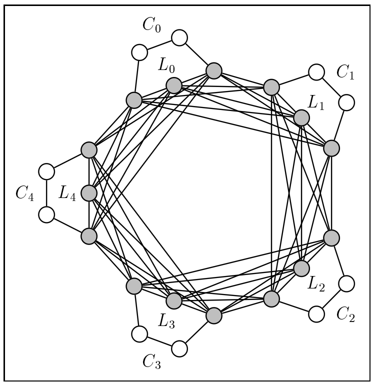
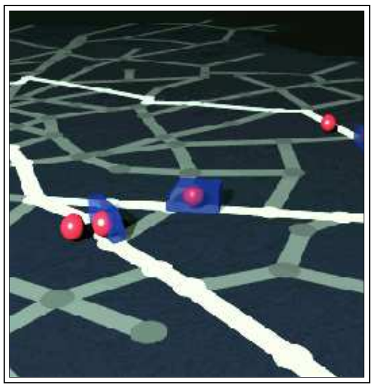
Aspects of Set Packing, Partitioning, and Covering

Ralf Borndörfer





Aspects of Set Packing, Partitioning, and Covering

---

vorgelegt von Diplom-Wirtschaftsmathematiker

Ralf Borndörfer

Vom Fachbereich 3 Mathematik der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

genehmigte Dissertation

Promotionsausschuß

Vorsitzender	Prof. Dr. Kurt Kutzler
Berichter	Prof. Dr. Martin Grötschel
Berichter	Prof. Dr. Robert Weismantel

Tag der wissenschaftlichen Aussprache: 30. März 1998

Berlin 1998
D 83

Zusammenfassung

Diese Dissertation befaßt sich mit ganzzahligen Programmen mit 0/1 Systemen: Set-Packing-, Partitioning- und Covering-Probleme. Die drei Teile der Dissertation behandeln polyedrische, algorithmische und angewandte Aspekte derartiger Modelle.
Teil 1 diskutiert polyedrische Aspekte. Den Auftakt bildet eine *Literaturübersicht* in Kapitel 1. In Kapitel 2 untersuchen wir *Set-Packing-Relaxierungen* von kombinatorischen Optimierungsproblemen über Azyklische Digraphen und Lineare Ordnungen, Schnitte und Multischnitte, Überdeckungen von Mengen und über Packungen von Mengen. Familien von Ungleichungen für geeignete Set-Packing-Relaxierungen sowie deren zugehörige Separierungsalgorithmen sind auf diese Probleme übertragbar.
Teil 2 ist algorithmischen und rechnerischen Aspekten gewidmet. Wir dokumentieren in Kapitel 3 die wesentlichen Bestandteile eines *Branch-And-Cut Algorithmus* zur Lösung von Set-Partitioning-Problemen. Der Algorithmus implementiert einige der theoretischen Ergebnisse aus Teil 2. Rechenergebnisse für Standardtestprobleme der Literatur werden berichtet.
Teil 3 ist angewandt. Wir untersuchen die Eignung von Set-Partitioning-Methoden zur Optimierung des Berliner Behindertenfahrdienstes *Telebus*, der mit einer Flotte von 100 Fahrzeugen täglich etwa 1.500 Fahrwünsche bedient. Der Branch-And-Cut Algorithmus aus Teil 2 ist ein Bestandteil eines Systems zur Fahrzeugeinsatzplanung, das seit dem 3. Juni 1995 in Betrieb ist. Dieses System ermöglichte Verbesserungen im Service und gleichzeitig erhebliche Kosteneinsparungen.

Schlüsselbegriffe. Ganzzahlige Programmierung, Polyedrische Kombinatorik, Schnittebenen, Branch-And-Cut, Anrufsammeltaxisysteme, Fahrzeugeinsatzplanung

Mathematics Subject Classification (MSC 1991). 90C10

Abstract

This thesis is about integer programs with 0/1 constraint systems: Set packing, partitioning, and covering problems. The three parts of the thesis investigate polyhedral, algorithmic, and application aspects of such models.

Part 1 discusses polyhedral aspects. Chapter 1 is a prelude that *surveys* results on integer 0/1 programs from the literature. In Chapter 2 we investigate *set packing relaxations* of combinatorial optimization problems associated with acyclic digraphs and linear orderings, cuts and multicuts, multiple knapsacks, set coverings, and node packings themselves. Families of inequalities that are valid for such a relaxation and the associated separation routines carry over to the problems under investigation.

Part 2 is devoted to algorithmic and computational aspects. We document in Chapter 3 the main features of a *branch-and-cut algorithm* for the solution of set partitioning problems. The algorithm implements some of the results of the theoretical investigations of the preceding part. Computational experience for a standard test set from the literature is reported.

Part 3 deals with an application. We consider in Chapter 4 set partitioning methods for the optimization of Berlin's *Telebus* for handicapped people that services 1,500 requests per day with a fleet of 100 mini busses. Our branch-and-cut algorithm of Part 2 is one module of a scheduling system that is in use since June 3, 1995 and resulted in improved service and significant cost savings.

Keywords. Integer Programming, Polyhedral Combinatorics, Cutting Planes, Branch-and-Cut, Vehicle Scheduling, Dial-A-Ride Systems

Mathematics Subject Classification (MSC 1991). 90C10

Preface

Aspects of set packing, partitioning, and covering is the title of this thesis, and it was chosen deliberately. The idea of the thesis is to try to bend the bow from *theory* via *algorithms* to a practical *application*, but the red thread is not always pursued conclusively. This resulted in three parts that correspond to the three parts of the bow and belong together, but that can also stand for themselves. This *self-containment* is reflected in separate indices and reference lists.

There is no explanation of *notation* or basic concepts of optimization. Instead, I have tried to resort to standards and in particular to the book Grötschel, Lovász & Schrijver (1988), *Geometric Algorithms and Combinatorial Optimization*, Springer Verlag, Berlin.

It is perhaps also useful to explain the system of *emphasis* that is at the bottom of the writing. Namely, emphasized words exhibit either the topic of the current paragraph and/or they mark contents of the various indices, or they sometimes just stress a thing.

I am grateful to the Senate of Berlin's Departments for Science, Research, and Culture and for Social Affairs that supported the Telebus project and to Fridolin Klostermeier and Christian Küttner for their cooperation in this project. I am indebted to the Konrad-Zuse-Zentrum for its hospitality and for its support in the publication of this thesis.

I would like to thank my supervisor Martin Grötschel for his example not only as a mathematician and especially for his patience. I also thank Andreas Schulz and Akiyoshi Shioura who have kindly pointed out a number of errors in an earlier version of this thesis. My friends Norbert Ascheuer, Bob Bixby, and Alexander Martin have helped me with many discussions on aspects of this thesis and I want to express my gratitude for this. A special thanks goes to Andreas Löbel for his friendship and support. My last special thanks goes to my friend Robert Weismantel. I simply want to say that without him not only this thesis would not be as it is.

I hope that whoever reads this can profit a little from these notes — and perhaps even enjoy them.

Berlin, August 1998 — Ralf Borndörfer

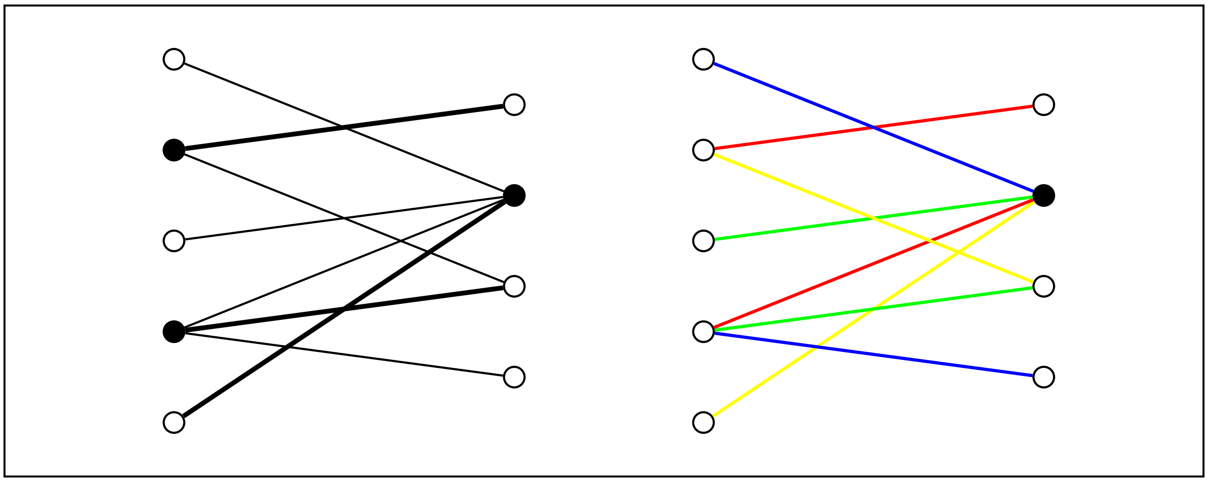
Contents

List of Tables

List of Figures

Part I

Polyhedral Aspects



Chapter 1

Integer 0/1 Programs

Summary. This chapter tries to survey some of the main results of the literature for integer programming problems associated with set packings, set partitionings, and set coverings: Blocking and anti-blocking theory, the field of perfect, ideal, and balanced matrices, and the results about the facial structure of set packing and set covering polyhedra.

1.1 Two Classical Theorems of König (Introduction)

König's book *Theorie der endlichen und unendlichen Graphen* of 1936 is the first systematic treatment of the mathematical discipline of *graph theory*[1]. Two hundred years after Euler's famous primer[2] on the bridges of Königsberg gave birth to this area of discrete mathematics, it was König's aim to establish his subject as "a branch of *combinatorics* and abstract *set theory*"[3]. In this spirit, he investigated structural properties of general and of special classes of graphs. Among the latter, *bipartite graphs* are the subject of two of his most famous theorems, the *König-Egerváry* and the *edge coloring theorem.* In his own words*, these results read as follows.

[1]Sachs [1986], page 314 of König [1986]. König [1936] himself made an effort to compile all previous references on graph theory.

[2]Euler attributes the notion of graph theory or *geometria situs*, as he called it, to Leibniz and gives some references in this direction, see Euler [1736], page 279 of König [1986].

[3]König [1936], preface, page 9: "Second one can perceive it [the theory of graphs] — abstracting from its continuos-geometric content — as a branch of *combinatorics* and abstract *set theory.* This book wants to emphasize this second point of view ..."*.

*Translation by the author.

1.1.1 Theorem (König-Egerváry Theorem, König [1931][4])
For any bipartite graph is the minimum number of nodes that drain the edges of the graph equal to the maximum number of edges that pairwise do not possess a common endpoint.

Here, we say that the nodes $A_1, A_2, \ldots, A_\nu$ «drain» the edges of a graph if every edge of the graph ends in one of the points $A_1, A_2, \ldots, A_\nu$.

1.1.2 Theorem (Edge Coloring Theorem, König [1936][5])
If at most g edges come together in every node of a finite bipartite graph G, one can subdivide all edges of the graph into g classes in such a way that every two edges, that come together in a node, belong to different classes.

König's theorems can be seen as combinatorial *min-max theorems*, and they are among the earliest known results of this type. Min-max theorems state a duality relation between two *optimization problems*, one a minimization and the other a maximization problem, hence the name. In the König-Egerváry case, these optimization problems are the minimum *node covering problem* and the maximum *matching problem* in a bipartite graph; the theorem states that the optimum solutions, the minimum *covers* and the maximum *matchings*, are of equal size. The edge coloring theorem involves also two optimization problems: The (trivial) task to compute the maximum degree in a bipartite graph is related to the problem to determine the minimum number of colors in an *edge coloring*. The relation is again that the best such values are equal. See the top of page 3 for an illustration of the two König theorems.
Min-max results are important from an *optimization* point of view, because they provide simple *certificates of optimality*. For example, to disperse any doubt whether some given cover is minimal, one can exhibit a matching of the same size. The technique works also the other way round, or it can be used to prove lower or upper *bounds* on the size of a minimal cover or maximum matching, respectively. And, most important of all, optimality criteria are the first step to design combinatorial optimization *algorithms*.
It goes without saying that the relevance of his theorems was more than clear to König[6] and he devoted two entire sections of his book[7] to their consequences. König showed, for instance, that the popular (but fortunately rarely applied) *marriage theorem* can be derived in this way. He noticed also that the two theorems *themselves* are related and proved that the edge coloring theorem *follows* from the König-Egerváry theorem[8]. Reading his book one has the impression that König looked at the first as a weaker result than the latter, and we could find no evidence that he considered the reverse implication. But we know today that exactly this is also true: It is one of the consequences of Fulkerson [1971]'s powerful *anti-blocking theory*, developed about 40 years later, that the König-Egerváry and the edge coloring theorem are *equivalent*. This means that for bipartite graphs not only node covering and matching are dual problems as well as edge coloring is dual to degree computation, but, going one step further, these two min-max relations form again a dual pair of equivalent *companion theorems*, as Fulkerson called it.

