Kombinatorische Optimierung und VLSI-Entwurf

Gerhard Reinelt

Institut für Angewandte Mathematik, Universität Heidelberg,
Im Neuenheimer Feld 294, D-69120 Heidelberg
Gerhard.Reinelt@IWR.Uni-Heidelberg.DE

1 Optimierungsprobleme beim VLSI-Entwurf

Elektronische Halbleiterchips sind in fast allen Bereichen unseres täglichen Lebens zu finden und übernehmen immer komplexere Aufgaben. Hochintegrierte Schaltungen mit mehreren hundertausend Transistoren können nur mit Computerunterstützung und dem Einsatz fortgeschrittener mathematischer Techniken entwickelt und produziert werden; ihre kostengünstige Herstellung ist eine Herausforderung an Techniker und Ingenieure.

Die Entwicklung eines Halbleiterchips gliedert sich in mehrere Stufen. Es erfolgt zunächst der **logische Entwurf**, in dem festgelegt wird, welche Funktionalität die Schaltung haben soll. Es wird spezifiziert, aus welchen Einzelkomponenten (Transistoren, logische Einheiten, Speicher, etc.) sie zusammengesetzt wird und wie diese Komponenten durch Netze verbunden werden. Ein Netz verbindet jeweils die Anschlüsse der Komponenten, die das gleiche Signal führen. Der nächste Schritt besteht aus dem eigentlichen **Layout-Design**. Die Lage der Einzelkomponenten auf dem Chip wird festgelegt (**Plazierung**) und die Verbindungsnetze werden konkret ausgelegt (**Verdrahtung**). Plazierung und Verdrahtung erfolgen wegen ihrer enormen Komplexität typischerweise in mehreren Phasen. Zunächst erfolgt eine vereinfachte globale Plazierung mit entsprechender globaler Verdrahtung, dann werden Plazierung und Verdrhatung lokal optimiert. Eine **Kompaktierung**, mit der eine noch weitere Verkleinerung der benötigten Chipfläche angestrebt wird, schließt das Layout-Design ab. Mit Hilfe von Schaltkreissimulationen erfolgt das **Testen** der Schaltung auf Funktionsfähigkeit. Der letzte Schritt besteht dann in der eigentlichen **Produktion** des Chips.

In allen diesen Phasen des Entwurfs und der Produktion treten mathematische Probleme hoher und sehr hoher Komplexität auf und es finden sich sehr viele unterschiedliche kombinatorische Optimierungsaufgaben, deren Lösung zur Chip-Herstellung benötigt wird. Auch ein nur grober Überblick würde den Rahmen dieses Artikels sprengen, wir verweisen auf die umfassende Diskussion aller Aspekte des VLSI-Entwurfs in [Le1].

In diesem Artikel soll exemplarisch ein spezielles Teilproblem des VLSI-Entwurfs, das Problem der **Via-Minimierung**, untersucht werden. Anhand dieses Problems wird die Vorgehensweise der mathematischen Modellierung erläutert, mehrere algorithmische Verfahren zur Lösung des entsprechenden mathematischen Optimierungsproblems werden behandelt. Die Leistungsfähigkeit der einzelnen Verfahren wird an Problembeispielen aus der Praxis untersucht.

Ebenfalls aus Platzgründen kann keine umfassende Einführung in die Konzepte der Graphentheorie und der linearen Optimierung erfolgen. Wir verweisen hierzu auf [NW1].

2 Modellierung des Via-Minimierungsproblems

Bei einer speziellen Entwurfstechnologie für Leiterplatten mit zwei Lagen erfolgt nach der Plazierung der Komponenten zunächst nur eine sogenannte **transiente Verdrahtung**. In ihr wird bereits die Lage der einzelnen Verbindungen bestimmt, es erfolgt aber noch keine Zuordnung der Verbindungssegmente zu den Lagen. Figur 1 zeigt eine transiente Verdrahtung für sechs Komponenten und elf Verbindungen, die jeweils zwei Anschlüsse verbinden.

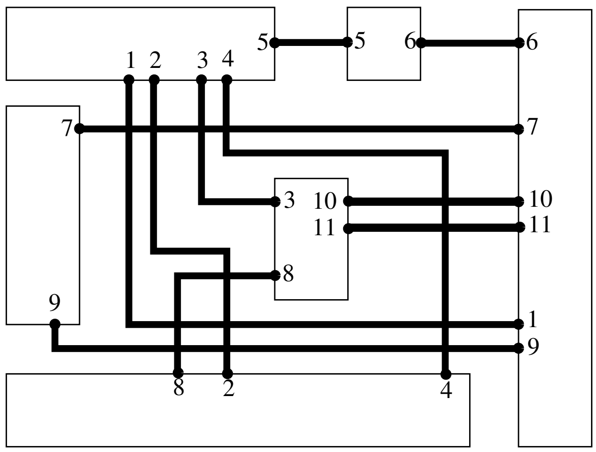

Fig. 1. Eine transiente Verdrahtung

Im nächsten Schritt muß nun die Lagenzuordnung für die einzelnen Verbindungssegmente bestimmt werden. Um Kurzschlüsse zu vermeiden, müssen Verbindungen Lagen wechseln können. Lagenwechsel erfolgen über Durchkontaktierungen (**Vias**). Vias sind verglichen mit den Verbindungen relativ groß und behindern daher spätere Kompaktierungsverfahren. Weiterhin verkomplizieren sie den Herstellungsprozeß; eine sehr große Anzahl von Vias kann die Ausbeute verringern. Ziel der Lagenzuordnung ist daher die **Via-Minimierung**, die kurzschlußfreie Realisierung der konkreten Verdrahtung mit möglichst wenig Durchkontaktierungen. Die folgende Darstellung basiert auf [CKC1] und [Pi1], die zuerst die polynomiale Lösbarkeit des Problems für zwei Lagen zeigten, und [BGJR1] zur Behandlung einiger Zusatzbedingungen.

2.1 Ein vereinfachtes Modell

Wir betrachten zunächst ein vereinfachtes Modell der realen Situation. Diese Vereinfachung erlaubt nur 2-Punkt-Verbindungen, d.h. jedes Netz verbindet genau zwei Anschlüsse.

In einem ersten Analyseschritt werden die Einzelverbindungen in **freie** und **kritische** Segmente unterteilt, je nachdem, ob auf ihnen eine Durchkontaktierung (d.h. die Plazierung eines Vias) erlaubt ist oder nicht. Kritische Segmente entstehen etwa, wenn sich zwei Verbindungssegmente in der transienten Verdrahtung kreuzen oder wenn ihr Abstand so klein ist, daß sie auf unterschiedlichen Lagen plaziert werden müssen. Eventuell müssen weitere Designregeln berücksichtigt werden. Da Vias relativ groß sind, muß ein gewisser Sicherheitsabstand eingehalten werden. Figur 2 zeigt die Klassifizierung von kritischen und freien Segmenten für das Beispiel. Die kritischen Segmente sind von 1 bis 17 durchnumeriert.

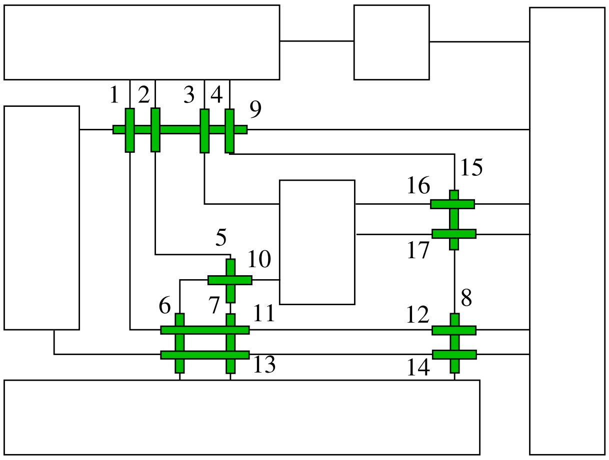

Fig. 2. Freie und kritische Segmente

Wir repräsentieren die Topologie der freien und kritischen Segemente im sogenannten **Layoutgraphen** $G = (V, E)$ mit **Knotenmenge** V und **Kantenmenge** E. Dieser Graph enthält einen Knoten für jedes kritische Segment und zwei Arten von Kanten. Zwei Knoten werden durch eine **Konfliktkante** verbunden, wenn die zugehörigen kritischen Segmente auf unterschiedlichen Lagen plaziert werden müssen. Zwei Knoten werden durch eine **Fortsetzungskante** verbunden, wenn sie durch ein freies Segment verbunden sind. Den Layoutgraphen für

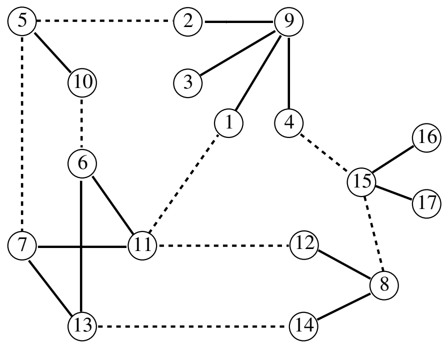

Fig. 3. Layoutgraph mit Konflikt- und Fortsetzungskanten

das Beispiel zeigt Figur 3 (Konfliktkanten sind durchgezogen, Fortsetzungskanten sind gestrichelt).

