, \ldots, h_{max} - h_{min} + 1$). Der Wert von $g^v(s,i,k,a)$ wird auf M gesetzt, wenn $b_x + w_i > s + w_{min} - 1$ gilt. Falls $b_x + w_i \leq s + w_{min} - 1$ ist, so wird $g^v(s,i,k,a)$ als 0 definiert. Analog gilt $g^h(u,i,k,a) = M$, wenn $b_y + h_i > u + h_{min} - 1$ ist. Anderenfalls setzen wir $g^h(u,i,k,a) = 0$.

b_x und b_y bezeichnen wiederum die horizontale und vertikale Koordinate der Basiszelle b.

Die Wahl der Zahl M stellt sicher, daß in einer Optimallösung obigen Modells die Variablen x_{ik}^a, h^u und v^s nie mit Werten belegt werden, so daß $g^h(u,i,k,a)x_{ik}^a h^u$ oder $g^v(s,i,k,a)x_{ik}^a v^s$ von Null verschieden ist. Somit ist garantiert, daß in der Optimallösung die Fläche exakt in die Zielfunktion eingeht.

Bezeichne $\lambda_F \in \mathbb{R}$ den Bestrafungsparameter für die Fläche, dann stellt sich eine Modellierung des Floorplaning-Problems so dar:

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k\in Z(i)} \sum_{l\in Z(j)} \sum_{a\in A(i)} \sum_{c\in A(j)} c_{ij} d(i,k,a,j,l,c) x_{ik}^a x_{jl}^c \quad +$$

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k\in Z(i)} \sum_{l\in Z(j)} \sum_{a\in A(i)} \sum_{c\in A(j)} \lambda_o o'(i,k,a,j,l,c) x_{ik}^a x_{jl}^c$$

$$+ \sum_{i=1}^{n} \sum_{k\in Z(i)} \sum_{a\in A(i)} \sum_{r\in R} c_{ij} x_{ik}^a d(i,k,a,r,e(r),a(r))$$

$$+ \lambda_F \sum_{u=1}^{h_{max}-h_{min}+1} \sum_{s=1}^{w_{max}-w_{min}+1} (s + w_{min} - 1)(u + h_{min} - 1) v^s h^u$$

$$+ \sum_{i=1}^{n-1} \sum_{k\in Z(i)} \sum_{a\in A(i)} \sum_{u=1}^{h_{max}-h_{min}+1} g^h(u,i,k,a) x_{ik}^a h^u$$

$$+ \sum_{i=1}^{n-1} \sum_{k\in Z(i)} \sum_{a\in A(i)} \sum_{s=1}^{w_{max}-w_{min}+1} g^v(s,i,k,a) x_{ik}^a v^s.$$

so daß $\sum_{k\in Z(i)} \sum_{a\in A(i)} x_{ik}^a = 1$, für alle $i = 1, \ldots n$.

$$\sum_{s=1}^{w_{max}-w_{min}+1} v^s = 1, \qquad \sum_{u=1}^{h_{max}-h_{min}+1} h^u = 1$$

$x_{ik}^a \in \{0,1\}$, für alle $i = 1, \ldots n$, $k \in Z(i)$, $a \in A(i)$,

$v^s \in \{0,1\}$, $h^u \in \{0,1\}$.

(f) Manche Anwendungen machen es erforderlich, daß ein Netz t eine bestimmte Länge L nicht über- (unter-)schreiten darf. Um diese technische Randbedingung in unserem Modell zu berücksichtigen, bestimmen wir die Zahl M als Summe aller positiven Matrixkoeffizienten minus der Summe aller negativen Matrixeinträge. Die Matrix

$Q' \in \mathbb{R}^{nm \times nm}$ definieren wir dann folgendermaßen:

$$q'_{ik,jl} = M, \text{ wenn } d(i,k,j,l) > (<)L \text{ und } i \in A_t,\ j \in A_t.$$
$$q'_{ik,jl} = c_{ij}d(i,k,j,l) + \lambda_o o(i,k,j,l), \text{ sonst.}$$

Die durch die Matrix Q' definierte Instanz von (P) modelliert schließlich obige Nebenbedingung.

Ähnlich modelliert man die Bedingung, daß Mindestdistanzen D zwischen Netzen t_1 und t_2 eingehalten werden müssen. Wir definieren eine Zahl M entsprechend und setzen

$$q'_{ik,jl} = M, \text{ wenn } d(i,k,j,l) < D \text{ und } i \in A_{t_1},\ j \in A_{t_2},$$
$$q'_{ik,jl} = M, \text{ wenn } d(i,k,j,l) < D \text{ und } i \in A_{t_2},\ j \in A_{t_1},$$
$$q'_{ik,jl} = c_{ij}d(i,k,j,l) + \lambda_o o(i,k,j,l), \text{ sonst.}$$

Damit haben wir die Präsentation unseres Plazierungsmodells abgeschlossen.

3.2 Eine graphentheoretische Formulierung von (P)

Gegeben sei eine Instanz von (P). Zur Vereinfachung der Darstellung gehen wir zunächst davon aus, daß $q_{ikik} = 0$ ist für alle $i = 1,\ldots,n,\ k \in Z(i)$. Mit der gegebenen Instanz assoziieren wir einen Graphen $G_C = (V_C, E_C, w_C)$ (den sogenannten **Cliquengraphen**) in folgender Weise:

Jeder Knoten $u \in V_C$ steht in eindeutiger Beziehung mit einer Variable x_{ik}, $i = 1,\ldots,n$, $k \in Z(i)$ und umgekehrt. Zwischen zwei Knoten $u, v \in V_C$, $u \neq v$, wobei u und v die Variablen x_{ik} und x_{jl} repräsentieren, führen wir eine Kante e genau dann ein, wenn $i \neq j$ ist. Ihr Gewicht $w_C(e) = w_C(u,v)$ ist durch den Koeffizienten $q_{ik,jl}$ der Matrix Q gegeben. Im folgenden werden wir für einen Knoten, welcher die Variable x_{ik} repräsentiert, auch die Notation ik verwenden.

Abbildung 3.2.1 zeigt den Graphen G_C für $n = 4$, $|Z(i)| = 2$ für alle $i = 1,2,3,4$.

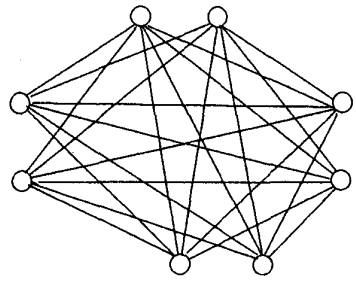


Abbildung 3.2.1

Gemäß Konstruktion gilt, daß ein Knoten ik inzident zu allen Knoten $u \in V_C$ ist mit $u \neq ik'$ für ein $k' \in Z(i)$. Dies impliziert, daß die maximalen stabilen Mengen in G_C genau die Mengen $\{ik \mid k \in Z(i)\}$ für alle $i = 1, \ldots, n$ sind. Ferner gilt, daß jede zulässige Lösung von (P) eine Abbildung
$f : \{1, 2, \ldots, n\} \rightarrow \cup_{i=1}^{n} Z(i)$ definiert, wodurch jeder Zelle $i \in \{1, 2, \ldots, n\}$ eine Basiszelle $f(i)$ aus $Z(i)$ zugewiesen wird. Der Zielfunktionswert dieser Lösung ist gegeben durch $\sum_{i=1}^{n} \sum_{j=i+1}^{n} q_{(i,f(i)),(j,f(j))}$. In dem Graphen G_C entspricht die Knotenmenge $\cup_{i=1}^{n} \{(i, f(i))\}$ einer Clique der Kardinalität n.
Umgekehrt läßt sich gemäß Definition des Graphen G_C die Menge V_C in Knotenmengen $S_1, S_2, \ldots, S_n$ zerlegen, so daß :

$$V_C = \cup_{i=1}^{n} S_i, \; S_i \cap S_j = \emptyset,$$

$$E_C(S_i) = \emptyset \text{ und } uv \in E_C \text{ für alle } u \in S_i, \; v \in S_j, \quad i, j \in \{1, 2, \ldots, n\}, \; i \neq j.$$

Das bedeutet, daß die mn Knoten in G_C die disjunkte Vereinigung von n stabilen Mengen sind. Ferner definiert der Untergraph $(S_i \cup S_j, E_C(S_i \cup S_j))$ einen vollständigen bipartiten Graphen für alle $i, j \in \{1, 2, \ldots, n\}$, $i \neq j$. Cliquen der Kardinalität n in G_C entsprechen daher Knotenmengen $\{u_1, u_2, \ldots, u_n\}$ in G_C, so daß $u_i \in S_i$ für alle $i \in \{1, 2, \ldots, n\}$. Die Variablenzuweisung $x_{u_i} = 1$, $x_u = 0$ für alle $u \in S_i \setminus \{u_i\}$, $i = 1, 2, \ldots, n$ definiert dann eine zulässige Lösung für (P).
Das Problem
"Finde unter allen Cliquen der Kardinaltät n in G_C eine minimalen Gewichts"
ist damit äquivalent zu dem quadratischen Optimierungsproblem (P).
Bei der Darstellung der Äquivalenz zwischen dem Problem (P) und obigem graphentheoretischen Problem sind wir zunächst davon ausgegangen, daß $q_{ikik} = 0$ ist für alle $i = 1, \ldots, n$, $k \in Z(i)$. Dies stellt keine Einschränkung dar, da wir anderenfalls eine Matrix Q' definieren können, welche letztere Bedingungen erfüllt. Dabei gilt:

$$q'_{ikjl} = q_{ikjl} + \frac{1}{n-1} q_{ikik} + \frac{1}{n-1} q_{jljl}$$
$$\text{für alle } i, j = 1, \ldots, n, \; i \neq j, \; k \in Z(i), \; l \in Z(j),$$
$$q'_{ikil} = q_{ikil} \text{ für alle } i = 1, \ldots, n, \; k, l \in Z(i), \; k \neq l,$$
$$q'_{ikik} = 0 \text{ für alle } i = 1, \ldots, n, \; k \in Z(i),$$

und jede zulässige Lösung von (P) mit der Matrix Q' ist auch eine Lösung von (P) bezüglich Q mit gleichem Zielfunktionswert und umgekehrt. Deshalb können wir im folgenden voraussetzen, daß für alle Instanzen von (P) obige Bedingung erfüllt ist.

3.3 Quadratische 0/1 Optimierung

Quadratische 0/1 Optimierungsprobleme haben in der Literatur beachtliches Interesse gefunden. Neben quadratischen Zuordnungsproblemen wurden insbesondere quadratische

0/1 Optimierungsprobleme ohne Nebenbedingungen studiert. Dieses Problem werden wir mit (Q) bezeichnen.
Im folgenden zeigen wir zunächst, daß sich jede Instanz von (Q) auf eine Instanz von (P) transformieren läßt. Die Umkehrung gilt mit der Einschränkung, daß bei der Transformation eine additive Konstante auftritt.

Gegeben sei $Q \in \mathbb{R}^{n \times n}$, $d \in \mathbb{R}^n$. Dann läßt sich eine Instanz von (Q) folgendermaßen angeben:

$$\begin{aligned} &\min \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j + \sum_{i=1}^{n} d_i x_i \\ &x \in \{0,1\}^n. \end{aligned} \tag{3.3.1}$$

Die Bedingung $x \in \{0,1\}^n$ impliziert zunächst, daß $x_i^2 = x_i$ für alle $i = 1, \ldots, n$. Setzt man

$$\begin{aligned} q'_{ij} &= q_{ij} \text{ für alle } i,j \in N,\ i \neq j \text{ und} \\ q'_{ii} &= q_{ii} + d_i \text{ für alle } i \in N, \end{aligned}$$

so ist (3.3.1) äquivalent zu (3.3.2).

$$\begin{aligned} &\min \sum_{i=1}^{n} \sum_{j=1}^{n} q'_{ij} x_i x_j \\ &x \in \{0,1\}^n. \end{aligned} \tag{3.3.2}$$

Wir definieren Variablen

$$x_{i1} = x_i \text{ und } x_{i2} = 1 - x_i \text{ für alle } i = 1, \ldots, n.$$

Ferner definieren wir eine Matrix $\hat{Q} \in \mathbb{Q}^{2n \times 2n}$, wobei gilt:

$$\begin{aligned} \hat{q}_{i1,j1} &= q'_{ij} \text{ für alle } i,j \in N, \\ \hat{q}_{i1,j2} &= \hat{q}_{i2,j1} = \hat{q}_{i2,j2} = 0. \end{aligned}$$

Dann gilt sicherlich, daß die durch (3.3.3) definierte Instanz des Problems (P_2) denselben Optimalwert hat wie die ursprüngliche Instanz.

$$\begin{aligned} &\min \sum_{i=1}^{n} \sum_{k=1,2} \sum_{j=1}^{n} \sum_{l=1,2} \hat{q}_{ikjl} x_{ik} x_{jl} \\ &x_{i1} + x_{i2} = 1 \text{ für alle } i \in N, \\ &x \in \{0,1\}^{2n}. \end{aligned} \tag{3.3.3}$$

Umgekehrt gilt: Jede Instanz des Problems (P) läßt sich bis auf Addition einer für die Instanz konstanten rationalen Zahl auf eine Instanz des Problems (Q) transformieren.
Gegeben eine Instanz von (P).

$$\min \sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{m} \hat{q}_{ikjl}x_{ik}x_{jl}$$

$$\sum_{k=1}^{m} x_{ik} = 1 \text{ für alle } i \in N, \tag{3.3.4}$$

$$x \in \{0,1\}^{mn}.$$

Löst man jede der n Gleichungen nach der Variable x_{im} auf, so erhält man das System (3.3.5).

$$\min \sum_{i=1}^{n}\sum_{k=1}^{m-1}\sum_{j=1}^{n}\sum_{l=1}^{m-1} \hat{q}_{ikjl}x_{ik}x_{jl} \quad +$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m-1}\sum_{j=1}^{n} \hat{q}_{ikjm}x_{ik}(1 - \sum_{l=1}^{m-1} x_{jl}) + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{m-1} \hat{q}_{imjl}(1 - \sum_{k=1}^{m-1} x_{ik})x_{jl} \quad +$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \hat{q}_{imjm}(1 - \sum_{k=1}^{m-1} x_{ik})(1 - \sum_{l=1}^{m-1} x_{jl})$$

$$\sum_{k=1}^{m-1} x_{ik} \leq 1 \text{ für alle } i \in N, \tag{3.3.5}$$

$$x_{ik} \in \{0,1\}, \text{ für alle } i \in N,\ 1 \leq k \leq m-1.$$

Die Nebenbedingungen implizieren zunächst: $x_{ik}x_{il} = 0$ für alle $i \in N,\ 1 \leq k,l \leq m-1,\ k \neq l$. Somit können wir ohne Beschränkung der Allgemeinheit annehmen, daß $\hat{q}_{ikil} = 0$ ist für alle $k \neq l$. Definiert man

$q'_{ikjl} = \hat{q}_{ikjl} - \hat{q}_{imjl} - \hat{q}_{ikjm} + \hat{q}_{imjm}$ für alle $i,j \in N,\ i \neq j,\ 1 \leq k,l \leq m-1$ und

$$q'_{ikik} = \hat{q}_{ikik} - \hat{q}_{ikim} - q_{imik} + \hat{q}_{imim} + \sum_{j=1}^{n} \hat{q}_{ikjm} + \sum_{j=1}^{n} \hat{q}_{jmik} - \sum_{j=1}^{n} \hat{q}_{imjm} - \sum_{j=1}^{n} \hat{q}_{jmim}$$

für alle $i \in N,\ 1 \leq k \leq m-1$,

$q'_{ikil} = 0$ für alle $i \in N,\ 1 \leq k,l \leq m-1,\ k \neq l$

so läßt sich (3.3.5) darstellen, gemäß:

$$\min \sum_{i=1}^{n}\sum_{k=1}^{m-1}\sum_{j=1}^{n}\sum_{l=1}^{m-1} q'_{ikjl}x_{ik}x_{jl} + \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{q}_{imjm}$$

(3.3.6)
$$\sum_{k=1}^{m-1} x_{ik} \leq 1 \text{ für alle } i \in N,$$
$$x_{ik} \in \{0,1\}, \text{ für alle } i \in N,\ 1 \leq k \leq m-1.$$

Sei

$$M = \sum_{i=1}^{n} \sum_{k=1}^{m-1} \sum_{j=1}^{n} \sum_{l=1}^{m-1} |q'_{ikjl}| + 1.$$

Ferner setzen wir $c = \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{q}_{imjm}$. Die Ungleichungen in (3.3.6) stellen sicher, daß für Zahlen k und l, $1 \leq k,l \leq m-1$, $k \neq l$ die Variablen x_{ik} und x_{il} nie beide den Wert 1 annehmen können.

Somit können wir Matrixeinträge mit

$$q_{ikjl} = q'_{ikjl} \text{ für alle } i,j \in N, 1 \leq k,l \leq m-1,\ i \neq j \text{ oder } k = l \text{ und}$$
$$q_{ikil} = M \text{ für alle } i \in N,\ 1 \leq k,l \leq m-1,\ k \neq l,$$

definieren, so daß der optimale Zielfunktionswert von (3.3.7) dem optimalen Zielfunktionswert von (3.3.6) entspricht.

$$\min \sum_{i=1}^{n} \sum_{k=1}^{m-1} \sum_{j=1}^{n} \sum_{l=1}^{m-1} q_{ikjl} x_{ik} x_{jl} \quad + \quad c$$
(3.3.7). $$x_{ik} \in \{0,1\}, \text{ für alle } i \in N,\ 1 \leq k \leq m-1,$$

Diese Behauptung ist richtig, da in einer Optimallösung von (3.3.7) für Zahlen k und l, $1 \leq k,l \leq m-1$, $k \neq l$ die Variablen x_{ik} und x_{il} nie beide den Wert 1 annehmen können. Anderenfalls wäre der Zielfunktionswert der optimalen Lösung größer oder gleich $M > 0$. Dies aber ist falsch, weil die Variablenzuweisung $x_{ik} = 0$ für alle $i \in N$, $1 \leq k \leq m-1$ eine zulässige Lösung von (3.3.7) mit Zielfunktionswert c definiert. Somit erfüllt die Optimallösung von (3.3.7) die Nebenbedingungen aus (3.3.6). Deshalb sind die Probleme (P) und (Q) bis auf Addition konstanter Zahlen gleich.

Wie bereits anfangs erwähnt, existiert eine Vielzahl von Veröffentlichungen zu quadratischer 0/1 Optimierung ohne Nebenbedingungen. In diesen werden einerseits polynomial lösbare Spezialfälle untersucht. In anderen Veröffentlichungen dagegen wird das Problem, nicht triviale obere oder untere Schranken zu bestimmen, behandelt. Die dabei verwendeten mathematischen Konzepte sind der Graphentheorie und polyedrischer Kombinatorik zuzurechnen. Die algorithmischen Methoden reichen von Schnittebenenverfahren über Lagrange Relaxierungen und "Branch and Bound" bis hin zu "Greedy"-Algorithmen.

Sei $f(x) = x^T Q x$ für $x \in \{0,1\}^n$, wobei Q eine rationale Matrix der Dimension $n \times n$ ist. Es bezeichne

$$(Q_{max}): \quad \max\{f(x) \mid x \in \{0,1\}^n\} \text{ und}$$

$$(Q_{min}): \quad \min\{f(x) \mid x \in \{0,1\}^n\}.$$

Offensichtlich gilt die Beziehung: $\min\{f(x) \mid x \in \{0,1\}^n\} = -\max\{-f(x) \mid x \in \{0,1\}^n\}$. Somit können wir in Anlehnung an die Notation in den verschiedenen Veröffentlichungen bei Gelegenheit von dem Maximierungsproblem (Q_{max}) zu dem Minimierungsproblem (Q_{min}) übergehen und umgekehrt.

Betrachten wir zunächst das Problem, nicht triviale obere Schranken für (Q_{max}) zu finden. Dabei wurde in den Veröffentlichungen [HHS84], [BCH89a] und [BCH89b] eine interessante Beziehung zwischen verschiedenen Formulierungen für (Q_{max}) hergestellt. Genauer gesagt, es besteht ein Zusammenhang zwischen den Relaxierungen verschiedener Formulierungen von (Q_{max}). Um dies zu konkretisieren, bezeichne (Q_{max}) das primale Problem. Sei ferner S eine Menge von Hyperebenen, welche die Beziehung $p(x) \geq f(x)$ für alle $x \in \{0,1\}^n$ und alle $p \in S$ erfüllt. Falls die Menge S **vollständig** ist, d.h. $f(x) = \min\{p(x) \mid p \in S\}$ für alle $x \in \{0,1\}^n$, dann ist sicherlich

$$\max\{f(x) \mid x \in \{0,1\}^n\} = \max\{\min\{p(x) \mid p \in S\} \mid x \in \{0,1\}^n\}.$$

Um eine obere Schranke für (Q_{max}) anzugeben, suchen wir nach Hyperebenen $p \in S$, für die der Wert $\max\{p(x) \mid x \in \{0,1\}^n\}$ dem Optimalwert $\max\{f(x) \mid x \in \{0,1\}^n\}$ möglichst nahe kommt, d.h. wir betrachten das Problem $\min\{\max\{p(x) \mid x \in \{0,1\}^n\} \mid p \in S\}$. Der optimale Wert letzteren Optimierungsproblems wird als **"roof dual"-Wert** bezeichnet. Andererseits existiert zu jeder quadratischen Funktion f eine quadratische Funktion g, deren Funktionswert von einem 0-1 Vektor x und seinem Komplement $\bar{x}$ abhängt. g hat dabei nur positive Koeffizienten, und es gilt die Beziehung $f(x) = c - g(x, \bar{x})$, wobei c eine konstante Zahl bezeichnet, und $\bar{x}$ dem Komplement von x entspricht. In diesem Fall gilt sicherlich, daß c eine obere Schranke für das Problem (Q_{max}) darstellt. Eine Funktion g mit obigen Eigenschaften heißt **quadratische Posiform**. Sei c_{min} die kleinste Zahl, so daß eine quadratische Posiform g existiert. Dann wird die zu c_{min} gehörende Funktion g üblicherweise als Komplement von f bezeichnet.
Schließlich betrachten wir noch die Linearisierung von (Q_{max}). Diese ist dadurch gegeben, daß zu jedem Paar von Variblen x_i, x_j, $i \neq j$ eine Variable $y_{ij} = x_i x_j = \min\{x_i, x_j\}$ eingeführt wird. Das daraus resultierende ganzzahlige Programm wird in der Literatur als **"Rhys"-Programm** bezeichnet.

Die Aussage ist nun die, daß der "roof dual"-Wert dem Optimalwert der linearen Relaxierung des Rhys-Programms entspricht, und diese beiden Werte mit c_{min} übereinstimmen. Ferner lassen sich optimale Lösungen dieser Relaxierungen in polynomialer Zeit konstruieren. Diese Beziehung gab den Anlaß zu einer Reihe von Veröffentlichungen. So wird beispielsweise in [BCH89a] und [BCH89b] obige Beziehung auf bestimme Folgen von Relaxierungen verallgemeinert.

Eine anderer Ansatz, um obere Schranken für (Q_{max}) beziehungsweise untere Schranken für (Q_{min}) zu erhalten, ergab sich aus dem direkten Studium von Polyedern, die durch Linearisierung des Problems (Q_{max}) definiert werden. Eine diesbezügliche Untersuchung wurde unter dem Namen "the boolean quadric polytope" von Padberg durchgeführt [P89].

Interessant dabei ist, daß dieses Polyeder einen Durchmesser von 1 hat, d.h. jede Ecke ist mit jeder anderen Ecke benachbart. Ferner besitzt jede Ecke der linearen Relaxierung des Polyeders Koordinaten, die alle den Wert 0, $\frac{1}{2}$ oder 1 annehmen. Andere polyedrische Studien nützen die Transformation des Problems (Q_{max}) auf das Maximalschnitt-Problem aus. Diese Reduktion geht auf [H65] zurück. In [BJR89] wurde diese Transformation verwendet, um ganzzahlige quadratische Probleme aus der Literatur zu lösen. Bezüglich theoretischer Untersuchungen für das Maximalschnitt-Polyeder existiert eine Vielfalt von Veröffentlichungen. Ohne Anspruch auf Vollständigkeit zu erheben sei diesbezüglich auf [BGM85], [BM86], [DL88], [DL89] und [BH90] verwiesen.

Der Inhalt einer Reihe von Veröffentlichungen zu quadratischer Optimierung ohne Nebenbedingungen ist ganz anderer Natur. So werden beispielsweise in [B70], [R70], [PR75], [H79], [B86], [HS86], [CHJ88] und [GKP88] Instanzen untersucht, die mit Hilfe polynomialer Algorithmen gelöst werden können. Zunächst ist es wohl bekannt, daß (Q_{min}) bzw. (Q_{max}) im allgemeinen Fall zu der Klasse der $\mathcal{NP}$-schweren Probleme zählt [SG74]. Ferner läßt sich (Q_{max}) auch schreiben als $\max\{c^T x + x^T Q' x \mid x \in \{0,1\}^n\}$. Dabei gilt $q'_{ij} = q_{ij}$ für alle $i,j = 1,\ldots,n$, $i \neq j$, $q'_{ii} = 0$ für alle $i = 1,\ldots,n$ und $c_i = q_{ii}$ für alle $i = 1,\ldots,n$. In dem Spezialfall, wenn Q nicht negativ ist und der Vektor c beliebige Werte annehmen kann, kann (Q_{max}) in polynomialer Laufzeit gelöst werden. Vergleiche hierzu [B70], [R70], [PR75], [H79] und [P89].
Mit einer Instanz von (Q_{min}) läßt sich in "natürlicher Weise" ein Graph assoziieren, indem die Knoten den Variablen entsprechen, und die Kanten die Matrixkoeffizienten, die von Null verschieden sind, repräsentieren. Falls der mit einer Instanz von (Q_{min}) assoziierte Graph eine konstante Baumweite hat, so ist das zugehörige Problem (Q_{min}) polynominal lösbar [CHJ88]. Die **Baumweite** eines Graphen G ist die kleinste Zahl k, so daß G einen **partiellen k-Baum** definiert, d.h. G kann durch Löschen von Kanten aus einem k-Baum erzeugt werden. Für einen **k-Baum** existiert eine Reihenfolge der Knoten (o.B.d.A. $v_1,\ldots,v_n$), so daß v_j in dem Untergraphen $G(v_j, v_{j+1},\ldots,v_n)$ inzident zu k Knoten ist, und diese zu v_j benachbarten Knoten einen vollständigen Untergraphen definieren. Als Spezialfall obigen Resultats kann man [B86] anführen. Denn darin wurde ein Algorithmus zur Lösung von (Q_{min}) vorgestellt. Dieser hat lineare Laufzeit in der Anzahl der Knoten, sofern der zugehörige Graph serienparallel ist, d.h. seine Baumweite ist durch die Zahl 2 beschränkt.

Im Rahmen der Darstellung der Literatur zu ganzzahliger, quadratischer Optimierung ohne Nebenbedingungen, haben wir bisher exemplarisch einige Resultate bezüglich oberer Schranken für (Q_{max}) und einige polynomial lösbare Fälle skizziert. Abschließend wollen wir noch einige Worte bezüglich unterer Schranken für (Q_{max}) verlieren.

Algorithmen in diesem Zusammenhang reichen von "Branch and Bound" über Schnittebenenverfahren und Lagrange-Relaxierung bis hin zu Heuristiken. Jedoch wurde nur ein kleiner Teil dieser Verfahren tatsächlich an Instanzen quadratischer Optimierungsprobleme unterschiedlicher Struktur getestet.

In [BM84] wurden beispielsweise nicht lineare, ganzzahlige Optimierungsprobleme allgemeinerer Struktur untersucht und über Rechenergebnisse berichtet. Umgekehrt wurde in [CHJ88] ein auf [HR68] zurückgehender Algorithmus zur Lösung quadratischer Instanzen mit sehr spezieller Struktur (der mit der Instanz assoziierte Graph definiert einen k-Baum) entwickelt. Zur Lösung beliebiger quadratischer Instanzen sind zunächst die "Branch and Bound" Verfahren von [C84] und [W85] zu erwähnen (vergleiche hierzu auch [PR88]). So erzeugt Williams [W85] zur Lösung von (Q_{max}) in jeder Iterationstiefe (des "Branch and Bound"-Baumes) über einen sogenannten "Reduktionsschritt" heuristisch eine untere Schranke. Als obere Schranke wählt er den Wert der linearen Relaxierung des "Rhys"-Programms. Eine Implementierung seiner Methode wurde an Problemen mit maximal 100 Variablen getestet, und mit Resultaten von [C84] verglichen. In [BJR89] wurden diesen Ergebnissen die Resultate eines Schnittebenenverfahrens gegenübergestellt (siehe oben). Neuere Methoden zur heuristische Lösung des Problems (Q_{min}) wurden beispielsweise in [BHS89] aufgezeigt. Die Idee dabei ist, anstelle der quadratischen Funktion f eine quadratische Posiform $\Phi(x, \bar{x})$ zu betrachten. Letztere ist charakterisiert durch die Beziehung $f(x) = c - \Phi(x, \bar{x})$ für alle $x \in \{0,1\}^n$, wobei c eine konstante Zahl bezeichnet. Ferner sind die Koeffizienten von Φ alle positiv. Die Idee dabei ist, die Koeffizienten in absteigender Reihenfolge bezüglich ihres Werts in Φ zu sortieren. In jeder Iteration wird der Koeffizient maximalen Werts (unter den noch übrigen) gewählt und die zugehörige Klausel (das Produkt von Variablen oder ihren Komplementen) auf Null gesetzt. Die sich daraus ergebenden logischen Implikationen werden ausgewertet und in die Funktion Φ eingesetzt. Dies definiert eine neue Funktion Φ', welche als Ausgangspunkt für die nächste Iteration gewählt wird. Die Laufzeit des Verfahrens ist durch $O(nt)$ beschränkt, wobei t die Anzahl der Klauseln in der ursprünglichen Posiform bezeichnet, und n der Anzahl Variablen entspricht.

Die Lösung des in Kapitel 3.2 vorgestellten Modells unter Verwendung letzteren Algorithmus würde (ohne die Transformation der Matrix auf eine Posiform zu rechnen) eine Laufzeit von $O(mnn^2) = O(mn^3)$ erfordern. Zur Lösung von Instanzen praktischer Plazierungsprobleme ($n \geq 1000, m \geq 9000$) sind daher Verfahren dieser Art leider nicht anwendbar. Dies findet auch seine Bestätigung in der Tatsache, daß in allen uns bekannten Veröffentlichungen die Algorithmen bisher nur an zufällig erzeugten Problemen mit maximal 400 Variablen getestet wurden.

Zusammenfassend läßt sich sagen, daß es bezüglich unbeschränkter quadratischer 0/1 Optimierungsprobleme viele interessante theoretische und algorithmische Konzepte gibt. Die in zahlreichen Veröffentlichungen herausgearbeiteten Beziehungen lassen sich sogar oft durch Beweise konstruktiver Natur verifizieren und in polynominale Algorithmen umsetzen. Unglücklicherweise lassen die Größenordnungen der quadratischen Instanzen, welche bei praktischen Plazierungsproblemen in dem von uns gewählten Modell auftreten, die Verwirklichung der in diesem Abschnitt dargestellten Konzepte nicht zu.

Kapitel 4 Komplexitätsaussagen bezüglich (P)

In diesem Kapitel wollen wir uns mit der Komplexität der Probleme (P) und (P_m) sowie deren Varianten für nicht-negative beziehungsweise nicht-positive Matrixkoeffizienten beschäftigen. Eine formale Definition dieser Probleme wird in Abschnitt 4.1 erfolgen. In Abschnitt 4.2 wird sich zeigen, daß alle Probleme $\mathcal{NP}$-schwer sind. Schließlich werden wir uns mit der Frage auseinandersetzen, für welche der oben aufgeführten Problemvarianten polynomiale Algorithmen existieren, die für alle Instanzen des jeweiligen Problems beweisbar gute Lösungen liefern. Dies ist der Inhalt von Abschnitt 4.3.

4.1 Das Problem (P) und seine Varianten

In Kapitel 3 haben wir eine Modellierung des Plazierungsproblems für den "Sea of cells"-Entwurfsstil vorgeschlagen. Ausgehend von dieser Formulierung definieren wir eine Instanz von (P) auf folgende Weise:

Gegeben Zahlen $n \in \mathbb{N}$, $m \in \mathbb{N}$ und eine Matrix $Q \in \mathbb{Q}^{nm \times nm}$.

$$\min x^T Q x$$

mit

$$(P) \qquad \begin{aligned} &\sum_{k=1}^{m} x_{ik} = 1, \text{ für alle } i = 1, \ldots n, \\ &x_{ik} \in \{0,1\}, \text{ für alle } i = 1, \ldots n, \ k = 1, \ldots, m. \end{aligned}$$

Für eine Zahl $m \in N$ definieren wir schließlich ein Problem (P_m).

Gegeben eine Zahl $n \in \mathbb{N}$ und eine Matrix $Q \in \mathbb{Q}^{nm \times nm}$.

$$\min x^T Q x$$

mit

$$(P_m) \qquad \begin{aligned} &\sum_{k=1}^{m} x_{ik} = 1, \text{ für alle } i = 1, \ldots n, \\ &x_{ik} \in \{0,1\}, \text{ für alle } i = 1, \ldots n, \ k = 1, \ldots, m. \end{aligned}$$

Probleme (P_m) und (P) unterscheiden sich darin, daß die Anzahl der Positionen für ein $i \in N$ beschränkt ist bzw. in den Input eingeht.

Setzen wir voraus, daß nur solche Instanzen von (P) beziehungsweise (P_m) zugelassen sind, die die Bedingungen

$$q_{ikjl} \geq 0 \text{ für alle } i,j = 1,\ldots,n,\ i \neq j \text{ und } k,l = 1,\ldots,m$$

erfüllen, so bezeichnen wir dieses Problem als (P^+) beziehungsweise (P_m^+). Analog definieren wir Probleme (P^-) und (P_m^-), wenn wir voraussetzen, daß die Matrixkoeffizienten der Instanzen von (P) beziehungsweise (P_m) alle nicht positiv sind.

4.2 Einige $\mathcal{NP}$-Vollständigkeitsresultate

(4.2.1) Satz: *(P_2) is $\mathcal{NP}$-schwer.*

Beweis: Betrachten wir das folgende Problem, wobei $Q \in \mathbb{Q}^{n\times n}$.

$$\text{(Q)} \qquad \begin{aligned} &\min x^T Q x \\ &x \in \{0,1\}^n. \end{aligned}$$

(Q) ist ein $\mathcal{NP}$schweres Optimierungsproblem [SG74].
Wir führen nun Entscheidungsvariablen x_{i1} und x_{i2} für $i = 1,2,\ldots,n$ ein. Diese lassen sich folgendermaßen interpretieren: $x_{i1} = x_i$ und $x_{i2} = 1 - x_i$. Die Matrix Q' definieren wir gemäß:

$$\begin{aligned} q'_{i1,j1} &= q_{ij} \quad \text{für alle } i,j \in \{1,2,\ldots,n\}, \\ q'_{i1,j2} &= 0 \quad \text{für alle } i,j \in \{1,2,\ldots,n\}, \\ q'_{i2,j1} &= 0 \quad \text{für alle } i,j \in \{1,2,\ldots,n\}, \\ q'_{i2,j2} &= 0 \quad \text{für alle } i,j \in \{1,2,\ldots,n\}. \end{aligned}$$

Somit entspricht

$$\begin{aligned} &\min x^T Q' x \\ x_{i1} + x_{i2} &= 1 \quad \text{für alle } i = 1,2,\ldots,n, \\ x_{i1},\ x_{i2} &\in \{0,1\} \text{ für alle } i = 1,2,\ldots,n \end{aligned}$$

einer Instanz von (P_2), deren optimale Lösungen mit denen der ursprünglichen Instanz von (QP) übereinstimmen. □

(4.2.2) Korollar:
(i) (P_2^+) ist $\mathcal{NP}$-schwer.
(ii) (P_2^-) ist $\mathcal{NP}$-schwer.

Beweis:
(i)
Gegeben sei eine Instanz von (P_2). Dann setzen wir

$$q = \min\{q_{ik,jl} \mid i,j \in \{1,2,\ldots,n\}, k,l \in \{1,2\}\}.$$

Falls $q < 0$ gilt, definieren wir eine neue Matrix $Q' \in \mathbb{Q}^{2n \times 2n}$, $Q' \geq 0$ mit der Eigenschaft, daß $q'_{ik,jl} = q_{ik,jl} - q$ für alle $i,j \in \{1,2,\ldots,n\}$ und $k,l \in \{1,2\}$.
Somit definiert

$$\begin{aligned} &\min x^T Q' x \\ &x_{i1} + x_{i2} = 1 \quad \text{für alle } i = 1,2,\ldots,n, \\ &x_{i1},\ x_{i2} \in \{0,1\} \text{ für alle } i = 1,2,\ldots,n \end{aligned}$$

eine Instanz von (P_2^+), welche die Eigenschaft $Q' \geq 0$ erfüllt. Ferner gilt: $x^T Q' x = x^T Q x - \frac{n(n-1)}{2} q$ für alle Lösungen x.

(ii)

Der Beweis von Satz 4.2.2 (ii) verläuft analog. Gegeben eine Instanz von (P_2). Dann setzen wir

$$q = \max\{q_{ik,jl} \mid i,j \in \{1,2,\ldots,n\}, k,l \in \{1,2\}\}.$$

Falls $q > 0$ gilt, definieren wir eine neue Matrix $Q' \in \mathbb{Q}^{2n \times 2n}$, $Q' \geq 0$ mit der Eigenschaft, daß $q'_{ik,jl} = q_{ik,jl} - q$ für alle $i,j \in \{1,2,\ldots,n\}$ und $k,l \in \{1,2\}$. Dies liefert uns eine Instanz von (P_2^-), für die gilt: $x^T Q' x = x^T Q x - \frac{n(n-1)}{2} q$ für alle Lösungen x.

□

Ähnlich läßt sich zeigen, daß die Probleme (P_m), (P_m^+), (P_m^-) für $m \geq 3$ sowie die Probleme (P), (P^+) und (P^-) $\mathcal{NP}$-schwer sind.

(4.2.3) Korollar:

(i) Für alle $m \geq 3$ sind die Probleme (P_m), (P_m^+) und (P_m^-) $\mathcal{NP}$-schwer.

(ii) Probleme (P), (P^+) und (P^-) sind $\mathcal{NP}$-schwer.

Beweis:

(i)

Sei $m \geq 3$. In Satz 4.2.1 bzw. 4.2.2 haben wir gezeigt, daß die Probleme (P_2), (P_2^+) und (P_2^-) $\mathcal{NP}$-schwer sind. Bezeichne I_2 eine Instanz von (P_2).

$$\begin{aligned} &\min x^T Q x \\ &x_{i1} + x_{i2} = 1 \quad \text{für alle } i = 1,2,\ldots,n, \\ &x_{i1},\ x_{i2} \in \{0,1\} \text{ für alle } i = 1,2,\ldots,n \end{aligned}$$

Wir führen nun Entscheidungsvariablen x_{ik} für $i = 1,2,\ldots,n$ und $k = 1,\ldots,m$ ein. Eine Matrix Q' definieren wir gemäß:

$$\begin{aligned} &q'_{ik,jl} = q_{ikjl} \quad \text{für alle } i,j \in \{1,2,\ldots,n\},\ k,l \in \{1,2\} \\ &q'_{ik,jl} = 0 \qquad \text{anderenfalls.} \end{aligned}$$

Dann entspricht

$$\begin{aligned} &\min\ x^T Q' x \\ &\textstyle\sum_{k=1}^m x_{ik} = 1 \text{ für alle } i = 1,2,\ldots,n, \\ &x_{ik} \in \{0,1\} \quad \text{für alle } i = 1,2,\ldots,n,\ k = 1,\ldots,m \end{aligned}$$

einer Instanz von (P_m), deren optimale Lösungen denen von I_2 entsprechen.
Analog gilt, daß (P_m^+) und (P_m^-) für $m \geq 3$ $\mathcal{NP}$-schwere Optimierungsprobleme sind.

(ii)

Da jede Instanz von (P_2) bzw. (P_2^+) bzw. (P_2^-) auch eine Instanz von (P) bzw. (P^+) bzw. (P^-) definiert, ist Teil (ii) des Korollars bewiesen.

□

4.3 Über die Existenz von ε-approximativen Algorithmen

Abschnitt (4.2) hat uns gezeigt, daß insbesondere das Plazierungsproblem in der von uns vorgeschlagenen Modellierung $\mathcal{NP}$-schwer ist. Aus theoretischer Sicht stellt sich anschließend die Frage, ob es möglich ist, Algorithmen zu entwickeln, welche in polynomialer Zeit eine beweisbar gute Lösung liefern. Der Begriff "beweisbar gut" läßt sich dabei formal folgendermaßen beschreiben:

Definition (4.3.1)
Sei $\varepsilon > 0$ vorgegeben.

(i) Wir nennen einen Algorithmus A **ε-approximativen Algorithmus** *für ein Minimierungsproblem, wenn A für jede Instanz I des Problems der folgenden Bedingung genügt:*

$$c_A^I \leq (1+\varepsilon)c_{\text{opt}}^I,$$

wobei $c_A^I \geq 0$ den Zielfunktionswert einer von A gefundenen zulässigen Lösung und $c_{\text{opt}}^I \geq 0$ den Zielfunktionswert einer optimalen Lösung für Instanz I bezeichnet. Handelt es sich um ein Maximierungsproblem, wobei $c_A^I, c_{\text{opt}}^I \leq 0$ für alle Instanzen gilt, so sprechen wir von einem ε-approximativen Algorithmus, wenn

$$c_A^I \geq (1+\varepsilon)c_{\text{opt}}^I$$

erfüllt ist. Die Zahl ε ist die **Gütegarantie** *von A.*

(ii) Ein Algorithmus A heißt **ε-approximativer Algorithmus** *für ein Maximierungsproblem, wenn für jede Instanz I des Problems gilt:*

$$c_A^I \geq (1-\varepsilon)c_{\text{opt}}^I.$$

Dabei fordern wir wieder, daß $c_A^I \geq 0$ und $c_{\text{opt}}^I \geq 0$ erfüllt ist. Handelt es sich um ein Minimierungsproblem, wobei $c_A^I, c_{\text{opt}}^I \leq 0$ für alle Instanzen gilt, so sprechen wir von einem ε-approximativen Algorithmus, wenn

$$c_A^I \leq (1-\varepsilon)c_{\text{opt}}^I$$

erfüllt ist. Die Zahl $1-\varepsilon$ wird als **worst case-Schranke** *bezeichnet (vergleiche hierzu [GJ79]).*

In diesem Abschnitt werden wir beweisen, daß, sofern wir $\mathcal{P} \neq \mathcal{NP}$ voraussetzen, keine polynomialen ε-approximativen Algorithmen für (P), (P^+), (P_m), (P_m^+), $m \geq 3$ und für (P_2) existieren. Dies gilt für alle $\varepsilon > 0$. Zur Lösung eines Problems (P_m^-), $m \geq 2$ werden wir einen $\frac{1}{m^2}$-approximativen Algorithmus vorstellen. Für die Probleme (P^-) und (P_2^+) konnten wir bisher weder zeigen, daß keine polynomialen ε-approximativen Algorithmen existieren, noch konnten wir einen solchen finden. Diese beiden Fälle bleiben daher offen.

(4.3.2) Satz: *Sei $m \geq 3$ und $\varepsilon > 0$ gegeben. Dann existiert kein ε-approximativer Algorithmus zur Lösung von (P_m^+), es sei denn $\mathcal{P} = \mathcal{NP}$.*

Beweis:
Angenommen es existiert ein polynomialer, ε-approximativer Algorithmus A für (P_m).
Das m-Färbbarkeitsproblem besteht darin, zu entscheiden, ob es für einen Graph $G = (V, E)$ mit Knotenmenge V und Kantenmenge E eine Funktion $\mu : V \to \{1, 2, \ldots, m\}$ gibt, so daß $\mu(v) \neq \mu(w)$ für alle Kanten $e = (v, w) \in E$.
Dieses Problem ist $\mathcal{NP}$-vollständig [K72].
Sei $n = |V|$. Um eine Färbung des Graphen mit m Farben darzustellen, führen wir mn Variablen ein mit folgender Interpretation:

$$x_{ik} = \begin{cases} 1, & \text{wenn } \mu(i) = k, \\ 0, & \text{anderenfalls.} \end{cases}$$

Ferner sei

$$M = \varepsilon \frac{n(n-1)}{2} + 2.$$

Für $k, l \in \{1, 2, \ldots, m\}$ und $i, j \in \{1, 2, \ldots, n\}$ definieren wir eine Matrix $Q \in \mathbb{R}^{mn \times mn}$, deren Koeffizienten sich folgendermaßen bestimmen.

$$q_{ik,jl} = \begin{cases} M, & \text{wenn } (i, j) \in E, \text{ und } k = l, \\ 1, & \text{anderenfalls.} \end{cases}$$

Offensichtlich gilt, daß G genau dann m-färbbar ist, wenn (P_m^+) mit Parametern n und Q eine zulässige Lösung besitzt, deren Zielfunktionswert $c_{\text{opt}} = \frac{n(n-1)}{2}$ ist.
Nun wenden wir Algorithmus A auf die oben definierte Instanz von (P_m^+) an.
Existiert in G eine m-Färbung, liefert A eine zulässige Lösung x mit Zielfunktionswert $c_A \leq (1+\varepsilon)\frac{n(n-1)}{2} < (\frac{n(n-1)}{2} - 1) + M$. Deshalb können die Variablen x_{ik} und x_{jl} nicht beide den Wert 1 haben, wenn $q_{ik,jl} = M$ ist. Indem wir $\mu(i) = k$ setzen, genau dann, wenn $x_{ik} = 1$ ist, repräsentiert die von A gefundene Lösung eine m-Färbung von G. Auf der anderen Seite liefert A eine Lösung mit Zielfunktionswert $c_A \geq c_{\text{opt}} \geq (\frac{n(n-1)}{2} - 1) + M$, wenn G nicht mit m Farben färbbar ist. Somit können wir Algorithmus A benutzen, um in polynomialer Zeit zu entscheiden, ob G m-färbbar ist. Dies ist unmöglich, wenn wir $\mathcal{P} \neq \mathcal{NP}$ voraussetzen. □

Satz (4.3.2) impliziert, daß es für die Probleme (P_m), $m \geq 3$, (P) und (P^+) ebenfalls keine ε-approximative Algorithmen geben kann.

(4.3.3) Korollar: *Setzt man $\mathcal{P} \neq \mathcal{NP}$ voraus, so existiert kein ε-approximativer Algorithmus zur Lösung von (P_m), $m \geq 3$, (P) und (P^+). Dies gilt für alle $\varepsilon > 0$.*

□

(4.3.4) Satz: *Es existiert kein ε-approximativer Algorithmus zur Lösung von (P_2), es sei denn $\mathcal{P} = \mathcal{NP}$. Dies gilt für alle $\varepsilon > 0$.*

Beweis:
Angenommen, es existiert ein Algorithmus A, der polynomiale Laufzeit hat und ein ε-approximativer Algorithmus für (P_2) ist.
Das 2-Partitionierungsproblem besteht darin, zu entscheiden, ob gegebene n Zahlen r_1, $r_2, \ldots, r_n$ in zwei Teilmengen $I_1, I_2 \subset \{1, 2, \ldots, n\}$ aufteilbar sind, so daß

$$I_1 \cup I_2 = \{1, 2, \ldots, n\} \text{ und}$$
$$\sum_{i \in I_1} r_i = \sum_{i \in I_2} r_i.$$

Es ist bekannt, daß dieses Problem $\mathcal{NP}$-vollständig ist [K72].
Sei $M = \varepsilon + 2$. Wir führen Entscheidungsvariablen x_{i1} und x_{i2} für alle $i = 1, 2, \ldots, n$ ein. Diese lassen sich folgendermaßen interpretieren:

$$x_{i1} = \begin{cases} 1, & \text{falls } i \text{ ein Element der Menge } I_1 \text{ ist,} \\ 0, & \text{anderenfalls,} \end{cases}$$

und

$$x_{i2} = \begin{cases} 1, & \text{falls } i \text{ ein Element der Menge } I_2 \text{ ist,} \\ 0, & \text{anderenfalls.} \end{cases}$$

Wir definieren eine Matrix $Q \in \mathbb{R}^{2n \times 2n}$ gemäß :

$$\begin{aligned}
q_{i1,i1} &= q_{i2,i2} = \tfrac{1}{n} + r_i^2 M && \text{für alle } i \in \{1, 2, \ldots, n\}, \\
q_{i1,i2} &= q_{i2,i1} = 0 && \text{für alle } i \in \{1, 2, \ldots, n\}, \\
q_{i1,j1} &= q_{i2,j2} = M r_i r_j && \text{für alle } i, j \in \{1, 2, \ldots, n\},\ i \neq j, \\
q_{i1,j2} &= q_{i2,j1} = -M r_i r_j && \text{für alle } i, j \in \{1, 2, \ldots, n\},\ i \neq j
\end{aligned}$$

und untersuchen die Probleminstanz I^2:

$$\begin{aligned}
&\min x^T Q x \\
&x_{i1} + x_{i2} = 1 && \text{für alle } i = 1, 2, \ldots, n, \\
&x_{i1}, x_{i2} \in \{0, 1\} && \text{für alle } i = 1, 2, \ldots, n.
\end{aligned}$$

Die Nebenbedingungen $x_{i1} + x_{i2} = 1$ $(i = 1, \ldots, n)$ implizieren zunächst, daß

$x_{i1}x_{i2} = 0$ ist für alle $i = 1, 2, \ldots, n$. Deshalb können wir die Zielfunktion x^TQx auch schreiben als:

$$\begin{aligned}
x^TQx &= \sum_{i=1}^{n}(x_{i1}^2 q_{i1,i1} + x_{i2}^2 q_{i2,i2}) + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} x_{i1}x_{j1}q_{i1j1} + \\
&\quad \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} x_{i2}x_{j2}q_{i2j2} + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n}(x_{i1}x_{j2}q_{i1j2} + x_{i2}x_{j1}q_{i2j1}) \\
&= \sum_{i=1}^{n}(\frac{1}{n} + Mr_i^2)(x_{i1}^2 + x_{i2}^2) + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} Mr_ir_jx_{i1}x_{j1} + \\
&\quad \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} Mr_ir_jx_{i2}x_{j2} - 2M\sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i1}x_{j2} \\
&= \sum_{i=1}^{n}\frac{1}{n}(x_{i1}^2 + x_{i2}^2) + M\Big(\sum_{i=1}^{n} r_i^2x_{i1}^2 + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i1}x_{j1} + \\
&\quad \sum_{i=1}^{n} r_i^2x_{i2}^2 + \sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i2}x_{j2} - 2\sum_{i=1}^{n}\sum_{j=1;i\neq j}^{n} r_ir_jx_{i1}x_{j2}\Big) \\
&= \sum_{i=1}^{n}\frac{1}{n}(x_{i1}^2 + x_{i2}^2) + M(\sum_{i=1}^{n} x_{i1}r_i - \sum_{i=1}^{n} x_{i2}r_i)^2.
\end{aligned}$$

Die Nebenbedingungen $x_{i1}, x_{i2} \in \{0, 1\}$ implizieren, daß $x_{i1}^2 = x_{i1}$ und $x_{i2}^2 = x_{i2}$ für alle $i = 1, 2, \ldots, n$ erfüllt ist. Desweiteren liefern die Bedingungen $x_{i1} + x_{i2} = 1$ für alle $i = 1, 2, \ldots, n$, daß

$$\sum_{i=1}^{n}\frac{1}{n}(x_{i1}^2 + x_{i2}^2) = 1.$$

Somit läßt sich (I^2) umschreiben als:

$$\begin{aligned}
&\min\ 1 + M(\sum_{i=1}^{n} x_{i1}r_i - \sum_{i=1}^{n} x_{i2}r_i)^2 \\
&x_{i1} + x_{i2} = 1 \text{ für alle } i = 1, 2, \ldots, n, \\
&x_{i1}, x_{i2} \in \{0, 1\} \text{ für alle } i = 1, 2, \ldots, n.
\end{aligned}$$

Dies ist äquivalent zu

$$\begin{aligned}
&\min\ 1 + M(\sum_{i\in I_1} r_i - \sum_{i\in I_2} r_i)^2 \\
&I_1 \cup I_2 = \{1, 2, \ldots, n\}, \\
&I_1 \cap I_2 = \emptyset, \\
&I_1, I_2 \subset \{1, 2, \ldots, n\}.
\end{aligned}$$

Falls eine 2-Partitionierung der Zahlen $r_1, r_2, \ldots, r_n$ existiert, gibt es eine optimale Lösung der oben definierten Probleminstanz (I^2) mit Zielfunktionswert $c_{\text{opt}} = 1$.

Wendet man Algorithmus A auf (P_2') an, so erhält man folgendes Resultat:
Falls eine 2-Partitionierung der gegebenen Zahlen existiert, so liefert A eine zulässige Lösung x_A mit Zielfunktionswert

$$c_A \leq (1+\varepsilon)c_{\text{opt}} = (1+\varepsilon) < M.$$

Aufgrund der Wahl der Zielfunktion für die Instanz (P_2') können wir schließlich folgern, daß

$$I_1 = \{i \in \{1,2,\ldots,n\} \mid x_{i1} = 1\}$$

und

$$I_2 = \{i \in \{1,2,\ldots,n\} \mid x_{i2} = 1\}$$

eine 2-Partitionierung von $r_1, r_2, \ldots, r_n$ definiert.
Umgekehrt, wenn es keine 2-Partitionierung von $r_1, r_2, \ldots, r_n$ gibt, gilt:

$$c_A \geq c_{\text{opt}} \geq 1 + M.$$

Somit können wir Algorithmus A benutzen, um in polynomialer Zeit zu entscheiden, ob eine gegebene Menge von Zahlen 2-partitionierbar ist. Dies ist unmöglich, wenn wir $\mathcal{P} \neq \mathcal{NP}$ voraussetzen. □

Wir haben festgestellt, daß es zur Lösung der Probleme (P_m^+), $m \geq 3$ keine ε-approximative Algorithmen gibt. Interessanterweise gilt dies nicht für Probleme (P_m^-), $m \geq 2$. Vielmehr läßt sich eine Heuristik angeben, die für alle Instanzen des Problems (P_m^-) eine worst case-Schranke von $\frac{1}{m^2}$ liefert. Damit werden wir uns in dem letzten Teil dieses Kapitelabschnitts beschäftigen. Dabei werden wir so vorgehen, daß wir zunächst den Algorithmus vorstellen. Anschließend werden wir beweisen, daß die von ihm gefundenen Lösungen eine worst case-Schranke von $\frac{1}{m^2}$ für das Problem (P_m^-) liefern.

Zur Darstellung des Algorithmus benötigen wir noch folgende Notation:
In Kapitel (3.2) haben wir mit einem Problem (P) bzw. (P_m) in eindeutiger Weise einen Graphen assoziiert. Diesen Graphen haben wir mit $G_C = (V_C, E_C, w_C)$ bezeichnet. Die Knotenmenge V_C ist dabei die Vereinigung von n stabilen Mengen der Kardinalität m. Die i-te stabile Menge haben wir mit S_i abgekürzt. Zusätzlich definieren wir die Mengen $V_i = \cup_{j=1}^{i} S_j$ und $E_i = E(V_i)$. Der von V_i induzierte Untergraph von G_C wird mit $G_i = (V_i, E_i, w)$ bezeichnet. Seien $C_1, \ldots, C_m$ Cliquen der Kardinalität $i-1$ in G_{i-1}. Dann bezeichnet $\hat{G}_i = (\hat{V}_i, \hat{E}_i, \hat{w}_i)$ den kontrahierten Graphen von G_i, welchen man erhält, indem jede Clique zu genau einem Knoten zusammengeschrumpft wird und die daraus resultierenden parallelen Kanten durch eine Kante ersetzt werden. Das Gewicht dieser Kante ist dabei die Summe der Gewichte der parallelen Kanten. Somit gilt: $\hat{G}_i$ ist ein vollständiger bipartiter Graph, dessen Knoten aus der Menge S_i und den kontrahierten Knoten bestehen.

Der im folgenden dargestellte Algorithmus (4.3.5) findet für jede Instanz des Problems (P_m^-) eine zulässige Lösung.

(4.3.5) Algorithmus:

Input: $G_C = (V_C, E_C, w_C)$, wobei $w_C(u,v) = q_{uv}$ für alle $u, v \in V_C$, $u \neq v$ gelten soll.

Output: Eine Clique C der Kardinalität n.

(1) Setze $i = 1$ und initialisiere die Mengen C_k^i wie folgt:
$C_k^i = \{ik\}$ für $k = 1, \ldots, m$.

(2) Setze $i = i + 1$.

(3) Solange $i \leq n$ ist, führe die folgenden Anweisungen durch.

(3.1) Konstruiere den bipartiten Graphen $\hat{G}_i = (\hat{V}_i, \hat{E}_i, \hat{w}_i)$. Den Knoten, der eine kontrahierte Clique C_k^{i-1} in $\hat{G}_i$ repräsentiert, bezeichnen wir mit c_k^{i-1}.

(3.2) Finde ein perfektes Matching minimalen Gewichts in $\hat{G}_i$. Bezeichne M^* dieses Matching. M^* definiert eine bijektive Abbildung $f^* : \cup_{k=1}^m c_k^{i-1} \rightarrow \{1, \ldots, m\}$, wodurch jedem Knoten c_k^{i-1} ein Index zugeordnet wird. Dieser Index charakterisiert den Endknoten der (eindeutigen) Matchingkante, die inzident zu c_k^{i-1} ist.

(3.3) Setze $C_k^i = C_k^{i-1} \cup \{(i, f^*(c_k^{i-1}))\}$ für alle $k = 1, \ldots, m$.

(3.4) Setze $i = i + 1$.

(4) Finde eine Clique C mit $w_C(E_C(C)) = \min\{w_C(E_C(C_k^n)) \mid k = 1, \ldots, m\}$.

Die Laufzeit des Algorithmus (4.3.5) ergibt sich folgendermaßen:

Die Konstruktion eines Graphen $\hat{G}_i$ benötigt eine Laufzeit von $O(mn)$. Diese Zeit ist erforderlich, um die Kantengewichte zu bestimmen.
In Schritt (3.2) des Algorithmus muß ein perfektes Matching maximalen Gewichts gefunden werden. Da der Graph $\hat{G}_i$ bipartit ist, lassen sich hierfür beispielsweise effiziente Netzwerk-Fluß-Algorithmen anwenden.

Bezeichne α die Laufzeit zur Lösung des Matching-Problems. Dann ergibt sich eine Gesamtlaufzeit von $O(mn^2 + n\alpha)$.

(4.3.6) Satz: *Gegeben $m \geq 2$. Dann liefert Algorithmus (4.3.5) für jede Instanz des Problems (P_m^-) in polynomialer Laufzeit eine Lösung mit worst case-Schranke $\frac{1}{m^2}$.*

Beweis:
Zunächst ist leicht einzusehen, daß Algorithmus (4.3.5) eine zulässige Lösung für alle Instanzen von (P_m^-) liefert. Bezeichne C_k^i $(k = 1, \ldots, m)$ die in Schritt (3.3) definierten Cliquen und C die von Algorithmus (4.3.5) gelieferte Lösung für eine Instanz von (P_m^-).
In Schritt (4) wird eine Clique maximalen Gewichts gewählt. Somit gilt:

$$w_C(E_C(C)) \leq \frac{1}{m} \sum_{k=1}^{m} w_C(E_C(C_k^n)).$$

Nach Anwendung des nachfolgenden Lemmas (4.3.7) können wir weiter folgern:

$$\sum_{k=1}^{m} w_C(E_C(C_k^n)) \leq \frac{1}{m} w_C(E_n) = \frac{1}{m} w_C(E_C) \leq \frac{1}{m} w_C(E_C(C^*)),$$

wobei C^* eine Optimallösung der Instanz von (P_m^-) bezeichnet. Insgesamt gilt dann

$$w_C(E_C(C)) \leq \frac{1}{m^2} w_C(E_C(C^*)).$$

□

(4.3.7) Lemma: *Bezeichne C_k^i die von Algorithmus (4.3.5) in Schritt (1) bzw. Schritt (3.3) definierten Mengen ($i = 1, \ldots, n,\ k = 1, \ldots, m$). Dann gilt:*

$$\sum_{k=1}^{m} w_C(E_C(C_k^i)) \leq \frac{1}{m} w_C(E_i),$$

für alle $i = 1, \ldots, n$.

Beweis:
Dies beweisen wir über Induktion nach $i \geq 2$ (der Fall $i = 1$ ist trivial).

$i = 2$:

Sei $\hat{G}_2$ der in Schritt (3.1) konstruierte Graph. Dann gilt, daß $\hat{G}_2$ ein vollständiger bipartiter Graph ist mit Knotenmenge $\hat{V}_2 = \cup_{k=1}^m \{c_k^1\} \cup S_2$.
Ferner bezeichne M^* das in Schritt (3.2) definierte, perfekte Matching in $\hat{G}_2$ minimalen Gewichts. Die das Matching definierende Abbildung kürzen wir mit f^* : $\cup_{k=1}^m c_k^1 \to \{1, \ldots, m\}$ ab. Sei $l_k^j = (f^*(c_k^1) + j - 1) \bmod m + 1$ für $k = 1, \ldots, m$ und $j = 0, \ldots, m-1$. Dann definieren die Kantenmengen $M_j = \cup_{k=1}^m \{[c_k^1, (2, l_k^j)]\}$ für $j = 0, \ldots, m-1$ eine Folge von perfekten Matchings in $\hat{G}_1$ mit folgenden Eigenschaften:

$$\begin{aligned}
&(i) && M_0 = M^*, \\
&(ii) && M_u \cap M_v = \emptyset \text{ für alle } u, v \in \{0, \ldots, m-1\},\ u \neq v, \\
&(iii) && \cup_{j=0}^{m-1} M_j = \hat{E}_2.
\end{aligned}$$

Es ist offensichtlich, daß die Bedingungen (i) und (ii) erfüllt sind. Da die Mengen M_j, $j = 0, \ldots, m-1$ disjunkt sind, ist die Gleichung $|\cup_{j=0}^{m-1} M_j| = \sum_{j=0}^{m-1} |M_j| = m^2 = |\hat{E}_2|$ erfüllt. Somit gilt auch Eigenschaft (iii): $\cup_{j=0}^{m-1} M_j = \hat{E}_2$.
Da M^* ein Matching minimalen Gewichts ist, folgt zunächst

$$\hat{w}_2(M^*) \leq \frac{1}{m} \sum_{j=0}^{m-1} \hat{w}_2(M_j) = \frac{1}{m} \hat{w}_2(\hat{E}_2).$$

Gemäß Wahl der Cliquen C_k^i in Schritt (3.3) ergibt sich das gewünschte Resultat.

$$\sum_{k=1}^{m} w_C(E_C(C_k^2)) = \hat{w}_2(M^*) \leq \frac{1}{m} w_C(E_2).$$

$i-1 \to i$:

Zunächst können wir aus der Definition der Cliquen C_k^{i-1} $(k = 1, \ldots, m)$ folgern:

$$C_k^{i-1} \cap C_l^{i-1} = \emptyset \text{ für alle } k, l \in \{1, \ldots, m\},\ k \neq l,$$
$$\cup_{k=1}^{m} C_k^{i-1} = V_{i-1}.$$

Sei nun $\hat{G}_i = (\hat{V}_i, \hat{E}_i)$ der in Schritt (3.1) konstruierte Graph. Da die Cliquen C_k^{i-1} $(k = 1, \ldots, m)$ disjunkt sind, folgt:

$$(4.3.7.1) \qquad w_C([S_i : V_{i-1}]) = \hat{w}_i(\hat{E}_i).$$

Bezeichne M^* das in Schritt (3.2) definierte, perfekte Matching in $\hat{G}_i$ minimalen Gewichts. Die das Matching definierende Abbildung kürzen wir wieder mit $f^* : \cup_{k=1}^{m} c_k^{i-1} \to \{1, \ldots, m\}$ ab. Sei $l_k^j = (f^*(c_k^{i-1}) + j - 1) \bmod m + 1$ für $k = 1, \ldots, m$ und $j = 0, \ldots, m-1$. Dann definieren analog zum Induktionsanfang die Kantenmengen $M_j = \cup_{k=1}^{m} \{(c_k^{i-1}, (2, l_k^j))\}$ für $j = 0, \ldots, m-1$ eine Folge von perfekten Matchings in $\hat{G}_i$ mit folgenden Eigenschaften:

$$M_0 = M^*,$$
$$M_u \cap M_v = \emptyset \text{ für alle } u, v \in \{0, \ldots, m-1\},\ u \neq v,$$
$$\cup_{j=0}^{m-1} M_j = \hat{E}_i.$$

Da M^* ein Matching minimalen Gewichts ist, folgt zunächst

$$(4.3.7.2) \qquad \hat{w}_i(M^*) \leq \frac{1}{m} \sum_{j=0}^{m-1} \hat{w}_i(M_j) = \frac{1}{m} \hat{w}_i(\hat{E}_i).$$

Zusammen mit Gleichung (4.3.7.1) ergibt sich:

$$(4.3.7.3) \qquad \hat{w}_i(M^*) \leq \frac{1}{m} \hat{w}_i(\hat{E}_i) = \frac{1}{m} w_C([S_i : V_{i-1}]).$$

Unter Berücksichtigung von (4.3.7.3) sowie der Induktionsannahme erhalten wir schließlich das gewünschte Resultat.

$$\begin{aligned}
\sum_{k=1}^{m} w_C(E_C(C_k^i)) = \sum_{k=1}^{m} w_C(E_C(C_k^{i-1})) + \hat{w}_i(M^*) \leq \\
\frac{1}{m} w_C(E_{i-1}) + \hat{w}_i(M^*) \leq \\
\frac{1}{m} w_C(E_{i-1}) + \frac{1}{m} w_C([S_i : V_{i-1}]) = \\
\frac{1}{m} w_C(E_i).
\end{aligned}$$

Dies schließt den Beweis ab. □

Fassen wir die Ergebnisse dieses Kapitels zusammen, so erhalten wir die folgende Tabelle 4.3.
Die drei Zeilen entsprechen den Problemen des Typs (P), (P_2) und (P_m), $m \geq 3$. In die Spalten tragen wir ein, ob als Probleminstanzen solche mit beliebigen Matrixeinträgen $(+-)$, solche mit nur nicht negativen Einträgen $(+)$ oder solche mit nur nicht positiven Einträgen $(-)$ zugelassen sind. Ein Eintrag ε bedeutet, daß für das zugehörige Problem ein $\varepsilon > 0$ existiert und ein polynomialer ε-approximativer Algorithmus zu seiner Lösung. Falls für ein Problem solch ein Algorithmus nur exisiteren kann, wenn $\mathcal{P}$=$\mathcal{NP}$ gilt, so tragen wir $\mathcal{NP}$ in die Tabelle ein. Die ungelösten Fälle werden mit einem ? gekennzeichnet.