[4] See also König [1936], Theorem XIV 13/14, page 249 of König [1986].
[5] See König [1936], Theorem XI 15, page 187 of König [1986].
[6] König [1936], page 191 of König [1986]: "Theorem [XIV] 13 is an important theorem that can be applied to problems of very different nature ..."* [Applications follow.]. Page 191 of König [1986]: "Theorem [XI] 13 [that is equivalent to the edge coloring Theorem XI 15] can be applied to various combinatorial problems ..."* [Applications follow.]
[7] König [1936], XI § 5 (edge coloring) and XIV § 3 (König-Egerváry). See (in both cases) also the preceeding paragraphs.
[8] König [1936], page 250 of König [1986].

Let's go through an *application* of anti-blocking theory to the König-Egerváry/edge coloring setting now to see how this theory works. The anti-blocking relation deals with *integer programs* of a certain "packing" type, and we start by formulating a *weighted generalization* of the matching problem in this way, the *bipartite matching problem* (BMP). Taking A as the node-edge incidence matrix of the bipartite graph of interest (a row for each node, a column for each edge), this BMP can be formulated as the *weighted packing problem*

$$\text{(BMP)} \quad \max\ w^{\mathrm{T}}x \quad Ax \leq \mathbf{1},\ x \geq 0,\ x \text{ binary.}$$

Here, $\mathbf{1}$ is a vector of all ones of compatible dimension, w is a vector of nonnegative integer weights, and taking $w := \mathbf{1}$ is to look for a matching of maximum cardinality. The "packing structure" in (BMP) is that the constraint system is 0/1 and of the form $Ax \leq \mathbf{1},\ x \geq 0$. Note that "matching" is a synonym for "edge packing", hence the name.

Now we apply a sequence of transformations to this program: Removing the integrality stipulations, taking the dual, and requiring the dual variables to be integral again

$$\begin{array}{llllllll} \max & w^{\mathrm{T}}x & \leq \max & w^{\mathrm{T}}x & = \min & y^{\mathrm{T}}\mathbf{1} & \leq \min & y^{\mathrm{T}}\mathbf{1} \\ & Ax \leq \mathbf{1} & & Ax \leq \mathbf{1} & & y^{\mathrm{T}}A \geq w^{\mathrm{T}} & & y^{\mathrm{T}}A \geq w^{\mathrm{T}} \\ & x \geq 0 & & x \geq 0 & & y^{\mathrm{T}} \geq 0^{\mathrm{T}} & & y^{\mathrm{T}} \geq 0^{\mathrm{T}} \\ & x \text{ integral} & & & & & & y^{\mathrm{T}} \text{ integral,} \end{array} \tag{1.1}$$

we arrive at another integer program on the right. This program is the weighted *bipartite node covering problem* (BCP) of edges by nodes

$$\text{(BCP)} \quad \min\ y^{\mathrm{T}}\mathbf{1} \quad y^{\mathrm{T}}A \geq w^{\mathrm{T}},\ y^{\mathrm{T}} \geq 0^{\mathrm{T}},\ y^{\mathrm{T}} \text{ integral.}$$

(BCP) is an example of a *weighted covering problem*, which means in general that the constraint system is of the form $y^{\mathrm{T}}A \geq w^{\mathrm{T}},\ y^{\mathrm{T}} \geq 0$ with a 0/1 matrix A and arbitrary integer weights w on the right-hand side.

But the BCP is, for $w = \mathbf{1}$, exactly the node covering problem of the König-Egerváry theorem! This relation allows us to paraphrase Theorem 1.1.1 in integer programming terminology as follows: For $w = \mathbf{1}$, the optimum objective values of the packing problem (BMP) and of the associated covering problem (BCP) are equal.

The key point for all that follows now is that this equality does not only hold for $w = \mathbf{1}$, but for *any* integral vector w. In other words, a weighted generalization of the König-Egerváry theorem as above holds, and this is equivalent to saying that the constraint system of the packing program (BMP) is *totally dual integral* (TDI). This situation —a TDI packing system $Ax \leq \mathbf{1}, x \geq 0$ with 0/1 matrix A— is the habitat of anti-blocking theory and whenever we can establish it, the anti-blocking machinery automatically gives us a second *companion* packing program, again with TDI constraint system and associated min-max theorem! In the König-Egerváry case, the companion theorem will turn out to be a weighted generalization of the edge coloring theorem for bipartite graphs.

The companion program is constructed as follows. We first set up the 0/1 incidence matrix B of all solutions of the packing program, i.e., in our case of all matchings versus edges (a row for each matching, a column for each edge). This matrix is called the *anti-blocker* of A; it serves as the constraint matrix of the companion packing program and its associated dual

$$\begin{array}{llllllll} \max & w^{\mathrm{T}}x & \leq \max & w^{\mathrm{T}}x & = \min & y^{\mathrm{T}}\mathbf{1} & \leq \min & y^{\mathrm{T}}\mathbf{1} \\ & Bx \leq \mathbf{1} & & Bx \leq \mathbf{1} & & y^{\mathrm{T}}B \geq w^{\mathrm{T}} & & y^{\mathrm{T}}B \geq w^{\mathrm{T}} \\ & x \geq 0 & & x \geq 0 & & y^{\mathrm{T}} \geq 0^{\mathrm{T}} & & y^{\mathrm{T}} \geq 0^{\mathrm{T}} \\ & x \text{ integral} & & & & & & y^{\mathrm{T}} \text{ integral.} \end{array} \tag{1.2}$$

The main result of anti-blocking theory is that, if the original packing program had a TDI constraint system, the companion packing program has again a TDI constraint system. This means that all inequalities in the sequence (1.2) hold with equality for all integral weights w, and this is the companion min-max theorem.

What does the companion theorem say in the König-Egerváry case for $w = \mathbf{1}$? The solutions of the left integer program in (1.2) are edge sets that intersect every matching at most once. Sets of edges that emanate from an individual node have this property, and a minute's thought shows that these are all possible solutions. $w = \mathbf{1}$ means to look for a largest such set, i.e., to compute the maximum node degree; this is one half of the edge coloring theorem. The second integer program on the right of (1.2) provides the second half, because it asks for a minimum cover of edges by matchings. But as the matchings are exactly the feasible color classes for edge colorings, the integer program on the right asks for a minimum edge coloring. And arbitrary weights give rise to a weighted generalization of the edge coloring theorem.

We can thus say that the weighted version of the König-Egerváry theorem implies, by virtue of anti-blocking theory, the validity of a companion theorem which is a weighted generalization of the edge coloring theorem. One can work out that it is possible to reverse this reasoning such that these two theorems form an equivalent pair. And one finally obtains the two König theorems by setting $w := \mathbf{1}$. The reader will have noticed that, in contrast to what we have claimed on page 4, this anti-blocking argument does *not* prove the equivalence of the two unweighted König theorems, that both only follow from their (equivalent) weighted relatives. Well — sometimes it's clearer to lie a little!

*

Our discussion of König's considerations was already in terms of weighted versions of his theorems, and further generalizations take us directly to today's areas of research on integer 0/1 programming problems.

The first question that comes up is whether TDI results with dual pairs of min-max theorems also hold for other 0/1 matrices than the incidence matrices of bipartite graphs? This question leads to *perfect graph theory*, where Lovász [1971] has shown that dual min-max theorems on stable sets and clique coverings on the one hand and cliques and node colorings on the other hold exactly for *perfect matrices*, the clique matrices of *perfect graphs*. This famous result, that was conjectured by Berge [1961] and is known as the *perfect graph theorem*, does *not* imply that the four optimization problems that we have just mentioned can be solved in time that is *polynomial* in the input length of the perfect *graph* and the objective, because the associated *clique matrix* and its anti-blocker can be exponentially large. But exactly this is nevertheless possible! Fundamental algorithmic results of Grötschel, Lovász & Schrijver [1988], often termed the *polynomial time equivalence of separation and optimization*, and techniques of *semidefinite programming* were the key innovations for this breakthrough.

Another appealing topic on perfect graphs and their clique matrices are *recognition problems*. An important result in this area, which follows from results of Padberg [1973b, 1976] but was first stated and proved (in a different way) by Grötschel, Lovász & Schrijver [1984], is that the recognition of perfect graphs is in co-$\mathcal{NP}$. This question, as well as the unsolved problem whether one can certify in polynomial time that a given graph is perfect or, weaker, whether a given 0/1 matrix is perfect, is intimately related to a stronger and also unresolved version of Berge's conjecture. This *strong perfect graph conjecture* states that a graph is perfect if and only if it does not contain an *odd hole* or its complement; it is known to hold for several subclasses of perfect graphs.

Another direction of research considers general 0/1 matrices that do not correspond to clique matrices of perfect graphs. The LP relaxations of the packing (and the dual covering) problems associated to such matrices are not integral, much less TDI, and min-max theorems do not hold in general. To solve such packing problems with LP techniques, additional inequalities are needed. One branch of research, pioneered by Padberg [1973a], is concerned with finding not only any feasible, but in a sense best possible *facet defining* inequalities and to develop computationally useful procedures to find them. For special classes of 0/1 matrices it is sometimes not only possible to determine some facets, but to obtain a *complete description*, i.e., a list of *all* facet defining inequalities. In such cases, there is a chance that it is possible to develop polynomial LP based or combinatorial optimization algorithms for the four optimization problems that come up in packing: Maximum stable set, minimum clique covering, maximum clique, and minimum coloring. And in very rare instances, complete descriptions give even rise to TDI systems with associated min-max theorems.

Analogous problems as in the packing case, but much less complete results exist for *set covering problems*. One obtains the four optimization problems of this area by simply reversing all inequalities in the four packing analogues. But this "technique" does not carry over to all theorems and proofs! It is in particular *not* true that every covering min-max theorem has an equivalent companion theorem, and the connection to graph theory is much weaker than in the packing case. The well behaved 0/1 matrices are called *ideal*, but there are no algorithmic results as for perfect matrices. The study of facet defining inequalities for the nonideal case seems to be more difficult as well and little is known here, but comparable (even though more difficult) results exist for the recognition of ideal matrices.

Finally, one can look at the equality constrained *partitioning* case, that leads to the consideration of a certain class of *balanced matrices*. These matrices give rise to partitioning programs with integer LP relaxations, but the balanced matrices are only a subclass of all matrices with this property. A spectacular result in this area is the recent solution of the recognition problem by Conforti, Cornuéjols & Rao [1991]. There are no investigations to determine further inequalities for programs with unbalanced matrices, because this question reduces to the packing and covering case.

*

The following eight sections of this chapter give a more detailed survey on results for the set packing, the set partitioning, and the set covering problem. Section 1.2 gives basic definitions and references to survey articles. Section 1.3 describes the fundamental connections of set packing to graph theory and of set covering to independence systems. Blocking and anti-blocking theory is visited a first time in Section 1.4. This topic extends to Section 1.5, where we discuss perfect and ideal matrices and the associated famous min-max results, the perfect graph theorem with its many variants and the width-length and max flow-min cut properties of ideal matrices. Section 1.6 is about the recognition of perfect and ideal matrices and, closely related, their characterization in terms of forbidden minors. Balanced matrices are treated in a separate Section 1.7. The last two sections survey polyhedral results. Section 1.8 deals with the set packing polytope, and Section 1.9 with the set covering polytope.