Einer Zuweisung der kritischen Segmente auf die beiden Lagen entspricht im Layoutgraphen offenbar eine Aufteilung der Knotenmenge V in zwei Teilmengen W_1 und W_2. Der durch W_1 und W_2 induzierte **Schnitt** F besteht aus den Kanten, die zwischen diesen beiden Teilmengen verlaufen und wird mit $(W_1 : W_2)$ bezeichnet. F muß alle Konfliktkanten enthalten, denn sonst wäre die Lagenzuweisung unzulässig (zwei kritische Segmente wären auf der gleichen Lage, obwohl sie in Konflikt stehen). Jede Fortsetzungskante in F erfordert die Plazierung eines Vias auf dem entsprechenden Verbindungssegment, da sich die beiden verbundenen kritischen Segmente auf unterschiedlichen Lagen befinden. Das Problem, eine Lagenzuweisung mit einer minimalen Anzahl von Vias zu bestimmen, korrespondiert somit dazu, im Layoutgraphen einen Schnitt F zu bestimmen, der alle Konfliktkanten und möglichst wenige Fortsetzungskanten enthält.

Eine genauere Betrachtung der Struktur von Layoutgraphen erlaubt es, das Problem zu vereinfachen. Man betrachte den Teilgraphen des Layoutgraphen, der nur aus den Konfliktkanten besteht. Dieser Graph zerfällt in Komponenten $G_i = (V_i, E_i)$, die untereinander nicht verbunden sind. Im Beispiel ergeben sich fünf Komponenten. Jede Komponente muß **bipartit** sein, d.h. ihre Knotenmenge muß sich so in zwei Teilmengen zerlegen lassen, daß alle Konfliktkanten zwischen diesen Teilmengen verlaufen. Andernfalls wäre überhaupt keine Lagenzuweisung möglich, und die transiente Verdrahtung müßte korrigiert werden. Eine weitere Beobachtung zeigt, daß die Zuweisung eines Segments einer Komponente zu einer der beiden Lagen die Lagenzuordnung für alle anderen Segmente der Komponente impliziert. Wird etwa im Beispiel das Segment 7 auf Lage 1 plaziert, dann müssen die Segmente 11 und 13 auf Lage 2 und das Segment 6 auf Lage 1 plaziert werden.

Wir nutzen diese Eigenschaften zur Vereinfachung des Problems und konstruieren einen sogenannten **reduzierten Layoutgraphen** $R = (W, F)$ wie folgt. Seien $G_i = (V_i, E_i)$, $i = 1, \ldots, k$, die Komponenten des Layoutgraphen. Aus jeder Komponente $G_i = (V_i, E_i)$ wird ein Knoten v_i als Repräsentant ausgewählt;

seine Lagenzuordnung impliziert die Lagenzuordnung für alle anderen Knoten der Komponente. Die Knotenmenge des reduzierten Layoutgraphen besteht aus diesen Repräsentanten, d.h. $W = \{v_1, v_2, \ldots, v_k\}$. Zwei Knoten werden durch eine Kante verbunden, wenn die zugehörigen Komponenten durch mindestens eine Fortsetzungskante verbunden sind, d.h. $F = \{v_i v_j \mid \text{es gibt } u \in V_i, v \in V_j \text{ mit } uv \in E\}$. Jeder Kante $v_i v_j$ werden zwei Zahlen α_{ij} und β_{ij} zugeordnet, die wie folgt definiert sind:

α_{ij} = Anzahl der Fortsetzungskanten zwischen V_i und V_j, die Knoten auf unterschiedlichen Lagen verbinden, wenn v_i und v_j derselben Lage zugewiesen werden,

β_{ij} = Anzahl der Fortsetzungskanten zwischen V_i und V_j, die Knoten auf unterschiedlichen Lagen verbinden, wenn v_i und v_j unterschiedlichen Lagen zugewiesen werden.

Sei nun $(W_1 : W_2)$ ein Schnitt im reduzierten Layoutgraphen $R = (W, F)$. Dann läßt sich mit Hilfe dieser Zahlen die zugehörige Anzahl $\nu(W_1, W_2)$ der erforderlichen Vias angeben als

$$\nu(W_1, W_2) = \sum_{\substack{v_i v_j \in F \\ v_i, v_j \in W_1}} \alpha_{ij} + \sum_{\substack{v_i v_j \in F \\ v_i, v_j \in W_2}} \alpha_{ij} + \sum_{\substack{v_i v_j \in F \\ v_i \in W_1, v_j \in W_2}} \beta_{ij} \ .$$

Mit der Konstante $C = \sum_{v_i v_i \in F} \alpha_{ij}$ ist dann

$$\nu(W_1, W_2) = C + \sum_{\substack{v_i v_j \in F \\ v_i \in W_1, v_j \in W_2}} (\beta_{ij} - \alpha_{ij}) \ .$$

Mit Kantengewichten $c_{ij} = \alpha_{ij} - \beta_{ij}$, für alle $ij \in F$, besteht dann das Via-Minimierungsproblem darin, im reduzierten Layoutgraphen R einen Schnitt $B = (W_1 : W_2)$ zu bestimmen, so daß die Summe der Kantengewichte des Schnitts $\sum_{v_i v_j \in B} c_{ij}$ maximal ist. Man bezeichnet dieses Optimierungsproblem als **Max-Cut-Problem**. Die zu einem maximalen Schnitt B gehörige Minimalzahl benötigter Vias ergibt sich dann als $C - \sum_{v_i v_j \in B} c_{ij}$.

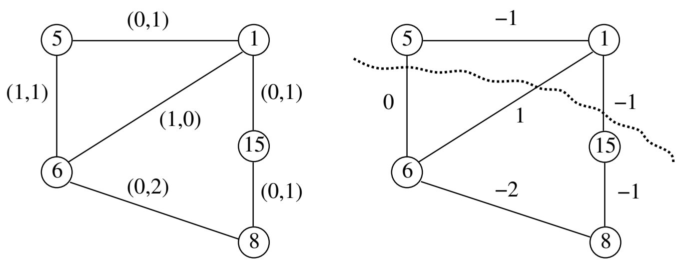

Fig. 4. Reduzierter Layoutgraph und maximaler Schnitt

Im Beispiel ergibt sich mit den Repräsentanten 1, 5, 6, 8 und 15 der linke Graph von Figur 4. Die Kanten sind jeweils mit den Zahlen $(\alpha_{ij}, \beta_{ij})$ beschriftet.

Die Transformation zum Max-Cut-Problem liefert schließlich den rechten Graphen von Figur 4 mit den angegebenen Kantengewichten. Ein maximaler Schnitt in diesem Graphen ist z. B. der Schnitt $(\{1,5\} : \{6,8,15\})$ mit Gewicht 0.

Mit $C = 2$ ergibt sich im Beispiel also eine Lagenzuordnung, die zwei Vias erfordert. Ein Via liegt auf dem freien Segment, das die kritische Segmente 5 und 7 verbindet, das andere Via auf dem Verbindungssegment zwischen 4 und 15. Eine entsprechende Lagenzuordnung ist in Figur 5 abgebildet.

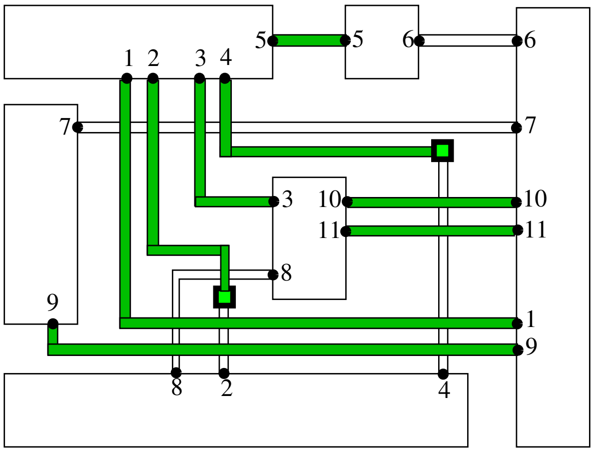

Fig. 5. Eine optimale Lagenzuweisung

2.2 Erweiterung des Modells

Die bisherige Darstellung entwickelte ein prinzipielles Vorgehen zur Modellierung des Via-Minimierungsproblems als Max-Cut-Problem. Es gibt aber einige weitere technische Anforderungen, die berücksichtigt werden müssen.

3-Weg-Verzweigungen. Die Beschränkung auf 2-Punkt-Verbindungen ist natürlich unzureichend zur Behandlung praktischer Layouts, da Netze typischerweise mehrere Anschlüsse verbinden. Verbindet ein Netz mehr als zwei Punkte, so muß es sich verzweigen. Verwendet man nur horizontale und vertikale Verbindungssegmente, so können 3-Weg- und 4-Weg-Verzweigungen auftreten.