	$+-$	$+$	$-$
(P)	$\mathcal{NP}$	$\mathcal{NP}$	?
(P_2)	$\mathcal{NP}$	?	ε
(P_m)	$\mathcal{NP}$	$\mathcal{NP}$	ε

Teil II

Heuristische Lösung des Plazierungsproblems

Wir haben gesehen, daß, sofern $\mathcal{P} \neq \mathcal{NP}$ gilt, keine Algorithmen existieren, welche für alle Instanzen des in Kapitel 3 eingeführten Problems in polynomialer Zeit beweisbar gute Lösungen liefern. In diesem Teil werden wir uns mit der praktischen Lösung des quadratischen Optimierungsproblems beschäftigen. Dazu zählen einerseits verschiedene Ansätze zur Dekomposition von Instanzen. Andererseits sollen primale Heuristiken vorgestellt werden. Dies beinhaltet auch einen Einblick in Implementierung und Datenstrukturen der Methoden. Um einen fairen Vergleich des von uns entwickelten Verfahrens mit anderen Plazierungsalgorithmen vorzunehmen, werden wir anhand vieler empirischer Tests die Parameter für unser Verfahren einstellen. Abschließend lösen wir eine lineare Relaxierung des quadratischen Optimierungsproblems.
Im Detail gliedert sich Teil II folgendermaßen:
In Kapitel 5 werden wir uns mit drei Verfahren beschäftigen, die eine Instanz des Problems (P^+) in eine Folge von Instanzen der Probleme (P_m^+) dekomponieren, wobei die Zahl m noch zu bestimmen ist. Zur Lösung des Problems (P_m^+) werden wir in Kapitel 6 verschiedene Primal- und Verbesserungsalgorithmen vorstellen. Dabei wollen wir auch auf deren Implementierung und die dazu benötigten Datenstrukturen eingehen. Bei der Lösung praktischer Plazierungsprobleme wird sich insbesondere zeigen, daß die richtige Einstellung der Parameter für die Qualität der von den Verfahren gelieferten Lösungen wesentlich ist. Zu diesem Zweck werden in Kapitel 7 eine Reihe von Tests durchgeführt. In Kapitel 8 gehen wir der Frage nach, wie das von uns entwickelte Verfahren im Vergleich mit anderen Algorithmen abschneidet. Zum Abschluß dieses Teils lösen wir eine lineare Relaxierung von (P_m^+).

Kapitel 5 Dekomposition des quadratischen 0/1 Modells

Zur Lösung praktischer Instanzen des Plazierungsproblems entwickeln wir zwei Dekompositionsverfahren. Das erste zerlegt eine Instanz des Problems (P^+) in eine Folge von Instanzen der Probleme (P_{16}^+). Dabei wird sich zeigen, daß das Verfahren im Fall $m = 16$ spezielle Eigenschaften aufweist, welche zur Lösung praktischer Problembeispiele besonders geeignet sind. Das zweite Dekompositionsverfahren zerlegt eine Instanz des Problems (P^+) in eine Folge von Instanzen der Probleme (P_9^+). Dabei müssen gegenüber der ersten Variante Modifikationen vorgenommen werden, damit die Eigenschaften des ersten Verfahrens erhalten bleiben. In dem abschließenden Abschnitt dieses Kapitels werden wir zeigen, wie man die Verteilung der Zellen in der Lösung der vorigen Iteration bei dem Entwurf des Dekompositionsverfahren für die gegenwärtige Iteration berücksichtigen kann.

Aufgrund der Komplexität des Plazierungsproblems und der Größenordnung praktischer Beispiele erscheint es vernünftig, eine Instanz rekursiv in kleinere zu zerlegen. Dieses Vorgehen ist in der Tat beim Entwurf elektronischer Schaltungen sehr weit verbreitet. Die diesbezüglich am meisten verwendete Methode besteht darin, einen k-nären Baum T, genannt Schnitt-Baum, aufzubauen. Die Wurzel von T repräsentiert die Menge aller Zellen und die Grundfläche. Sei t ein Knoten im Baum, welcher eine Fläche a_t und eine Menge von Zellen c_t repräsentiert. Dann repräsentieren die k Kinder des Knotens, $t_1, \ldots, t_k$, rechteckige Teilflächen $a_{t_1}, \ldots, a_{t_k}$ und Zellenmengen $c_{t_1}, \ldots, c_{t_k}$ mit den Eigenschaften:

$$\cup_{j=1}^{k} a_{t_j} = a_t \text{ und}$$
$$\cup_{j=1}^{k} c_{t_j} = c_t.$$

Darüber hinaus wird für einen Knoten t, welcher nicht einem Blatt in dem Baum entspricht, genau angegeben, wie die Teilflächen $a_{t_1}, \ldots, a_{t_k}$ zusammengefügt werden müssen, um das Rechteck a_t zu ergeben. Abbildung 5.1 illustriert dieses Vorgehen im Fall $k = 4$.

Solche Bäume werden gewöhnlich über "bottom-up clustering", "top down-Dekomposition" oder eine Kombination beider Techniken generiert (vergleiche [M90] für eine detaillierte Beschreibung). Dabei wird versucht, möglichst wenige Netze zwischen den Söhnen eines Knotens zu trennen (Min-Cut-Plazierungsansatz).
Obwohl dieser Ansatz in der Praxis sehr weit verbreitet ist, ergeben sich bedeutende Nachteile. Zunächst wächst die Komplexität der zugehörigen Optimierungsproblems beträchtlich mit dem Parameter k. Deshalb ist die Wahl $k = 2$ üblich. Andererseits wäre es erforderlich, $k \geq 5$ zu wählen, um eine Plazierung – wie in Abbildung 5.2 dargestellt – zu erhalten.

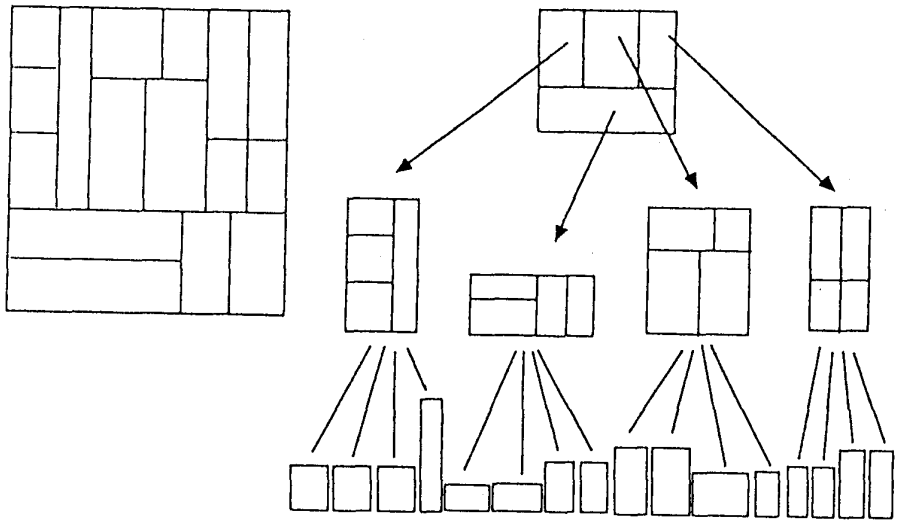


Abbildung 5.1

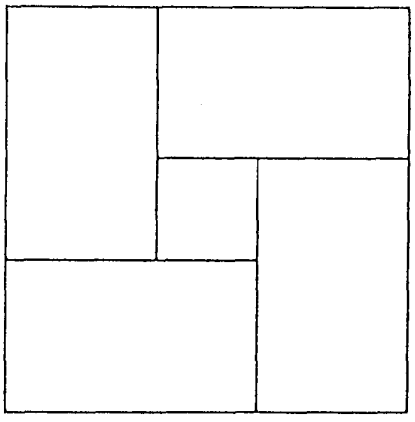


Abbildung 5.2

Ein weiterer Nachteil ist, daß die globale Sicht des ursprünglichen Problems verloren geht, da die Teilprobleme, die nicht in der Wurzel des Baumes auftreten, unabhängig voneinander gelöst werden. Dies kann bewirken, daß auf höherer Ebene Entscheidungen getroffen werden, die in den Folgeiterationen irreparabel sind.

Aus diesem Grund haben wir uns damit beschäftigt, einen neuen Dekompositionsansatz zu entwickeln, welcher die soeben dargestellten Nachteile nicht aufweist.

5.1 Ein Dekompositionsverfahren mit 16 Positionen für jede Zelle

Zur Erläuterung des Verfahrens gehen wir von einem Koordinatensystem für den Master aus. Dessen Ursprung befindet sich in der linken unteren Basiszelle. Es bezeichne w die Anzahl von Basiszellen in horizontaler Richtung und h die Anzahl von Basiszellen in vertikaler Richtung. Die Menge aller Basiszellen des Masters bezeichnen wir mit B.

Die Basiszellen seien linear durchnumeriert von $0, \ldots, hw-1$. Zur Vereinfachung der folgenden Erklärungen bezeichnen wir Basiszelle $k \in \{0, \ldots, hw-1\}$ auch als Tupel (x, y), wobei

$$x = k \bmod w,\ y = \lfloor k/w \rfloor \text{ und somit } k = y \cdot w + x.$$

Zu Beginn des Verfahrens unterteilen wir den Master in 16 gleich große Rechtecke (bis auf Rundungsfehler). Dazu wählen wir 16 Basiszellen $(\lfloor \frac{uw}{4} \rfloor, \lfloor \frac{vh}{4} \rfloor)$, $u = 0,1,2,3$, $v = 0,1,2,3$ und definieren die Menge B^1 als

$$B^1 := \cup_{u=0}^{3} \cup_{v=0}^{3} \{(\lfloor \frac{uw}{4} \rfloor, \lfloor \frac{vh}{4} \rfloor)\}$$

(vergleiche Abbildung 5.1.1).

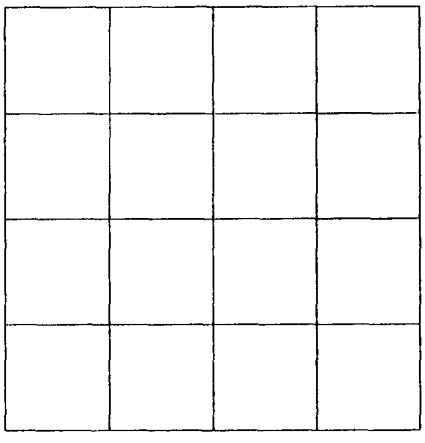


Abbildung 5.1.1

Jede Basiszelle $b = (\lfloor \frac{uw}{4} \rfloor, \lfloor \frac{vh}{4} \rfloor) \in B^1$ entspricht der linken unteren Ecke eines der 16 Rechtecke. Dies läßt sich auch so interpretieren, daß b ein Gebiet des Master, d.h. eine Menge von Basiszellen, repräsentiert. Diese Menge ist gegeben als:

$$\{(x,y) : \lfloor \frac{uw}{4} \rfloor \leq x < \lfloor \frac{(u+1)w}{4} \rfloor,\ \lfloor \frac{vh}{4} \rfloor \leq y < \lfloor \frac{(v+1)h}{4} \rfloor\}.$$

Die Menge B^1 bezeichnen wir deshalb als die Menge der zulässigen **repräsentativen Basiszellen** für die erste Iteration.

Für jede Zelle $i = 1, \ldots, n$ definieren wir noch eine Menge $Z^1(i) \subseteq B^1$, $|Z^1(i)| \leq 16$, welche aus der Menge aller in der ersten Iteration zulässigen repräsentativen Basiszellen für i besteht. Eine Basiszelle $b = (b_x, b_y) \in B^1$ ist zulässig für Zelle i, wenn gilt:

$$b_x + w_i \leq w \text{ und } b_y + h_i \leq h.$$

Wir definieren nun eine Instanz des quadratischen 0/1 Problems (P_{16}^+) (darauf werden wir noch näher eingehen). Die Variablenzuweisung $x_{ik} = 1$ wird dabei so interpretiert, daß Zelle i der Basiszelle $k \in Z^1(i)$ zugeordnet wird. (Diese Notation wurde in Kapitel 3 eingeführt.) Nach der Lösung des Problembeispiels ist schließlich jeder Zelle eine repräsentative Basiszelle $b \in Z^1(i)$ zugewiesen.

In der zweiten Iteration wird jedes der 16 Gebiete der ersten Iteration in vier gleich große Rechtecke unterteilt (bis auf Rundungsfehler). Die linken unteren Ecken dieser Rechtecke definieren die repräsentativen Basiszellen in der zweiten Iteration. Genauer gilt:

$$B^2 = \{(\lfloor \frac{uw}{8} \rfloor, \lfloor \frac{vh}{8} \rfloor),\ u = 0, 1, \ldots, 7,\ v = 0, 1, \ldots, 7\}.$$

Für $i = 1, \ldots, n$ bezeichnen wir mit $Z^2(i)$ die zulässigen repräsentativen Basiszellen für i in der zweiten Iteration. In die Konstruktion der Mengen $Z^2(.)$ werden wir dabei die Information, welche durch die Plazierung in der vorigen Iteration gegeben ist, einfließen lassen.

Angenommen Zelle i wurde in der ersten Iteration der repräsentativen Basiszelle $b_0 = (\lfloor \frac{u_0 w}{4} \rfloor, \lfloor \frac{v_0 h}{4} \rfloor) \in B^1$ zugewiesen für $u_0, v_0 \in \{0, 1, 2, 3\}$. Dann fordern wir, daß $Z^2(i)$ die folgenden drei Eigenschaften erfüllt:

(1) $b_x + w_i \leq w$ und $b_y + h_i \leq h$ für alle $(b_x, b_y) \in Z^2(i)$.

(2) $Z^2(i) \subset B^2$ und $|Z^2(i)| \leq 16$.

(3) Die Distanz (gemessen in der L_1-Norm) zwischen b_0 und jedem Element aus $Z^2(i)$ muß kleiner oder gleich der Distanz zwischen b_0 und jedem Element aus $B^2 \setminus Z^2(i)$ sein.

Definiert man

$$\begin{aligned} Z^2(i) = \{(\lfloor \tfrac{uw}{8} \rfloor, \lfloor \tfrac{vh}{8} \rfloor) \mid \lfloor \tfrac{uw}{8} \rfloor + w_i \leq w,\ \lfloor \tfrac{vh}{8} \rfloor + h_i \leq h, \\ u = \max\{0, 2u_0 - 1\}, 2u_0, \min\{w - 1, 2u_0 + 1\}, \min\{w - 1, 2u_0 + 2\}, \\ v = \max\{0, 2v_0 - 1\}, 2v_0, \min\{h - 1, 2v_0 + 1\}, \min\{h - 1, 2v_0 + 2\}\}, \end{aligned}$$

so sind die Eigenschaften (1) bis (3) erfüllt. Zusätzlich ist gewährleistet, daß die Vereinigung der Gebiete, die von den Basiszellen aus $Z^2(i)$ repräsentiert werden, das Gebiet, repräsentiert durch b_0, überdeckt (vgl. Abbildungen 5.1.2 und 5.1.3).

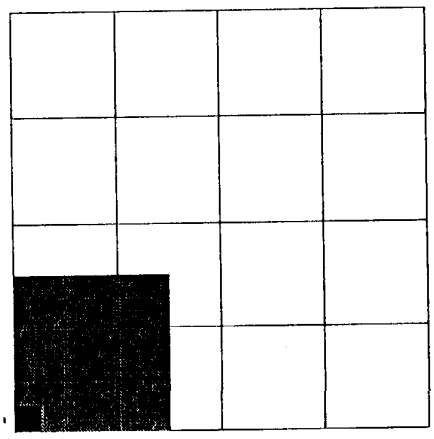


Abbildung 5.1.2

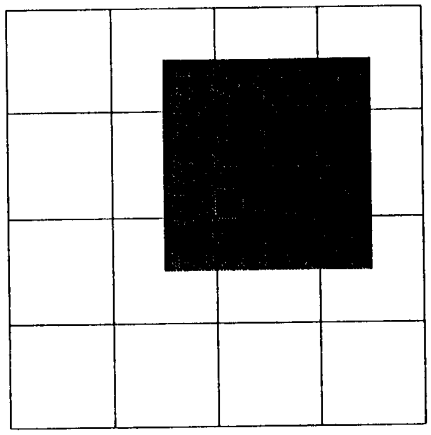
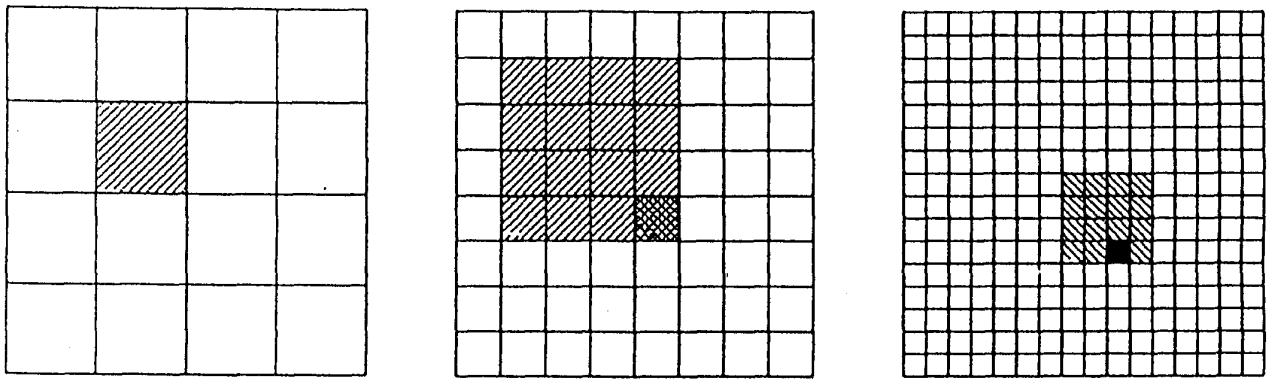


Abbildung 5.1.3

Dies wiederum stellt sicher, daß Zellen in der zweiten Iteration das Gebiet verlassen können, in welchem sie in der ersten Iteration durch Zuweisung an dessen repräsentative Basiszelle plaziert wurden. Diese Eigenschaft werden wir als die Eigenschaft des **Wanderns von Zellen** bezeichnen. Falls b_0 kein Randgebiet des Masters repräsentiert, gilt $|Z^{j+1}(i)| = 16$. In diesem Fall wird das Gebiet, repräsentiert durch b_0, von dem Gebiet, repräsentiert durch die Menge $Z^{j+1}(i)$, symmetrisch eingeschlossen. In Abbildung 5.1.2 wird für eine Zelle i, die in der ersten Iteration der linken unteren Basiszelle zugewiesen wurde, die Menge $Z^2(i)$ dargestellt. In Abbildung 5.1.3 wurde eine Zelle i in der ersten Iteration einer Basiszelle zugewiesen, welche kein Randgebiet des Masters repräsentiert. In diesem Fall schließen die durch $Z^2(i)$ repräsentierten Basiszellen (in der Abbildung grau gezeichnet) das alte Gebiet symmetrisch ein.
Der Prozeß des Unterteilens, des Konstruierens der Mengen B^j und des Generierens der Zulässigkeitsmengen $Z^j(i)$ für alle Zellen $i = 1, \ldots, n$ wird solange fortgesetzt, bis in der letzten Iteration (j_0) die Menge B^{j_0} aus allen Basiszellen des Masters besteht.

Dabei bleibt die Eigenschaft des Wanderns in jeder Iteration erhalten. Es gilt somit:
Für eine Zelle i, welche in Iteration j Basiszelle b_0 zugeordnet wurde, schließt die Menge der Basiszellen, repräsentiert durch $Z^{j+1}(i)$, das Gebiet, repräsentiert durch b_0, ein.
Ferner gilt für eine Basiszelle $b_0 \in B^j$, welche nicht ein Randgebiet des Masters repräsentiert, daß für alle Zellen i, welche in Iteration j b_0 zugewiesen wurden, $|Z^{j+1}(i)| = 16$. In diesem Fall wird das Gebiet, repräsentiert durch b_0, von dem Gebiet, repräsentiert durch die Menge $Z^{j+1}(i)$, symmetrisch eingeschlossen. Dadurch können die b_0 zugewiesenen Zellen das alte Gebiet in der darauffolgenden Iteration nach allen Richtungen hin verlassen. Diese Eigenschaft bezeichnen wir als die Eigenschaft des **symmetrischen Umschließens**.
Dies ist auch der Grund, weshalb wir in jeder Iteration 16 Basiszellen für jede Zelle zulassen.

Die Abbildungen 5.1.4 a bis c verdeutlichen nochmals diese Verfeinerungsschritte an einem Beispiel.



Abbildungen 5.1.4 a,b,c

Angenommen eine bestimmte Zelle (i) wird in der ersten Iteration der markierten repräsentativen Basiszelle in Abbildung 5.1.4 a zugewiesen. Dann sind genau die 16 markierten repräsentativen Basiszellen in Abbildung 5.1.4 b zulässig für diese Zelle. In Anlehnung an obige Definitionen ist also die Menge $Z^2(i)$ die Vereinigung dieser repräsentativen Basiszellen. Angenommen, i wird in der zweiten Iteration der speziell markierten repräsentativen Basiszelle in Abbildung 5.1.4 b zugewiesen, dann sind die für i in der dritten Iteration zulässigen repräsentativen Basiszellen entsprechend in Abbildung 5.1.4 c schraffiert.

Bei einem solchen Vorgehen lösen wir pro Iteration (j) eine Instanz des Problems (P_{16}^+). Diese läßt sich folgendermaßen darstellen:

$$\min x^T Q x$$

$$\sum_{k \in Z^j(i)} x_{ik} = 1 \quad \text{für alle } i = 1, 2, \ldots, n,$$

$$x_{ik} \in \{0, 1\} \quad \text{für alle } i = 1, 2, \ldots, n \text{ und für alle } k \in Z^j(i).$$

Bezeichne $Z^j(i)$ die Menge der für Zelle i in der momentanen Iteration (j) zulässigen repräsentativen Basiszellen. Die Variablenzuweisung $x_{ik} = 1$ wird dann so interpretiert, daß Zelle i der repräsenativen Basiszelle $k \in Z^j(i)$ zugewiesen wird. Dies gilt für alle $i = 1, \ldots, n,\ k \in Z^j(i)$. Die Koeffizienten $q_{ik,jl}$ der Matrix Q bestimmen sich als

$$c_{ij}d(i,k,j,l) + \lambda \cdot o(i,k,j,l).$$

Analog zu der in Kapitel 3 verwendeten Notation bezeichnet c_{ij} den Affinitätskoeffizienten zwischen Zellen i und j. Ferner bezeichnen $d(i,k,j,l)$ die "shortest Manhattan"-Distanz und
$o(i,k,j,l)$ die Anzahl von überlappenden Basiszellen, wenn Zellen i, j Basiszellen k, l zugewiesen werden.

Bei Verwendung dieser Zielfunktion wird in den Anfangsiterationen — gemeint sind alle Iterationen mit $|B^j| < n$ — eine gleichmäßige Verteilung der Zellen auf die repräsentativen Basiszellen angestrebt. Dies ist wichtig, da in den ersten Iterationen Überlappungen der Zellen zugelassen sein müssen (in der ersten Iteration müssen alle Zellen auf 16 repräsenative Basiszellen verteilt werden, was für mehr als 16 Zellen zwangsläufig zu Überlappungen führt). Je mehr Zellen mit ihrer linken unteren Ecke auf einer repräsentativen Basiszelle b_0 plaziert werden, desto stärker wird der Überlappungsanteil in der Zielfunktion, da alle Paare von Zellen, welche b_0 zugewiesen wurden, positiven Überlappungskoeffizienten haben. Dadurch werden ausgewogene Verteilungen der Zellen auf die repräsentativen Basiszellen belohnt.
In den Enditerationen — gemeint sind alle Iterationen mit $|B^j| > n$ — werden durch die Zielfunktion Plazierungen bevorzugt, welche überlappungsfrei sind.
Zusammenfassend läßt sich sagen, daß die Koeffizienten $o(i,k,j,l)$ in den Anfangsiterationen der Bestrafung einer ungleichmäßigen Verteilung (globale Überlappung) der Zellen auf die Einbauplätze dienen. In den Enditerationen wird die lokale Überlappung von Zellen bestraft.

5.2 Ein Dekompositionsverfahren mit 9 Positionen für jede Zelle

In Abschnitt 5.1 haben wir ein Verfahren vorgestellt, welches das Plazierungsproblem (P^+) in eine Folge von Problemen (P^+_{16}) dekomponiert. Offensichtlich läßt sich dieser Algorithmus für beliebiges $2 \leq m \leq \ldots \leq 16$ formulieren läßt. Im folgenden werden wir mit dec_m das Verfahren bezeichnen, welches analog zu der in Abschnitt 5.1 beschriebenen Vorgehensweise das Plazierungsproblem (P^+) in eine Folge von Problemen (P^+_m) dekomponiert. Die Instanz eines Problems (P^+_m) hat $O(m^2n^2)$ viele Matrixeinträge. Geht man davon aus, daß jede Heuristik zur Lösung eines Problembeispiels von (P^+_m) jeden Matrixkoeffizienten mindestens einmal berücksichtigt, so ist die Laufzeit für die heuristische Lösung eines solchen Problembeispiels direkt proportional zu der Anzahl Matrixeinträge, also $O(m^2n^2)$. Somit ergibt sich eine Gesamtlaufzeit $K(dec_m)$ zur Lösung einer Instanz von (P) als:

$$K(dec_m) = O((1 + \lceil log_4 \frac{ab}{m} \rceil)m^2n^2c).$$

b bzw. a bezeichnet die Anzahl von Basiszellen des Masters in horizontaler bzw. vertikaler Richtung und c die Proportionalitätskonstante der Laufzeit der Heuristik zu der Anzahl der Matrixeinträge in den Instanzen der Probleme (P_m^+). Der Ausdruck $1 + \lceil log_4 \frac{ab}{m} \rceil$ entspricht der Anzahl von Iterationen, vorausgesetzt, die Anzahl von repräsentativen Basiszellen vervierfacht sich von einer Iteration auf die nächste.

Im Fall des Testbeispiels soc_2 ergibt sich mit $b = 130$, $a = 68$, $ab = 8840$ und $n = 1022$ eine Gesamtlaufzeit $K(dec_{16}) = 1604327424 \cdot c$. Für einen Dekompositionsalgorithmus dec_9 würde sich die Gesamtlaufzeit $K(dec_9)$ auf den Wert $507619224 \cdot c$ verringern. Das wiederum entspräche einer Laufzeitersparnis von nahezu 70%. Zusammenfassend werden in Tabelle 5.2.1 die fiktiven Laufzeiten unter Verwendung eines Dekompositionsverfahren dec_m mit $m = 2, 3, \ldots, 16$ dargestellt.

Wie in Abschnitt 5.1 erläutert wurde, hat das Verfahren dec_{16} die folgende Eigenschaft:

Zelle i sei in Iteration j Basiszelle b_0 zugewiesen worden, und b_0 repräsentiere kein Randgebiet des Masters. Ferner bezeichne $Z^j(i)$ die Menge der für Zelle i in Iteration j zulässigen repräsentativen Basiszellen. Dann gilt, daß das Gebiet, repräsentiert durch $Z^{j+1}(i)$ das Gebiet, repräsentiert durch b_0, symmetrisch einschließt.

Diese Eigenschaft garantiert, daß Zelle i in den Folgeiterationen das Gebiet, repräsentiert durch b_0, nach allen Richtungen hin verlassen kann.

	$K(dec_m)$
$m = 2$	$32\ n^2c$
$m = 3$	$63\ n^2c$
$m = 4$	$112\ n^2c$
$m = 5$	$175\ n^2c$
$m = 6$	$252\ n^2c$
$m = 7$	$343\ n^2c$
$m = 8$	$448\ n^2c$
$m = 9$	$486\ n^2c$
$m = 10$	$600\ n^2c$
$m = 11$	$726\ n^2c$
$m = 12$	$864\ n^2c$
$m = 13$	$1014\ n^2c$
$m = 14$	$1176\ n^2c$
$m = 15$	$1350\ n^2c$
$m = 16$	$1536\ n^2c$

Tabelle 5.2.1

Unter Verwendung eines Dekompositionsverfahrens dec_m mit $m < 16$ geht obige Eigenschaft verloren. In Abbildung 5.2.2 wurde eine Zelle i in der ersten Iteration einer Basiszelle zugewiesen, welche kein Randgebiet des Masters repräsentiert. In diesem Fall schließen unter Verwendung des Dekompositionsverfahrens dec_9 die durch $Z^2(i)$ repräsentierten Basiszellen (in der Abbildung grau gezeichnet) das alte Gebiet nicht symmetrisch ein.

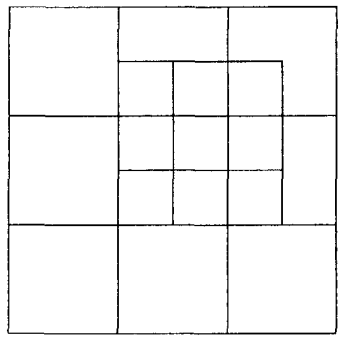


Abbildung 5.2.2

Abbildung 5.2.3 zeigt eine Plazierung des Testbeispiels soc_1, wobei in allen Iterationen das Dekompositionsverfahren dec_9 verwendet wurde.

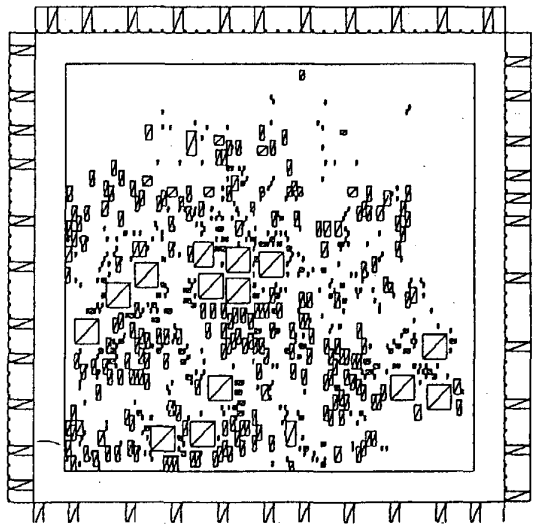


Abbildung 5.2.3

In diesem Beispiel zeigt sich deutlich, daß die Zellen nach links unten tendieren. Dies widerspricht jedoch unserem Ziel, durch eine möglichst gleichmäßige Verteilung der Zellen auf dem Master die Wahrscheinlichkeit für die Verdrahtbarkeit des Chips zu erhöhen.

In dem folgenden Teil des Kapitelabschnitts wollen wir unser bisheriges Dekompositionsverfahren so modifizieren, daß für jede Zelle in jeder Iteration maximal 9 zulässige Einbauplätze zur Verfügung stehen und zudem die obige Eigenschaft gewährleistet bleibt. Zur Darstellung des Verfahrens greifen wir auf die in Abschnitt 5.1 bereits eingeführte Notation zurück.

Wir gehen von einem Koordinatensystem für den Master aus mit Ursprung in der linken unteren Basiszelle. Es bezeichne w die Anzahl von Basiszellen in horizontaler Richtung und h die Anzahl von Basiszellen in vertikaler Richtung. Die Menge aller Basiszellen des Masters bezeichnen wir mit B.

Die Basiszellen seien linear durchnumeriert von $0, \ldots, hw-1$. Basiszelle $k \in \{0, \ldots, hw-1\}$ wird auch als Tupel (x, y) bezeichnet, wobei

$x = k \bmod w$, $y = \lfloor k/w \rfloor$ und somit $k = y \cdot w + x$. Zusätzlich bezeichnen wir den oberen, linken, unteren und rechten Rand des Masters mit to, le, bu, und ri. Genauer gilt:

$to = \{(x, y) \in B \mid y = h - 1\}$,
$bu = \{(x, y) \in B \mid y = 0\}$,
$ri = \{(x, y) \in B \mid x = w - 1\}$,
$le = \{(x, y) \in B \mid x = 0\}$.

In der ersten Iteration unterteilen wir den Master in 9, bis auf Rundungen gleich große Rechtecke. Diese werden mit $R_1^1, \ldots, R_9^1$ abgekürzt. Jedes Rechteck repräsentiert die Menge von Basiszellen, deren linke untere Ecke innerhalb des Rechtecks liegt. Für ein Rechteck R bezeichne $(u_x(R), u_y(R)) \in B$ bzw. $(o_x(R), o_y(R)) \in B$ die beiden Basiszellen, welche die linke untere bzw. rechte obere Ecke von R enthalten. Jedem der Rechtecke R_1 bis R_9 ordnen wir eine repräsentative Basiszelle gemäß der Funktion

$$f_{\{R_1^1, \ldots, R_9^1\}} : \{R_1^1, \ldots, R_9^1\} \to B \text{ zu.}$$

$$f_{\{R_1^1, \ldots, R_9^1\}}(R_i^1)/w = \lceil \tfrac{u_y(R_i^1) + o_y(R_i^1)}{2} \rceil \text{ für alle } i = 1, \ldots, 9.$$

$$f_{\{R_1^1, \ldots, R_9^1\}}(R_i^1) \bmod w = \lceil \tfrac{u_x(R_i^1) + o_x(R_i^1)}{2} \rceil \text{ für alle } i = 1, \ldots, 9.$$

Die Funktion $f_{\{R_1^1, \ldots, R_9^1\}}$ ordnet bis auf Rundung jedem Rechteck der ersten Iteration die Basiszelle zu, welche den Mittelpunkt des Rechtecks enthält. In Abbildung 5.2.4 werden die neun Rechtecke der ersten Iteration dargestellt. Die rechten oberen Ecken der zugehörigen repräsentativen Basiszellen sind durch einen Punkt gekennzeichnet.

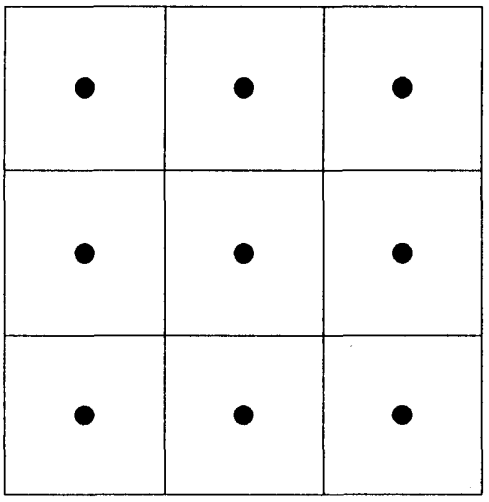


Abbildung 5.2.4

Die Menge der repräsentativen Basiszellen in der ersten Iteration (B^1) bestimmt sich dann als

$$B^1 = \cup_{j=1}^{9} \{f_{\{R_1^1, \ldots, R_9^1\}}(R_j^1)\}.$$

Die Zuweisung einer Zelle i mit Breite w_i und Höhe h_i an die repräsentative Basiszelle $f_{\{R_1^1,\ldots,R_9^1\}}(R_j^1) = (f^x_{\{R_1^1,\ldots,R_9^1\}}(R_j^1), f^y_{\{R_1^1,\ldots,R_9^1\}}(R_j^1))$, wird so interpretiert, daß die linke untere Ecke der Zelle mit einer Basiszelle (x_0, y_0) aus B zur Deckung gebracht wird. Letztere ist durch die folgende Formel eindeutig bestimmt.

$$x_0 = f^x_{\{R_1^1,\ldots,R_9^1\}}(R_j^1) - \lceil \frac{f^x_{\{R_1^1,\ldots,R_9^1\}}(R_i^1) - u_x(R_j^1)}{o_x(R_j^1) - u_x(R_j^1)} \rceil w_i.$$

$$y_0 = f^y_{\{R_1^1,\ldots,R_9^1\}}(R_j^1) - \lceil \frac{f^y_{\{R_1^1,\ldots,R_9^1\}}(R_j^1) - u_y(R_i^1)}{o_y(R_j^1) - u_y(R_j^1)} \rceil h_i.$$

Zur Veranschaulichung sei erwähnt, daß Zellen entsprechend der Lage der repräsentativen Basiszelle eines Rechtecks positioniert werden. Abbildung 5.2.5 stellt dar, wie 9 Zellen gleicher Breite und Höhe den 9 verschiedenen repräsentativen Basiszellen der ersten Iteration zugewiesen werden.

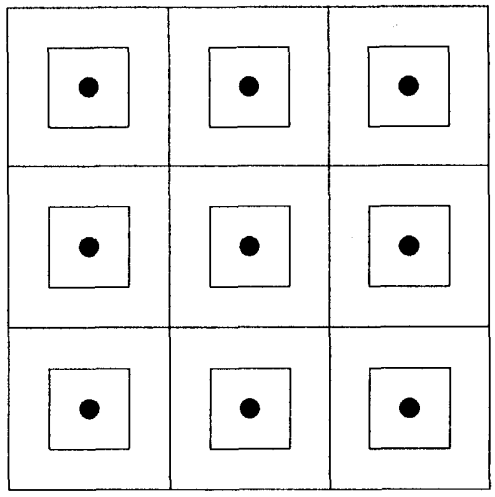


Abbildung 5.2.5

Für jede Zelle $i = 1, \ldots, n$ definieren wir die Zulässigkeitsmenge $Z^1(i)$ in der ersten Iteration. Diese besteht aus allen repräsentativen Basiszellen $b \in B^1$, so daß die Zuweisung von i an b auf dem Master realisiert werden kann.

Analog zu Abschnitt 5.1 definieren wir eine Instanz des quadratischen 0/1 Problems (P_9^+). Nach der Lösung des Problembeispiels ist schließlich jeder Zelle eine repräsentative Basiszelle $b \in Z^1(i)$ zugewiesen.

Zur korrekten Darstellung der folgenden Ausführungen verallgemeinern wir zunächst die oben definierte Funktion f in der Hinsicht, daß beliebige Zerlegungen des Masters in Rechtecke betrachtet werden.

Es sei M_R eine Zerlegung des Masters in eine Menge von paarweise disjunkten Rechtecken. Für eine solche Zerlegung definieren wir eine Funktion f_{M_R}, welche jedem Rechteck aus M_R eine Basiszelle innerhalb des Rechtecks zuordnet.

$$f_{M_R} : M_R \to B \text{ mit den Eigenschaften:}$$

$$f_{M_R}(R)/w = \begin{cases} h-1, & \text{wenn } R \cap to \neq \emptyset, \\ 0, & \text{wenn } R \cap bu \neq \emptyset, \\ \lceil \frac{u_y(R)+o_y(R)}{2} \rceil, & \text{wenn } R \cap (bu \cup to) = \emptyset. \end{cases}$$

$$f_{M_R}(R) \bmod w = \begin{cases} w-1, & \text{wenn } R \cap ri \neq \emptyset, \\ 0, & \text{wenn } R \cap le \neq \emptyset, \\ \lceil \frac{u_x(R)+o_x(R)}{2} \rceil, & \text{wenn } R \cap (le \cup ri) = \emptyset. \end{cases}$$

Dies gilt für alle $R \in M_R$.

Die Menge der Rechtecke für die zweite Iteration $\{R_1^2, R_2^2, \ldots, R_{49}^2\}$ bestimmt sich folgendermaßen:
Das Rechteck mit der Nummer R_{u+7v+1}^2 für $u = 0,\ldots,6,\ v = 0,\ldots,6$ besteht aus der Menge aller Basiszellen mit:

$$\{(x,y) \in B \mid$$
$$\max\{0, \lfloor \frac{w}{12} + \frac{(u-1)w}{6} + 1 \rfloor\} \leq x < \min\{w, \lfloor \frac{w}{12} + \frac{uw}{6} \rfloor\}$$
$$\max\{0, \lfloor \frac{h}{12} + \frac{(v-1)h}{6} + 1 \rfloor\} \leq y < \min\{h, \lfloor \frac{h}{12} + \frac{vh}{6} \rfloor\}\}.$$

In Abbildung 5.2.6 sind die Rechtecke der ersten Iteration gestrichelt gezeichnet. Die Rechtecke der zweiten Iteration werden mit durchgehenden Linien dargestellt.

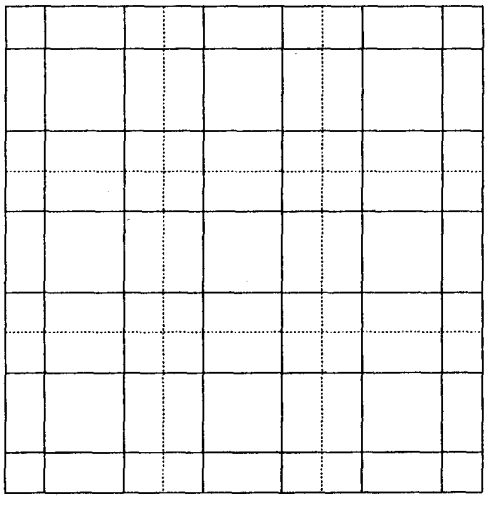


Abbildung 5.2.6

Durch Anwendung der Funktion $f_{\{R_1^2,\ldots,R_{49}^2\}} : \{R_1^2,\ldots,R_{49}^2\} \to B$ wird jedem der Rechtecke der 2. Itertaion genau eine repräsentative Basiszelle zugeordnet. Die Menge der repräsentativen Basiszellen in der zweiten Iteration (B^2) bestimmt sich dann als

$$B^2 = \cup_{i=1}^{49} \{f_{\{R_1^2,\ldots,R_{49}^2\}}(R_i^2)\}.$$

In Abbildung 5.2.7 werden die Rechtecke und die oberen, rechten Ecken der repräsentativen Basiszellen für die zweite Iteration dargestellt.

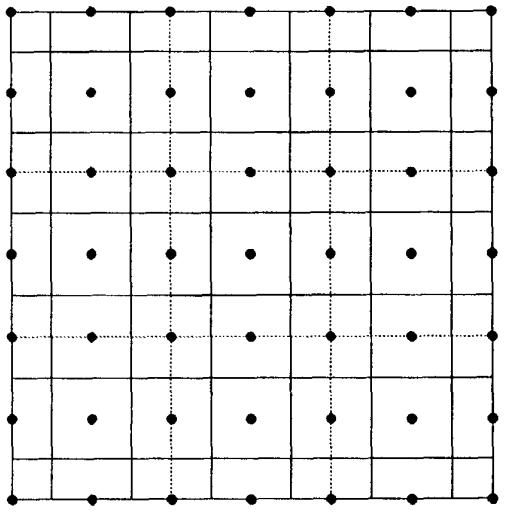


Abbildung 5.2.7

Analog zu Abschnitt 5.1 bezeichnen wir mit $Z^2(i)$ die Menge der für Zelle $i \in \{1,\ldots,n\}$ zulässigen repräsentativen Basiszellen für die zweite Iteration. Die Zuweisung einer Zelle i mit Breite w_i und Höhe h_i an die repräsentative Basiszelle $f_{\{R_1^1,\ldots,R_{49}^1\}}(R_j^2) = (f^x_{\{R_1^1,\ldots,R_{49}^1\}}(R_j^2), f^y_{\{R_1^1,\ldots,R_{49}^1\}}(R_j^2))$ interpretieren wir wiederum so, daß die linke untere Ecke der Zelle mit einer Basiszelle (x_0, y_0) aus B zur Deckung gebracht wird. Letztere berechnet sich, analog zu der ersten Iteration, gemäß

$$x_0 = f^x_{\{R_1^2,\ldots,R_{49}^2\}}(R_j^2) - \lceil \frac{f^x_{\{R_1^2,\ldots,R_{49}^2\}}(R_j^2) - u_x(R_j^2)}{o_x(R_j^2) - u_x(R_j^2)} \rceil w_i,$$

$$y_0 = f^y_{\{R_1^2,\ldots,R_{49}^2\}}(R_j^2) - \lceil \frac{f^y_{\{R_1^2,\ldots,R_{49}^2\}}(R_j^2) - u_y(R_j^2)}{o_y(R_j^2) - u_y(R_j^2)} \rceil h_i.$$

Angenommen Zelle i wurde in der ersten Iteration der repräsentativen Basiszelle b_0 des Rechtecks $R^1_{\psi_0}$ für ein $\psi_0 \in \{1,\ldots,9\}$ zugewiesen. Dann bestimmen wir Zahlen $u_0, v_0 \in \{0,1,\ldots,6\}$, so daß $f_{\{R_1^2,\ldots,R_{49}^2\}}(R^2_{u_0+7v_0+1}) = f_{\{R_1^1,\ldots,R_9^1\}}(R^1_{\psi_0})$. Solche Werte u_0 und v_0 existieren, da nach Konstruktion jede repräsentative Basiszelle der ersten Iteration auch

repräsentative Basiszelle in der zweiten Iteration ist. Somit läßt sich $Z^2(i)$ folgendermaßen darstellen:

$$\begin{aligned} Z^2(i) = &\{f_{\{R_1^2,\ldots,R_{49}^2\}}(R^2_{u+7v+1}) \mid x(u) + w_i \leq w,\ y(v) + h_i \leq h, \\ &u = \max\{0, u_0 - 1\}, u_0, \min\{w - 1, u_0 + 1\}, \\ &v = \max\{0, v_0 - 1\}, v_0, \min\{h - 1, v_0 + 1\}, \end{aligned}$$

Das Tupel $(x(u), y(v))$ bezeichnet die Basiszelle, welche mit der linken unteren Ecke der Zelle i bei Zuweisung an die repräsentative Basiszelle $f_{\{R_1^2,\ldots,R_{49}^2\}}(R^2_{u+7v+1})$ zur Deckung gebracht wird.
In Abbildung 5.2.8 wurde eine Zelle in der ersten Iteration der repräsentativen Basiszelle des mittleren Rechtecks zugewiesen. Die repräsentativen Basiszellen der neun kleineren Rechtecke definieren in diesem Beispiel die Zulässigkeitsmenge $Z^2(i)$.

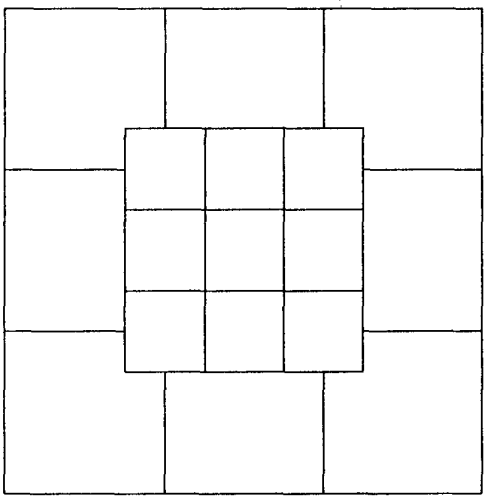


Abbildung 5.2.8

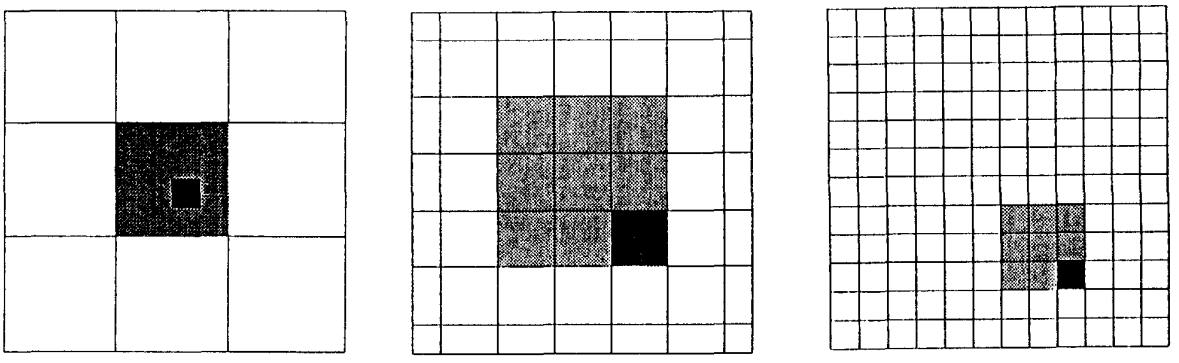
Dieses Vorgehen gewährleistet, daß die Vereinigung der Gebiete, die von den Basiszellen aus $Z^2(i)$ repräsentiert werden, das Gebiet, repräsentiert durch b_0, überdeckt (vgl. Abbildung 5.2.8).
Dies wiederum stellt sicher, daß Zellen in der nachfolgenden Iteration das Gebiet verlassen können, in welchem sie in der vorangegangenen Iteration durch Zuweisung an dessen repräsentative Basiszelle plaziert wurden. Angenommen Zelle i wurde in Iteration 1 der repräsentativen Basiszelle b_0 zugewiesen. Dann gilt $|Z^2(i)| = 9$, sofern b_0 kein Randgebiet des Masters repräsentiert. In diesem Fall wird das Gebiet, repräsentiert durch b_0, von dem Gebiet, repräsentiert durch die Menge $Z^2(i)$, symmetrisch eingeschlossen (vergleiche Abbildung 5.2.8).
Der Prozeß des Unterteilens, des Konstruierens der Mengen B^j und des Generierens der Zulässigkeitsmengen $Z^j(i)$ für alle Zellen $i = 1, \ldots, n$ wird analog zur Vorgehensweise in der zweiten Iteration fortgesetzt, bis in der letzten Iteration (j_0) die Menge B^{j_0} aus allen Basiszellen des Masters besteht.

Die Menge der Rechtecke für die j-te Iteration $\{R_1^j, R_2^j, \ldots, R_{(3j+1)^2}^j\}$ bestimmt sich allgemein folgendermaßen:
Das Rechteck mit der Nummer $R^j_{u+(3j+1)v+1}$ für $u = 0, \ldots, 3j$, $v = 0, \ldots, 3j$ besteht aus der Menge aller Basiszellen mit:

$$\{(x,y) \in B \mid$$
$$\max\{0, \lfloor\frac{w}{6j} + \frac{(u-1)w}{3j}\rfloor\} \leq x < \min\{w, \lfloor\frac{w}{6j} + \frac{uw}{3j}\rfloor\}$$
$$\max\{0, \lfloor\frac{h}{6j} + \frac{(v-1)h}{3j}\rfloor\} \leq y < \min\{h, \lfloor\frac{h}{6j} + \frac{vh}{3j}\rfloor\}$$

Das Konstruieren der Mengen B^j und das Generieren der Zulässigkeitsmengen $Z^j(i)$ für alle Zellen $i = 1, \ldots, n$ geht dann, wie oben beschrieben, analog.
Die Eigenschaften des Wanderns von Zellen und des symmetrischen Einschließens übertragen sich dabei auf alle Iterationen $j \geq 2$.

Die Abbildungen 5.2.9 a bis c verdeutlichen nochmals diese Verfeinerungsschritte an einem Beispiel.



Abbildungen 5.2.9 a,b,c

Angenommen eine Zelle (i) wird in der ersten Iteration der repräsentativen Basiszelle zugewiesen, die in Abbildung 5.2.9 a grau gezeichnet ist. Dann sind genau die repräsentativen Basiszellen der in Abbildung 5.2.9 b hellgrau gekennzeichneten Rechtecke zulässig für diese Zelle. In Anlehnung an obige Definitionen ist die Menge $Z^2(i)$ die Vereinigung dieser repräsentativen Basiszellen. Angenommen i wird in der zweiten Iteration in dem in Abbildung 5.2.9 b dunkler gekennzeichneten Rechteck plaziert, dann sind die für i in der dritten Iteration zulässigen repräsentativen Basiszellen entsprechend in Abbildung 5.2.9 c dargestellt.

5.3 Dynamische Dekomposition

Die bislang vorgestellten Dekompositionsverfahren zerlegen eine Instanz des Problems (P^+) in eine Folge von Instanzen der Probleme (P_m^+) für festes $m \in \{1, 2, \ldots, 16\}$. Die Ver-

fahren sind so konzipiert, daß die Information, welche durch die Lösung einer Instanz von (P_m^+) in der j-ten Iteration gegeben ist, die $j+1$-te Iteration beeinflußt. Diese Beeinflussung erfolgt insofern, als die Plazierung eine Zelle i in der j-ten Iteration die Zulässigkeitsmenge $Z^{j+1}(i)$ bestimmt. Bei der Konstruktion der Menge $Z^{j+1}(i)$ berücksichtigen wir aber nur die Zelle und die Position der Zelle in der vorigen Iteration, nicht aber die Gesamtsituation der vorigen Lösung. Um die Verteilung der Zellen in der Plazierung der vorigen Iteration besser berücksichtigen zu können, werden wir in diesem Abschnitt die beiden Dekompositionsverfahren aus den Abschnitten 5.1 und 5.2 modifizieren.
Betrachten wir zunächst das in Abschnitt 5.1 vorgestellte Verfahren.

Sei Zelle i mit Breite w_i und Höhe h_i in Iteration j der repräsentativen Basiszelle $b_0 = (\lfloor \frac{u_0 w}{4 \cdot 2^{j-1}} \rfloor, \lfloor \frac{v_0 h}{4 \cdot 2^{j-1}} \rfloor) \in B^j$ für ein $u_0, v_0 \in \{0, 1, \ldots, 4 \cdot 2^{j-1} - 1\}$ zugewiesen worden (vergleiche Abbildung 5.3.1). Davon ausgehend definierten wir eine Zulässigkeitsmenge für Zelle i in Iteration $j+1$. Letztere kürzen wir mit $Z_*^{j+1}(i)$ ab. Dabei gilt:

$$\begin{aligned} Z_*^{j+1}(i) = \{(\lfloor \frac{uw}{4 \cdot 2^i} \rfloor, \lfloor \frac{vh}{4 \cdot 2^j} \rfloor) \mid \lfloor \frac{uw}{4 \cdot 2^j} \rfloor + w_i \leq w,\ \lfloor \frac{vh}{4 \cdot 2^j} \rfloor + h_i \leq h, \\ u = \max\{0, 2u_0 - 1\}, 2u_0, \min\{w-1, 2u_0+1\}, \min\{w-1, 2u_0+2\}, \\ v = \max\{0, 2v_0 - 1\}, 2v_0, \min\{h-1, 2v_0+1\}, \min\{h-1, 2v_0+2\}\}, \end{aligned}$$

w bzw. h sind die Anzahl Basiszellen des Masters in horizontaler bzw. vertikaler Richtung.

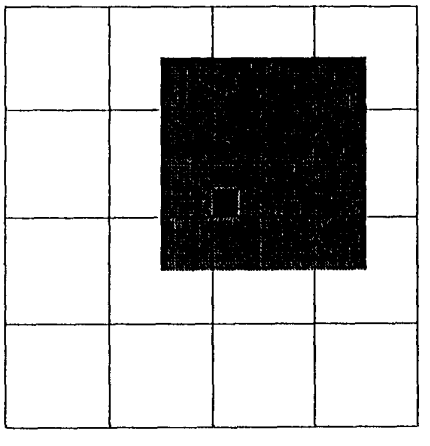


Abbildung 5.3.1

Zur Modifikation des Dekompositionsverfahrens aus Kapitel 5.1 gehen wir folgendermaßen vor:
Wir bestimmen zunächst eine Menge $\mathcal{Z}^{j+1}(i)$, die eine Teilmenge der Potenzmenge von (B^{j+1}) ist. Jedes Element aus $\mathcal{Z}^{j+1}(i)$ läßt sich als Kandidat für die Zulässigkeitsmenge von i in Iteration $j+1$ interpretieren. Auf $\mathcal{Z}^{j+1}(i)$ definieren wir eine Bewertungsfunktion $\phi_{\mathcal{Z}^{j+1}(i)} : \mathcal{Z}^{j+1}(i) \rightarrow \mathbb{R}$. Für ein $Z \in \mathcal{Z}^{j+1}(i)$ bestimmt sich der Wert $\phi_{\mathcal{Z}^{j+1}(i)}(Z)$ als Summe der Flächen derjenigen Zellen, welche in Iteration j einer der repräsentativen

Basiszellen aus Z zugewiesen wurden. Als Zulässigkeitsmenge wählen wir ein Element aus $\mathcal{Z}^{j+1}(i)$, welches die Funktion $f_{\mathcal{Z}^{j+1}(i)}$ minimiert. Falls das Minimum nicht eindeutig ist, und der Funktionswert der in Abschnitt 5.1 definierten Menge $Z_*^{j+1}(i)$ den Optimalwert annimmt, wird $Z_*^{j+1}(i)$ die Zulässigkeitsmenge für Zelle i in Iteration $j+1$.

Detailliert läßt sich dieses Vorgehen folgendermaßen beschreiben:

Bezeichne dazu

$$U(u) = \{\max\{0, u-1\}, u, \min\{w-1, u+1\}, \min\{w-1, u+2\}\ \}$$

für alle $0 \leq u \leq w \mid \lfloor \frac{uw}{4 \cdot 2^j} \rfloor + w_i \leq w$.

$$V(v) = \{\max\{0, v-1\}, v, \min\{h-1, v+1\}, \min\{h-1, v+2\}\ \}$$

für alle $0 \leq v \leq h \mid \lfloor \frac{vh}{4 \cdot 2^j} \rfloor + h_i \leq h$.

Dann definieren wir:

$$\begin{aligned}\mathcal{Z}^{j+1}(i) = &\{Z \mid Z = \{(\lfloor \frac{u'w}{4 \cdot 2^j} \rfloor, \lfloor \frac{v'h}{4 \cdot 2^j} \rfloor)\} \text{ für} \\ &u' \in U(u'),\ 2u_0 \in U(u'),\ 0 \leq u' \leq w-1,\ \lfloor \frac{u'w}{4 \cdot 2^j} \rfloor + w_i \leq w, \\ &v' \in V(v'),\ 2v_0 \in V(v'),\ 0 \leq v' \leq h-1,\ \lfloor \frac{v'h}{4 \cdot 2^j} \rfloor + h_i \leq h\}.\end{aligned}$$

Man beachte dabei, daß die Basiszelle $(\lfloor \frac{2u_0 w}{4 \cdot 2^j} \rfloor, \lfloor \frac{2v_0 h}{4 \cdot 2^j} \rfloor)$ der Position von Zelle i in Iteration j entsprach.

Die Abbildungen 5.3.2 a bis d stellen exemplarisch vier Kandidaten als Zulässigkeitsmenge für die in den Abbildungen dunkel gezeichnete Zelle dar.

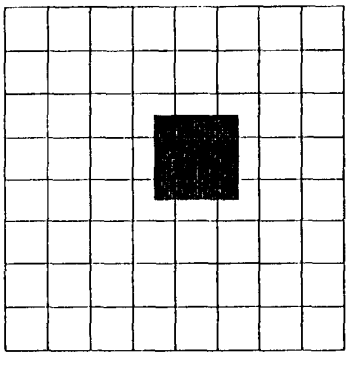


Abbildung 5.3.2 a

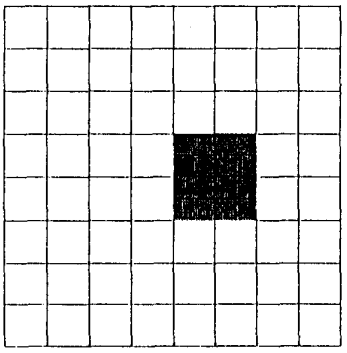


Abbildung 5.3.2 b

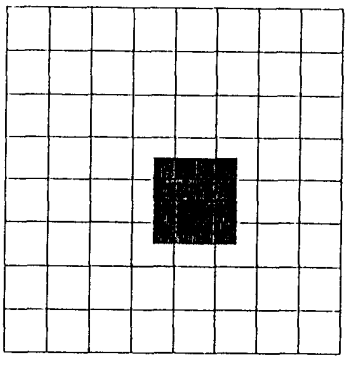


Abbildung 5.3.2 c

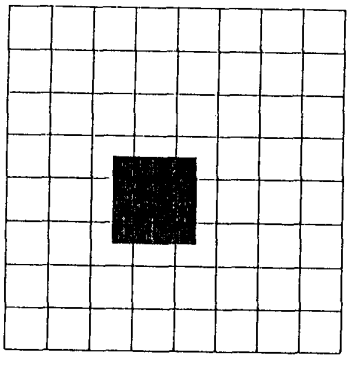


Abbildung 5.3.2 d

Den Wert der Funktion $\phi_{Z^{j+1}(i)} : Z^{j+1}(i) \rightarrow \mathbb{R}$ für ein Element $Z \in Z^{j+1}(i)$ berechnen wir gemäß:

$$\phi_{Z^{j+1}(i)}(Z) = \sum_{b \in Z \cap B^{j+1}} \sum_{1 \leq u \leq n | b_u = b} w_u h_u.$$

In obigem Ausdruck bezeichnet b_u die repräsentative Basiszelle, welcher Zelle u in Iteration j zugewiesen wurde $(u = 1, \ldots, n)$.

Zulässigkeitsmenge für i in Iteration j wird eine Menge Z^* mit den Eigenschaften (5.3.3).

$$
\begin{aligned}
&(1) Z^* \in \mathcal{Z}^{j+1}(i),\\
&(2) \phi_{\mathcal{Z}^{j+1}(i)}(Z^*) = \min\{\phi_{\mathcal{Z}^{j+1}(i)}(Z) \mid Z \in \mathcal{Z}^{j+1}(i)\},\\
&(3) \text{ wenn } \phi_{\mathcal{Z}^{j+1}(i)}(Z_*^{j+1}(i)) = \min\{\phi_{\mathcal{Z}^{j+1}(i)}(Z) \mid Z \in \mathcal{Z}^{j+1}(i)\},\\
&\text{dann setze } Z^* = Z_*^{j+1}(i).
\end{aligned} \tag{5.3.3}
$$

Wir haben bis jetzt gezeigt, wie wir das in Abschnitt 5.1 vorgestellte Dekompositionsverfahren so modifizieren können, daß die Verteilung der Zellen bereits in dem Dekompositionsverfahren Berücksichtigung findet. Die Beschränkung auf die Modifikation dieses Verfahrens erfolgte, damit die Erklärungen vereinfacht werden konnten. Zur Modifizierung des in Abschnitt 5.2 beschriebenen Verfahrens gehen wir analog vor. Mit $Z_*^{j+1}(i)$ bezeichnen wir die in Abschnitt 5.2 definierte Zulässigkeitsmenge.

Gegeben sei eine Zelle i mit Breite w_i und Höhe h_i, die in der j-ten Iteration der repräsentativen Basiszelle b_0 zugewiesen wurde. In diesem Fall wissen wir, daß es Zahlen $u_0, v_0 \in \{1, \ldots, 3j\}$ gibt, so daß

$$f_{\{R_1^{j+1},\ldots,R_{(3j+1)^2}^{j+1}\}}(R^{j+1}_{u_0+(3j+1)v_0+1}) = b_0.$$

Die Menge $\mathcal{Z}^{j+1}(i)$ bestimmt sich dann gemäß:

$$
\begin{aligned}
&\mathcal{Z}^{j+1}(i) = \{Z \mid b_0 \in Z,\\
&Z = \{f_{\{R_1^j,\ldots,R^i_{(3j+1)^2}\}}(R^j_{u+(3j+1)v+1}) \mid x(u) + w_i \le w,\ y(v) + h_i \le h,\\
&u = \max\{0, u_0 - 1\}, u_0, \min\{w - 1, u_0 + 1\},\\
&v = \max\{0, v_0 - 1\}, v_0, \min\{h - 1, v_0 + 1\}\ \}.
\end{aligned}
$$

Als Zulässigkeitsmenge wählen wir schließlich ein Element aus $\mathcal{Z}^{j+1}(i)$, welches die Eigenschaften (5.3.3) erfüllt.

Hiermit schließen wir das Kapitel über heuristische Dekompositionsverfahren zur Lösung des Plazierungsproblems ab. In Kapitel 7 werden wir insbesondere anhand praktischer Problembeispiele die Verfahren der Abschnitte 5.1 bis 5.3 testen.

Kapitel 6 Heuristische Lösung der Probleme (P_m^+)

In dem vorhergehenden Kapitel haben wir uns damit beschäftigt, eine Instanz des Plazierungsproblems rekursiv in kleinere zu zerlegen. Für reale Anwendungen sind jedoch diese kleineren Instanzen noch immer von erheblicher Größenordnung (vergleiche Kapitel 5). Ferner wissen wir aus Kapitel 4, daß diese kleineren Probleme $\mathcal{NP}$-schwer sind. Aus diesem Grund ist nicht zu erwarten, daß wir praktische Beispiele in kurzer Zeit optimal lösen können. Vielmehr werden wir uns im folgenden darauf konzentrieren, Heuristiken zur Lösung der durch Dekomposition enstandenen Probleme zu entwickeln. Dieses Kapitel gliedert sich so, daß zunächst die Grundideen der Algorithmen dargestellt werden. Dies erfolgt in Abschnitt 6.1. Anschließend werden wir uns damit befassen, wie diese Methoden effizient implementiert werden können. Dazu ist es erforderlich, Datenstrukturen einzuführen, die die spezielle Struktur der Instanzen von (P_m^+) berücksichtigen. Dies ist der Inhalt von Abschnitt 6.2. Anhand der Datenstrukturen werden wir schließlich in Abschnitt 6.3 auf die eigentliche Implementierung eingehen.

6.1 Heuristische Grundideen

Zur Darstellung der Algorithmen werden wir uns an die in Kapitelabschnitt (3.2) eingeführte Notation halten. Dort haben wir mit einem Problem (P) bzw. (P_m) in eindeutiger Weise einen Graphen $G_C = (V_C, E_C, w_C)$ assoziiert. Die Knotenmenge V_C ist dabei die Vereinigung von n stabilen Mengen der Kardinalität m. Die i-te stabile Menge haben wir mit S_i abgekürzt. Ferner bezeichnet $S(u)$ die maximale stabile Menge in G_C, zu welcher Knoten $u \in V_C$ gehört.
Aus Kapitel 3 wissen wir ferner, daß das Problem (P) bzw. (P_m) äqivalent zu dem folgenden Problem ist:

(6.1.1) Finde unter alle Cliquen der Kardinalität n in G_C eine minimalen Gewichts.

Die im folgenden präsentierten Algorithmen werden eine zulässige (nicht notwendigerweise optimale) Lösung für (6.1.1) liefern.
Die erste Heuristik ist ein "greedy"-Algorithmus. Wir beginnen mit einer Clique der Kardinalität 2. In jedem Schritt des Verfahren fügen wir einen Knoten zu der gegenwärtigen Clique hinzu. Das Verfahren endet schließlich, wenn wir eine Clique maximaler Kardinalität gefunden haben.

Heuristik 1

Input: $G_C = (V_C, E_C, w_C)$.

Output: Eine Clique Q der Kardinalität n.

(1) Sei $w_C(u_0, v_0) = \min\{w_C(u,v) \mid (u,v) \in E_C\}$.