1.2 The Set Packing, Partitioning, and Covering Problem

Let A be an $m \times n$ 0/1 matrix and w an integer n-vector of weights. The *set packing* (SSP), the *set partitioning* (SPP), and the *set covering problem* (SCP) are the *integer 0/1 programs*[9]

$$\text{(SSP)}\quad \begin{aligned}\max\ & w^{\mathrm{T}}x\\ Ax &\le \mathbf{1}\\ x &\ge 0\\ x &\in \{0,1\}^n\end{aligned} \qquad \text{(SPP)}\quad \begin{aligned}\min\ & w^{\mathrm{T}}x\\ Ax &= \mathbf{1}\\ x &\ge 0\\ x &\in \{0,1\}^n\end{aligned} \qquad \text{(SCP)}\quad \begin{aligned}\min\ & w^{\mathrm{T}}x\\ Ax &\ge \mathbf{1}\\ x &\ge 0\\ x &\in \{0,1\}^n.\end{aligned}$$

Associated to these three programs are six polyhedra:

$$\begin{aligned} P_I(A) &:= \operatorname{conv}\{x \in \{0,1\}^n : Ax \le \mathbf{1}\} & P(A) &:= \operatorname{conv}\{x \in \mathbb{R}_+^n : Ax \le \mathbf{1}\}\\ P_I^=(A) &:= \operatorname{conv}\{x \in \{0,1\}^n : Ax = \mathbf{1}\} & P^=(A) &:= \operatorname{conv}\{x \in \mathbb{R}_+^n : Ax = \mathbf{1}\}\\ Q_I(A) &:= \operatorname{conv}\{x \in \{0,1\}^n : Ax \ge \mathbf{1}\} & Q(A) &:= \operatorname{conv}\{x \in \mathbb{R}_+^n : Ax \ge \mathbf{1}\}.\end{aligned}$$

The *set packing polytope* $P_I(A)$, the *set partitioning polytope* $P_I^=(A)$, and the *set covering polytope* $Q_I(A)$ are defined as the convex hull of the set of feasible solutions of (SSP), (SPP), and (SCP), respectively, the polyhedra $P(A)$, $P^=(A)$, and $Q(A)$ denote *fractional* relaxations (*fractional set packing polytope* etc.). The *fundamental theorem of linear programming*, that guarantees the existence of an optimal basic (vertex) solution, allows to state the three integer programs above as linear programs over the respective integer polytope:

$$\text{(SSP)}\quad \begin{aligned}\max\ & w^{\mathrm{T}}x\\ x &\in P_I(A)\end{aligned} \qquad \text{(SPP)}\quad \begin{aligned}\min\ & w^{\mathrm{T}}x\\ x &\in P_I^=(A)\end{aligned} \qquad \text{(SCP)}\quad \begin{aligned}\min\ & w^{\mathrm{T}}x\\ x &\in Q_I(A).\end{aligned}$$

Let us quickly point out some technicalities. (i) *Empty columns* or *rows* in the constraint matrix A are either redundant, lead to unboundedness, or to infeasibility, and we can assume without loss of generality that A does not contain such columns or rows. (ii) If A does not contain empty rows or columns, $P_I(A)$ and $Q_I(A)$ are always nonempty, but $P_I^=(A) = \emptyset$ is possible. (iii) By definition, $P_I^=(A) = P_I(A) \cap Q_I(A)$, i.e., it is enough to study $P_I(A)$ and $Q_I(A)$ to know $P_I^=(A)$. (iv) The set covering polytope $Q_I(A)$, as we have defined it, is bounded, but the relaxation $Q(A)$ is *not*. This "trick" is convenient for duality arguments and does not give away information because all vertices of $Q(A)$ lie within the unit cube. (v) The two packing polytopes $P_I(A)$ and $P(A)$ are *down monotone*, the covering polyhedra $Q_I(A)$ and $Q(A)$ are (in slightly different senses) *up monotone*. (vi) These observations can be used to assume w.l.o.g. that set packing or covering problems have a *nonnegative* (or positive) *objective*, and so for set partitioning problems as well by adding appropriate multiples of rows to the objective. (vii) Similar techniques allow *transformations* between the three integer 0/1 programs, see Garfinkel & Nemhauser [1972] and Balas & Padberg [1976] for details.

All three integer 0/1 programs have interpretations in terms of *hypergraphs* that show their *combinatorial significance* and explain their names. Namely, look at A as the edge-node incidence matrix of a hypergraph $\mathcal{A}$ (on the groundset $\{1,\ldots,n\}$ of columns of A) with node weights w_j. Then the packing problem asks for a maximum weight set of nodes that intersects all edges of $\mathcal{A}$ at most once, a maximum *packing*, the covering case is about a minimum weight set that intersects each edge at least once, a minimum *cover* or (old fashioned) *transversal*, while in the last case a best *partition* of the groundset has to be determined.

[9] We distinguish "0/1 integer programs" with 0/1 variables and "integer 0/1 programs" with 0/1 matrices.

We suggest the following survey articles on integer 0/1 programs: Fulkerson [1971] (blocking and anti-blocking theory), Garfinkel & Nemhauser [1972, Chapter 8] (set partitioning, set covering), Balas & Padberg [1976] (applications, set packing, set partitioning, set packing polytope, algorithms), Padberg [1977, 1979] (set packing polytope), Schrijver [1979] (blocking and anti-blocking, perfection, balancedness, total unimodularity, extensions), Lovász [1983] (perfect graphs), Grötschel, Lovász & Schrijver [1988] (set packing polytope, perfect graphs), Ceria, Nobili & Sassano [1997] (set covering), Conforti et al. [1994] and Conforti, Cornuéjols, Kapoor & Vušković [1997] (perfect, ideal, and balanced 0/1 and $0/\pm 1$ matrices), Schrijver [1986, Chapter 9 & 22] (textbook), and finally Balinski [1965] as a "historical" article.

1.3 Relations to Stable Sets and Independence Systems

We discuss in this section two insights that are the foundations for the combinatorial study of the set packing and the set covering problem: The correspondence between set packings and stable sets, that builds the bridge from packing 0/1 integer programs to graph theory, and the relation of set covering to independence systems.

$$
\begin{array}{llllllll}
x_1 & & +x_3 & & & & \le 1 \\
 & & x_3 & +x_4 & & +x_6 & \le 1 \\
 & x_2 & & +x_4 & +x_5 & & \le 1 \\
x_1 & +x_2 & & & & & \le 1 \\
 & & & & & x & \in \{0,1\}^6
\end{array}
$$

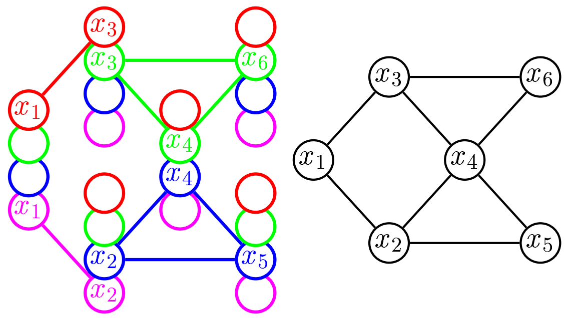


Figure 1.1: Constructing a Column Intersection Graph.

We start with *set packing*. Edmonds [1962, last two sentences on page 498] came up with the idea to associate to a set packing problem (SSP) the following *conflict* or *column intersection graph* $G(A)$: The nodes of $G(A)$ are the column(indice)s of A, and there is an edge between two column(node)s i and j if they intersect, i.e., $A_{i\cdot} \cdot A_{j\cdot} \neq 0$, see Figure 1.1. The construction has the property that the incidence vectors of *stable sets* in $G(A)$, i.e., sets of pairwise nonadjacent nodes, are exactly the feasible solutions of the packing program (SSP). This means that the set packing program (SSP) is simply an integer programming formulation of the *stable set problem* (SSP) on the associated conflict graph $G(A)$ with node weights w_j. For this reason, we will occasionally also denote $P_I(A)$ by $P_I(G(A))$.

Two consequences of this equivalence are: (i) Two 0/1 matrices A and A' give rise to the same set packing problem if and only if their intersection graphs coincide. (ii) Every row of A is the incidence vector of a *clique* in $G(A)$, i.e., a set of pairwise adjacent nodes. In particular, $G(A) = G(A')$ if A' is the clique-node incidence matrix of *all* cliques in $G(A)$, or of a set of cliques such that each edge is contained in some clique, or of all *maximum* cliques with respect to set inclusion, see Padberg [1973a]. Note that the last matrix contains a maximum of clique information without any redundancies.

Set covering is known to be equivalent to optimization over *independence systems*, see, e.g., Laurent [1989] or Nobili & Sassano [1989], by the affine transformation $y := \mathbf{1} - x$:

$$\begin{array}{llll} \min\ w^{\mathrm{T}}(\mathbf{1}-y) & = w^{\mathrm{T}}\mathbf{1} - & \text{(ISP)} & \max\ w^{\mathrm{T}}y \\ \quad A(\mathbf{1}-y) \geq \mathbf{1} & & \text{(i)} & \quad Ay \leq (A-I)\mathbf{1} \\ \quad (\mathbf{1}-y) \leq \mathbf{1} & & \text{(ii)} & \quad y \geq 0 \\ \quad (\mathbf{1}-y) \geq 0 & & \text{(iii)} & \quad y \leq \mathbf{1} \\ \quad (\mathbf{1}-y) \in \{0,1\}^n & & \text{(iv)} & \quad y \in \{0,1\}^n. \end{array}$$

To see that the program on the right is an optimization problem over an independence system, we have to construct a suitable independence system. To do this, note first that one can delete from (ISP) any row that strictly contains some other row. A 0/1 matrix without such redundant rows is called *proper* (Fulkerson [1971]). Assuming w.l.o.g. that A is proper, we can take its rows as the incidence vectors of the *circuits* of an *independence system* $\mathfrak{I}(A)$ on the groundset of column(indice)s of A. Then the right-hand side $(A-I)_{i\cdot}\mathbf{1} = |\operatorname{supp} A_{i\cdot}| - 1$ of every constraint i in (ISP) equals the *rank* of the circuit $\operatorname{supp} A_{i\cdot}$ and (ISP) is an integer programming formulation of the problem to find an independent set of maximum weight with respect to w in $\mathfrak{I}(A)$.
We remark that there is also a graph theoretic formulation of the set covering problem in terms of a bipartite row-column incidence graph that has been proposed, e.g., by Sassano [1989] and Cornuéjols & Sassano [1989].
Thinking again about the *relation* of set packing and set covering in terms of stable sets and independence systems, one makes the following observations. (i) The stable sets in a graph form an independence system, i.e., set packing is a special case of set covering with additional structure. (ii) This argument holds for almost any other combinatorial optimization problem as well; we mention here in particular the *generalized set packing problem* and the *generalized set covering problem*, that arise from their standard relatives by allowing for an arbitrary uniform right-hand side, see Sekiguchi [1983]. (iii) Not every independence system can be obtained from stable sets of some appropriately constructed graph, see Nemhauser & Trotter [1973, Theorem 4.1] or Padberg [1973b, Remark 3.15] for details.

1.4 Blocking and Anti-Blocking Pairs

The theory of *blocking* and *anti-blocking* pairs of matrices and polyhedra, developed in Fulkerson [1970, 1971, 1972], provides a framework for the study of packing and covering problems that explains why packing and covering theorems occur in dual pairs. Its technical vehicle is the *duality* (or *polarity*, who likes the term better) between constraints and vertices/extreme rays of polyhedra. We discuss the basics of the theory here in a general setting for nonnegative matrices and specialize to the combinatorial 0/1 case in the following Sections 1.5 and 1.6.
The center of the theory is the notion of a blocking and anti-blocking pair of matrices and polyhedra that we introduce now. Consider a *nonnegative* (not necessarily 0/1) matrix A and the associated *fractional packing problem* (FPP) and the *fractional covering problem* (FCP)

$$\begin{array}{lll} \text{(FPP)} & \max\ w^{\mathrm{T}}x & \\ & \quad Ax \leq \mathbf{1} & \\ & \quad x \geq 0 & \end{array} \qquad \begin{array}{lll} \text{(FCP)} & \min\ w^{\mathrm{T}}x & \\ & \quad Ax \geq \mathbf{1} & \\ & \quad x \geq 0. & \end{array}$$

Associated to these problems are the *fractional packing polytope* and the *fractional covering polyhedron*, that we denote, slightly extending our notation, by $P(A)$ and $Q(A)$, respectively.