3-Weg-Verzweigungen werden wie folgt modelliert. Führt jede der Verzweigungen einer 3-Weg-Verzweigung zu einem kritischen Segment (andernfalls muß die Verzeigung nicht gesondert behandelt werden), so werden die entsprechenden Knoten im Layoutgraphen durch Verbindungskanten mit Gewicht 1/2 verbunden (die „normalen“ Verbindungskanten, die nicht an Verzweigungen beteiligt sind, erhalten das Gewicht 1). Die Definition der Zahlen α_{ij} und β_{ij} wird wie folgt erweitert:

α_{ij} = Summe der Gewichte der Fortsetzungskanten zwischen V_i und V_j, die Knoten auf unterschiedlichen Lagen verbinden, wenn v_i und v_j derselben Lage zugewiesen werden,

β_{ij} = Summe der Gewichte der Fortsetzungskanten zwischen V_i und V_j, die Knoten auf unterschiedlichen Lagen verbinden, wenn v_i und v_j unterschiedlichen Lagen zugewiesen werden.

Man überzeugt sich leicht, daß 3-Weg-Verzweigungen korrekt behandelt werden. Der Beitrag zum Gewicht eines Schnitts ist genau dann 1, wenn nicht alle drei beteiligten Segmente auf der gleichen Lage liegen: in diesem Fall ist ein Via erforderlich, welches im Verzweigunspunkt plaziert werden kann. Die Modellierung ist im linken Teil von Figur 6 dargestellt.

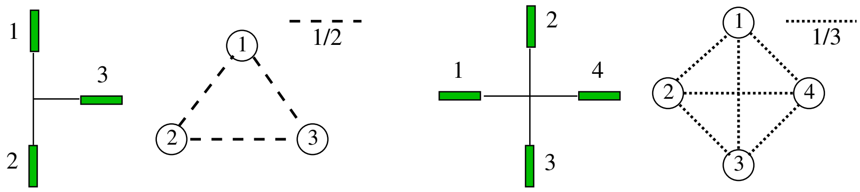

Fig. 6. Modellierung von 3- und 4-Weg-Verzweigungen

4-Weg-Verzweigungen. In Layouts können auch 4-Weg-Verzweigungen auftreten. Diese bereiten Modellierungsprobleme, wenn alle vier von der Verzweigung ausgehenden Verbindungen zu kritischen Segmenten führen. Eine näherungsweise Modellierung ist die folgende. Man verbindet je zwei der Knoten, die die vier kritischen Segmente repräsentieren, mit Verbindungskanten des Gewichts 1/3. Bis auf den Fall, daß zwei Segmente auf der einen und zwei auf der anderen Lage plaziert werden, erfolgt ein korrekter Beitrag zum Wert eines Schnitts. In diesem speziellen Fall ist der Beitrag 4/3 anstelle des richtigen Werts 1 (Plazierung eines Vias im Verzweigungspunkt). Die Modellierung ist im rechten Teil von Figur 6 veranschaulicht.

Bei Vorliegen von 3-Weg- oder 4-Weg-Verzweigungen können im reduzierten Layoutgraphen **Schleifen** entstehen, d.h. Kanten, deren beide Endknoten gleich sind. Diese Schleifen können bei der Bestimmung eines maximalen Schnitts ignoriert werden, zur Berechnung der korrekten Anzahl der Vias müssen sie allerdings mit ihren Beiträgen zu den Zahlen α_{ij} und β_{ij} berücksichtigt werden.

Fixierung von Anschlüssen. Eine weitere Komplikation tritt auf, wenn für einige Anschlüsse vorgegeben ist, auf welcher Lage sie anzusteuern sind (**Preassignment**).

Um dies zu modellieren, wird zunächst für jeden fixierten Anschluß ein kritisches Segment (ohne geometrische Ausdehnung) generiert. Der Layoutgraph wird wie oben beschrieben mit diesen zusätzlichen Segmenten konstruiert. Es wird weiterhin ein neuer Knoten 0 zum reduzierten Layoutgraphen hinzugefügt, der mit allen Segmenten, die fixierten Anschlüssen entsprechen, verbunden wird. Der Knoten 0 repräsentiert eine der beiden Lagen. Ist ein Anschluß auf dieser Lage fixiert, wird der entsprechende Knoten über eine Verbindungskante mit Gewicht $-M$ mit Knoten 0 verbunden. Ist der Anschluß auf der anderen Lage fixiert, erhält die Kante zum Knoten 0 das Gewicht $+M$. Hierbei ist M eine genügend große Zahl, die erzwingt, daß alle Kanten mit Gewicht $+M$ und keine der Kanten mit Gewicht $-M$ im maximalen Schnitt enthalten sind.

Man überlegt sich leicht eine bessere Modellierung von Anschlußfixierungen, die ohne Gewichte $+M$ oder $-M$ auskommt. Man erzeugt wieder ein kritisches Segment für jeden fixierten Anschluß und bereits im Layoutgraphen einen Knoten 0, der eine der beiden Lagen repräsentiert. Anschlüsse, die auf der anderen Lage fixiert sind, werden über eine Konfliktkante mit 0 verbunden. Knoten, die Anschlüssen entsprechen, die auf der gleichen Lage liegen, werden mit dem Knoten 0 identifiziert, d.h. an ihnen anliegende Kanten werden jetzt direkt mit dem Knoten 0 verbunden. Die Komponente des Layoutgraphen, die den Knoten 0 enthält, repräsentiert jetzt alle Anschlußvorbelegungen und ebenso der zugehörige Knoten im reduzierten Layoutgraphen.

Lagenbevorzugung. Es ist in der Praxis durchaus möglich, daß die beiden Lagen nicht gleichwertig sind, sondern daß eine der beiden Lagen günstigere elektrische Eigenschaften hat und deswegen bevorzugt verwendet werden sollte. Durch entsprechend viele Durchkontaktierungen kann man natürlich fast die gesamte Verdrahtung auf einer Lage realisieren. Es ist aber abzuwägen zwischen Minimierung von Vias und Maximierung der Gesamtlänge der Verbindungen, die der bevorzugten Lage zugewiesen werden.

Lagenbevorzugung läßt sich in das hier vorgestellte Modell einbringen. Man fügt auf allen oder zumindest den genügend langen freien Segmenten ein künstliches (dimensionsloses) kritisches Segment ein. Der reduzierte Layoutgraph wird durch einen Knoten 0 erweitert, der die bevorzugte Lage repräsentiert. Jedes neue kritische Segment wird mit diesem Knoten über eine Kante mit Gewicht $-h \cdot l$ verbunden, wobei l der Länge des entsprechenden freien Segments entspricht und h ein Parameter ist, der die Wichtigkeit der Lagenbevorzugung modelliert. Ist h genügend groß, dann werden alle berücksichtigten freien Segmente auf die bevorzugte Lage plaziert, da dann der Zielfunktionsanteil für die Via-Minimierung vernachlässigt wird. Im Fall $h = 0$ erfolgt eine reine Via-Minimierung. Man beachte, daß auch bei gleichzeitiger Berücksichtigung von Anschlußfixierungen und Lagenbevorzugung nur ein zusätzlicher Knoten erforderlich ist.

Mehrlagenverdrahtung. In einigen Produktionstechnologien ist es auch möglich, die Verbindungsnetze auf mehr als zwei Lagen auszulegen. Die Modellierung der 2-Lagen-Verdrahtung läßt sich auf diese Fälle nicht erweitern. Probleme der Mehrlagenverdrahtung werden in [Mo1] diskutiert.

2.3 Komplexitätstheoretische Aspekte

Das Max-Cut-Problem ist im allgemeinen Fall $\mathcal{NP}$-schwierig und gehört damit in einem strengen mathematischen Sinn zu den schwierigsten kombinatorischen Optimierungsproblemen. Nach komplexitätstheoretischen Maßstäben ist es genauso schwierig wie das bekannte **Traveling Salesman Problem**. Nach dem gegenwärtigen Stand der Forschung ist daher nicht zu erwarten, daß ein Lösungsalgorithmus für das Max-Cut-Problem entwickelt werden kann, der allgemeine Probleme in beliebiger Größenordnung effizient (d.h. in polynomialer Zeit) löst. Einige Spezialfälle sind hier von Interesse.

Falls $c_{ij} \leq 0$ für alle Kanten ij des reduzierten Layoutgraphen gilt, dann ist das Problem leicht lösbar. Man formuliert dieses spezielle Problem normalerweise in der Minimierungsversion (nach Multiplikation aller Kantengewichte mit -1 ist anstatt des maximalen ein minimaler Schnitt zu bestimmen). Dies ist das klassische **Min-Cut-Problem**.