Setze $Q = \{u_0, v_0\}$ und $\overline{Q} = S(u_0) \cup S(v_0)$.

(2) Solange $|Q| < n$ ist, führe die folgenden Anweisungen durch:

(2.1) Sei $\sum_{v \in Q} w_C(v, w_0) = \min\{\sum_{v \in Q} w_C(v, w) \mid w \in V_C \setminus \overline{Q}\}$.

(2.2) Setze $Q = Q \cup \{w_0\}$ und $\overline{Q} = \overline{Q} \cup S(w_0)$.

Offensichtlich ist die Laufzeit dieses Verfahrens im schlechtesten Fall $O(mn^3)$. In jedem Schritt der Heuristik wird ein Knoten w_0 zu der gegenwärtigen Clique hinzugefügt. Dies impliziert, daß, abgesehen von w_0, alle Knoten, die zu der stabilen Menge $S(w_0)$ gehören, simultan gelöscht werden.
Die zweite Heuristik vermeidet dieses simultane Löschen von mehr als einem Knoten. Stattdessen löschen wir aus dem Graphen in jeder Iteration einen Knoten, bis eine Clique der Kardinalität n übrigbleibt. Hierzu weisen wir jedem Knoten und jeder maximalen stabilen Menge in G_C zunächst ein Gewicht zu.
In jedem Schritt suchen wir eine maximale stabile Menge S_0 mit höchstem Gewicht und einen Knoten u_0 von S_0 mit höchstem Gewicht (unter allen Knoten, die zu S_0 gehören). Der Knoten u_0 wird anschließend aus dem Graphen G_C gelöscht. Diese Vorgehensweise wird wiederholt angewendet, bis jede maximale stabile Menge in G_C nur noch aus genau einem Knoten besteht. Die verbleibenden Knoten definieren eine Clique der Kardinalität n.
Bezeichne $\delta(v)$ die Menge von Kanten in G_C, welche inzident zu $v \in V_C$ sind.

Heuristik 2

Input: $G_C = (V_C, E_C, w_C)$.

Output: Eine Clique Q der Kardinalität n.

(1) Weise jedem Knoten $v \in V_C$ ein Knotengewicht zu, gemäß der Formel:
$g(v) = \frac{1}{|\delta(v)|} \sum_{(u,v) \in E_C} w_C(u, v)$.

(2) Sei $\gamma(S) = \max_{v \in S} g(v) / \min_{v \in S} g(v)$ für alle maximalen stabilen Mengen S in G_C . Setze $Q = \emptyset$.

(3) Solange $|V_C| > n$ ist, führe die folgenden Anweisungen durch:

(3.1) Finde eine maximale stabile Menge S_0 mit
$\gamma(S_0) = \max\{\gamma(S) \mid S$ ist eine maximale stabile Menge in $G_C, |S| \geq 2\}$. Sei ferner v_0 ein Knoten in S_0 mit $g(v_0) = \max_{v \in S_0} g(v)$.

(3.2) Setze $V_C = V_C \setminus \{v_0\}$ und $E_C = E_C \setminus \delta(v_0)$.

(3.3) Update: Aktualisiere die Werte $g(v)$ für alle $v \in V_C$ und berechne $\gamma(S)$ für alle maximalen stabilen Mengen S in G_C.

(4) Setze $Q = V_C$.

Offensichtlich beträgt die Laufzeit des Verfahrens im schlechtesten Fall $O(m^2 n^2)$.

Die zwei Heuristiken, welche wir bis jetzt vorgestellt haben, lassen sich auf alle dekomponierten Probleme anwenden. Die dritte Heuristik ist dagegen entwickelt worden, um Instanzen des Problems (P_m^+) zu lösen, wobei bereits Information aus vorangegangenen Iterationen zur Verfügung steht (d.h. das erste dekomponierte Problem muß gelöst sein). Angenommen, wir haben eine Iteration bereits gerechnet. Bezeichne o_i die Gesamtanzahl von Basiszellen, welche von i und den anderen Zellen in der Plazierung der vorigen Iteration gemeinsam überdeckt werden ($i = 1, \ldots, n$). Die Basiszelle, der i in der Lösung der vorigen Iteration zugewiesen wurde, wird in dem neuen Graphen G_C der gegenwärtigen Iteration durch einen Knoten ik repräsentiert. Wir verwenden nun die Zahl o_i als Gewicht der maximalen stabilen Menge S_i und sortieren diese maximalen stabilen Mengen in absteigender Reihenfolge bezüglich ihres Gewichts. Ohne Beschränkung der Allgemeinheit sei $S_1, S_2, \ldots, S_n$ die geordnete Folge.

Wir beginnen mit m Cliquen der Kardinalität 1, welche den m Knoten der maximalen stabilen Menge S_1 entsprechen. In einem Schritt des Verfahrens erweitern wir jede dieser Cliquen um einen Knoten. Somit werden wir mit m Cliquen der Kardinalität n enden. Anschließend versuchen wir, jede dieser m Cliquen durch Anwendung von 1er-Austausch-Schritten zu verbessern. Genauer wird ein Knoten in einer Clique durch einen anderen ersetzt, wenn dies zu einer Veringerung des Gewichts der Clique führt. Schließlich wählen wir unter den m Cliquen eine minimalen Gewichts.

Heuristik 3

Input: $G_C = (V_C, E_C, w_C)$.

Output: Eine Clique Q der Kardinalität n.

(1) Seien $\{v_i^1, v_i^2, \ldots, v_i^m\}$ die Knoten der stabilen Menge S_i. Setze $Q_1^k = \{v_1^k\}$, und $q(Q_1^k) = 0$ für alle $k = 1, \ldots, m$. (Q_i^k bezeichnet die k-te Clique im i-ten Schritt.) Setze $i = 2$.

(2) Solange $i \leq n$ ist, führe die folgenden Anweisungen durch:

(2.1) Setze $I = \{1, \ldots, m\}$.

(2.2) Für alle $k = 1, \ldots, m$ führe folgende Anweisungen durch:

(2.2.1) Berechne
$$q_i^k = q(Q_{i-1}^k) + \min_{l \in I}\Big(\sum_{u \in Q_{i-1}^k} w_C((u, v_i^l))\Big).$$

(2.2.2) Sei $l_0 \in I$ ein Index, so daß $q_i^k = q(Q_{i-1}^k) + \sum_{u \in Q_{i-1}^k} w_C((u, v_i^{l_0}))$ gilt. Setze $q(Q_i^k) = q_i^k$.

(2.2.3) Setze $I = I \setminus \{l_0\}$ und $Q_i^k = Q_{i-1}^k \cup \{v_i^{l_0}\}$

(2.3) Setze $i = i + 1$.

(3) Setze $Q_k = Q_n^k$ für alle $k = 1, \ldots, m$.

(4) Führe für jede der Cliquen Q_k, $k = 1, 2, \ldots, m$, 1er-Austausch-Schritte durch, solange das Gewicht $q(Q_k)$ verringert werden kann.

(5) Wähle eine Clique Q mit $q(Q) = \min\{q(Q_k) \mid k = 1, 2, \ldots, m\}$.

Zur Durchführung von Schritt (2) ist eine Laufzeit von $O(m^2n^2)$ erforderlich. Die Laufzeit von Heuristik 3 hängt ferner von der Anzahl der in Schritt 4 durchgeführten 1-er Austausch-Schritte ab. Falls die in Schritt 4 benötigte Laufzeit durch α beschränkt ist, erhalten wir eine Gesamtlaufzeit von $O(m^2n^2 + \alpha)$.

6.2 Datenstrukturen

In diesem Abschnitt werden wir einige Ideen vorstellen, welche helfen, den immensen Bedarf an Speicherplatz zu reduzieren. Desweiteren wollen wir die Datenstrukturen erklären, die für eine effiziente Implementierung der Heuristiken benötigt werden. Diese beziehen sich im wesentlichen darauf, wie die relevanten Matrixeinträge gespeichert werden müssen, damit in den Heuristiken ein schneller Zugriff auf die Matrixkoeffizienten möglich ist.
Zu Demonstrationszwecken verwenden wir Beispiel soc_1 aus der Industrie. Dieses besteht aus 1022 logischen Zellen, 1023 Netzen und 63 vorplazierten Randzellen. Die Anzahl der Basiszellen des Masters beträgt 8840.

Das erste Problem besteht darin, alle Informationen so kompakt wie möglich zu speichern, ohne dabei zu viel Zeit zu verlieren, um die relevante Information den Datenstrukturen zu entnehmen. Der schnellste Zugriff auf die Daten erfolgt sicherlich dann, wenn wir jeden Koeffizienten der Matrix Q explizit speichern. In unserem Beispiel wären dazu pro Iteration ($m = 16$) 540 Megabytes erforderlich. Dabei sind wir davon ausgegangen, daß die Speicherung einer ganzen Zahl bzw. einer Gleitkommazahl 4 Bytes beansprucht. Um diesen Speicherplatzbedarf zu reduzieren, untersuchen wir zunächst die Zielfunktion etwas genauer. Die Koeffizienten von Q bestimmen sich gemäß

$$q_{ik,jl} = c_{ij} \cdot d(i,k,j,l) + \lambda \cdot o(i,k,j,l),$$

wobei

c_{ij} = Affinitätskoeffizient zwischen den Zellen i und j,
$d(i,k,j,l)$ = "shortest Manhattan"-Distanz zwischen den Zellen i und j, wenn sie Basiszellen k und l zugewiesen wurden,
$o(i,k,j,l)$ = Anzahl von überlappenden Basiszellen, wenn Zellen i und j Basiszellen k und l zugewiesen wurden.

Betrachten wir die Koeffizienten c_{ij} $i, j = 1, \ldots, n$, $i \neq j$. Jede Zelle hat relativ wenige Pins. Daher kann eine Zelle nur an wenige Netze angeschlossen sein. Zusätzlich ist es sehr

wahrscheinlich, daß Netze großer Kardinalität über den gesamten Chip hinweg verdrahtet werden müssen. Unserer Meinung nach geben solch große Netze wenig Information, wo Zellen plaziert werden sollten. Aus diesem Grund eliminieren wir große Netze. Welche Netze als groß bezeichnet werden, hängt von den Eigenschaften des gegeben Chips ab. Wir analysieren deshalb zunächst die Netze und entscheiden dann heuristisch, welche Netze groß sind. In unserem Beispiel werden Netze der Kardinalität größer 33 eliminiert. Somit ist eine Zelle im Durchschnitt nur noch mit 10 der Zellen verbunden. Dies wiederum liefert uns eine beträchtliche Einsparung bei der Speicherung der Affinitätskoeffizienten.

Als nächstes wenden wir uns den Koeffizienten $d(i,k,j,l)$ zu. In unserem Modell legen wir die "shortest Manhattan"-Distanz als Maß für die Distanz zugrunde. Ferner gehen wir von einem Koordinatensystem aus, dessen Ursprung die linke untere Ecke des Masters bildet. In dem "sea of cells"-Entwurfsstil ist jeder Zelle i ein vordefinierter Typ $t(i)$ zugeordnet. Zwei Typen unterscheiden sich, wenn sie unterschiedliche Breite, Höhe oder Anzahl Pins haben (in unserem Beispiel existieren nur 34 verschiedene Typen von logischen Zellen). Falls wir die Distanz berechnen wollen, wenn Zellen i und j Basiszellen k und l zugewiesen wurden, ist es ausreichend, die Distanz zwischen Typen $t(i)$ und $t(j)$ zu berechen, wenn eine Zelle vom Typ $t(i)$ Basiszelle k zugewiesen wurde, und eine Zelle vom Typ $t(j)$ Basiszelle l zugewiesen wurde. Desweiteren nützen wir die Tatsache aus, daß die "shortest Manhattan"-Distanz gleich ist der Summe der Distanzen in horizontaler und vertikaler Richtung. Ferner ist die Distanz invariant gegenüber Verschiebung, d.h. Zellen können um den gleichen Betrag in horizontaler un vertikaler Richtung bewegt werden ohne daß sich die Distanz ändert. Letztere Tatsache impliziert, daß wir ohne Beschränkung der Allgemeinheit davon ausgehen können, daß Zelle i im Ursprung plaziert ist.
Zusammenfassend können wir die die Koeffizienten $d(i,k,j,l)$ somit folgendermaßen berechnen:

$$d_{ik,jl} = \mathrm{dist}\Big(t(i), \binom{0}{0}, t(j), \binom{l_x - k_x}{0}\Big) + \mathrm{dist}\Big(t(i), \binom{0}{0}, t(j), \binom{0}{l_y - k_y}\Big),$$

wobei p_x und p_y die horizontalen und vertikalen Koordinaten von Basiszelle p bezeichnen. $\mathrm{dist}\Big(t(i), \binom{a}{b}, t(j), \binom{u}{v}\Big)$ bezeichnet die "shortest Manhattan"-Distanz, wenn eine Zelle des Typs $t(i)$ bzw. $t(j)$ mit ihrer linken unteren Ecke dem Punkt mit Koordinaten $\binom{a}{b}$ bzw. $\binom{u}{v}$ zugewiesen wird.

Eine analoge Argumentation trifft für die Berechnung der Koeffizienten $o_{ik,jl}$ zu. Der einzige Unterschied besteht darin, daß die Anzahl von überlappenden Basiszellen in x- und y-Richtung miteinander multipliziert werden müssen. Somit lassen sich also die Koeffizienten $o(i,k,j,l)$ folgendermaßen berechnen:

$$o_{ik,jl} = \mathrm{ov}_x\Big(t(i), \binom{0}{0}, t(j), \binom{l_x - k_x}{0}\Big) \cdot \mathrm{ov}_y\Big(t(i), \binom{0}{0}, t(j), \binom{0}{l_y - k_y}\Big),$$

wobei ov_x und ov_y die Anzahl von Überlappungen in horizontaler und vertikaler Richtung

bezeichnen (vergleiche Abbildung 6.2.1).

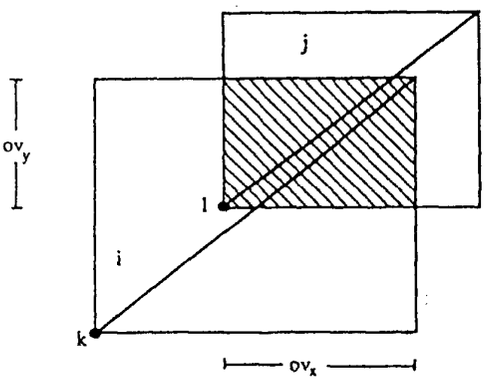


Abbildung 6.2.1

Auf diese Weise erhalten wir eine enorme Reduktion in der Anzahl der zu speichernden Daten. In unserem Beispiel benötigen wir anstelle der 540 megabytes lediglich 2,8 Megabytes an Speicherplatz. Ein weiterer Vorteil ist, daß all diese Werte nur einmal (zu Beginn der ersten Iteration) berechnet werden müssen.

Die Entscheidung, welche Datenstrukturen am besten geeignet sind, muß sicherlich sehr eng an die Wahl des Algorithmus gebunden werden. Wir haben in dem obigen Teil dargestellt, wie die Koeffizienten der Matrix möglichst effizient berechnet werden können. Betrachten wir nun Heuristik 2. Hier stellt sich die Aufgabe, sukzessive einen Knoten maximalen Gewichts zu löschen ((3.2)). Anschließend muß ein "Update" durchgeführt werden. Dabei werden insbesondere die Knotengewichte all derjenigen Knoten aktualisiert, die in G_C positives Gewicht mit dem zu löschenden Knoten haben. Um diese Anweisung effizient realisieren zu können, führen wir die im folgenden erläuterten Datenstrukturen ein:

Zur Speicherung der Affinitätskoeffizienten verwenden wir eine sogenannte Adjazenzliste, d.h. für jedes Element $i = 1, \ldots, n$ werden die Zellen gespeichert, welche mit i positiven Affinitätskoeffizienten haben.

Damit ein schneller Zugriff auf die nicht-Null Koeffizienten der Überlappungen möglich ist, werden ferner folgende zwei Datenstrukturen eingeführt:

Bezeichne hierzu b_x, beziehungsweise b_y die horizontalen beziehungsweise vetikalen Koordinaten der Basiszelle b. Wir definieren einen Array, dessen Länge der Anzahl Basiszellen des Masters entspricht. Für jede Basiszelle k tragen wir all die Knoten jl ein mit den folgenden Eigenschaften:

(i) $l_x \leq k_x$ und $l_y \leq k_y$.

(ii) Wenn Zelle j der Basiszelle l zugeordnet wird, dann überdeckt sie Basiszelle k (siehe Abbildung 6.2.2).

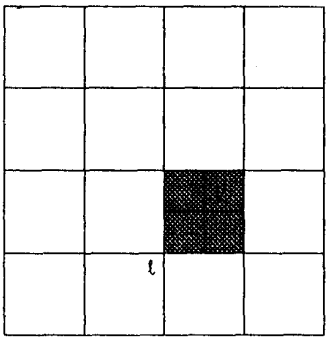


Abbildung 6.2.2

Angenommen Zelle i überdeckt genau einen Gitterpunkt in dem Gitter der momentanen Iteration (dessen repräsentative Basiszelle sei k). Sei ik der zugehörige Knoten aus $S(i)$. In diesem Fall lassen sich alle Knoten mit positivem Überlappungskoeffizient zu Knoten ik aus den Einträgen der ersten Datenstruktur ablesen (an der k-ten Stelle in dieser Datenstruktur). Wenn andererseits eine Zelle in der gegenwärtigen Iteration mehr als nur einen Gitterpunkt überdeckt, reicht die erste Datenstruktur nicht aus, um effizient auf die nicht-Null Überlappungskoeffizienten zugreifen zu können (Abbildung 6.2.3).

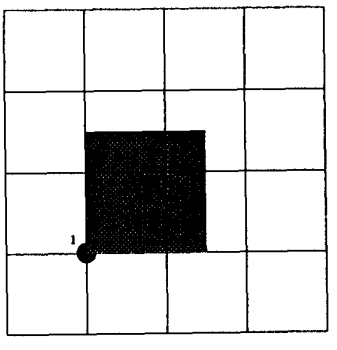


Abbildung 6.2.3

Angenommen Zelle i auf Platz k überdeckt die Basiszellen k_1, k_2, k_3 und k_4 und angenommen Zelle j hat eine Breite von 3 Gitterbreiten und eine Höhe von 3 Gitterhöhen (Vergleiche hierzu Abbildung 6.2.4).

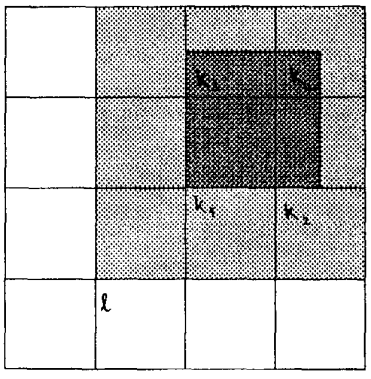


Abbildung 6.2.4

Angenommen, Basiszelle l (l ist auch ein Gitterpunkt in der gegenwärtigen Iteration) besitzt die Koordinaten $k_x - 2$ und $k_y - 2$, dann ist der Knoten jl in die Liste der Knoten der momentanen Gitterpunkte k_1, k_2, k_3 und k_4 (der ersten Datenstruktur) eingetragen worden. Will man nun die Überlappungen des Knoten ik mit allen anderen Knoten bestimmen, so kann man nicht einfach die erste Datenstruktur an den Stellen k_1, k_2, k_3, k_4 ablaufen und die Überlappungskoeffizienten zwischen ik und den dort eingetragenen Knoten aufaddieren, da bei einem solchen Vorgehen die Überlappungen zwischen ik und jl mehrfach gezählt würden (insgesamt $o_{ik_1,jl} + o_{ik_2,jl} + o_{ik_3,jl} + o_{ik_4,jl}$). Um dies zu umgehen, führen wir eine zweite Datenstruktur ein. Falls eine Zelle bei Zuordnung zu der Basiszelle k (Gitterpunkt k) in der momentanen Iteration mehr als einen Gitterpunkt überdeckt (denn nur dann wird es schwierig), werden alle Knoten jl, welche zu einer anderen maximalen stabilen Menge als $S(i)$ gehören, gespeichert, sofern $o_{ik,jl} > 0$ ist und sofern sie mehr als einen Gitterpunkt überdecken (in dem Gitter der momentanen Iteration).

Die gesamten Überlappungen zwischen dem Knoten ik und allen anderen Knoten ergeben sich dann folgendermaßen:
Wir durchlaufen die Gitterpunkte k_1, k_2, k_3, k_4 und addieren die Überlappungskoeffizienten zwischen Knoten ik und den in die erste Datenstruktur eingetragenen " One-overlapper "- Knoten auf. Anschließend summieren wir über die Überlappungskoeffizienten zwischen Knoten ik und den an der Stelle ik in der zweiten Datenstruktur eingetragenen Knoten. Die Summe aus beiden Werten liefert die exakten Überlappungen zwischen Knoten ik und allen anderen Knoten.

Während der ersten Iteration ist die zweite Datenstruktur in allen praktischen Beispielen leer, da nur 16 Gitterpunkte vorhanden sind und keine Zelle mehr als einen Gitterpunkt überdeckt. Während des Verfahrens könnte diese zweite Datenstruktur theoretisch genauso viele Einträge erhalten wie die Matrix Q Elemente hat. Da aber die Anzahl von Zellen in der lokalen Nachbarschaft einer Zelle i bei zunehmender Iterationstiefe abnimmt, wird dies nie passieren. In unserem Beispiel benötigen beide Datenstrukturen zusammen nicht mehr als 1 Megabyte an Speicherplatz.

6.3 Implementierung der Heuristiken

In diesem Abschnitt wollen wir an Beispielen demonstrieren, wie verschiedene Algorithmen effizient implementiert werden können. Insbesondere werden wir auf Heuristik 2, sowie das Verbesserungsverfahren näher eingehen.

Zunächst benötigen wir noch eine Definition.

Definition

Ein Heap in aufsteigender Reihenfolge ist eine Tripel $(\mathcal{S}, \pi, w)$ mit den Eigenschaften:

$\mathcal{S}$ ist eine n-elementige Menge,

$w : \mathcal{S} \to \mathbb{R}$ ist eine Funktion, die jedem Element aus $\mathcal{S}$ eine reele Zahl (das Gewicht) zuordnet,

$\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ ist eine bijektive Abbildung, so daß

$$w(S_{\pi(i)}) \leq w(S_{\pi(2i)}) \text{ und } w(S_{\pi(i)}) \leq w(S_{\pi(2i+1)})$$

für alle $i \in \{1, \ldots, \lfloor \frac{n}{2} \rfloor\}$, $2i + 1 \leq n$.

Entsprechend definieren wir einen Heap in absteigender Reihenfolge.

Bezeichne $\mathcal{S}$ die Menge der maximal stabilen Mengen S_i, $i = 1, \ldots, n$ in G_C. Unter Verwendung eines Heaps können wir Algorithmus 2 modifizieren. Diese Variante – Heuristik 2a genannt – läßt sich folgendermaßen darstellen.

Heuristik 2a

Input: $G_C = (V_C, E_C, w_C)$, mit $|V_C| = mn$, $m \geq 2$.

Output: Eine Clique Q der Kardinalität n.

(1) Weise jedem Knoten $v \in V_C$ ein Knotengewicht zu, gemäß der Formel:
$g(v) = \frac{1}{|\delta(v)|} \sum_{(u,v) \in E_C} w_C(u, v)$.

(2) Berechne $\gamma(S) = \max_{v \in S} g(v) / \min_{v \in S} g(v)$ für alle maximalen stabilen Mengen S in G_C . Initialisiere einen Array K der Länge n, der für jede der Mengen S_i, $i = 1, \ldots, n$ einen Knoten v_i speichert mit $g(v_i) = \max_{v \in S_i} g(v)$. Setze $Q = \emptyset$.

(3) Konstruiere einen Heap $(\mathcal{S}, \pi, \gamma)$ in absteigender Reihenfolge.

(4) Solange $|V_C| > n$ ist, führe die folgenden Anweisungen durch:

(4.1) Sei $S_0 = S_{\pi(1)}$. Setze $v_0 = K[\pi(1)]$.

(4.2) Setze $V_C = V_C \setminus \{v_0\}$ und $E_C = E_C \setminus \delta(v_0)$. Falls $|S_0| = 1$ ist, so setze $\gamma(S_0) = -\infty$.

(4.3) Aktualisiere die Werte $g(v)$ für alle $v \in V_C$. Aktualisiere die Werte $\gamma(S)$ und den Array K.

(4.4) Konstruiere den Heap $(\mathcal{S}, \pi, \gamma)$.

(5) Setze $Q = V_C$.

In Schritt (1) müssen für alle $v \in V_C$ die Summen der zu v inzidenten Kantengewichte aufaddiert werden. Die Durchführung dieses Schrittes läßt sich mit den in Abschnitt (6.2) vorgestellten Datenstrukturen sehr effizient realisieren. Wir bestimmen die Zellen (maximalen stabilen Mengen in G_C), welche mit $S(v)$ positiven Affinitätskoeffizienten haben und addieren die entsprechenden Gewichte $c_{ij} \cdot d(ik, jl)$ auf. Anschließend durchlaufen wir die Datenstrukturen für die Überlappungen und addieren ebenfalls die entsprechenden Koeffizienten $o(ik, jl)$ auf. Dieses Vorgehen stellt sicher, daß in Schritt (1) alle positiven Matrixkoeffizienten bei der Berechnung der Knotengewichte berücksichtigt werden. Bezeichne ψ die Anzahl positiver Kantengewichte in G_C.
Schritt (2) erfordert eine Laufzeit von nm. Für die Durchführung von Schritt (3) ist im schlechtesten Fall $O(nlogn)$ Laufzeit zu veranschlagen. Die Anweisungen in (4.1) und (4.2) können in konstanter Zeit durchgeführt werden. Für die effiziente Implementierung von Schritt (4.3) sind wieder die in Kapitelabschnitt (6.2) definierten Datenstrukturen sehr geeignet, da sich nur die Knotengewichte solcher Knoten ändern, die mit v_0 positiven Matrixkoeffizienten haben. Da wir jeden Knoten höchstens einmal löschen (insgesamt werden $n(m-1)$ Knoten in G_C gelöscht), werden insgesamt nicht mehr als ψ viele Knotengewichte verändert. Für die maximalen stabilen Mengen, deren Knoten ihr Gewicht geändert haben, müssen Gewichte neu berechnet und es muß der Heap wieder aufgebaut werden. Dadurch hat sich die Geamtlaufzeit im schlechtsten Fall auf $(O(m^2n^2log(n))$ erhöht (im Vergleich zu $O(m^2n^2)$ für Heuristik 2). Dieser Fall kommt bei der Lösung unserer Instanzen von (P_{16}^+) jedoch nicht vor. Die Laufzeit für Heuristik 2a ergibt sich im wesentlichen daraus, wie lange der "Update" in (4.3) dauert, bzw. wie viele maximale stabilen Mengen in G_C ihr Gewicht verändern. Denn diese müssen in den Heap neu eingeordnet werden.
Zunächst ist die Anzahl von Knotenpaaren, für die der Ausdruck $c_{ij}d(i,k,j,l)$ von Null verschieden ist, über alle Iterationen hinweg durch den Ausdruck $m^2\alpha$ beschränkt. α bezeichnet dabei die Anzahl von positiven Affinitätskoeffizienten. Wir haben bereits in dem vorigen Kapitelabschnitt gesehen, daß α für unsere Instanzen durch eine Konstante multipliziert mit der Anzahl Zellen beschränkt ist (≤ 50). Im Mittel müßten daher nicht mehr als durch eine Konstante beschränkte Anzahl maximaler stabilen Mengen in Schritt (4.4) pro Löschen eines Knotens in den Heap neu eingeordnet werden. Die tatsächliche Laufzeit für die Durchführung eines Schritts (4.4) hängt somit von der Anzahl der positven Überlappungskoeffizienten ab. In der ersten Iteration stehen für jede Zelle 16 repräsentative Basiszellen zur Verfügung, wobei jede Zelle jeder dieser Basiszellen zugewiesen werden kann. Das bedeutet, daß jeder Knoten $v \in V_C$ mit genau Anzahl Zellen vielen anderen Knoten einen positiven Überlappungskoeffizient hat. In den späteren Iterationen verändert sich die Situation. Die Anzahl von Knotenpaaren mit positivem Überlappungskoeffizient verringert sich deutlich, da mehr repräsentative Basiszellen zur Verfügung stehen, auf welche sich die Zellen verteilen. Somit ist die Anzahl von Zellen in der lokalen Nachbarschaft einer Zelle i relativ klein. Daraus resultiert eine geringe Anzahl von Knotenpaaren mit

positivem Überlappungskoeffizient. In Kapitel 7 werden wir deutlich feststellen können, wie sich die Laufzeit für Heuristik 2a in den verschiedenen Iterationen verhält.

Wie wir gesehen haben, wird für die Berechnung der exakten Anzahl von Überlappungen in der ersten Iteration sehr viel Rechenzeit benötigt. Dies liegt darin begründet, daß aufgrund der geringen Anzahl von repräsentativen Basiszellen die Anzahl von positiven Überlappungskoeffizienten $O(n^2)$ ist. Deshalb entschieden wir uns dafür, eine vereinfachte Berechnung der Überlappungen zu untersuchen. Für jede Basiszelle b bestimmen wir die Anzahl von Knoten jl mit der Eigenschaft, daß Zelle j, wenn sie Basiszelle l zugewiesen wurde, diese Basiszelle überdeckt. Diese Zahl bezeichnen wir mit $u(b)$. Wenn wir für einen Knoten ik die Gesamtzahl der Überlappungen mit allen anderen Knoten berechnen müssen, laufen wir alle Basiszellen ab, welche von Zelle i überdeckt werden, wenn sie der Basiszelle k zugewiesen ist und addieren die entsprechen Werte auf. Diese Summe liefert eine Abschätzung für die Überlappung von ik mit den anderen Knoten. Zudem liefern uns die Werte $u(.)$ ein Maß, wie gut oder schlecht die Zellen auf dem Master verteilt sind. Ausgehend von dieser Idee werden wir im folgenden eine zweite Version von Heuristik 2a vorstellen und zwei Variantern von Verbesserungsverfahren präsentieren.

Bezeichne im folgenden $U(jl)$ die Menge von Basiszellen des Masters, welche überdeckt werden, wenn Zelle j Basiszelle l zugewiesen wird. Wie bereits oben eingeführt, bezeichnet $u(b)$ die Anzahl von Knoten jl mit der Eigenschaft, daß Zelle j, wenn sie Basiszelle l zugewiesen wurde, Basiszelle b überdeckt. Ferner sei $\mathcal{S} = \{S_1, \ldots, S_n\}$ die Menge der maximalen stabilen Mengen in G_C. Mit Hilfe dieser Notation können wir eine zweite Variante von Algortihmus 2 darstellen (im folgenden Heuristik 2b genannt).

Heuristik 2b

Input: $G_C = (V_C, E_C, w_C)$, mit $|V_C| = mn$, $m \geq 2$.

Output: Eine Clique Q der Kardinalität n.

(1) Weise jedem Knoten $v = ik \in V_C$ ein Knotengewicht zu, gemäß der Formel: $g(ik) = \sum_{j|c_{ij}>0} \sum_{l=1}^{m} c_{ij} d(ik, jl) + \sum_{b \in U(ik)} u(b)$.

(2) Berechne $\gamma(S) = \max_{v \in S} g(v) / \min_{v \in S} g(v)$ für alle maximalen stabilen Mengen S in G_C . Initialisiere einen Array K der Länge n, der für jede der Mengen S_i, $i = 1, \ldots, n$, einen Knoten v_i speichert mit $g(v_i) = \max_{v \in S_i} g(v)$. Setze $Q = \emptyset$.

(3) Konstruiere einen Heap $(\mathcal{S}, \pi, \gamma)$ in absteigender Reihenfolge.

(4) Solange $|V_C| > n$ ist, führe die folgenden Anweisungen durch:

(4.1) Sei $S_0 = S_{\pi(1)}$. Setze $ik = K[\pi(1)]$.

(4.2) Setze $V_C = V_C \backslash \{ik\}$ und $E_C = E_C \backslash \delta(ik)$. Falls $|S_0| = 1$ ist, so setze $\gamma(S_0) = -\infty$.

(4.3) Update: Für alle $j = 1, \ldots, n$ mit $c_{ij} > 0$, $l = 1, \ldots, m$ setze $g(jl) = g(jl) - c_{ij} d(ik, jl) - \lambda |U(ik) \cap U(jl)|$. Setze $u(b) = u(b) - 1$ für alle $b \in U(ik)$. Aktualisiere

die Werte $\gamma(S_j)$ und die entsprechenden Einträge in dem Array K für alle j mit $c_{ij} > 0$.

(4.4) Konstruiere den Heap $(\mathcal{S}, \pi, \gamma)$.

(5) Setze $Q = V_C$.

Im Gegensatz zu Heuristik 2 und 2a wird in Schritt 4 nicht notwendigerweise eine stabile Menge maximalen Gewichts (bezüglich der Funktion γ) gefunden. Sei ik der zu löschende Knoten in der gegenwärtigen Iteration. Dann werden in dem "Update" nur solche Knoten berücksichtigt, die zu einer maximalen stabilen Menge S_j mit positivem Affinitätskoeffizienten c_{ij} gehören. Es werden zwar die Einträge $u(b)$ für alle von i auf ik überdeckten Basiszellen berichtigt, jedoch müßten alle anderen Knoten jl mit $U(jl) \cap U(ik) \neq \emptyset$ aktualisiert werden, um eine stabile Menge maximalen Gewichts in Schritt (4.1) zu erhalten. Da beispielsweise in der ersten Iteration jede Zelle jeder repräsentativen Basiszelle zugewiesen werden kann, wäre auch unter Verwendung der vereinfachten Zielfunktion eine Laufzeit von $O(nlog(n))$ für einen Durchlauf der Schritte (4.1) bis (4.4) erforderlich. Setzen wir wieder voraus, daß jede Zelle zu nicht mehr als konstant vielen Zellen einen positiven Affinitätskoeffizienten hat, so können die Schritte (4.1) bis (4.4) in konstanter Zeit durchlaufen werden. Dabei sind wir davon ausgegangen, daß auch Breite und Höhe der Zellen über alle Instanzen unserer Anwendungen konstant ist. Diese Annahme ist realistisch.
Die modifizierte Berechnung der Überlappungen bewirkt insbesondere, daß die Laufzeit für die Durchführung der Schritte (1) bis (3) durch $O(nlogn)$ beschränkt ist.
Somit ergibt sich für Heuristik 2b eine Gesamtlaufzeit von $O(nlogn)$ im schlechtesten Fall. In Kapitel 7 wird sich insbesondere in den ersten Iterationen der Laufzeitunterschied zwischen den Algorithmen 2a und 2b an den praktischen Problembeispielen zeigen.

Wie oben bereits angedeutet, haben wir unter Anwendung der vereinfachten Zielfunktionsberechnung zwei Implementationen des 1er-Austausch-Verbesserungsverfahrens entwickelt. Bezeichne hierzu B^r die Menge der zulässigen repräsentativen Basiszellen in Iteration r. Für eine gegebene Zelle $j = 1, \ldots, n$ kürzen wir die Basiszelle, welcher j in der gegenwärtigen Lösung zugewiesen ist, mit b_j ab. Sei jl_0 der zugehörige Knoten. Unter den **Affinitätskosten** $a[ik]$ eines Knotens $ik \in V_C$ verstehen wir den Ausdruck

$$a[ik] = \sum_{j|c_{ij}>0} c_{ij} d(i,k,j,b_j).$$

Die erste Variante des Verbesserungsverfahrens (im folgenden Verbesserungsverfahren 1a genannt) beruht auf der Idee, insbesondere in den ersten Iterationen die Verteilung der Zellen in der gegenwärtigen Lösung zu verbessern. Dabei sollen die Affinitätskosten berücksichtigt werden.

Zunächst unterteilen wir die Elemente von B^r in "unterbesetze", "normal besetze" und "überbesetzte" ein. Für einen Knoten jl werden ferner die Affinitätskosten berechnet,

sofern die Basiszelle l nicht "überbesetzt" ist. Anderenfalls setzen wir $a[jl] = +\infty$. Anschließend bauen wir einen Heap auf, der als Wert für eine Zelle den Knoten mit minimalen Affinitätskosten verwendet. Pro Iteration wählen wir uns den ersten Knoten ik im Heap. Falls sich der Wert der Zielfunktion durch den Austausch verringert, so akzeptieren wir ihn. Anderenfalls weisen wir dem Array an der Stelle i einen großen Wert zu und ordnen ihn wieder in den Heap ein. Das Verfahren endet, wenn wir eine vorgegebene Anzahl von Iterationen durchgeführt haben.

Zur Darstellung des Verfahrens greifen wir auf die in Kapitel 3 eingeführte Notation zurück. Sei $N = \{1, \ldots, n\}$ und bezeichne f_i, $i = 1, \ldots, n$ die Fläche von Zelle i. Die Gesamtfläche der Zellen (f) ergibt sich dann als $f = \sum_{i=1}^{n} f_i$. Sei Q eine Clique der Kardinalität n in G_C. Dann bezeichnen wir mit jl_0 den Knoten in Q, der zu der maximal stabilen Menge S_j gehört. Die zugehörige repräsentative Basiszelle l_0 kürzen wir auch mit b_j ab. Für ein $jl \in V_C$ bezeichnet $U(jl)$ die Menge von Basiszellen des Masters, welche überdeckt werden, wenn Zelle j Basiszelle l zugewiesen wird. Bezeichne ferner $u(b)$ die Anzahl von Knoten jl_0 in der Clique , mit der Eigenschaft, daß Zelle j, wenn sie Basiszelle l_0 zugewiesen wurde, Basiszelle b überdeckt.

Verbesserungsverfahren 1a

Input: $G_C = (V_C, E_C, w_C)$, mit $|V_C| = mn$, $m \geq 2$ und eine Clique Q der Kardinalität n. Gegeben ist ferner die Iteration r und B^r, die Menge der zulässigen repräsentativen Basiszellen in Iteration r.

Output: Eine Clique Q der Kardinalität n.

(1) Berechne den Erwartungswert E_f der Fläche pro Basiszelle und eine Varianz. $E_f = \frac{f}{|B^r|}$ und $\sigma_f = \frac{E_f}{10}$. Setze "enditeration = n" und "iteration = 1". Setze $nr[ik] = 0$ für alle $ik \in V_C$. Der array nr gibt an, wie oft bereits versucht wurde, einen Knoten in die Lösung aufzunehmen. Wenn diese Anzahl eine vorgegebene Konstante mnr überschreitet, wird dieser Knoten nicht mehr zugelassen.

(2) Initialisiere einen Array D (Dichte der Basiszellen), der Länge $|B^r|$ gemäß :

$$D[b] = \begin{cases} \text{"überbesetzt"}, & \text{wenn } \sum_{i|b_i=b} f_i > E_f + \sigma_f, \\ \text{"unterbesetzt"}, & \text{wenn } \sum_{i|b_i=b} f_i < E_f - \sigma_f, \\ \text{"normal besetzt"}, & \text{sonst.} \end{cases}$$

Dies gilt für alle $b \in B^r$.

(3) Initialisiere die Arrays a und a'. Setze hierzu für alle Knoten ik in V_C:

$$a[ik] = \begin{cases} \sum_{j|c_{ij}>0} c_{ij} d(i,k,j,b_j), & \text{wenn } D[k] \neq \text{"überbesetzt"}, \\ +\infty, & \text{anderenfalls.} \end{cases}$$

Setze $a'[i] = \min\{a[ik] \mid k = 1, \ldots, m\}$ für alle $i = 1, \ldots, n$.

(4) Konstruiere einen Heap (N, π, a') in aufsteigender Reihenfolge.

(5) Solange iteration $\leq$ enditeration ist, führe die folgenden Anweisungen durch:

(5.1) Sei $i = \pi(1)$. Berechne die Werte $a[ik]$ für alle $k = 1, \ldots, m$ und setze $\hat{a} = \min\{a[ik] + \lambda \sum_{b \in U(ik)} u(b) \mid k = 1, \ldots, m\}$. Bestimme einen Knoten ik mit $\hat{a} = a[ik] + \lambda \sum_{b \in U(ik)} u(b)$. Setze $nr[ik] = nr[ik] + 1$.

(5.2) Falls $\hat{a} = +\infty$, gehe zu Schritt (6).
Falls $nr[ik] = mnr$ ist, setze $a'[i] = +\infty$ und gehe zu Schritt (5.4).
Falls $D[k_0]$ ="überbesetzt", setze $a'[i] = +\infty$ und gehe zu Schritt (5.4).
Setze $Q' = (Q \setminus \{ik_0\}) \cup \{ik\}$. Falls $w_C(Q') \geq w_C(Q)$, setze $a'[i] = +\infty$ und gehe zu Schritt (5.4).

(5.3) Aktualisiere $D[b_i]$ und $D[k]$. Setze $b_i = k$, $ik_0 = ik$ und $Q = Q'$. Aktualisiere die Werte $a[ik']$ für alle $k' = 1, \ldots, m$ und den Wert $a'[i]$.
Für alle $j = 1, \ldots, n$ mit $c_{ij} > 0$, $l = 1, \ldots, m$ aktualisiere die Werte $a[jl]$ und bestimme den Wert $a'[j]$. Dabei gilt, daß $a[jl]$ den Wert $+\infty$ zugewiesen bekommt, wenn $D[l]$ = "überbesetzt" oder wenn $nr[jl] \geq mnr$ gilt.

(5.4) Konstruiere den Heap (N, π, a'). Setze "iteration = iteration + 1".

(6) Gib Q aus.

Zu Beginn des Verfahrens werden in dem Heap alle Knoten berücksichtigt, die eine nicht "überbesetzte" Basiszelle repräsentieren. Da sich nach einigen Iterationen die Lösung geändert hat, sind die Einträge in a' nicht notwendigerweise diesem aktuellen Stand angepaßt. Vielmehr können inzwischen Basiszellen, die zu Beginn "überbesetzt" waren, "normal"- oder "unterbesetzt" sein. Für Zellen, die mit keiner der bis zu diesem Punkt ausgetauschten Zellen positiven Affinitätskoeffizienten haben, sind die Werte $a'[.]$ nicht aktualisiert. Wollte man diesen Nachteil beheben, so würde das erfordern, daß pro Austausch $O(n)$ viele $a[ik]$-Werte neu berechnet werden müßten. Dies widerspricht unserem Ziel, ein schnelles Verbesserungsverfahren zu entwickeln. Um den Nachteil jedoch einzudämmen, wählen wir für eine Zelle unter allen Knoten, die keine "überbesetzte" Basiszelle repräsentieren, den mit minimalen Kosten aus.
Der Array nr wurde zur Vermeidung des Falls eingeführt, daß eine oder mehrere Zellen in einer bestimmten Reihenfolge ihre Position(en) tauschen ("looping"). Solch eine Situation kann entstehen, wie folgendes Beispiel zeigt.
Seien $i \neq j$ zwei Zellen mit positivem Affinitätskoeffizienten, die in der momentanen Lösung Basiszellen b_i beziehungsweise b_j zugewiesen sind. Nehmen wir an, Zelle i wird in Schritt (5.1) ausgewählt und einer Basiszelle b_i' zugewiesen, so daß minimale Kosten entstehen. Nehmen wir weiter an, daß durch diesen 1-er Austausch der Zielfunktionswert der neuen Lösung gleich dem der alten bleibt. Anschließend erhält der Array a' an der Stelle i den Wert $+\infty$ und wird in den Heap eingeordnet. Beim nächsten Durchlauf wird in Schritt 5.1 Zelle j ausgewählt und der Basiszelle b_j' zugewiesen. Wir gehen wiederum davon aus, daß

sich dadurch der Wert der Zielfunktion nicht ändert. Dann aber erhält der Eintrag $a'[i]$ im "Update"-Schritt wieder einen Wert, der von ∞ verschieden ist. Somit kann Zelle i an die erste Stelle des Heaps gelangen. Anschließend wird i wieder der Basiszelle b_i zugewiesen, ohne den Zielfunktonswert zu beeinflussen. Nun wird in dem "Update"-Schritt j ein Wert zugewiesen. Falls daraufhin j in Schritt 5.1 ausgewählt und der Basiszelle b_j zugewiesen wird, so würde der Algorithmus "loopen". Diese Situation läßt sich mit Hilfe des Arrays nr vermeiden.

Zur Analyse des Verfahrens gehen wir davon aus, daß die Anzahl Basiszellen des Masters durch eine Konstante multipliziert mit der Anzahl Zellen beschränkt ist.
Da ferner jede Zelle nur mit konstant vielen anderen positiven Affinitätskoeffizienten hat, ergibt sich eine Laufzeit von $O(nlog(n))$ für die Schritte (1) bis (4). Die Schritte (5.1) bis (5.4) werden insgesamt n mal abgelaufen. Jeder Durchlauf läßt sich unter obigen Annahmen in konstanter Zeit realisieren. Somit ergibt sich ein Gesamtlaufzeit von $O(nlog(n))$.

Die zweite Variante des Verbesserungsverfahrens (Verbesserungsverfahren 1b) soll insbesondere in den Enditerationen lokale Überlappungen unter Berücksichtigung der Affinitätskosten beseitigen. Hierzu bestimmen wir für jede repräsentative Basiszelle b ihre Auslastung $A(b)$ (bezüglich der gegenwärtigen Lösung) und ihre Kapazität $cap(b)$. Für jede Zelle bestimmen wir anschließend den Knoten mit minimalen Affinitätskosten und setzen diesen Wert in Relation zu den Affinitätskosten des Knotens, der in der momentanen Lösung aufzufinden ist. Zellen, die einer repräsentativen Basiszelle zugewiesen wurden und deren Auslastung die Kapazität überschreitet, erhalten eine Bonus. Solche Zellen sollen dadurch eine möglichst gute Position in dem Heap bekommen. Der Heap ist bezüglich des oben beschriebenen Arrays aufgebaut. Pro Iteration wählen wir das erste Element des Heaps und führen eine Austausch durch, falls dies zu einer Verringerung des Zielfunktionswerts führt. Sei Q eine Clique der Kardinalität n in G_C. Dann bezeichnen wir mit jl_0 den Knoten in Q, der zu der maximal stabilen Menge S_j gehört. Die zugehörige repräsentative Basiszelle l_0 kürzen wir auch mit b_j ab. Für ein $jl \in V_C$ bezeichnet $U(jl)$ die Menge von Basiszellen des Masters, welche überdeckt werden, wenn Zelle j Basiszelle l zugewiesen wird. Bezeichne ferner $u(b)$ die Anzahl von Knoten jl_0 in der Clique , mit der Eigenschaft, daß Zelle j, wenn sie Basiszelle l_0 zugewiesen wurde, Basiszelle b überdeckt. Für eine repräsentative Basiszelle b definieren wir ihre Auslastung $A(b) = \sum_{b' \in B'} u(b')$, wobei B' die Menge der Basiszellen ist, die durch b repräsentiert werden (vergleiche hierzu auch Kapitel 5). Sei schließlich $N = \{1, \ldots, n\}$.

Verbesserungsverfahren 1b

Input: $G_C = (V_C, E_C, w_C)$, mit $|V_C| = mn$, $m \geq 2$ und eine Clique Q der Kardinalität n. Gegeben ist ferner die Iteration r und B^r, die Menge der zulässigen repräsentativen Basiszellen in Iteration r.

Output: Eine Clique Q der Kardinalität n.

(1) Für jede repräsentative Basiszelle $b \in B^r$ bestimme ihre Kapazität $cap(b)$. Diese berechnet sich als Summe der Basiszellen des Masters, welche durch b repräsentiert werden. Initialisiere die Zahlen "iteration, enditeration" und mnr. Initialisiere ferner die Arrays $A(.), u(.)$ und $nr(.)$.

(2) Initialisiere den Array a. Setze hierzu für alle Knoten ik in V_C:

$$a[ik] = \sum_{j|c_{ij}>0} c_{ij} d(i,k,j,b_j)$$

(3) Für alle $i = 1, \ldots, n$ führe folgende Anweisungen durch:

(3.1) Gilt $A(b_i) > cap(b_i)$, setze bonus = 1, anderenfalls setze bonus = 0.

(3.2) Bestimme $a'[i] = \frac{\min\{a[ik] | k=1,\ldots,m\}}{a[ik_0]}$ - bonus.

(4) Konstruiere einen Heap (N, π, a') in aufsteigender Reihenfolge.

(5) Solange iteration $\leq$ enditeration ist, führe die folgenden Anweisungen durch:

(5.1) Sei $i = \pi(1)$. Berechne die Werte $a[ik]$ für alle $k = 1, \ldots, m$. Setze $\hat{a} = \min\{a[ik] + \lambda \sum_{b \in U(ik)} u(b) \mid k = 1, \ldots, m\}$. Bestimme einen Knoten ik mit $\hat{a} = a[ik] + \lambda \sum_{b \in U(ik)} u(b)$. Setze $nr[ik] = nr[ik] + 1$.

(5.2) Falls $\hat{a} = +\infty$, gehe zu Schritt (6).
Falls $nr[ik] = mnr$ ist, setze $a'[i] = +\infty$ und gehe zu Schritt (5.4).
Setze $Q' = (Q \setminus \{ik_0\}) \cup \{ik\}$. Falls $w_C(Q') \geq w_C(Q)$, setze $a'[i] = +\infty$ und gehe zu Schritt (5.4).

(5.3) Aktualisiere $u(b_i), A(b_i)$ und $u(k), A(k)$. Setze $b_i = k$, $ik_0 = ik$ und $Q = Q'$. Aktualisiere den Wert $a'[i]$.
Für alle $l = 1, \ldots, m$ und $j = 1, \ldots, n$ mit $c_{ij} > 0$ aktualisiere die Werte $a[jl]$ und bestimme den Wert $a'[j]$ (wie in Schritt 2 erläutert). Falls $nr[jl] \geq mnr$ gilt, so setzen wir $a'[j] = +\infty$.

(5.4) Konstruiere den Heap (N, π, a'). Setze "iteration = iteration + 1".

(6) Gib Q aus.

Aus analogen Gründen gilt, wie bei dem Verbesserungsverfahren 1a beschrieben, daß wir in Schritt (4.1) unter allen Knoten den mit minimalen Kosten suchen müssen, weil die Werte im Heap nach einigen Iterationen nicht unbedingt den aktuellen Werten entsprechen.
Die Laufzeit dieser Variante ist, wie bereits die des Verbesserungsverfahrens 1a, durch $O(nlog(n))$ beschränkt.

Hiermit schließen wir das Kapitel über Heuristiken, Datenstrukturen und Implementierung. Wir werden in dem folgenden Teil eine Vielzahl von Tests durchführen, die uns erlauben, eine empirische Bewertung der in diesem Kapitel diskutierten Algorithmen vorzunehmen.

Kapitel 7 Tests zur Parametereinstellung

Ziel dieses Kapitels ist es, durch empirische Tests eine geeignete Wahl für die Einstellung der Parameter in dem Plazierungsprogramm zu finden. Wir werden dabei zunächst die Heuristiken aus Kapitel 6 vergleichen. Aus diesen Ergebnissen wollen wir zum einen schlußfolgern, welcher Algorithmus in welcher Hierarchiestufe (bezüglich der Dekompositionsverfahren aus Kapitel 5) angewendet werden soll. Zum anderen wollen wir herausfinden, wie der Parameter für die Bestrafung der Überlappungen a-priori bestimmt werden kann. Dies werden wir in Abschnitt 7.2 behandeln. Um die Verdrahtbarkeitsbedingung in unserem Plazierungsalgorithmus besser berücksichtigen zu können, führen wir einen Parameter ein. Der Frage, inwieweit dieser Parameter die Lösungen beeinflußt, wird in Abschnitt 7.3 nachgegangen. Im letzten Teil vergleichen wir die in Kapitel 5 vorgestellten Dekompositionsverfahren miteinander.

7.1 Vergleich der Heuristiken

In diesem Teil beziehen wir uns auf die in Kapitel 6 vorgestellten Heuristiken. Insbesondere vergleichen wir Heuristik 1 mit den beiden Implementierungen von Heuristik 2 (als 2a bzw. 2b in Abschnitt 6.3 bezeichnet) und mit Heuristik 3. Bei der Durchführung der Testserie verwenden wir das in Abschnitt 5.1 vorgestellte Dekompositionsverfahren. Bezüglich der Tests gehen wir so vor, daß auf jede Lösung der Primalheuristik je einmal die beiden Verbesserungsverfahren 1a und 1b angewendet werden. Als Testschaltungen wählen wir die Beispiele soc_2 und soc_3, die von ihrer Struktur sehr verschieden sind und in weiterem Sinne als repräsentativ bezeichnet werden können. soc_2 besteht aus 1022 logischen Zellen, 1025 Netzen und 63 vorplazierten Randzellen. Die Dichte der Schaltung beträgt 50%. Ferner sind die Zellen von heterogener Struktur, d.h. betrachtet man die Zellenflächen bzw. die Anzahl Pins einer Zelle als Zufallsvariable, so ist die Varianz der entsprechenden Folge von Zufallsvariablen groß (relativ zu anderen Schaltungen). soc_3 hingegen ist eine sehr homogene Schaltung. Sie besteht aus 2293 logischen Zellen, 2776 Netzen und 157 vorplazierten Randzellen. Die Dichte von soc_3 beträgt 23.6%.
In unserer Testserie beginnen wir mit Beispiel soc_2. Wir vergleichen die Heuristiken für verschiedene Werte λ des Bestrafungsparameters in der ersten Iteration. Anschließend wählen wir eine Lösung aus, die bezüglich der abgeschätzten Verdrahtungslänge und bezüglich der Überlappungen möglichst gut ist. Die Testläufe werden zeigen, daß eine solche Lösung nicht eindeutig ist, da in der Regel eine geringere abgeschätzte Verdrahtungslänge auf Kosten der Gesamtanzahl der Überlappungen geht und umgekehrt. Die Lösung verwenden wir als Startlösung für unsere Testserie in der zweiten Iteration. Daraufhin vergleichen wir die Heuristiken bezüglich verschiedener λ Werte in der zweiten Iteration und bestimmen eine möglichst gute Lösung, welche als Startlösung für die folgende Iteration verwendet wird. Dieser Ablauf wird fortgesetzt, bis die letzte Iteration erreicht ist. Ausgehend von dem Parameter λ in der Startlösung für die zweite Iteration (dies

sei λ_1) berechnen wir die λ Werte in der Testserie als Produkt aus λ_1 und den Zahlen aus $\{0.5, 1, 1.5, \ldots, 6\} \cup \{8, 10, 12, 14, 16\}$. Bezeichne λ_i den Bestrafungsparameter in der Startlösung für Iteration $i+1$, $i \geq 2$. Dann vergleichen wir die Heuristiken bezüglich der λ-Werte, die sich als Produkt aus λ_i und den Zahlen der Menge $\{\frac{1}{3}, \frac{2}{3}, \ldots, 6\}$ ergeben.
Aufgrund der Vielzahl dieser Daten haben wir für die Schaltung soc_3 die Heuristiken nur noch in den ersten drei Iteration bezüglich verschiedener λ-Werte verglichen. Wir werden sehr deutlich sehen, daß diese Testläufe ausreichend viel Information liefern, um eine Bewertung der Heuristiken vorzunehmen. Damit in diesem Abschnitt die Übersichtlichkeit gewährleistet bleibt, präsentieren wir hier nur eine kleinen Auszug aus der Testserie für Schaltung soc_2. Die vollständige Testserie ist im Anhang aufgelistet.
In Spalte 1 tragen wir die jeweilige Heuristik ein. In Spalte 2 ist die Abschätzung der Verdrahtungslänge, wie in Kapitel 3 beschrieben, angegeben. Die Gesamtanzahl der Überlappungen ist in Spalte 3 dargestellt. Den Zielfunktionswert der Lösungen zeigt Spalte 4 auf. Dieser besteht aus der Summe der abgeschätzten Verdrahtungslänge und der Anzahl an Überlappungen, gewichtet mit dem Parameter λ. Die CPU Zeit für die jeweilige Heuristik entnehmen wir der letzten Spalte. Die Laufzeit ist in Minuten angegeben. Die Testläufe wurden auf einer Sun 4/50 durchgeführt.

Auszug aus der Testserie für die erste Iteration bezüglich der Schaltung soc_2

λ =0.010

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	164142.09	3464162.00	198783.71	1:51
Heuristik 2a	108239.84	4404858.00	152288.42	16:04
Heuristik 2b	119923.45	4523759.00	165161.03	1:19
Heuristik 3	150897.91	4494674.00	195844.65	0:23

Tabelle 7.1.3

λ =0.020

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	177647.41	3377195.00	245191.30	1:52
Heuristik 2a	115831.35	3803475.00	191900.85	15:59
Heuristik 2b	137707.34	3597566.00	209658.66	1:21
Heuristik 3	167609.19	3678570.00	241180.59	0:23

Tabelle 7.1.4

λ =0.050

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	168885.84	3513932.00	344582.45	1:53
Heuristik 2a	126377.67	3577735.00	305264.42	16:04
Heuristik 2b	149238.33	3489141.00	323695.38	1:24
Heuristik 3	177558.11	3581612.00	356638.71	0:24

Tabelle 7.1.7

Zunächst liefert die Analyse der Tests, daß Heuristik 2a über alle Iterationen hinweg für gegebenen Wert λ eine Lösung liefert, die bezüglich der Zielfunktion (gewichtete Summe aus abgeschätzter Verdrahtungslänge und Überlappungen) minimal ist. Die zweitbeste Lösung bezüglich der Zielfunktion liefert in der Regel in den ersten beiden Iterationen Heuristik 2b. In deutlichem Abstand dazu folgt Heuristik 3. Die sicherlich schlechtesten Lösungen sind die der Heuristik 1. Ab der dritten Iteration ist insbesondere der Abstand zwischen den Zielfunktionswerten der Heuristiken 1, 2b und 3 relativ klein.
Die Testergebnisse zeigen ferner, daß für gegebenes λ Heuristik 2a in den meisten Beispielen eine Lösung liefert, die (im Vergleich zu den Lösungen der anderen Heuristiken) minimale abgeschätzte Verdrahtungslänge hat. An zweiter Stelle liegen wiederum in der Regel die Lösungen von Heuristik 2b. Die schlechtesten Ergebnisse erzielen Heuristiken 1 und 3.
Die Anzahl von Überlappungen in den Lösungen der Heuristiken liegen für größere λ sehr eng beisammen. Für die kleineren Werte dagegen liefern Heuristik 1 und 3 in der Regel bessere Resultate als die Heuristiken vom Typ 2.

Interessanter jedoch ist es, die Heuristiken bezüglich unterschiedlicher λ Werte zu vergleichen. Hierzu gehen wir folgermaßen vor:
Wir geben uns einen Parameter λ und eine Heuristik vor. Dies liefert uns einen Wert α für die abgeschätzte Verdrahtungslänge und einen Wert β für die Gesamtanzahl der Überlappungen in der Lösung. Nun suchen wir beispielsweise für alle anderen Heuristiken ein Testergebnis, in welchem die betreffende Heuristik eine abgeschätzte Verdrahtungslänge von nicht mehr als $(1+\epsilon)\alpha$ aufweist und vergleichen den Wert für die Überlappungen mit β. Umgekehrt suchen wir für alle anderen Heuristiken ein Testergebnis, in welchem die Anzahl Überlappungen in der Lösung der jeweiligen Heuristik den Wert $(1+\epsilon)\beta$ nicht übersteigt und vergleichen den Wert für die abgeschätzte Verdrahtungslänge mit α.
Betrachten wir zum Beispiel Tabelle 7.1.4. Hier liefert Heurstik 3 einen Wert von β = 3678570.0 und α = 167609.19. Wir setzen ϵ = 0.1%. Nun suchen wir für Heuristik 1, 2a und 3 nach einem Testbeispiel, in welchem die Überlappungen den Wert 3682248.570 nicht übersteigen. Für Heuristik 1 werden wir in Tabelle 7.1.3 fündig. Für die Heuristiken 2a und 2b liefert Tabelle 7.1.7 entsprechende Werte. Als Resultat ergibt sich, daß Heuristik 2a mit einem Wert von 126377.67 vor Heuristik 2b mit einem Wert von 149238.33 kommt. Den drittbesten Wert von 164162 liefert Heuristik 1. Schlechteste Lösung ist in diesem Beispiel somit die von Heuristik 3.

Bewertet man die Heuristiken nach diesem Vorgehen, so gelangt man zu dem Resultat, daß die Lösungen von Heuristik 2a denen der anderen vorzuziehen sind. Dabei müssen jedoch insbesondere in den ersten zwei Iterationen beträchtlich höhere Laufzeiten in Kauf genommen werden. Die Heuristiken 1, 2b und 3 sind im Vergleich hierzu wesentlich schneller, liefern aber schlechtere Ergebnisse. Unter diesen Heuristiken ist Heuristik 2b im allgemeinen der Gewinner. Insbesondere liefert sie in der ersten Iteration Ergebnisse mit deutlich besserer abgeschätzter Verdrahtungslänge.

Zusammenfassend läßt sich sagen, daß wir uns in allen Iterationen für Heuristik 2a entscheiden müssen, da es unser primäres Ziel ist, möglichst gute Lösungen zu erhalten. Dafür wird eine höhere Laufzeit in Kauf genommen.

7.2 Einstellung des Strafparameters für die Überlappungen

Die Testserie zu Kapitel 7.1 soll neben der Bewertung der Heuristiken noch dem Zweck dienen, ein Verfahren zu finden, wie der Parameter für die Überlappungen a-priori zu wählen ist, damit Heuristik 2a "gute Lösungen liefert. Hierzu analysieren wir zunächst noch einmal die Ergebnisse der Tests für die erste Iteration.
Im Beispiel der soc_2 Schaltung erkennen wir, daß für $\lambda \in \{0.02, 0.03, 0.04, 0.05, 0.1\}$ Heuristik 2a akzeptable Lösungen liefert. Dabei können wir feststellen, daß der Quotient aus "λ mal Anzahl der Überlappungen" und dem Wert für die abgeschätzte Verdrahtungslänge in dem Intervall $[0.65, 3]$ liegt. Die Tests für die erste Iteration bezüglich der Schaltung soc_3 liefern uns, daß die besten Ergebnisse bei einem λ-Wert im Bereich $[0.02, 0.05]$ erzielt werden. Daraus resultiert das Intervall $[1.18, 2.29]$ für den Quotienten aus "λ mal Anzahl der Überlappungen" und dem Wert für die abgeschätzte Verdrahtungslänge. Somit können wir beispielsweise einen Schätzwert von $\hat{q} = 1.5$ für den Quotienten aus "λ mal Anzahl der Überlappungen" und dem Wert für die abgeschätzte Verdrahtungslänge in einer guten Lösung ansetzen. Wenn wir also ein schnelles Verfahren entwickeln könnten, welches uns eine Schätzung der Werte für die Anzahl von Überlappungen und die abgeschätzte Verdrahtungslänge in einer guten Lösung liefert, so versetzt uns das in die Lage, einen vernünftigen Parameter λ für die erste Iteration aus folgender Formel zu bestimmen:

$$\lambda = \hat{q}\frac{\hat{\alpha}}{\hat{\beta}}.$$

$\hat{\alpha}$ bezeichnet darin die Schätzung der Verdrahtungslänge und $\hat{\beta}$ die Schätzung für den Wert der Überlappungen in einer guten Lösung.
Im Fall der soc_2-Schaltung liegt die abgeschätzte Verdrahtungslänge für eine gute Lösung in dem Intervall $[115000, 140000]$. Bei der soc_3-Schaltung erhalten wir das Intervall $[41000, 580000]$ für die abgeschätzte Verdrahtungslänge in einer "guten" Lösung. Betrachten wir nun einmal die Testergebnisse der Heuristik 2b für $\lambda = 0$, so erhalten wir eine abgeschätzte Verdrahtungslänge von 108931.92 und 420251.25 für beide Beispiele. Die

Werte entsprechen genau dem, was wir gesucht haben, nämlich einem Schätzwert für die Verdrahtungslänge in einer guten Lösung. Somit entscheiden wir uns anhand der Tests, den Wert für die abgeschätzte Verdrahtungslänge der von Heuristik 2b für $\lambda = 0$ gelieferten Lösung als $\hat{\alpha}$ zu wählen.
Ähnlich erkennt man recht schnell, daß für große λ-Werte (beispielsweise $\lambda = 1$) Heuristik 1 stets eine Lösung mit wenigen Überlappungen liefert. Somit definieren wir diesen Wert als $\hat{\beta}$. Da die Laufzeiten für Heuristiken 1 und 2b im Vergleich zu Heuristik 2a niedrig sind, haben wir ein effizientes Verfahren gefunden, um einen Anfangsparameter λ zu bestimmen.

Nun bleibt noch die Aufgabe, aus einem λ-Parameter für Iteration j den Parameter für die nachfolgende Iteration zu bestimmen.
Dazu vergegenwärtigen wir uns noch einmal den Ablauf der Dekompositionsverfahren. Bezeichne B^j die Menge der repräsentativen Basiszellen in der gegenwärtigen Iteration j (dies sei nicht die letzte Iteration).
Falls wir davon ausgehen, daß λ für Iteration j richtig gewählt war, d.h. die Lösung in dieser Iteration gut ist, so erscheint es vernünftig, den Parameter $\frac{|B^{j+1}|}{|B^j|}\lambda$ als Bestrafungsparameter für Iteration $j+1$ zu wählen. Dies ist insofern plausibel, als wir erwarten können, daß sich in guten Lösungen die Zellen auf die zur Verfügung stehenden Basiszellen gleichmäßig verteilen. Somit würden wir, falls sich die Werte für die abgeschätzte Verdrahtungslänge in Folgeiterationen nur geringfügig ändern, erreichen, daß der Quotient aus gewichteten Überlappungen durch abgeschätzte Verdrahtungslänge von einer Iteration auf die nächste nur wenig abweicht. Dabei muß vorausgesetzt werden, daß die Anzahl von Überlappungen von Null verschieden ist. Falls wir aber in einer Iteration eine überlappungsfreie Lösung finden, erübrigt sich die Fragestellung, um welchen Faktor der Bestrafungsparameter erhöht werden muß.
Um diese Hypothese zu überprüfen, analysieren wir die Testserie zu Abschnitt 7.1. In diesen Beispielen gilt, daß sich die Anzahl repräsentativer Basiszellen von einer Iteration auf die folgende vervierfacht. Gemäß obigen Ausführungen müßten wir ausgehend von einer guten Lösung in einer Iteration eine gute Lösung für die nächste Iteration finden (mit Heuristik 2a), indem der Bestrafungsparameter um den Faktor 4 erhöht wird. Wählen wir beispielsweise die Lösungen von Heuristik 2a für $\lambda = 0.02$ in der ersten Iteration bezüglich der Schaltungen soc_2 und soc_3, so bestätigen die Testergebnisse unsere Hypothese.