By Weyl's description theorem, see, e.g., Schrijver [1986, Corollary 7.1b], these bodies are generated by their vertices and extreme rays. Denote by $\operatorname{abl} A$ the matrix that has the vertices of $P(A)$ as its *rows*, and by $\operatorname{bl} A$ the matrix that has the vertices of $Q(A)$ as its *rows*. Then we have

$$P(A) = \{x \in \mathbb{R}^n_+ : Ax \leq \mathbf{1}\} = \operatorname{conv} \operatorname{vert} P(A) \qquad = \operatorname{conv} (\operatorname{abl} A)^{\mathrm{T}}$$
$$Q(A) = \{x \in \mathbb{R}^n_+ : Ax \geq \mathbf{1}\} = \operatorname{conv} \operatorname{vert} Q(A) + \mathbb{R}^n_+ = \operatorname{conv} (\operatorname{bl} A)^{\mathrm{T}} + \mathbb{R}^n_+.$$

(We must assume that A does not contain empty columns for the packing equations to hold.) $\operatorname{abl} A$ is called the *anti-blocker* of the matrix A, $\operatorname{bl} A$ is the *blocker* of A. Associated to these matrices are again a fractional packing polytope and another fractional covering polyhedron.

$$\operatorname{abl} P(A) := \{y \in \mathbb{R}^n_+ : x^{\mathrm{T}} y \leq 1 \ \forall x \in P(A)\} = \{y \in \mathbb{R}^n_+ : \operatorname{abl} A y \leq \mathbf{1}\} = P(\operatorname{abl} A)$$
$$\operatorname{bl} Q(A) := \{y \in \mathbb{R}^n_+ : y^{\mathrm{T}} x \geq 1 \ \forall x \in Q(A)\} = \{y \in \mathbb{R}^n_+ : \ \operatorname{bl} A y \geq \mathbf{1}\} = Q(\operatorname{bl} A).$$

$\operatorname{abl} P(A)$ is called the *anti-blocker* of the polytope $P(A)$, $\operatorname{bl} Q(A)$ is the *blocker* of $Q(A)$. The general duality between constraints and vertices/extreme rays of polyhedra translates here into a duality relation between anti-blocking and blocking matrices and polyhedra.

1.4.1 Theorem (Blocking and Anti-Blocking Pairs, Fulkerson [1971])
For any nonnegative matrix A holds:

(i) *If a is a vertex of $\operatorname{abl} P(A)$, $a^{\mathrm{T}} x \leq 1$ is either a facet of $P(A)$, or can be obtained from a facet by setting some left-hand side coefficients to zero. In particular,*

(ii) $\operatorname{abl}^2 P(A) = P(A)$.

(iii) *If A has no empty column, so does* $\operatorname{abl} A$.

(iv) *If a is a vertex of $\operatorname{bl} Q(A)$, $a^{\mathrm{T}} x \geq 1$ is a facet of $Q(A)$. In particular,*

(v) $\operatorname{bl}^2 Q(A) = Q(A)$.

(vi) $\operatorname{bl} A$ *is proper and*
$\operatorname{bl}^2 A = A \iff A$ *is proper.*

Here, abl^2 is short for abl abl, and so on. Theorem 1.4.1 (ii) and (v) state that the anti-blocking relation gives indeed rise to a dual *anti-blocking pair of polyhedra* and the blocking relation to a dual *blocking pair of polyhedra*. This duality carries over to the associated matrices. Theorem 1.4.1 (iv) and (vi) establishes a *blocking pair of proper matrices*. The duality is a bit distorted in the anti-blocking case, because the anti-blocking relation produces dominated vertices/rows. Since only the maximal rows give rise to facets, one does not insist on including dominated rows in a packing matrix, and calls two matrices A and B an *anti-blocking pair of matrices*, if the associated packing polyhedra constitute an anti-blocking pair. Blocking and anti-blocking pairs of matrices (and polyhedra) are *characterized* by a set of four relations that provide a link to optimization. Let A and B be two nonnegative matrices and consider the equalities

$$\begin{aligned} \min \ & y^{\mathrm{T}} \mathbf{1} = \max \ Bw \\ & y^{\mathrm{T}} A \geq w^{\mathrm{T}} \\ & y^{\mathrm{T}} \ \geq 0^{\mathrm{T}} \end{aligned} \tag{1.3}$$

$$\begin{aligned} \max \ & y^{\mathrm{T}} \mathbf{1} = \min \ Bw. \\ & y^{\mathrm{T}} A \leq w^{\mathrm{T}} \\ & y^{\mathrm{T}} \ \geq 0^{\mathrm{T}} \end{aligned} \tag{1.4}$$

Here, $\min \ Bw$ is short for $\min \ \{B_{i\cdot} w : i = 1, \ldots, m\}$, and so on. If (1.3) holds *for all nonnegative vectors w*, we say that the *min-max equality* holds for the *ordered* pair of matrices A, B. If (1.4) holds *for all nonnegative vectors w*, we say that the *max-min equality* holds for the *ordered* pair of matrices A, B.

The other two relations are inequalities:

$$\max\ Al \cdot \max\ Bw \geq l^{\mathrm{T}}w \qquad (1.5) \qquad\qquad \min\ Al \cdot \min\ Bw \leq l^{\mathrm{T}}w. \qquad (1.6)$$

If (1.5) holds *for all nonnegative vectors w and l*, we say that the *max-max inequality* holds for the (unordered) pair of matrices A and B. If (1.6) holds *for all nonnegative vectors w and l*, we say that the *min-min inequality* holds for the (unordered) pair of matrices A and B. These equations and inequalities are related to the anti-blocking and the blocking relation via appropriate scalings of the vectors w and l such that the above optima become one; this is always possible except in the trivial cases $w = 0$ and/or $l = 0$. Such a scaling makes w and l a member of the anti-blocking/blocking polyhedron. These arguments can be used to prove

1.4.2 Theorem (Characterization of Blocking and Anti-Blocking Pairs, Fulkerson [1971])

For any pair of nonnegative matrices A and B with no empty columns, the following statements are equivalent:

(i) A and B are an anti-blocking pair.

(ii) $P(A)$ and $P(B)$ are an anti-blocking pair.

(iii) The min-max equality holds for A, B.

(iv) The min-max equality holds for B, A.

(v) The max-max inequality holds for A and B.

For any pair of proper nonnegative matrices A and B, the following statements are equivalent:

(vi) A and B are a blocking pair.

(vii) $Q(A)$ and $Q(B)$ are a blocking pair.

(viii) The max-min equality holds for A, B.

(ix) The max-min equality holds for B, A.

(x) The min-min inequality holds for A and B.

Theorem 1.4.2 bears on *dual min-max results* for packing and covering *optimization problems*. We give an interpretation of the anti-blocking part (iii) and (iv) of Theorem 1.4.2 in terms of the fractional packing problem, the covering case is analogous. The min-max equality (1.3) can be interpreted as a "weighted max fractional packing-min fractional covering theorem": The rows of A are used for covering, the rows of B, that correspond to the feasible solutions of (FPP), for packing. If this min-max theorem can be established, anti-blocking theory yields a second, equivalent theorem of the same type, where the covering-packing roles of A and B are exchanged.

1.5 Perfect and Ideal Matrices

The main point of interest in anti-blocking and blocking theory is the study of anti-blocking and blocking pairs of matrices A and B that are *both 0/1*. Saying that a 0/1 matrix A has a 0/1 anti-blocking matrix B is *by definition* equivalent to integrality of the fractional packing polytope associated to A; a 0/1 matrix A that gives rise to such an integral packing polytope $P(A) = P_I(A)$ is called *perfect*. Analogous for covering: 0/1 blocking matrices correspond to integral covering polyhedra $Q(A) = Q_I(A)$; a 0/1 matrix A with this property is called *ideal*. By Theorem 1.4.2, *perfect matrices occur in anti-blocking pairs* and so do *ideal matrices occur in blocking pairs*. Associated to an anti-blocking/blocking pair of perfect/ideal matrices is a pair of equivalent min-max/max-min equalities and one can either prove one of the equalities to establish the second plus the anti-blocking/blocking property plus perfection/ideality of a 0/1 matrix pair, or one can prove one of the latter two properties to obtain two min-max/max-min results.

Anti-blocking/blocking pairs of perfect/ideal matrices often have combinatorial significance and this brings up the existence question for *combinatorial covering and packing theorems.* The min-max/max-min equalities (1.3) and (1.4) are not of combinatorial type, because they allow for *fractional* solutions of the covering/packing program. But consider stronger, *integer* forms of these relations for 0/1 matrices A and B:

$$\begin{array}{ll} \min & y^{\mathrm{T}}\mathbf{1} \\ & y^{\mathrm{T}}A \geq w^{\mathrm{T}} \\ & y^{\mathrm{T}} \geq 0^{\mathrm{T}} \\ & y^{\mathrm{T}} \in \mathbb{Z}^m \end{array} = \max\ Bw \qquad (1.7) \qquad\qquad \begin{array}{ll} \max & y^{\mathrm{T}}\mathbf{1} \\ & y^{\mathrm{T}}A \leq w^{\mathrm{T}} \\ & y^{\mathrm{T}} \geq 0^{\mathrm{T}} \\ & y^{\mathrm{T}} \in \mathbb{Z}^m \end{array} = \min\ Bw. \qquad (1.8)$$

If (1.7) holds *for all nonnegative integer vectors* w, we say that the *strong min-max equality* holds for the *ordered* pair of 0/1 matrices A and B; this is equivalent to stating that the packing system $Ax \leq \mathbf{1}, x \geq 0$ is TDI. If (1.8) holds *for all nonnegative integer vectors* w, we say that the *strong max-min equality* holds for the *ordered* pair of 0/1 matrices A and B; this relation corresponds to a TDI covering system $Ax \geq \mathbf{1}, x \geq 0$. The *combinatorial content* of these relations is the following. The strong min-max equality can be interpreted as a combinatorial min covering-max packing theorem for an anti-blocking pair of perfect matrices: The smallest number of rows of A such that each column j is covered by at least w_j rows is equal to the largest packing of columns with respect to w, where the packings are encoded in the rows of B. An analogous statement holds in the strong max-min case for a blocking pair of ideal matrices.
We mention two famous examples of such relations to point out the significance of this concept. *Dilworth's theorem* is an example of a well-known strong min-max equality in the context of *partially ordered sets.* Let A be the incidence matrix of all *chains* of some given poset, let B be the incidence matrix of all its *antichains*, and consider the strong min-max equality for A, B: It states that, for any nonnegative integer vector w of weights associated to the elements of the poset, the smallest number of chains such that each element is contained in at least w_j chains is equal to the maximum w-weight of an antichain. For $w = \mathbf{1}$, this is the classical Dilworth theorem, and one can generalize it to the weighted case by appropriate "replications" of poset elements (the reader may verify that this is *easy*). The validity of this weighted generalization of Dilworth's theorem implies that A and B form an anti-blocking pair of perfect matrices, because the strong min-max equality for A, B yields, trivially, the fractional min-max equality for A, B. This argument implies in turn the min-max equality for B, A in its fractional form. What about the strong, integer version for B, A? One can work out that it holds as well — and this is not a strike of luck! But let's stop here for the moment and just consider the combinatorial content of the strong min-max equality for B, A: This theorem is identical to the weighted Dilworth theorem, except that the words "antichain" and "chain" have changed their places — a combinatorial companion theorem.
The most famous example of a strong max-min equality is probably the *max flow-min cut theorem* of Ford, Jr. & Fulkerson [1962] for two-terminal networks. Taking A as the incidence matrix of all (s,t)-paths versus edges and B as the incidence matrix of all (s,t)-cuts versus edges, the max flow-min cut theorem turns out to be exactly the strong max-min equality for A, B. Hence, the incidence matrices of (s,t)-paths and (s,t)-cuts in a two-terminal network form a blocking pair of ideal matrices. Can one also produce a companion theorem by interchanging the roles of paths and cuts as we did with the antichains and chains in Dilworth's theorem? The answer is yes and no: One can in this particular case, but not in general.

We have already hinted at one of the main insights of anti-blocking theory in the Dilworth example and we state this result now: The perfection of a matrix A is equivalent to the validity of the strong min-max equality for A and $\operatorname{abl} A$ which is itself equivalent to the validity of a *companion min-max theorem* for $\operatorname{abl} A$ and A.