Ein Graph heißt **planar**, wenn er sich so in der Ebene zeichnen läßt, daß sich keine Kanten überkreuzen. Das Max-Cut-Problem für planare Graphen ist polynomial mit Matchingtechniken lösbar ([Ha1] und [OD1]). Falls keine Anschlußfixierungen oder Lagenbevorzugung vorliegen, dann ist der reduzierte Layoutgraph planar, denn seine Kanten korrespondieren zu freien Segmenten der transienten Verdrahtung und diese kreuzen sich nach Voraussetzung nicht. Unter Umständen ist der reduzierte Layoutgraph auch noch bei Anschlußfixierungen planar, z.B. dann, wenn alle fixierten Anschlüsse am Rand des Layouts liegen.

Fast-planare Graphen sind Graphen, die zwar nicht planar sind, die aber planar werden, wenn ein geigneter Knoten mit allen anliegenden Kanten entfernt wird. Dies ist im vorliegenden Fall immer dann gegeben, wenn ein Zusatzknoten 0 verwendet wird und der reduzierte Layoutgraph nicht planar ist. Elimination dieses Knotens liefert einen planaren Graphen nach den obigen Bemerkungen. Das Max-Cut-Problem für diese Graphenklasse ist aber bereits $\mathcal{NP}$-schwierig, d.h. im komplexitätstheoretischen Sinn genauso schwierig wie das allgemeine Max-Cut-Problem ([Ba1]).

2.4 Problembeispiele

Wir werden in den folgenden Abschnitten konkrete Rechnungen für sechs Beispiele aus der Praxis durchführen. Es handelt sich hier um Layouts, die von der Firma SIEMENS zur Verfügung gestellt wurden.

Wir betrachten jedes Problem in zwei Varianten. In der einen Variante sind alle Anschlüsse auf einer der beiden Lagen fixiert, in der anderen Variante wurde angenommen, daß Anschlüsse wahlweise auf einer der beiden Lagen erfolgen können. Die Bezeichnungen der Probleme sind dann VIA1y bis VIA6y für die Probleme, bei denen die Anschlüsse fixiert wurden, und VIA1n bis VIA6n für die anderen Probleme. Die Daten der Probleme bzw. der zugehörigen reduzierten Layoutgraphen sind in Tabelle 1 angegeben. Die Layoutgraphen zu den Problemvarianten mit fixierten Anschlüssen haben jeweils einen Knoten mehr als die entsprechenden Varianten ohne Fixierungen. Die Anzahl der Kanten zeigt, daß es sich um Graphen mit relativ wenigen Kanten handelt. In den Beispielen ist die Kantenzahl etwa doppelt so hoch wie die Knotenzahl. Industrielösungen lagen nur für die Problemvarianten mit Anschlußfixierungen vor. Lagenzuordnungen erfolgten im wesentlichen nach der einfachen Vorschrift, daß alle vertikalen Segmente auf die eine Lage und alle horizontalen Segmente auf die andere Lage plaziert wurden (Anschlußfixierungen wurden über zusätzliche Vias berücksichtigt.) Im Beispiel aus Figur 1 würde diese Regel acht Vias generieren.

Tabelle 1. Beispielprobleme

Problem	Anschlüsse	Netze	Reduz. Layoutgraph Knoten	Kanten	Anzahl Vias im Original-Layout
VIA1n	427	198	827	1445	–
VIA1y	427	198	828	1779	421
VIA2n	443	205	979	1775	–
VIA2y	443	205	980	2102	434
VIA3n	676	300	1326	2480	–
VIA3y	676	300	1327	2844	683
VIA4n	748	352	1365	2606	–
VIA4y	748	352	1366	2915	650
VIA5n	770	360	1201	2234	–
VIA5y	770	360	1202	2557	782
VIA6n	555	213	946	1703	–
VIA6y	555	213	947	1976	608

Die Konstruktion der reduzierten Layoutgraphen aus den Daten der transienten Verdrahtung ist recht aufwendig und eine nichttriviale Implementierungsaufgabe. Die Erzeugung der Graphen dauert mehrere Sekunden.

3 Heuristiken für das Max-Cut-Problem

Da es sich beim Max-Cut-Problem um ein $\mathcal{NP}$-schwieriges Optimierungsproblem handelt, kann man nicht erwarten, alle in der Praxis auftretenden Beispiele in vernünftiger Zeit optimal lösen zu können. Man benötigt daher auch Verfahren, die in kurzer Zeit zumindest relativ gute Layouts liefern. In diesem Abschnitt sollen zunächst derartige approximative Algorithmen (Heuristiken) diskutiert werden. Die Heuristiken sowie das exakte Verfahren aus Kapitel 4 nutzen nicht eine eventuelle Planarität der Graphen aus; sie sind allgemein einsetzbar. Für die Beschreibung spezieller effizienter (allerdings recht komplexer) Algorithmen zur Lösung planarer Max-Cut-Probleme sei auf [Mu1] verwiesen.

3.1 Konstruktionsverfahren

Von Konstruktionsverfahren spricht man bei Heuristiken, die sukzessive aufgrund eines lokalen Kriteriums einen Schnitt konstruieren. So bestimmte Lösungen können dann Ausgangspunkt für weitere Heuristiken sein.

Im folgenden sei für einen gegebenen Graphen $G = (V, E)$ mit Kantengewichten c_{uv}, für alle $uv \in E$, ein maximaler Schnitt zu bestimmen. Wir vereinbaren $c_{uv} = 0$, falls $uv \notin E$. Die Heuristiken bestimmen jeweils einen Schnitt $(S : T)$.

Greedy-Verfahren. Dieses Verfahren teilt die Knoten nacheinander den beiden Seiten des Schnitts zu. Das lokale Kriterium ist, daß jeweils der Knoten zugeteilt wird, der aktuell den größten Zielfunktionszuwachs liefert.

Greedy-Heuristik

1. Wähle einen beliebigen Knoten $v \in V$ und setze $S = \{v\}$, $T = \emptyset$ und $W = V \setminus \{v\}$.
2. Solange W nicht leer ist:
   (a) Berechne für alle $w \in W$ die Werte $d(w) = \max\{c_S(w), c_T(w)\}$, wobei $c_S(w) = \sum_{u \in S} c_{uw}$ und $c_T(w) = \sum_{u \in T} c_{uw}$.
   (b) Wähle $w^* \in W$ mit maximalem Wert $d(w^*)$.
   (c) Setze $W = W \setminus w^*$ und $S = S \cup \{w^*\}$, falls $d(w^*) = c_S(w^*)$, bzw. $T = T \cup \{w^*\}$ sonst.

Die Laufzeit des Verfahrens ist $O(E \log E)$, falls z.B. eine Heap-Datenstruktur zur Berechnung der Maxima verwendet wird. Eine einfache Implementierung liefert die Laufzeit $O(V^2)$.

Baum-Heuristik. Die folgende Heuristik ist speziell für Graphen mit wenigen Kanten geeignet, wie sie hier beim Via-Minimierungsproblem auftreten. Es wird zunächst ein maximaler aufspannender Baum in G bezüglich neuer Kantengewichte d, definiert durch $d_{uv} = |c_{uv}|$, für alle $uv \in E$, bestimmt. Ausgehend von diesem Baum läßt sich leicht ein Schnitt konstruieren, der alle Kanten des Baums enthält, die positive Originalgewichte haben, und keine der Kanten des Baums, die negative Originalgewichte haben.

Baum-Heuristik

1. Definiere neue Kantengewichte d_{uv} für alle $uv \in E$ durch $d_{uv} = |c_{uv}|$.
2. Berechne einen maximalen aufspannenden Baum in G bezüglich dieser neuen Kantengewichte.
3. Zeichne einen Knoten r als Wurzel des Baums aus und setze $S = \{r\}$ und $T = \emptyset$.
4. Durchsuche den Baum ausgehend von r. Sei hierbei uv eine Kante, die während des Durchsuchens durchlaufen wird, wobei u Vater von v im Baum ist (d.h. u ist insbesondere schon S oder T zugeordnet worden).
   (a) Falls $c_{uv} > 0$, dann setze $S = S \cup \{v\}$, falls $u \in T$, bzw. $T = T \cup \{v\}$, falls $u \in S$.
   (b) Falls $c_{uv} \leq 0$, dann setze $S = S \cup \{v\}$, falls $u \in S$, bzw. $T = T \cup \{v\}$, falls $u \in T$.

Enthält der Graph nur relativ wenige Kanten, so wird ein Schnitt konstruiert, der die wichtigsten Kanten des Graphen richtig berücksichtigt. Die Laufzeit ist im wesentlichen bestimmt durch die Zeit zur Berechnung des maximalen aufspannenden Baumes. Mit einer effizienten Implementierung des Algorithmus von Kruskal ergibt sich auch hier eine Laufzeit $O(E \log E)$, einfachere Implementierungen liefern die Laufzeit $O(V^2)$.