Somit haben wir eine Methode gefunden, um in allen Iterationen den Parameter für die Überlappungen a-priori vernünftig zu wählen.

7.3 Berücksichtigung der Verdrahtbarkeitsbedingung

Bei der Modellierung des Plazierungsproblems hat sich herausgestellt, daß es $\mathcal{NP}$-vollständig ist, zu entscheiden, ob eine gegebene Plazierung verdrahtbar ist. Wir haben ferner festgestellt, daß Plazierungsverfahren im allgemeinen versuchen, eine Abschätzung der Verdrahtungslänge und gegebenfalls die Gesamtfläche zu minimieren. Die Verdrahtbarkeitsbedingung bleibt dabei unberücksichtigt. Vielmehr werden bei Plazierungsverfahren

durch die Zielfunktion solche Plazierungen bevorzugt, die die Zellen möglichst dicht packen. Dadurch verkleinert sich die Fläche des minimalen Rechtecks, welches alle Zellen beinhaltet. Dies aber impliziert, daß die Fläche, die für die Verdrahtung innerhalb des minimal einschließenden Rechtecks zur Verfügung steht, abnimmt. Sind, wie beispielsweise im "sea of cells"-Entwurfsverfahren, die Ausmaße des Masters vorgegeben, so bewirkt eine zu dichte Packung, daß manche Netze nicht mehr innerhalb des minimal einschließenden Rechtecks der Zellen realisiert werden können. Somit müssen entweder lange Umwege in Kauf genommen werden, oder Netze können überhaupt nicht mehr auf dem Master realisiert werden. Abbildung 7.3.1 zeigt zwei schraffiert gezeichnete Zellen, die durch ein Netz verbunden werden müssen. Da eine der beiden Zellen von anderen Zellen eingeschlossen wird, ist diese Verbindung nicht mehr zu realisieren.

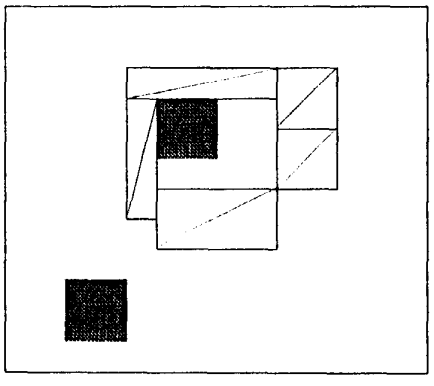


Abbildung 7.3.1

Im Fall des von uns hier vorgestellten Plazierungsverfahrens werden in den Anfangsiterationen (die Anzahl repräsentativer Basiszellen ist kleiner als die Anzahl Zellen) Lösungen bevorzugt, welche die Zellen gleichmäßig auf die zur Verfügung stehenden Basiszellen verteilen. Sind in tieferen Iterationen mehr repräsentative Basiszellen als Zellen vorhanden, so stellt der Plazierungsalgorithmus lediglich sicher, daß überlappungsfreie Plazierungen gegenüber solchen mit Überlappungen belohnt werden. Insbesondere wird in den Enditerationen der Bestrafungsparameter für Überlappungen so gewählt, daß die Zellen nicht so dicht gepackt werden, daß sie sich überlappen.
Die Minimierung der abgeschätzten Verdrahtungslänge in den Lösungen von Instanzen (P_m^+) bewirkt jedoch, daß dichtere Packungen der Zellen bevorzugt werden.
Aus diesem Grund haben wir eine einfache Methode entwickelt, um solche Situationen zu vermeiden. Die Idee dabei ist, daß wir in den Enditerationen sukzessive Zellen künstlich vergrößern, bis ein vorgegebener Wert erreicht wird. Da wir die Verdrahtbarkeitsbedingung berücksichtigen wollen, hängt das Kriterium, welche Zellen um welchen Betrag vergrößert werden sollen, ab von der Anzahl Netze, mit denen die Zelle verbunden ist, beziehungsweise von der Anzahl Pins der Zelle.

Konkret sei c eine Zahl mit $0 \leq c \leq 1$. Dann definieren wir die **Erweiterung einer Schaltung** als $c(1-\rho)b$, wobei b die Anzahl Basiszellen des Masters und ρ die Dichte der Schaltung bezeichnet. Die Zahl c nennen wir **Erweiterungsfaktor**. Der Ausdruck $(1-\rho)b$ ist gemäß Definition der Dichte einer Schaltung die Anzahl Basiszellen, welche in einer Plazierung nicht von Zellen überdeckt werden.
Für jedes Pin einer Zelle bestimmen wir den äußeren Rand mit dem kleinsten Abstand zu den Pins. Dies definiert für jeden äußeren Rand einer Zelle eine Anzahl von Pins. Anschließend sortieren wir die Ränder der Zellen in absteigender Reihenfolge. Diese sortierte Liste durchlaufen wir und erweitern sukzessive Breite oder Höhe der entsprechenden Zelle. Wenn wir beim Abarbeiten der Liste an der k-ten Stelle angelangt sind, so gehen wir folgendermaßen vor:
Sei i die Zelle, zu welcher der Rand R an der k-ten Stelle der Liste gehört. Falls R oberer oder unterer Rand ist, vergrößern wir die Höhe der Zelle um 1. Dadurch erhöht sich die Anzahl der von einer Plazierung belegten Basiszellen gerade um die Breite der Zelle. (vergleiche Abbildung 7.3.2). Anderenfalls wird die Breite von i um 1 erhöht.

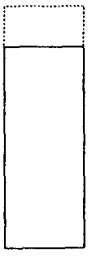


Abbildung 7.3.2

Das Verfahren bricht ab, wenn entweder alle Ränder abgearbeitet wurden oder die nächste Vergrößerung einer Zelle den vorgegebenen Wert der Erweiterung übersteigen würde.

Auf den folgenden Seiten haben wir für verschiedene Erweiterungsfaktoren die Ergebnisse zusammengestellt. Dabei vergleichen wir die Endplazierungen, wobei Heuristik 2a und die beiden Verbesserungsverfahren in jeder Iteration angewendet wurden. Als Testschaltungen dienten uns die Beispiele soc_1 bis soc_4. soc_1 besteht aus 602 Zellen, 44 Randzellen und 820 Netzen. Ihre Dichte beträgt 19.2%. Die Schaltung soc_2 besteht aus 1022 Zellen, 63 Randzellen und 1025 Netzen und hat eine Dichte von 50%. Das Beispiel soc_3 besteht aus 2293 Zellen, 2776 Netzen und 157 vorplazierten Randzellen. Die Dichte von soc_3 beträgt 23.6%. Schließlich untersuchen wir noch die Ergebnisse bei der Testschaltung soc_4. Diese besteht aus 2670 Zellen, 3128 Netzen und 100 Randzellen. Die Dichte beträgt 53%. Zur Bewertung der Plazierung haben wir ein in der Industrie eingesetztes Verdrahtungsprogramm verwendet.
Eine erstes Maß für die Qualität einer Plazierung ist die geschätzte Verdrahtungslänge, welche sich als die Summe der aufspannenden Bäume für jedes Netz berechnet. Dies stellt

lediglich eine Abschätzung dar, da die Netze als Steiner Bäume verdrahtet werden und daher eventuelle Umwege für bestimmte Netze in Kauf genommen werden müssen, damit der Chip verdrahtbar wird. Nichtsdestoweniger ist diese Zahl ein akzeptiertes Maß, um Plazierungen miteinander zu vergleichen.
Zusätzliches verwendeten wir einen Maze-Router, der in der Industrie eingesetzt wird, damit eine tatsächliche Verdrahtung des Chips realisiert werden konnte. Dieser Verdrahter kann nur selten eine überlappungsfreie Plazierung verdrahten. Ein interessante Maßzahl dabei ist die Anzahl von zwei-Pin-Verbindungen, welche von dem Maze-Router nicht realisiert werden konnten. Dies entspricht dem Arbeitsaufwand, der noch interaktiv zu erbringen ist, um den Chip produktionsfähig zu machen.

Tabelle 7.3.3 stellt die Ergebnisse unter Verwendung eines Erweiterungsfaktors von $c = 20\%$ dar. In Spalte 2 ist die abgeschätzte Verdrahtungslänge angegeben. In Spalte 3 werden die Anzahl nicht realisierter zwei-Pin-Verbindungungen dargestellt und Spalte 4 zeigt die CPU-Zeit für eine Plazierung der entsprechenden Schaltungen. Letztere ist in "Minuten: Sekunden" gemessen. Die Abbildung 7.3.5 zeigt die Endplazierung der Schaltung soc_2, wobei obiger Erweiterungsfaktor verwendet wurde.
Analog sind die folgenden Tabellen und Abbildungen zu lesen. Die Tabellen 7.3.4 beziehungsweise 7.3.7 entsprechen den Ergebnissen mit einem Erweiterungsfaktor von $c = 40\%$ beziehungsweise $c = 60\%$. Die Abbildungen 7.3.6 und 7.3.8 veranschaulichen die Plazierungen der Schaltung soc_2, wobei der Parameter c den Wert $c = 40\%$ beziehungsweise $c = 60\%$ annimmt.

$c = 20\%$	gesch. Verdrahtungsl.	nicht verbunden	CPU
soc_1	159492	23	12:13
soc_2	180101	17	26:07
soc_3	5535750	48	129:39
soc_4	472473	309	159:22

Tabelle 7.3.3

$c = 40\%$	gesch. Verdrahtungsl.	nicht verbunden	CPU
soc_1	163919	16	12:11
soc_2	185592	14	26:08
soc_3	570387	57	129:24
soc_4	486175	278	159:21

Tabelle 7.3.4

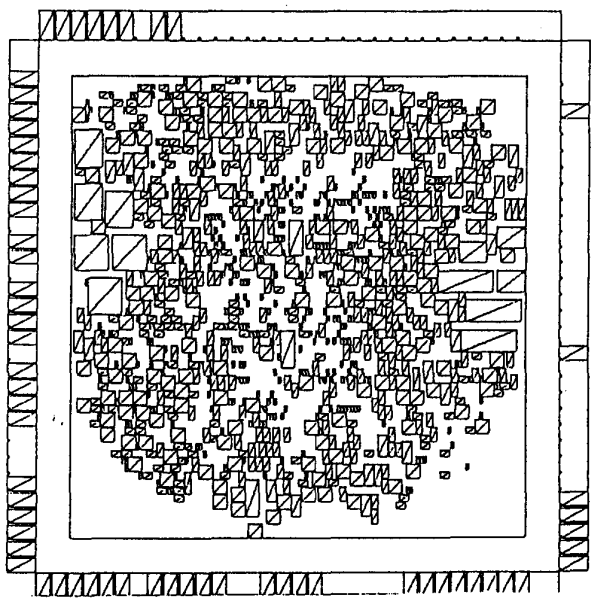


Abbildung 7.3.5

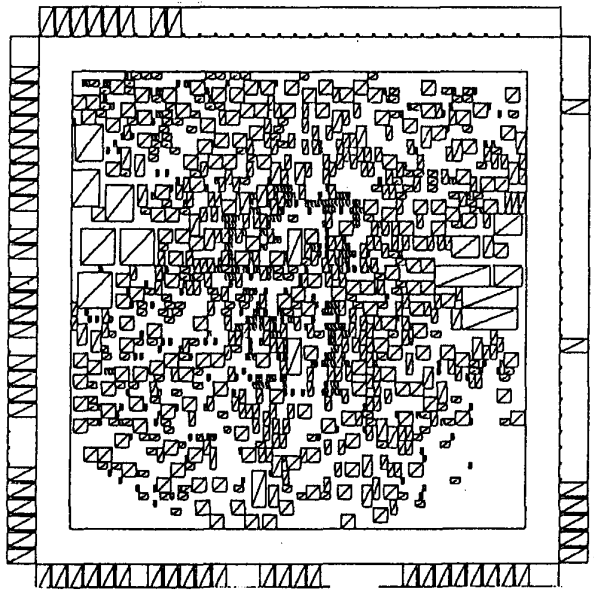


Abbildung 7.3.6

$c = 60\%$	gesch. Verdrahtungsl.	nicht verbunden	CPU
soc_1	169892	14	12:12
soc_2	191577	14	26:09
soc_3	576845	50	129:37
soc_4	497285	260	159:23

Tabelle 7.3.7

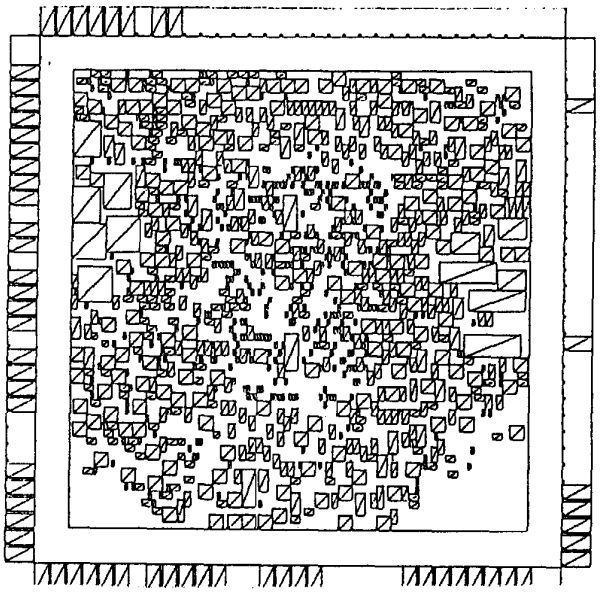


Abbildung 7.3.8

Die Ergebnisse zeigen ganz deutlich, daß mit wachsendem Erweiterungsfaktor die geschätzte Verdrahtungslänge zunimmt. Tendentiell nimmt hingegen bei größerem Erweiterungswert die Anzahl nicht realisierter zwei-Pin-Verbindungen ab. Eine Ausnahme bildet hier beispielsweise die Schaltung soc_3, denn die Anzahl nicht realisierter zwei-Pin-Verbindungen erhöht sich von 48 ($c = 20\%$) auf 57 ($c = 40\%$) beziehungsweise auf 50 ($c = 60\%$).
Diese Ausnahmen lassen sich jedoch auf den Maze-Router zurückführen, der versucht, die Netze auf dem Chip nach lokalen Gesichtspunkten zu legen.
Desweiteren führen die Tests zu dem Ergebnis, daß der Erweiterungsfaktor in Abhängigkeit von der konkreten Schaltung zu wählen ist. Ein Faktor c = 20% empfiehlt sich für das Beispiel soc_3. Für soc_2 erscheint die Wahl $c = 40\%$ optimal, wohingegen für die Schaltungen soc_1 und soc_4 ein Erweiterungsfaktor von c = 60% die besten Ergebnisse liefert. Ausgehend von diesen Ergebnissen stellen wir den Parameter c folgendermaßen ein:

Für Schaltungen mit homogener Struktur der Zellen und geringer Dichte ($< 30\%$) wählen wir den Parameter $c = 20\%$.
Für Schaltungen mit sehr heterogener Struktur oder einer Dichte über 50% setzten wir $c = 60\%$.
In allen anderen Fällen gehen wir von einem Parameter $c = 40\%$ aus.

7.4 Vergleich der Dekompositionsverfahren

Den Abschluß in unserer Testserie bildet der Vergleich der in Kapitel 5 vorgestellten Dekompositionsverfahren. Ausgangspunkt werden wiederum die Schaltungen soc_1 bis soc_4 sein. Als Bewertungskriterium werden die Endplazierungen der Schaltungen unter Verwendung des jeweiligen Dekompositionsverfahrens herangezogen.
Zur Lösung der Instanzen von (P_m^+) verwenden wir in allen Iterationen Heuristik 2a. Der Erweiterungsfaktor wird entsprechend des Ergebnisses in Kapitel 7.3 für die jeweiligen Schaltungen gesetzt.
Somit ergibt sich für die Schaltung soc_3 ein Erweiterungsfaktor von $c = 20\%$. Die Schaltung soc_2 wird um 40% erweitert, und für die Schaltungen soc_1 und soc_4 setzten wir $c = 60\%$.
Der Anfangsparameter für die Überlappungen ist entsprechend den Ausführungen in Abschnitt 7.2 gesetzt. In Iterationen der Tiefe größer oder gleich 2 vervierfacht sich der Bestrafungsparameter gegenüber der vorangegangenen Iteration.

Zunächst erwartet man, daß bei Anwendung der dynamischen Dekompositionsmethode aus Kapitel 5.3 eine gleichmäßigere Verteilung der Zellen auf Kosten einer schlechteren abgeschätzten Verdrahtungslänge erreicht wird (gegenüber einer statischen Dekomposition). Dies liegt darin begründet, daß wir es einerseits in unterbesetzten Gebieten Zellen verbieten, in alle Richtungen wandern zu dürfen. Andererseits werden die zulässigen repräsentativen Basiszellen einer Zelle, die in der Lösung der gegenwärtigen Iteration in einem dicht besetzten Gebiet plaziert wurde, so gewählt, daß die Zelle, wenn möglich, in unterbesetzte Gebiete abwandert.
Da wir versuchen, eine Strategie zu finden, nach der alle Parameter des Plazierungsalgorithmus a-priori gesetzt werden können, testen wir zunächst die Dekompositionsverfahren der Abschnitte 5.1 und 5.2 gegeneinander. Dies führt zu einer klaren Entscheidung für eines der beiden Verfahren. Anschließend testen wir das bessere der beiden Dekompositionsverfahren an den obigen Schaltungen mit und ohne Verwendung von dynamischer Dekomposition.
In den Tabellen 7.4.1 bis 7.4.4 sind in der zweiten Zeile die geschätzte Verdrahtungslänge, die Anzahl nicht verbundener 2-er Verbindungen und die CPU Zeiten, gemessen in Minuten, unter Verwendung des Dekompositionsverfahren aus Kapitel 5.1 dargestellt. Die dritte Zeile entspricht jeweils den Resultaten unter Verwendung des in Abschnitt 5.2 beschriebenen Verfahrens.
In Tabelle 7.4.1 werden die Endplazierung der Schaltung soc_1, in Tabelle 7.4.2 die der Schaltung soc_2 und in Tabelle 7.4.3 die der Schaltung soc_3 miteinander verglichen. Die

Ergebnisse für soc_4 sind entsprechend in Tabelle 7.4.4 aufgezeigt.

Die anschließenden Abbildungen 7.4.5 beziehungsweise 7.4.6 zeigen die Plazierungen der Schaltung soc_2 unter Verwendung der Verfahren aus den Abschnitten 5.1 beziehungsweise 5.2:

soc_1	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	169892	14	12:12
9 Einbauplätze	166384	35	6:48

Tabelle 7.4.1

soc_2	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	185592	14	26:08
9 Einbauplätze	189363	17	13:57

Tabelle 7.4.2

soc_3	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	553575	48	129:39
9 Einbauplätze	729729	62	69:02

Tabelle 7.4.3

soc_4	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	497285	260	159:23
9 Einbauplätze	457004	292	84:03

Tabelle 7.4.4

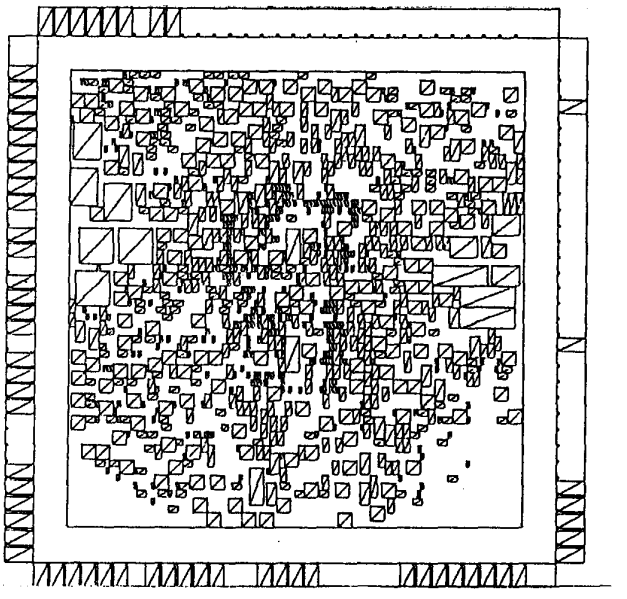


Abbildung 7.4.5

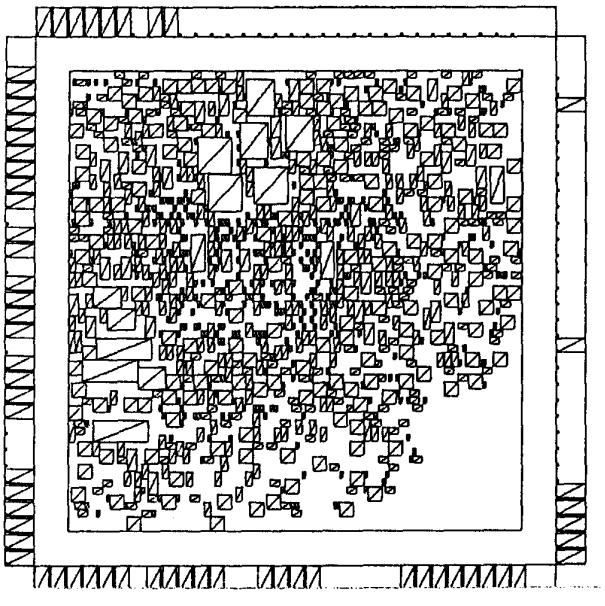


Abbildung 7.4.6

Die Bilder zeigen in völliger Übereinstimmung mit den Zahlen aus Tabelle 7.4.2, daß die Zerlegung des Plazierungsproblems in eine Folge von Problemen (P_{16}^{+}) eine weitaus bessere Verteilung der Zellen auf dem Master erreicht. Dies korreliert mit der Anzahl nicht verbundener Pinpaare. Im Beispiel der soc_3 Schaltung ist die abgeschätzte Verdrahtungslänge in

der Lösung, die unter Verwendung des Verfahrens dec_{16} zustandekam, deutlich besser. In den Beispielen soc_1 und soc_2 sind die beiden Lösungen bezüglich der abgeschätzten Verdrahtungslänge vergleichbar. Für die Schaltung soc_4 liefert das Verfahren aus Abschnitt 5.2 eine Lösung geringerer abgeschätzter Verdrahtungslänge, als das bei dem Dekompositionsverfahrens aus Abschnitt 5.1 der Fall ist. Zusammenfassend bewerten wir die Ergebnisse so, daß das Verfahren dec_{16} als Gewinner aus diesem Vergleich hervorgeht. Dafür nehmen wir eine höhere Laufzeit in Kauf.

soc_1	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	169892	14	12:12
16 Ebp, dynam.	176851	9	12:59

Tabelle 7.4.7

soc_2	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	185592	14	26:08
16 Ebp, dynam.	185237	46	28:47

Tabelle 7.4.8

soc_3	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	553575	48	129:39
16 Ebp, dynam.	580876	45	133:15

Tabelle 7.4.9

soc_4	gesch. Verdrahtungsl.	nicht verbunden	CPU
16 Einbauplätze	497285	260	159:23
16 Ebp, dynam.	528759	449	167:40

Tabelle 7.4.10

Auf dieser und der nächsten Seite sind die Ergebnisse des Vergleichs der Dekompositionsverfahren dec_{16} mit und ohne Verwendung dynamischer Dekomposition zusammengefasst. Die Tabellen 7.4.7 bis 7.4.10 entsprechen der Bewertung der Endplazierungen bezüglich der Schaltungen soc_1 bis soc_4. In der zweiten Zeile sind jeweils die entsprechenden Werte unter Verwendung des Dekompositionsverfahrens aus Kapitel 5.1 dargestellt. Die Zahlen

in Zeile 3 sind unter Verwendung des Algorithmus dec_{16} und dynamischer Dekomposition zustandegekommen.
Die Abbildungen 7.4.11 beziehungsweise 7.4 12 stellen die Endplazierung der Schaltung soc_2 im Vergleich dar.

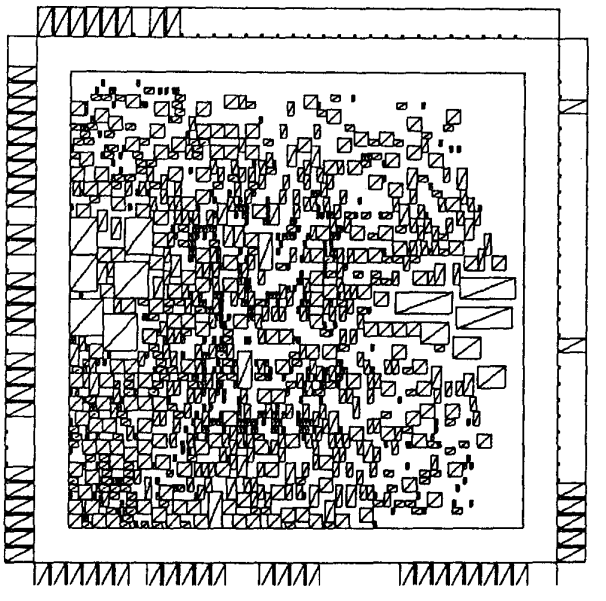


Abbildung 7.4.11

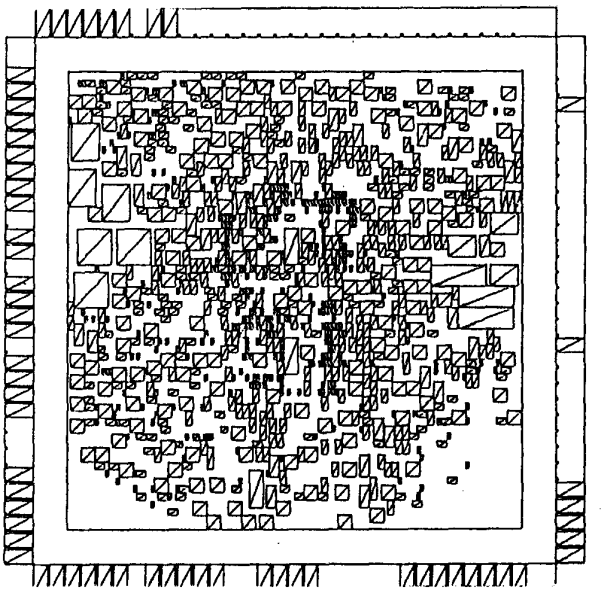


Abbildung 7.4.12

Die Ergebnisse zeigen deutlich, daß in allen Beispielen die geschätzte Verdrahtungslänge in den Lösungen, die ohne dynamische Dekomposition zustandekamen, besser ist als unter Verwendung dynamischer Dekomposition (sieht man von dem geringfügigen Unterschied bei den Plazierungen der Schaltung soc_2 ab).
Ferner ist die Anzahl nicht verbundener Pinpaare in allen Beispielen kleiner oder geringfügig schlechter, wenn das statische Verfahren dec_{16} verwendet wird. Dies zeigt sich deutlich an den Plazierungen der Schaltungen soc_2 und soc_4. Ausnahmen bilden die Plazierungen der Schaltung soc_1 und soc_3, bei denen in der Lösung, die mit dynamischer Dekomposition zustandekam, 5 bzw. 3 Pinpaare mehr verbunden werden konnten.
Die Ergebnisse lassen es vernünftig erscheinen, das Dekompositionsverfahren aus Kapitel 5.1 zur Lösung aller Plazierungsinstanzen einzusetzen.

Somit haben wir alle Parameter beziehungsweise deren a-priori Berechnung in unserem Plazierungsprogramm bestimmt. Dies gestattet, in dem folgenden Kapitel den Plazierungsalgorithmus mit anderen in der Industrie eingesetzten Verfahren zu vergleichen.

Kapitel 8 Vergleich des Plazierungsverfahrens

Nachdem wir das von uns entwickelte Plazierungsverfahren im Detail dargestellt haben, stellt sich die Frage, inwieweit dieses Verfahren mit anderen Algorithmen konkurrieren kann. Dieser Frage soll im folgenden nachgegangen werden.

Unsere experimentelle Software wurde mit industriell eingesetzten Plazierungsalgorithmen verglichen. Die Kriterien hierfür sind bereits in Abschnitt 7.4 erläutert worden. Der Vollständigkeit halber sollen sie an dieser Stelle noch einmal wiederholt werden. Die von den jeweiligen Algorithmen gelieferten Lösungen wurden auf zwei Weisen bewertet.
Ein erstes Maß für die Qualität einer Plazierung ist die geschätzte Verdrahtungslänge. Diese berechnet sich aus der Summe der aufspannenden Bäume für jedes Netz. Dies stellt lediglich eine Abschätzung dar, da die Netze als Steiner Bäume verdrahtet werden, und daher eventuelle Umwege für bestimmte Netze in Kauf genommen werden müssen, damit der Chip verdrahtbar wird. Nichtsdestoweniger ist diese Zahl ein akzeptiertes Maß, um Plazierungen miteinander zu vergleichen.
Zusätzlich verwendeten wir einen Maze-Router, der in der Industrie eingesetzt wird, damit eine tatsächliche Verdrahtung des Chips realisiert werden konnte. Dieser Maze-Router wurde als eine "black box" verwendet, d.h. wir konnten diesen Verdrahtungsalgorithmus nicht an unser Plazierungsverfahren anpassen. Dieser Verdrahter kann nur selten eine überlappungsfreie Plazierung verdrahten. Ein interessante Maßzahl dabei ist die Anzahl von zwei-Pin-Verbindungen, welche von dem Maze-Router nicht realisiert werden konnten. Dies entspricht dem Arbeitsaufwand, der noch interaktiv zu erbringen ist, um den Chip produktionsfähig zu machen. Unsere Erfahrungen mit diesem Router haben gezeigt, daß die Plazierungen die Anzahl nicht realisierter 2-Pin-Verbindungen nur in einem bestimmten Rahmen beeinflussen. Viele nicht verbundene Pins enstehen vielmehr aus Situationen, die durch die "Greedy"-Vorgehensweise des Routers verursacht sind, denn alle Paare von Pins werden von dem Maze-Router in einer bestimmten Reihenfolge abgearbeitet. Somit wird die Verbindung nachfolgender Paare von Pins unter Umständen blockiert.
Trotzdem ist diese Anzahl ein Anzeichen für die Verdrahtbarkeit einer gegebenen Plazierung, so daß die abgeschätzte Verdrahtungslänge in Kombination mit der Anzahl nicht realisierter Verbindungen eine vernünftige Bewertung verschiedener Plazierungsansätze erlaubt.

Zur Bewertung unseres Verfahrens ziehen wir sowohl einen Plazierungsalgorithmus heran, welcher auf der Min-Cut-Idee (vergleiche Kapitel 2.4) beruht als auch das Verfahren Gordian. Letzteres geht von einer Modellierung des Plazierungsproblems als Kräftemodell aus (vergleiche Kapitel 2.2).
Wir berichten hier über die Ergebnisse mit den Testschaltungen soc_1 bis soc_4, die in den Abschnitten 7.3 und 7.4 ausführlich vorgestellt wurden. Für diese Beispiele standen uns Vergleichsdaten von Min-Cut und Gordian zur Verfügung. Fehlende Einträge, d.h.

Testläufe, für die uns keine Daten zur Verfügung standen, sind durch einen Gedankenstrich gekennzeichnet.

Tabellen 8.1 bis 8.4 zeigen die Ergebnisse bezüglich der Schaltungen soc_1 bis soc_4. soc_1 besteht aus 602 Zellen, 44 Randzellen und 820 Netzen. Ihre Dichte beträgt 19.2%. Die Schaltung soc_2 besteht aus 1022 Zellen, 63 Randzellen und 1025 Netzen und hat eine Dichte von 50%. Das Beispiel soc_3 besteht aus 2293 Zellen, 2776 Netzen und 157 vorplazierten Randzellen. Die Dichte von soc_3 beträgt 23.6%. Schließlich untersuchen wir noch die Plazierung der Schaltung soc_4. Diese besteht aus 2670 Zellen, 3128 Netzen und 100 Randzellen. Die Dichte beträgt 53%. In der ersten Zeile sind jeweils die Ergebnisse des Min-Cut-Verfahrens dargestellt. Die zweite Zeile in den Tabellen 8.1 bis 8.4 entspricht den Ergebnissen einer Plazierung mit Gordian. In der letzten Zeile wird schließlich die Bewertung der von unserem Verfahren gelieferten Lösungen vorgenommen (Quazo steht für "quadratic zero-one optimization"). Die Ergebnisse sind mit der in den Abschnitten 7.1 bis 7.4 angegebenen Parametereinstellung erzielt worden. In diesen Abschnitten wurden die Berechnung des Strafparameters für die Überlappungen und die Bestimmung des Erweiterungsfaktors dargestellt sowie eine Entscheidung bezüglich der Auswahl der Heuristiken und des Dekompositionsverfahrens getroffen. In Spalte 2 der nachfolgenden Tabellen ist die abgeschätzte Verdrahtungslänge angegeben. Spalte 3 entspricht der Anzahl nicht verbundener Pinpaare. In der letzten Spalte stehen die Laufzeiten, gemessen in Minuten, die für die Plazierung mit den entsprechenden Verfahren benötigt wurden. Die Abbildungen 8.5, 8.6, 8.7 und 8.8 stellen die Plazierung der Schaltungen soc_1, soc_2, soc_3 und soc_4 unter Verwendung unseres Verfahrens dar.

soc_1	gesch. Verdrahtungsl.	nicht verbunden	CPU
Min-Cut	212258	-	-
Gordian	184045	9	-
Quazo	176851	14	12:12

Tabelle 8.1

soc_2	gesch. Verdrahtungsl.	nicht verbunden	CPU
Min-Cut	194732	24	4:25
Gordian	189683	19	3:43
Quazo	185592	14	28:47

Tabelle 8.2

soc_3	gesch. Verdrahtungsl.	nicht verbunden	CPU
Min-Cut	796622	33	15:54
Gordian	652129	51	3:13
Quazo	553575	48	129:39

Tabelle 8.3

soc_4	gesch. Verdrahtungsl.	nicht verbunden	CPU
Min-Cut	623159	551	21:04
Gordian	506160	253	6:52
Quazo	497285	260	159:23

Tabelle 8.4

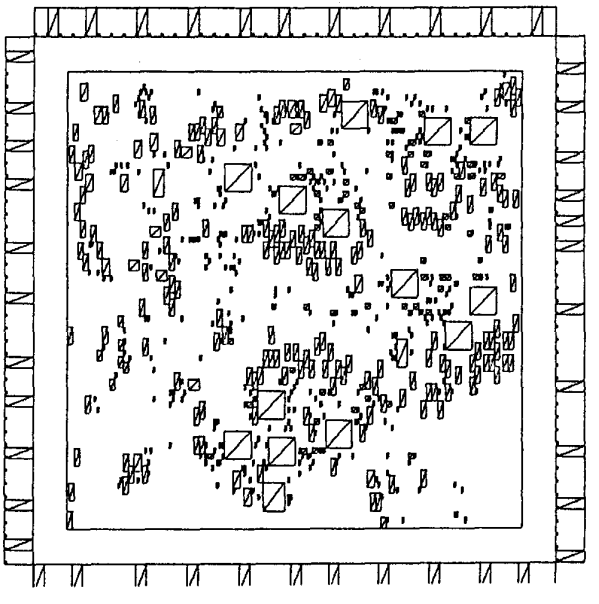


Abbildung 8.5

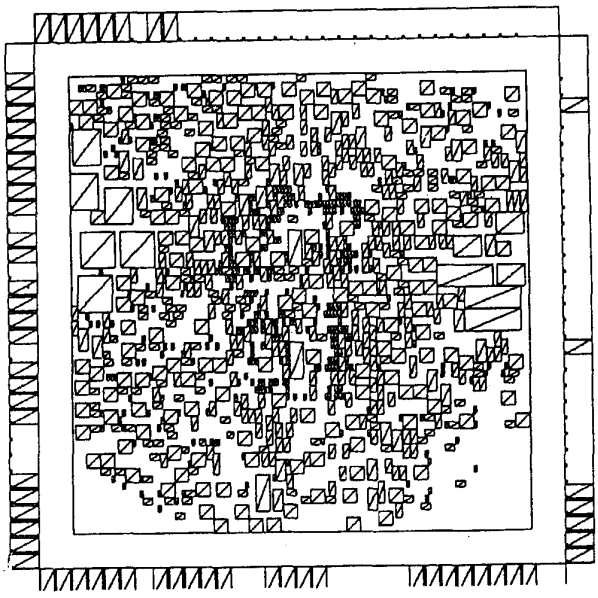


Abbildung 8.6

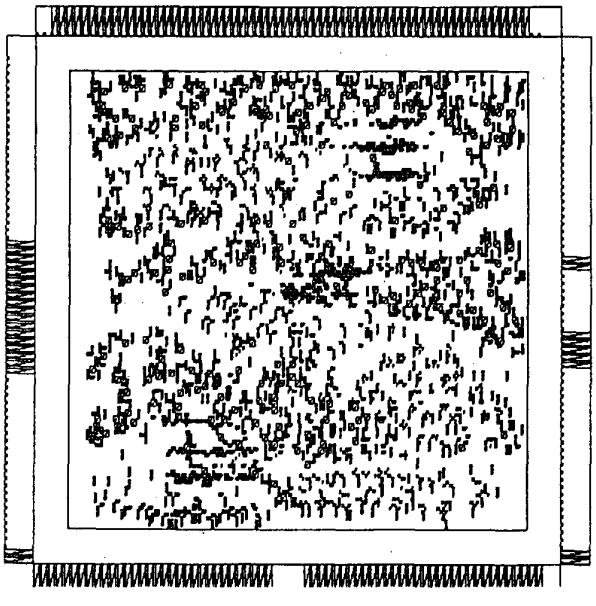


Abbildung 8.7

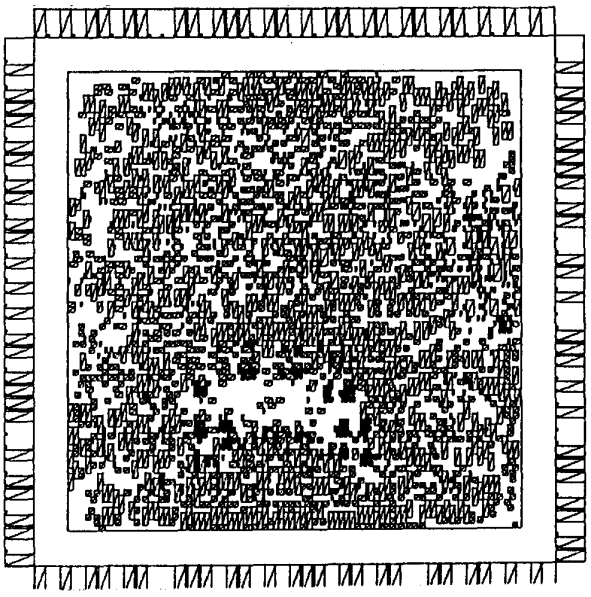


Abbildung 8.8

Zunächst zeigt sich ganz deutlich, daß insbesondere bei den größeren Schaltungen soc_3 und soc_4 das Min-Cut-Verfahren wesentlich schlechtere Ergebnisse liefert als Gordian oder das von uns im Rahmen dieser Arbeit vorgestellte Verfahren.
In den Beispielen soc_1 und soc_4 liefert Quazo gegenüber Gordian Lösungen, die nur geringfügig mehr nicht verbundene Pinpaare aufweisen. Dafür ist der Wert für die abgeschätzte Verdrahtungslänge besser. Die von Gordian gelieferten Plazierungen der Schaltungen soc_2 und soc_3 weisen bezüglich beider Bewertungskriterien schlechtere Werte auf als bei einer Plazierung mit unserem Verfahren. Im Fall der Schaltung soc_3 ist die abgeschätzte Verdrahtungslänge unserer Lösung sogar um 16% besser als die von Gordian.
Die Laufzeiten unseres Algorithmus sind deutlich höher als die der anderen Verfahren. Sie sind jedoch in jedem Fall akzeptabel, wenn man sich vergegenwärtigt, daß es zwischen 6 Stunden im Fall der Schaltung soc_1 und 35 Stunden für das Beispiel soc_4 benötigt, um eine Plazierung zu verdrahten. Dabei gehen wir von dem uns zur Verfügung gestellten Verdrahtungsprogramm aus, welches auf einer Workstation installiert war.
Insbesondere war es nicht unser Ziel, einen möglichst schnellen Plazierungsalgorithmus zu entwerfen. Vielmehr war uns wichtig, ein Verfahren zu entwickeln, das auf einer Modellierung des Problems, in der zahlreiche technische Restriktionen berücksichtigt werden können (vergleiche hierzu Kapitel 3), basiert. Die Methode sollte ferner bei Anwendung aud den "Sea of cells"-Entwurfsstil zu guten Ergebnissen führen.
Zieht man in Betracht, daß unsere Implementierung ein Prototyp ist, in der zum Beispiel technische Details der Schaltung teilweise unberücksichtigt bleiben (vergleiche beispielsweise den Erweiterungsfaktor, Kapitel 7.4) und die Laufzeit nicht optimiert wurde, sind wir – unserer Meinung nach – unserer ursprünglichen Zielvorgabe gerecht geworden. Insbesondere rechtfertigen die Resultate, daß der vorgelegte Ansatz weiter verfolgt werden kann.

Kapitel 9 Lösung einer linearen Relaxierung von (P)

Wir haben bereits festgestellt, daß zur Lösung des quadratischen 0/1 Optimierungsproblems (P) keine Algorithmen existieren, welche für alle Instanzen eine beweisbar gute Lösung in polynomialer Zeit finden (sofern $\mathcal{P} \neq \mathcal{NP}$ vorausgesetzt wird). Andererseits haben wir im Teil 2 dieser Dissertation Heuristiken vorgestellt, die – wie die Ergebnisse zeigen – vernünftige Resultate für unsere speziellen Anwendungen liefern. Aus theoretischer Sicht stellt sich deshalb die Frage, ob es nicht möglich ist, nicht triviale untere Schranken für verschiedene Instanzen von (P_m) zu finden.

In diesem Kapitel greifen wir auf die Notation im ersten Teil der Arbeit zurück. Insbesondere bezeichne $N = \{1, \ldots, n\}$ die Menge der Zellen und $Z(i)$ die Menge der zulässigen Einbauplätze für i. Ferner sei $G_C = (V_C, E_C, w_C)$ der in Kapitel 3.2 eingeführten Cliquengraphen. Ein Knoten $ik \in V_C$ repräsentiert die Zelle i auf Einbauplatz $k \in Z(i)$. Mit $S_i = \{ik \mid k \in Z(i)\}$ bezeichnen wir die maximalen stabilen Mengen in G_C $(i = 1, \ldots, n)$. Betrachten wir zunächst das in Abschnitt 3.2 definierte Cliquenproblem:

(CP) Finde unter allen Cliquen der Kardinaltät n in $G_C = (V_C, E_C, w_C)$ eine minimalen Gewichts.

In Kapitel 3 haben wir gezeigt, daß (CP) äquivalent zu dem quadratischen Problem (P), bzw. durch entsprechende Wahl der Mengen $Z(i)$ äquivalent zu einem Problem (P_m) ist. Ferner gilt, daß eine Clique C der Kardinalität n in G_C für jede maximale stabile Menge genau einen Knoten enthält, d.h. $|C \cap S_i| = 1$ für alle $i = 1, \ldots, n$. Unser Ziel wird es zunächst sein, (CP) als ganzzahliges Programm mit einer linearen Zielfunktion und linearen Nebenbedingungen darzustellen.
Um die Darstellung im folgenden zu vereinfachen, betrachten wir von nun an eine Clique als Kantenmenge.

Zunächst führen wir für jede Kante $[ik, jl] \in E_C$ eine Variable ein.

Variablen

Die Variablen lassen sich wie folgt interpretieren.

$$y_{ik,jl} = \begin{cases} 1, & \text{wenn die Kante } [ik, jl] \text{ in der Lösung ist,} \\ 0, & \text{anderenfalls.} \end{cases}$$

Nebenbedingungen

(1) Zwischen je zwei maximalen stabilen Mengen muß eine Kante in der Lösung sein. Diese Nebenbedingung kann formal folgendermaßen beschrieben werden.

$\sum_{k\in Z(i)} \sum_{l\in Z(j)} y_{ik,jl} = 1$ für alle $i = 1,\ldots,n,\ j = i+1,\ldots,n$.

(2) Wenn eine Kante $[ik,(i+1)l]$ bzw. $[nk,1l]$ in der Clique ist, dann müssen $(n-2)$ weitere Kanten in der Clique sein, die inzident zu ik bzw. nk sind. Umgekehrt dürfen keine Kanten in der Lösung inzident zu einem Knoten ik bzw. nk sein, wenn $\sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0$ bzw. $\sum_{l\in Z(1)} y_{nk,1l} = 0$ ist. Dies läßt sich folgendermaßen ausdrücken:

$\sum_{j\notin\{i,i+1\}} \sum_{l\in Z(j)} y_{ik,jl} - (n-2)\sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0$
für alle $i = 1,\ldots,n-1,\ k \in Z(i)$.

$\sum_{j\notin\{1,n\}} \sum_{l\in Z(j)} y_{nk,jl} - (n-2)\sum_{l\in Z(1)} y_{nk,1l} = 0$
für alle $k \in Z(n)$.

(3) Alle Variablen müssen ganzzahlig sein.

$y_{ik,jl} \in \{0,1\}$ für alle $[ik,jl] \in E_C$.

Zielfunktion

Minimiere das Gesamtgewicht der Kanten in der Lösung.

$$\min \sum_{[ik,jl]\in E_C} w_C([ik,jl])y_{ik,jl}.$$

(9.1) Lemma:
Obiges Programm stellt eine ganzzahlige Modellierung des Problems (CP) dar.

Beweis: Gegeben sei eine Clique der Kardinalität n in G_C, welche die Lösung einer Instanz von (CP) ist. Sei $y \in \mathbb{R}^{nm\times nm}$ definiert gemäß:

$$y_{ik,jl} = \begin{cases} 1, & \text{wenn die Kante } [ik,jl] \text{ in der Lösung ist,} \\ 0, & \text{anderenfalls.} \end{cases}$$

Dann ist offensichtlich, daß y die Bedingungen (1) bis (3) in obigem Modell erfüllt.

Sei also umgekehrt $y \in \mathbb{R}^{nm\times nm}$ ein Vektor, welcher die Bedingungen (1) bis (3) erfüllt, und bezeichne $C = \{[ik,jl] \in E_C \mid y_{ik,jl} = 1\}$.

(i) Für jedes Paar maximaler stabilen Mengen S_i und S_j, $i \neq j$ existiert genau eine Kante $[ik,jl]$ in C (Bedingung (1)). Die Bedingungen des Typs (1) garantieren ferner, daß insgesamt $\frac{n(n-1)}{2}$ Kanten in der Lösung sind, d.h. $|C| = \frac{n(n-1)}{2}$. Es bleibt also zu zeigen, daß alle Kanten in C, die inzident zu einem Knoten aus S_i sind, inzident zu demselben Knoten aus S_i sind.

(ii) Sei dazu zunächst $1 \leq i \leq n-1$, und sei $[ik,(i+1)l] \in C$ (nach obigen Ausführungen existiert eine solche Kante). Dann stellen die entsprechenden Bedingungen des Typs (2) $(\sum_{j \notin \{i,i+1\}} \sum_{l \in Z(j)} y_{ik,jl} - (n-2) \sum_{l \in Z(i+1)} y_{ik,(i+1)l} = 0)$ sicher, daß es noch $n-2$ weitere Kanten in der Lösung gibt, die inzident zu ik sind.
Da insbesondere die Ungleichungen

$$\sum_{l \in Z(j)} y_{ik,jl} \leq 1$$

für alle $i,j \in \{1,\ldots,n\}$, $i \neq j$, $k \in Z(i)$ gelten, gibt es für jede maximale stabile Menge S_j, $j \neq i$ genau eine Kante in der Lösung, die inzident zu ik ist.

(iii) Eine analoge Argumentation trifft für $i = n$ zu.

(i), (ii) und (iii) implizieren, daß die Kantenmenge C eine Clique der Kardinalität n definiert. □

Obigem Programm werden wir aus Gründen, die im folgenden klar werden, zusätzliche, gültige Ungleichungen hinzufügen. Dies definiert eine redundante, lineare, ganzzahlige Formulieung für (CP). Anschließend werden wir letzteres Programm verwenden, um einen Lagrange-Relaxierungsansatz zu entwickeln.

Das Verfahren zur **Lagrange-Relaxierung** geht auf Held und Karp zurück [HK70], [HK71] und läßt sich folgerndermaßen skizzieren (vergleiche [S89]):
Gegeben Matrizen $A_1 \in \mathbb{R}^{m_1 \times n}$, $A_2 \in \mathbb{R}^{m_2 \times n}$ und ein Vektor $c \in \mathbb{R}^n$. Nehmen wir an, wir wollen das folgende Problem lösen:

$$\min\{c^T x \mid A_1 x = b_1,\ A_2 x = b_2,\ x \text{ ganzzahlig}\}.$$

Dann läßt sich eine untere Schranke angeben:

$$\min\{c^T x \mid A_1 x = b_1,\ A_2 x = b_2,\ x \text{ ganzzahlig}\} \geq$$
$$\max\{f(\gamma) \mid \gamma \in \mathbb{R}^{m_1}\}, \text{ wobei}$$
$$f(\gamma) = \min\{c^T x + \gamma^T (A_1 x - b_1) \mid A_2 x = b_2,\ x \text{ ganzzahlig}\}.$$

Die Funktion f wird als **Lagrange-Funktion** bezeichnet. Sie besitzt die interessante Eigenschaft, konkav zu sein (vergleiche [S89]). Die Zahl $\max\{f(\gamma) \mid \gamma \in \mathbb{R}^{m_1}\}$ wird als **Lagrange-Wert** bezeichnet, und die Komponenten einer Optimallösung γ^* heißen **Lagrange-Multiplikatoren.**
Zur Lösung des Problems $\max\{f(\gamma) \mid \gamma \in \mathbb{R}^{m_1}\}$ schlugen Held und Karp folgendes Verfahren vor:
Sei $j = 0$. Man starte mit einem Vektor γ_j und berechne $f(\gamma_j)$. Setze $j = j+1$ und bestimme einen Vektor $\gamma_j = \gamma_{j-1} + t_j u_j$, wobei u_j ein **Subgradient** für f im Punkt γ_{j-1} ist, d.h. $f(\gamma_{j-1} + z) - f(\gamma_{j-1}) \leq z u_j$ für alle $z \in \mathbb{R}^{m_1}$. Die Zahlen $t_j \geq 0$ werden **Schrittlängen** genannt. Basierend auf einem Resultat von [P67] gilt, daß für eine Folge von Schrittlängen $t_j \geq 0$, $j = 0,\ldots,\infty$, welche den Bedingungen $\lim_{j \to \infty} t_j = 0$ und $\sum_{j=0}^{\infty} t_j = \infty$ genügt, die Folge von Werten $f(\gamma_j)$ gegen das Maximum der Funktion f konvergiert, sofern ein solches existiert.

Falls man also in der Lage ist, die Funktion f an jeder Stelle γ effizient (in polynomialer Laufzeit) auswerten zu können, so hat man eine praktikable Methode gefunden, um eine untere Schranke für das Ausgangsproblem zu berechnen.
Interessant für Lagrange-Relaxierungen sind insbesondere solche Probleme, für die gilt:

$$\{\min\{c^T x + \gamma^T(A_1 x - b_1) \mid A_2 x = b_2,\ x \text{ ganzzahlig}\} =$$
$$\{\min\{c^T x + \gamma^T(A_1 x - b_1) \mid A_2 x = b_2\}.$$

In diesem Fall folgt aus der Dualität linearer Programme, daß der Lagrange-Wert dem Optimalwert der linearen Relaxierung des Ausgangsproblems entspricht [S89].
Diese Vorgehensweise wollen wir im folgenden auf das zu Beginn des Kapitels definierte Cliquenproblem anwenden. Dazu betrachten wir das in (9.2) definierte Optimierungsproblem.

$$\min \sum_{[ik,jl]\in E_C} w_C([ik,jl]) y_{ik,jl}$$

mit

(9.2)

$$(1) \quad \sum_{k\in Z(i)} \sum_{l\in Z(j)} y_{ik,jl} = 1 \text{ für alle } i = 1,\dots,n,\ j = i+1,\dots,n,$$

$$(2i) \quad \sum_{j\notin\{i,i+1\}} \sum_{l\in Z(j)} y_{ik,jl} - (n-2) \sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0$$
$$\text{für alle } i = 1,\dots,n-1,\ k \in Z(i),$$

$$(2ii) \quad \sum_{j\notin\{1,n\}} \sum_{l\in Z(j)} y_{nk,jl} - (n-2) \sum_{l\in Z(1)} y_{nk,1l} = 0$$
$$\text{für alle } k \in Z(n),$$

$$(3) \quad \sum_{j=i+2}^{n} \sum_{l\in Z(j)} y_{ik,jl} - (n-i-1) \sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0$$
$$\text{für alle } i = 1,\dots,n-2,\ k \in Z(i),$$

$$(4) \quad y_{ik,jl} \in \{0,1\}, \text{ für alle } [ik,jl] \in E_C.$$

Die Ungleichungen des Typs (3) stellen in gewisser Weise eine "Abschwächung" der Ungleichungen des Typs (2) dar. Aus den folgenden Erläuterungen wird hervorgehen, warum wir diese Ungleichungen in dem linearen Programm berücksichtigen.

(9.3) Lemma:
Das System (9.2) stellt eine (redundante) ganzzahlige Modellierung des Problems (CP) dar.

Beweis.
Es ist offensichtlich, daß jede Lösung von (CP) die Bedingungen des Typs (3) erfüllt. Da umgekehrt nach Lemma (9.1) jeder Vektor, der die Bedingungen (1),(2) und (4) erfüllt,

eine Lösung der entsprechenden Instanz von (CP) definiert, gilt dies auch für einen Vektor, der zusätzlich die Bedingungen des Typs (3) erfüllt.

□

Ausgehend von diesem Programm definieren wir eine Lagrange-Funktion f. Um ihren Funktionswert an einer gegebenen Stelle zu berechnen, muß ein Optimierungsproblem gelöst werden. Dessen Zielfunktion setzt sich aus der ursprünglichen Zielfunktion und den Ungleichungen des Typs (2) zusammen. Die Nebenbedingungen sind durch die Ungleichungen (1), (3) und (4) bestimmt.

$$
\begin{aligned}
f(\gamma) = \min & \sum_{[ik,jl]\in E_C} w_C([ik,jl])y_{ik,jl} + \\
& \sum_{i=1}^{n-1} \sum_{k\in Z(i)} \gamma_{ik}\Big(\sum_{j\notin\{i,i+1\}} \sum_{l\in Z(j)} y_{ik,jl} - (n-2) \sum_{l\in Z(i+1)} y_{ik,(i+1)l}\Big) + \\
& \sum_{k\in Z(n)} \gamma_{nk}\Big(\sum_{j\notin\{1,n\}} \sum_{l\in Z(j)} y_{nk,jl} - (n-2) \sum_{l\in Z(1)} y_{nk,1l}\Big)
\end{aligned}
$$

mit

(9.4)

$$
\begin{aligned}
&(1) \quad \sum_{k\in Z(i)} \sum_{l\in Z(j)} y_{ik,jl} = 1 \text{ für alle } i = 1,\dots,n,\ j = i+1,\dots,n,\\
&(3) \quad \sum_{j=i+2}^{n} \sum_{l\in Z(j)} y_{ik,jl} - (n-i-1) \sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0 \\
&\qquad \text{für alle } i = 1,\dots,n-2,\ k\in Z(i),\\
&(4) \quad y_{ik,jl} \in \{0,1\}, \text{ für alle } [ik,jl]\in E_C.
\end{aligned}
$$

Um die Funktion f an einer beliebigen Stelle berechnen zu können, muß das folgende Problem (9.5) für alle Vektoren $w'_C \in \mathbb{R}^{nm\cdot nm}$ gelöst werden können:

(9.5) $\min\{\sum_{e\in E_C} w'_C y_e \mid$ y erfüllt (1),(3) und (4)$\}$.

Analysiert man die Bedingungen (1), (3) und (4) etwas genauer, so stellt man fest, daß sich diese in Systeme von Nebenbedingungen zerlegen lassen, die unabhängig voneinander sind. Konkret läßt sich für alle $i = 1,\dots,n-2$ das folgende System von Nebenbedingungen angeben:

(9.6)

$$
\begin{aligned}
&(a) \quad \sum_{k\in Z(i)} \sum_{l\in Z(j)} y_{ik,jl} = 1 \text{ für alle } j = i+1,\dots,n,\\
&(b) \quad \sum_{j=i+2}^{n} \sum_{l\in Z(j)} y_{ik,jl} - (n-i-1) \sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0 \\
&\qquad \text{für alle } k\in Z(i),\\
&(c) \quad y_{ik,jl} \in \{0,1\}, \text{ für alle } [ik,jl]\in E_C.
\end{aligned}
$$

Dadurch, daß in (9.4) die Ungleichungen des Typs (2) nicht mehr als Restriktionen bestehen, existiert keine "Rückkopplung" zwischen den Bedingungen für $i \in \{1,\ldots,n-2\}$ und $j \in \{1,\ldots,n-2\}$, sofern i und j verschieden sind. Die Nebenbedingungsmatrix hat demnach Blockgestalt (vergleiche Abbildung (9.7)).

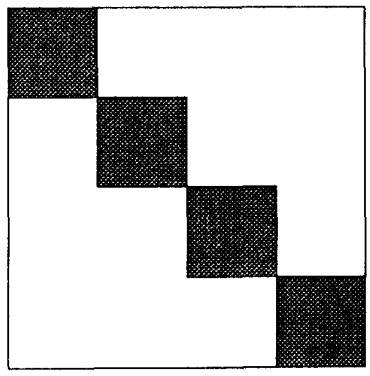


Abbildung 9.7

Zur Lösung von (9.5) bietet sich der folgende polynomiale Algorithmus an:

(9.8) Algorithmus

Input: $G_C = (V_C, E_C, w'_C)$.

Output: Eine Lösung von (9.5).

(1) Sei $i = 1$.

(2) Solange $i < n$ ist, führe die folgenden Anweisungen durch:

(2.1) Für alle $k \in Z(i)$ führe folgende Anweisungen durch:

(2.1.1) Setze $E_k = \emptyset$ und $g_k = 0$.

(2.1.2) Für alle $j > i$ führe folgende Anweisungen aus.

(2.1.2.1) sei $w_C(ik, jl_0) = \min\{w_C(ik, jl),\ l \in Z(j)\}$.

(2.1.2.1) Setze $g_k = g_k + w_C(ik, jl_0)$ und $E_k = E_k \cup \{[ik, jl_0]\}$.

(2.2) Berechne $g_{k_0} = \min\{g_k,\ k \in Z(i)\}$ und setze $y_e = 1$ für alle $e \in E_{k_0}$. Inkrementiere i.

(3) Gib y aus.

Es ist zunächst offensichtlich, daß dieses Verfahren polynomiale Laufzeit in n und m hat. Ferner läßt sich sehr einfach einsehen, daß Algorithmus (9.8) auch eine Optimallösung für die lineare Relaxierung von (9.5) liefert.

Somit befinden wir uns in der Situation, die Lagrange-Funktion für alle Zielfunktionen in polynomialer Laufzeit auswerten zu können. Aus obigen Ausführungen wissen wir dann, daß der maximale Zielfunktionswert von f dem Wert der linearen Relaxierung des Programms (9.2) entspricht. Diesen Wert hätten wir explizit mit einem linearen Optimierungscode für unsere Problembeispiele nicht bestimmen können. Denn das Ausgangsproblem (9.4) hat m^2n^2 viele Variablen. Für das Beispiel soc_2 ergibt sich damit bereits für $m = 2$ eine Anzahl von 4177936 Variablen. Für $m = 4$ erhöht sich diese Zahl auf 16711744.
Diesen Lagrange-Relaxierungsansatz haben wir implementiert. Für die Tests verwenden wir die Schaltungen soc_1, soc_2 und $mini$. Letztere besteht aus 69 logischen Zellen, 33 Randzellen und 102 Netzen.
Die Zielfunktionswerte $w_C([ik,jl])$, $[ik,jl] \in E_C$ für die Beispiele bestimmen sich folgendermaßen:
Zunächst generieren wir für jede der drei Schaltungen Instanzen von (P_2^+), welche unter Verwendung des Dekompositionsverfahrens dec_2 in der ersten Iteration entstehen (vergleiche hierzu Kapitel 5). Die Koeffizienten $(q_{ik,jl})$ der zugehörigen Matrix definieren die Kantengewichte $w_C([ik,jl])$. Für diese Problembeispiele berechnen wir mit Hilfe des oben dargestellten Lagrange-Relaxierungsansatzes eine untere Schranke. Als obere Schranke wählen wir den Zielfunktionswert der quadratischen 0/1-Lösung, welche durch Anwendung von Heuristik 2a und den beiden Verbesserungsalgorithmen (vergleiche Kapitel 6) zustandekommt.
In den Zeilen 2, 3 und 4 von Tabelle 9.9 sind die Ergebnisse für die drei Schaltungen dargestellt. In Spalte 2 ist die obere Schranke eingetragen. Die Werte in Spalte 3 entsprechen der unteren Schranke. Das prozentuale Verhältnis aus "oberer Schranke - unterer Schranke" dividiert durch die obere Schranke ist in Spalte 4 dargestellt. Die Laufzeit, welche erforderlich ist, um die jeweilige untere Schranke zu erhalten, wird in Spalte 4 in der Einheit "Minuten: Sekunden"angegeben. Die Tests wurden auf einer Sparc-Station durchgeführt.

	obere Schranke	untere Schranke	Differenz	CPU-Zeit
$mini$	4155.67	3027.01	27%	41:07
soc_1	73494.25	41633.94	44%	422:34
soc_2	76434.15	22852.14	71%	1060:35

Tabelle 9.9

Zusammenfassend läßt sich bemerken, daß die Ergebnisse keineswegs zufriedenstellend sind, da in den beiden größeren Beispielen (soc_1 und soc_2) die untere und die obere Schranke um mindestens 44% auseinanderliegen.
Aus diesem Grund haben wir versucht, durch Hinzunahme weiterer Ungleichungen den Lagrange-Relaxierungsansatz zu verbessern.

Betrachten wir diesbezüglich die Ungleichungen:

$$(9.10) \quad \sum_{r\in Z(v)} y_{vr,ik} + \sum_{r\in Z(v)} y_{vr,jl} - y_{ikjl} \leq 1$$

für alle paarweise verschiedenen $i,j,v \in N,\ k \in Z(i),\ l \in Z(j)$.

Eine Ungleichung aus dieser Klasse läßt sich folgendermaßen interpretieren:
Wenn zwei Kanten $[vr,ik]$ und $[vr,jl]$ zu der Clique gehören, so muß auch die Kante $[jl,ik]$ in der Clique sein (Dreiecksungleichung).
Offensichtlich erfüllt jede Lösung von (CP) diese Ungleichungen. Deshalb können wir nach Lemma (9.3) die Ungleichungen (9.10) dem System (9.2) hinzufügen. Somit erhalten wir das System (9.11)

$$\min \sum_{[ik,jl]\in E_C} w_C([ik,jl])y_{ik,jl}$$

mit

(9.11)

(1) $\sum_{k\in Z(i)} \sum_{l\in Z(j)} y_{ik,jl} = 1$ für alle $i = 1,\ldots,n,\ j = i+1,\ldots,n,$

(2i) $\sum_{j\notin\{i,i+1\}} \sum_{l\in Z(j)} y_{ik,jl} - (n-2) \sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0$

für alle $i = 1,\ldots,n-1,\ k \in Z(i)$,

(2ii) $\sum_{j\notin\{1,n\}} \sum_{l\in Z(j)} y_{nk,jl} - (n-2) \sum_{l\in Z(1)} y_{nk,1l} = 0$

für alle $k \in Z(n)$,

(3) $\sum_{j=i+2}^{n} \sum_{l\in Z(j)} y_{ik,jl} - (n-i-1) \sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0$

für alle $i = 1,\ldots,n-2,\ k \in Z(i)$,

(4) $y_{ik,jl} \in \{0,1\}$, für alle $[ik,jl] \in E_C$,

(5) $\sum_{r\in Z(v)} y_{vr,ik} + \sum_{r\in Z(v)} y_{vr,jl} - y_{ikjl} \leq 1$ für alle

paarweise verschiedenen $i,j,v \in N,\ k \in Z(i),\ l \in Z(j)$.

Ausgehend von diesem Programm, definieren wir in Analogie zu oben eine Lagrange-Funktion f' gemäß:

$$f'(\gamma,\mu) = \min \sum_{[ik,jl]\in E_C} w_C([ik,jl])y_{ik,jl} +$$

$$\sum_{i=1}^{n-1} \sum_{k\in Z(i)} \gamma(ik)\Big(\sum_{j\notin\{i,i+1\}} \sum_{l\in Z(j)} y_{ik,jl} - (n-2) \sum_{l\in Z(i+1)} y_{ik,(i+1)l}\Big) +$$

$$\sum_{k\in Z(n)} \gamma(nk)(\sum_{j\notin\{1,n\}}\sum_{l\in Z(j)} y_{nk,jl} - (n-2)\sum_{l\in Z(1)} y_{nk,1l}) +$$
$$\sum_{i\in N}\sum_{j\in N\setminus\{i\}}\sum_{v\in N\setminus\{i,j\}}\sum_{k\in Z(i)}\sum_{l\in Z(j)} \mu(i,j,v,k,l)(\sum_{r\in Z(v)} y_{vr,ik} +$$
$$\sum_{r\in Z(v)} y_{vr,jl} - y_{ikjl} - 1)$$

mit

(9.12) (1) $\sum_{k\in Z(i)}\sum_{l\in Z(j)} y_{ik,jl} = 1$ für alle $i = 1,\ldots,n,\ j = i+1,\ldots,n,$

(3) $\sum_{j=i+2}^{n}\sum_{l\in Z(j)} y_{ik,jl} - (n-i-1)\sum_{l\in Z(i+1)} y_{ik,(i+1)l} = 0$
für alle $i = 1,\ldots,n-2,\ k \in Z(i),$

(4) $y_{ik,jl} \in \{0,1\}$, für alle $[ik,jl] \in E_C$,
$\gamma(ik) \in \mathbb{R}$ für alle $i \in N,\ k \in Z(i)$,
$\mu(i,j,v,k,l) \geq 0$ für alle $k \in Z(i),\ l \in Z(j)$
und für alle paarweise verschiedenen $i,j,v \in N$.