1.5.1 Theorem (Strong Min-Max Equality, Fulkerson [1971])
Let A be a 0/1 matrix without empty columns. The following statements are equivalent:

(i) *A is perfect.*
(ii) *$\operatorname{abl} A$ is perfect.*
(iii) *The system $Ax \le \mathbf{1}, x \ge 0$ is integral.*
(iv) *The system $Ax \le \mathbf{1}, x \ge 0$ is TDI.*
(v) *The strong min-max equality holds for $A, \operatorname{abl} A$.*
(vi) *The strong min-max equality holds for $\operatorname{abl} A, A$.*

Interpreting this result in terms of the stable set problem, see Section 1.3, we enter the realm of *perfect graph theory*. A minute's thought shows that the only candidate for a 0/1 anti-blocker of the incidence matrix B of all stable sets of some given graph G is the incidence matrix A of all cliques versus nodes. Now consider the two possible strong min-max equations; the optima of the four associated optimization problems are commonly denoted by

$$\begin{array}{llll@{\qquad}llll}
\overline{\chi}_w(G) & := \min & y^{\mathrm{T}}\mathbf{1} & & \chi_w(G) & := \min & y^{\mathrm{T}}\mathbf{1} & \\
 & & y^{\mathrm{T}}A \ge w^{\mathrm{T}} & & & & y^{\mathrm{T}}B \ge w^{\mathrm{T}} & \\
 & & y^{\mathrm{T}} \ \ \ge 0^{\mathrm{T}} & & & & y^{\mathrm{T}} \ \ \ge 0^{\mathrm{T}} & \\
 & & y^{\mathrm{T}} \ \ \text{integral} & & & & y^{\mathrm{T}} \ \ \text{integral} & \\
\alpha_w(G) & := \max & Bw & & \omega_w(G) & := \max & Aw. &
\end{array}$$

$\overline{\chi}_w(G)$ is called the weighted *clique covering number* of G, $\alpha_w(G)$ is the weighted *stability number*, $\chi_w(G)$ the weighted *coloring number*, and $\omega_w(G)$ the weighted *clique number*. With this terminology, the strong min-max equality for A, B translates into the validity of the equation $\overline{\chi}_w(G) = \alpha_w(G)$ for any nonnegative integer vector w, and a graph with this property is called *$\overline{\chi}$-pluperfect*. Similarly, a *χ-pluperfect* graph satisfies the second strong min-max equality $\chi_w(G) = \omega_w(G)$ forall $w \in \mathbb{Z}^n_+$, and a *pluperfect* graph is both $\overline{\chi}$- and χ-pluperfect. Theorem 1.5.1 reads in this language as follows.

1.5.2 Theorem (Pluperfect Graph Theorem, Fulkerson [1971])
A graph is $\overline{\chi}$-pluperfect if and only if it is χ-pluperfect if and only if it is pluperfect.

This theorem can also be stated in terms of *complement graphs* by noting that $\overline{\chi}$-pluperfection of a graph G is equivalent to χ-pluperfection of the complement graph $\overline{G}$. This equivalent version is: A graph is $\overline{\chi}$-pluperfect if and only if its complement is.

One of the big questions in this context and the original motivation for the development of the entire anti-blocking theory was the validity of Berge [1961]'s famous *perfect graph conjecture*. The conjecture claimed a stronger form of the pluperfect graph theorem where w is not required to run through all nonnegative integer vectors w, but only through all *0/1* vectors. In exactly the same way as in the pluperfect case, this concept gives rise to *$\overline{\chi}$-perfect*, *χ-perfect*, and *perfect* graphs, hence the conjecture's name. Fulkerson's idea to prove it was to show its equivalence to the pluperfect graph theorem; to establish this it is enough to prove the following *replication lemma*: Duplicating a vertex of a perfect graph and joining the obtained two vertices by an edge gives again a perfect graph. The replication lemma and hence the conjecture was proved by Lovász [1971] and, shortly after the result had become known, also by Fulkerson [1973].

1.5.3 Theorem (Perfect Graph Theorem, Lovász [1971])
A graph is $\overline{\chi}$-perfect if and only if it is χ-perfect if and only if it is perfect if and only if it is pluperfect.

There is also a complement version of the perfect graph theorem: A graph is $\overline{\chi}$-perfect if and only if its complement is. And let us further explicitly state an integer programming form of the perfect graph theorem, that will turn out to have a blocking analogon. We include a strong version of the max-max inequality with *identical* 0/1 vectors w and l, also proved by Lovász [1972] and Fulkerson [1973].

1.5.4 Theorem (Perfect Graph Theorem, Lovász [1971], Fulkerson [1973])
For 0/1 matrices A and B without empty columns, the following statements are equivalent.

(i) A and B are an anti-blocking pair.
(ii) The strong min-max equality holds for A, B and all nonnegative integer vectors w.
(iii) The strong min-max equality holds for B, A and all nonnegative integer vectors w.
(iv) The strong min-max equality holds for A, B and all 0/1 vectors w.
(v) The strong min-max equality holds for B, A and all 0/1 vectors w.
(vi) The max-max inequality holds for A and B and all nonnegative integer vectors w and l.
(vii) The max-max inequality holds for A and B and all 0/1 vectors $w = l$.

Here, we have used the expression "the strong min-max equality holds" in an obvious sense, slightly extending our terminology. A third interesting linear programming form of the perfect graph theorem is again due to Lovász [1971].

1.5.5 Theorem (Perfect Graph Theorem, Lovász [1971])
A 0/1 matrix A without empty columns is perfect if and only if the linear program max $w^{\mathrm{T}}x$, *$Ax \leq \mathbf{1}, x \geq 0$ has an integer optimum value for all 0/1 vectors w.*

Let's take a break from anti-blocking and perfect graphs at this point and turn to the *blocking* case. Unfortunately, the anti-blocking results of this section do not all carry over: It is *not* true and the main difference between blocking and anti-blocking theory that the integrality of the fractional covering polyhedron corresponds to a TDI constraint system, neither is it true that the strong max-min inequality for A, B implies the strong max-min equality for B, A, see Fulkerson [1971] for a counterexample. And there are also no results that compare to perfect graph theory, because there is no suitable graph version of the set covering problem.
The other Theorems 1.5.4 and 1.5.5 have analogues that are due to Lehman [1979, 1981]; proofs of these difficult results are given in Padberg [1993] (from a polyhedral point of view) and Seymour [1990] (from a hypergraph point of view). We state them in the following two theorems, where we adopt the conventions that $0 \cdot \infty = 0$ (Theorem 1.5.6 (iii)) and that ∞ is an integer (Theorem 1.5.7).

1.5.6 Theorem (Width-Length Property of Ideal Matrices, Lehman [1979, 1981]
For 0/1 matrices A and B, the following statements are equivalent:

(i) A and B are a blocking pair.
(ii) The min-min inequality holds for all nonnegative integer vectors w and l.
(iii) The min-min inequality holds for all vectors w and l restricted to coefficients 0, 1, ∞, and at most one occurrence of another coefficient that is equal to the number of 1-coefficients minus one. (The fourth type of coefficients is solely needed to exclude the incidence matrices of "degenerate projective planes", see the following Section 1.6.)

1.5.7 Theorem (Max Flow-Min Cut Property of Ideal Matrices, Lehman [1981])
A 0/1 matrix is ideal if and only if the linear program min $w^{\mathrm{T}}x, Ax \geq \mathbf{1}, x \geq 0$ *has an integer optimum value for all 0/1/∞ vectors w.*

The names for these results come from Lehman's terminology; his *width-length inequality* is the same as the min-min inequality, the *max flow-min cut equality* is the max-min equality. Generalizing the concepts of perfection and ideality to $0/\pm 1$ *matrices*, we enter an area of research that is related to the study of *totally unimodular matrices*. It is beyond the scope of this chapter to discuss these fields or integral/TDI $0/\pm 1$ systems in general; surveys on these topics are given in Padberg [1975a] and Conforti, Cornuéjols, Kapoor & Vušković [1997].

1.6 Minor Characterizations

Both the perfect graph theorem and the max flow-min cut characterization of ideal matrices have alternative interpretations in terms of *matrix minors* and, in the anti-blocking case, also of *graph minors* that we discuss in this section. The study of minors bears on the recognition problem for perfect and ideal matrices.
We start in the *anti-blocking* setting. Consider the perfect graph theorem in its linear programming form 1.5.5 and note that setting an objective coefficient w_j to zero has the same effect on the optimum objective value as removing column $A_{\cdot j}$ from the matrix A. Equivalently, we could remove node j from the column intersection graph or, yet another equivalent version, we could intersect the fractional packing polytope $P(A)$ with the hyperplane $x_j = 0$ and eliminate coordinate j. The operation that we have just described is called a *contraction* of coordinate (or column) j of the matrix A or of the intersection graph $G(A)$ or of the fractional packing polytope $P(A)$, and the resulting matrix or graph or polytope is a *contraction minor* of the original object. With this terminology, considering all 0/1 objectives is the same as considering objective $\mathbf{1}$ for all contraction minors and one obtains various minor forms of the perfect graph theorem by replacing the expression "for all 0/1 vectors w" with "for all contraction minors and $w = \mathbf{1}$". For example, Theorems 1.5.5 and 1.5.4 translate (in different ways) into the following minor results.

1.6.1 Theorem (Perfect Graph Theorem, Lovász [1971])
A 0/1 matrix A without empty columns is perfect if and only if the linear program max $\mathbf{1}^{\mathrm{T}}x$, $A'x \leq \mathbf{1}, x \geq 0$ *has an integer optimum value for all contraction minors A' of A.*

1.6.2 Theorem (Perfect Graph Theorem, Lovász [1971, 1972])
The following statements are equivalent for a graph G:

(i) G is perfect.

(ii) $\overline{\chi}(G') = \alpha(G')$ for all minors G' of G.

(iii) $\chi(G') = \omega(G')$ for all minors G' of G.

(iv) $\alpha(G')\omega(G') \geq |V(G')|$ for all minors G' of G.

(Here, a minor is always a contraction minor.)

The contraction technique can be used also in the *blocking* scenario to deal with the zero objective coefficients in Theorem 1.5.7. A little more difficult is the treatment of the ∞-coefficients. $w_j = \infty$ amounts to forcing x_j to one; this effect can also be obtained by removing column j from the matrix A as well as all rows that $A_{\cdot j}$ intersects, or by an intersection of the fractional covering polyhedron $Q(A)$ with the hyperplane $x_j = 1$ and a subsequent elimination of coordinate j. This operation is called a *deletion* of coordinate (or column) j of the matrix A or the polyhedron $Q(A)$ and its result is a *deletion minor*. It is straightforward to show that contraction and deletion commute and one can thus call *the* matrix A' the arises by contracting and deleting some set of coordinates of A a (contraction-deletion) *minor* of A. This nomenclature gives again rise to a number of minor theorems for ideal matrices, like

1.6.3 Theorem (Minor Characterization of Ideal Matrices, Lehman [1981])
A 0/1 matrix is ideal if and only if the linear program $\min\ \mathbf{1}^{\mathrm{T}}x, A'x \geq \mathbf{1}, x \geq 0$ *has an integer optimum value for all contraction-deletion minors* A' *of* A.

The minor characterizations for perfect and ideal matrices bear on the *recognition problems* for these classes: Given a 0/1 matrix A, is it perfect/ideal or not? It is not known whether any of the recognition problems is in $\mathcal{NP}$ or not, but Theorems 1.6.1 and 1.6.3 give a first co-$\mathcal{NP}$ answer. Recognizing perfect and ideal matrices is in co-$\mathcal{NP}$, if the input length is assumed to be $O(n \times m)$, i.e., if we consider A the input: Just exhibit a minor such that 1.6.1 or 1.6.3 fail and verify this by solving a linear program! This result is not very deep, however, because one doesn't need the perfect graph theorem or the max flow-min cut characterization to come up with polynomial certificates for the existence of a fractional basic solution of an explicitly given linear system.

Anyway, researchers are not satisfied with results of this type and we explain now why this is so for the perfection test. The problem is that the recognition of *imperfect matrices* does not carry over to the recognition of *imperfect graphs*. The reason is that although we could verify a clique matrix of a graph as imperfect in polynomial time, this does not help much for an effective investigation of some given graph, because a clique matrix has in general already exponential size in the encoding length of the graph. From this point of view, a co-$\mathcal{NP}$ complexity result as above "seems to be cheating; what we really want are algorithms with running time polynomial in the number of vertices [columns of A]" (Seymour [1990]). And nothing else but exactly this is in fact possible! One can devise such algorithms for the verification of imperfection as well as for the verification of nonideality, the latter in a sense that is yet to be made precise.