Zufälliger Schnitt. Ein Schnitt kann natürlich auch zufällig generiert werden. Diese „Konstruktionsheuristik“ liefert normalerweise sehr schlechte Ergebnisse, hat aber eine gewisse Berechtigung, da zufällig generierte Schnitte oft durchaus sinnvolle Startlösungen für gewisse Verbesserungsverfahren darstellen. Für unsere Beispiele hat diese Heuristik keine Bedeutung; wir verwenden sie nur, um Leistungsunterschiede von Verbesserungsverfahren zu dokumentieren.

Rechenergebnisse. Die drei Heuristiken wurden auf die 12 Beispielprobleme angewendet. Tabelle 2 gibt die erzielten Ergebnisse sowie die Laufzeiten entsprechender Implementierungen auf einer SPARCstation 10/20 in Sekunden an. Die Baum-Heuristik ist der Greedy-Heuristik deutlich überlegen, die Zufallslösung liefert erwartungsgemäß sehr hohe Via-Zahlen.

Tabelle 2. Ergebnisse der Konstruktionsheuristiken

Problem	Zufallsschnitt		Greedy-Heuristik		Baum-Heuristik	
	Vias	CPU	Vias	CPU	Vias	CPU
VIA1n	1742	0.00	592	0.24	282	0.17
VIA1y	1937	0.00	466	0.24	318	0.17
VIA2n	1770	0.00	677	0.36	365	0.22
VIA2y	1999	0.00	559	0.38	405	0.26
VIA3n	2459	0.01	900	0.81	534	0.41
VIA3y	2789	0.01	887	0.74	594	0.43
VIA4n	2948	0.01	839	0.81	496	0.44
VIA4y	3334	0.01	722	0.80	525	0.45
VIA5n	2652	0.01	1001	0.58	633	0.33
VIA5y	2998	0.01	998	0.59	677	0.35
VIA6n	1999	0.01	812	0.39	532	0.24
VIA6y	2292	0.01	664	0.33	589	0.22

3.2 Verbesserungsverfahren

Nach der Konstruktion eines Schnitts kann man versuchen, diesen durch Modifikationen zu verbessern. Auch hier bieten sich verschiedene Möglichkeiten an, von denen einige diskutiert werden sollen.

Im folgenden berechnen wir für jeden Knoten u in Abhängigkeit vom aktuellen Schnitt zwei Zahlen E_u und I_u. Sei $(S : T)$ der aktuelle Schnitt. Für $u \in S$ definieren wir $E_u = \sum_{v \in T} c_{uv}$ und $I_u = \sum_{v \in S} c_{uv}$; für $u \in T$ ist entsprechend $E_u = \sum_{v \in S} c_{uv}$ und $I_u = \sum_{v \in T} c_{uv}$. Für jeden Knoten u setzen wir $D_u = E_u - I_u$.

1-Opt und 2-Opt. Ein sehr einfaches Verbesserungsverfahren testet, ob es günstiger ist, einzelne Knoten der anderen Seite des Schnitts zuzuordnen. Die **1-Opt-Heuristik** ist wie folgt formuliert.

1-Opt-Heuristik

1. Sei $(S : T)$ der aktuelle Schnitt.
2. Für jeden Knoten $v \in V$ führe aus: Falls $D_v < 0$, dann setze im Fall $v \in S$ $S = S \setminus \{v\}$ und $T = T \cup \{v\}$ und im Fall $v \in T$ setze $T = T \setminus \{v\}$ und $S = S \cup \{v\}$.
3. Falls für keinen Knoten in 2 eine Verbesserung des aktuellen Schnitts möglich war, dann beende das Verfahren, andernfalls wiederhole Schritt 2.

Die Laufzeit der Heuristik hängt von der Anzahl der Verbesserungschritte und damit von der Qualität der Startlösung ab. Hier kann i.a. keine polynomiale Laufzeit garantiert werden. Im Fall der reinen Via-Minimierung kann Schritt 2 aber höchstens $|E|$-mal ausgeführt werden. Der Aufwand für Schritt 2 läßt sich durch $O(V^2)$ abschätzen.

Die **2-Opt-Heuristik** testet, ob es sinnvoll ist, zwei Knoten gleichzeitig der jeweils anderen Seite des Schnitts zuzuordnen. Ein solcher Test ist nur eine Erweiterung des 1-Opt-Austauschs, wenn diese beiden Knoten durch eine Kante verbunden sind, andernfalls kann man die Knoten einzeln auf einen möglichen 1-Opt-Austausch untersuchen.

2-Opt-Heuristik

1. Sei $(S : T)$ der aktuelle Schnitt.
2. Für jede Kante $uv \in E$ führe aus:
   (a) Falls $uv \in (S : T)$ (mit o.B.d.A. $u \in S$ und $v \in T$) und $D_u + D_v - 2c_{uv} < 0$ dann setze $S = S \cup \{v\}$, $T = T \setminus \{v\}$, $T = T \cup \{u\}$ und $S = S \setminus \{u\}$.
   (b) Falls $u \in S$, $v \in S$ und $D_u + D_v + 2c_{uv} < 0$ dann setze $S = S \setminus \{u, v\}$ und $T = T \cup \{u, v\}$.
   (c) Falls $u \in T$, $v \in T$ und $D_u + D_v + 2c_{uv} < 0$ dann setze $T = T \setminus \{u, v\}$ und $S = S \cup \{u, v\}$.
3. Falls für keine Kante in 2 eine Verbesserung des aktuellen Schnitts möglich war, dann beende das Verfahren, andernfalls wiederhole Schritt 2.

Für die Laufzeit gelten im Prinzip die gleichen Bemerkungen wie für 1-Opt. Der Aufwand für Schritt 2 läßt sich durch $O(E)$ abschätzen.

Wegen der Einfachheit dieser lokalen Modifikationen kann man nicht allzu gute Resultate erwarten. Rechenexperimente zeigen, daß es nicht sinnvoll ist, mit

Zufalls- oder Greedy-Lösungen zu starten, da die erzielten Resultate viel schlechter als die Ergebnisse der Baum-Heuristik sind. Tabelle 3 zeigt daher nur die Ergebnisse der 1-Opt-Heuristik, der 2-Opt-Heuristik, und der 1-Opt-Heuristik mit anschließender 2-Opt-Heuristik, wenn jeweils mit der Lösung der Baum-Heuristik gestartet wurde. Die 1-Opt-Heuristik ist besser als die 2-Opt-Heuristik, die besten Lösungen lieferte ein kombiniertes Verfahren. Der Rechenzeitaufwand ist gering.

Tabelle 3. Ergebnisse der 1-Opt und 2-Opt-Verbesserungen

Problem	1-Opt		2-Opt		1-Opt/2-Opt	
	Vias	CPU	Vias	CPU	Vias	CPU
VIA1n	272	0.01	282	0.01	272	0.01
VIA1y	312	0.01	308	0.08	302	0.08
VIA2n	347	0.00	362	0.02	344	0.03
VIA2y	383	0.00	400	0.06	382	0.06
VIA3n	513	0.01	530	0.04	513	0.04
VIA3y	573	0.01	584	0.14	569	0.15
VIA4n	490	0.01	494	0.03	488	0.04
VIA4y	516	0.01	524	0.09	516	0.07
VIA5n	619	0.01	627	0.05	613	0.04
VIA5y	657	0.01	667	0.12	657	0.08
VIA6n	517	0.00	520	0.03	511	0.03
VIA6y	571	0.01	582	0.08	568	0.08

Kernighan-Lin-Heuristik. Bei einfachen Austauschverfahren stellt man fest, daß sehr schnell relativ schlechte lokale Maxima erreicht werden, die dann wegen der begrenzten Modifikationsmöglichkeiten nicht mehr verlassen werden können. Dies tritt inbesondere dann auf, wenn mit weniger guten Ausgangslösungen gestartet wird. Die folgende Heuristik versucht, komplexere Austauschoperationen ohne zu hohen Rechenaufwand zu generieren. Eine besondere Eigenschaft ist, daß auch Modifikationen betrachtet werden, die zunächst zu einer Verschlechterung der aktuellen Lösung führen, aber die Chance bieten, neue Verbesserungsmöglichkeiten zu eröffnen.

Die Heuristik von Kernighan und Lin wurde ursprünglich dafür entwickelt, die Knotenmenge eines Graphen so in zwei gleichgroße Teilmengen zu zerlegen, daß der entstehende Schnitt maximal ist. Dieses Problem ist bei einem speziellen Plazierungsverfahren (Min-Cut-Verfahren) zu lösen; es ist $\mathcal{NP}$-schwierig. Die Heuristik läßt sich aber auch für allgemeine Schnittprobleme anwenden, indem man isolierte Knoten zum Graphen hinzufügt. Zur vereinfachten Darstellung nehmen wir daher im folgenden an, daß $G = (V, E)$ mit $|V| = 2n$ vorliegt.

Das Verfahren bestimmt eine Folge von n tentativen Austauschen, die jeweils durch das Paar (s'_p, t'_p) beschrieben werden. Es wird dann die beste Teilfolge dieser Austausche bestimmt und realisiert, falls sie eine Verbesserung des Schnitts liefert, andernfalls terminiert das Verfahren.