Da die Anzahl von Ungleichungen des Typs (9.10) $O(n^3m^2)$ ist, müßten wir insbesondere diese $O(n^3m^2)$ vielen Ungleichungen auswerten, um den Funktionswert von f' an einer Stelle (γ,μ) zu bestimmen. Dies würde bei einer Größenordnung von $n = 1000$ zu immensen Laufzeiten führen. Aus diesem Grund haben wir uns entschieden, nur eine Teilklasse von Ungleichungen zu betrachten:

$$(9.13)\quad \sum_{r\in Z(i-2)} y_{(i-2)r,ik} + \sum_{r\in Z(i-2)} y_{(i-2)r,(i-1)l} - y_{ik(i-1)l} \leq 1$$
für alle $i = 3,\ldots,n,\ k \in Z(i),\ l \in Z(i-1).$

Wir definieren dann analog eine Lagrange-Funktion $f''(\gamma,\mu')$, wobei $\mu' \geq 0$ den Vektor der Multiplikatoren für die Ungleichungen des Types (9.13) darstellt.

$$f''(\gamma,\mu') = \min \sum_{e\in E_C} w_C(e)y_e +$$
$$\sum_{i=1}^{n-1}\sum_{k\in Z(i)} \gamma(ik)(\sum_{j\notin\{i,i+1\}}\sum_{l\in Z(j)} y_{ik,jl} - (n-2)\sum_{l\in Z(i+1)} y_{ik,(i+1)l}) +$$
$$\sum_{k\in Z(n)} \gamma(nk)(\sum_{j\notin\{1,n\}}\sum_{l\in Z(j)} y_{nk,jl} - (n-2)\sum_{l\in Z(1)} y_{nk,1l}) +$$
$$\sum_{i=3}^{n}\sum_{k\in Z(i)}\sum_{l\in Z(i-1)} \mu'(i,k,l)(\sum_{r\in Z(v)} y_{(i-2)r,ik} +$$

$$\sum_{r \in Z(i-2)} y_{(i-2)r,(i-1)l} - y_{ik(i-1)l} - 1)$$

mit

$$\begin{aligned} (9.14)\quad & (1),(3),(4),(5), \\ & \gamma(ik) \in \mathbb{R} \text{ für alle } i \in N,\ k \in Z(i), \\ & \mu'(i,k,l) \geq 0 \text{ für alle } i = 3,\ldots,n,\ k \in Z(i),\ l \in Z(i-1). \end{aligned}$$

Da das Problem

$$\min\{w'_C(e)y_e \mid y \text{ erfüllt } (1),(3),(4), \text{ und } (5)\}$$

mit Hilfe von Algorithmus (9.8) für alle Zielfunktionen w'_C gelöst werden kann, läßt sich analog zu oben ein Lagrange-Relaxierungsansatz mit der Lagrange-Funktion f'' durchführen. Diesen haben wir implementiert und wiederum bezüglich der Schaltungen *mini*, soc_1 und soc_2 getestet. Die Zielfunktionswerte $w_C(e)$, $e \in E_C$ sind analog zu oben bestimmt (vergleiche die Erklärungen zu Tabelle 9.9).

	obere Schranke	untere Schranke	Differenz	CPU-Zeit
mini	4155.67	3797.66	9%	46:31
soc_1	73494.25	66004.15	10%	907:31
soc_2	76434.15	71592.36	6%	1728:01

Tabelle 9.15

Die Ergebnisse zeigen eine deutliche Verbesserung gegenüber den Resultaten aus Tabelle 9.10. Insbesondere erreichen wir in allen Beispielen eine Abweichung zwischen oberer und unterer Schranke von weniger als 10%. Für das Beispiel soc_1 liegt der Wert sogar nur bei 6%.

Die Zielfunktionswerte $w_C([ik,jl])$, $[ik,jl] \in E_C$ in obigen Beispielen entsprechen den Matrixkoeffizienten $c_{ij}d_{ik,jl} + \lambda o_{ik,jl}$ der Instanzen von (P_2^+), die unter Verwendung des Dekompositionsverfahrens dec_2 in der ersten Iteration entstehen (vergleiche Kapitel 3 und 5).

Uns schien es interessant, eine analoge Testserie durchzuführen, wobei diesmal der Bestrafungsparameter λ verdoppelt wird. Dadurch wird – so vermuten wir – in der Lösung, die die obere Schranke definiert, eine bessere Verteilung der Zellen auf die repräsentativen Basiszellen erreicht, was zu einer erhöhten abgeschätzten Verdrahtungslänge führt. Daneben werden aufgrund des größeren Bestrafungsparameters die gewichteten Überlappungskosten insgesamt steigen. Die Ungleichungen des Typs (9.13) reichen in diesem Fall wahrscheinlich nicht aus, um eine sehr kleine Abweichung zwischen oberer und unterer Schranke zu

erhalten. In Tabelle 9.16 sind die Ergebnisse dieser Testreihe dargestellt.

	obere Schranke	untere Schranke	Differenz	CPU-Zeit
mini	6424.80	4940.63	23%	151:52
soc_1	90062.23	69699.12	23%	1929:41
soc_2	109670.31	80157.85	27%	4320:57

Tabelle 9.16

Die Resultate bestätigen unsere Hypothese. Insgesamt hat sich die Differenz zwischen oberer und unterer Schranke deutlich verschlechtert. Die Abweichung beträgt in allen Fällen jedoch weniger als 30%, ein Wert, der unserer Meinung nach noch immer akzeptabel ist, da die Anzahl von Variablen in den linearen Programmen im Bereich mehrerer Millionen liegt.

In einer abschließenden Testreihe generieren wir Instanzen von (P_4^+), welche unter Verwendung des Dekompositionsverfahrens dec_4 in der ersten Iteration entstehen. Die Koeffizienten der zugehörigen Matrix definieren die Kantengewichte $w_C(e)$, $e \in E_C$ in diesem Fall. Tabelle 9.17 zeigt die diesbezüglichen Ergebnisse.

	obere Schranke	untere Schranke	Differenz	CPU-Zeit
mini	3553.73	2248.29	37%	7:59
soc_1	104269.42	43707.26	58%	3165:19
soc_2	160058.20	48275.37	70%	3970:17

Tabelle 9.17

Es zeigt sich, daß die prozentuale Abweichung zwischen oberer und unterer Schranke in den beiden größeren Beispielen über 50% liegt und damit wesentlich größer ist als im Fall $m = 2$. Dies läßt den Schluß zu, daß die Ungleichungen des Typs (9.13) offensichtlich noch nicht ausreichen, um eine akzeptable untere Schranke zu erhalten.

Zusammenfassend läßt sich sagen, daß wir unter Verwendung der Lagrange-Funktion f'' im Fall $m = 2$ gute bis sehr gute Ergebnisse erhalten. Bei einer Erhöhung von m zeigt sich, daß die Differenz zwischen oberer und unterer Schranke wesentlich größer wird. Tendentiell lassen die Ergebnisse jedoch den Schluß zu, daß die Berücksichtigung von zusätzlichen Ungleichungen in dem Lagrange-Ansatz zu einer enormen Verbesserungen der unteren Schranke führt.

Teil III

Das Clusteringproblem

In dem dritten Teil dieser Arbeit beschäftigen wir uns mit dem Clustering-Problem. Letzteres hat zahlreiche Anwendungen und nicht zuletzt deshalb große Beachtung in der Literatur gefunden. Unsere Motivation, dieses Problem anzugehen, ist die, daß wir eine Lösung des Clusteringproblems dazu verwenden wollen, um die Größenordnung von Plazierungsinstanzen zu reduzieren. Im Detail gliedert sich Teil III folgendermaßen:
In Kapitel 10 werden wir das Clusteringproblem definieren und den Zusammenhang zum VLSI-Design herstellen. Ferner werden wir verschiedene Modellierungen vorstellen. In Kapitel 11 geben wir einige Heuristiken an, die in unseren Plazierungsalgorithmus integriert wurden. In den folgenden Kapiteln soll das mit einem Modell aus Kapitel 10 assoziierte Polyeder untersucht werden. Eine formale Definition wird in Kapitel 12 erfolgen. Die grundlegenden Facetten dieses Polyeders werden in Kapitel 13 dargestellt. In Kapitel 14 werden wir dann das Problem des Liftens behandeln und zwei allgemeine Eigenschaften von Facetten aufzeigen. Zum Abschluß des dritten Teils werden mehrere Klassen von nicht-trivialen Facetten für das in Kapitel 10 definierte Polyeder vorgestellt werden.

Kapitel 10 Das Clusteringproblem und seine Varianten

In den Teilen I und II dieser Arbeit haben wir uns ausführlich mit dem Plazierungsproblem und dessen Lösbarkeit beschäftigt. Dabei hat sich insbesondere gezeigt, daß eine Vielzahl von Methoden und "Tricks" entwickelt werden mußte, um die Größenordnung praktischer Problembeispiele zu bewältigen. Eine weitere Möglichkeit, die Größenordnung von Instanzen des Plazierungsproblems zu reduzieren, wird uns zu dem sogenannten Clusteringproblem führen. In den Abschnitten 10.1 bis 10.4 werden wir die genaue Problemstellung sowie eine Modellierung des Clusteringproblems und eines Spezialfalls davon präsentieren.

10.1 Motivation

In den vorhergehenden Kapiteln haben wir uns mit dem Plazierungsproblem beschäftigt. Wir haben dieses als quadratisches 0/1 Problem modelliert, das Modell komplexitätstheoretisch analysiert und Methoden zu seiner Lösung behandelt. Dabei hat sich insbesondere gezeigt, daß eine Vielzahl von "Tricks" entwickelt werden mußte, um praktische Problembeispiele lösen zu können. Praktische Größenordnungen beginnen bei etwa 1000 Zellen, 1000 Netzen und 9000 Basiszellen. Instanzen des quadratischen Modells (P) haben deshalb Millionen von Matrixeinträgen. Aus diesem Grund haben wir ein Dekompositionsverfahren vorgestellt, welches pro Iteration die Anzahl von Basiszellen beschränkt, denen eine Zelle zugewiesen werden kann.
Eine weitere Möglichkeit, die Größenordnung von Instanzen des Plazierungsproblems zu reduzieren, beruht darauf, die Anzahl von Zellen zu verringern. Die grundlegende Idee besteht darin, mehrere Zellen zu einer sogenannten Superzelle zusammmenzufassen. Anschließend wird das Plazierungsproblem für die Liste der Superzellen gelöst. Dadurch ist jeder Superzelle ein Einbauplatz zugewiesen. In dem folgenden Schritt wird versucht, ausgehend von der Plazierung der Superzellen eine zulässige Plazierung der Zellen zu finden. Die wesentliche Fragestellung bei diesem Ansatz bezieht sich darauf, welche Zellen zu einer Superzelle zusammengefaßt werden sollen. Da es unser Ziel ist, eine Plazierung mit geringer (abgeschätzter) Verdrahtungslänge zu erhalten, erscheint es naheliegend, die Zellen, die stark miteinander verbunden sind, zu derselben Superzelle zusammenfassen. Anders ausgedrückt wollen wir die Anzahl von Verbindungen zwischen Zellen, die verschiedenen Superzellen zugewiesen werden, minimieren. Dieses Problem läßt sich in einem Graphen formulieren und führt zu dem sogenannten Mehrfachschnitt-Problem bzw. Varianten davon.
Bevor wir den Zusammenhang präzisieren, wollen wir zunächst die notwendigen Begriffe und Bezeichnungsweisen einführen (vergleiche hierzu auch [DGL91], [DGL92]).

Definition:

(i) Gegeben sei eine Instanz des Plazierungsproblems. Diese ist unter anderem durch eine Menge von Zellen und die Affinitätskoeffizienten charakterisiert. Sei $N = \{1, \ldots, n\}$

die Menge von Zellen und seien $c_{ij} \geq 0$ die in Kapitel 3 eingeführten Affinitätskoeffizienten zwischen Zellen i und j, $i,j \in N$, $i \neq j$. Es bezeichne ferner f die Liste der Zellen-Flächen. Die Fläche von Zelle $i \in N$ werden wir stets mit f_i abkürzen.

Unter diesen Voraussetzungen bezeichnen wir den im folgenden definierten Graphen $G_{\mathcal{A}} = (N, E)$ als **Affinitätsgraphen.**
Eine Zelle mit Nummer $i \in N$ wird durch einen Knoten gleicher Nummer repräsentiert. Zwischen zwei Knoten i und j, $i,j \in N$, $i \neq j$ führen wir eine Kante ein, wenn $c_{ij} > 0$ ist. Das Gewicht der Kante mit den beiden Endknoten i und j ist durch den Affinitätskoeffizienten c_{ij} gegeben.

(ii) Sei S eine Menge. Dann heißt $S_1, \ldots, S_r$ eine **Partition** von S, wenn gilt:
$S_i \neq \emptyset$ für alle $i = 1, \ldots, r$,
$S_i \cap S_j = \emptyset$ für alle $i,j = 1, \ldots, r$, $i \neq j$,
$S = \bigcup_{i=1}^{r} S_i$.
Die Mengen S_i werden Elemente der Partition genannt. Gegeben sei ein Graph $G = (V, E)$, und sei $S_1, \ldots, S_r$ eine Partition von V.
Dann wird $\delta(S_1, \ldots, S_r) = \{uv \in E \mid$ es existieren $i \neq j$ mit $u \in S_i, v \in S_j\}$ als **r-Schnitt** bezeichnet. Falls die Anzahl r nicht spezifiziert ist, so spricht man von einem **Mehrfachschnitt.** Zur Vereinfachung der Notation wird ein h-Schnitt mit $h \leq r$ bzw. $h \geq r$ auch als **$\leq r$-Schnitt** bzw. **$\geq r$-Schnitt** bezeichnet. Ein Schnitt in G ist in obiger Notation ein ≤ 2-Schnitt. Dafür verwenden wir auch das Symbol $\delta(S) = \{ij \in E \mid i \in S, j \in V \setminus S\}$.
Gegeben sei ein Graph $G = (V, E)$ mit Kantengewichten $w_e \in \mathbb{Q}$, $e \in E$. Das Maximalschnitt- bzw. Minimalschnitt-Problem besteht darin, einen Schnitt maximalen bzw minimalen Gewichts in G zu finden. Eine Lösung des **Mehrfachschnitt-Problems** ist ein Mehrfachschnitt minimalen Gewichts. Analog läßt sich das **r-Schnitt-Problem**, **$\leq r$-Schnitt-Problem** bzw. **$\geq r$-Schnitt-Problem** formulieren: Gesucht ist ein r-Schnitt, $\leq r$-Schnitt bzw. $\geq r$-Schnitt minimalen Gewichts in G.

Betrachten wir an dieser Stelle noch einmal die eingangs motivierte Problemstellung: Gegeben sei ein Affinitätsgraph. Gesucht ist eine Partition der Knoten, so daß die Summe der Gewichte der Kanten im Schnitt zwischen den verschiedenen Elementen der Partition minimal ist. Da die Kantengewichte im Affinitätsgraphen alle positiv sind, ist die Optimallösung dieses Problems offensichtlich die Partiton, deren einziges Element die Knotenmenge enthält. Um dies zu vermeiden, führen wir Knotengewichte und eine obere Schranke für die Summe der Knotengewichte eines Elements der Partition ein.
An dieser Stelle können wir nun das Problem formal definieren:

10.2 Problemstellung

Aus technischen Gründen wollen wir an dieser Stelle noch eine Notation einführen: Sei $G = (V, E)$ ein Graph.

(i) Unter einem **Cluster** verstehen wir entweder ein Element einer Partition oder die leere Menge.

(ii) Seien $S_1, \ldots, S_r$ Cluster mit $\bigcup_{i=1}^{r} S_i = V$. Dann definieren wir das Symbol

$$\delta(S_1, \ldots, S_r) := \delta(S_{i_1}, \ldots, S_{i_h}),$$

wobei $S_{i_1}, \ldots, S_{i_h}$ die nichtleeren Cluster sind ($h \leq r$, $\{i_1, \ldots, i_h\} \subseteq \{1, \ldots, r\}$).

Im folgenden definieren wir das Clusteringproblem, Kardinalitäts-Clusteringproblem, r-Clusteringproblem und das r-Kardinalitäts-Clusteringproblem.

Das Clusteringproblem

Gegeben sei ein Graph $G = (V, E)$ mit positiven Kantengewichten $w_e \in \mathbb{Q}$, $e \in E$. Gegeben seien ferner Knotengewichte $f_u \in \mathbb{Q}$, $f_u > 0$, $u \in V$ und eine Zahl $F \in \mathbb{N}$. Gesucht sind Cluster $S_1, \ldots, S_r$, $\bigcup_{i=1}^{r} S_i = V$, so daß $w(\delta(S_1, \ldots, S_r))$ minimal ist und die folgenden Nebenbedingungen erfüllt sind:

$$\sum_{j \in S_i} f_j \leq F \text{ für alle } i = 1, \ldots, r.$$

Die Zahl F bezeichnen wir als **Clusterkapazität**.
Sind die Knotengewichte des Graphen identisch 1 , so nennen wir diese Variante **Kardinalitäts-Clusteringproblem.**

Entsprechend definieren wir das **r-(Kardinalitäts)-Clusteringproblem.** In diesem Fall ist die Anzahl der Cluster (r) vorgegeben. Gesucht sind Cluster $S_1, \ldots, S_r$, $\bigcup_{i=1}^{r} S_i = V$, so daß $w(\delta(S_1, \ldots, S_r))$ minimal ist und die folgenden Nebenbedingungen erfüllt sind:

$$\sum_{j \in S_i} f_j \leq F \text{ für alle } i = 1, \ldots, r.$$

Bemerkung:

(i) Das Clusteringproblem bzw. entsprechende Varianten sind auf Graphen mit positiven Knoten- und Kantengewichten definiert. Falls wir im folgenden Clusteringprobleme im Zusammenhang mit dem Plazieren erwähnen, so gehen wir stets davon aus, daß der Affinitätsgraph zugrundeliegt.

(ii) Bei der Definition des r-(Kardinalitäts)-Clusteringproblems sind wir davon ausgegangen, daß F die Kapazität für alle Cluster bezeichnet. Das Problem läßt sich analog definieren, wenn wir für jedes Cluster i a-proiori eine eigene Kapazität F_i zugrunde gelegt hätten. Da dieser allgemeinere Fall für unsere Anwendung jedoch ohne Bedeutung ist, haben wir uns entschieden, das Problem mit uniformer Clusterkapazität zu formulieren.

In den folgenden Kapiteln werden wir uns einerseits mit der heuristischen Lösung des Kardinalitäts-Clusteringproblems beschäftigen. Andererseits ist es unser Ziel, das r-

Clusteringproblem polyedrisch zu untersuchen. Aus diesem Grund stellen wir in dem verbleibenden Teil dieses Kapitels Modellierungsvorschläge für diese beiden Problemvarianten vor.

10.3 Modellierung des r-Clusteringproblems

Aus technischen Gründen führen wir zunächst eine Indexmenge $M = \{1, \ldots, r\}$ ein und denken uns die (noch zu bestimmenden) Cluster von 1 bis r numeriert. Anstelle der Schreibweise "das Cluster mit Nummer k werden wir die Abkürzung Cluster k gebrauchen. Ferner bezeichne F die Clusterkapazität. Der Graph $G = (V, E)$ habe positive Knotengewichte $f_i \in \mathbb{Q}$, $i \in V$ und Kantengewichte $c_e \in \mathbb{Q}$, $e \in E$.
Desweiteren gelte die zu Anfang des Kapitels eingeführte Notation.

Variablen

Für alle $i \in V$, $k \in M$ führen wir Variablen ein. Diese lassen sich wie folgt interpretieren.

$$x_{ik} = \begin{cases} 1, & \text{wenn Knoten } i \text{ dem} \\ & \text{Cluster } k \text{ zugewiesen wird.} \\ 0, & \text{sonst.} \end{cases}$$

Zusätzlich repräsentieren wir jede Kante $ij \in E$ durch eine entsprechende y_{ij}-Variable mit der Interpretation:

$y_{ij} = 1$, wenn Knoten i und j
verschiedenen Clustern zugewiesen wurden.
Anderenfalls kann der Wert 0 oder 1 angenommen werden.

Modelltechnisch wäre es eigentlich sauberer, die Variable y_{ij} so zu definieren, daß, wenn Zellen i und j dem gleichen Cluster zugewiesen werden, sie den Wert 0 annimmt. Wie sich gezeigt hat, führt diese "verschärfte Definition" zu zusätzlichen mathematischen Schwierigkeiten und erheblichen beweistechnischen Komplikationen. Wir haben uns daher für diese zunächst "unscharf" erscheinende Definition entschieden. Durch die Wahl der Zielfunktion wird jedoch sichergestellt, daß in jeder Optimallösung die Variable y_{ij} den Wert 0 annimmt, wenn die zugehörigen Knoten i und j in demselben Cluster sind.

Nebenbedingungen

(1) Jeder Knoten muß einem Cluster zugewiesen werden.

$\sum_{k \in M} x_{ik} = 1$ für alle $i \in V$.

(2) Die Clusterkapazität darf nicht überschritten werden.

$\sum_{i \in V} f_i x_{ik} \leq F$ für alle $k \in M$.

(3) Wenn Knoten i und j verschiedenen Clustern zugewiesen wurden, muß die zugehörige Kantenvariable den Wert 1 erhalten.

$x_{ik} + x_{jl} - y_{ij} \leq 1$ für alle $k, l \in M,\ k \neq l$ und $ij \in E$.

(4) Alle Variablen müssen 0/1-wertig sein.

$x_{ik},\ y_{ij} \in \{0,1\}$ für alle $i \in V,\ k \in M,\ ij \in E$.

Zielfunktion

Die Zielfunktion besteht aus der Summe der Gewichte der Kanten, die im Mehrfachschnitt zwischen verschiedenen Clustern liegen.

$$\min \sum_{ij \in E} c_{ij} y_{ij}.$$

10.4 Modellierung des Kardinalitäts-Clusteringproblems

Im Unterschied zu dem r-Clusteringproblem ist hier einerseits die Anzahl von Clustern nicht vorgegeben. Andererseits sind die Knotengewichte alle identisch 1. Dies führt zu einem sehr kompakten Modell, in welchem lediglich Kantenvariablen benötigt werden.
Sei $M = \{1, \ldots, r\}$ die Indexmenge für die Cluster. Ferner bezeichne F die Clusterkapazität. Der Graph $G = (V, E)$ habe positive Kantengewichte $c_e \in \mathbb{Q}$, $e \in E$.

Variablen

Für alle Kanten $ij \in E$ führen wir Variablen mit folgender Interpretation ein:

$$z_{ij} = \begin{cases} 1, & \text{wenn Knoten } i \text{ und } j \\ & \text{verschiedenen Clustern zugewiesen wurden.} \\ 0 \text{ oder } 1, & \text{anderenfalls.} \end{cases}$$

Nebenbedingungen

(1) Bezeichne $\mathcal{T}_F$ die Menge aller Bäume in G mit $|V(T)| = F + 1$ für alle $T \in \mathcal{T}_F$. Für jeden Baum $T \in \mathcal{T}_F$ dürfen die Knoten aus $V(T)$ nicht alle demselben Cluster zugeordnet werden. Dies modelliert die folgende Klasse von Ungleichung:

$z(T) \geq 1$ für alle Bäume $T \in \mathcal{T}_F$.

(2) Alle Variablen müssen 0/1-wertig sein.

$z_{ij} \in \{0,1\}$ für alle $ij \in E$.

Zielfunktion

Die Zielfunktion ist analog zu (10.2.1) definiert.

$$\min \sum_{ij \in E} c_{ij} z_{ij}.$$

(10.4.1) Lemma:

Obiges Modell stellt eine ganzzahlige Modellierung des Kardinalitäts-Clusteringproblems dar.

Beweis:

Sei z^* ein 0/1-Vektor, der alle Ungleichungen $z^*(T) \geq 1$ erfüllt. Seien ferner $S_1, \ldots, S_r$ die Knotenmengen der nicht-leeren, zusammenhängenden Komponenten in dem Untergraphen $(V, \hat{E})$, wobei $\hat{E} = \{ij \in E \mid z^*_{ij} = 0\}$. Dann definiert $\delta(S_1, \ldots, S_r)$ eine Lösung des Kardinalitäts-Clusteringproblems. Denn angenommen es existiert eine Cluster S, welchem mehr als F viele Elemente aus V zugewiesen wurden. Sei $S' \subseteq S$ so gewählt, daß $(S', \hat{E}(S'))$ zusammenhängend ist und $|S'| = F + 1$ gilt. Sei ferner T ein aufspannender Baum in dem Untergraphen $(S', \hat{E}(S'))$. Dann aber gilt sicherlich, daß $T \in \mathcal{T}_F$ und $z^*(T) = 0$ ist. Dies ist ein Widerspruch, da nach Voraussetzung z^* alle Ungleichungen des Typs (1) erfüllt. Umgekehrt ist es sicherlich richtig, daß jede Lösung des Kardinalitäts-Clusteringproblems die Bedingungen in obigem Modell erfüllt.

□

(10.4.2) Bemerkung:

Geht man davon aus, daß $G = (V, E)$ der vollständige Graph auf der Knotenmenge V ist, so lassen sich die Bedingungen (1) in Modell (10.4) durch die Bedingungen

$$(1') \sum_{j=1}^{i-1} y_{ji} + \sum_{j=i+1}^{n} y_{ij} \geq n - F \text{ für alle } i \in V$$

ersetzen.

In den folgenden Kapiteln werden wir uns einerseits damit beschäftigen, wie das Kardinalitäts-Clusteringproblem heuristisch gelöst werden kann. Andererseits werden wir das r-Clusteringproblem polyedrisch untersuchen.

Kapitel 11 Heuristische Lösung des Kardinalitäts-Clusteringproblems

Es hat sich in dem vorhergehenden Kapitel gezeigt, daß eine Lösung des Clusteringproblems oder einer Variante davon verwendet werden kann, um die Größenordnung von Instanzen des Plazierungsproblems zu reduzieren. In diesem Zusammenhang wollen wir nun einige heuristische Ideen zur Lösung des Kardinalitäts-Clusteringproblems vorstellen. Ferner werden wir aufzeigen, wie sich dieser Ansatz in das Plazierungsverfahren integrieren läßt. Die mit den Verfahren erzielten Ergebnisse an Testschaltungen werden abschließend präsentiert.

Aus Kapitel 10 wissen wir, daß sich das Clusteringproblem und Varianten davon als Mehrfachschnitt-Probleme mit zusätzlichen Nebenbedingungen betrachten lassen.
In der Literatur wird die (heuristische) Lösung des Mehrfachschnitt-Problems meist auf eine Folge von Schnittproblemen zurückgeführt. Zur heuristischen Lösung von Schnittproblemen existiert eine Vielzahl von Algorithmen. Interessante Ansätze in diesem Zusammenhang sind zunächst die iterativen Verbesserungsverfahren. Das bekannteste hiervon geht auf Kernighan und Lin [KL70] zurück, die ursprünglich das Problem, einen Graphen in zwei Partitionen gleichen Gewichts aufzuteilen, untersuchten. Die Idee dabei ist mit einer Anfangslösung zu starten und sukzessive zwei Knoten aus jeweils den beiden Partitionen mit minimalen Kosten zu tauschen. Dabei kann sich momentan der Zielfunktionswert gegenüber dem der vorangegangenen Iteration verschlechtern. Zu Ende des Verfahrens wird, beginnend mit der Anfangslösung, diese Sequenz des Austauschens bis zu dem Schritt durchlaufen, in welchem die Zielfunktion ihr Minimum (unter allen Schritten) annimmt. Dieses Verfahren wurde anschließend von Fiduccia und Mattheyses weiterentwickelt bzw. verallgemeinert, um das Problem, einen Hypergraphen in zwei Partitionen gleichen Gewichts aufzuteilen, effizient zu lösen [FM82]. Andere algorithmische Ideen beruhen auf Maximalfluß-Techniken, denen das "max flow - min cut"-Theorem für Graphen zugrundeliegt. Wieder andere Methoden erzeugen nach lokalen Gesichtspunkten eine Lösung des Mehrfachschnitt-Problems und können somit der Klasse der Greedy-Algorithmen zugerechnet werden. Schließlich werden in der Literatur noch nicht-deterministische Verfahren, wie zum Beispiel Simulated Annealing, zitiert. Es würde bei weitem den Rahmen dieses Kapitels sprengen, ausführlich auf heuristische Methoden zur Lösung von Mehrfach- bzw. Schnittproblemen einzugehen. Hierzu verweisen wir wiederum auf [L90].
Vielmehr wollen wir im folgenden kurz einige Algorithmen zur Lösung des Kardinalitäts-Clusteringproblems in dem mit einer Instanz des Plazierungsproblems assoziierten Affinitätsgraphen vorstellen. Selbstverständlich lassen sich diese auf beliebige Graphen anwenden. Da wir jedoch daran interessiert sind, Beispiele im Rahmen des Plazierungsproblems zu lösen, gehen wir stets davon aus, daß ein Affinitätsgraph zugrundeliegt. Die

im folgenden dargestellten Algorithmen wurden von uns implementiert und in das Plazierungsverfahren integriert. Für eine ausführlichere Darstellung verweisen wir auf die Arbeit [H92].

Gegeben sei eine Instanz des Plazierungsproblems. Es bezeichne N die Menge der Zellen und c_{ij} die Affinitätskoeffienten zwischen Zellen i und j. Davon ausgehend definieren wir den Affinitätsgraphen $G_{\mathcal{A}} = (N, E)$, dessen Kantengewichtung durch die Affinitätskoeffizienten gegeben ist. In diesem Zusammenhang läßt sich das Kardinalität-Clusteringproblem (KCP) folgendermaßen formulieren:
Gesucht sind Cluster $S_1, \ldots, S_r$, $\bigcup_{i=1}^{r} S_i = V$, so daß $w(\delta(S_1, \ldots, S_r))$ minimal ist und die folgenden Nebenbedingungen erfüllt sind:

$$\sum_{j \in S_i} f_j \leq F \text{ für alle } i = 1, \ldots, r.$$

Die Lösung $\delta(S_1, \ldots, S_r)$ soll minimales Gewicht haben.

Die von uns entwickelten Verfahren zur Lösung dieser Aufgabenstellung lassen sich in sogenannte "bottom-up"- und "top-down"-Heuristiken einteilen. Ersteren Verfahren ist gemeinsam, daß sie mit einer Partition beginnen, wobei jedes Element der Partition aus genau einem Knoten besteht. Sukzessive werden dann Elemente der Partition miteinander verschmolzen, solange dies zu einer Reduktion des Zielfunktionswerts führt. "top down"-Heuristiken dagegen gehen den umgekehrten Weg. Sie starten mit der (üblicherweise unzulässigen) Partiton, die als einziges Element die Knotenmenge V enthält. Anschließend wird iterativ ein Element der Partition in zwei kleinere aufgeteilt. Dieser Prozeß setzt sich fort, bis alle Elemente der Partition die entsprechenden Nebenbedingungen erfüllen. Von den fünf, im folgenden dargestellten Heuristiken, sind die ersten zwei "buttom up"-Heuristiken. Die letzten drei lassen sich den "top down"-Heuristiken zurechnen.

Das erste Verfahren besteht darin, sukzessive zwei Knoten zu kontrahieren. Bei der Aktualisierung der Kantengewichte werden nicht die Kantengewichte der kontrahierten Knoten aufaddiert. Vielmehr würde solch eine Vorgehen bewirken, daß der durch die Kontraktion entstandene neue Knoten in Folgeiterationen zu attraktiv wird und immer mehr Knoten anzieht. Durch die in Schritt 3.2 angegebene Berechnung findet eine Gewichtung statt, die dies verhindern soll.

Heuristik 11.1

Input: $G_{\mathcal{A}} = (N, E)$. Die Kantengewichte entsprechen den Affinitätskoeffizienten c_{ij} Gegeben ist ferner die Clusterkapazität F .

Output: Eine Lösung $\delta(S_1, \ldots, S_r)$ des Kardinalitäts-Clusteringproblems.

(1) Setze $S_i = \{i\}$ für alle $i \in N$, $r = n$ und $\mathcal{P} = \{S_1, \ldots, S_r\}$.

(2) Setze $A = \{ij \in E \mid |S_i| + |S_j| \leq F\}$

(3) Solange $A \neq \emptyset$ ist, führe die folgenden Anweisungen durch:

(3.1) Sei $c_{ij} = \max\{c_e \mid e \in A\}$.

(3.2) Kontrahiere i und j. Sei w der neue Knoten, der aus Kontraktion von i und j entsteht. Setze:

$c_{wk} = c_{ik}$, für alle $ik \in E$, $jk \notin E$,

$c_{wk} = c_{jk}$, für alle $jk \in E$, $ik \notin E$,

$c_{wk} = \frac{|S_i|c_{ik}+|S_j|c_{jk}}{|S_i|+|S_j|}$, für alle $ik \in E$, $jk \in E$.

Alle anderen Kantengewichte bleiben unverändert.

(3.3) Aktualisiere den Graphen $G_{\mathcal{A}}$, die Mengen A, $\mathcal{P}$ und die Zahl r.

Im Unterschied zu Heuristik 11.1 wird in dem folgenden Verfahren in jedem Schritt ein Matchingproblem gelöst und somit mehrere Knotenpaare simultan kontrahiert.

Heuristik 11.2

Input: $G_{\mathcal{A}} = (N, E)$. Ferner sei F die Clusterkapazität.

Output: Eine Lösung $\delta(S_1, \ldots, S_r)$ des Kardinalitäts-Clusteringproblems.

(1) Setze $S_i = \{i\}$ für alle $i \in N$, $r = n$ und $\mathcal{P} = \{S_1, \ldots, S_r\}$.

(2) Setze $A = \{ij \in E \mid |S_i| + |S_j| \leq F\}$

(3) Solange $A \neq \emptyset$ ist, führe die folgenden Anweisungen durch.

(3.1) Sei M ein Matching maximalen Gewichts in $G_{\mathcal{A}}$.

(3.2) Kontrahiere alle Knotenpaare (i, j), die inzident zu einer Kante $ij \in M$ sind. Sei w der neue Knoten, der aus Kontraktion von i und j entsteht. Setze:

$c_{wk} = c_{ik}$, für alle $ik \in E$, $jk \notin E$,

$c_{wk} = c_{jk}$, für alle $jk \in E$, $ik \notin E$,

$c_{wk} = \frac{|S_i|c_{ik}+|S_j|c_{jk}}{|S_i|+|S_j|}$, für alle $ik \in E$, $jk \in E$.

(3.3) Aktualisiere den Graphen $G_{\mathcal{A}}$, die Mengen A, $\mathcal{P}$ und die Zahl r.

Wie bereits zu Beginn des Kapitels erwähnt, teilen "top down"-Heuristiken sukzessive Partitionen in zwei kleinere auf. Algorithmisch lassen sich diese Verfahren also folgendermaßen beschreiben:

Input: $G_{\mathcal{A}} = (N, E)$. Ferner sei F die Clusterkapazität.

Output: Eine Lösung $\delta(S_1, \ldots, S_r)$ des Kardinalitäts-Clusteringproblems.

(1) Setze $\mathcal{P} = N$, $r = 1$ und $L = \{S \in \mathcal{P} \mid |S| > F\}$.

(2) Solange $L \neq \emptyset$ ist, führe die folgenden Anweisungen durch.

(2.1) Wähle eine Partiton $S \in L$.

(3.2) Bestimme Partitionen S^1 und S^2 von S.

(3.3) Setze $\mathcal{P} = (\mathcal{P} \setminus \{S\}) \cup \{S^1\} \cup \{S^2\}$, $r = r + 1$ und $L = \{S \in \mathcal{P} \mid |S| > F\}$.

Varianten solcher Verfahren unterscheiden sich dadurch, wie das Aufteilen einer Partition in zwei kleinere gelöst wird. Eine Möglichkeit der Aufteilung ist die folgende:

Heuristik 11.3

Sei $P = \{i_1, \ldots, i_p\}$ die aufzuteilende Partition. Setze $S^1 = \{i_1, \ldots, i_{\lfloor \frac{p}{2} \rfloor}\}$ und $S^2 = S \setminus S^1$.

Wende die Kernighan-Lin Heuristik auf die Partitionen S^1 und S^2 an.

Eine andere Idee, um den Aufteilungsschritt zu lösen, verwenden wir in **Algorithmus 11.4**. Sei S die aufzuteilende Partition. Dann bestimmen wir den minimalen Schnitt in dem durch S induzierten Untergraphen von $G_{\mathcal{A}}$. Dies läßt sich sehr effizient mit Hilfe des Gomory-Hu-Algorithmus [GH61] verwirklichen.
Nach umfangreichen Tests (vergleiche hierzu [H92]) haben wir uns dazu entschieden, eine Kombination der beiden Verfahren 11.3 und 11.4 als "top down"-Heuristik zu wählen. Bezeichne S wiederum die aufzuteilende Partiton.

Heuristik 11.5

Falls $|S| < \alpha$, wende Algorithmus (11.3) an,
anderenfalls wende Algorithmus (11.4) an.

Die empirischen Tests haben gezeigt, daß ein Parameter $\alpha = 50$ vernünftig erscheint.

Zur Verbesserung der durch die Primalheuristiken (11.1) bis (11.5) gelieferten Lösungen haben wir Verbesserungsverfahren implementiert. Diese führen eine Sequenz von 1-er oder 2-er Austauschschritten durch.

In dem folgenden möchten wir kurz erläutern, wie sich die oben vorgestellten Algorithmen in das Plazierungsprogramm integrieren lassen.

Zunächst lösen wir das Kardinalitäts-Clusteringproblem mit den entsprechenden Verfahren. Dies liefert uns eine Menge von Clustern. Für jedes der Cluster bestimmen wir ein Rechteck, welches die Fläche eben dieses Clusters repräsentieren soll. In der momentanen Implementierung verwenden wir eine sehr einfache Abschätzung der Clusterfläche, indem wir das Quadrat bestimmen, dessen Fläche der Summe der Zellenflächen des Clusters entspricht. Breite und Höhe des Quadrats erweitern wir anschließend um einen konstanten Faktor (20%). Letztere Zahlen definieren wir als Breite und Höhe des Clusters. Zwischen zwei Clustern i und j berechnet sich der Affinitätskoeffizient als Summe der Affinitätskoeffizienten zwischen den Zellen aus Cluster i und denen aus Cluster j. Die Liste der Cluster mit zugehörigen Breiten, Höhen und Affinitätskoeffizienten bildet dann den Input für das Plazierungsprogramm. Nachdem dieses, wie in Teil II dieser Arbeit ausgeführt, durchlaufen wurde, lösen wir die Cluster wieder auf. Konkret weisen wir alle Zellen eines Clusters mit ihrer linken unteren Ecke der Basiszelle zu, die mit der linken unteren Ecke des zugehörigen Clusters in der "Cluster"-Lösung zur Deckung gebracht wurde. Abbildung 11.6 zeigt die Auflösung der Cluster und damit die zugehörige Plazierung der Zellen im Beispiel der Schaltung soc_3. Dieses Vorgehen bewirkt, daß nach Auflösung der Cluster die

Plazierung nicht mehr notwendigerweise überlappungsfrei ist (vergleiche Abbildung 11.6). Deshalb wenden wir auf letztere Lösung wiederholt die Verfahren aus Kapitel 6.1 an, bis schließlich eine zulässige Lösung des Plazierungsproblems erreicht ist.

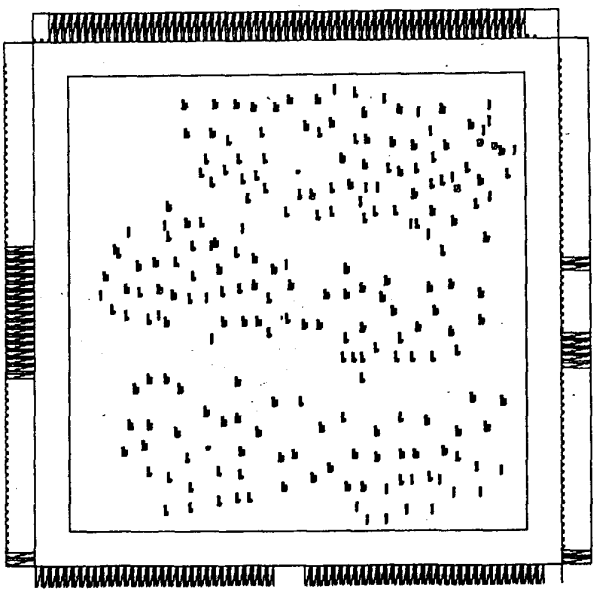


Abbildung 11.6

Exemplarisch stellen wir im folgenden einige Ergebnisse, welche wir aus den Tests bezüglich des Kardinalitäts-Clusteringproblem gewonnen haben, dar. In Tabelle 11.7 wird eine Bewertung der Endplazierungen der Schaltungen soc_1 bis soc_4 vorgenommen. Zur Lösung der entsprechenden Kardinalitäts-Clusteringprobleme verwendeten wir in allen Beispielen Algorithmus (11.5) und setzten die Clusterkapazität auf $F = 14$. Spalte 2 stellt wiederum Werte für die abgeschätzte Verdrahtungslänge dar. In Spalte 3 wurden die nicht realisierten 2-Pin-Verbindungen eingetragen, und die Zahlen in Spalte 4 entsprechen der benötigten CPU-Zeit, gemessen in Minuten. Die Abbildungen 11.8, 11.9, 11.10 und 11.11 zeigen die Endplazierung der Schaltungen soc_1, soc_2, soc_3 und soc_4.

	gesch. Verdrahtungsl.	nicht verbunden	CPU
soc_1	147024	43	4:35
soc_2	184715	6	9:01
soc_3	503481	50	35:05
soc_4	502372	532	28:07

Tabelle 11.7

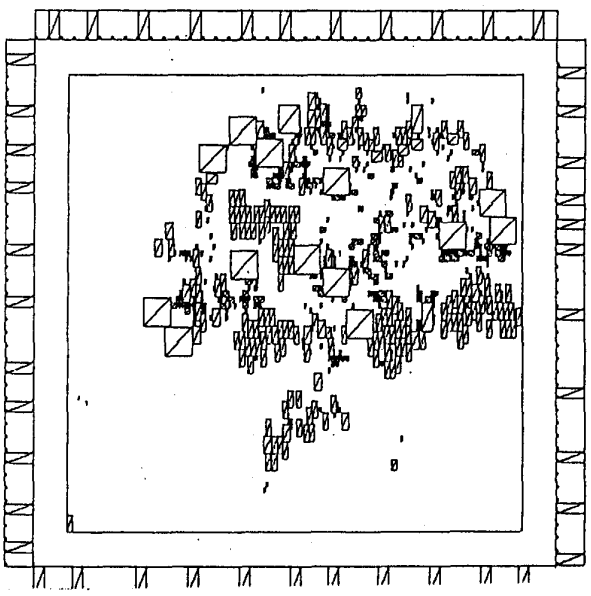


Abbildung 11.8

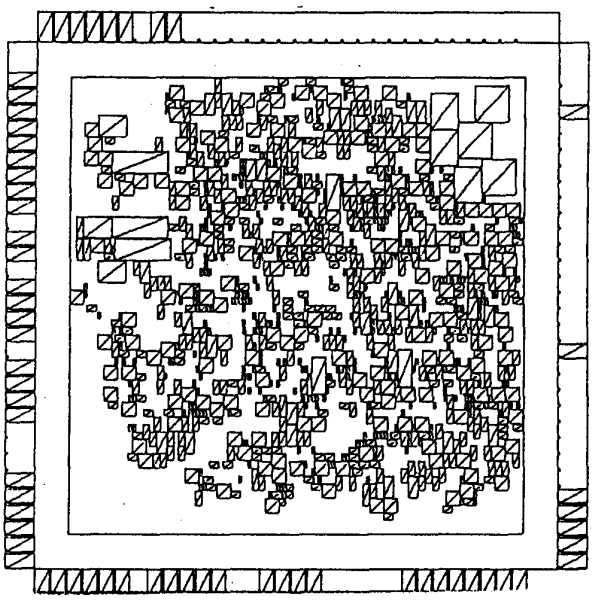


Abbildung 11.9

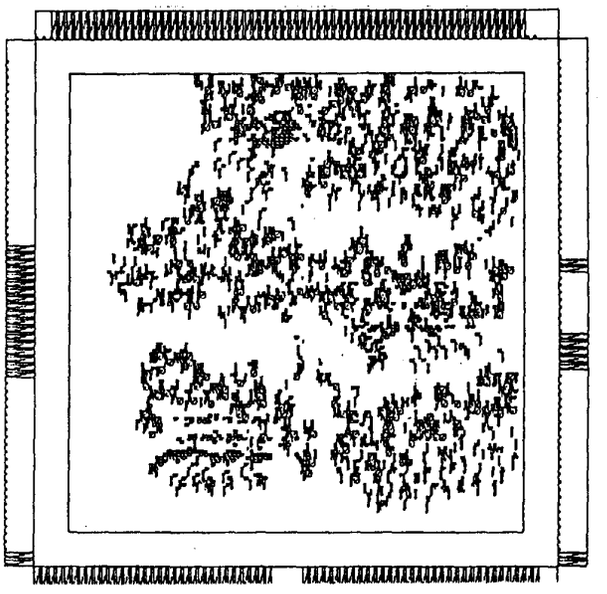


Abbildung 11.10

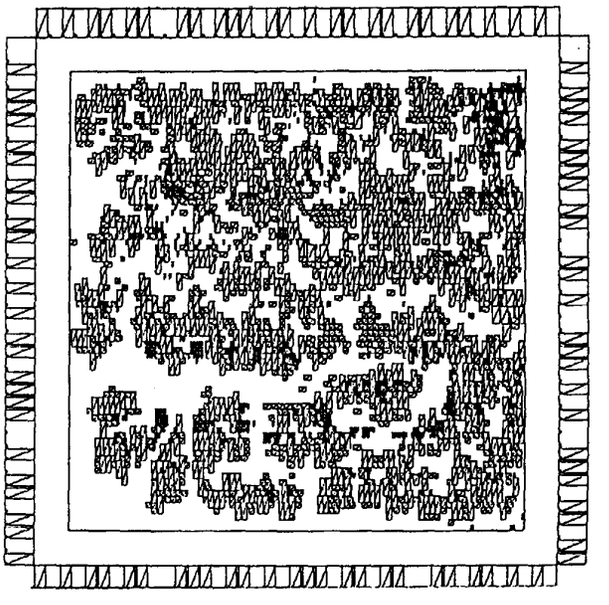


Abbildung 11.11

Die Ergebnisse zeigen, daß in der Plazierung der Schaltungen soc_1 und soc_4 bei vergleichbarer, abgeschätzter Verdrahtungslänge mehr als doppelt so viele Pinpaare nicht verbunden werden konnten, wie dies in den Lösungen, die ohne Clustering generiert wurden, der Fall

ist (vergleiche Kapitel 8). Für die Plazierungen der Schaltungen soc_2 und soc_3 trifft genau das Gegenteil zu. Hier liefern die Lösungen, die unter Verwendung der Clusteringheuristiken und der entsprechenden Parameter zustandekamen, sowohl eine geringere abgeschätzte Verdrahtungslänge als auch weniger nicht realisierte zwei-Pin-Verbindungen. Letztere zwei Ergebnisse sind insbesondere im Vergleich zu den von Min-Cut und Gordian gelieferten Resultaten sehr gut. Im Fall der Schaltung soc_3 erreichen wir im Bezug auf die abgeschätzte Verdrahtungslänge sogar eine Verbesserung von 37% bzw. 23% gegenüber den Verfahren Min-Cut bzw. Gordian.
Unübersehbar bleiben jedoch die schlechten Ergebnisse bezüglich der Schaltungen soc_1 und soc_4. Diese Ergebnisse sind keineswegs zufriedenstellend. Das kann natürlich daran liegen, daß in obiger Testreihe für alle Schaltungen der gleiche Parameter und der gleiche Algorithmus zur Lösung des KCP gewählt wurde. Aus diesem Grund haben wir für die Beispiele soc_1 und soc_4 weitere Tests durchgeführt.
Die Tabellen 11.12 und 11.13 bewerten die Endplazierung der Schaltungen soc_1 und soc_4 unter Verwendung anderer Algorithmen und anderer Parameter zur Lösung des KCP. Zum Vergleich haben wir in Zeile 2 der folgenden Tabellen nochmals das Ergebnis aus obigem Testlauf eingetragen. Das Ergebnis, welches in Zeile 3 der Tabelle 11.12 angegeben ist, kam unter Verwendung des Algorithmus 11.3 mit $F = 5$ zustande. In Zeile 4 der Tabelle 11.12 wird eine Bewertung der Plazierung vorgenommen, wobei das Problem KCP mit Hilfe von Algorithmus 11.3 und einer Clusterkapzität von $F = 3$ gelöst wurde. Zeile 3 der Tabelle 11.13 stellt die Ergebnisse dar, wobei Algorithmus 11.2 mit $F = 6$ zur Lösung des KCP eingesetzt wurde. Das Resultat, welches man für die Schaltung soc_4 unter Verwendung von Algorithmus 11.13 und $F = 3$ erhält, ist in Zeile 4 der Tabelle 11.13 dargestellt.

soc_1	gesch. Verdrahtungsl.	nicht verbunden	CPU
$F = 14$	147024	43	4:35
$F = 5$	143375	25	5:21
$F = 3$	150498	14	6:22

Tabelle 11.12

soc_4	gesch. Verdrahtungsl.	nicht verbunden	CPU
$F = 14$	502372	532	28:07
$F = 6$	476857	391	33:33
$F = 3$	445028	410	65:04

Tabelle 11.13

Analysiert man die Tabellen 11.12 und 11.13, so zeigt sich, daß die beiden Ergebnisse, die unter Verwendung einer kleineren Clusterkapazität zustande kamen, wesentlich besser sind,

als das, welches jeweils in Zeile 2 angegeben ist. Zu bemerken ist ferner, daß die Anzahl nicht realisierter 2-Pin-Verbindungen tendentiell mit abnehmender Clusterkapazität sinkt. Eine mögliche Erklärung dafür ist, daß es bei beiden Schaltungen einige Zellen gibt, die sehr viele Pins haben und mit vielen anderen Zellen durch Drähte verbunden werden müssen. Unter Verwendung der Clusteringprozeduren werden zunächst mehrere Zellen zusammengefasst und überlappen damit beim Auflösen der Cluster. In den Folgeschritten werden von dem Plazierungsprogramm diese Überlappungen aufgelöst, jedoch werden all die Zellen, die in einem Cluster zusammengefasst waren, sehr dicht beieinander plaziert (vergleiche Abbildungen 11.8 bis 11.11). Je mehr Zellen in einem Cluster sind, desto dichter wird im allgemeinen diese Packung. Bei Zellen, aus denen viele Drähte herausgehen, kann das zu nicht realisierten 2-Punkt-Verbindungen führen. Dies freilich ist nur eine plausible Erklärung und würde bedeuten, daß zur Lösung des KCP für Schaltungen mit komplizierter Netzstruktur sehr kleine Clusterkapazitäten gewählt werden sollten. Um dies endgültig zu entscheiden, sind sicherlich noch eine Vielzahl von Tests und weitere Analysen notwendig. Jedoch muß – so vermuten wir – für Schaltungen unterschiedlicher Struktur eine unterschiedliche Einstellung des Parameters F vorgenommen werden. Die Ergebnisse lassen in jedem Fall den Schluß zu, daß die Parameter zur Lösung des Kardinalitäts-Clusteringproblems im Rahmen des Plazierungsprogramms sehr sensitiv sind.
Zusammenfassend läßt sich sagen, daß es uns mit Hilfe der Lösung des Kardinalitäts-Clusteringproblems gelungen ist, für die Schaltungen soc_2 und soc_3 ausgezeichnete Lösungen zu finden. Die Ergebnisse für die Schaltungen soc_1 und soc_3 liefern eine gute, abgeschätzte Verdrahtungslänge. Die Anzahl nicht realisierter 2-Pin-Verbindungen hängt dabei von dem gewählten Verfahren zur Lösung des KCP und von der Clusterkapazität F ab. In den meisten Testläufen ist diese Anzahl größer als das der Fall ist, wenn man das Plazierungsprogramm in der Version, wie in Teil II beschrieben, ablaufen läßt. Ferner ist festzustellen, daß obige Resultate uns noch nicht ausreichend Information geben, um eine a-priori Einstellung der Parameter vorzunehmen. Die Laufzeiten unter Verwendung von Clustering reduzieren sich auf $\frac{1}{6}$ bis zu $\frac{1}{10}$ der Laufzeit des ursprünglichen Programms. Die genaue Zeitersparnis hängt von der Wahl der Algorithmen zur Lösung des KCP ab (vergleiche wiederum [H92]).

Kapitel 12 Das r-Clusteringpolytop

In diesem Kapitel werden wir eine Modellierung des r-Clusteringproblems näher untersuchen. Ausgangspunkt der Untersuchung ist das in Kapitel 10.4 vorgestellte Modell. Dieses werden wir transformieren in der Weise, daß das Polytop, definiert durch die konvexe Hülle der Lösungen des transformierten Modells, volldimensional ist.

Betrachten wir das in Kapitel 10.4 eingeführte Modell:

$$\min \sum_{ij \in E} c_{ij} y_{ij}$$

mit

(12.1)
$$\begin{aligned}
&\sum_{k \in M} x_{ik} = 1, \text{ für alle } i \in V,\\
&\sum_{i \in V} f_i x_{ik} \leq F, \text{ für alle } k \in M,\\
&x_{ik} + x_{jl} - y_{ij} \leq 1, \text{ für alle } ij \in E,\ k,l \in M,\ k \neq l,\\
&x_{ik} \in \{0,1\}, \text{ für alle } i \in V,\ k \in M,\\
&y_{ij} \in \{0,1\}, \text{ für alle } ij \in E.
\end{aligned}$$

Dabei gehen wir davon aus, daß $M = \{1,\ldots,r\}$ die Indexmenge für die Cluster ist. Die (noch zu bestimmenden) Cluster denken wir uns von 1 bis r numeriert. Mit $G = (V,E)$ bezeichnen wir den Graphen, wobei $V = \{1,\ldots,n\}$ der Menge der Knoten und E der Menge der Kanten entspricht. Die Clusterkapazität kürzen wir mit F ab. Ferner bezeichnet $f = (f_1, f_2, \ldots, f_n)$ die Liste der Knotengewichte. Das Gewicht einer Kante $e \in E$ werde mit c_e abgekürzt. Wir setzen voraus, daß alle Kanten- und Knotengewichte positiv sind. Die Variablenzuweisung $x_{ik} = 1$ ist so zu interpretieren, daß Knoten i dem Cluster mit der Nummer k zugewiesen wird. Mit einer Kante $ij \in E$ assoziieren wir eine Variable y_{ij}. Die Bedingungen $x_{ik} + x_{jl} - y_{ij} \leq 1$ für alle $k,l \in M,\ k \neq l,\ ij \in E$ drücken aus, daß die Variable y_{ij} den Wert 1 annehmen muß, wenn Knoten i und j verschiedenen Clustern zugewiesen werden.

Im folgenden werden wir zunächst Modell (12.1) so transformieren, daß die Menge der Gleichungen durch Ungleichungen ersetzt werden kann. Diese Transformation geht auf [MT81] zurück.

Wir definieren eine Zahl Γ gemäß: $\Gamma := \sum_{ij \in E} c_{ij} + 1$. Durch Hinzufügen von Hilfsvariablen v_i, $i \in V$ in (12.1) erhalten wir das folgende Modell.

$$\min \sum_{ij \in V} c_{ij} y_{ij} + \sum_{i \in V} \Gamma v_i$$

mit

(12.2)
$$\begin{aligned}
&\sum_{k \in M} x_{ik} + v_i = 1, \text{ für alle } i \in V,\\
&\sum_{i \in V} f_i x_{ik} \leq F, \text{ für alle } k \in M,\\
&x_{ik} + x_{jl} - y_{ij} \leq 1, \text{ für alle } ij \in E,\ k,l \in M,\ k \neq l,\\
&x_{ik} \in \{0,1\}, \text{ für alle } i \in V,\ k \in M,\\
&y_{ij} \in \{0,1\}, \text{ für alle } ij \in E,\\
&v_i \in \{0,1\}, \text{ für alle } i \in V.
\end{aligned}$$

Eliminiert man nun in den Gleichungen von (12.2) die Variablen v_i, $i = 1, \ldots, n$ so ergibt sich das zu (12.2) äquivalente System (12.3).

$$\min \sum_{ij \in E} c_{ij} y_{ij} + \sum_{i \in V} \Gamma (1 - \sum_{k \in M} x_{ik})$$

mit

(12.3)
$$\begin{aligned}
&\sum_{k \in M} x_{ik} \leq 1, \text{ für alle } i \in V,\\
&\sum_{i \in V} f_i x_{ik} \leq F, \text{ für alle } k \in M,\\
&x_{ik} + x_{jl} - y_{ij} \leq 1, \text{ für alle } ij \in E,\ k,l \in M,\ k \neq l,\\
&x_{ik} \in \{0,1\}, \text{ für alle } i \in V,\ k \in M,\\
&y_{ij} \in \{0,1\}, \text{ für alle } ij \in E.
\end{aligned}$$

Durch Umschreiben der Zielfunktion und Eliminieren der Konstante $n \cdot \Gamma$ in (12.3) erhält man schließlich Modell (12.4).

$$\min \sum_{ij \in E} c_{ij} y_{ij} - \Gamma \sum_{i \in V} \sum_{k \in M} x_{ik}$$

mit

(12.4)
$$\sum_{k \in M} x_{ik} \leq 1, \text{ für alle } i \in V,$$

$$\sum_{i \in V} f_i x_{ik} \leq F, \text{ für alle } k \in M,$$
$$x_{ik} + x_{jl} - y_{ij} \leq 1, \text{ für alle } ij \in E,\ k,l \in M,\ k \neq l,$$
$$x_{ik} \in \{0,1\}, \text{ für alle } i \in V,\ k \in M,$$
$$y_{ij} \in \{0,1\}, \text{ für alle } ij \in E.$$

Es ist offensichtlich, daß jede Optimallösung von (12.1) auch Optimallösung von (12.4) ist und umgekehrt. Die Zielfunktionswerte der Optimallösungen von (12.1) und (12.4) stimmen bis auf die additive Konstante $n \cdot \Gamma$ überein.

Für den Rest des Kapitels wird Modell (12.4) bzw. das mit ihm assoziierte Polytop der Gegenstand unserer Untersuchungen sein. Für eine Instanz des r-Clusteringproblems definieren wir das **r-Clusteringpolytop** $\mathbf{C}^r(G,f,F)$ folgendermaßen :
(12.5) Definition: $\mathbf{C}^r(G,f,F) = conv\{(x^T,y^T)^T \in \mathbb{R}^{nr} \times \mathbb{R}^z \mid (x^T,y^T)^T$ erfüllt (12.4)$\}$, wobei $f = (f_1,\ldots,f_n)$ gelten soll.

Bevor wir damit beginnen, die Facettenstruktur des r-Clusteringpolytops näher zu untersuchen, führen wir noch einige Notationen ein. Zunächst werden wir anstelle der Schreibweise $(x^T,y^T)^T \in \mathbb{R}^{nr} \times \mathbb{R}^z$ die Abkürzung $(x,y) \in \mathbb{R}^{nr} \times \mathbb{R}^z$ verwenden. Mit e_v^s bezeichnen wir den v-ten Einheitsvektor im s-dimensionalen Vektorraum über $\mathbb{R}$. Sei P ein Polyeder, welches eine Teilmenge des $\mathbb{R}^s$ ist,und sei $c^Tu \leq \gamma$ gültige Ungleichung für P. Dann bezeichnen wir mit $EQ(P,c^Tu \leq \gamma)$ die Menge aller Vektoren in P, welche $c^Tu = \gamma$ erfüllen. Falls aus dem Zusammenhang hervorgeht, auf welches Polyeder sich die Ungleichung $c^Tu \leq \gamma$ bezieht, so verwenden wir die abkürzende Schreibweise $EQ(c^Tu \leq \gamma)$. Der Vektor $\mathbf{1}^s$ bezeichnet den s-dimensinalen Vektor, dessen Komponenten alle den Wert 1 haben. Analog entspricht der Vektor $\mathbf{0}^s$ dem s-dimensinalen Vektor, dessen Komponenten alle identisch 0 sind.
Falls aus dem Zusammenhang die Dimension des Vektorraums klar ersichtlich ist, werden wir auch oft die abkürzenden Schreibweisen $e_v, \mathbf{0}, \mathbf{1}$ benutzen.

Zunächst bestimmen wir die Dimension des Polyeders $\mathbf{C}^r(G,f,F)$. Dies ist das Resultat des Lemmas 12.6.

(12.6) Lemma: $dim(\mathbf{C}^r(G,f,F)) = nr + z - |\gamma(N,M)|$, *wobei*
$$\gamma(N,M) := \{(i,k) \in V \times M \mid f_i > F\}.$$

Beweis. Es ist offensichtlich, daß für alle $i_0 \in V$ und $k_0 \in M$ mit $(i_0,k_0) \in \gamma(N,M)$ gilt: $\mathbf{C}^r(G,f,F) \subseteq \{(x,y \in \mathbb{R}^{nr} \times \mathbb{R}^z \mid x_{i_0k_0} = 0\}$. Deshalb ist
$$dim(\mathbf{C}^r(G,f,F)) \leq nr + z - |\gamma(N,M)|.$$
Der Vektor $(\mathbf{0},\mathbf{0}) \in \mathbb{R}^{nr} \times \mathbb{R}^z$ ist in $\mathbf{C}^r(G,f,F)$ enthalten. Ferner sind die Vektoren $(e_{ik},\mathbf{0}) \in \mathbb{R}^{nr} \times \mathbb{R}^z$ ($i \in V,\ k \in M$ mit $(i_0,k_0) \notin \gamma(N,M)$) Elemente des Polytops $\mathbf{C}^r(G,f,F)$. Da für alle $ij \in E$ die Vektoren $(\mathbf{0},e_{ij})$ ebenfalls in $\mathbf{C}^r(G,f,F)$ enthalten sind, existieren $nr + z - |\gamma(N,M)| + 1$ affin unabhängige Vektoren. Da diese Elemente des Polytops $\mathbf{C}^r(G,f,F)$ sind, gilt $dim(\mathbf{C}^r(G,f,F)) \geq nr + z - |\gamma(N,M)|$.

Dies beweist Lemma (12.6). □

In den folgenden Kapitel werden wir das Polytop $\mathbf{C}^r(G, f, F)$ durch Facetten partiell beschreiben. Aus Kapitel 10 wissen wir, daß sich das Clusteringproblem und Varianten davon als Mehrfachschnitt-Probleme mit zusätzlichen Nebenbedingungen betrachten lassen. Polyedrische Studien für das Mehrfachschnitt-Problem finden sich unter anderem in [W86], [CR89], [CR91], [DGL91] und [DGL92].

Für die folgenden Untersuchungen setzen wir stets voraus, daß $f_i \leq F$ für alle $f_i \in f$ gilt. In diesem Fall ist nach Lemma 12.6 das r-Clusteringpolytop volldimensional.

Kapitel 13 Grundlegende Facetten des r-Clusteringpolytops

In diesem Kapitel untersuchen wir, ob und gegebenenfalls unter welchen Bedingungen die in Modell (12.4) vorkommenden Ungleichungen Facetten von $\mathbf{C}^r(G,f,F)$ definieren.

Für die Darstellung dieses Kapitels greifen wir auf die Notation aus Kapitel 10 zurück. Insbesondere bezeichnet $G=(V,E)$ den zugrundeliegenden Graphen und M die Indexmenge für die Cluster. Es gelte ferner $n=|V|$, $r=|M|$ und $z=|E|$.

(13.1) Lemma: *Gegeben seien $i_0 \in V$ und $k_0 \in M$. Dann definiert die Ungleichung $x_{i_0k_0} \geq 0$ eine Facette von $\mathbf{C}^r(G,f,F)$.*

Beweis:
Die Ungleichung ist trivialerweise gültig.
Der Vektor $(\mathbf{0},\mathbf{0}) \in \mathbb{R}^{rm} \times \mathbb{R}^z$ ist in $EQ(x_{i_0k_0} \geq 0)$ enthalten. Ferner erfüllen die Vektoren $(e_{ik},\mathbf{0}) \in \mathbb{R}^{rm} \times \mathbb{R}^z$ für alle $i \in V \setminus \{i_0\}$, $k \in M \setminus \{k_0\}$ die Gleichung $x_{i_0k_0}=0$. Für alle $ij \in E$ sind schließlich die Vektoren $(\mathbf{0},e_{ij})$ in $EQ(x_{i_0k_0} \geq 0)$ enthalten. Somit existieren $dim(\mathbf{C}^r(G,f,F))$ affin unabhängige Vektoren aus $\mathbf{C}^r(G,f,F)$, welche obige Ungleichung mit Gleichheit erfüllen. □

Ähnlich läßt sich zeigen, daß die Ungleichungen $y_{ij} \geq 0$ und $y_{ij} \leq 1$ Facetten von $\mathbf{C}^r(G,f,F)$ definieren.

(13.2) Lemma: *Gegeben sei $i_0j_0 \in E$.*
(i) Die Ungleichung $y_{i_0j_0} \geq 0$ definiert eine Facette von $\mathbf{C}^r(G,f,F)$.
(ii) Die Ungleichung $y_{i_0j_0} \leq 1$ definiert eine Facette von $\mathbf{C}^r(G,f,F)$.

Beweis:
Die Ungleichungen sind trivialerweise gültig.

(i)

Der Vektor $(\mathbf{0},\mathbf{0}) \in \mathbb{R}^{rm} \times \mathbb{R}^z$ ist in $EQ(y_{i_0j_0} \geq 0)$ enthalten. Ferner erfüllen die Vektoren $(e_{ik},\mathbf{0}) \in \mathbb{R}^{rm} \times \mathbb{R}^z$ für alle $i \in V$, $k \in M$ die Ungleichung $y_{i_0j_0} \geq 0$ mit Gleichheit. Für alle $ij \in E \setminus \{i_0j_0\}$ sind schließlich die Vektoren $(\mathbf{0},e_{ij})$ in $EQ(y_{i_0j_0} \geq 0)$ enthalten. Somit existieren $dim(\mathbf{C}^r(G,f,F))$ viele affin unabhängige Vektoren aus $\mathbf{C}^r(G,f,F)$, welche die Ungleichung $y_{i_0j_0} \geq 0$ mit Gleichheit erfüllen.