The methods that resolve these questions are based on the concepts of *minimally imperfect* (or *almost perfect*) and *minimally nonideal* (or *almost ideal*) 0/1 matrices, that are not perfect/ideal themselves, but any of their deletion/contraction-deletion minors is. Obviously, any imperfect/nonideal matrix must contain such a structure and a recognition algorithm can in principle certify perfection by making sure that no such minor exists, imperfection by exhibiting one, and so for the ideality test. One approach to the recognition problem is hence to study the structure of minimally imperfect and nonideal matrices. This structure is still not fully understood, but to a significant extent and there are, in particular, complete characterizations of minimally imperfect and minimally nonideal matrices, and of perfect and ideal matrices in terms of *forbidden minors*. A final terminological remark: As usual, there are also *minimally imperfect* (or *almost perfect*) *graphs*, and the same concepts exist for the fractional packing and covering polyhedra, that are called *almost integral*.

We begin with results on *minimal imperfection*, where the matrix structures of interest have the following appearance. We say that an $m \times n$ 0/1 matrix A has *property* $\pi_{\omega,n}$ if

(i) A contains a regular $n \times n$ matrix A_1 with column and row sums all equal to ω,

(ii) each row of A which is not a row of A_1 is either equal to some row of A_1 or has row sum strictly less than ω.

The matrix A_1, that is obviously unique up to permutations of rows whenever it exists, is called the *core* of A and denoted by $\operatorname{core} A$.

1.6.4 Theorem (Minimally Imperfect Matrices, Padberg [1973b, 1976])
An $m \times n$ 0/1 matrix A is minimally imperfect if and only if

(i) A has property $\pi_{\omega,n}$, where $n \equiv 1 \mod \omega$ and either $\omega = n-1$ or $2 \le \omega \le \lfloor (n-1)/2 \rfloor$.
(ii) A has no $m \times k$ contraction minor A' with property $\pi_{\omega,k}$ for any $k < n$ and any ω such that $2 \le \omega \le k-1$.

An $m \times n$ 0/1 matrix A is perfect if and only if A does not contain any $m \times k$ contraction minor A' having property $\pi_{\omega,k}$ for $3 \le k \le \min\,\{m, n\}$ and either $\omega = k-1$ or $2 \le \omega \le \lfloor (k-1)/2 \rfloor$.

This theorem makes some progress toward the co-$\mathcal{NP}$ complexity part of the recognition problem for perfect graphs, because a core has an encoding length that is polynomial in n and looks like a good candidate to certify property $\pi_{\omega,k}$ for some contraction minor A' of the (only implicitly known) clique matrix A of some given graph G. The only problem that remains is to verify that some 0/1 matrix A_1 is a core of A'. In other words: How does one prove that all cliques in $G[\operatorname{supp} A']$ of size ω are already contained in A_1 and that there are no larger ones? The answer to this question is based on strong structural properties of — dual *pairs of minimally imperfect matrices*, how could it be different!
To start, note that the core of a minimally imperfect matrix A with property $\pi_{\omega,n}$ produces a fractional vertex $\overline{x} = (\operatorname{core} A)^{-1}\mathbf{1} = (1/\omega, \ldots, 1/\omega)$ of the almost integral polytope $P(A)$. Padberg [1976] has shown that this is the *only* fractional vertex. And much more is true:

1.6.5 Theorem (Pairs of Minimally Imperfect Matrices, Padberg [1973b, 1976])
Let A be an $m \times n$ 0/1 matrix and let $B = \operatorname{abl}_I A$ be the integral part of its anti-blocker. Suppose A is minimally imperfect with property $\pi_{\omega,n}$. Then:

(i) B is also minimally imperfect.
(ii) A has property $\pi_{\omega,n}$ and B has property $\pi_{\alpha,n}$ where $\omega\alpha + 1 = n$.
A and B have unique cores that satisfy the matrix equation $\operatorname{core} A (\operatorname{core} B)^{\mathrm{T}} = E - I$.
(iii) $P(A)$ has the unique fractional vertex $\overline{x} = (1/\omega, \ldots, 1/\omega)$.
$\overline{x}$ is adjacent to precisely n vertices of $P(A)$, namely, the rows of $\operatorname{core} B$.
Moreover, $P_I(A) = \{Ax \le \mathbf{1}, x \ge 0, \mathbf{1}^{\mathrm{T}}x \le \alpha\}$.

Here, E is a *matrix of all ones*, I is the *identity matrix*, and the matrix equation in (ii) is supposed to be understood modulo suitable column and row permutations.
Theorem 1.6.5 has interesting consequences. Note that part (iii) states that all that misses to make an almost integral packing polytope integral is one simple rank facet. This situation can come up in two ways. The first case is when A is not a clique matrix of its conflict graph $G(A)$, i.e., some clique row is missing. As A is minimally imperfect, it must have property $\pi_{n-1,n}$, $G(A)$ must be a clique, and the missing row is $\mathbf{1}^{\mathrm{T}}x \le 1$. The second and exciting case is when A is a clique matrix. Then we see from Theorem 1.6.5 the following.

(i) $G = G(A)$ has exactly n maximum cliques of size $\omega = \omega(G)$ and exactly n maximum stable sets of size $\alpha = \alpha(G)$; the incidence vectors of these maximum cliques and stable sets are linearly independent. Each maximum clique intersects all but exactly one maximum stable set, its so-called *partner*, and vice versa.
(ii) For every node j, $G - j$ can be partitioned into α maximum cliques of size ω and ω maximum stable sets of size α, where $\alpha\omega + 1 = n$.

Here, e_j denotes the *unit vector* that has a one in coordinate j, and $G - j$ is the minor that arises from G by contracting node j. (i) is derived from column j of the matrix equation $\operatorname{core} A \operatorname{core} B_{j\cdot}^{\mathrm{T}} = \mathbf{1} - e_j$, (ii) using Theorem 1.6.2 (iv).

A graph that satisfies the strong condition (ii) on the preceding page is called *partitionable*. Note that, for such a graph G, $\alpha = \alpha(G)$ and $\omega = \omega(G)$ must hold, and since $\alpha(G)\omega(G) = \alpha\omega = n - 1 < n$, partitionable graphs are imperfect by virtue of Theorem 1.6.2. But it is easy to verify that a graph or a contraction minor of a graph is partitionable, and this finally proves that perfection of a graph is a property in co-$\mathcal{NP}$. This complexity result was first stated (and proved in a different way) by Grötschel, Lovász & Schrijver [1984].

1.6.6 Theorem (Recognition of Perfect Graphs, Padberg [1973b, 1976], Grötschel, Lovász & Schrijver [1984]) *The recognition problem for perfect graphs is in co-$\mathcal{NP}$.*

But is that all that one can derive from Padberg's strong conditions (i) and (ii)? One can not help thinking that they stop just by a hair short of a much more *explicit* characterization of all minimally imperfect matrices, which is a long standing research objective. In fact, only two infinite, but simple classes of minimally imperfect matrices are known: The *circulants* $C(2k + 1, 2)$, that are the incidence matrices of *odd holes* (that we denote with the same symbol), and their anti-blockers $\operatorname{abl}_I C(2k+1, 2)$, the incidence matrices of the *odd antiholes*, the complements of the odd holes. Is that all? The *strong perfect graph conjecture* of Berge [1961], which is perhaps the most famous open question in graph theory, claims that it is! If so, odd holes and antiholes furnish simple minor certificates of imperfection. But there is more: It does not seem to be completely out of the question to detect the presence or the absence of odd holes and antiholes in polynomial time, although nobody knows for now if this is possible or not. But if the strong perfect graph conjecture holds, and if the recognition problems for odd holes and antiholes can be solved in polynomial time as well, these results together would solve the recognition problem for perfect graphs.

Chvátal [1976] pointed out that the strong perfect graph conjecture holds if one can show that every minimally imperfect graph G contains a spanning circulant $C(\alpha\omega + 1, \omega)$, i.e., the nodes of G can be numbered $0, \ldots, \alpha\omega$ such that any ω successive nodes $i, \ldots, i + \omega - 1$ (indices taken modulo $\alpha\omega + 1$) form a clique; here, we denote $\alpha = \alpha(G)$, $\omega = \omega(G)$. When Padberg's conditions became known, there was some hope that they would be strong enough to establish this circulant structure in every minimally imperfect graph. But Bland, Huang & Trotter [1979] showed that one can not prove the strong perfect graph conjecture in this way, because Padberg's condition (i) follows from (ii), and the partitionable graphs, that satisfy (ii), do not all contain spanning circulants $C(\alpha\omega + 1, \omega)$.

We turn now to the *minimally nonideal* matrices, where minor characterizations are known that are similar to the packing case, but more complicated. We start with the analogon of the imperfection property $\pi_{\omega,n}$. We say an $m \times n$ 0/1 matrix A has *property* $\varphi_{\alpha,n}$ if

(i) A contains a regular $n \times n$ matrix A_1 with column and row sums all equal to α,

(ii) each row of A which is not a row of A_1 is either equal to some row of A_1 or has row sum strictly larger than α.

The matrix A_1 is again unique up to permutations of rows whenever it exists, and it is also called the *core* of A and denoted by $\operatorname{core} A$.

Unlike in the packing case there is, however, an infinite class of minimally nonideal matrices that do not have constant row and column sums. These incidence matrices of *degenerate projective planes* (points versus lines) read

$$J_n = \begin{pmatrix} 0 & \mathbf{1}^{\mathrm{T}} \\ \mathbf{1} & I_{n-1} \end{pmatrix},$$

where I_{n-1} denotes the $(n - 1) \times (n - 1)$ identity matrix.

1.6.7 Theorem (Minimally Nonideal Matrices, Lehman [1981], Padberg [1993])
If a proper $m \times n$ 0/1 matrix is minimally nonideal then either $A = J_n$ or

(i) A has property $\varphi_{\alpha,n}$, where $n \not\equiv 0 \mod \alpha$.

(ii) A has no $m \times k$ contraction-deletion minor A' with property $\varphi_{\alpha,k}$ for any $k < n$ and any α such that $k \not\equiv 0 \mod \alpha$.

An $m \times n$ 0/1 matrix A is ideal if and only if A does not contain any $m \times k$ contraction-deletion minor A' having property $\varphi_{\alpha,k}$ for $3 \le k \le \min\{m,n\}$.

The requirement that A is proper can also be removed, but then we must change (i) from "$A = J_n$" into "A contains J_n and some additional redundant rows". Note also that we have not claimed an equivalence as for property $\pi_{\omega,n}$.
As the minimally imperfect matrices occur in anti-blocking pairs, so do their minimally nonideal relatives.

1.6.8 Theorem (Pairs of Minimally Nonideal Matrices, Lehman [1981], see also Padberg [1993] and Seymour [1990])
Let A be a proper $m \times n$ 0/1 matrix and let $B = \mathrm{bl}_I A$ be the integral part of its blocker. Suppose A is minimally nonideal. Then:

(i) B is also minimally nonideal.

(ii) Either

(a) $A = B = J_n$.

(b) $Q(A)$ has the unique fractional vertex $\overline{x} = ((n-2)/(n-1), 1/(n-1), \ldots, 1/(n-1))$.
$\overline{x}$ is adjacent to precisely n vertices of $Q(A)$, namely, the rows of B.
Moreover, $Q_I(A) = \{Ax \ge \mathbf{1}, x \ge 0, (n-2)x_1 + \sum_{j=2}^{n} x_j \ge n-1\}$.

or

(c) A has property $\varphi_{\alpha,n}$ and B has property $\varphi_{\beta,n}$ where $\alpha\beta = n + r$, $0 < r < \min\{\alpha, \beta\}$.
A and B have unique cores that satisfy the matrix equation $\mathrm{core}\, A (\mathrm{core}\, B)^{\mathrm{T}} = E + rI$.

(d) $Q(A)$ has the unique fractional vertex $\overline{x} = (1/\alpha, \ldots, 1/\alpha)$.
$\overline{x}$ is adjacent to precisely n vertices of $Q(A)$, namely, the rows of $\mathrm{core}\, B$.
Moreover, $Q_I(A) = \{Ax \le \mathbf{1}, x \ge 0, \mathbf{1}^{\mathrm{T}} x \ge \beta\}$.