Kernighan-Lin-Heuristik

1. Bestimme einen Schnitt $(S : T)$ in G mit $|S| = |T| = n$.
2. Setze $p = 1$, $S_p = S$ und $T_p = T$.
3. Bestimme $s_i \in S_p$, $t_j \in T_p$ mit maximalem Wert $g_p = D_{s_i} + D_{t_j} - 2c_{s_i t_j}$.
4. Setze $s'_p = s_i$, $t'_p = t_j$, $S_{p+1} = S_p \setminus \{s_i\}$ und $T_{p+1} = T_p \setminus \{t_j\}$.
5. Falls $p < n$, setze $p = p + 1$, $D_u = D_u + 2c_{us_i} - 2c_u t_j$, für alle $u \in S_p$, $D_v = D_v + 2c_{vt_j} - 2c_{vs_i}$, für alle $v \in T_p$, und gehe zu Schritt 3.
6. Bestimme $1 \le k \le n$, so daß $G = \sum_{i=1}^{k} g_i$ maximal ist.
7. Falls $G > 0$, setze $S = S \setminus \{s'_1, \ldots, s'_k\} \cup \{t'_1, \ldots, t'_k\}$, $T = T \setminus \{t'_1, \ldots, t'_k\} \cup \{s'_1, \ldots, s'_k\}$, und gehe zu Schritt 1.

Um die hohe Rechenzeit zu reduzieren, bestimmt man in Schritt 2 nicht das beste Austauschpaar, sondern berücksichtigt aus S_p und T_p jeweils nur die l Knoten mit den höchsten D-Werten. Weiterhin kann man die Länge der Austauschfolgen durch einen Parameter q beschränken. Für feste l und q beträgt die Laufzeit zur Bestimmung eines Austauschs $O(V^2)$ im Gegensatz zur Zeit $O(V^4)$ in der oben dargestellten Version. Rechenergebnisse sind in Tabelle 4 dargestellt. Es wurden hier die Parameter $l = 20$ und $q = 20$ gesetzt.

Tabelle 4. Ergebnisse der Kernighan-Lin-Heuristik

Problem	Zufallsschnitt		Greedy-Heuristik		Baum-Heuristik	
	Vias	CPU	Vias	CPU	Vias	CPU
VIA1n	514	6.19	467	2.45	272	0.81
VIA1y	373	8.72	316	2.03	302	0.81
VIA2n	529	8.79	549	5.33	339	0.96
VIA2y	757	9.05	400	4.92	373	1.53
VIA3n	946	11.28	763	3.31	511	1.31
VIA3y	1023	15.31	611	9.83	565	1.31
VIA4n	878	20.54	567	9.81	488	1.44
VIA4y	1062	19.76	540	5.55	504	2.14
VIA5n	991	13.55	818	5.52	609	1.82
VIA5y	920	20.41	780	4.75	647	1.79
VIA6n	717	7.18	654	3.92	508	0.97
VIA6y	809	9.01	595	2.81	566	0.94

Man sieht, daß die Rechenzeit für zufällige Startlösungen recht hoch ist, aber dennoch keine guten Ergebnisse geliefert werden. Durch Erhöhung von l und q kann man hier Verbesserungen erzielen, die aber zu noch höheren Rechenzeiten führen. Beim Start mit der Baum-Heuristik werden sehr gute Lösungen in kurzer Zeit berechnet.

Lokale Enumeration. Eine weitere lokale Modifikation ist besonders für Graphen mit wenigen Kanten und kleinen Knotengraden geeignet, wie sie bei der Via-Minimierung auftreten.

Lokale Enumeration

1. Sei $(S:T)$ der aktuelle Schnitt.
2. Für jeden Knoten $u \in V$ führe aus: Bestimme für u und alle Nachbarn von u durch Enumeration die beste Zuordnung dieser Knoten zu den beiden Seiten des Schnitts und aktualisiere S und T entsprechend.
3. Falls für keinen Knoten in 2 eine Verbesserung des aktuellen Schnitts möglich war, dann beende das Verfahren, andernfalls wiederhole Schritt 2.

Die Enumeration sollte nur für Knoten mit kleinen Grad ausgeführt werden, da der Aufwand exponentiell mit der Anzahl der Nachbarn wächst. In den vorliegenden Beispielen wurde die Enumeration für alle Knoten außer dem zusätzlichen Knoten 0 (falls vorhanden) ausgeführt.

Die Ergebnisse dieser Heuristik für drei verschiedene Startlösungen sind in Tabelle 5 aufgelistet. Auch hier werden die besten Lagenzuordnungen erzielt, wenn mit der Baum-Heuristik gestartet wird.

Tabelle 5. Ergebnisse der lokalen Enumeration

Problem	Zufallsschnitt		Greedy-Heuristik		Baum-Heuristik	
	Vias	CPU	Vias	CPU	Vias	CPU
VIA1n	386	0.68	445	0.38	272	0.27
VIA1y	360	1.11	305	0.70	302	0.52
VIA2n	461	0.79	510	0.58	343	0.42
VIA2y	444	1.42	405	0.89	373	0.66
VIA3n	731	1.13	665	1.00	513	0.62
VIA3y	800	1.85	593	1.18	567	1.04
VIA4n	678	1.30	604	0.73	488	0.63
VIA4y	702	2.13	571	0.93	504	0.91
VIA5n	740	0.94	747	0.64	605	0.57
VIA5y	826	1.73	766	1.33	647	0.88
VIA6n	640	0.78	613	0.51	508	0.44
VIA6y	625	1.22	575	1.05	566	0.67

Die Ergebnisse der Enumerationsheuristik sind wesentlich besser als die der Kernighan-Lin-Heuristik, falls mit einer zufälligen Lösung oder einer Greedy-Lösung gestartet wird. Wird zunächst die Baum-Heuristik ausgeführt, sind die Ergebnisse beider Verfahren von gleicher Qualität.

Es gibt noch eine Reihe weiterer Heuristiken. In erster Linie ist hier **Simulated Annealing** zu nennen, das oft für Max-Cut-Probleme angewendet wird, die beim Studium von Spingläsern in der theoretischen Physik vorkommen ([LA1]).

4 Exakte Lösung von Max-Cut-Problemen

Zur exakten Lösung schwieriger kombinatorischer Optimierungsprobleme werden üblicherweise Branch & Bound-Algorithmen eingesetzt. Ein solches Verfahren besteht aus Heuristiken zur Bestimmung unterer Schranken (zulässiger Lösungen), Methoden zur Berechnung oberer Schranken für den Wert einer Optimallösung und einem Enumerationsschema. Sukzessive wird hierbei das Ausgangsproblem in Teilprobleme partitioniert mit dem Ziel, durch Berechnung guter Schranken auf die Betrachtung vieler Teilprobleme verzichten zu können. Obere Schranken werden durch Vereinfachung (Relaxierung) des Ausgangsproblems bestimmt. Ihre Güte bestimmt wesentlich die Leistungsfähigkeit eines Branch & Bound-Verfahren.

4.1 Der polyedertheoretische Ansatz

Ein spezieller Ansatz zur Bestimmung oberer Schranken basiert auf polyedertheoretischen Betrachtungen und liefert im Regelfall bessere Schranken als einfache kombinatorische Relaxierungen. Oft können Probleme sogar ohne Zerlegung in Teilprobleme optimal gelöst werden. Wir skizzieren das prinzipielle Vorgehen beim Entwurf eines polyedrischen Verfahrens zur Lösung eines kombinatorischen Optimierungsproblems.

Im allgemeinen ist ein kombinatorische Optimierungsproblem über einer endlichen Grundmenge $E = \{e_1, e_2, \ldots, e_m\}$ definiert. Die Menge $\mathcal{I}$ seiner zulässigen Lösungen ist eine Teilmenge der Potenzmenge von E. Zur Bewertung der Lösungen gibt es Gewichte c_e für jedes Element e aus E. Das Gewicht einer Lösung $I \in \mathcal{I}$ ist dann gegeben durch $c(I) = \sum_{e \in I} c_e$ und das Optimierungsproblem besteht darin, eine zulässige Lösung mit maximalem Gewicht zu finden. Beim Max-Cut-Problem ist E die Kantenmenge des ungerichteten Graphen G, das System $\mathcal{I}$ besteht aus den Kantenmengen, die Schnitte in G repräsentieren.

Wir transformieren ein solches kombinatorisches Problem in einer Weise, die es erlauben wird, Methoden aus der linearen Optimierung anzuwenden.