(ii)

Der Vektor $(\mathbf{0},e_{i_0j_0}) \in \mathbb{R}^{rm} \times \mathbb{R}^z$ ist in $EQ(y_{i_0j_0} \geq 0)$ enthalten. Ferner erfüllen die Vektoren $(e_{ik},e_{i_0j_0}) \in \mathbb{R}^{rm} \times \mathbb{R}^z$ für alle Paare $(i,k) \in V \times M$ die Ungleichung $y_{i_0j_0} \geq 0$ mit Gleichheit. Für alle $ij \in E \setminus \{i_0j_0\}$ sind dann die Vektoren $(\mathbf{0},e_{ij}+e_{i_0j_0})$ in $EQ(y_{i_0j_0} \geq 0)$ enthalten. Somit existieren $dim(\mathbf{C}^r(G,f,F))$ viele affin unabhängige Vektoren aus $\mathbf{C}^r(G,f,F)$, welche die Ungleichung $y_{i_0j_0} \leq 1$ mit Gleichheit erfüllen. □

Offensichtlich definiert eine Ungleichung $x_{i_0k_0} \leq 1$ für ein $i_0 \in V$ und ein $k_0 \in M$ keine Facette von $\mathbf{C}^r(G,f,F)$, da sie durch eine positive Linearkombination der gültigen Ungleichungen $\sum_{k\in M} x_{i_0k} \leq 1$ und $-x_{i_0k} \leq 0$ für alle $k \in M \setminus \{k_0\}$ erzeugt werden kann.
Eine Ungleichung $\sum_{k\in M} x_{i_0k} \leq 1$ für ein $i_0 \in V$ definiert eine Facette von $\mathbf{C}^r(G,f,F)$. Dies zeigen wir in Lemma 13.3.

(13.3) Lemma: *Gegeben sei $i_0 \in V$.*
Dann definiert die Ungleichung $\sum_{k\in M} x_{i_0k} \leq 1$ eine Facette von $\mathbf{C}^r(G,f,F)$.

Beweis:
Angenommen, $EQ(\sum_{k\in M} x_{i_0k} \leq 1) \subseteq \{(x,y) \in \mathbb{R}^{rm} \times \mathbb{R}^z \mid a^Tx + b^Ty \leq \beta\}$, wobei die Ungleichung $a^Tx + b^Ty \leq \beta$ eine Facette für $\mathbf{C}^r(G,f,F)$ definiert.
Sei $k_0 \in M$. Dann sind die Vektoren $(e_{i_0k_0}, \mathbf{0})$ und $(e_{i_0k_0}, e_{ij})$ Elemente von $EQ(\sum_{k\in M} x_{i_0k} \leq 1)$. Dies gilt für alle $ij \in E$. Obige Annahme impliziert dann, daß $a_{i_0k_0} + b_{ij} = a_{i_0k_0}$. Deshalb ist $b_{ij} = 0$ für alle $ij \in E$.
Ferner sind die Vektoren $(e_{i_0k_0}, \mathbf{1})$ und $(e_{i_0k_0} + e_{i'k'}, \mathbf{1})$ Elemente von $EQ(\sum_{k\in M} x_{i_0k} \leq 1)$. Dies gilt für alle $i' \in V \setminus \{i_0\}$ und alle $k' \in M \setminus \{k_0\}$. Folglich sind die Vektoren $(e_{i_0k_0}, \mathbf{1})$ und $(e_{i_0k_0} + e_{i'k'}, \mathbf{1})$ auch Elemente von $EQ(a^Tx + b^Ty \leq \beta)$. Dies liefert: $a_{i'k'} = 0$ für alle $i' \in V \setminus \{i_0\}$ und $k' \in M \setminus \{k_0\}$.
Sei $k_1 \in M \setminus \{k_0\}$ gegeben. Dann sind die Vektoren $(e_{i_0k_1}, \mathbf{1})$ und $(e_{i_0k_1} + e_{i'k_0}, \mathbf{1})$ Elemente von $EQ(\sum_{k\in M} x_{i_0k} \leq 1)$ und damit auch in $EQ(a^Tx + b^Ty \leq \beta)$ enthalten. Dies gilt für alle $i' \in V \setminus \{i_0\}$. Diese Lösungen erzwingen, daß $a_{i'k_0} = 0$ ist für alle $i' \in V \setminus \{i_0\}$.
Da für alle Paare von Zahlen (k_0,k_1), $k_0 \neq k_1$, $k_0,k_1 \in M$ die Vektoren $(e_{i_0k_0}, \mathbf{0})$ und $(e_{i_0k_1}, \mathbf{0})$ zu der Menge $EQ(\sum_{k\in M} x_{i_0k} \leq 1)$ gehören gilt $a_{i_0k_0} = a_{i_0k_1}$ für alle (k_0,k_1), $k_0 \neq k_1$, $k_0,k_1 \in M$.
Somit haben wir gezeigt, daß die Ungleichungen $a^Tx + b^Ty \leq \beta$ und $\sum_{k\in M} x_{i_0k} \leq 1$ bis auf Multiplikation mit einer skalaren Größe identisch sind. Dies beweist Lemma 13.3. □

Schließlich bleibt uns noch die Aufgabe, die Ungleichungen $x_{ik} + x_{jl} - y_{ij} \leq 1$ aus Modell (12.4) für alle $ij \in E$, $k,l \in M$, $k \neq l$ zu untersuchen. Diese Ungleichungen definieren äußert selten Facetten für $\mathbf{C}^r(G,f,F)$. Vielmehr läßt sich zeigen, daß eine modifizierte Version dieser Ungleichungen facetten-defineirend ist. Das ist das Resultat des Lemmas (13.4).

(13.4) Lemma: *Seien $i_0,j_0 \in V$, $i_0 \neq j_0$ und $l_0 \in M$. Ferner sei $i_0j_0 \in E$.*
(i) Gilt $f_{i_0} + f_{j_0} \leq F$, dann definiert die Ungleichung

$$\sum_{k\in M\setminus l_0} x_{i_0k} + x_{j_0l_0} - y_{i_0j_0} \leq 1. \tag{13.4.1}$$

eine Facette von $\mathbf{C}^r(G,f,F)$.
(ii) Gilt $f_{i_0} + f_{j_0} > F$, dann definiert die Ungleichung

(13.4.2) $$\sum_{k\in M} x_{i_0 k} + \sum_{k\in M} x_{j_0 k} - y_{i_0 j_0} \leq 1.$$

eine Facette von $\mathbf{C}^r(G,f,F)$.

Beweis:

(i)

Wir beweisen zunächst die Gültigkeit der Ungleichung (13.4.1).

Alle Punkte (x,y) *aus* $\mathbf{C}^r(G,f,F)$ *erfüllen die Ungleichungen:*

$$\sum_{k\in M\setminus l_0} x_{i_0 k} \leq 1 \text{ und } x_{j_0 l_0} \leq 1.$$

Nehmen wir an (x,y) *erfüllt* $\sum_{k\in M\setminus l_0} x_{i_0 k} = 1$ *und* $x_{j_0 l_0} = 1$. *Dann existiert ein* $k_0 \in M \setminus \{l_0\}$, *so daß* $x_{i_0 k_0} = 1$ *gilt. Daraus folgt, daß* $y_{i_0 j_0}$ *den Wert 1 annehmen muß. Somit gilt:*

$\sum_{k\in M\setminus l_0} x_{i_0 k} + x_{j_0 l_0} - y_{i_0 j_0} \leq 1.$

Die Ungleichung $\sum_{k\in M\setminus l_0} x_{i_0 k} + x_{j_0 l_0} - y_{i_0 j_0} \leq 1$ *definiert eine Facette von* $\mathbf{C}^r(G,f,F)$. *Um dies zu beweisen, nehmen wir an, daß*

$$EQ((13.4.1)) \subseteq \{(x,y) \in \mathbb{R}^{rm} \times \mathbb{R}^z \mid a^T x + b^T y = \beta\},$$

wobei die Ungleichung $a^T x + b^T y \leq \beta$ *eine Facette für* $\mathbf{C}^r(G,f,F)$ *definiert.*

(1) Sei $k_1 \in M$, $k_1 \neq l_0$. *Dann sind die Vektoren* $(e_{i_0 k_1}, \mathbf{0})$ *und* $(e_{i_0 k_1}, e_{ij})$ *für alle* $ij \in E$, $ij \neq i_0 j_0$ *Elemente von* $EQ((13.4.1))$. *Dies impliziert* $b_{ij} = 0$ *für alle* $ij \in E$, $ij \neq i_0 j_0$.

(2) Sei $k_1 \in M \setminus \{l_0\}$. *Für alle* $i \in V \setminus \{i_0, j_0\}$, $k \in M \setminus \{k_1\}$ *erfüllen die Vektoren* $(e_{i_0 k_1}, \mathbf{1} - e_{i_0 j_0})$ *und* $(e_{i_0 k_1} + e_{ik}, \mathbf{1} - e_{i_0 j_0})$ *obige Ungleichung mit Gleichheit. Da* $k_1 \in M$ *beliebig wählbar ist, folgt:* $a_{ik} = 0$ *für alle* $i \in V \setminus \{i_0, j_0\}$, $k \in M$. *Sei wiederum* $k_1 \in M \setminus \{l_0\}$ *beliebig. Dann erfüllen die Vektoren* $(e_{i_0 k_1}, \mathbf{0})$ *und* $(e_{i_0 k_1} + e_{j_0 k_1}, \mathbf{0})$ *obige Ungleichung mit Gleichheit. Dies impliziert:* $a_{j_0 k_1} = 0$ *für alle* $k_1 \in M \setminus \{l_0\}$. *Analog können wir folgern, daß* $a_{i_0 l_0} = 0$ *ist.*

(3) Seien $k_1, k_2 \in M$, $k_1, k_2 \neq l_0$, $k_1 \neq k_2$. *Dann gehören die Vektoren* $(e_{i_0 k_1}, 0)$, $(e_{i_0 k_2}, \mathbf{0})$ *und* $(e_{j_0 l_0}, \mathbf{0})$ *der Seitenfläche* $EQ((13.4.1))$ *an. Somit gilt* $a_{i_0 k_1} = a_{i_0 k_2} = a_{j_0 l_0}$ *für alle* $k_1, k_2 \in M$, $k_1, k_2 \neq l_0$, $k_1 \neq k_2$. *Da schließlich auch der Vektor* $(e_{i_0 k_1} + e_{j_0 l_0}, e_{i_0 j_0})$ *für* $k_1 \in M$, $k_1 \neq l_0$ *ein Element von* $EQ((13.4.1))$ *ist, gilt* $b_{i_0 j_0} = -a_{i_0 k_1}$.

Dies schließt den Beweis von (13.4 i) ab.

Der Beweis von (13.4 ii) verläuft ähnlich. Der Vollständigkeit wegen führen wir ihn trotzdem aus.

(ii)

Wir beweisen wieder zunächst die Gültigkeit der Ungleichung.

Alle Punkte (x,y) aus $\mathbf{C}^r(G,f,F)$ erfüllen die Ungleichungen:

$$\sum_{k\in M} x_{i_0 k} \leq 1 \text{ und } \sum_{k\in M} x_{j_0 k} \leq 1.$$

Nehmen wir an (x,y) erfüllt die beiden Gleichungen: $\sum_{k\in M} x_{i_0 k} = 1$ und $\sum_{k\in M} x_{j_0 k} = 1$. Gemäß Voraussetzung können wir annehmen, daß $x_{i_0 k_1} = 1$ und $x_{j_0 k_2} = 1$ für $k_1 \neq k_2$. Dann aber nimmt die Variable $y_{i_0 j_0}$ den Wert 1 an. Somit gilt:
$\sum_{k\in M} x_{i_0 k} + \sum_{k\in M} x_{j_0 k} - y_{i_0 j_0} \leq 1$.
Ungleichung(13.4.2) definiert eine Facette von $\mathbf{C}^r(G,f,F)$.
Angenommen,

$$EQ((13.4.2)) \subseteq \{(x,y) \in \mathbb{R}^{rm} \times \mathbb{R}^z \mid a^T x + b^T y = \beta\},$$

wobei die Ungleichung $a^T x + b^T y \leq \beta$ eine Facette für $\mathbf{C}^r(G,f,F)$ definiert.

(1) Sei $k_1 \in M$. Dann sind die Vektoren $(e_{i_0 k_1}, \mathbf{0})$ und $(e_{i_0 k_1}, e_{ij})$ für alle $ij \in E$, $ij \neq i_0 j_0$ Elemente von $EQ((13.4.2))$. Dies impliziert $b_{ij} = 0$ für alle $ij \in E$, $ij \neq i_0 j_0$.

(2) Seien $k, k_0 \in M$, $k \neq k_0$ gegeben. Für alle $i \in V \setminus \{i_0, j_0\}$ erfüllen die Vektoren $(e_{i_0 k_0}, \mathbf{1} - e_{i_0 j_0})$ und $(e_{i_0 k_0} + e_{ik}, \mathbf{1} - e_{i_0 j_0})$ obige Ungleichung mit Gleichheit. Da k und k_0 beliebig wählbar sind, folgt: $a_{ik} = 0$ für alle $k \in M$ und $i \in V \setminus \{i_0, j_0\}$.

(3) Seien nun $k_1, k_2 \in M$, $k_1 \neq k_2$. Dann gehören die Vektoren $(e_{i_0 k_1}, \mathbf{0})$ und $(e_{i_0 k_2}, \mathbf{0})$ der Seitenfläche $EQ((13.4.2))$ an. Gleiches gilt für die Vektoren $(e_{j_0 k_1}, 0)$ und $(e_{j_0 k_2}, \mathbf{0})$. Somit gilt $a_{i_0 k_1} = a_{i_0 k_2} = a_{j_0 k_1} = a_{j_0 k_2}$ für alle $k_1, k_2 \in M$, $k_1 \neq k_2$. Da schließlich auch der Vektor $(e_{i_0 k_1} + e_{j_0 k_2}, e_{i_0 j_0})$ für $k_1, k_2 \in M$, $k_1 \neq k_2$ ein Element von $EQ((13.4.2))$ ist, gilt $b_{i_0 j_0} = -a_{i_0 k_1}$.

Somit ist auch der zweite Teil des Lemmas 13.4 bewiesen.

□

Kapitel 14 Einige allgemeine Resultate

Von dem r-Clusteringpolytop haben wir bisher einige einfache Facetten bestimmt. Interessanterweise gibt es Eigenschaften, die allgemein von facetten-definierenden Ungleichungen für $\mathbf{C}^r(G,f,F)$ erfüllt werden. Mit Resultaten dieser Art wollen wir uns in diesem Kapitel beschäftigen. Ferner werden wir das Problem des "Lifting" (siehe [P75], [Z78]) behandeln.

Für die Darstellung dieses Kapitels greifen wir auf die Notation aus Kapitel 10 zurück. Insbesondere bezeichnet $G=(V,E)$ den zugrundeliegenden Graphen und M die Indexmenge für die Cluster. Es gelte ferner $n=|V|$, $r=|M|$ und $z=|E|$. Ohne Beschränkung der Allgemeinheit setzen wir im Folgenden $V=\{1,\ldots,n\}$ voraus.

(14.1) Lemma:
Sei $a^Tx+b^Ty\leq\beta$ eine nicht triviale Facette für $\mathbf{C}^r(G,f,F)$. Dann sind die Koeffizienten von a alle nicht negativ und die von b alle nicht positiv.

Beweis:
Nehmen wir zunächst an, es existiert ein $i_0\in V$ und ein $k_0\in M$ mit $a_{i_0k_0}<0$. Da die Facette $a^Tx+b^Ty\leq\beta$ nicht trivial für $\mathbf{C}^r(G,f,F)$ ist, gibt es einen Vektor $(x,y)\in EQ(a^Tx+b^Ty\leq\beta)$ mit $x_{i_0k_0}=1$. Anderenfalls wäre $EQ(a^Tx+b^Ty\leq\beta)\subseteq\{(x,y)\in\mathbf{C}^r(G,f,F)|x_{i_0k_0}=0\}$. Die Ungleichung $a^Tx+b^Ty\leq\beta$ muß insbesondere für den Vektor $(x-e_{i_0k_0},y)\in\mathbf{C}^r(G,f,F)$ gültig sein. Es gilt aber, $a^T(x-e_{i_0k_0})+b^Ty=a^Tx+b^Ty-a_{i_0k_0}>\beta$. Dies steht im Widerspruch zu der Voraussetzung, daß $a^Tx+b^Ty\leq\beta$ eine Facette für $\mathbf{C}^r(G,f,F)$ definiert.

Nun angenommen, es existiert ein $i_0j_0\in E$ mit $b_{i_0j_0}>0$. Da die facetten-definierende Ungleichung $a^Tx+b^Ty\leq\beta$ nicht trivial für $\mathbf{C}^r(G,f,F)$ ist, gibt es eine Vektor $(x,y)\in EQ(a^Tx+b^Ty\leq\beta)$, so daß $y_{i_0j_0}=0$ ist. Da der Vektor $(x,y+e_{i_0j_0})$ ein Element des Polyeders $\mathbf{C}^r(G,f,F)$ ist, muß $a^Tx+b^T(y+e_{i_0j_0})\leq\beta$ erfüllt sein. Dies steht im Widerspruch zu $a^Tx+b^T(y+e_{i_0j_0})=a^Tx+b^Ty+b_{i_0j_0}>\beta$.

□

Eine weitere Eigenschaft, die facetten-definierenden Ungleichungen für das r-Clusteringpolytop gemeinsam ist, zeigt das folgende Lemma auf.

(14.2) Lemma:
Sei $a^Tx+b^Ty\leq\beta$ nicht triviale, facetten-definierende Ungleichung für $\mathbf{C}^r(G,f,F)$. Dann sind die Koeffizienten von a nicht alle 0.

Beweis:
Nehmen wir an, $b^Ty\leq\beta$ ist eine nicht triviale, facetten-definierende Ungleichung für $\mathbf{C}^r(G,f,F)$.
Nach Lemma 14.1 gilt: $b_{ij}\leq 0$ für alle $ij\in E$. Da $(0,0)$ ein Vektor des Polyeders $\mathbf{C}^r(G,f,F)$ ist, folgt nach Einsetzen in die Ungleichung: $\beta\geq 0$.

Wäre $\beta > 0$, so würde gelten:

$$EQ(\mathbf{C}^r(G,f,F), b^T y \leq \beta) = \emptyset,$$

da die Koeffizienten b_{ij}, $ij \in E$ alle nicht positiv sind. Dies widerspräche der Voraussetzung, daß $b^T y \leq \beta$ eine Facette definiert.
Somit muß $\beta = 0$ gelten. In diesem Fall aber ist die Ungleichung eine Linearkombination der Ungleichungen $-y_{ij} \leq 0$ für alle $ij \in E$ mit $b_{ij} \neq 0$. Dies ist ebenfalls ein Widerspruch. □

Die folgenden Sätze haben einge "Lifting"-Resultate zum Inhalt. Die Methode des Lifting geht auf Padberg (vergleiche hierzu [P75]) zurück. Sei $a^T x + b^T y \leq \beta$ eine Ungleichung, welche eine nicht triviale Facette für $\mathbf{C}^r(G,f,F)$ definiert. Gegeben sei ferner ein r'-Clusteringpolytop $\mathbf{C}^{r'}(G',f',F)$ mit $f'_v = f_v$ für alle $v \in V$ und $V \subseteq V'$, $E \subseteq E'$. Unter dem **"Lifting"** der Facette $EQ(\mathbf{C}^r(G,f,F), a^T x + b^T y \leq \beta)$ zu einer Facette für das Polytop $\mathbf{C}^{r'}(G',f',F)$ verstehen wir eine Ungleichung $a'^T x + b'^T y \leq \beta$ mit folgenden Eigenschaften:

$V \subseteq V'$, $E \subseteq E'$, $r \leq r'$, $z = |E| \leq |E'| = z'$,
$a'_{ik} = a_{ik}$ für alle $i \in V$, $k \in M$,
$b'_{ij} = b_{ij}$ für alle $ij \in E$,
$a'^T x + b'^T y \leq \beta$ ist facetten-definierende Ungleichung für $\mathbf{C}^{r'}(G',f',F)$.

Falls $r = r'$ und $E = E'$ gilt, so sprechen wir von **"Knoten-Lifting"**. Gilt $r = r'$, $V = V'$ und $f = f'$, so bezeichnen wir diesen Spezialfall als **"Kanten-Lifting"**. Wenn $E = E'$, $V = V'$ und $f = f'$ gilt, dann bezeichnen wir dies als **"Cluster-Lifting"**. Zunächst betrachten wir das "Kanten-Lifting".

(14.3) Satz über das simultane, triviale "Kanten-Lifting":
Sei $a^T x + b^T y \leq \beta$ eine Ungleichung, welche eine nicht triviale Facette für $\mathbf{C}^r(G,f,F)$ definiert. Ferner bezeichne E' eine Kantenmenge mit $E \subset E'$ und sei $G' = (V,E')$, wobei $z' = |E'|$ gesetzt wird. Sei $b' \in \mathbb{R}^{z'}$ der Vektor mit $b'_{ij} = b_{ij}$ für alle $ij \in E$ und $b'_{ij} = 0$ für alle $ij \in E' \setminus E$. Dann definiert $a^T x + b'^T y \leq \beta$ eine Facette für das r-Clusteringpolytop $\mathbf{C}^r(G',f,F)$.

Beweis: Die Gültigkeit der gelifteten Ungleichung ist offensichtlich. Zum Beweis, daß die Ungleichung eine Facette für das Polytop $\mathbf{C}^r(G',f,F)$ definiert, gehen wir folgendermaßen vor.

Angenommen,
$EQ(\mathbf{C}^r(G',f,F), a^T x + b'^T y \leq \beta) \subseteq EQ(\mathbf{C}^r(G',f,F), \hat{a}^T x + \hat{b}^T y \leq \hat{\beta})$, wobei die Ungleichung $\hat{a}^T x + \hat{b}^T y \leq \hat{\beta}$ eine Facette für $\mathbf{C}^r(G',f,F)$ definiert.

(1)

Sei $i_0j_0 \in E' \setminus E$ und sei $(x^0, y^0) \in EQ(\mathbf{C}^r(G,f,F), a^Tx + b^Ty \leq \beta)$ mit $x^0_{i_0k} = 0$ für alle $k \in M$. Solch ein Vektor existiert, da anderenfalls $EQ(\mathbf{C}^r(G,f,F), a^Tx + b^Ty \leq \beta) \subset EQ(\mathbf{C}^r(G,f,F), \sum_{k\in M} x_{i_0k} \leq 1)$ wäre. Dies widerspricht der Voraussetzung, obige Ungleichung ist nicht trivial.
Definieren wir $y'(i_0j_0)$ gemäß:

$$y'(i_0j_0)_{ij} = \begin{cases} y^0_{ij}, & \text{für alle } ij \in E, \\ 0, & \text{für } ij = i_0j_0, \\ 1, & \text{anderenfalls,} \end{cases}$$

so gilt: Die Vektoren $(x^0, y'(i_0j_0))$ und $(x^0, y'(i_0j_0) + e_{i_0j_0})$ sind Elemente von $EQ(\mathbf{C}^r(G',f,F), a^Tx + b'^Ty \leq \beta)$. Folglich erfüllen sie auch die Ungleichung $\hat{a}^Tx + \hat{b}^Ty \leq \hat{\beta}$ mit Gleichheit. Dies beweist $\hat{b}_{i_0j_0} = b'_{i_0j_0} = 0$ für alle $i_0j_0 \in E' \setminus E$.

(2)

Sei $(x,y) \in \mathbb{R}^{nr} \times \mathbb{R}^z$ ein Element von $EQ(\mathbf{C}^r(G,f,F), a^Tx + b^Ty \leq \beta)$. Definieren wir $y' \in \mathbb{R}^{z'}$ gemäß:

$$y'_{ij} = \begin{cases} y_{ij}, & \text{für alle } ij \in E, \\ 1, & \text{anderenfalls,} \end{cases}$$

so erfüllt der Vektor (x,y') die Gleichung $a^Tx + b'^Ty' = \beta$ und damit $\hat{a}^Tx + \hat{b}^Ty' = \hat{\beta}$. Es ist leicht einzusehen, daß der durch die Menge von Vektoren
$\bigcup\{(x,y') \mid (x,y) \in EQ(\mathbf{C}^r(G,f,F), a^Tx + b^Ty \leq \beta)\}$ definierte lineare Unterraum gleich dem durch $EQ(\mathbf{C}^r(G,f,F), a^Tx + b^Ty \leq \beta)$ definierten Unterraum ist. Da die Ungleichung $a^Tx + b^Ty \leq \beta$ eine Facette für $\mathbf{C}^r(G,f,F)$ definiert, läßt sich somit zeigen, daß $\hat{a}_{ik} = a'_{ik} = a_{ik}$ für alle $i \in V$, $k \in M$ und $\hat{b}_{ij} = b'_{ij} = b_{ij}$ für alle $ij \in E$ gilt.
Dies beweist die Behauptung.

□

Aus Satz (14.3) können wir insbesondere folgern, daß alle Facetten für das "Multiple Knapsack Polytope" (abgekürzt mit $MK(V,f,F)$) auch Facetten für die Polyeder von Instanzen des r-Clusteringproblems $\mathbf{C}^r(G,f,F)$ definieren.
(14.4) Korollar: Sei $MK(V,f,F) := conv\{x \in \mathbb{R}^{nr} \mid x \text{ erfüllt das System (14.5)}\}$.

$$\begin{aligned} &\sum_{k\in M} x_{ik} \leq 1, \text{ für alle } i \in V, \\ &\sum_{i\in V} f_i x_{ik} \leq F, \text{ für alle } k \in M, \\ &x_{ik} \in \{0,1\}, \text{ für alle } i \in V,\ k \in M, \end{aligned} \tag{14.5}$$

Dann definieren alle Facetten von $MK(V,f,F)$ auch Facetten für die Polyeder von Instanzen des r-Clusteringproblems $\mathbf{C}^r(G,f,F)$. Dies gilt für alle Kantenmengen E mit $E \subset 2^N$.

Studien zum "Multiple Knapsack Polytope" finden sich beispielsweise in [CJP83] [GR90] und [GR90a].

Als nächstes werden wir untersuchen, inwieweit Facetten für das r-Clusteringpolytop $\mathbf{C}^r(G,f,F)$ auch Facetten für das Polytop $\mathbf{C}^r(G',f',F)$ definieren ($V \subset V'$, $f' = (f_v, v \in V')$).

Leider werden wir nur eine Prozedur angeben können, welche sequentiell liftet, d.h. aus den Facetten für das Polytop einer Instanz von $\mathbf{C}^r(G,f,F)$ Facetten einer Instanz von $\mathbf{C}^r(G',f',F)$ generiert, wobei $V' = V \cup \{n+1\}$, $f' = (f_1,\ldots,f_n,f_{n+1})$ mit $f_{n+1} \in \mathbb{Q}$ ist. Diese Prozedur wurde in [P75] vorgestellt und läßt sich allgemein für 0/1-Polytope anwenden.

(14.6) Satz über das sequentielle "Knoten-Lifting":
Sei $a^Tx + b^Ty \leq \beta$ eine nicht triviale, facetten-definierende Ungleichung für $\mathbf{C}^r(G,f,F)$. Ferner sei $V' = V \cup \{n+1\}$, $f' = (f_1,\ldots,f_n,f_{n+1})$ mit $f_{n+1} \in \mathbb{Q}$ und sei $n' = n+1$. Wir setzen $G' = (V',E)$. Dann definiert $a'^Tx + b^Ty \leq \beta$ eine Facette für das r-Clusteringpolytop $\mathbf{C}^r(G',f',F)$. $a' \in \mathbb{R}^{n'r}$ ist der Vektor mit $a'_{ik} = a_{ik}$ für alle $i \in V$, $k \in M$. Für alle $k \in M$ ergibt sich der Koeffizient $a'_{n'k}$ als Lösung des Optimierungsproblems (14.7).

$$\min(\beta - \sum_{i\in V}\sum_{l\in M} a_{il}u_{il} - \sum_{ij\in E} b_{ij}v_{ij})$$

mit

(14.7)
$$\sum_{l\in M} u_{il} \leq 1, \text{ für alle } i \in V,$$
$$\sum_{i\in V} f_i u_{ik} \leq F - f_{n+1},$$
$$\sum_{i\in V} f_i u_{il} \leq F \text{ für alle } l \in M,\ l \neq k$$
$$u_{ik} + u_{jl} - v_{ij} \leq 1, \text{ für alle } (i,j) \in E,$$
$$u_{ik} \in \{0,1\}, \text{ für alle } i \in V,\ k \in M,$$
$$v_{ij} \in \{0,1\}, \text{ für alle } ij \in E.$$

Es stellt sich nun noch die Frage, ob Facetten für das r-Clusteringpolytop $\mathbf{C}^r(G,f,F)$ zu Facetten für das Polytop $\mathbf{C}^{r'}(G,f,F)$ geliftet werden können ($r' > r$). Diese Fragestellung wird uns im folgenden beschäftigen.

Gehen wir von dem einfachsten Fall aus:
Sei $a^Tx + b^Ty \leq \beta$ eine nicht triviale, facetten-definierende Ungleichung für $\mathbf{C}^r(G,f,F)$. Sei $r' = r+1$. Unser Ziel ist es, einen Vektor $a' \in \mathbb{R}^{nr'}$ zu finden, so daß $a'^Tx' + b^Ty \leq \beta$ facetten-definierende Ungleichung für $\mathbf{C}^{r'}(G,f,F)$ wird. Dabei werden

wir wiederum die allgemeine Lifting-Prozedur von Padberg (vergleiche [P75] und [Z78]) für unser Problem spezifizieren. Wir setzten ferner— wie bereits in den anderen "Lifting"-Prozeduren — voraus, daß $a'_{ik} = a_{ik}$ für alle $i \in V$ und $k \in M$ gelten soll. Damit $a'^T x' + b^T y \leq \beta'$ gültige Ungleichung für das Polytop $\mathbf{C}^{r'}(G, f, F)$ ist, müssen die Koeffizienten $a'_{ir'}$ für alle $i \in V$ nachfolgenden Bedingungen genügen.

(14.8) Lemma:
Sei $a^T x + b^T y \leq \beta$ eine nicht triviale, facetten-definierende Ungleichung für $\mathbf{C}^r(G, f, F)$. Ferner sei $r' = r+1$. Dann definiert $a'^T x + b^T y \leq \beta$ eine gültige Ungleichung, wenn gilt:

$$(i)\ a'_{ik} = a_{ik} \text{ für alle } i \in V,\ k \in M,$$

$$(ii)\ \text{Alle } B \subseteq V \text{ mit } \sum_{i \in B} f_i \leq F \text{ erfüllen die Bedingung:}$$

$$\sum_{i \in B} a'_{ir'} \leq (\beta - \sum_{j \in V \setminus B} \sum_{k \in M} a_{jk} u_{jk} - \sum_{ij \in E} b_{ij} v_{ij})$$

$$\text{für alle Lösungen } (u, v) \in \mathbb{R}^{nr'} \times \mathbb{R}^z \text{ von (14.9).}$$

Das System (14.9) ist folgendermaßen definiert:

$$\begin{aligned}
&u_{ir'} = 1 \text{ für alle } i \in B,\\
&u_{ir'} = 0 \text{ für alle } i \in V \setminus B,\\
&\sum_{k \in M} u_{jk} \leq 1 \text{ für alle } j \in V,\\
&\sum_{j \in V} f_j u_{jk} \leq F \text{ für alle } k = 1, \ldots, r', \qquad (14.9)\\
&u_{ik} + u_{jl} - v_{ij} \leq 1, \text{ für alle } ij \in E,\ k, l = 1, \ldots, r',\ k \neq l,\\
&u_{jk} \in \{0, 1\}, \text{ für alle } j \in V,\ k \in M,\\
&v_{ij} \in \{0, 1\}, \text{ für alle } ij \in E.
\end{aligned}$$

Beweis:
Sei $(x', y) \in \mathbf{C}^{r'}(G, f, F)$. Mit B bezeichnen wir die maximale Teilmenge von V mit $x'_{ir'} = 1$ für alle $i \in B$. Dann ist (x', y) eine zulässige Lösung von (14.9). Gemäß Voraussetzung in Lemma (14.8) gilt somit: $\sum_{i \in B} a'_{ir'} \leq (\beta - \sum_{j \in V \setminus B} \sum_{k \in M} a_{jk} x'_{jk} - \sum_{ij \in E} b_{ij} y_{ij})$.
Daraus läßt sich folgern:
$a'^T x' + b^T y = \sum_{i \in B} a'_{ir'} + \sum_{j \in V \setminus B} \sum_{k \in M} a_{jk} x'_{jk} - \sum_{ij \in E} b_{ij} y_{ij} \leq$
$(\beta - \sum_{j \in V \setminus B} \sum_{k \in M} a_{jk} x'_{jk} - \sum_{ij \in E} b_{ij} y_{ij}) + \sum_{j \in V \setminus B} \sum_{k \in M} a_{jk} x'_{jk} - \sum_{ij \in E} b_{ij} y_{ij} = \beta.$
Dies beweist die Gültigkeit der Ungleichung.

□

Um zeigen zu können, daß $a'^T x + b^T y \leq \beta$ eine Facette definiert, müssen noch zusätzliche Bedingungen an die Koeffizienten $a'_{ir'}$, $i \in V$ gestellt werden. Dies wird in Lemma (14.10) abgehandelt.

(14.10) Lemma:
Sei $a^Tx + b^Ty \leq \beta$ eine nicht triviale, facetten-definierende Ungleichung für $\mathbf{C}^r(G,f,F)$. Ferner sei $r' = r+1$. Dann definiert $a'^Tx + b^Ty \leq \beta$ eine Facette für $\mathbf{C}^{r'}(G,f,F)$, wenn gilt:

(i) *Lemma (14.8) ist erfüllt.*

(ii) *Für alle $c \in V$ existiert eine nicht-leere Menge $B^*(c) \subseteq V$ und ein Vektor $(u(c),v(c)) \in \mathbb{R}^{nr'} \times \mathbb{R}^z$, so daß System (14.11) erfüllt ist.*

(iii) *Die Menge von Vektoren $\bigcup_{c\in V}\{(u(c),v(c))\}$ ist linear unabhängig.*

Das System (14.11) *ist folgendermaßen definiert:*

$$\sum_{i\in B^*(c)} a'_{ir'} = \beta - \sum_{j\in V\setminus B}\sum_{k\in M} a_{jk}u(c)_{jk} - \sum_{ij\in E} b_{ij}v(c)_{ij}$$

mit

$$\begin{aligned}
&u(c)_{ir'} = 1 \text{ für alle } i \in B^*(c),\\
&u(c)_{ir'} = 0 \text{ für alle } i \in V \setminus B^*(c),\\
&\sum_{k\in M} u(c)_{jk} \leq 1 \text{ für alle } j \in V,\\
&\sum_{j\in V} f_j u(c)_{jk} \leq F \text{ für alle } k = 1,\ldots,r',\\
&u(c)_{ik} + u(c)_{jl} - v(c)_{ij} \leq 1, \text{ für alle } ij \in E,\ k,l = 1,\ldots,r',\ k \neq l,\\
&u(c)_{jk} \in \{0,1\}, \text{ für alle } j \in V,\ k \in M,\\
&v(c)_{ij} \in \{0,1\}, \text{ für alle } ij \in E.
\end{aligned} \tag{14.11}$$

Beweis:
Die Gültigkeit der Ungleichung $a'^Tx + b^Ty \leq \beta$ wird durch Bedingung (i) sichergestellt. Betrachten wir zunächst die Menge von Vektoren $\bigcup_{c\in V}\{(u(c),v(c))\}$. Für alle $c \in V$ gilt: $a'^T(u(c)) + b^Tv(c) = \sum_{i\in B^*(c)} a'_{ir'} + \sum_{j\in V\setminus B^*(c)}\sum_{k\in M} a_{jk}u(c)_{jk} - \sum_{ij\in E} b_{ij}v(c)_{ij}$
$= (\beta - \sum_{j\in V\setminus B^*(c)}\sum_{k\in M} a_{jk}u(c)_{jk} - \sum_{ij\in E} b_{ij}v(c)_{ij}) + \sum_{j\in V\setminus B^*(c)}\sum_{k\in M} a_{jk}u(c)_{jk} - \sum_{ij\in E} b_{ij}v(c)_{ij} = \beta$. Das bedeutet, daß die Menge $\bigcup_{c\in V}\{(u(c),v(c))\}$ in der Seitenfläche $EQ(\mathbf{C}^{r'}(G,f,F), a'^Tx' + b^Ty \leq \beta)$ enthalten ist.
Desweiteren ist es offensichtlich, daß für alle $(x,y) \in EQ(\mathbf{C}^r(G,f,F), a^Tx + b^Ty \leq \beta)$ der Vektor (x',y) mit $x'_{ik} = x_{ik}$ für alle $i \in V$, $k \in M$ und $x'_{n'k} = 0$ für alle $k \in M$ obige Ungleichung mit Gleicheit erfüllt. Bezeichne Q die Vereinigung all dieser Vektoren, d.h. $Q = \bigcup\{(x',y) \mid (x,y) \in EQ(\mathbf{C}^r(G,f,F), a^Tx + b^Ty \leq \beta)\}$. Dann ist leicht einsehbar,

daß die Menge von Vektoren $\bigcup_{c \in V}\{u(c), v(c))\} \cup Q$ eine Menge von affin unabhängigen Vektoren der Dimension $dim(\mathbf{C}^{r'}(G, f, F)$ ist. Daraus folgt die Behauptung.

□

Es ist offensichtlich, daß Lemma (14.10) ein rein theoretisches Resultat bezüglich des "Cluster-Liftings" darstellt. Gegeben sei eine facetten-definierende Ungleichung $a^T x + b^T y \leq \beta$ für das Polytop $\mathbf{C}^r(G, f, F)$. Sei $r' = r' + 1$. Damit die (geliftete) Ungleichung $a'^T x + b^T y \leq \beta$, $a'_{ik} = a_{ik}$, $i \in V$, $k \in M$ gültige Ungleichung für das Polytop $\mathbf{C}^r(G, f, F)$ ist, müssen exponentiell viele Systeme vom Typ (14.8) ausgewertet werden. Algorithmisch kann daher dieses "Lifting"-Resultat wohl nicht umgesetzt werden.
Spezielle Facetten eines Polytops $\mathbf{C}^r(G, f, F)$ lassen sich jedoch sehr einfach zu Facetten für ein Polytop $\mathbf{C}^{r'}(G, f, F)$ liften ($r' > r$).

(14.12) Bemerkung: Sei $r' > r$. Sei $a^T x \leq \beta$ eine nicht triviale, facetten-definierende Ungleichung für $\mathbf{C}^r(G, f, F)$. Dann definiert die Ungleichung $a'^T x \leq \beta$ eine Facette für das Polytop $\mathbf{C}^{r'}(G, f, F)$. Dabei gilt: $a'_{ik} = a_{ik}$ für alle $i \in V$, $k \in M$ und $a'_{ir'} = 0$ für alle $i \in V$.

Beweis:
Offensichtlich ist die geliftete Ungleichung gültig.

Für alle $i \in V$ existiert ein Vektor $(x, y) \in EQ(\mathbf{C}^r(G, f, F), a^T x \leq \beta)$, welcher $x_{ik} = 0$ erfüllt für alle $k \in M$. Anderenfalls ist obige Facette trivial. Wir definieren den Vektor $x'(i) \in \mathbb{R}^{nr'}$ gemäß:
$x'(i)_{ik} = x_{ik}$ für alle $i \in V$, $k \in M$. Ferner setzen wir $x'(i)_{ir'} = 0$ für alle $i \in V$. Damit sind die beiden Vektoren $(x'(i), \mathbf{1})$ und $(x'(i) + e_{ir'}, \mathbf{1})$ Elemente von $EQ(\mathbf{C}^r(G, f, F), a^T x \leq \beta)$. Ferner ist offensichtlich, daß die Menge $\cup_{i \in V}\{(x(i), y(i))\} \cup EQ(\mathbf{C}^r(G, f, F), a^T x \leq \beta)$ ein affin unabhängiges System der Dimension $dim(\mathbf{C}^r(G, f, F))$ bildet.
Daraus folgt die Behauptung. □
Für die Klassen von sogenannten kombinierten Knoten-Kanten-Facetten, welche wir in dem nächsten Kapitel vorstellen, soll das "Lifting" von einem Polytop $\mathbf{C}^r(G, f, F)$ zu einem Polytop $\mathbf{C}^{r'}(G, f, F)$ speziell untersucht werden.

Kapitel 15 Kombinierte Knoten-Kanten-Facetten

Das mit dem r-Clusteringproblem assoziierte Modell besteht aus einer Kombination von Knotenvariablen der Form x_{ik} und Kantenvariablen y_{ij}. Deshalb ist zu erwarten, daß zur Beschreibung des r-Clusteringpolytops eine Vielzahl von Ungleichungen der Form $a^T x + b^T y \leq \beta$ erforderlich sind, wobei $a \neq 0$ und $b \neq 0$ gilt. Ungleichungen dieser Art bezeichnen wir als kombinierte Knoten-Kanten-Ungleichungen. In diesem Kapitel wollen wir einige solcher Klassen von Facetten für das r-Clusteringpolytop vorstellen.

Für die Darstellung dieses Kapitels greifen wir auf die Notation aus Kapitel 10 zurück. Insbesondere bezeichnet $G = (V, E)$ den zugrundeliegenden Graphen und M die Indexmenge für die Cluster. Es gelte ferner $n = |V|$, $r = |M|$ und $z = |E|$.

Definition:

(i) Gegeben sei ein Cluster $k \in M$. Sei $S \subseteq V$ eine Menge von Knoten mit der Eigenschaft:

$$\sum_{i \in S} f_i > F,$$

dann nennen wir S eine **Überdeckung** von k.
Erfüllt S zusätzlich die Bedingungen

$$\sum_{i \in S \setminus \{s\}} f_i \leq F \text{ für alle } s \in S,$$

so ist S eine **minimale Überdeckung** von k.

(ii) Gegeben sei $k \in M$, eine Menge $Q \subseteq V$ und ein Element $s \in V \setminus Q$. Falls gilt:

$$\sum_{i \in Q} f_i \leq F,$$
$$\sum_{i \in Q'} f_i + f_s > F, \text{ für alle } Q' \subseteq Q,\ |Q'| = d,$$
$$\sum_{i \in Q'} f_i + f_s \leq F, \text{ für alle } Q' \subseteq Q,\ |Q'| = d - 1,$$

so nennen wir $Q \cup \{s\}$ eine $(1, d)$ **Konfiguration** bezüglich k.

Bemerkung:

(i) Sei $Q \cup \{s\}$ eine $(1, d)$ Konfiguration bezüglich k. Dann muß s zusammen mit allen d-elementigen Teilmengen von Q eine minimale Überdeckung von k definieren.

(ii) In [B75], [HJP75] und [W75] wurde gezeigt, daß die Ungleichung $\sum_{i \in S} x_{ik} \leq |S| - 1$ eine Facette für das "single knapsack" Polytop $SK(V, f, F)$ definiert. Dabei ist S eine minimale Überdeckung bezüglich $k = 1$ und $V = S$.

(iii) Sei $Q \cup \{s\} = \{1, \ldots, n+1\}$ eine (1,d)-Konfiguration bezüglich $k = 1$, dann definiert die Ungleichung $\sum_{u \in Q} x_{uk} + (|Q| - d + 1)x_{sk} \leq |Q|$ eine Facette für das Polyeder $SK(V, f, F)$, wobei $V = \{1, \ldots, n+1\}$ und $f = \{f_1, \ldots, f_{n+1}\}$ vorausgesetzt wird. Diese Klasse von Ungleichungen geht auf [P80] zurück.

In den vorhergehenden Kapiteln haben wir bereits gesehen, wie wir solche Überdeckungs- bzw. $(1, d)$-Konfigurations-Ungleichungen liften können. Interessant ist jedoch die Frage, ob und gegebenenfalls wie wir solche Klassen von Ungleichungen modifizieren können, so daß Kantenvariablen in die linke Seite der Ungleichung mit negativen Koeffizienten eingehen, und diese modifizierte Ungleichung eine Facette für das r-Clusteringpolyeder bestimmter Instanzen definiert. Dieses Thema wird uns im folgenden beschäftigen.
Die Bezeichnung der Ungleichungen, auf welche wir im folgenden näher eingehen werden, sollen dabei zum Ausdruck bringen, wie die jeweilige Struktur der Knoten und Kanten in dem zugehörigen Graphen ist. In den Abschnitten 15.1 bis 15.6 werden wir jeweils eine Klasse von facetten-definierenden Ungleichungen vorstellen. Dabei werden wir stets zunächst den mathematischen Satz formulieren und anschließend die betreffende Ungleichung veranschaulichen bzw erläutern.

15.1 Ungleichungen aus einer Überdeckung und disjunkten Sternen

(15.1.1) Satz: *Gegeben sei eine Instanz des r-Clusteringproblems mit den folgenden Eigenschaften:*
Sei $M = \{1, \ldots, r\}$ und $k \in M$. Ferner definiere $V = \{1, \ldots, n\}$ eine minimale Überdeckung von k. Die Kantenmenge E des Graphen $G = (V, E)$ sei die Vereinigung von disjunkten Sternen, d.h. es existieren eine natürliche Zahl $b \leq n$ und Knoten i_v, $v = 1, \ldots, b$, $i_{v_1} \neq i_{v_2}$ für $v_1 \neq v_2$, so daß $E = \cup_{v=1}^{b} \delta(i_v)$ ist, und es gelte $V(\delta(i_{v_1})) \cap V(\delta(i_{v_2})) = \emptyset$ für alle $v_1, v_2 \in \{1, \ldots, b\}$, $v_1 \neq v_2$. Bezeichne $c_v = |\delta(i_v)|$ für $v = 1, \ldots, b$, und es gelte die Ungleichung $\sum_{v=1}^{b} c_v \leq |V| - 1$. Falls zusätzlich die Bedingungen

$$f_{i_v} + f_p \leq F \text{ für alle } v = 1, \ldots, b \text{ und } i_v p \in \delta(i_v)$$

erfüllt sind, dann definiert die Ungleichung

$$\sum_{u \in V} x_{uk} + \sum_{l \in M \setminus \{k\}} \sum_{v=1}^{b} c_v x_{i_v l} - \sum_{ij \in E} y_{ij} \leq |V| - 1. \tag{15.1.2}$$

eine Facette für das Polyeder $\mathbf{C}^r(G, f, F)$.

Bemerkung:
Anschaulich läßt sich die Ungleichung folgendermaßen interpretieren:
Die Knotenmenge V entspricht einer minimalen Überdeckung bezüglich eines Clusters k, d.h. höchstens $|V| - 1$ Elemente von V können in Cluster k zusammengefaßt werden. Falls

ein Knoten i_v einem anderen Cluster l zugewiesen wird, so werden in der Ungleichung alle Kanten gezählt, die inzident zu i_v und einem Knoten aus Cluster k sind.
Die folgende Abbildung stellt einen Graphen für $|V| = 10$ und $b = 3$ dar.

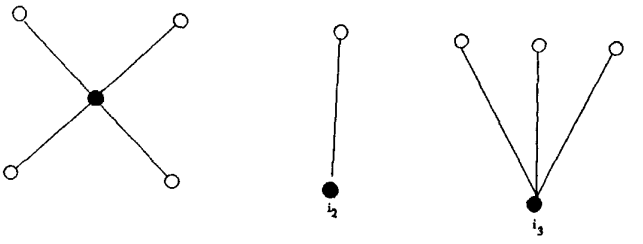


Abbildung 15.1.3

Beweis:
Wir beweisen zunächst die Gültigkeit der obigen Ungleichung.

Sei $(x, y) \in \mathbf{C}^r(G, f, F)$. Dann sind folgende Ungleichungen gültig.

$$(i) \sum_{u \in V} x_{uk} \leq |V| - 1,$$

$$(ii) \sum_{l \in M \setminus \{k\}} c_v x_{i_v l} \leq c_v \text{ für alle } v = 1, \ldots, b.$$

Wenn $\sum_{u \in V} x_{uk} + \sum_{l \in M \setminus \{k\}} \sum_{v=1}^{b} c_v x_{i_v l} \leq |V| - 1$, dann ist obige Ungleichung sicherlich gültig. Anderenfalls sei $W \subseteq \{1, \ldots, b\}$ mit der Eigenschaft:

Für alle $v \in W$ existiert eine Zahl $l_{i_v} \in M \setminus \{k\}$, so daß $x_{i_v l_{i_v}} = 1$ gilt.
Da $\sum_{u \in V} x_{uk} + \sum_{l \in M \setminus \{k\}} \sum_{v=1}^{b} c_v x_{i_v l} > |V| - 1$, folgt zunächst $|W| > 1$.
Sei nun $P_w = \{p \in V \mid i_w p \in \delta(i_w) \text{ und } x_{pk} = 1\}$ für alle $w \in W$. Dann folgt, daß die Variable $y_{i_w p}$ den Wert 1 annehmen muß. Dies gilt für alle $w \in W$, $p \in P_w$. Daraus ergibt sich:

$$\sum_{u \in V} x_{uk} \leq |V| - |W| - \sum_{w \in W} (c_w - |P_w|),$$

$$\sum_{l \in M \setminus \{k\}} \sum_{v=1}^{b} c_v x_{i_v l} \leq \sum_{w \in W} c_w,$$

$$\sum_{ij \in E} y_{ij} \geq \sum_{w \in W} |P_w|.$$

Durch Summieren erhält man:

$$\sum_{u\in V} x_{uk} + \sum_{l\in M\setminus\{k\}} \sum_{v=1}^{b} c_v x_{i_v l} - \sum_{ij\in E} y_{ij} \leq$$
$$|V| - |W| - \sum_{w\in W} (c_w - |P_w|) + \sum_{w\in W} c_w - \sum_{w\in W} |P_w| =$$
$$|V| - |W| \leq |V| - 1.$$

Im folgenden Teil beweisen wir, daß (15.1.2) eine Facette definiert.

Angenommen,
$EQ((15.1.2)) \subseteq EQ(a^T x + b^T y \leq \beta)$, wobei die Ungleichung $a^T x + b^T y \leq \beta$ eine Facette für $\mathbf{C}^r(G, f, F)$ definiert.

(1)

Sei $s \in V$. Dann sind die Vektoren $(\sum_{u\in V\setminus\{s\}} e_{uk}, 0)$ Elemente der Menge $EQ((15.1.2))$ und damit in $EQ(a^T x + b^T y \leq \beta)$ enthalten.
Dies liefert die Bedingungen $a_{sk} = a_{s'k}$ für alle $s \neq s'$, $s, s' \in V$.

(2)

Sei nun $v \in \{1, \ldots, b\}$, $l \in M \setminus \{k\}$. Dann sind die Lösungen
$(\sum_{u\in V\setminus\{i_v\}} e_{uk} + e_{i_v l}, \sum_{ij\in\delta(i_v)} e_{ij})$ in $EQ((15.1.2))$ enthalten. Dies impliziert:

$$a_{i_v l} = -\sum_{ij\in\delta(i_v)} b_{ij}.$$

(3)

Für alle $pi_v \in \delta(i_v)$ erfüllen die Vektoren
$(\sum_{u\in V\setminus\{i_v,p\}} e_{uk} + e_{i_v l}, \sum_{ij\in\delta(i_v)\setminus\{i_v p\}} e_{ij})$ und $(\sum_{u\in V\setminus\{i_v\}} e_{uk} + e_{i_v l}, \sum_{ij\in\delta(i_v)} e_{ij})$ die Ungleichung (15.1.2) mit Gleichheit. Zusammen mit (2) ergibt sich dann $a_{pk} = -b_{i_v,p}$.

(4)

Da in (2) bzw. (3) v und $pi_v \in E$ beliebig waren, ergibt sich zusammen mit (1):

$$b_{ij} = -a_{uk},\ a_{i_v l} = -c_v a_{uk} \text{ für alle } ij \in E,\ v = 1, \ldots, b \text{ und } u \in V.$$

(5)

Sei nun $s \in V$ mit $i_v, s \notin E$ für alle $v = 1, \ldots, b$. Dann sind die beiden Vektoren $(\sum_{u\in V\setminus\{s\}}, 0)$ und $(\sum_{u\in V\setminus\{s\}} e_{uk} + e_{sl}, 0)$ in $EQ((15.1.2))$ enthalten. Das gilt für alle $l \in M$, $l \neq k$. Somit können wir folgern, daß $a_{sl} = 0$ ist für alle $l \in M \setminus \{k\}$ und $s \in V$ mit $\delta(s) = \emptyset$.

(6)
Gegeben sei $v \in \{1,\ldots,b\}$ und ein $p \in V$ mit $pi_v \in \delta(i_v)$. Die Vektoren

$$(\sum_{u \in V \setminus \{p,i_v\}} e_{uk} + e_{i_v l}, \sum_{ij \in \delta(i_v) \setminus \{i_v p\}} e_{ij}) \text{ und}$$
$$(\sum_{u \in V \setminus \{i_v,p\}} e_{uk} + e_{i_v l} + e_{pl}, \sum_{ij \in \delta(i_v) \setminus \{i_v p\}} e_{ij})$$

erfüllen Ungleichung (15.1.2) mit Gleichheit. Dies liefert schließlich $a_{pl} = 0$ für alle $l \in M$, $l \neq k$.
Dies schließt den Beweis ab. □

15.2 Ungleichungen aus einer Überdeckung und einem Baum

(15.2.1) Satz:
Gegeben sei eine Instanz des r-Clusteringproblems mit den folgenden Eigenschaften:
Sei $M = \{1,\ldots,r\}$, $k_0 \in M$ und V eine minimale Überdeckung von k_0. Sei ferner $D \subseteq V$, $|D| \geq 2$ und $d \in V \setminus D$. Die Kantenmenge E des zugehörigen Graphen $G = (V,E)$ definiere einen aufspannenden Baum bezüglich $D \cup \{d\}$. Zusätzlich muß gelten, daß für alle $k \in M \setminus \{k_0\}$ die Ungleichung $\sum_{i \in D} f_i + f_d \leq F$ erfüllt ist. Dann definiert die Ungleichung

$$(15.2.2) \quad \sum_{i \in V} x_{ik_0} + \sum_{k \in M \setminus \{k_0\}} \sum_{i \in D} x_{ik} - \sum_{ij \in E} y_{ij} \leq |V| - 1.$$

eine Facette für das Polyeder $\mathbf{C}^r(G,f,F)$.

Bemerkung:
Anschaulich läßt sich Ungleichung (15.2.2) folgendermaßen interpretieren:
Die Knotenmenge V entspricht einer minimalen Überdeckung bezüglich eines Clusters k_0, d.h. höchstens $|V|-1$ Elemente von S können in Cluster k zusammengefaßt werden. Eine Teilmenge D und ein weiteres Element d von V sind in dem Graphen über einen Baum verbunden. Falls eine Teilmenge von D einem anderen Cluster l zugewiesen wird, so werden in der Ungleichung die Kanten aus dem zugehörigen Schnitt gezählt.
Die folgende Abbildung stellt den Graphen für ein Beispiel mit $|V| = 12$ und $|D| = 7$ dar. Die Knoten aus $V \setminus (D \cup \{d\})$ sind dick schwarz gezeichnet. Die restlichen Knoten gehören der Menge $D \cup \{d\}$ an. Letztere ist durch einen aufspannenden Baum verbunden.

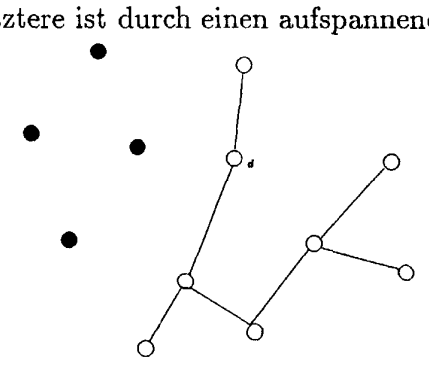


Abbildung 15.2.3

Beweis:
Wir beweisen zunächst die Gültigkeit der Ungleichung.
Da D eine Teilmenge von V ist, gilt für alle $(x,y) \in \mathbf{C}^r(G,f,F)$ die Ungleichung:

$$\sum_{i \in V} x_{ik_0} + \sum_{k \in M \setminus \{k_0\}} \sum_{i \in D} x_{ik} \leq |V|.$$

Sei nun $(x,y) \in \mathbf{C}^r(G,f,F)$ ein Vektor, für den gilt:
$\sum_{i \in V} x_{ik_0} + \sum_{k \in M \setminus \{k_0\}} \sum_{i \in D} x_{ik} = |V|$ (anderenfalls ist die Ungleichung trivialerweise erfüllt).
Dies impliziert, daß

$$x_{ik_0} = 1 \text{ für alle } i \in V \setminus D \text{ und}$$
$$\sum_{i \in D} \sum_{k \in M} x_{ik} = |D|.$$

Da V minimale Überdeckung von k_0 ist, muß $\sum_{i \in D} x_{ik_0} < |D|$ sein. Ferner gilt $x_{dk_0} = 1$. Daraus können wir ableiten, daß mindestens zwei Teilmengen $D_1, D_2 \subseteq D \cup \{d\}$, $D \cup \{d\} = D^1 \cup D^2$, $D^1 \cap D^2 = \emptyset$ und Zahlen $k, l \in M$, $k \neq l$ existieren, so daß $x_{ik} = 1$ für alle $i \in D^1$ und $x_{il} = 1$ für alle $i \in D^2$ gilt.
Da der Graph $G = (D,E)$ einen aufspannenden Baum bezüglich $D \cup \{d\}$ definiert, existiert mindestens eine Kante e, welches im Schnitt $[D_1 : D_2]$ liegt. Somit gilt $\sum_{i \in V} x_{ik_0} + \sum_{i \in D} \sum_{k \in M \setminus \{k_0\}} x_{ik} - \sum_{ij \in E} y_{ij} \leq |V| - y_e = |V| - 1$. Dies beweist die Gültigkeit der Ungleichung.

Obige Ungleichung definiert eine Facette für das Polyeder $\mathbf{C}^r(G,f,F)$.

Nehmen wir wiederum an,
$EQ((15.2.2)) \subseteq EQ(a^T x + b^T y \leq \beta)$, wobei die Ungleichung $a^T x + b^T y \leq \beta$ eine Facette für $\mathbf{C}^r(G,f,F)$ definiert.

(1)

Sei $s \in V$. Dann sind die Vektoren $(\sum_{u \in V \setminus \{s\}} e_{uk_0}, 0)$ Elemente der Menge $EQ((15.2.2))$ und damit in $EQ(a^T x + b^T y \leq \beta)$ enthalten.
Dies liefert die Bedingungen $a_{sk_0} = a_{s'k_0}$ für alle $s \neq s'$, $s, s' \in V$.

(2)

Sei $ij \in E$ und $l \in M \setminus \{k_0\}$. Da der Graph einen Baum in $D \cup \{d\}$ definiert, existiert eine eindeutige Zerlegung von $D \cup \{d\}$ in zwei Mengen D_1 und D_2, so daß der Schnitt zwischen D_1 und D_2 aus genau der Kante ij besteht. Ohne Beschränkung der Allgemeinheit können wir $d \in D_1$, $i \in D_1$ und $j \in D_2$ voraussetzen.
Damit erfüllen die beiden Vektoren $v_1 := (\sum_{u \in (V \setminus D) \cup D_1} e_{uk_0} + \sum_{u \in D_2} e_{ul}, e_{ij})$ und $v_2 := (\sum_{u \in (V \setminus D) \cup (D_1 \setminus \{i\})} e_{uk_0} + \sum_{u \in D_2} e_{ul}, 0)$ Ungleichung (15.2.2) mit Gleichheit. Daher gilt $(a^T, b^T)v_1 = (a^T, b^T)v_2$. Dies liefert die Beziehung:

$$b_{ij} = -a_{ik_0}.$$

Da $ij \in E$ beliebig wählbar ist, haben wir zusammen mit (1) das Resultat:

$$a_{sk_0} = a_{s'k_0} = -b_{ij} \text{ für } s \neq s',\ s, s' \in V,\ ij \in E.$$

(3)

Es bleibt noch zu zeigen, daß $a_{il} = a_{ik_0}$ für alle $i \in D$ gilt.
Sei $i \in D$ beliebig gewählt. Dann existiert ein eindeutiger Weg in dem Graphen von i nach d. Bezeichne $W = (i, t_0, j, t_1, \ldots, d)$ diesen Weg. Ferner gilt, daß die Kante t_0 die Menge $D \cup \{d\}$ in zwei Mengen D_1 und D_2 zerlegt, so daß der Schnitt zwischen D_1 und D_2 aus genau der Kante t_0 besteht. Ohne Beschränkung der Allgemeinheit können wir $i \in D_1$ voraussetzen. Daraus folgt: $d \in D_2$.
Dann aber erfüllen die Lösungen $v_1 = (\sum_{u \in (V \setminus D) \cup D_2} e_{uk_0} + \sum_{u \in D_1} e_{ul}, e_{t_0})$ und $v_2 = (\sum_{u \in (V \setminus D) \cup D_2} e_{uk_0} + \sum_{u \in D_1 \setminus \{i\}} e_{ul}, 0)$ Ungleichung (15.2.2) mit Gleichheit. Daher gilt $(a^T, b^T)v_1 = (a^T, b^T)v_2$. Dies liefert die Bedingungen

$$b_{ij} = -a_{il}$$

für alle $l \in M,\ l \neq k_0$ und $i \in D$. Zusammen mit (3) folgt: $a_{il} = a_{jk_0} = -b_{ij}$ für alle $i \in V,\ j \in D,\ ij \in E,\ l \in M \setminus \{k_0\}$.

(4)

Da für alle $i \in (V \setminus D) \setminus \{d\}$ die Vektoren $(\sum_{u \in V \setminus \{i\}} e_{uk_0}, 0)$ und $(\sum_{u \in V \setminus \{i\}} e_{uk_0} + e_{il}, 0)$ $(l \in M \setminus \{k_0\})$ Elemente der Menge $EQ((15.2.2))$ sind, sind sie auch in $EQ(a^T x + b^T y \leq \beta)$ enthalten. Dies liefert die Bedingungen $a_{il} = 0$ für alle $i \in (V \setminus D) \setminus \{d\},\ l \in M \setminus \{k_0\}$.

(5)

Es bleibt noch zu zeigen, daß $a_{dl} = 0$ ist für alle $l \in M \setminus \{k_0\}$. Sei $l \in M \setminus \{k_0\}$ und $v_1 = (\sum_{u \in (V \setminus D) \setminus \{d\}} e_{uk_0} + \sum_{i \in D} e_{il}, 0)$. Wir definieren den Vektor v_2 als $v_2 = v_1 + (e_{dl}, 0)$. Beide Vektoren gehören der durch (15.2.2) definierten Seitenfläche an. Somit können wir den Koeffizienten a_{dl} auf den Wert 0 fixieren.

Dies beweist Satz (15.2.2).

□

15.3 Ungleichungen aus einer (1,d)-Konfiguration und einem Stern

(15.3.1) Satz: *Gegeben sei eine Instanz des r-Clusteringproblems mit den folgenden Eigenschaften:*
Sei $M = \{1, \ldots, r\}$ und sei $k \in M$. Sei ferner $Q = \{1, \ldots, n\}, s = n+1$ und $V = Q \cup \{s\}$. Es definiere $Q \cup \{s\}$ eine (1,d)-Konfiguration bezüglich k. Die Kantenmenge E des Graphen $G = (V, E)$ sei ein Stern, wobei alle Kanten inzident zu s sind, d.h. $E = \delta(s)$. Es

gelte $z = |E|$. *Schließlich bezeichne* I_Q *die Menge der zu* s *in dem Graphen* $G = (V, E)$ *inzidenten Knoten, d.h.* $I_Q = V(E) \setminus \{s\}$.

(i)

Falls für alle $i \in Q$ $f_s + f_i \leq F$ *gilt, so definiert die Ungleichung*

$$(15.3.2) \quad \sum_{u \in Q} x_{uk} + (|Q| - d + 1)x_{sk} + \sum_{l \in M \setminus \{k\}} z x_{sl} - \sum_{ij \in E} y_{ij} \leq |Q|$$

eine Facette für das Polyeder $\mathbf{C}^r(G, f, F)$.

(ii) Falls $z + d - 1 \leq |Q|$ *ist, und* $f_s + \sum_{i \in I_Q} f_i \leq F$ *gilt, definiert die Ungleichung*

$$(15.3.3) \quad \sum_{u \in Q} x_{uk} + (|Q| - d + 1)x_{sk} + \sum_{l \in M \setminus \{k\}} \sum_{i \in I_Q} x_{il} - \sum_{ij \in E} y_{ij} \leq |Q|.$$

eine Facette für das Polyeder $\mathbf{C}^r(G, f, F)$.

Bemerkung:

Anschaulich lassen sich die Ungleichungen folgendermaßen interpretieren:
Gegeben sei eine Menge $Q \cup \{s\}$, die eine $(1, d)$ Konfiguration bezüglich eines Clusters k definiert. Daraus leitet sich die Ungleichung

$$\sum_{u \in Q} x_{uk} + (|Q| - d + 1)x_{sk} \leq |Q|$$

ab. Gegeben ist ferner ein Stern in dem Graphen, wobei jede Kante inzident zu Knoten s und je einem Knoten aus $I_Q \subseteq Q$ ist. Falls s einem Cluster $l \neq k$ zugewiesen wird, so werden in der Ungleichung (15.3.2) alle Kanten gezählt, die inzident zu einem Knoten aus I_Q sind, sofern dieser dem Cluster k zugewiesen wurde. Umgekehrt wird in Ungleichung (15.3.3) für jeden Knoten aus I_Q, der einem Cluster l zugewiesen wurde, eine Kante gezählt, sofern s in Cluster k zusammengefaßt ist.
Die folgende Abbildung stellt den Graphen für ein Beispiel mit $|Q| = 7$, $|I_Q| = 5$ dar. Die Knoten aus Q sind schwarz gezeichnet.

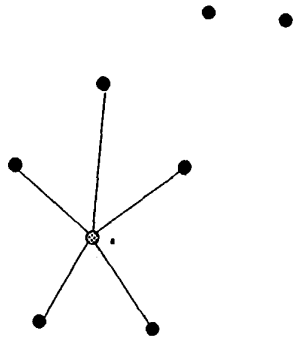


Abbildung 15.3.4

Beweis:
Wir beweisen zunächst die Gültigkeit der beiden Ungleichungen.

(i)

Sei $(x,y) \in \mathbf{C}^r(G,f,F)$. Da die Ungleichung $\sum_{u\in Q} x'_{uk} + (|Q|-d+1)x'_{sk} \leq |Q|$ eine Facette für das Polyeder $SK(V,f,F)$ definiert [P80], gilt:
$\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} \leq |Q|$. Da $\sum_{l\in M\setminus\{k\}} x_{sl} \leq 1$ ist, gilt ferner, daß

$$\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} + \sum_{l\in M\setminus\{k\}} z x_{sl} \leq |Q| + z.$$

Bezeichne $\Gamma_Q = \{i \in Q \mid x_{ik} = 1\}$. Angenommen (x,y) verletzt Ungleichung (15.3.2), d.h. $\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} + \sum_{l\in M\setminus\{k\}} z x_{sl} > |Q|$. Dies impliziert, daß $x_{sl} = 1$ für ein $l \in M\setminus\{k\}$ und $|\Gamma_Q| \geq |Q|-z+1$ ist. Sei $|\Gamma_Q| = |Q|-z+\gamma$ für ein $\gamma \geq 1$. Dann aber existieren Knoten $i_1,\ldots,i_\gamma$, so daß $si_v \in E$ und $i_v \in \Gamma_Q$ für alle $v = 1,\ldots,\gamma$ ist. Somit gilt:
$\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} + \sum_{l\in M\setminus\{k\}} z x_{sl} - \sum_{ij\in E} y_{ij} \leq |\Gamma_Q| + z$
$- \sum_{si_v\in E,\, 1\leq v\leq\gamma} y_{ij} = |Q| - z + \gamma + z - \gamma = |Q|$.