The assumption that A is proper can again be removed as in Theorem 1.6.7. Compare also the coefficients in the left-hand side of the additional facet in Theorem 1.6.8 (ii) (b) to the objective coefficients in Theorem 1.5.6 (iii) to see that the fourth type of objective coefficients (the $n-2$) was only needed to deal with the degenerate projective planes J_n.
Seymour [1990] used Lehman's minor characterization 1.6.8 (for which he also gives a proof) to establish that ideality is a co-$\mathcal{NP}$ property in a sense that can be seen as the analogon of Theorem 1.6.6 on the recognition of perfection. Seymour views the $m \times n$ 0/1 matrix A of interest as the incidence matrix of a hypergraph that "should" have an encoding length that is polynomial in the number n of elements. This creates the problem that the encoding length of an $m \times n$ 0/1 matrix A is in general certainly not polynomial in n. Seymour assumes thus that A is given in the form of a *filter oracle*, that decides in constant time whether a given 0/1 vector contains a row of A or not. Calling this oracle a number of times that is polynomial in n, one can certify the existence of blocking matrices $\mathrm{core}\, A$ and $\mathrm{core}\, B$ with properties as in Lehman's Theorem 1.6.8 that ensure that A is nonideal.

1.6.9 Theorem (Recognition of Ideal Matrices, Seymour [1990])
The recognition problem for ideal matrices that are given by a filter oracle is in co-$\mathcal{NP}$.

There are some further results toward a more *explicit* characterization of minimally nonideal matrices. Lehman [1979] gave three *infinite* families of minimally nonideal matrices: The incidence matrices of the degenerate projective planes J_n (which are self-dual in the sense they coincide with their blockers, i.e., $J_n = \mathrm{bl}_I\, J_n$), the odd circulants $C(2k+1,2)$, and their blockers $\mathrm{bl}_I\, C(2k+1,2)$ that coincide via permutation of rows and columns with the circulants $C(2k+1,k+1)$. But, different to the packing case, many more minimally nonideal matrices are known.
First, researchers have compiled a substantial, but *finite* list of "exception" matrices, that do not belong to the three infinite classes of Lehman. The incidence matrix of the *Fano plane*

$$F_7 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix},$$

is one such exception matrix, see Cornuéjols & Novik [1989] and Lütolf & Margot [1998] for comprehensive lists. But the situation is more complicated than this, because further *infinite* classes of minimally nonideal matrices have been constructed. For example, Cornuéjols & Novik [1989] prove (and give a reference to a similar result) that one can add a row $e_i+e_j+e_k$, $i<j<k$, to any odd circulant $C(2k+1,2)$, $2k+1 \geq 9$, where $j-i$ and $k-j$ are both odd, and doing so one obtains a minimally nonideal matrix.
Does all of this mean that the set of minimally nonideal matrices is just a chaotic tohuwabohu? Cornuéjols & Novik [1989] say no and argue that all minimally nonideal matrices in the known infinite non-Lehman classes have core $C(2k+1,2)$ or $C(2k+1,k+1)$. This means geometrically that the associated fractional covering polyhedra arise from $Q(C(2k+1,2))$ or $Q(C(2k+1,k+1))$ by adjoining additional *integral* vertices. Or, to put it differently, the crucial part of a minimally nonideal matrix is its core and there, if one forgets about the exception list, only the three Lehman classes have been encountered. These findings motivate the following conjecture, that can be seen as the covering analogon of the strong perfect graph conjecture.

1.6.10 Conjecture (Ideal Matrix Conjecture, Cornuéjols & Novik [1989])
There is some natural number n_0 such that every $m \times n$ minimally nonideal matrix A with $n \geq n_0$ has core either $C(2k+1,2)$ or $C(2k+1,k+1)$.

1.7 Balanced Matrices

Perfect and ideal matrices were defined in terms of integral polyhedra; their characterization through forbidden minors was and still is a major research problem. The study of *balanced matrices*, that were invented by Berge [1971], goes the other way round: This class is *defined* in terms of forbidden minors and one investigates the combinatorial and polyhedral consequences of this construction. It turns out that these properties subsume all characteristics of perfect and ideal matrices, and balanced matrices give, in particular, rise to a multitude of combinatorial packing and covering problems. But not only do results from perfect and ideal matrix theory carry over. There are additional genuine consequences of balancedness that include TDIness of both associated covering systems and bear on combinatorial *partitioning* theorems. And there is another recent spectacular result that does have no parallel (yet!): Balanced matrices can be recognized in polynomial time!

It is not the aim of this section to give an overview on the entire field of balanced matrices, to say nothing of their $0/\pm 1$ generalizations and/or the connections to *totally unimodular matrices*. Such surveys can be found in Padberg [1975a], Conforti et al. [1994], and Conforti, Cornuéjols, Kapoor & Vušković [1997], we summarize here just some basic results.

A matrix A is *balanced* if it does not contain an odd square *minor* with row and column sums all equal to two or, equivalently, a row and column permutation of the circulant $C(2k+1, 2)$. In the context of balancedness, it is understood that *minors* are not restricted to contraction and deletion minors; instead, any subset I of rows and J of columns of A induces a minor $A' = A_{IJ}$. As an immediate consequence, every such minor A' of a balanced matrix A must also be balanced, so is the transpose A^{T}, and so is also any matrix that arises from A by replicating one or several columns any number of times. Note that the excluded odd hole minors $C(2k+1, 2)$ are "one half" of the known structures that cause imperfection; and note also that a balanced matrix does not only contain no odd hole *contraction* minor, but no odd hole at all, i.e., no odd hole as *any* minor. The possible existence of such different, but similar forbidden minor characterizations for balanced and perfect matrices allows to view the study of balancedness as a precursor to a possible future branch of perfect matrix and graph theory after a successful resolution of the strong perfect graph conjecture.

Back to the present (and actually 25 years to the past), it is easy to see that the edge-node incidence matrices of bipartite or, equivalently, 2-colorable graphs are balanced, and balancedness is in fact a generalization of the concept of *2-colorability* to *hypergraphs*. The connection between 0/1 matrices and colorings of hypergraphs arises from an interpretation of the first as incidence matrices of the latter that goes as follows. We associate to a 0/1 matrix A the hypergraph $\mathcal{H} = \mathcal{H}(A)$, that has the *rows* of A as its nodes and the *columns* as edges[10]; $\mathcal{H}$ is called *balanced* if and only if A is. Hypergraphs can be colored just like graphs: A *node coloring* of $\mathcal{H}$ assigns a color to each node such that no edge contains only nodes of a single color, the *chromatic number* $\chi(\mathcal{H})$ is the minimum number of colors in such a node coloring, and $\mathcal{H}$ is *2-colorable* if $\chi(\mathcal{H}) \leq 2$. It is not so obvious that 2-colorability leads again back to balancedness, but exactly this was Berge [1971]'s idea and his motivation to introduce the whole concept.

1.7.1 Theorem (Balancedness and 2-Colorability, Berge [1971])
A 0/1 matrix A is balanced if and only if $\mathcal{H}(A')$ is 2-colorable for all minors A' of A.

Many combinatorial properties of bipartite graphs carry over to their balanced hypergraph relatives. These similarities arise from (or are reflected in, who likes this better) analogous symmetries between the *totally unimodular* and balanced incidence matrices of bipartite and balanced hypergraphs, that are stressed in the "minor presentation" of the following theorem.

1.7.2 Theorem (Balanced Matrices, Berge [1971], Fulkerson, Hoffman & Oppenheim [1974]) *For a 0/1 matrix A, the following statements are equivalent:*

(i) A is balanced.

(ii) A' is perfect for all minors A' of A.

(iii) A' is ideal for all minors A' of A.

(iv) $P(A')$ is integral for all minors A' of A.

(v) $Q(A')$ is integral for all minors A' of A.

(vi) $P^{=}(A')$ is integral for all minors A' of A.

[10] This is just custom (cf. the König examples of Section 1.1), the *transposed* way would be feasible as well.

We do not delve further into the relations between total unimodularity and balancedness here and consider instead the amazing connections to perfection and ideality: The balanced matrices are exactly those that have only perfect or ideal minors. This has two consequences. First, balanced matrices inherit the properties of their perfect and ideal superclasses *for every minor*, which includes in particular all combinatorial min-max and max-min results. Second, Theorem 1.7.2 can be extended by many other equivalent characterizations of balanced matrices in combinatorial, polyhedral, and in integer programming terminology just like in the theory of perfect and ideal matrices. We give three *examples* to illustrate these points.

The first example is another combinatorial min-max characterization of balancedness that we derive with perfect matrix techniques. Consider the strong min-max equality for $A', \operatorname{abl} A'$ and objective $w = \mathbf{1}$, where A' is any minor of A. Interpreting this relation in terms of the hypergraph $\mathcal{H}(A)$ is to say that for any "partial subhypergraph" $\mathcal{H}(A')$ of $\mathcal{H}(A)$ the maximum size of a matching (edge packing) is equal to the minimum size of a transversal; the equivalence of this relation with balancedness is Berge [1971]'s Theorem 4.

Example two is an alternative integer programming characterization of balanced matrices, that we obtain from transformations of Theorem 1.7.2 (ii). Namely, this statement is equivalent to saying that the integer program max $\mathbf{1}^{\mathrm{T}}x, A'x \le \mathbf{1}, x \ge 0$ has an integer optimum value for any minor A' of A, which holds if and only if the linear program max $b^{\mathrm{T}}x, Ax \le w, x \ge 0$ has an integer optimum value for any $b \in \{0,1\}^n$ and $w \in \{1, +\infty\}^m$. This is true if and only if the dual program min $y^{\mathrm{T}}w, y^{\mathrm{T}}A \ge b, y^{\mathrm{T}} \ge 0$ has an integer optimum value for any $b \in \{0,1\}^n$ and $w \in \{1, +\infty\}^m$ (here, ∞ is not considered to be an integer), and this holds if and only if the program min $y^{\mathrm{T}}\mathbf{1}, y^{\mathrm{T}}A \ge b, 0^{\mathrm{T}} \le y^{\mathrm{T}} \le w^{\mathrm{T}}$ has an integer optimum value for any $b \in \{0,1\}^n$ and $w \in \{0,1\}^m$. The equivalence of this last statement with balancedness is Berge [1971]'s Theorem 6.

As a third and last example, we show that balanced hypergraphs have the *Helly property*: The transpose A^{T} of a balanced matrix A is also balanced, hence A^{T} is perfect, hence it is a clique matrix of a graph; but the cliques of a graph have the Helly property that if any two of a set of cliques have a common vertex, they all have a common vertex, and the same holds for the edges of a balanced hypergraph; this is Berge [1971]'s Proposition 7.

We turn next to two properties of balanced matrices that are "genuine" in the sense that they do not have this inheritance flavour: TDIness of balanced covering and their blocking systems, and a strengthening of this last result to one of the rare and precious *combinatorial partitioning max-min theorems.*

1.7.3 Theorem (TDI Balanced Covering and Blocking Systems, Fulkerson, Hoffman & Oppenheim [1974])
If A is a balanced 0/1 matrix, the strong max-min equality holds for $A, \operatorname{bl} A$ and for $\operatorname{bl} A, A$.

Hence, the balanced matrices satisfy an integrality relation that does not hold in the general blocking case. To avoid misunderstandings, we point out that the blocker of a balanced matrix is in general *not* balanced, see Fulkerson, Hoffman & Oppenheim [1974] for a counterexample. It is surprising and remarkable that the strong max-min equality for $\operatorname{bl} A, A$, can (in a certain sense) be strengthened further into a combinatorial max partioning-min covering theorem.

1.7.4 Theorem (Partition into Transversals, Berge [1971])
Let A^{T} be a balanced matrix and $B = \operatorname{bl} A^{\mathrm{T}}$ its blocker. Then:

$$\max_{y^{\mathrm{T}}B = \mathbf{1}^{\mathrm{T}},\ y^{\mathrm{T}} \ge 0^{\mathrm{T}},\ y^{\mathrm{T}} \text{ integral}} y^{\mathrm{T}}\mathbf{1} = \min\ A^{\mathrm{T}}\mathbf{1}.$$

To state Theorem 1.7.4 in terms of hypergraphs, note that the blocker B of A^{T} is the incidence matrix of all transversals of $\mathcal{H}(A)$ versus nodes (each row is a transversal, each column a node). Then the theorem states the following: If $k = \min \ A^{\mathrm{T}}\mathbf{1}$ is the minimum size of an edge of a balanced hypergraph $\mathcal{H}(A)$, there exist k transversals in $\mathcal{H}(A)$ that partition the vertices or, to put it differently, there is a k-coloring of $\mathcal{H}(A)$ such that each edge contains a node of each color; this is Berge [1971]'s Theorem 2.