Jeder zulässigen Lösung I wird ein **Inzidenzvektor** $\chi^I \in \{0,1\}^m$ zugeordnet über

$$\chi^I_{e_i} = \begin{cases} 0 & \text{falls } e_i \in I, \\ 1 & \text{falls } e_i \notin I. \end{cases}$$

Ein Inzidenzvektor hat also m Komponenten mit Werten 0 oder 1, die mit den Elementen aus E indiziert sind. Im nächsten Schritt bilden wir die konvexe Hülle $P_\mathcal{I}$ der Inzidenzvektoren der zulässigen Lösungen. $P_\mathcal{I}$ ist ein Polytop, dessen Ecken genau den zulässigen Lösungen entsprechen. Prinzipiell könnte daher das kombinatorische Problem als lineares Optimierungsproblem (LP)

$$\max\{c^T x \mid x \in P_\mathcal{I}\}$$

mit Verfahren der linearen Optimierung gelöst werden, die optimale Ecklösungen liefern. Figur 7 zeigt das Polytop der Schnitte zum vollständigen Graphen K_3 auf 3 Knoten. Dieses Polytop hat vier Ecken, die durch die Inzidenzvektoren der

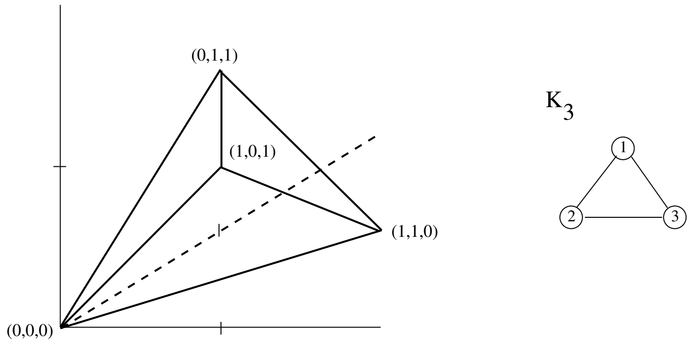

Fig. 7. Das Polytop der Schnitte zum Graphen K_3

Schnitte $(\{1,2,3\} : \emptyset)$, $(\{1\} : \{2,3\})$, $(\{2\} : \{1,3\})$ und $(\{3\} : \{1,2\})$ gegeben sind.

Die Definition von $P_{\mathcal{I}}$ als konvexe Hülle von Inzidenzvektoren ist allerdings algorithmisch unbrauchbar. Um Algorithmen der linearen Optimierung anwenden zu können, benötigen wir eine Beschreibung von $P_{\mathcal{I}}$ mit linearen Gleichungen und Ungleichungen, d. h. eine Darstellung

$$P_{\mathcal{I}} = \{x \in \mathbf{R}^m \mid Bx = d, Ax \leq b\}.$$

Die algorithmisch beste Beschreibung enthält so wenige Gleichungen und Ungleichungen wie möglich. Man hat dann ein minimales Gleichungssystem $Bx = d$ und jede Ungleichung in $Ax \leq b$ entspricht genau einer maximalen Seitenfläche (Facette) von $P_{\mathcal{I}}$.

Wir geben einige Ergebnisse für das Max-Cut-Problem ([BM1]). Sei $G = (V, E)$ ein Graph und $P_{\mathrm{CUT(G)}}$ das zugehörige Polytop definiert als konvexe Hülle der Inzidenzvektoren der Schnitte in G. Wir bezeichnen die Variablen zur Formulierung der Ungleichungen mit x_e für jede Kante $e \in E$. Für $F \subseteq E$ wird mit $x(F)$ die Summe $\sum_{e \in F} x_e$ bezeichnet. Es gelten:

1. Es gibt keine Gleichung, die von allen Punkten aus $P_{\mathrm{CUT(G)}}$ erfüllt wird, d. h. das System $Bx = d$ ist leer.
2. Die Ungleichungen $x_e \geq 0$ und $x_e \leq 1$ sind zulässig für $P_{\mathrm{CUT(G)}}$ für alle $e \in E$. Sie liefern genau dann Facetten, wenn die Kante e in keinem Kreis der Länge 3 von G enthalten ist.
3. Für jeden Kreis C von G und jede Teilmenge F von C mit ungerader Kardinalität ist $x(F) - x(C \setminus F) \leq |F| - 1$ eine zulässige Ungleichung für $P_{\mathrm{CUT(G)}}$. Eine Facette wird genau dann geliefert, wenn C keine Diagonalkanten hat.
4. Sei $E_p \subseteq E$ die Kantenmenge eines vollständigen Untergraphen von G mit p Knoten. Dann ist $x(E_p) \leq \lfloor \frac{p}{2} \rfloor \lceil \frac{p}{2} \rceil$ eine zulässige Ungleichung, die genau dann eine Facette definiert, wenn p ungerade ist.

Es gibt noch sehr viele weitere Klassen von facettendefinierenden Ungleichungen und auch Kompositionstheoreme zur Herleitung von neuen Ungleichungen aus bekannten.

Die Ungleichungen 3. sind die sogenannten **Kreisungleichungen**, die später noch besonders untersucht werden. Für planare Graphen liefern die Kreisungleichungen und die trivialen Ungleichungen $0 \leq x_e \leq 1$ eine vollständige Beschreibung von $P_{\mathrm{CUT(G)}}$.

Das theoretische Studium der Facettialstruktur eines dem Problem zugeordneten Polytops ist der erste wichtige Schritt zur Entwicklung von polyedrischen Methoden. Dieses Studium ist für jedes Problem neu zu leisten. Oft kann man allerdings vorhandene Kenntnisse über verwandte Problemklassen ausnutzen. Aus komplexitätstheoretischen Gründen kann man nicht erwarten, für schwierige Problem eine vollständige lineare Beschreibung des Polytops $P_{\mathcal{I}}$ herleiten zu können, für den praktischen Einsatz genügt aber oft auch eine partielle Beschreibung.

4.2 Schnittebenenverfahren

Nach Herleitung einer polyedrischen Beschreibung ist man von der algorithmischen Umsetzung in der Regel aber noch weit entfernt. Der naheliegende Ansatz, alle bekannten Ungleichungen aufzulisten und dann ein Verfahren der linearen Optimierung zur Lösung des Maximierungsproblems anzuwenden, ist undurchführbar. Auch wenn nur eine partielle lineare Beschreibung bekannt ist, wächst die Zahl der entsprechenden Ungleichungen typischerweise exponentiell mit der Problemgröße.

Die zentrale Idee eines **Schnittebenenverfahrens** besteht darin, Ungleichungen nur bei Bedarf zu generieren. Anschaulich ist klar, daß nur Seitenflächen „in der Nähe“ der Optimallösung relevant sind. Im Prinzip läuft ein solches Verfahren wie folgt ab.

Schnittebenenverfahren

1. Wähle aus den bekannten Ungleichungen die Klassen aus, die im Verfahren berücksichtigt werden sollen.
2. Initialisiere ein Anfangs-Polytop, z. B. $\{x \mid 0 \leq x \leq 1\}$.
3. Bestimme einen Punkt y, der die Zielfunktion c über dem aktuellen Polytop maximiert.
4. Entspricht die Optimallösung einer zulässigen Lösung des kombinatorischen Problems, dann ist dieses gelöst, und das Verfahren kann beendet werden.
5. Ist dies nicht der Fall, so suche verletzte Ungleichungen (d.h. eine der in 1. ausgewählten Ungleichungen, die von y nicht erfüllt werden) und füge diese zum LP hinzu. Gehe zu Schritt 3.

Ist die lineare Beschreibung von $P_{\mathcal{I}}$ nur partiell bekannt, so kann in Schritt 5 der Fall eintreten, daß zwar keine zulässige Lösung vorliegt, aber dennoch keine verletzte Ungleichung gefunden werden kann. Wir gehen auf diese Situation später noch ein. Die Hoffnung ist allerdings, bereits in Schritt 4 eine optimale Lösung zu finden.

Im Laufe des Verfahrens geht man also zu immer besseren Relaxierungen des eigentlichen Problems über. In jedem Schritt 5 wird die Relaxierung durch Hinzunahme weiterer Ungleichungen verschärft. Diese Ungleichungen „schneiden“ den Optimalpunkt des jeweils aktuellen Problems ab. Daher resultiert auch die Bezeichnung Schnittebenenverfahren für diese Klasse von Algorithmen.

Der wesentliche Punkt bei der Realisierung des Verfahrens ist die Erkennung von verletzten Ungleichungen in Schritt 5. Dies ist das sogenannte **Separationsproblem**, welches darin besteht, für die Lösung y der aktuellen LP-Relaxierung festzustellen, ob sie alle Ungleichungen der ausgewählten Klassen erfüllt, und andernfalls eine verletzte Ungleichung zu liefern.

Die grundlegende Bedeutung dieses Problems ist auch theoretisch nachgewiesen. Ein wichtiges Resultat besagt (vereinfacht), daß das zu lösende Ausgangsproblem genau dann polynomial lösbar ist, wenn das Separationsproblem polynomial lösbar ist. Eine ausführliche Darstellung der zugehörigen Theorie ist [GLS1].

Für den erfolgreichen Einsatz von Schnittebenenverfahren ist es also notwendig, das Separationsproblem für möglichst große Klassen von Ungleichungen effizient lösen zu können. Wir skizzieren einen Algorithmus zur Lösung des Separationsproblems für die Kreisungleichungen.