(ii)

Der Beweis der Gültigkeit von Ungleichung (15.3.3) geht ähnlich zu dem Fall (i).

Sei $(x,y) \in \mathbf{C}^r(G,f,F)$. Da die Ungleichung $\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} \leq |Q|$ eine Facette für das Polyeder $\mathbf{C}((V,\emptyset),f,F)$ definiert, gilt:
$\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} \leq |Q|$. Ferner gilt nach Voraussetzung, daß

$$\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} + \sum_{l\in M\setminus\{k\}} \sum_{i\in Q} x_{il} \leq |Q| + z.$$

Bezeichne Γ_Q die Menge: $\Gamma_Q = \{i \in Q \mid x_{il} = 1 \text{ für ein } l \in M\setminus\{k\}\}$.
Angenommen (x,y) verletzt Ungleichung (15.3.3), d.h. $\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} +$
$\sum_{l\in M\setminus\{k\}} \sum_{i\in Q} x_{il} - \sum_{ij\in E} y_{ij} > |Q|$. Dann folgt: $x_{sk} = 1$. Ferner ist $|\Gamma_Q| \geq 1$. Sei $|\Gamma_Q| = \gamma$ für ein $\gamma \geq 1$. Dann existieren Knoten $i_1,\ldots,i_\gamma$ mit $si_v \in E$ und $i_v \in \Gamma_Q$ für alle $v = 1,\ldots,\gamma$. Somit gilt auch im Fall (b):
$\sum_{u\in Q} x_{uk} + (|Q|-d+1)x_{sk} + \sum_{l\in M\setminus\{k\}} \sum_{i\in I_Q} x_{il} - \sum_{ij\in E} y_{ij} \leq d-1+(|Q|-d+$
$1) + |\Gamma_Q| - \sum_{(s,i_v)\in E,\, 1\leq v\leq\gamma} y_{ij} \leq d-1+(|Q|-d+1)+\gamma-\gamma = |Q|$.

Damit sind die Ungleichungen (15.3.2) und (15.3.3) gültig.

Im folgenden beweisen wir die Eigenschaft, daß die beiden Ungleichungen Facetten für das Polyeder $\mathbf{C}^r(G,f,F)$ definieren.

(i)

Angenommen, $EQ((15.3.2)) \subseteq EQ(a^Tx + b^Ty \leq \beta)$, wobei die Ungleichung $a^Tx + b^Ty \leq \beta$ eine Facette für $\mathbf{C}^r(G,f,F)$ definiert.

(1)

Sei $\Gamma \subseteq Q$ mit $|\Gamma| = d-1$. Dann erfüllt der Vektor $(\sum_{i\in\Gamma} e_{ik} + e_{sk}, 0)$ Ungleichung (15.3.2) mit Gleichheit. Da dies für alle $d-1$-elementigen Teilmengen von Q zutrifft, erhalten wir die Bedingungen $a_{ik} = a_{jk}$ für alle $i,j \in Q$, $i \neq j$.

(2)

Die beiden Lösungen $v_1 = (\sum_{i\in Q} e_{ik}, 0)$ und $v_2 = (\sum_{i\in\Gamma} e_{ik} + e_{sk}, 0)$, $\Gamma \subseteq Q$, $|\Gamma| = d-1$ gehören zu der Seitenfläche $EQ((15.3.2))$. Deshalb gilt:
$(a^T, b^T)v_1 = (a^T, b^T)v_2$. Daraus erhalten wir die Bedingung
$a_{sk} = (|Q| - d + 1)a_{ik}$, $i \in Q$.

(3)

Seien $e_1, e_2 \in E$, $e_1 \neq e_2$. Dann gilt $b_{e_1} = b_{e_2}$.
Sei $i_v \in I_Q$. Dann sind die Vektoren $(\sum_{i\in Q\setminus I_Q} e_{ik} + e_{sl}, 0)$ und $(\sum_{i\in Q\setminus I_Q} e_{ik} + e_{i_v k} + e_{sl}, e_{si_v})$ Elemente der Menge $EQ((15.3.2))$. Dies liefert die Bedingungen $b_{si_v} = -a_{i_v k}$. Da $i_v \in I_Q$ beliebig wählbar ist, erhalten wir zusammen mit (1) die Behauptung.

(4)

Es bleibt zu zeigen, daß $a_{sl} = z$ gilt für alle $l \in M \setminus \{k\}$.
Sei also $l \in M \setminus \{k\}$ vorgegeben. Wir wählen die Lösungen $(\sum_{i\in Q} e_{ik}, 0)$ und $(\sum_{i\in Q} e_{ik} + e_{sl}, \sum_{ij\in E} e_{ij})$. Dann erfüllen beide Vektoren Ungleichung (15.3.2) mit Gleichheit. Daraus folgt zusammen mit (3): $a_{sl} = z$.

(5)

Sei $l \in M \setminus \{k\}$.
Für alle $i \in I_Q$ sind die Lösungen $(e_{sl} + \sum_{j\in Q\setminus I_Q} e_{jk}, 0)$ und $(e_{sl} + e_{il} + \sum_{j\in Q\setminus I_Q} e_{jk}, 0)$ Elemente der Seitenfläche $EQ((15.3.2))$. Dies impliziert: $a_{il} = 0$ für alle $l \in M \setminus \{k\}$ und $i \in I_Q$.
Sei $i \in Q \setminus I_Q$. Ferner sei $Q' \subseteq Q \setminus \{i\}$, $|Q'| = d-1$. Dann sind die Lösungen $(e_{sk} + \sum_{j\in Q'} e_{jk}, 0)$ und $(e_{sk} + e_{il} + \sum_{j\in Q'} e_{jk}, 0)$ Elemente der Seitenfläche $EQ((15.3.2))$. Dies impliziert: $a_{il} = 0$ für alle $l \in M \setminus \{k\}$ und $i \in Q \setminus I_Q$.

Damit ist Teil (i) des Beweises vollständig.

(ii)

Angenommen,
$EQ((15.3.3)) \subseteq EQ(a^T x + b^T y \leq \beta)$, wobei die Ungleichung $a^T x + b^T y \leq \beta$ eine Facette für $\mathbf{C}^r(G, f, F)$ definiert.

(1)+(2)

Der Beweis in (1) und (2) im Teil (i) liefert uns analog: Für alle $i,j \in Q$, $i \neq j$ gilt die Beziehung $a_{ik} = a_{jk}$. Ferner gilt $a_{sk} = (|Q| - d + 1)a_{ik}$ für alle $i \in Q$.

(3)

Sei $i \in I_Q$ und $l \in M \setminus \{k\}$. Dann erfüllen die Vektoren $(\sum_{j\in Q} e_{jk}, 0)$ und $(\sum_{j\in Q\setminus\{i\}} e_{jk} + e_{il}, 0)$ Ungleichung (15.3.3) mit Gleichheit. Dies liefert zusammen mit (1) die Beziehung: $a_{ik} = a_{il}$ für alle $i \in I_Q$ und $l \in M \setminus \{k\}$.

(4)

Für $e_1 \neq e_2 \in E$ gilt $b_{e_1} = b_{e_2}$.
Sei $i_v \in I_Q$ und $Q' \subset Q \setminus I_Q$, $|Q'| = d-1$. Eine solche Menge existiert, da nach Voraussetzung $z + d - 1 \leq |Q|$ ist. Dann aber sind die Vektoren $(\sum_{i\in Q'} e_{ik} + e_{sk}, 0)$ und $(\sum_{i\in Q'} e_{ik} + e_{i_v l} + e_{sk}, e_{si_v})$ Elemente der Menge $EQ((15.3.3))$. Dies liefert die Bedingungen $b_{si_v} = -a_{i_v l}$ für alle $i_v \in I_Q$. Da v beliebig wählbar ist, erhalten wir zusammen mit (3) die Behauptung.

(5)

Sei $l \in M \setminus \{k\}$.
Für alle $i \in I_Q$ sind die Lösungen $(\sum_{i\in I_Q} e_{il} + \sum_{j\in Q\setminus I_Q} e_{jk}, 0)$ und $(\sum_{i\in I_Q} e_{il} + e_{sl} + \sum_{j\in Q\setminus I_Q} e_{jk}, 0)$ Elemente der Seitenfläche $EQ((15.3.3))$. Dies impliziert:
$a_{sl} = 0$ für alle $l \in M \setminus \{k\}$.
Sei $i \in Q \setminus I_Q$. Ferner sei $Q' \subseteq Q \setminus \{i\}$, $|Q'| = d-1$. Dann sind die Lösungen $(e_{sk} + \sum_{j\in Q'} e_{jk}, 0)$ und $(e_{sk} + e_{il} + \sum_{j\in Q'} e_{jk}, 0)$ Elemente der Seitenfläche $EQ((15.3.3))$. Dies impliziert: $a_{il} = 0$ für alle $l \in M \setminus \{k\}$ und $i \in Q \setminus I_Q$.

Damit ist auch Teil (ii) des Beweises vollständig.

□

15.4 Ungleichungen aus zwei Überdeckungen und einem Baum

(15.4.1) Satz:
Gegeben sei eine Instanz des r-Clusteringproblems mit den folgenden Eigenschaften:
Sei $M = \{1, \ldots, r\}$ *und* $k_0, l_0 \in M$, $k_0 \neq l_0$. *Sei ferner* $V = S^{k_0} \cup S^{l_0}$, *wobei* S^{k_0} *bzw.* S^{l_0} *minimale Überdeckungen von* k_0 *bzw.* l_0 *sind. Es bezeichne* $D = S^{k_0} \cap S^{l_0}$. *Die Kantenmenge* E *des Graphen* $G = (V, E)$ *definiere einen aufspannenden Baum bezüglich* D. *Zusätzlich seien die Ungleichungen* $\sum_{i\in D\setminus\{d\}} f_i \leq F$ *für alle* $k \in M \setminus \{k_0, l_0\}$ *erfüllt, wobei* d *ein spezielles Element aus* D *bezeichnet. Falls ferner* $|D| \geq 2$, $\sum_{i\in S^{k_0}\setminus D} f_i + f_j \leq F$ *für alle* $j \in S^{l_0} \setminus D$ *und* $\sum_{i\in S^{l_0}\setminus D} f_i + f_j \leq F$ *für alle* $j \in S^{k_0} \setminus D$ *gilt, so definiert die Ungleichung*

$$(15.4.2) \quad \sum_{i\in S^{k_0}} x_{ik_0} + \sum_{i\in S^{l_0}} x_{il_0} + \sum_{i\in D\setminus\{d\}} \sum_{k\in M\setminus\{k_0,l_0\}} x_{ik} - \sum_{ij\in E} y_{ij} \leq |S^{k_0}| + |S^{l_0}| - |D| - 1.$$

eine Facette für das Polyeder $\mathbf{C}^r(G, f, F)$.

Bemerkung:

Anschaulich läßt sich die Ungleichung (15.4.2) folgendermaßen interpretieren:

Gegeben sei eine minimale Überdeckung S^{k_0} für ein $k_0 \in M$ und eine minimale Überdeckung für $l_0 \in M \setminus \{k_0\}$. Ferner bezeichne D den Durchschnitt der beiden Überdeckungen. Dann ist sicherlich folgende Ungleichung gültig:

$$\sum_{i \in S^{k_0}} x_{ik_0} + \sum_{i \in S^{l_0}} x_{il_0} \leq |S^{k_0}| + |S^{l_0}| - |D|$$

Gegeben ist ferner eine Kantenmenge in dem Graphen. Diese definiert einen aufspannenden Baum bezüglich D. Falls mehr als $|S^{k_0}| + |S^{l_0}| - |D| - 1$ Knotenvariablen den Wert 1 annehmen, muß die Menge D auf mindesten zwei verschiedene Cluster aufgeteilt werden. Da E einen aufspannenden Baum bezüglich D definiert, existiert dann mindestens eine Kante in dem entsprechenden Schnitt.

Die folgende Abbildung stellt den Graphen für ein Beispiel mit $|N| = 12$ und $|D| = 8$ dar. Die Knoten aus D sind schwarz gezeichnet. Die Knoten aus $V \setminus D$ sind nur mit einem schwarzen Rand versehen.

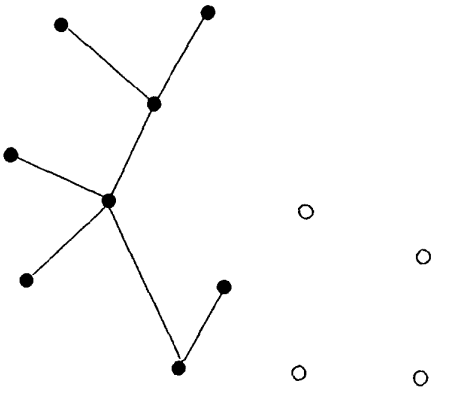


Abbildung 15.4.3

Beweis:

Wir beweisen zunächst die Gültigkeit der Ungleichung.

Da $D = S^{k_0} \cap S^{l_0}$, gilt für alle $(x, y) \in \mathbf{C}^r(G, f, F)$ die Ungleichung:

$\sum_{i \in S^{k_0}} x_{ik_0} + \sum_{i \in S^{l_0}} x_{il_0} + \sum_{i \in D \setminus \{d\}} \sum_{k \in M \setminus \{k_0, l_0\}} x_{ik} \leq |S^{k_0}| - |D| + |S^{l_0}| - |D| + |D| = |S^{k_0}| + |S^{l_0}| - |D|$.

Sei nun $(x, y) \in \mathbf{C}^r(G, f, F)$ ein Vektor, für den gilt:

$\sum_{i \in S^{k_0}} x_{ik_0} + \sum_{i \in S^{l_0}} x_{il_0} + \sum_{i \in D \setminus \{d\}} \sum_{k \in M \setminus \{k_0, l_0\}} x_{ik} = |S^{k_0}| + |S^{l_0}| - |D|$.

Dies impliziert:

$$x_{ik_0} = 1 \text{ für alle } i \in S^{k_0} \setminus D,$$
$$x_{il_0} = 1 \text{ für alle } i \in S^{l_0} \setminus D, \text{ und}$$
$$\sum_{i \in D} \sum_{k \in M} x_{ik} = |D|.$$

Ferner gilt, daß $\sum_{i \in D} x_{ik_0} < |D|$ und $\sum_{i \in D} x_{il_0} < |D|$ ist, da S^{k_0} bzw. S^{l_0} minimale Überdeckungen von k_0 bzw. l_0 sind. Daraus können wir ableiten, daß mindestens zwei Teilmengen $D_1, D_2 \subseteq D$, $D = D^1 \cup D^2$, $D^1 \cap D^2 = \emptyset$ und Zahlen $k, l \in M$, $k \neq l$ existieren, so daß $x_{ik} = 1$ für alle $i \in D^1$ und $x_{il} = 1$ für alle $i \in D^2$.
Da der Graph einen aufspannenden Baum bezüglich D definiert, existiert mindestens eine Kante e, welches im Schnitt $[D_1 : D_2]$ liegt. Somit gilt $\sum_{i \in S^{k_0}} x_{ik_0} + \sum_{i \in S^{l_0}} x_{il_0} + \sum_{i \in D \setminus \{d\}} \sum_{k \in M \setminus \{k_0, l_0\}} x_{ik} - \sum_{ij \in E} y_{ij} \leq |S^{k_0}| + |S^{l_0}| - |D| - y_e = |S^{k_0}| + |S^{l_0}| - |D| - 1$.
Dies beweist die Gültigkeit der Ungleichung.

Obige Ungleichung definiert eine Facette für das Polyeder $\mathbf{C}^r(G, f, F)$.

Angenommen, $EQ((15.4.2)) \subseteq EQ(a^T x + b^T y \leq \beta)$, wobei die Ungleichung $a^T x + b^T y \leq \beta$ eine Facette für $\mathbf{C}^r(G, f, F)$ definiert.

(1)

Sei $s \in S^{k_0}$. Dann sind die Vektoren $(\sum_{u \in S^{k_0} \setminus \{s\}} e_{uk_0} + \sum_{u \in S^{l_0} \setminus D} e_{ul_0}, 0)$ Elemente der Menge $EQ((15.4.2))$ und damit in $EQ(a^T x + b^T y \leq \beta)$ enthalten.
Dies liefert die Bedingungen $a_{sk_0} = a_{s'k_0}$ für alle $s \neq s'$, $s, s' \in S^{k_0}$.

(2)

Analog zeigen wir $a_{sl_0} = a_{s'l_0}$ für alle $s \neq s'$, $s, s' \in S^{l_0}$.
Sei $s \in S^{l_0}$. Dann sind die Vektoren $(\sum_{u \in S^{l_0} \setminus \{s\}} e_{ul_0} + \sum_{u \in S^{k_0} \setminus D} e_{uk_0}, 0)$ Elemente der Menge $EQ((15.4.2))$ und damit in $EQ(a^T x + b^T y \leq \beta)$ enthalten.
Dies liefert die Bedingungen $a_{sl_0} = a_{s'l_0}$ für alle $s \neq s'$, $s, s' \in S^{l_0}$.

(3)

Wir zeigen nun: $a_{sk_0} = a_{sl_0}$ für ein $s \in D$. Zusammen mit (1) und (2) folgt daraus, $a_{ik_0} = a_{jl_0}$ für alle $i \in S^{k_0}, j \in S^{l_0}$.
Sei $s_0 \in D$ ein Blatt in dem Baum, welcher durch die Kantenmenge E gegeben ist, d.h. s_0 ist inzident zu genau einer Kante in dem Graphen. Die Existenz eines solchen ist gesichert, da die Kantenmenge einen Baum definiert. Ferner bezeichne $t_0 \in E$ die Kante, welche inzident zu s_0 ist. Dann erfüllen die beiden Vektoren $v_1 := (\sum_{u \in (S^{k_0} \setminus D) \cup \{s_0\}} e_{uk_0} + \sum_{u \in S^{l_0} \setminus \{s_0\}} e_{ul_0}, e_{t_0})$ und $v_2 = (\sum_{u \in S^{k_0} \setminus \{s_0\}} e_{uk_0} + \sum_{u \in (S^{l_0} \setminus D) \cup \{s_0\}} e_{ul_0}, e_{t_0})$ Ungleichung (15.4.2) mit Gleichheit. Daher gilt $(a^T, b^T)v_1 = (a^T, b^T)v_2$. Zusammen mit (1) und (2) ergibt sich daraus:

$$(|S^{k_0}| - |D| + 1)a_{s_0 k_0} + (|S^{l_0}| - 1)a_{s_0 l_0} + b_{t_0} =$$
$$(|S^{k_0}| - 1)a_{s_0 k_0} + (|S^{l_0}| - |D| + 1)a_{s_0 l_0} + b_{t_0}.$$

Daraus folgt schließlich $a_{s_0 k_0} = a_{s_0 l_0}$.

(4)

Sei $ij \in E$. Da der Graph einen aufspannenden Baum bezüglich D definiert, existiert eine eindeutige Zerlegung von D in zwei Mengen D_1 und D_2, so daß der Schnitt zwischen D_1 und D_2 aus genau der Kante ij besteht. Ohne Beschränkung der Allgemeinheit können wir $i \in D_1$ voraussetzen.
Damit erfüllen die beiden Vektoren $v_1 := (\sum_{u \in (S^{k_0} \setminus D) \cup D_1} e_{uk_0} + \sum_{u \in (S^{l_0} \setminus D) \cup D_2} e_{ul_0}, e_{ij})$ und $v_2 := (\sum_{u \in (S^{k_0} \setminus D) \cup (D_1 \setminus \{i\})} e_{uk_0} + \sum_{u \in (S^{l_0} \setminus D) \cup D_2} e_{ul_0}, 0)$ Ungleichung (15.4.2) mit Gleichheit. Daher gilt $(a^T, b^T)v_1 = (a^T, b^T)v_2$. Dies impliziert zusammen mit (1):

$$b_{ij} = -a_{ik_0} = -a_{uk_0} \text{ für alle } u \in V \text{ und für alle } ij \in E.$$

(5)

Sei $l \in M \setminus \{k_0\}$. Da für alle $i \in S^{k_0} \setminus D$ die Vektoren $(\sum_{u \in S^{k_0} \setminus \{i\}} e_{uk_0} + \sum_{u \in S^{l_0} \setminus D} e_{ul_0}, 0)$ und $(\sum_{u \in S^{k_0} \setminus \{i\}} e_{uk_0} + \sum_{u \in S^{l_0} \setminus D} e_{ul_0} + e_{il}, 0)$ in der durch (15.4.2) definierten Seitenfläche enthalten sind, sind sie auch Elemente der durch $a^T x + b^T y \leq \beta$ definierten Facette. Dies liefert die Bedingungen $a_{il} = 0$ für alle $i \in S^{k_0} \setminus D$, $l \in M \setminus \{k_0\}$.

(6)

Analog zeigt man, $a_{il} = 0$ für alle $i \in S^{l_0} \setminus D$, $l \in M \setminus \{l_0\}$.

(7)

Es bleibt noch zu zeigen: $a_{il} = a_{ik_0}$ für alle $i \in D \setminus \{d\}$ und $a_{dl} = 0$ für $l \in M \setminus \{k_0, l_0\}$.
Für alle $i \in D \setminus \{d\}$ existiert ein eindeutiger Weg in dem Graphen von i nach d. Bezeichne $W = (i, t_0, j, t_1, \ldots, d)$ diesen Weg. Ferner gilt, daß die Kante t_0 die Grundmenge D in zwei Mengen D_1 und D_2 zerlegt, so daß der Schnitt zwischen D_1 und D_2 aus genau der Kante t_0 besteht. Ohne Beschränkung der Allgemeinheit können wir $i \in D_1$ voraussetzen. Daraus folgt: $d \in D_2$.
Dann erfüllen die Lösungen $v_1 = (\sum_{u \in (S^{k_0} \setminus D) \cup D_2} e_{uk_0} + \sum_{u \in S^{l_0} \setminus D} e_{ul_0} + \sum_{u \in D_1} e_{ul}, e_{t_0})$ und $v_2 = (\sum_{u \in (S^{k_0} \setminus D) \cup D_2} e_{uk_0} + \sum_{u \in S^{l_0} \setminus D} e_{ul_0} + \sum_{u \in D_1 \setminus \{i\}} e_{ul}, 0)$ Ungleichung (15.4.2) mit Gleichheit. Daher gilt $(a^T, b^T)v_1 = (a^T, b^T)v_2$. Zusammen mit (4) ergibt sich daraus:

$$b_{t_0} = -a_{il}.$$

Dies gilt für alle $l \in M \setminus \{k_0, l_0\}$. Somit folgt: $a_{il} = a_{ik_0}$ für alle $i \in D \setminus \{d\}$, $l \in M \setminus \{k_0, l_0\}$.
Ferner sind die Vektoren $v_1 := (\sum_{u \in S^{k_0} \setminus \{d\}} e_{uk_0} + \sum_{u \in S^{l_0} \setminus D} e_{ul_0}, 0)$ und
$v_2 := (\sum_{u \in S^{k_0} \setminus D} e_{uk_0} + \sum_{u \in S^{l_0} \setminus D} e_{ul_0} + \sum_{u \in D} e_{ul}, 0)$ in der durch $a^T x + b^T y \leq \beta$ definierten Facette enthalten. Daraus ergibt sich $a_{dl} = 0$ für alle $l \in M \setminus \{k_0, l_0\}$.
Damit ist der Beweis vollständig. □

In den letzten zwei Abschnitten von Kapitel 15 untersuchen wir noch zwei speziellere Klassen von Ungleichungen.

15.5 Ungleichungen aus einer Überdeckung und einer Kante

(15.5.1) Satz: *Gegeben sei eine Instanz des r-Clusteringproblems mit den folgenden Eigenschaften:*
Sei $M = \{1, \ldots, r\}$, $k \in M$ *und* $S = \{1, \ldots, n\}$ *eine minimale Überdeckung von* k. *Ferner seien* $s, s' \in S$, $s \neq s'$ *gegeben. Die Kantenmenge* E *des Graphen* $G = (S, E)$ *bestehe nur aus der Kante* s, s', *und es gelte* $f_s + f_{s'} \leq F$.
Dann definiert die Ungleichung

$$(15.5.2) \quad 2\sum_{u \in S} x_{uk} + \sum_{l \in M \setminus \{k\}} (x_{sl} + x_{s'l}) - y_{ss'} \leq 2|S| - 2.$$

eine Facette für das Polyeder $\mathbf{C}^r(G, f, F)$.

Bemerkung:
Da S eine Überdeckung von k definiert, gilt sicherlich die Ungleichung

$$2\sum_{u \in S} x_{uk} \leq 2|S| - 2.$$

Falls die beiden Knoten s oder s' zwei verschiedenen Clustern zugeordnet werden, muß die Variable $y_{ss'}$ den Wert 1 erhalten.
Beweis:
Wir beweisen zunächst die Gültigkeit der Ungleichung.
Sei $(x, y) \in \mathbf{C}^r(G, f, F)$. Dann sind folgende Ungleichungen gültig.

$$2\sum_{u \in S} x_{uk} \leq 2|S| - 2,$$
$$\sum_{l \in M} x_{sl} \leq 1 \text{ und } \sum_{l \in M} x_{s'l} \leq 1$$

Wenn $x_{sk} = x_{s'k} = 1$, dann gilt:
$2\sum_{u \in S} x_{uk} + \sum_{l \in M \setminus \{k\}} (x_{sl} + x_{s'l}) \leq 2(|S| - 1) = 2|S| - 2$.
Wenn $\sum_{l \in M \setminus \{k\}} x_{sl} = \sum_{l \in M \setminus \{k\}} x_{s'l} = 1$, dann gilt:
$2\sum_{u \in S} x_{uk} + \sum_{l \in M \setminus \{k\}} (x_{sl} + x_{s'l}) \leq 2(|S| - 2) + 2 = 2|S| - 2$.
Wenn $x_{sk} = 1$ und $\sum_{l \in M \setminus \{k\}} x_{s'l} = 1$ oder umgekehrt, dann gilt:
$2\sum_{u \in S} x_{uk} + \sum_{l \in M \setminus \{k\}} (x_{sl} + x_{s'l}) \leq 2(|S| - 1) + 1 = 2|S| - 1$.
In allen anderen Fällen ist die Ungleichung trivialerweise gültig.

Nehmen wir an, $2\sum_{u \in S} x_{uk} + \sum_{l \in M \setminus \{k\}} (x_{sl} + x_{s'l}) = 2|S| - 1$. Ohne Beschränkung der Allgemeinheit können wir voraussetzen, daß $x_{sk} = 1$ und $\sum_{l \in M \setminus \{k\}} x_{s'l} = 1$.
Da $(x, y) \in \mathbf{C}^r(G, f, F)$, muß in diesem Fall $y_{ss'} = 1$ gelten.

Dies beweist die Gültigkeit der Ungleichung (15.5.2).

Im folgenden Teil beweisen wir, daß Ungleichung (15.5.2) eine Facette definiert.

Angenommen,
$EQ((15.5.2)) \subseteq EQ(a^T x + b^T y \leq \beta)$, wobei die Ungleichung $a^T x + b^T y \leq \beta$ eine Facette für $\mathbf{C}^r(G, f, F)$ definiert.

(1)

Sei $i \in S$. Dann sind die Vektoren $(\sum_{u \in S \setminus \{i\}} e_{uk}, 0)$ Elemente der Menge
$EQ((15.5.2))$ und damit in $EQ(a^T x + b^T y \leq \beta)$ enthalten.
Dies liefert die Bedingungen $a_{ik} = a_{jk}$ für alle $i \neq j$, $i, j \in S$.

(2)

Sei $l \in M \setminus \{k\}$. Wir definieren $v_1 = (\sum_{u \in S \setminus \{s\}} e_{uk} + e_{sl}, e_{ss'})$ und $v_2 = (\sum_{u \in S \setminus \{s'\}} e_{uk} + e_{s'l}, e_{ss'})$. v_1 und v_2 sind Elemente der durch (15.5.2) definierten Seitenfläche. Daraus folgt: $(a^T, b^T)v_1 = (a^T, b^T)v_2$. Da wir aus (1) $a_{sk} = a_{s'k}$ voraussetzen können, ergibt sich $a_{sl} = a_{s'l}$ für alle $l \in M \setminus \{k\}$.

(3)

Wir zeigen $b_{ss'} = -a_{sl}$ für ein $l \in M \setminus \{k\}$. Sei $l \in M \setminus \{k\}$. Betrachten wir die folgenden zwei Vektoren: $v_1 = (\sum_{u \in S \setminus \{s\}} e_{uk} + e_{sl}, e_{ss'})$ und $v_2 = (\sum_{u \in S \setminus \{s\}} e_{uk}, 0)$. Beide Vektoren erfüllen (15.5.2) mit Gleichheit. Dies liefert die Behauptung.

(4)

Wir zeigen $a_{sk} = a_{sl} + a_{s'l}$ für ein $l \in M \setminus \{k\}$. Sei $l \in M \setminus \{k\}$. Wir betrachten die folgenden zwei Vektoren: $v_1 = (\sum_{u \in S \setminus \{s'\}} e_{uk}, 0)$ und $v_2 = (\sum_{u \in S \setminus \{s.s'\}} e_{uk} + e_{sl} + e_{s'l}, 0)$.
Gemäß (1) und (2) gilt:
$(a^T, b^T)v_1 = (|S|-1)a_{sk} = (|S|-2)a_{sk} + a_{sl} + a_{s'l} = (a^T, b^T)v_2$. Dies liefert die Bedingung $a_{sk} = a_{sl} + a_{s'l} = 2a_{sl}$.

(5)

Schließlich bestimmen wir noch die Koeffizienten a_{il} für alle $l \in M \setminus \{k\}$ und $i \in S \setminus \{s, s'\}$. Sei also $l \in M \setminus \{k\}$ und $i \in S \setminus \{s, s'\}$ beliebig gegeben. Dann sind die beiden Lösungen $(\sum_{u \in S \setminus \{i\}} e_{uk}, 0)$ und $(\sum_{u \in S \setminus \{i\}} e_{uk} + e_{il}, 0)$ Elemente der durch (15.5.2) definierten Seitenfläche. Dies liefert die Bedingung: $a_{il} = 0$.

Somit ist der Beweis von Satz (15.5.1) vollständig.

15.6 Eine kleine Klasse von kombinierten Knoten-Kanten-Ungleichungen

Die linken Seiten der im folgenden vorgestellten Ungleichungen kombinieren im wesentlichen zwei minimale Überdeckungen, die verschieden gewichtet werden. In der Ungleichung treten dabei Koeffizienten auf, die sehr spezielle Eigenschaften erfüllen müssen. Wir wollen daher vorab die technische Definition der Koeffizienten angeben.

Definition (15.6.1): Gegeben sei eine Instanz des r-Clusteringproblems mit folgenden Eigenschaften:
Sei $k, l \in M$, $k \neq l$. Ferner definieren die Mengen S^k und S^l minimale Überdeckungen von k bzw. l. Es gelte $S^k \cap S^l = \{s, s'\}$ und $V = S^k \cup S^l \cup \{j_2\}$, wobei $j_2 \notin S^k \cup S^l$. Gegeben sei ferner $j_1 \in S^l \setminus S^k$, und es gelte $E = \{ss'\}$. Dann definieren wir zwei Zahlen a_{j_1} und a_{j_2} gemäß:

$$a_{j_1} = \min\{2|S^k| - 2|D_1| - 3 \mid D_1 \subseteq S^k \setminus \{s, s'\}, \sum_{i \in D_1} f_i + f_{j_1} \leq F\},$$

$$a_{j_2} = \min\{2|S^k| - 2|D_2| - 3 \mid D_2 \subseteq S^k \setminus \{s, s'\}, \sum_{i \in D_2} f_i + f_{j_2} \leq F\}.$$

Mit $D_1^* \subseteq S^k \setminus \{s, s'\}$ bzw. $D_2^* \subseteq S^k \setminus \{s, s'\}$ bezeichnen wir die Mengen, welche die Gleichungen $a_{j_1} = 2|S^k| - 2|D_1^*| - 3$ bzw. $a_{j_2} = 2|S^k| - 2|D_2^*| - 3$ erfüllen.
Dabei setzen wir voraus, daß

$$\sum_{i \in D_1^*} f_i + f_s + f_{j_1} \leq F,$$

$$\sum_{i \in D_1^*} f_i + f_s' + f_{j_1} \leq F,$$

$$\sum_{i \in D_1} f_i + f_s + f_{s'} + f_{j_1} > F \text{ für alle } D_1 \subseteq S^k \setminus \{s, s'\} \text{ mit } |D_1| = |D_1^*|.$$

Ferner gelte, daß

$$\sum_{i \in W} f_i + f_{j_2} > F, \text{ für alle } W \subseteq S^k, \ |W| = |D_2^*| + 1,$$

$$\sum_{u \in D_2^*} f_u + f_{j_2} + f_i \leq F, \text{ für alle } i \in S^l \setminus \{s, s', j_1\}.$$

Schließlich sei
$d^* \leq |D_1^*| + |D_2^*| - |S^k| + 1$, wobei

$$d^* = \max\{|D| \mid D \subseteq S^k, \sum_{i \in D} f_i + f_{j_1} + f_{j_2} \leq F\},$$

(15.6.2) Satz: *Gegeben sei eine Instanz des r-Clusteringproblems, welche die Voraussetzungen in Definition (15.6.1) erfüllt. Ferner gelten die Ungleichung*
$\sum_{u \in S^l \setminus \{s,s'\}} f_u + f_i \leq F$, *für alle* $i \in S^k \setminus \{s, s'\}$,
$\sum_{i \in S^l \setminus \{s,s'\}} f_i + f_{j_2} \leq F$ *und*
$f_s + f_{s'} \leq F$.

Dann definiert die Ungleichung

$$(15.6.3)\quad 2\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} + a_{j_1}x_{j_1k} + a_{j_2}x_{j_2k} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq 2|S^k| + |S^l| - 4$$

eine Facette für das Polyeder $\mathbf{C}^r(G,f,F)$, *wobei die Koeffizienten* a_{j_1k} *und* a_{j_2k} *gemäß Definition (15.6.1) bestimmt werden.*

Bemerkung:
Auf das wesentliche beschränkt, läßt sich die Ungleichung sehr einfach interpretieren:
Die Knotenmengen S^k bzw. S^l entsprechen minimalen Überdeckungen bezüglich der Cluster k bzw. l. Der Durchschnitt beider Mengen besteht aus den zwei Elementen s und s'. Da in der Ungleichung (15.6.3) die Elemente des Clusters k doppelt gezählt werden, ergibt sich sofort die Ungleichung

$$2\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} \leq 2|S^k| + |S^l| - 3.$$

Ferner kann man sich sehr einfach überlegen, daß $2\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} = 2|S^k| + |S^l| - 3$ nur gelten kann, wenn s und s' in verschiedenen Cluster sind. Deshalb wird in diesem Fall die Kante ss' getrennt. Somit erhalten wir die Ungleichung

$$2\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} - y_{ss'} \leq 2|S^k| + |S^l| - 4.$$

Desweiteren ist offensichtlich, daß die rechte Seite unverändert bleibt, wenn wir auf der linken Seite den Ausdruck $\sum_{v\in M\setminus\{k,l\}} x_{sv}$ hinzunehmen.
Die Koeffizienten a_{j_1} und a_{j_2} bestimmen sich relativ kompliziert. Dies ist aber aus technischen Gründen nicht zu vermeiden. Anderenfalls könnte man nicht beweisen, daß Ungleichung (15.6.3) eine Facette definiert.

Beweis: Zur Darstellung des Beweises verwenden wir die Bezeichnungen aus Definition (15.6.1).
Wir beweisen zunächst die Gültigkeit der Ungleichung (15.6.3).

Sei $(x,y) \in \mathbf{C}^r(G,f,F)$. Wir unterscheiden folgende Fälle:

(i)

Sei $x_{j_1k} = x_{j_2k} = 1$. Bezeichne D die Teilmenge von S^k, mit $x_{ik} = 1$ für alle $i \in D$. Nach Voraussetzung gilt:

$$|D| \leq d^* \leq |D_1^*| + |D_2^*| - |S^k| + 1.$$

Da $x_{j_1k} = 1$ und $j_1 \in S^l$ gilt ferner die Ungleichung:

$$\sum_{i\in S^l} x_{il} + \sum_{v\in M\setminus\{k,l\}} x_{sv} \leq |S^l| - 1.$$

Insgesamt erhalten wir somit:

$\sum_{u \in S^k} x_{uk} + \sum_{u \in S^l} x_{ul} + a_{j_1} + a_{j_2} + \sum_{v \in M \setminus \{k,l\}} x_{sv} - y_{ss'} \leq$
$(2|S^k| - 2|D_1^*| - 3) + (2|S^k| - 2|D_2^*| - 3) + 2(|D_1^*| + |D_2^*| - |S^k| + 1) + |S^l| - 1 =$
$2|S^k| + |S^l| - 5.$

Dies beweist die Gültigkeit in dem Fall $x_{j_1 k} = x_{j_2 k} = 1$.

(ii)

Sei $x_{j_1 k} = 1$ und $x_{j_2 k} = 0$. Bezeichne D_1 die Teilmenge von $S^k \setminus \{s, s'\}$, mit $x_{ik} = 1$ für alle $i \in D_1$.

(1) Wenn nun $x_{sk} = x_{s'k} = 1$, dann gilt die Beziehung: $|D_1| \leq |D_1^*| - 1$. Denn angenommen, $|D_1| = |D_1^*|$, so könnten x_{sk} und $x_{s'k}$ nach Voraussetzung nicht beide den Wert 1 annehmen. Desweiteren gilt die Ungleichung $\sum_{i \in S^l} x_{il} \leq |S^l| - 3$, da j_1, s und s' in Cluster k zusammengefasst sind.

Insgesamt erhalten wir somit:

$\sum_{u \in S^k} x_{uk} + \sum_{u \in S^l} x_{ul} + a_{j_1} + \sum_{v \in M \setminus \{k,l\}} x_{sv} - y_{ss'} \leq$
$(2|S^k| - 2|D_1^*| - 3) + 2(|D_1| + x_{sk} + x_{s'k}) + |S^l| - 3 \leq$
$2|S^k| - 2|D_1^*| - 3 + 2(|D_1^*| - 1 + 2) + |S^l| - 3 =$
$2|S^k| + |S^l| - 4.$

(2) Wenn $x_{sk} = 1$ und $x_{s'k} = 0$, dann gilt sicherlich $|D_1| \leq |D_1^*|$. Ferner gilt die Ungleichung $\sum_{i \in S^l} x_{il} \leq |S^l| - 2$, da j_1 und s Cluster k zugewiesen sind. Falls $x_{s'l} = 1$, dann muß die Variable $y_{ss'}$ den Wert 1 erhalten. Somit erhalten wir die Ungleichung:

$\sum_{i \in S^l} x_{il} - y_{ss'} \leq |S^l| - 3.$

Insgesamt ergibt sich schließlich:

$\sum_{u \in S^k} x_{uk} + \sum_{u \in S^l} x_{ul} + a_{j_1} + \sum_{v \in M \setminus \{k,l\}} x_{sv} - y_{ss'} \leq$
$2|S^k| - 2|D_1^*| - 3 + 2(|D_1| + 1) + |S^l| - 3 \leq 2|S^k| - 2|D_1^*| - 3 + 2|D_1^*| + 2 + |S^l| - 3 =$
$2|S^k| + |S^l| - 4.$

(3) Wenn $x_{s'k} = 1$ und $x_{sk} = 0$, dann gilt wiederum $|D_1| \leq |D_1^*|$. Ferner gilt die Ungleichung $\sum_{i \in S^l} x_{il} \leq |S^l| - 2$, da j_1 und s' Cluster k zugewiesen sind. Falls $\sum_{v \in M \setminus \{k\}} x_{sv} = 1$, dann muß die Variable $y_{ss'}$ den Wert 1 erhalten. Somit erhalten wir die Ungleichung:

$\sum_{i \in S^l} x_{il} + \sum_{v \in M \setminus \{k,l\}} x_{sv} - y_{ss'} \leq |S^l| - 3.$

Insgesamt ergibt sich somit analog zu (2):

$\sum_{u \in S^k} x_{uk} + \sum_{u \in S^l} x_{ul} + a_{j_1} + \sum_{v \in M \setminus \{k,l\}} x_{sv} - y_{ss'} \leq$
$2|S^k| - 2|D_1^*| - 3 + 2(|D_1| + 1) + |S^l| - 3 \leq 2|S^k| - 2|D_1^*| - 3 + 2|D_1^*| + 2 + |S^l| - 3 =$
$2|S^k| + |S^l| - 4.$

(4) Sei also $x_{sk} = x_{s'k} = 0$. Damit gilt: $|D_1| \leq |D_1^*|$. Da j_1 Cluster k zugewiesen wurde, ist die Ungleichung $\sum_{i \in S^l} x_{il} + \sum_{v \in M \setminus \{k,l\}} x_{sv} \leq |S^l| - 1$ gültig.

Folglich gilt:

$\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} + a_{j_1} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq$
$(2|S^k| - 2|D_1^*| - 3) + 2|D_1| + |S^l| - 1 \leq$
$2|S^k| - 2|D_1^*| - 3 + 2|D_1^*| + |S^l| - 1 =$
$2|S^k| + |S^l| - 4.$

Dies beweist die Gültigkeit auch in dem Fall $x_{j_1 k} = 1$ und $x_{j_2 k} = 0$.

(iii)

Sei $x_{j_1 k} = 0$ und $x_{j_2 k} = 1$. Bezeichne D_2 die Teilmenge von $S^k \setminus \{s, s'\}$, mit $x_{ik} = 1$ für alle $i \in D_2$.

(1) Wenn $x_{sk} = x_{s'k} = 1$, dann muß $|D_2| \leq |D_2^*| - 2$ sein. Diese Ungleichung ist gültig, da zunächst nach Voraussetzung $|D_2| \leq |D_2^*|$ ist. Falls $|D_2| \geq |D_2^*| - 1$, so existiert eine Menge $W := D_2 \cup \{s, s'\}$ mit $\sum_{i\in W} f_i + f_{j_2} \leq F$ und $|W| = |D_2^*| + 1$. Dies widerspräche den Voraussetzungen des Satzes (15.6.2).
Desweiteren gilt die Beziehung: $\sum_{i\in S^l} x_{il} \leq |S^l| - 2$, da s und s' Cluster k zugewiesen sind.
Insgesamt erhalten wir somit:
$\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} + a_{j_2} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq$
$(2|S^k| - 2|D_2^*| - 3) + 2(|D_2| + 2) + |S^l| - 2 \leq$
$2|S^k| - 2|D_2^*| - 3 + 2(|D_2^*| - 2 + 2) + |S^l| - 2 =$
$2|S^k| + |S^l| - 5.$

(2) Wenn $x_{sk} = 1$ und $x_{s'k} = 0$, dann gilt die Ungleichung: $|D_2| \leq |D_2^*| - 1$. Diese Ungleichung ist gültig, da zunächst nach Voraussetzung $|D_2| \leq |D_2^*|$ ist. Falls $|D_2| = |D_2^*|$, so existiert eine Menge $W := D_2 \cup \{s\}$ mit $\sum_{i\in W} f_i + f_{j_2} \leq F$ und $|W| = |D_2^*| + 1$. Dies widerspräche den Voraussetzungen des Satzes (15.6.2).
Ferner gilt: $\sum_{i\in S^l} x_{il} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq |S^l| - 1$. Insgesamt ergibt sich somit:
$\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} + a_{j_2} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq$
$(2|S^k| - 2|D_2^*| - 3) + 2(|D_2| + 1) + |S^l| - 1 \leq$
$2|S^k| - 2|D_2^*| - 3 + 2(|D_2^*| - 1 + 1) + |S^l| - 1 =$
$2|S^k| + |S^l| - 4.$

(3) Wenn $x_{s'k} = 1$ und $x_{sk} = 0$, dann gilt analog zu (iii.2) die Ungleichung: $|D_2| \leq |D_2^*| - 1$. Ferner gilt: $\sum_{i\in S^l} x_{il} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq |S^l| - 1$, denn wenn $\sum_{v\in M\setminus\{k,l\}} x_{sv}$ den Wert 1 animmt, dann muß $y_{ss'}$ ebenfalls 1 sein. Insgesamt ergibt sich somit:
$\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} + a_{j_2} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq$
$(2|S^k| - 2|D_2^*| - 3) + 2(|D_2| + 1) + |S^l| - 1 \leq$
$2|S^k| - 2|D_2^*| - 3 + 2(|D_2^*| - 1 + 1) + |S^l| - 1 =$
$2|S^k| + |S^l| - 4.$

(4) Sei also $x_{sk} = x_{s'k} = 0$. Dann können wir annehmen, daß $|D_2| \leq |D_2^*|$ ist. Ferner gilt: $\sum_{i\in S^l} x_{il} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq |S^l| - 1$.

Damit ergibt sich die Abschätzung:
$\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} + a_{j_2} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq$
$(2|S^k| - 2|D_2^*| - 3) + 2|D_2| + |S^l| - 1 \leq$
$2|S^k| - 2|D_2^*| - 3 + 2|D_2^*| + |S^l| - 1 =$
$2|S^k| + |S^l| - 4.$

Dies beweist die Gültigkeit auch in dem Fall $x_{j_1 k} = 0$ und $x_{j_2 k} = 1$.

(iv)

Es bleibt noch der abschließende Fall $x_{j_1 k} = x_{j_2 k} = 0$ zu untersuchen. In diesem Fall wissen wir zunächst, daß $\sum_{i\in S^k} x_{ik} \leq |S^k| - 1$ ist.

Falls $\sum_{i\in S^k} x_{ik} \leq |S^k| - 2$, so folgt die Richtigkeit der Ungleichung sofort. Nehmen wir also an, $\sum_{i\in S^k} x_{ik} = |S^k| - 1$. Dies impliziert daß $x_{sk} = 1$ oder $x_{s'k} = 1$ sein muß. Sei o.B.d.A. $x_{s'k} = 1$. Falls dann $x_{sl} = 1$ oder $\sum_{v\in M\setminus\{k,l\}} x_{sv} = 1$ ist, so muß auch $y_{ss'} = 1$ sein. Somit gilt: $\sum_{i\in S^l} x_{il} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} \leq |S^l| - 2$.

Durch Aufsummieren erhält man:
$\sum_{u\in S^k} x_{uk} + \sum_{u\in S^l} x_{ul} + \sum_{v\in M\setminus\{k,l\}} x_{sv} - y_{ss'} =$
$2(|S^k| - 1) + |S^l| - 2 = 2|S^k| + |S^l| - 4.$

Dies beweist die Gültigkeit der Ungleichung (15.6.3).

Ungleichung (15.6.3) definiert eine Facette für das Polyeder $\mathbf{C}^r(G, f, F)$.
Angenommen, $EQ((15.6.3)) \subseteq EQ(a'^T x + b^T y \leq \beta)$, wobei die Ungleichung $a'^T x + b^T y \leq \beta$ eine Facette für $\mathbf{C}^r(G, f, F)$ definiert.

(1)

Sei $i \in S^k$. Dann sind die Vektoren $(\sum_{u\in S^k\setminus\{i\}} e_{uk} + \sum_{i\in S^l\setminus\{s,s'\}}, 0)$ in der Seitenfläche $EQ((15.6.3))$ enthalten. Dies liefert die Bedingungen $a'_{ik} = a'_{jk}$ für alle $i \neq j,\ i, j \in S^k$.

(2)

Sei $i \in S^l$. Dann gilt: $2\sum_{u\in D_2^*} 1 + \sum_{u\in S^l\setminus\{i\}} 1 + a_{j_2 k} = 2|D_2^*| + |S^l| - 1 + 2|S^k| - 2|D_2^*| - 3 = 2|S^k| + |S^l| - 4$.
Damit ist der Vektor $(\sum_{u\in D_2^*} e_{uk} + e_{j_2 k} + \sum_{j\in S^l\setminus\{i\}}, 0)$ ein Element der Facette $EQ(a'^T x + b^T y \leq \beta)$.
Da dies für alle $i \in S^l$ gilt, folgt $a'_{il} = a'_{jl}$ für alle $i \neq j,\ i, j \in S^l$.

(3)

Wir zeigen nun, daß $a'_{sk} = 2a'_{sl}$. Zusammen mit (1) und (2) folgt daraus: $a'_{ik} = 2a'_{jl}$ für alle $i \in S^k, j \in S^l,\ j \neq j_1$.
Es gilt: $2\sum_{u\in D_1^*\cup\{s\}} 1 + \sum_{u\in S^l\setminus\{j_1,s,s'\}} 1 + a_{j_1 k} = 2|D_1^*| + 2 + |S^l| - 3 + 2|S^k| - 2|D_1^*| - 3 = 2|S^k| + |S^l| - 4 = 2|D_1^*| + |S^l| - 1 + 2|S^k| - 2|D_1^*| - 3 = 2\sum_{u\in D_1^*} 1 + \sum_{u\in S^l\setminus\{j_1\}} 1 + a_{j_1 k}$.
Damit sind die beiden Vektoren $(\sum_{u\in D_1^*} e_{uk} + e_{sk} + e_{j_1 k} + \sum_{j\in S^l\setminus\{j_1,s,s'\}}, 0)$ und $(\sum_{u\in D_1^*} e_{uk} + e_{j_1 k} + \sum_{j\in S^l\setminus\{j_1\}}, 0)$ in der Facette $EQ(a'^T x + b^T y \leq \beta)$ enthalten.
Daraus folgt schließlich $a'_{sk} = a'_{sl} + a'_{s'l} = 2a'_{sl}$.

(4)

Um den Koeffizienten $b_{ss'}$ zu fixieren, betrachten wir folgende zwei Lösungen:
$v_1 = (\sum_{u \in S^k \setminus \{s\}} e_{uk} + \sum_{j \in S^l \setminus \{s,s'\}}, 0)$ und $v_2 = (\sum_{u \in S^k \setminus \{s\}} e_{uk} + \sum_{j \in S^l \setminus \{s'\}}, e_{ss'})$. Beide Vektoren erfüllen Ungleichung (15.6.3) mit Gleichheit. Dies impliziert, daß $b_{ss'} + a'_{sl} = 0$ ist. Daraus folgt $b_{ss'} = -a'_{sl}$

(5)

Aus (1) bis (4) wissen wir, daß die Vektoren $(\sum_{u \in D_2^*} e_{uk} + e_{j_2 k} + \sum_{j \in S^l \setminus \{s\}} e_{jl}, 0)$ und $(\sum_{u \in S^k \setminus \{s\}} e_{uk} + \sum_{j \in S^l \setminus \{s,s'\}} e_{jl}, 0)$ in der Facette $EQ(a'^T x + b^T y \leq \beta)$ enthalten sind. Daraus können wir ableiten, daß
$2|D_2^*|a'_{sl} + (|S^l| - 1)a'_{sl} + a'_{j_2 k} = 2(|S^k| - 1)a'_{sl} + (|S^l| - 2)a'_{sl}$. Löst man diese Gleichung bezüglich $a'_{j_2 k}$ auf, so erhalten wir $a'_{j_2 k} = (2|S^k| - 2|D_2^*| - 3)a'_{sl}$.

Analog bestimmen wir den Koeffizienten $a'_{j_1 k}$.
Die Vektoren $(\sum_{u \in D_1^*} e_{uk} + e_{j_1 k} + \sum_{j \in S^l \setminus \{s\}} e_{jl}, 0)$ und $(\sum_{u \in S^k \setminus \{s\}} e_{uk} + \sum_{j \in S^l \setminus \{s,s'\}} e_{jl}, 0)$ sind in der Facette $EQ(a'^T x + b^T y \leq \beta)$ enthalten.
Dies liefert die Beziehung $a'_{j_1 k} = (2|S^k| - 2|D_1^*| - 3)a'_{sl}$.

(6)

Es bleibt noch zu zeigen, daß $a'_{sv} = a'_{sl}$ für alle $v \in M \setminus \{k, l\}$. Sei also $v \in M \setminus \{k, l\}$. Wir betrachten die beiden Lösungen $(\sum_{u \in S^k \setminus \{s\}} e_{uk} + \sum_{j \in S^l \setminus \{s'\}} e_{jl}, e_{ss'})$ und $(\sum_{u \in S^k \setminus \{s\}} e_{uk} + \sum_{j \in S^l \setminus \{s,s'\}} e_{jl} + e_{sv}, e_{ss'})$. Beide Vektoren erfüllen Ungleichung (15.6.3) mit Gleichheit.
Daraus folgt $a'_{sv} = a'_{sl}$.

(7)

Sei $v \in M \setminus \{k\}$. Sei ferner $i \in S^k \setminus \{s, s'\}$. Dann sind die Vektoren $(\sum_{u \in S^k \setminus \{i\}} e_{uk} + \sum_{j \in S^l \setminus \{s,s'\}} e_{jl}, 0)$ und $(\sum_{u \in S^k \setminus \{i\}} e_{uk} + \sum_{j \in S^l \setminus \{s,s'\}} e_{jl} + e_{iv}, 0)$ in der Seitenfläche $EQ((15.6.3))$ enthalten. Daraus ergibt sich, daß $a'_{iv} = 0$ ist.

(8)

Sei $v \in M \setminus \{k\}$. Für ein $i \in S^l \setminus \{s, s'\}$ sind die Vektoren $(\sum_{u \in D_2^*} e_{uk} + e_{j_2 k} + \sum_{j \in S^l \setminus \{i\}}, 0)$ und $(\sum_{u \in D_2^*} e_{uk} + e_{j_2 k} + \sum_{j \in S^l \setminus \{i\}} e_{jl} + e_{iv}, 0)$ in der Seitenfläche $EQ((15.6.3))$ enthalten. Dies liefert auch in diesem Fall $a'_{iv} = 0$.

(9)

Sei $i \in S^l \setminus \{s, s', j_1\}$. Nach Voraussetzung sind die Vektoren $v_1 = (\sum_{u \in D_2^*} e_{uk} + e_{j_2 k} + \sum_{j \in S^l \setminus \{i\}}, 0)$ und $v_2 = v_1 + (e_{ik}, 0)$ in der Seitenfläche $EQ((15.6.3))$ enthalten. Dies beweist, daß $a'_{ik} = 0$ ist für alle $i \in S^l \setminus \{s, s', j_1\}$.

(10)

Wir zeigen nun, daß $a'_{j_2 v} = a'_{j_2 l} = 0$ für alle $v \in M \setminus \{k, l\}$. Sei $v \in M \setminus \{k\}$. Wir betrachten die Lösungen $v_1 = (\sum_{u \in S^k \setminus \{s\}} e_{uk} + \sum_{j \in S^l \setminus \{s,s'\}} e_{jl}, 0)$ und $v_2 = v_1 + (e_{j_2 v}, 0)$. Beide Vektoren erfüllen Ungleichung (15.6.3) mit Gleichheit. Daraus folgt die Behauptung.

(11)

Schließlich bleibt uns noch die Aufgabe, die Koeffizienten $a'_{s'v}$ für alle $v \in M \setminus \{k,l\}$ zu fixieren. Sei $v \in M \setminus \{k,l\}$ beliebig.

Wir wählen die Lösungen $(\sum_{u \in D_2^*} e_{uk} + e_{j_2 k} + \sum_{j \in S^l \setminus \{s\}}, 0)$ und $(\sum_{u \in D_2^*} e_{uk} + e_{j_2 k} + \sum_{j \in S^l \setminus \{s,s'\}} + e_{sv} + e_{s'v}, 0)$. Beide Lösungen gehören zu der durch (15.6.3) definierten Seitenfläche. Da $a'_{sv} = a'_{sl}$ gilt, ergibt sich $a'_{s'v} = 0$ für alle $v \in M \setminus \{k,l\}$.

Damit ist der Beweis von Satz (15.6.2) abgeschlossen.

□

Ausblick

In den vorhergehenden Kapiteln haben wir ein ganzzahliges quadratisches Modell für das Plazierungsproblem untersucht und uns mit verschiedenen Ansätzen zur Lösung des Clusteringproblems beschäftigt. An dieser Stelle möchten wir die Diskussionen mit einem kurzen Überblick über noch offene Probleme beziehungsweise zukünftige Forschungsthemen abschließen.

Zunächst hat sich gezeigt, daß zur Lösung der quadratischen 0/1 Probleme (P), (P^+), (P_2) und (P_3^+) keine beweisbar guten, polynomialen Algorithmen existieren. Hingegen konnten wir zur Lösung von (P_m^-), $m \geq 2$ einen $\frac{1}{m^2}$ approximativen Algorithmus angeben. Ungelöst bleibt dies für die Probleme (P_2^+) und (P^-). Wir vermuten, daß es zu deren Lösung ebenfalls keine ϵ-approximativen Algorithmen gibt, jedoch konnten wir das bisher weder beweisen noch widerlegen.
Zur Lösung unseres Plazierungsmodells haben wir verschiedene Dekompositionsverfahren und Heuristiken entwickelt. Diese liefern im Vergleich zu "state of the art"-Plazierungsverfahren gute Ergebnisse, benötigen aber wesentlich mehr Laufzeit. Diesbezüglich ist es sicherlich eine Herausforderung, schnelle Algorithmen zu entwerfen, die ähnlich gute Ergebnisse liefern.
Schließlich haben wir uns in Teil 3 unter anderem mit einem polyedrischen Ansatz zum Clusteringproblem beschäftigt und dabei mehrere Klassen von facetten-definierenden Ungleichungen vorgestellt. In diesem Zusammenhang stellt sich zunächst die Frage nach der Lösbarkeit der zugehörigen Separierungsprobleme. Ferner bleibt es zukünftigen Arbeiten überlassen, einen Schnittebenenansatz zu entwerfen, mit dessen Hilfe das Clusteringproblem für praktische Instanzen gelöst werden kann.

Anhang I

Testserie zu Kapitel 7.1

Testserie für die Schaltung soc_2

Vergleich der Heuristiken in der 1. Iteration

λ =0.000

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	37950.07	34101872.00	37950.07	1:52
Heuristik 2a	104034.98	7038083.00	104034.98	16:11
Heuristik 2b	108931.92	6891304.00	108931.92	1:16
Heuristik 3	112723.99	10684430.00	112723.99	0:24

Tabelle 7.1.1

λ =0.005

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	162005.20	3548794.00	179749.17	1:53
Heuristik 2a	92544.62	6649278.00	125791.01	15:55
Heuristik 2b	108254.98	5748370.00	136996.83	1:18
Heuristik 3	145909.62	4933083.00	170575.04	0:22

Tabelle 7.1.2

λ =0.010

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	164142.09	3464162.00	198783.71	1:51
Heuristik 2a	108239.84	4404858.00	152288.42	16:04
Heuristik 2b	119923.45	4523759.00	165161.03	1:19
Heuristik 3	150897.91	4494674.00	195844.65	0:23

Tabelle 7.1.3

λ =0.020

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	177647.41	3377195.00	245191.30	1:52
Heuristik 2a	115831.35	3803475.00	191900.85	15:59
Heuristik 2b	137707.34	3597566.00	209658.66	1:21
Heuristik 3	167609.19	3678570.00	241180.59	0:23

Tabelle 7.1.4

λ =0.030

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	178684.47	3395918.00	280562.01	1:52
Heuristik 2a	140958.95	3529062.00	246830.81	16:00
Heuristik 2b	142297.34	3545029.00	248648.21	1:22
Heuristik 3	166603.91	3601852.00	274659.46	0:24

Tabelle 7.1.5

λ =0.040

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	175072.53	3395469.00	310891.29	1:52
Heuristik 2a	130643.65	3630821.00	275876.49	15:56
Heuristik 2b	150038.47	3476558.00	289100.79	1:23
Heuristik 3	176618.77	3598365.00	320553.36	0:23

Tabelle 7.1.6

λ =0.050

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	168885.84	3513932.00	344582.45	1:53
Heuristik 2a	126377.67	3577735.00	305264.42	16:04
Heuristik 2b	149238.33	3489141.00	323695.38	1:24
Heuristik 3	177558.11	3581612.00	356638.71	0:24

Tabelle 7.1.7

λ =0.100

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	166468.28	3538745.00	520342.79	1:55
Heuristik 2a	128594.70	3577056.00	486300.30	16:11
Heuristik 2b	154026.88	3468343.00	500861.18	1:25
Heuristik 3	193614.25	3486670.00	542281.26	0:33

Tabelle 7.1.8

λ =0.200

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	187788.28	3346797.00	857147.69	1:57
Heuristik 2a	144199.61	3479370.00	840073.62	16:06
Heuristik 2b	162574.92	3509103.00	864395.53	1:30
Heuristik 3	185381.47	3671397.00	919660.88	0:26

Tabelle 7.1.9

λ =0.500

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	207566.34	3392324.00	1903728.34	2:00
Heuristik 2a	163641.33	3337534.00	1832408.33	16:03
Heuristik 2b	172473.69	3389341.00	1867144.19	1:36
Heuristik 3	200948.58	3416768.00	1909332.58	0:27

Tabelle 7.1.10

λ =1.000

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	252589.58	3292949.00	3545538.58	1:57
Heuristik 2a	162646.34	3326057.00	3488703.34	16:11
Heuristik 2b	186753.95	3329741.00	3516494.95	1:32
Heuristik 3	249586.62	3350949.00	3600535.62	0:27

Tabelle 7.1.11

λ =5.000

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	329008.78	3249696.00	16577488.78	1:52
Heuristik 2a	202969.58	3263630.00	16521119.58	16:02
Heuristik 2b	223340.97	3309058.00	16768630.97	1:29
Heuristik 3	311208.19	3405840.00	17340408.19	0:24

Tabelle 7.1.12

λ =10.000

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	368563.81	3250620.00	32874763.81	1:52
Heuristik 2a	216077.69	3264024.00	32856317.69	16:04
Heuristik 2b	234095.27	3310294.00	33337035.27	1:30
Heuristik 3	358453.88	3364370.00	34002153.88	0:23

Tabelle 7.1.13

λ =100.000

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	409291.47	3251448.00	325554091.47	1:53
Heuristik 2a	293404.56	3260451.00	326338504.56	14:39
Heuristik 2b	257312.12	3351668.00	335424112.12	1:30
Heuristik 3	440795.22	3396723.00	340113095.22	0:22

Tabelle 7.1.14

Vergleich der Heuristiken in der 2. Iteration

Als Lösung für die erste Iteration wählten wir das Ergebnis, welches von Heuristik 2a mit $\lambda = 0.02$ geliefert wurde (vergleiche Tabelle 7.1.4).

$\lambda = 0.010$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	246537.20	3321568.00	279752.88	1:49
Heuristik 2a	144639.97	4228852.00	186928.49	18:54
Heuristik 2b	171515.36	4290194.00	214417.30	1:2
Heuristik 3	206374.58	3537159.00	241746.17	0:19

Tabelle 7.1.15

$\lambda = 0.020$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	234184.83	3280135.00	299787.53	1:54
Heuristik 2a	127683.00	3738513.00	202453.26	19:20
Heuristik 2b	170908.50	3751342.00	245935.34	1:28
Heuristik 3	208608.66	3587504.00	280358.73	0:2

Tabelle 7.1.16

$\lambda = 0.030$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	234806.83	3266243.00	332794.12	1:48
Heuristik 2a	136310.02	3534792.00	242353.77	19:31
Heuristik 2b	161097.61	3628568.00	269954.65	1:30
Heuristik 3	210786.94	3592303.00	318556.03	0:19

Tabelle 7.1.17

$\lambda = 0.040$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	234247.53	3270973.00	365086.45	1:49
Heuristik 2a	137856.62	3515267.00	278467.30	18:56
Heuristik 2b	164277.67	3583777.00	307628.75	1:29
Heuristik 3	218041.11	3582507.00	361341.39	0:19

Tabelle 7.1.18

$\lambda = 0.050$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	235613.77	3257108.00	398469.16	1:51
Heuristik 2a	139815.69	3415675.00	310599.43	19:05
Heuristik 2b	162813.14	3502088.00	337917.53	1:30
Heuristik 3	213234.17	3530427.00	389755.51	0:18

Tabelle 7.1.19

$\lambda = 0.060$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	236098.97	3256359.00	431480.50	1:47
Heuristik 2a	142317.12	3361241.00	343991.58	19:09
Heuristik 2b	165062.28	3459610.00	372638.88	1:32
Heuristik 3	214758.06	3478186.00	423449.22	0:19

Tabelle 7.1.20

$\lambda = 0.070$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	239664.33	3259862.00	467854.67	1:47
Heuristik 2a	142065.70	3357932.00	377120.94	18:59
Heuristik 2b	168796.59	3425575.00	408586.84	1:30
Heuristik 3	219228.44	3480561.00	462867.71	0:19

Tabelle 7.1.21

$\lambda = 0.080$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	239128.14	3251161.00	499221.01	1:49
Heuristik 2a	137093.86	3378222.00	407351.61	18:58
Heuristik 2b	171221.84	3403653.00	443514.08	1:30
Heuristik 3	225533.53	3449291.00	501476.81	0:19

Tabelle 7.1.22

$\lambda = 0.090$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	236275.12	3246018.00	528416.73	1:48
Heuristik 2a	147674.80	3373620.00	451300.58	19:03
Heuristik 2b	174319.09	3361617.00	476864.61	1:31
Heuristik 3	218887.27	3466907.00	530908.88	0:19

Tabelle 7.1.23

$\lambda = 0.100$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	235722.83	3256894.00	561412.21	1:59
Heuristik 2a	138027.36	3350872.00	473114.54	19:02
Heuristik 2b	178044.16	3338469.00	511891.04	1:31
Heuristik 3	230393.50	3416013.00	571994.78	0:18

Tabelle 7.1.24

$\lambda = 0.110$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	249796.06	3248479.00	607128.75	1:47
Heuristik 2a	152036.53	3332944.00	518660.37	18:57
Heuristik 2b	177383.08	3341129.00	544907.27	1:31
Heuristik 3	223189.09	3434261.00	600957.80	0:19

Tabelle 7.1.25

$\lambda = 0.120$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	241371.12	3249188.00	631273.68	1:50
Heuristik 2a	145407.91	3367501.00	549508.02	19:18
Heuristik 2b	181265.75	3318693.00	579508.90	1:36
Heuristik 3	234973.61	3385725.00	641260.60	0:19

Tabelle 7.1.26

$\lambda = 0.160$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	249135.11	3239397.00	767438.62	1:46
Heuristik 2a	146614.42	3382356.00	687791.37	18:57
Heuristik 2b	185506.98	3294593.00	712641.85	1:32
Heuristik 3	228349.72	3365364.00	766807.95	0:19

Tabelle 7.1.27

$\lambda = 0.200$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	254032.09	3236417.00	901315.46	1:59
Heuristik 2a	157506.80	3312731.00	820052.96	18:55
Heuristik 2b	190331.98	3281473.00	846626.55	1:31
Heuristik 3	238697.66	3342195.00	907136.62	0:19

Tabelle 7.1.28

$\lambda = 0.240$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	253387.09	3236206.00	1030076.52	1:53
Heuristik 2a	166802.20	3276155.00	953079.39	19:11
Heuristik 2b	194039.91	3270912.00	979058.77	1:32
Heuristik 3	241871.91	3297621.00	1033300.93	0:19

Tabelle 7.1.29

$\lambda = 0.280$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	259312.12	3234726.00	1165035.41	1:48
Heuristik 2a	163960.23	3271208.00	1079898.48	19:08
Heuristik 2b	197964.14	3262172.00	1111372.30	1:32
Heuristik 3	244019.84	3318689.00	1173252.77	0:19

Tabelle 7.1.30

$\lambda = 0.320$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	268235.56	3229626.00	1301715.86	1:49
Heuristik 2a	170770.58	3253567.00	1211911.99	19:06
Heuristik 2b	201619.08	3254844.00	1243169.13	1:32
Heuristik 3	252144.56	3314766.00	1312869.66	0:19

Tabelle 7.1.31

Vergleich der Heuristiken in der 3. Iteration

Als Verfahren für die ersten zwei Iterationen wählten wir Heuristik 2a. Der Parameter λ wurde in der ersten Iteration auf 0.02 und in der zweiten Iteration auf 0.08 gesetzt (vergleiche Tabellen 7.1.4 und 7.1.22).