Separation für Kreisungleichungen

1. Sei y die Lösung des aktuellen linearen Programms. Wir können annehmen, daß $0 \leq y \leq 1$ erfüllt ist.
2. Konstruiere aus $G = (V, E)$ einen neuen Graphen $H = (V' \cup V'', E' \cup E'' \cup E''')$, der aus zwei Kopien $G' = (V', E')$ und $G'' = (V'', E'')$ von G und den folgenden zusätzlichen Kanten E''' besteht. Für jede Kante uv des Originalgraphen erzeugen wir zwei Kanten $u'v''$ und $u''v'$. Den Kanten $u'v'$ und $u''v''$ wird das Gewicht y_{uv} zugeordnet, die Kanten $u'v''$ und $u''v'$ erhalten das Gewicht $1 - y_{uv}$.
3. Für jedes Knotenpaar u' und u'' wird der kürzeste verbindende Weg (bezüglich der neu definierten Kantengewichte) in H berechnet. Ein solcher Weg enthält eine ungerade Anzahl von Kanten aus E''' und liefert zurückinterpretiert in G einen geschlossenen Kantenzug, der u enthält. Hat ein solcher (u',u'')-Weg eine Länge kleiner als 1, dann existiert ein Kreis $C \subseteq E$ und eine Kantenmenge $F \subseteq C$ ungerader Kardinalität, so daß y die zugehörige Kreisungleichung verletzt (C und F können leicht identifiziert werden).
4. Ist keiner dieser Wege kürzer als 1, so werden alle Kreisungleichungen von der aktuellen LP-Lösung erfüllt.

Das Separationsproblem für Kreisungleichungen kann somit durch Berechnung von V kürzesten Wegen in polynomialer Zeit gelöst werden, ist aber für große praktische Probleme unter Umständen nicht schnell genug, um häufig aufgerufen werden zu können. Man wird daher immer versuchen, in den ersten Phasen des Schnittebenenverfahrens schnelle Heuristiken zum Auffinden verletzter Ungleichungen einzusetzen und erst als letztes Hilfsmittel auf langsamere exakte Verfahren zurückgreifen.

Bei der praktische Lösung von allgemeinen Max-Cut-Problemen werden Heuristiken und exakte Verfahren für weitere Klassen von Ungleichungen eingesetzt.

4.3 Branch-and-Cut

Wie bereits erwähnt, kann problemabhängig immer der Fall eintreten, daß das reine Schnittebenenverfahren nicht zum Ziel führt, da zu irgendeinem Zeitpunkt zum „Abschneiden“ einer unzulässigen Lösung keine Ungleichung mehr gefunden wird. In einer solchen Situation muß man auf das Branch & Bound-Prinzip zurückgreifen und sukzessive Teilprobleme, die nicht gelöst werden können, weiter zerlegen. Im Gegensatz zum üblichen Branch & Bound-Verfahren hat man aber mit Schnittebenen in der Regel größere Chancen, sehr gute obere Schranken zu finden und den Zerlegungsaufwand in engeren Grenzen zu halten. Schnittebenen lassen sich an jedem Knoten des Verzweigungsbaums weiterhin einsetzen. Man bezeichnet die hier beschriebene Verfahrensklasse allgemein als **Branch & Cut-Verfahren**, da sie eine Synthese aus Schnittebenenverfahren und Methoden der impliziten Enumeration darstellen.

Zur konkreten Implementierung sind nicht nur effiziente Separationsverfahren zu entwickeln, sondern es sind noch einige weitere Aspekte zu beachten. Wir führen einige dieser Fragen auf.

1. Welches System soll in das Anfangs-LP eingehen?
2. Soll das minimale Gleichungssystem (falls vorhanden) explizit ins LP aufgenommen werden oder zur Elimination von Variablen verwendet werden?
3. In welcher Reihenfolge sollen Separationsverfahren angewendet werden?
4. Welche gefundenen Schnittebenen sollen zum LP hinzugefügt werden?
5. Sollen im Verlauf des Verfahrens Ungleichungen eliminiert werden, um das LP nicht zu groß werden zu lassen?
6. Soll man anfangs nur eine Teilmenge der Variablen berücksichtigen?
7. Welche Verzweigungsstrategie soll verwendet werden?
8. Wie sollen zulässige Lösungen bestimmt werden?
9. Welche Software soll zur Lösung der LPs benutzt werden?

Die Antworten auf diese Fragen können problemabhängig unterschiedlich ausfallen. Erfolgreiche Implementierungen basieren auf Erfahrung und problemspezifischen Maßnahmen. Ausführliche Darstellungen von Branch-and-Cut-Verfahren zur Lösung des Max-Cut-Problems sind [BGJR1] und [BJR1]. Eine allgemeine Diskussion des Entwurfs von Branch-and-Cut-Verfahren ist [JRT1].

4.4 Optimallösungen der Via-Minimierungsprobleme

Tabelle 6 zeigt die Ergebnisse, die mit einem Schnittebenenverfahren für die 12 Beispielprobleme erzielt wurden.

Da es sich jeweils um planare Graphen handelte, konnten alle Probleme mit den Kreisungleichungen (ohne Branch-and-Bound) gelöst werden. Die Rechenzeiten sind klein genug, so daß die exakte Optimierung auch in der Praxis eingesetzt werden kann. Die Berücksichtigung von Lagenbevorzugung wird in [GJR1] diskutiert.

Der Vergleich der Minimalzahl erforderlicher Vias mit den heuristisch gefundenen Lösungen zeigt, daß die Baum-Heuristik mit anschließender Verbesserung

Tabelle 6. Optimallösungen

Problem	Optimum		Reduktion
	Vias	CPU	(in %)
VIA1n	272	3.43	
VIA1y	302	6.92	28.3
VIA2n	338	3.27	
VIA2y	373	17.50	14.1
VIA3n	509	14.47	
VIA3y	563	25.90	17.6
VIA4n	475	3.95	
VIA4y	504	13.22	22.5
VIA5n	602	10.52	
VIA5y	645	14.28	17.6
VIA6n	502	7.22	
VIA6y	560	11.53	7.9

durch die Kernighan-Lin-Heuristk oder lokale Enumeration durchaus brauchbare Ergebnisse liefert. In einigen Fällen wurden sogar Optimallösungen bestimmt. Gegenüber den Originallösungen ergaben sich Einsparungen von 7.9% bis 28.3% der Vias. Kann auf eine Fixierung der Anschlüsse verzichtet werden, so ergeben sich weitere Einsparungen.

Literatur

[Ba1] Barahona, F.: On the Computational Complexity of Ising Spin Glass Models. Journal of Physics A: Math. Gen. **15** (1982) 3241–3253

[BJR1] Barahona, F., Jünger, M., Reinelt, G.: Experiments in Quadratic 0-1-Programming. Mathematical Programming **44** (1989) 127–137

[BGJR1] Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An Application of Combinatorial Optimization to Stastistical Physics and Circuit Layout Design. Operations Research **36** (1989) 493–513

[BM1] Barahona, F., Mahjoub, A.R.: On the Cut Polytope. Mathematical Programming **36** (1986) 157–173

[CKC1] Chen, R.-W., Kajitani, Y., Chan, S.-P.: A Graph-theoretic Via Minimization Algorithm for Two-layer Printed Circuit Boards. IEEE Transactions on Circuits and Systems **30** (1984) 291–308

[GJR1] Grötschel, M., Jünger, M., Reinelt, G.: Via Minimization with Pin Preassignments and Layer Preference. Zeitschrift für Angewandte Mathematik und Mechanik **69** (1989) 393–399

[GLS1] Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer-Verlag (1988)

[Ha1] Hadlock, F.O.: Finding a Maximum Cut of a Planar Graph in Polynomial Time. SIAM Journal on Computing **3** (1975) 221–225

[JRT1] Jünger, M., Reinelt, G., Thienel, S.: Practical Problem Solving with Cutting Plane Algorithms in Combinatorial Optimization. erscheint in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1994)

[KL1] Kernighan, B., Lin, S.: An Efficient Heuristic Procedure for Partitioning Graphs. Bell Systems Technical Journal **49** (1979) 291–308

[LA1] Laarhoven, P.J.M. van, Aarts, E.H.L.: Simulated Annealing: Theory and Applications. Reidel, Dordrecht (1987)

[Le1] Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. John Wiley & Sons (1990)

[Mo1] Molitor, P.: On the Contact Minimization Problem. in: Proc. of the Fourth Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 247, Springer-Verlag (1986)

[Mu1] Mutzel, P.: Implementierung und Analyse eines Max-Cut-Algorithmus für planare Graphen. Diplomarbeit, Universität Augsburg (1990)

[NW1] Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. John Wiley & Sons (1988)

[OD1] Orlova, G.I., Dorfman, Y.G.: Finding the Maximal Cut in a Graph. Engineering Cybernetics **10** (1972) 502–506

[Pi1] Pinter, R.Y.: Optimal Layer Assignment for Interconnect. Journal of VLSI and Computer Systems **1** (1984) 123–137

This article was processed using the LaTeX macro package with LMAMULT style