$\lambda = 0.027$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	102061.02	310175.00	110332.36	1:21
Heuristik 2a	91655.14	418938.00	102826.82	1:32
Heuristik 2b	95073.83	381287.00	105241.48	0:49
Heuristik 3	103595.81	299922.00	111593.73	0:25

Tabelle 7.1.32

$\lambda = 0.053$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	111074.34	254197.00	124631.52	1:21
Heuristik 2a	96616.92	343919.00	114959.27	1:32
Heuristik 2b	99403.36	319616.00	116449.55	0:46
Heuristik 3	110456.92	247378.00	123650.41	0:28

Tabelle 7.1.33

$\lambda = 0.080$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	115056.92	230587.00	133503.88	1:21
Heuristik 2a	102096.01	305478.00	126534.25	1:32
Heuristik 2b	102525.95	293157.00	125978.50	0:46
Heuristik 3	116445.23	220897.00	134116.99	0:24

Tabelle 7.1.34

$\lambda = 0.107$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	119960.01	208660.00	142217.07	1:23
Heuristik 2a	106713.28	269103.00	135417.60	1:32
Heuristik 2b	106935.38	263282.00	135018.79	0:46
Heuristik 3	119632.44	216679.00	142744.86	0:27

Tabelle 7.1.35

$\lambda = 0.133$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	122226.88	199281.00	148797.67	1:22
Heuristik 2a	111087.52	252052.00	144694.45	1:37
Heuristik 2b	111496.88	241729.00	143727.41	0:46
Heuristik 3	123329.32	207559.00	151003.85	0:26

Tabelle 7.1.36

$\lambda = 0.160$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	125599.26	198281.00	157324.22	1:21
Heuristik 2a	114748.73	228988.00	151386.81	1:34
Heuristik 2b	114444.68	238469.00	152599.72	0:46
Heuristik 3	124652.70	202104.00	156989.33	0:24

Tabelle 7.1.37

$\lambda = 0.187$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	127811.09	192128.00	163674.98	1:26
Heuristik 2a	117146.76	213199.00	156943.90	1:34
Heuristik 2b	116585.86	223536.00	158312.58	0:49
Heuristik 3	127464.26	197645.00	164357.99	0:26

Tabelle 7.1.38

$\lambda = 0.213$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	128966.97	194631.00	170488.25	1:23
Heuristik 2a	118547.42	216098.00	164648.33	1:34
Heuristik 2b	119184.65	214067.00	164852.27	0:47
Heuristik 3	128697.47	194048.00	170094.37	0:25

Tabelle 7.1.39

$\lambda = 0.240$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	131348.08	188202.00	176516.56	1:23
Heuristik 2a	120281.53	211476.00	171035.77	1:39
Heuristik 2b	118447.12	212243.00	169385.44	0:56
Heuristik 3	131274.62	192752.00	177535.10	0:25

Tabelle 7.1.40

$\lambda = 0.267$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	132902.84	182485.00	181565.51	1:24
Heuristik 2a	124497.11	202846.00	178589.37	1:35
Heuristik 2b	121217.71	204006.00	175619.31	0:52
Heuristik 3	133998.27	188049.00	184144.66	0:24

Tabelle 7.1.41

$\lambda = 0.293$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	133190.86	185246.00	187529.68	1:23
Heuristik 2a	122939.47	199013.00	181316.61	1:35
Heuristik 2b	125074.20	203684.00	184821.50	0:47
Heuristik 3	134203.59	188398.00	189467.00	0:25

Tabelle 7.1.42

$\lambda = 0.320$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	135147.73	180994.00	193065.81	1:20
Heuristik 2a	127063.93	196052.00	189800.57	1:35
Heuristik 2b	122711.81	200184.00	186770.69	0:49
Heuristik 3	136083.19	185368.00	195400.95	0:28

Tabelle 7.1.43

$\lambda = 0.347$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	134589.95	178412.00	196439.45	1:21
Heuristik 2a	128818.41	192378.00	195509.45	1:36
Heuristik 2b	124973.88	202009.00	195003.67	0:48
Heuristik 3	136234.23	185530.00	200551.30	0:26

Tabelle 7.1.44

$\lambda = 0.373$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	136515.59	177406.00	202747.17	1:19
Heuristik 2a	128258.38	190035.00	199204.78	1:37
Heuristik 2b	128483.52	199404.00	202927.68	0:48
Heuristik 3	139013.70	179598.00	206063.62	0:26

Tabelle 7.1.45

$\lambda = 0.400$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	139524.53	177719.00	210612.13	1:21
Heuristik 2a	129783.74	191456.00	206366.14	1:37
Heuristik 2b	130208.85	197442.00	209185.65	0:56
Heuristik 3	140270.50	181448.00	212849.70	0:26

Tabelle 7.1.46

$\lambda = 0.427$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	137796.44	176583.00	213138.51	1:28
Heuristik 2a	131660.25	185121.00	210645.21	1:38
Heuristik 2b	129473.95	194253.00	212355.22	0:48
Heuristik 3	141274.94	183137.00	219413.39	0:26

Tabelle 7.1.47

$\lambda = 0.453$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	141027.11	174236.00	220014.09	1:25
Heuristik 2a	132871.61	182492.00	215601.31	1:36
Heuristik 2b	127511.84	194211.00	215554.16	0:46
Heuristik 3	143275.08	178978.00	224411.77	0:27

Tabelle 7.1.48

$\lambda = 0.480$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	141092.59	174060.00	224641.39	1:23
Heuristik 2a	133960.73	180739.00	220715.45	1:37
Heuristik 2b	129007.45	192157.00	221242.80	0:47
Heuristik 3	143703.73	177355.00	228834.13	0:31

Tabelle 7.1.49

Vergleich der Heuristiken in der 4. Iteration

Die ersten drei Iterationen sind mit Heuristik 2a gerechnet. Der Parameter λ wurde in der ersten Iteration auf 0.02, in der zweiten Iteration auf 0.08 und in der dritten auf 0.32 gesetzt (vergleiche Tabellen 7.1.4, 7.1.22 und 7.1.43).

$\lambda = 0.098$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	108297.89	37092.00	111920.54	2:00
Heuristik 2a	100127.87	48851.00	104898.98	1:02
Heuristik 2b	100333.06	47664.00	104988.25	0:55
Heuristik 3	110013.20	32861.00	113222.62	0:30

Tabelle 7.1.50

$\lambda = 0.195$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112990.80	26670.00	118200.35	1:58
Heuristik 2a	104313.30	39051.00	111941.26	1:03
Heuristik 2b	104267.30	38309.00	111750.32	0:56
Heuristik 3	114308.23	26021.00	119391.00	0:29

Tabelle 7.1.51

$\lambda = 0.293$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116684.70	23346.00	123525.07	1:59
Heuristik 2a	107760.80	33092.00	117456.76	1:01
Heuristik 2b	106387.52	33142.00	116098.13	0:56
Heuristik 3	118419.84	22648.00	125055.70	0:29

Tabelle 7.1.52

$\lambda = 0.391$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	118856.77	22553.00	127667.47	2:00
Heuristik 2a	109877.34	28792.00	121125.42	1:01
Heuristik 2b	109390.00	30411.00	121270.56	0:56
Heuristik 3	121894.57	20622.00	129950.90	0:28

Tabelle 7.1.53

$\lambda = 0.488$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	120683.58	21072.00	130973.74	1:59
Heuristik 2a	111894.66	28517.00	125820.46	1:01
Heuristik 2b	111289.49	28690.00	125299.78	0:56
Heuristik 3	122437.59	20415.00	132406.91	0:27

Tabelle 7.1.54

$\lambda = 0.586$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	122575.91	19917.00	134247.28	1:59
Heuristik 2a	114577.58	25699.00	129637.19	1:01
Heuristik 2b	112084.13	25943.00	127286.73	0:56
Heuristik 3	125468.34	19552.00	136925.81	0:28

Tabelle 7.1.55

$\lambda = 0.684$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	123803.99	18898.00	136723.93	1:59
Heuristik 2a	115290.42	24525.00	132057.35	1:01
Heuristik 2b	112586.98	25647.00	130120.98	1:01
Heuristik 3	127092.45	19519.00	140436.94	0:27

Tabelle 7.1.56

$\lambda = 0.781$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	124565.47	19129.00	139511.60	1:59
Heuristik 2a	117291.76	24059.00	136089.86	1:01
Heuristik 2b	114164.66	25952.00	134441.82	0:57
Heuristik 3	127595.75	19503.00	142834.10	0:28

Tabelle 7.1.57

$\lambda = 0.879$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	126052.48	18557.00	142364.09	1:59
Heuristik 2a	117625.34	23237.00	138050.66	1:01
Heuristik 2b	114545.43	25358.00	136835.11	0:56
Heuristik 3	128805.46	18993.00	145500.31	0:28

Tabelle 7.1.58

$\lambda = 0.977$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	126986.32	18073.00	144637.62	1:59
Heuristik 2a	120031.14	22100.00	141615.47	1:00
Heuristik 2b	115215.05	24368.00	139014.47	0:56
Heuristik 3	130339.12	18719.00	148621.35	0:27

Tabelle 7.1.59

$\lambda = 1.074$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	127869.58	17721.00	146907.84	1:58
Heuristik 2a	120717.78	20968.00	143244.40	1:00
Heuristik 2b	116258.29	23687.00	141706.03	0:57
Heuristik 3	131102.22	18386.00	150854.91	0:28

Tabelle 7.1.60

$\lambda = 1.172$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	129154.70	17677.00	149872.14	1:58
Heuristik 2a	122190.20	20587.00	146318.16	1:01
Heuristik 2b	116578.57	24553.00	145354.69	0:56
Heuristik 3	131860.56	18155.00	153138.22	0:28

Tabelle 7.1.61

$\lambda = 1.270$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	131491.78	17985.00	154326.74	1:58
Heuristik 2a	121670.50	19992.00	147053.68	1:04
Heuristik 2b	117893.74	25209.00	149900.77	0:56
Heuristik 3	132165.78	17880.00	154867.42	0:27

Tabelle 7.1.62

$\lambda = 1.367$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	130439.36	17116.00	153842.64	1:58
Heuristik 2a	122852.58	19303.00	149246.21	1:00
Heuristik 2b	119505.57	23439.00	151554.50	0:56
Heuristik 3	132223.72	17880.00	156671.64	0:27

Tabelle 7.1.63

$\lambda = 1.465$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	132198.06	17600.00	157982.06	1:59
Heuristik 2a	123774.81	18541.00	150937.38	1:00
Heuristik 2b	118293.80	23047.00	152057.66	0:55
Heuristik 3	133271.64	18430.00	160271.59	0:28

Tabelle 7.1.64

$\lambda = 1.563$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	132308.06	16984.00	158848.39	1:58
Heuristik 2a	125067.17	18027.00	153237.37	1:01
Heuristik 2b	118716.10	24140.00	156438.88	0:56
Heuristik 3	133241.89	18320.00	161869.95	0:28

Tabelle 7.1.65

$\lambda = 1.660$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	132085.64	16643.00	159718.57	1:59
Heuristik 2a	124110.69	19639.00	156717.98	1:01
Heuristik 2b	120946.50	23494.00	159954.37	0:56
Heuristik 3	133880.50	18210.00	164115.17	0:28

Tabelle 7.1.66

$\lambda = 1.758$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	133276.41	16544.00	162360.7	1:59
Heuristik 2a	125195.42	18925.00	158465.5	1:00
Heuristik 2b	120614.53	21390.00	158218.15	0:56
Heuristik 3	134400.31	18045.00	166123.4	0:27

Tabelle 7.1.67

Vergleich der Heuristiken in der 5. Iteration

Die ersten vier Iterationen sind mit Heuristik 2a gerechnet. Der Parameter λ wurde in der ersten Iteration auf 0.02, in der zweiten Iteration auf 0.08, in der dritten auf 0.32 und in der vierten auf 1.27 gesetzt (vergleiche Tabellen 7.1.4, 7.1.22, 7.1.43 und 7.1.62).

$\lambda = 0.358$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110539.77	5384.00	112467.25	2:09
Heuristik 2a	106529.97	4937.00	108297.41	1:01
Heuristik 2b	105306.74	5759.00	107368.46	1:12
Heuristik 3	110414.94	3463.00	111654.69	0:35

Tabelle 7.1.68

$\lambda = 0.716$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	111829.40	3732.00	114501.51	2:07
Heuristik 2a	108939.37	2192.00	110508.84	0:59
Heuristik 2b	106923.89	3205.00	109218.67	1:10
Heuristik 3	111189.41	2810.00	113201.37	0:31

Tabelle 7.1.69

$\lambda = 1.074$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112053.74	3679.00	116004.99	2:06
Heuristik 2a	110417.62	2672.00	113287.35	0:58
Heuristik 2b	107616.14	3363.00	111228.00	1:10
Heuristik 3	111619.28	1883.00	113641.62	0:32

Tabelle 7.1.70

$\lambda = 1.432$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112468.39	3495.00	117473.23	2:05
Heuristik 2a	111496.90	2453.00	115009.59	1:05
Heuristik 2b	107931.09	2461.00	111455.24	1:12
Heuristik 3	111755.40	1697.00	114185.50	0:31

Tabelle 7.1.71

$\lambda = 1.790$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112468.39	3495.00	118724.44	2:05
Heuristik 2a	112469.58	1999.00	116047.79	1:00
Heuristik 2b	108363.62	2189.00	112281.93	1:11
Heuristik 3	111699.45	1708.00	114756.77	0:31

Tabelle 7.1.72

$\lambda = 2.148$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112468.39	3495.00	119975.65	2:05
Heuristik 2a	112545.52	1821.00	116457.02	1:00
Heuristik 2b	108828.76	2293.00	113754.12	1:11
Heuristik 3	112297.38	1737.00	116028.45	0:32

Tabelle 7.1.73

$\lambda = 2.506$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112468.39	3495.00	121226.86	2:05
Heuristik 2a	112854.68	1855.00	117503.31	1:00
Heuristik 2b	109133.81	1727.00	113461.67	1:10
Heuristik 3	112335.45	1675.00	116533.00	0:33

Tabelle 7.1.74

$\lambda = 2.864$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112468.39	3495.00	122478.07	2:05
Heuristik 2a	113189.02	1160.00	116511.26	1:00
Heuristik 2b	108950.70	1903.00	114400.89	1:11
Heuristik 3	112335.45	1675.00	117132.65	0:33

Tabelle 7.1.75

$\lambda = 3.222$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112504.60	3495.00	123765.49	2:05
Heuristik 2a	113393.16	1360.00	117775.08	1:02
Heuristik 2b	109327.51	2007.00	115794.06	1:11
Heuristik 3	112335.45	1675.00	117732.30	0:33

Tabelle 7.1.76

$\lambda = 3.580$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112504.60	3495.00	125016.70	2:05
Heuristik 2a	113837.25	1541.00	119354.03	1:04
Heuristik 2b	110063.27	1621.00	115866.45	1:11
Heuristik 3	112335.45	1675.00	118331.95	0:33

Tabelle 7.1.77

$\lambda = 3.938$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112569.70	3495.00	126333.01	2:04
Heuristik 2a	113864.46	1348.00	119172.88	1:01
Heuristik 2b	109665.38	1393.00	115151.02	1:11
Heuristik 3	112322.95	1675.00	118919.10	0:31

Tabelle 7.1.78

$\lambda = 4.296$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112569.70	3495.00	127584.22	2:05
Heuristik 2a	113881.85	1267.00	119324.88	1:00
Heuristik 2b	109573.15	1445.00	115780.87	1:11
Heuristik 3	112335.45	1675.00	119531.25	0:33

Tabelle 7.1.79

$\lambda = 4.654$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112605.93	3495.00	128871.66	2:04
Heuristik 2a	114709.28	2034.00	124175.52	0:59
Heuristik 2b	109879.02	1621.00	117423.16	1:11
Heuristik 3	112335.45	1675.00	120130.89	0:33

Tabelle 7.1.80

$\lambda = 5.012$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112605.93	3495.00	130122.87	2:04
Heuristik 2a	114725.62	1183.00	120654.82	1:00
Heuristik 2b	109810.98	1544.00	117549.50	1:10
Heuristik 3	112335.45	1675.00	120730.55	0:33

Tabelle 7.1.81

$\lambda = 5.370$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112569.70	3495.00	131337.85	2:04
Heuristik 2a	115169.40	939.00	120211.83	0:59
Heuristik 2b	109854.53	1390.00	117318.83	1:13
Heuristik 3	112322.95	1675.00	121317.70	0:31

Tabelle 7.1.82

$\lambda = 5.728$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112569.70	3495.00	132589.06	2:04
Heuristik 2a	114933.02	1325.00	122522.62	0:59
Heuristik 2b	110181.62	1500.00	118773.62	1:11
Heuristik 3	112322.95	1675.00	121917.35	0:31

Tabelle 7.1.83

$\lambda = 6.086$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112569.70	3495.00	133840.27	2:04
Heuristik 2a	115358.88	1039.00	121682.24	0:59
Heuristik 2b	109637.92	1588.00	119302.49	1:10
Heuristik 3	112335.45	1675.00	122529.50	0:33

Tabelle 7.1.84

$\lambda = 6.444$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	112569.70	3495.00	135091.48	2:04
Heuristik 2a	114882.72	1793.00	126436.81	0:59
Heuristik 2b	109828.68	1500.00	119494.68	1:12
Heuristik 3	112315.56	1675.00	123109.26	0:33

Tabelle 7.1.85

Vergleich der Heuristiken in der 6. Iteration

Die ersten fünf Iterationen sind mit Heuristik 2a gerechnet. Der Parameter λ wurde in der ersten Iteration auf 0.02, in der zweiten Iteration auf 0.08, in der dritten auf 0.32, in der vierten auf 1.27 und in der fünften auf 5.012 gesetzt (vergleiche Tabellen 7.1.4, 7.1.22, 7.1.43 , 7.1.62 und 7.1.81).

$\lambda = 1.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110207.94	3737.00	115190.60	2:10
Heuristik 2a	108398.87	2005.00	111072.20	1:03
Heuristik 2b	105756.42	3375.00	110256.42	1:31
Heuristik 3	109503.35	2514.00	112855.35	0:37

Tabelle 7.1.86

$\lambda = 2.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	119876.41	2:10
Heuristik 2a	109946.73	1612.00	114245.39	1:05
Heuristik 2b	106580.70	2489.00	113218.03	1:32
Heuristik 3	109104.91	2181.00	114920.91	0:36

Tabelle 7.1.87

$\lambda = 4.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	124741.74	2:10
Heuristik 2a	110468.56	563.00	112720.56	1:07
Heuristik 2b	106877.09	2791.00	118041.09	1:34
Heuristik 3	109969.55	2059.00	118205.55	0:44

Tabelle 7.1.88

$\lambda = 5.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	129607.08	2:10
Heuristik 2a	111257.44	792.00	115481.44	1:05
Heuristik 2b	106948.48	2735.00	121535.15	1:30
Heuristik 3	109970.02	2191.00	121655.36	0:44

Tabelle 7.1.89

$\lambda = 6.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110156.41	3649.00	134483.08	2:10
Heuristik 2a	111304.95	514.00	114731.61	1:05
Heuristik 2b	107502.73	2431.00	123709.39	1:30
Heuristik 3	109920.47	2176.00	124427.14	0:44

Tabelle 7.1.90

$\lambda = 8.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	139337.74	2:10
Heuristik 2a	112168.89	44.00	112520.89	1:10
Heuristik 2b	107836.38	1976.00	123644.38	1:29
Heuristik 3	110579.36	2064.00	127091.36	0:38

Tabelle 7.1.91

$\lambda = 9.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.73	3649.00	144203.07	2:10
Heuristik 2a	112619.33	239.00	114849.99	1:05
Heuristik 2b	107768.26	1980.00	126248.26	1:35
Heuristik 3	109859.88	1995.00	128479.88	0:44

Tabelle 7.1.92

$\lambda = 10.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	149068.41	2:10
Heuristik 2a	112571.88	187.00	114566.55	1:03
Heuristik 2b	107276.38	1983.00	128428.38	1:33
Heuristik 3	110498.72	1560.00	127138.72	0:38

Tabelle 7.1.93

$\lambda = 12.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	153933.74	2:10
Heuristik 2a	112553.65	286.00	115985.65	1:03
Heuristik 2b	107589.85	1960.00	131109.85	1:32
Heuristik 3	110550.84	1986.00	134382.84	0:36

Tabelle 7.1.94

$\lambda = 13.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110156.41	3649.00	158809.75	2:10
Heuristik 2a	112467.49	385.00	117600.83	1:05
Heuristik 2b	107707.93	1884.00	132827.93	1:30
Heuristik 3	110074.38	1295.00	127341.05	0:36

Tabelle 7.1.95

$\lambda = 14.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	163664.41	2:10
Heuristik 2a	112748.59	445.00	119275.26	1:10
Heuristik 2b	107794.93	1801.00	134209.60	1:33
Heuristik 3	110069.04	1295.00	129062.37	0:36

Tabelle 7.1.96

$\lambda = 16.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	168529.74	2:10
Heuristik 2a	113377.81	286.00	117953.81	1:06
Heuristik 2b	108187.30	1857.00	137899.30	1:33
Heuristik 3	110005.84	1295.00	130725.84	0:43

Tabelle 7.1.97

$\lambda = 17.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	173395.08	2:11
Heuristik 2a	112939.13	341.00	118849.80	1:06
Heuristik 2b	107955.27	1901.00	140905.93	1:31
Heuristik 3	109995.19	1295.00	132441.85	0:43

Tabelle 7.1.98

$\lambda = 18.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.73	3649.00	178260.40	2:11
Heuristik 2a	112653.34	242.00	117170.68	1:08
Heuristik 2b	108042.06	2175.00	148642.06	1:32
Heuristik 3	109995.19	1295.00	134168.52	0:43

Tabelle 7.1.99

$\lambda = 20.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	183125.74	2:11
Heuristik 2a	113060.92	0.00	113060.92	1:04
Heuristik 2b	108025.34	2087.00	149765.34	1:33
Heuristik 3	109995.19	1295.00	135895.19	0:43

Tabelle 7.1.100

$\lambda = 21.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	187991.08	2:11
Heuristik 2a	112437.51	0.00	112437.51	1:06
Heuristik 2b	108096.29	2087.00	152618.96	1:36
Heuristik 3	109990.08	1435.00	140603.41	0:36

Tabelle 7.1.101

$\lambda = 22.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.73	3649.00	192856.40	2:11
Heuristik 2a	112917.67	88.00	114912.34	1:09
Heuristik 2b	108074.83	2131.00	156377.49	1:32
Heuristik 3	110024.78	1391.00	141554.11	0:36

Tabelle 7.1.102

$\lambda = 24.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	110145.74	3649.00	197721.74	2:11
Heuristik 2a	112779.80	187.00	117267.80	1:11
Heuristik 2b	108191.05	2055.00	157511.05	1:28
Heuristik 3	109979.34	1391.00	143363.34	0:40

Tabelle 7.1.103

Vergleich der Heuristiken in der 7. Iteration

Alle vorhergehenden Iterationen sind mit Heuristik 2a gerechnet. Der Parameter λ wurde in der ersten Iteration auf 0.02, in der zweiten Iteration auf 0.08, in der dritten auf 0.32, in der vierten auf 1.27, in der fünften auf 5.012 und in der sechsten auf 20.0 gesetzt (vergleiche Tabellen 7.1.4, 7.1.22, 7.1.43 , 7.1.62 , 7.1.81 und 7.1.100).

$\lambda = 5.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116345.16	0.00	116345.16	2:16
Heuristik 2a	112350.88	0.00	112350.88	1:05
Heuristik 2b	108758.54	0.00	108758.54	1:31
Heuristik 3	110473.19	0.00	110473.19	0:36

Tabelle 7.1.104

$\lambda = 10.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:11
Heuristik 2a	113218.33	0.00	113218.33	1:06
Heuristik 2b	108240.43	0.00	108240.43	1:32
Heuristik 3	110925.39	0.00	110925.39	0:36

Tabelle 7.1.105

$\lambda = 16.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.33	0.00	116366.33	2:19
Heuristik 2a	113210.82	0.00	113210.82	1:05
Heuristik 2b	108316.73	0.00	108316.73	1:30
Heuristik 3	110596.36	0.00	110596.36	0:35

Tabelle 7.1.106

$\lambda = 21.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:13
Heuristik 2a	113798.84	0.00	113798.84	1:08
Heuristik 2b	108011.88	0.00	108011.88	1:33
Heuristik 3	113045.93	0.00	113045.93	0:39

Tabelle 7.1.107

$\lambda = 26.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:14
Heuristik 2a	113831.36	0.00	113831.36	1:06
Heuristik 2b	107832.70	0.00	107832.70	1:34
Heuristik 3	113023.64	0.00	113023.64	0:46

Tabelle 7.1.108

$\lambda = 32.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.33	0.00	116366.33	2:21
Heuristik 2a	113785.13	0.00	113785.13	1:05
Heuristik 2b	108540.77	0.00	108540.77	1:33
Heuristik 3	112908.89	0.00	112908.89	0:45

Tabelle 7.1.109

$\lambda = 37.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.33	0.00	116366.33	2:23
Heuristik 2a	113140.71	0.00	113140.71	1:12
Heuristik 2b	108032.57	0.00	108032.57	1:31
Heuristik 3	112687.90	0.00	112687.90	0:51

Tabelle 7.1.110

$\lambda = 42.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:11
Heuristik 2a	113277.46	0.00	113277.46	1:03
Heuristik 2b	108708.12	0.00	108708.12	1:37
Heuristik 3	112585.12	0.00	112585.12	0:43

Tabelle 7.1.111

$\lambda = 48.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:10
Heuristik 2a	112723.22	0.00	112723.22	1:05
Heuristik 2b	108775.26	0.00	108775.26	1:31
Heuristik 3	112585.12	0.00	112585.12	0:37

Tabelle 7.1.112

$\lambda = 53.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:11
Heuristik 2a	113104.39	0.00	113104.39	1:03
Heuristik 2b	108878.72	0.00	108878.72	1:31
Heuristik 3	112642.90	0.00	112642.90	0:44

Tabelle 7.1.113

$\lambda = 58.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:11
Heuristik 2a	112810.52	0.00	112810.52	1:05
Heuristik 2b	108473.90	0.00	108473.90	1:31
Heuristik 3	112642.90	0.00	112642.90	0:44

Tabelle 7.1.114

$\lambda = 64.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:11
Heuristik 2a	113549.19	0.00	113549.19	1:07
Heuristik 2b	108469.69	0.00	108469.69	1:31
Heuristik 3	112642.90	0.00	112642.90	0:37

Tabelle 7.1.115

$\lambda = 69.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:10
Heuristik 2a	112902.17	0.00	112902.17	1:02
Heuristik 2b	108476.18	0.00	108476.18	1:33
Heuristik 3	112642.90	0.00	112642.90	0:37

Tabelle 7.1.116

$\lambda = 74.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:13
Heuristik 2a	113028.23	0.00	113028.23	1:07
Heuristik 2b	108428.62	0.00	108428.62	1:33
Heuristik 3	112642.90	0.00	112642.90	0:45

Tabelle 7.1.117

$\lambda = 80.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:12
Heuristik 2a	112976.07	0.00	112976.07	1:04
Heuristik 2b	108420.06	0.00	108420.06	1:37
Heuristik 3	112642.90	0.00	112642.90	0:38

Tabelle 7.1.118

$\lambda = 85.333$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:10
Heuristik 2a	112816.94	0.00	112816.94	1:03
Heuristik 2b	108472.35	0.00	108472.35	1:41
Heuristik 3	112642.90	0.00	112642.90	0:37

Tabelle 7.1.119

$\lambda = 90.667$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:13
Heuristik 2a	113069.49	0.00	113069.49	1:01
Heuristik 2b	108351.46	0.00	108351.46	1:41
Heuristik 3	112642.90	0.00	112642.90	0:44

Tabelle 7.1.120

$\lambda = 96.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	116366.34	0.00	116366.34	2:13
Heuristik 2a	113120.73	0.00	113120.73	1:05
Heuristik 2b	108409.51	0.00	108409.51	1:35
Heuristik 3	112642.90	0.00	112642.90	0:44

Tabelle 7.1.121

Testserie für die Schaltung soc_3

Vergleich der Heuristiken in der 1. Iteration

$\lambda = 0.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	296840.66	93712368.00	296840.66	9:21
Heuristik 2a	235153.84	79618080.00	235153.84	80:12
Heuristik 2b	420251.25	85382096.00	420251.25	5:38
Heuristik 3	337830.94	80635472.00	337830.94	1:34

Tabelle 7.1.122

$\lambda = 0.005$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	533241.12	31451832.00	690500.28	9:12
Heuristik 2a	349303.47	40296328.00	550785.10	79:42
Heuristik 2b	566964.12	38584256.00	759885.40	6:01
Heuristik 3	802481.44	23466528.00	919814.07	1:31

Tabelle 7.1.123

$\lambda = 0.010$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	544904.44	24439944.00	789303.87	9:14
Heuristik 2a	480439.00	28737876.00	767817.75	79:38
Heuristik 2b	638989.00	29266216.00	931651.15	6:07
Heuristik 3	846002.88	23978024.00	1085783.11	1:30

Tabelle 7.1.124

$\lambda = 0.020$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	637028.06	22946672.00	1095961.49	9:14
Heuristik 2a	410082.94	24408896.00	898260.85	79:30
Heuristik 2b	723850.44	22862348.00	1181097.39	6:17
Heuristik 3	860650.44	22978372.00	1320217.87	1:31

Tabelle 7.1.125

$\lambda = 0.030$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	643180.38	22433448.00	1316183.80	9:16
Heuristik 2a	584774.31	23009960.00	1275073.10	79:30
Heuristik 2b	763596.06	22323200.00	1433292.05	6:18
Heuristik 3	822046.38	22411148.00	1494380.80	1:32

Tabelle 7.1.126

$\lambda = 0.040$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	604759.62	22115540.00	1489381.21	9:19
Heuristik 2a	574705.25	22771616.00	1485569.87	79:38
Heuristik 2b	776083.75	22099032.00	1660045.01	6:21
Heuristik 3	851779.25	22194544.00	1739560.99	1:33

Tabelle 7.1.127

$\lambda = 0.050$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 2b	794348.56	22001072.00	1894402.18	6:23
Heuristik 2a	492098.59	22570116.00	1620604.41	80:08
Heuristik 3	981023.81	22266844.00	2094366.03	1:33
Heuristik 1	636004.00	22101448.00	1741076.42	9:19

Tabelle 7.1.128

$\lambda = 0.100$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	742663.75	21928544.00	2935518.18	9:22
Heuristik 2a	608071.00	22220992.00	2830170.23	79:18
Heuristik 2b	830908.38	21932464.00	3024154.81	6:31
Heuristik 3	884231.31	21992484.00	3083479.75	1:34

Tabelle 7.1.129

$\lambda = 0.200$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	827591.12	21867824.00	5201155.99	9:23
Heuristik 2a	660936.31	22039476.00	5068831.58	80:04
Heuristik 2b	838799.31	21902320.00	5219263.38	6:34
Heuristik 3	906908.31	21888076.00	5284523.58	1:37

Tabelle 7.1.130

$\lambda = 0.300$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	878596.50	21944152.00	7461842.36	6:44
Heuristik 2a	585320.25	21851964.00	7140909.71	79:58
Heuristik 2b	841195.94	21852124.00	7396833.40	6:41
Heuristik 3	967195.50	21794476.00	7505538.56	1:45

Tabelle 7.1.131

$\lambda = 0.400$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	895468.56	21853692.00	9636945.49	6:45
Heuristik 2a	609135.06	21847256.00	9348037.59	80:06
Heuristik 2b	850585.81	21858800.00	9594105.94	6:43
Heuristik 3	989960.75	21912020.00	9754768.88	1:45

Tabelle 7.1.132

$\lambda = 0.500$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	937779.94	21873796.00	11874677.94	6:45
Heuristik 2a	658983.12	21655856.00	11486911.12	79:44
Heuristik 2b	867980.31	21772304.00	11754132.31	6:43
Heuristik 3	985116.56	21770404.00	11870318.56	1:47

Tabelle 7.1.133

$\lambda = 1.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	1057703.62	21724828.00	22782531.62	6:48
Heuristik 2a	728516.69	21614124.00	22342640.69	78:49
Heuristik 2b	951000.88	21689072.00	22640072.88	6:44
Heuristik 3	1157405.50	21753696.00	22911101.50	1:40

Tabelle 7.1.134

$\lambda = 10.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	1843731.00	21513768.00	216981411.00	6:42
Heuristik 2a	985770.12	21624992.00	217235690.12	77:34
Heuristik 2b	1207225.50	21607264.00	217279865.50	6:46
Heuristik 3	1831848.50	21513412.00	216965968.50	1:37

Tabelle 7.1.135

$\lambda = 100.000$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	2561432.00	21502624.00	2152823832.00	6:40
Heuristik 2a	1413441.88	21644444.00	2165857841.88	75:15
Heuristik 2b	1277437.88	21630676.00	2164345037.88	6:52
Heuristik 3	2736664.50	21504024.00	2153139064.50	1:33

Tabelle 7.1.136

Vergleich der Heuristiken in der 2. Iteration

Als Lösung für die erste Iteration wählten wir das Ergebnis, welches von Heuristik 2a mit $\lambda = 0.02$ geliefert wurde (vergleiche Tabelle 7.1.125).

$\lambda = 0.010$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	404022.50	8176857.00	485791.07	7:47
Heuristik 2a	317405.75	9406357.00	411469.32	16:22
Heuristik 2b	342600.53	8255102.00	425151.55	2:53
Heuristik 3	473937.53	7434682.00	548284.35	1:19

Tabelle 7.1.137

$\lambda = 0.020$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	424588.31	6878349.00	562155.29	7:48
Heuristik 2a	342969.78	7405203.00	491073.84	16:33
Heuristik 2b	365265.69	6990278.00	505071.24	2:54
Heuristik 3	492070.31	6930039.00	630671.09	1:20

Tabelle 7.1.138

$\lambda = 0.030$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	438639.84	6370501.00	629754.87	7:48
Heuristik 2a	363519.44	6680998.00	563949.37	16:41
Heuristik 2b	376895.00	6414299.00	569323.97	2:55
Heuristik 3	488241.56	6699352.00	689222.12	1:22

Tabelle 7.1.139

$\lambda = 0.040$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	442071.91	6148770.00	688022.70	7:50
Heuristik 2a	380099.59	6165614.00	626724.15	16:44
Heuristik 2b	389593.69	6156201.00	635841.72	2:55
Heuristik 3	492757.97	6303675.00	744904.96	1:24

Tabelle 7.1.140

$\lambda = 0.050$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	462517.81	5910168.00	758026.19	7:51
Heuristik 2a	397136.91	5929581.00	693615.94	16:49
Heuristik 2b	398486.28	6040388.00	700505.66	2:56
Heuristik 3	494559.94	6226931.00	805906.47	1:23

Tabelle 7.1.141

$\lambda = 0.060$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	459185.62	5891741.00	812690.08	7:50
Heuristik 2a	402982.94	5883273.00	755979.31	16:51
Heuristik 2b	401877.41	5965213.00	759790.18	2:56
Heuristik 3	499716.66	6203823.00	871946.03	1:22

Tabelle 7.1.142

$\lambda = 0.070$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	465810.56	5807225.00	872316.31	7:51
Heuristik 2a	413717.28	5828065.00	821681.83	16:57
Heuristik 2b	407972.78	5928073.00	822937.89	2:57
Heuristik 3	516800.31	6043557.00	939849.30	1:24

Tabelle 7.1.143

$\lambda = 0.080$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	472570.22	5781402.00	935082.37	7:51
Heuristik 2a	416673.19	5785099.00	879481.10	17:01
Heuristik 2b	413514.59	5803258.00	877775.22	2:56
Heuristik 3	514775.09	6038189.00	997830.20	1:24

Tabelle 7.1.144

$\lambda = 0.090$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	479663.75	5702015.00	992845.08	7:51
Heuristik 2a	426144.47	5758593.00	944417.82	17:02
Heuristik 2b	419146.84	5775662.00	938956.40	2:57
Heuristik 3	516745.97	5877300.00	1045702.95	1:23

Tabelle 7.1.145

$\lambda = 0.100$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	483228.84	5654299.00	1048658.71	7:52
Heuristik 2a	434978.78	5698856.00	1004864.35	17:05
Heuristik 2b	417520.72	5836338.00	1001154.48	2:59
Heuristik 3	528549.25	5866009.00	1115150.12	1:24

Tabelle 7.1.146

$\lambda = 0.110$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	491067.00	5639789.00	1111443.79	7:53
Heuristik 2a	429752.31	5706217.00	1057436.18	17:00
Heuristik 2b	419699.75	5768085.00	1054189.10	2:57
Heuristik 3	521774.75	5910188.00	1171895.43	1:24

Tabelle 7.1.147

$\lambda = 0.120$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	502138.91	5641043.00	1179064.05	7:55
Heuristik 2a	427370.03	5705543.00	1112035.18	17:03
Heuristik 2b	421431.38	5787126.00	1115886.48	2:58
Heuristik 3	525667.06	5801097.00	1221798.69	1:25

Tabelle 7.1.148

$\lambda = 0.140$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	498642.78	5648768.00	1289470.30	7:55
Heuristik 2a	433500.31	5736568.00	1236619.84	17:04
Heuristik 2b	425975.69	5744299.00	1230177.55	3:00
Heuristik 3	518772.62	5721786.00	1319822.67	1:26

Tabelle 7.1.149

$\lambda = 0.160$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	503926.34	5567579.00	1394738.96	7:55
Heuristik 2a	433669.25	5716014.00	1348231.47	17:08
Heuristik 2b	434385.72	5743875.00	1353405.70	2:59
Heuristik 3	519587.97	5706389.00	1432610.19	1:25

Tabelle 7.1.150

Vergleich der Heuristiken in der 3. Iteration

Als Verfahren für die ersten zwei Iterationen wählten wir Heuristik 2a. Der Parameter λ wurde in der ersten Iteration auf 0.02 und in der zweiten Iteration auf 0.08 gesetzt (vergleiche Tabellen 7.1.125 und 7.1.144).

$\lambda = 0.027$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	379658.97	2279099.00	440434.94	9:30
Heuristik 2a	326259.00	2263445.00	386617.53	7:06
Heuristik 2b	339745.53	2001051.00	393106.89	3:19
Heuristik 3	404256.38	1685265.00	449196.77	2:23

Tabelle 7.1.151

$\lambda = 0.053$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	393476.50	1746457.00	486620.87	9:38
Heuristik 2a	349052.25	1787178.00	444368.41	7:06
Heuristik 2b	352280.12	1707037.00	443322.09	3:21
Heuristik 3	414147.09	1567166.00	497729.28	2:23

Tabelle 7.1.152

$\lambda = 0.080$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	407559.81	1545947.00	531235.57	8:48
Heuristik 2a	359546.16	1600553.00	487590.39	6:36
Heuristik 2b	362044.28	1628053.00	492288.52	3:05
Heuristik 3	410149.78	1502922.00	530383.54	1:35

Tabelle 7.1.153

$\lambda = 0.107$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	415480.19	1447427.00	569872.39	8:38
Heuristik 2a	368535.56	1525684.00	531275.18	6:35
Heuristik 2b	371772.44	1526943.00	534646.35	3:07
Heuristik 3	416119.78	1430707.00	568728.52	1:32

Tabelle 7.1.154

$\lambda = 0.133$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	420550.25	1399296.00	607123.04	8:38
Heuristik 2a	375800.09	1455209.00	569827.95	6:40
Heuristik 2b	379482.03	1455246.00	573514.82	3:05
Heuristik 3	426009.50	1422097.00	615622.42	1:32

Tabelle 7.1.155

$\lambda = 0.160$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	427195.31	1391459.00	649828.75	8:41
Heuristik 2a	383287.06	1422814.00	610937.30	6:33
Heuristik 2b	386597.56	1393236.00	609515.32	3:02
Heuristik 3	432989.00	1359287.00	650474.92	1:33

Tabelle 7.1.156

$\lambda = 0.187$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	433607.84	1338196.00	683404.43	8:55
Heuristik 2a	392943.72	1368801.00	648453.24	6:35
Heuristik 2b	388670.34	1369711.00	644349.73	3:19
Heuristik 3	442789.16	1360597.00	696767.26	1:35

Tabelle 7.1.157

$\lambda = 0.213$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	437475.00	1326916.00	720550.40	8:41
Heuristik 2a	395210.03	1366661.00	686764.36	6:32
Heuristik 2b	394571.53	1355901.00	683830.40	3:05
Heuristik 3	439088.12	1365352.00	730363.21	1:33

Tabelle 7.1.158

$\lambda = 0.240$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	442262.81	1318045.00	758593.61	8:42
Heuristik 2a	396654.22	1354498.00	721733.73	6:31
Heuristik 2b	397752.56	1347869.00	721241.12	3:04
Heuristik 3	444479.69	1346255.00	767580.88	1:36

Tabelle 7.1.159

$\lambda = 0.267$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	447153.59	1299910.00	793796.24	8:48
Heuristik 2a	404339.19	1322665.00	757049.83	6:31
Heuristik 2b	398630.75	1344730.00	757225.40	3:16
Heuristik 3	446877.97	1337159.00	803453.68	1:34

Tabelle 7.1.160

$\lambda = 0.293$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	450801.84	1304942.00	833584.82	8:51
Heuristik 2a	409373.41	1314319.00	794906.96	6:42
Heuristik 2b	402110.12	1337205.00	794356.91	3:07
Heuristik 3	453097.78	1334869.00	844659.34	1:45

Tabelle 7.1.161

$\lambda = 0.320$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	453969.09	1287023.00	865816.44	9:06
Heuristik 2a	414622.84	1306229.00	832616.11	7:01
Heuristik 2b	404695.59	1330207.00	830361.82	3:19
Heuristik 3	461052.88	1328386.00	886136.39	1:44

Tabelle 7.1.162

$\lambda = 0.347$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	458467.94	1292295.00	906463.53	8:54
Heuristik 2a	413421.31	1300835.00	864377.44	6:54
Heuristik 2b	407230.69	1317339.00	863908.20	3:04
Heuristik 3	467354.66	1342496.00	932753.27	1:56

Tabelle 7.1.163

$\lambda = 0.373$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	463984.81	1290114.00	945627.37	8:42
Heuristik 2a	419276.66	1284800.00	898935.32	6:59
Heuristik 2b	412108.44	1309125.00	900848.44	3:07
Heuristik 3	471632.56	1334962.00	970018.38	1:44

Tabelle 7.1.164

$\lambda = 0.400$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	466278.47	1283808.00	979801.64	8:45
Heuristik 2a	422510.19	1279473.00	934299.36	7:01
Heuristik 2b	410933.22	1310868.00	935280.39	3:05
Heuristik 3	472339.84	1309451.00	996120.21	1:44

Tabelle 7.1.165

$\lambda = 0.427$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	464674.50	1287840.00	1014152.87	8:56
Heuristik 2a	423983.97	1281038.00	970560.16	6:35
Heuristik 2b	411607.91	1307941.00	969662.71	3:16
Heuristik 3	477771.31	1306166.00	1035068.78	1:46

Tabelle 7.1.166

$\lambda = 0.453$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	469751.25	1287938.00	1053616.46	8:57
Heuristik 2a	427463.03	1278804.00	1007187.49	6:37
Heuristik 2b	414547.81	1301337.00	1004487.23	3:22
Heuristik 3	485153.12	1311181.00	1079555.16	1:49

Tabelle 7.1.167

$\lambda = 0.480$

	Verdr. länge	Überlappungen	Zielfunktion	CPU-Zeit
Heuristik 1	468960.81	1288514.00	1087447.52	8:53
Heuristik 2a	430409.22	1270828.00	1040406.65	6:47
Heuristik 2b	422450.94	1299675.00	1046294.92	3:35
Heuristik 3	482826.84	1306272.00	1109837.39	2:00

Tabelle 7.1.168

Anhang II

Grundlagen

Wir gehen davon aus, daß der Leser mit den grundlegenden Begriffen und Definitionen aus den Bereichen lineare Algebra, Graphen-, Polyeder- und Komplexitätstheorie vertraut ist. An dieser Stelle erklären wir kurz einige Begriffe aus der Graphen- und Polyedertheorie, auf die wir in der Arbeit häufig zurückgreifen werden.

Grundlagen der Graphentheorie

Ein **Graph** $G = (V, E)$ besteht aus einer **Knotenmenge** V und einer **Kantenmenge** E. Eine Kante $e \in E$ schreiben wir kurz als $e = uv$, wenn u und v **inzident** zu e, bzw. u und v die beiden **Endknoten** von e sind. Sind W eine Knotenmenge und F eine Kantenmenge in $G = (V, E)$, so bezeichnen wir mit $E(W)$ die Menge aller Kanten in E, derer beider Endknoten in W sind. Umgekehrt bezeichnet $V(F)$ die Menge der Knoten, die Endknoten mindestens einer Kante aus F sind. Eine Menge F von Kanten heißt **Schnitt**, wenn es eine Knotenmenge $W \subseteq V$ gibt, so daß $F = \delta(W) = \{uv \in E \mid u \in W, v \in V \setminus W\}$. Mit dem Symbol $[W_1 : W_2] = \{uv \in E \mid u \in W_1,\ v \in W_2\}$ kürzen wir die Menge der Kanten im Schnitt zwischen den Knotenmengen $W_1 \subseteq V$ und $W_2 \subseteq V$ ab. Ein Graph heißt **bipartit**, wenn es zwei Knotenmengen W_1, W_2 gibt, so daß $W_1 \cup W_2 = V$, $W_1 \neq \emptyset$, $W_2 \neq \emptyset$ und alle Kanten aus E in dem Schnitt zwischen W_1 und W_2 liegen. Ein Graph heißt **vollständig**, wenn je zwei Knoten durch eine Kante verbunden sind.
Sind $G = (V, E)$ und $H = (W, F)$ Graphen, so daß $W \subseteq V$ und $F \subseteq E$ gilt, dann heißt H **Untergraph** von G. Sei $W \subseteq V$, dann sagen wir, W **induziert** den Untergraphen $H = (W, E(W))$.
Eine **Clique** in einem Graphen G ist eine Knotenmenge W, so daß der durch W induzierte Untergraph vollständig ist. Eine **stabile Menge** ist eine Knotenmenge S, so daß je zwei Knoten aus S nicht durch eine Kante verbunden sind.
Eine Kantenmenge M in G heißt **Matching**, wenn je zwei Kanten in M keinen gemeinsamen Endknoten haben. Ein **Stern** in G ist eine Kantenmenge F, so daß alle Kanten aus F einen gemeinsamen Endknoten haben.
Ein **Kette** von einem Knoten v_0 zu einem Knoten v_n ist eine endliche Folge $(v_0, e_0, v_1, e_1, \ldots, e_{n-1}, v_n)$, die mit v_0 beginnt und mit v_n endet und in der Knoten und Kanten alternierend auftreten, so daß jede Kante e_i inzident zu den Knoten v_i und v_{i+1} ist. Der Knoten v_0 heißt **Anfangsknoten**, v_n **Endknoten** der Kette. Die Knoten $v_1, \ldots, v_{n-1}$ heißen **innere Knoten**. Die Zahl n ist die **Länge** der Kette. Eine Kette nennen wir **geschlossen**, wenn sie positive Länge hat und Anfangs und Endknoten übereinstimmen. Ein **Weg** ist eine Kette, in der alle Knoten verschieden sind. Eine geschlossene Kette, in der

Anfangsknoten und alle inneren Knoten verschieden sind, heißt **Kreis**. Wir nennen einen Graphen **zusammenhängend**, wenn es zu je zwei verschiedenen Knoten einen Weg in dem Graphen gibt. Eine Kantenmenge F nennen wir **Baum**, wenn der Graph $(V(F), F)$ zusammenhängend ist und keinen Kreis enthält. Ein **aufspannender Baum** in einem Graphen $G = (V, E)$ ist ein Baum $F \subseteq E$ mit $V(F) = V$. Ein Knoten v heißt **Blatt** in einem Baum, wenn er Endknoten genau einer Kante in dem Baum ist.
Für eine ausführlichere Darstellung der Begriffe aus der Graphentheorie verweisen wir auf das Buch [BM76].

Grundlagen der Polyedertheorie

Im folgenden bezeichne $\mathbf{K}$ stets einen Körper.
Sei $A \in \mathbf{K}^{m \times n}$ eine Matrix und $b \in \mathbf{K}^m$ ein Vektor. Ein **Polyeder** P ist eine Teilmenge des $\mathbf{K}^n$, die sich darstellen läßt als : $P = \{x \in \mathbf{K}^n \mid Ax \leq b\}$.
Falls für alle $x \in P$ $|x| \leq c$ für eine reele Zahl c gilt, so nennen wir P ein **Polytop**. Die **Dimension** einer Teilmenge des $\mathbf{K}^n$ ist die Anzahl affin unabhängiger Vektoren dieser Teilmenge minus Eins. Die Dimension eines Polyeders kürzen wir mit $dim(P)$ ab. Ein Poyeder $P \subseteq \mathbf{K}^n$ heißt **volldimensional**, wenn $dim(P) = n$ ist.
Eine lineare Ungleichung $a^T x \leq \alpha$ ist **gültig** für ein Polyeder P, wenn für alle $x' \in P$ $a^T x' \leq \alpha$ erfüllt ist. Eine Teilmenge $F \subseteq \mathbf{K}^n$ heißt **Seitenfläche**, wenn es eine gültige Ungleichung $a^T x \leq \alpha$ gibt, so daß $F = P \cap \{x \in \mathbf{K}^n \mid a^T x \leq \alpha\}$. Eine **Facette** eines Polyeders P ist eine nicht-leere Seitenfläche $F \neq P$ der Dimension $dim(P) - 1$.
Gegeben sei ein ganzzahliges lineares Programm, welches sich schreiben läßt gemäß:

$$\{x \in \mathbf{K}^n \mid Ax \leq b,\ x \text{ ganzzahlig}\}.$$

Unter der **LP-Relaxierung** dieses Programm verstehen wir das Polyeder

$$\{x \in \mathbf{K}^n \mid Ax \leq b\}.$$

Für eine detailliertere Beschreibung der in der Polyedertheorie verwendeten Konzepte verweisen wir auf [S89].

Als Einführung in die Komplexitätstheorie führen wir [GJ79] an.

Literaturverzeichnis

[B70] : M. Balinsky, "On a selection problem", Management Science 17, 1970, pp 230–231.

[B75] : E. Balas, "Facets of the knapsack polytope", Mathematical Programming 8, 1975, pp 146–164.

[B77] : M.A. Breuer, "A class of min-cut placement algorithms", Proc. 14th. Design Automation Conference New Orleans, 1977, pp 284–290.

[B85] : J.P. Blank, "Near-optimal placement using a quadratic function", Proc. 22nd. DAC, 1985, pp 609–615.

[B86] : F. Barahona, "A solvable case of quadratic 0-1 programming", Discrete Applied Mathematics 13, 1986, pp 23–26.

[BCH89a] : E. Boros, Y. Crama, P.L. Hammer, "Chvatal cuts and odd cycle inequalities in quadratic 0-1 optimization", Rutcor Research Report 48, Rutgers University, New Jersey, 1989.

[BCH89b] : E. Boros, Y. Crama, P.L. Hammer, "Upper bounds for quadratic 0-1 maximization problems", Rutcor Research Report 14-89, Rutgers University, New Jersey, 1989.

[BGM85] : F. Barahona, M.Grötschel, A.R. Mahjoub, "Facets of the bipartite subgraph polytope", Mathematics of Operations Research, 1985, pp 340–385.

[BH89] : E. Boros, P.L. Hammer, "A max flow approach to improved roof duality in quadratic 0-1 minimization", Rutcor Research Report 15-89, Rutgers University, New Jersey, 1989.

[BH90] : E. Boros, P.L. Hammer, "Cut-polytopes, boolean quadric polytopes and nonnegative quadratic pseudo-boolean functions", Rutcor Research Report 24-90, Rutgers University, New Jersey, 1990.

[BHS89] : E. Boros, P.L. Hammer,X. Sun "The DDT method for quadratic 0-1 minimization", 1989, Manuskript.

[BJR87] : F. Barahona, M. Jünger, G. Reinelt, "Experiments in quadratic 0-1 programming", Mathematical Programming 44, 1989, pp 127–138.

[BM76] : J.A. Bondy, S.R. Murty, Graph theory with applications, American Elsevier, London, 1976.

[BM84] : E. Balas, J.B. Mazzola, "Non linear 0-1 programming: I. Linearization techniques, II. Dominance relations and algorithms", Mathematical Programming 30, 1984, pp 1–45.

[BM86] : F. Barahona, A,R. Mahjoub, "On the cut polytope", Mathematical programming 36, 1986, pp 157–173.

[C84] : M. W. Carter, "The indefinite zero-one quadratic problem", Discrete Applied Mathematics, 1984, pp 23–44.

[CHJ88] : Y. Crama, P. Hansen, B. Jaumard, "The basic algorithm for pseudo-boolean programming revisited", Technical Report 35, GERAD and Department of Applied Mathematics, Montreal, 1988.

[CJP83] : H. Crowder, E. L. Johnson, M. W. Padberg, "Solving large-scale zero-one linear programming problems", Operations Research Vol 31, No. 5, 1983.

[CR89] : S. Chopra, M.R. Rao, "Facets of the k-partitoning polytope", Manuskript.

[CR91] : S. Chopra, M.R. Rao, "On the multiway cut polyhedron", Networks 21, 1991, pp 51–89.

[DL88] : M. Deza, L. Laurent, "Facets for the complete cut cone 1,"1988, erscheint in Mathematical programming.

[DGL91] : M. Deza, M. Grötschel, L. Laurent, , "Complete descriptions of small multicut polytopes", in P. Gritzmann, B. Sturmfels (Hrsg), "Applied Geometry and Discrete Mathematics": the Victor Klee Festschrift, DIMACS Series in discrete mathematics and theoretical computer science, AMS ACM 4, 1991.

[DGL92] : M. Deza, M. Grötschel, L. Laurent, "Clique-web-facets for multicut polytopes", erscheint in Mathematical programming 1992.

[DL89] : M. Deza, L. Laurent, "Facets for the complete cut cone 2: Clique web inequalities", 1989, erscheint in Mathematical programming.

[FM82] : C.M. Fiduccia, R.M. Mattheyses, "A linear-time heuristic for improving network partitions, Proc. 19th. DAC, 1982, pp 175–181.

[GH61] : R.E. Gomory, T.C. Hu, "Multi-terminal network flows", Journal of the Society for Industrial and Applied Mathematics 9, 1961, pp 551–570.

[GJ77] : M.R. Garey, D.S. Johnson, "The rectilinear Steiner tree problem is $\mathcal{NP}$-complete", SIAM Journal on Applied Mathematics 32, 1977, pp 826–834.

[GJ79] : M.R. Garey, D.S. Johnson, Computers and Intractability. A guide to the theory of $\mathcal{NP}$-completeness, Freeman, New York, 1979.

[GKP88] : G.Georgakopoulos, D. Kavvadias, C.H. Papadimitriou, "Probabilistic satisfiability", Journal of Complexity 4, 1988, pp 1–11.

[GR90] : E. S. Gottlieb, M. R. Rao, "The generalized assignment problem: Valid inequalities and facets", Mathematical Programming 46, 1990, pp 31 – 52 .

[GR90a] : E S. Gottlieb, M. R. Rao, "(1,k)-configuration facets for the generalized assignment problem", Mathematical Programming 46, 1990, pp 53 – 60.

[H65] : P.L. Hammer, "Some network flow problems solved with pseudo-boolean programming", Operations Research 13, 1965, pp 388–399.

[H79] : P. Hansen, "Methods of nonlinear 0-1 programming", Annals of Discrete Mathematics 5, 1979, pp 53–70.

[H92] : W. Hermann, "Clustering Verfahren zur Plazierung von Zellen im Sea-of-Cells Entwurfsverfahren, Diplomarbeit, Universität Augsburg, 1992.

[HHS84] : P.L. Hammer, P. Hansen, B. Simeone, "Roof duality, complementation and persistency in quadratic 0-1 optimization", Mathematical Programming 28, 1984, pp 121–155.

[HJP75] : P.L. Hammer, E.L. Johnson, U.N. Peled, "Facets of regular 0-1 polytopes", Mathematical Programming 8, 1975, pp 179–206.

[HK70] : M. Held, R.M. Karp, "The traveling-salesman problem and minimum spanning trees", Operations Research 18, 1970, pp 1138–1162.

[HK71] : M. Held, R.M. Karp, "The traveling-salesman problem and minimum spanning trees: part II", Mathematical Programming 1, 1971, pp 6–25.

[HR68] : P.L. Hammer, S. Rudeanu, Boolean methods in Operations Research and related areas, Springer, New York, 1968.

[HS86] : P. Hansen, B. Simeone, "Unimodular functions", Discrete Applied Mathematics 14, 1986, pp 269–281.

[JJA83] : F.M. Johannes, K.M. Just, K.J. Antreich, "On the force placement of logic arrays", Proc. 6-th., European Conference on Circuit Theory and Design, 1983, pp. 203–206.

[JK85] : K.M. Just, J.M Kleinhans, "On the simultaneous placement of modules of integrated circuits", AEÜ, Vol. 39, No. 4, 1985, pp. 217–224.

[K72] : R.M. Karp, Reducibility among combinatorial problems Plenum Press, New York, 1972.

[K83] : S. Kirkpatrick, C. Gelatt, M. Vecchi, "Optimization by Simulated Annealing", Science 220, 1983, pp 671–680.

[KL70] : B.W. Kernighan, S. Lin, "An efficient heurisitic procedure for partitioning graphs", Bell Systems Technical Journal, 1970, pp 291–307.

[KL84] : M. R. Kramer, J. van Leeuwen, "The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits", Advances in Computing Research 2, 1984, pp 129–146.

[KPS90] : B. Korte, H.J. Prömel, A. Steger, "Steiner trees in VLSI-layout", Paths, Flows, and VLSI-Layout, Springer-Verlag, Berlin Heidelberg, 1990, pp 185–214.

[KSFA91] : J.M Kleinhans, G. Sigl, F.M. Johannes, K.J. Antreich, "Gordian: VLSI placement by quadratic programming and slicing optimization", IEEE Transactions on computer aided design 10, No. 3, 1991.

[KSJ88] : J.M Kleinhans, G. Sigl, F.M. Johannes, "Gordian: A new global optimization/ rectangle dissection method for cell placement", IEEE Int. Conference on CAD ICCAD-88, 1988, pp. 506–509.

[L79] : U. Lauther, "A min-cut placement algorithm for general cell assemblies based on a graph representation", ACM/ICEE Proc. 16th DAC, 1979, pp 1–10.

[L90] : T. Lengauer, Combinatorial algorithms for integrated circuit layout, Wiley, Chichester, 1990.

[M53] : N. Metropolis et al., "Equations of state calculations by fast computer machines", Journal of Chemical Physics 21, 1953, pp1087–1092.

[M90] : R. Müller, "Hierarchisches Floorplanning mit integrierter globaler Verdrahtung", Dissertation, Universität-Gesamthochschule Paderborn, 1990.

[MT81] : S. Martello, P. Toth, "An algorithm for the generalized assignment problem", Operations Research 1981, Vol. 81,pp 589–603.

[O84] : R. Otten, L. van Ginnecken, "Floorplan design using Simulated Annealing", Proc. IEEE Internation Conference on Computer Aided Design, 1984, pp 96–98.

[P67] : B.T. Polyak, "A general method of solving extremum problems" (auf Russisch), Doklady Akademii Nauk SSSR 174 (1), 1967, pp 33–36 [in englischer Übersetzung: Soviet Mathematics Doklady 8, 1967, pp 593–597.

[P75] : M. W. Padberg, "A Note on 0-1 Programming", Operations Research 23, 1975, pp 833 – 837.

[P80] : M. W. Padberg, "(1,k) configuration and facets for packing problems", Mathematical Programming 18, 1980, pp 94–99.

[P89] : M. Padberg, "The boolean quadric polytope: Some characteristics, facets and relatives", Mathematical programming 45, 1989, pp 132–172.

[PR75] : J. Picard, D. Ratliff, "Minimum cuts and related problems", Networks 5, 1975, pp 357–370.

[PR88] : P.M. Pardalos, G. Rodgers, "Computational aspects of a branch and bound algorithm for quadratic zero one programming", Technical report CS-88-04, Computer Science Department, Pennsylvania State University, 1988.

[R70] : J. Rhys, "A selection problem of shared fixed costs and networks, Management science 7, 1970, pp 200–207.

[S89] : A. Schrijver, Theory of linear and integer programming, Wiley, Chichester, 1989.

[SG74] : S. Sahni, T. Gonzalez, "Computationally related problems", SIAM Journal on Computing 3, 1974, pp 262–279.

[SK72] : D.G. Schweikert, B.W. Kernighan, "A proper model for the partitioning of electrical circuits", Proc. Design Automation Conference, 1972, pp 56–62.

[W75] : L. A. Wolsey, "Faces of linear inequalities in 0-1 variables", Mathematical Programming 8, 1975, pp 165–178.

[W85] : A.C. Williams, "Quadratic 0-1 programming using the roof dual with computational results", RUTCOR Research Report 8-85, Rutgers University, New Brunswick, N.J., 1985.

[W86] : Y. Wakabayashi, "Aggregation of binary relations-algorithmic and polyhedral investigations", Dissertation, Universität Augsburg, 1986.

[Z78] : E. Zemel, "Lifting the facets of zero-one polytopes", Mathematical Programming 15, 1978, pp 268–277.

Index

Allgemeine Zellen **5**
Affinitätsgraph **113**
Affinitätskoeffizienten **17**
Affinitätskosten **74**
Anfangsknoten **213**
Aufspannender Baum **214**
Basiszellen **5**
Baum **214**
Baumweite **29**
bipartit **213**
Blatt **214**
Chip **2**
Clique **213**
Cliquengraph **23**
Cluster **114**
Clusterkapazität **114**
Cluster-Lifting **136**
Clusteringproblem **114**
Dimension **214**
Endknoten **213**
Erweiterung einer Schaltung **85**
Erweiterungsfaktor **85**
ε-approximativer Algorithmus **34**
Facette **214**
Floorplaning **20**
Gate-Array **3**
Graph **213**
Gültige Ungleichung **214**
Gütegarantie **34**
Heap **71**
induzieren **213**
Innerer Knoten **213**
inzident **213**
k-Baum **29**
Kanten-Lifting **136**
Kantenmenge **213**
Kardinalitäts-Clusteringproblem **114**
Kette **213**
Knoten-Lifting **136**
Knotenmenge **213**
Kompaktierung **6**
Kreis **214**
$(1, d)$-Konfiguration **142**
Lagrange-Funktion **102**
Lagrange-Multiplikatoren **102**
Lagrange-Relaxierung **102**
Lagrange-Wert **102**
Länge einer Kette **213**
Lifting **136**
logische Zelle **2**
LP-Relaxierung **214**
Master **2**
Matching **213**
Mehrfachschnitt **113**
Mehrfachschnitt-Problem **113**
Minimale Überdeckung **142**
Netz **2**
Partieller k-Baum **29**
Partition **113**
Pin **2**
Plazierung **7**
Polyeder **214**
Polytop **214**
Quadratische Posiform **28**
r-Clusteringpoytop **129**
r-Clusteringproblem **114**
r-Kardinalitäts-Clusteringproblem **114**
r-Schnitt **113**
r-Schnitt-Problem **113**
Randzelle **2**
Realisierung **2**
Repräsentative Basiszelle **46**
Rhys-Programm **28**
Roof dual-Wert **28**
Schnitt **213**
Schrittlänge **102**

Sea of cells **4**
Seitenfläche **214**
Shortest Manhaaten-Distanz **17**
Standardzellen **5**
Stabile Menge **213**
Stern **213**
Subgradient **102**
Symmetrisches Umschließen **49**
Transistor **2**
Überdeckung **142**
Untergraph **213**
Verdrahtung **6**
volldimendional **214**
Vollständiger Graph **213**
Vollständige Menge **28**
Vollständiges Graphenmodell **8**
Via **6**
Wandern von Zellen **48**
Weg **213**
Worst case-Schranke **34**
Zelle **2**
Zusammenhängender Graph **214**

Lebenslauf

29.06.1965	geboren in München als Sohn von Gundolf Weismantel und Margot Weismantel, geb. Deublein
1971–1972	Besuch der Volksschule in Gemünden a. Main, Bayern
1972–1975	Besuch der Volksschule in Miltenberg, Bayern
1975–1984	Besuch des Johannes-Butzbach-Gymnasiums in Miltenberg, Bayern
29.06.1984	Abitur
1984–1988	Student der Wirtschaftsmathematik an der Universität Augsburg, Bayern
22.12.1988	Erlangung des Titels Dipl. Math. oec. (UNIV) an der Universität Augsburg, Bayern
seit 1.1.1989	wissenschaftlicher Mitarbeiter am Lehrstuhl für Angewandte Mathematik II der Universität Augsburg
seit 1.9.1991	wissenschaftlicher Mitarbeiter am Konrad-Zuse-Zentrum für Informationstechnik Berlin