We have seen by now that balanced matrices have analogous, but stronger combinatorial properties than perfect and ideal ones and this trend continues in the study of the *recognition problem*. The scenario differs slightly from the one for perfection and ideality testing, though. First, we explicitly know the complete (infinite) list of all forbidden minors. Second, there is no controversy about using the matrix itself as the input to the recognition algorithm: Nobody has suggested a graphical (or other) representation of an $m \times n$ balanced matrix that is polynomial in n, and mn is accepted as just fine an encoding length. In this setting, one of the most startling results on balanced matrices was the recent construction of an algorithm that recognizes this class in polynomial time by Conforti, Cornuéjols & Rao [1991]. This algorithm is based on decomposition methods, that recursively break a 0/1 matrix into "elementary pieces" in such a way that the balancedness of the whole is equivalent to balancedness of the pieces, and such that the pieces are of combinatorial types whose balancedness can be established or disproved. The recognition of the pieces is based on earlier work on classes of so-called totally balanced, strongly balanced, and linearly balanced matrices.

1.7.5 Theorem (Recognition of Balancedness, Conforti, Cornuéjols & Rao [1991]) *The recognition problem for balanced matrices is in $\mathcal{P}$.*

Like for perfect and ideal matrices, there is a new branch of research that investigates the more general class of *balanced* $0/\pm 1$ *matrices*. Conforti & Cornuéjols [1992] show, for instance, that the members of this class can also be characterized in terms of 2-colorability and that the associated packing, covering, and partitioning system are TDI, even in arbitrary "mixes". An overview on balanced $0/\pm 1$ matrices can be found in the survey article Conforti, Cornuéjols, Kapoor & Vušković [1997].

We close this section with a remark on the integrality of *fractional set partitioning polytopes*. By Theorem 1.7.2 (vi), the balanced matrices form a class that gives rise to integral polytopes of this type, like perfect and ideal matrices do, too, but these are *not* all matrices with this property. For a trivial example, consider the matrix

$$A = \left(\begin{array}{c|ccc|ccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right),$$

that is composed from an imperfect "top" and a nonideal "bottom". It is easy to see that A is neither perfect, nor ideal, nor balanced, but one can verify that the fractional set partitioning polytope $P^{=}(A)$ is integral; $P^{=}(A)$ consists, in fact, of the single point e_1. We see that the occurrence of forced variables allows to blow up a matrix with all kinds of "garbage" and difficulties of this sort are the reason why there is no minor theory for matrices with integer set partitioning polytopes.

1.8 The Set Packing Polytope

The set packing problems of the previous sections were almost always assumed to have a constraint matrix that is perfect; now we turn to the general case with arbitrary 0/1 matrices. Such matrices lead to nonintegral systems $Ax \leq \mathbf{1}$, $x \geq 0$ that do not suffice to describe the set packing polytope $P_I(A)$. The *polyhedral study* of general set packing polytopes aims at identifying the missing inequalities and at developing methods for their effective computation. Such knowledge of the facial structure of set packing polytopes is useful in three ways: To develop polynomial time algorithms for classes of stable set problems, to derive combinatorial min-max results, and computationally in branch-and-cut codes for the solution of set packing or set partitioning problems. Let us say a word about each of these points.

The link between *polynomial time algorithms* and facial investigations is a fundamental algorithmic result of Grötschel, Lovász & Schrijver [1988] that is often termed the *polynomial time equivalence of separation and optimization.* It is based on the concept of a *separation oracle* for a polyhedron P[11] that takes an arbitrary point $\overline{x}$ as input and decides if it is contained in P, or, if not, returns an inequality that separates $\overline{x}$ and P. The theory asserts that, whenever such an oracle is at hand, one can optimize over P in *oracle polynomial time*, where each call of the oracle is counted as taking constant time. When the separation can also be done in polynomial time, this results in a polynomial optimization algorithm — even and in particular when a *complete description* of P by linear inequalities has exponential size! And it turns out that one can construct such polynomial separation oracles for the set packing polytopes of quite some classes of graphs, most notably for perfect graphs.

Combinatorial min-max results require explicit complete descriptions by TDI systems. It is theoretically easy to "make a linear system TDI", but it is difficult to obtain systems of this type with "combinatorial meaning". In fact, besides perfect and line graphs there seems to be only one class of "odd K_4 free" graphs where a combinatorial min-max result is known.

The *computational use* of set packing inequalities goes to the other extreme: Anything goes, valid inequalities can be used as well as facet defining ones, and whether exact separation is always preferable to heuristics — well, it's wiser not to enter this discussion!

We try to survey in this section the main results of the polyhedral approach to the set packing problem. The organization of the section is as follows. Subsection 1.8.1 introduces the concept of facet defining graphs and gives a list of known such structures as well as of graphs where these inequalities yield complete descriptions. Subsection 1.8.2 deals with composition procedures, that construct from simple inequalities more complicated ones. Some results on a special class of claw free graphs are collected in Subsection 1.8.3. Quadratic and semidefinite approaches are treated in Subsection 1.8.4. The final Subsection 1.8.5 states some adjacency results, that bear on primal algorithms.

Some basic properties of set packing polytopes for reference in subsequent subsections are:

1.8.1 Observation (Dimension, Down Monotonicity, Nonnegativity)
Let A be a 0/1 matrix and $P_I(A)$ be the associated set packing polytope.

(i) *$P_I(A)$ is full dimensional.*

(ii) *$P_I(A)$ is down monotone, i.e., $x \in P_I(A) \Longrightarrow y \in P_I(A)$ for all $0 \leq y \leq x$.*
In particular, all nontrivial facets of $P_I(A)$ have all nonnegative coefficients.

(iii) *The nonnegativity constraints $x_j \geq 0$ induce facets of $P_I(A)$.*

[11] The theory works also for convex bodies.

1.8.1 Facet Defining Graphs

There are three general techniques to find valid or facet defining inequalities for the set packing polytope: The study of *facet defining graphs*, the study of *semidefinite relaxations* of the set packing polytope, and the study of *combinatorial relaxations*. We discuss in this section the first technique, semidefinite relaxations are treated in Subsection 1.8.4, and combinatorial relaxations in Chapter 2 and in particular in Section 2.5.
The polyhedral study of general set packing polytopes through classifications of graphs, initiated by Padberg [1973a], is based on the down monotonicity of $P_I(A)$. Namely, this property implies that if $H = G[W]$ is some node induced subgraph of some given graph $G = (V, E)$ and the inequality $a^{\mathrm{T}}x \leq \alpha$ is valid for $P_I(G) \cap \{x \in \mathbb{R}^V \mid x_j = 0 \quad \forall j \notin W\}$ and has $a_j = 0$ for all $j \notin W$, it is also valid for $P_I(G)$. The consequence is that *substructures* of a graph give rise to valid inequalities for the set packing polytope of the whole graph, a relation that can be stressed by identifying the polytopes $P_I(G) \cap \{x \in \mathbb{R}^V \mid x_j = 0 \quad \forall j \notin W\}$ and $P_I(H)$ (and we want to use this notation here and elsewhere in this section).
An investigation of the rules that govern the transfer of inequalities from set packing subpolytopes to the whole and vice versa leads to the concepts of *facet defining graphs* and *lifting*, see Padberg [1973a]. We say that a node induced subgraph $H = G[W]$ of G *defines* the facet $a^{\mathrm{T}}x \leq \alpha$ if this inequality is essential for $P_I(H)$. Now, among all node induced subgraphs $H = G[W]$ of G are those of particular interest that are minimal in the sense that they give rise to a facet "for the first time". This is not always the case: If $H = G[W]$ defines the facet $a^{\mathrm{T}}x \leq \alpha$ for $P_I(G[W])$, it is possible that there is a smaller subgraph $G[U] \subseteq G[W]$ ($U \subseteq W$), such that $a^{\mathrm{T}}x \leq \alpha$ defines already a facet of $P_I(G[U])$. If this is not the case for all $U \subseteq W$ such that $|U| = |W| - 1$, the subgraph $G[W]$ is "elementary plus/minus one node" and said to *produce* $a^{\mathrm{T}}x \leq \alpha$, see Trotter [1975], and if this property extends to any subset $U \subseteq W$, the subgraph $G[W]$ is said to *strongly produce* the inequality. Having mentioned these concepts, we do, however, restrict our attention in the sequel to facet defining graphs and refer the reader to the survey article of Padberg [1977] for a discussion of facet producing graphs. Moving in the other direction again, from small to large, the question of what kind of extensions of valid inequalities/facets from subgraphs result in valid inequalities/facets for set packing polytopes of supergraphs is precisely the lifting problem that we discuss in the next section.
We give next a *list* of facet defining classes of graphs. For each such class $\mathcal{L}$, one can try to determine a corresponding class of $\mathcal{L}$*-perfect* graphs, whose associated set packing polytopes can be described completely in terms of $\mathcal{L}$ (plus the edge inequalities, where appropriate). This concept, invented by Grötschel, Lovász & Schrijver [1988], provides a general technique to identify classes of graphs with polynomially solvable stable set problems: Namely, to establish such a result, one merely has to prove that the inequalities from $\mathcal{L}$ can be separated in polynomial time! Our list includes also these results as far as we are aware of them.

Edge Inequalities. Associated to each edge ij of a graph $G = (V, E)$ is the *edge inequality* $x_i + x_j \leq 1$. Edge inequalities are special cases of clique inequalities and inherit the facetial properties of this larger class, see next paragraph. The *edge perfect graphs* are exactly the bipartite graphs without isolated nodes, and these have polynomially solvable stable set problems. For general graphs G, the system of edge inequalities (plus the nonnegativity inequalities) $A(G)x \leq \mathbf{1}, x \geq 0$ defines an *edge relaxation* of $P_I(A)$. This relaxation has been investigated by a number of authors, including Padberg [1973a] and Nemhauser & Trotter [1973], and displays some initially promising looking properties. Namely, $P(A(G))$ has only *half integral* vertices (all components are 0, 1/2, or 1 only) and, stronger, all integer components of a solution of the associated fractional set packing problem have the same value in some optimal integral solution and can thus be fixed! Unfortunately, this almost never happens in computational practice and neither does it happen in theory: Pulleyblank [1979] proved that the probability that the edge relaxation of a set packing problem with $w = \mathbf{1}$ on a random graph has an all 1/2 optimal solution tends to one when the number of nodes tends to infinity. And this is not only asymptotically true: For $n = 100$, the probability of a single integer component is already less than 1.4×10^{-8}.

Clique Inequalities, Fulkerson [1971], Padberg [1973a]. A *clique* in a graph $G = (V, E)$ is a set Q of mutually adjacent nodes, see Figure 1.2. Associated to such a structure is the *clique inequality*

$$\sum_{i \in Q} x_i \leq 1.$$

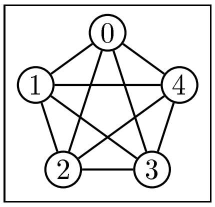


Figure 1.2: A 5-Clique.

(The support graphs of) Clique inequalities are trivially *facet* defining. Moreover, Fulkerson [1971] and Padberg [1973a] have shown that such a constraint induces also a facet for the stable set polytope of a supergraph if and only if the clique is maximal with respect to set inclusion in this supergraph. By definition, the *clique perfect graphs* coincide with the perfect graphs. *Separation* of clique inequalities is $\mathcal{NP}$-hard, see Garey & Johnson [1979], but this complexity result is irrelevant because Grötschel, Lovász & Schrijver [1988] have shown that the clique inequalities are contained in a larger class of polynomially separable *orthogonality inequalities*, that we will discuss in Subsection 1.8.4. This implies that the stable set problem for perfect graphs can be solved in polynomial time! This result, one of the most spectacular advances in combinatorial optimization, subsumes a myriad of statements of this type for subclasses of perfect graphs, see Grötschel, Lovász & Schrijver [1988] for a survey.

Odd Cycle Inequalities, Padberg [1973a]. An *odd cycle* C in a graph $G = (V, E)$ consists of an odd number $2k + 1$ of nodes $0, \ldots, 2k$ and the edges $(i, i + 1)$ for $i = 0, \ldots, 2k$ (where indices are taken modulo $2k + 1$), see Figure 1.3. Any additional edge ij between two nodes of a cycle that is not of the form $(i, i + 1)$ is a *chord*. An odd cycle without chords is an *odd hole*; the odd holes coincide with the circulant(graph)s $C(2k + 1, 2)$. Associated to a not necessarily chordless odd cycle C on $2k + 1$ nodes is the *odd cycle inequality*

$$\sum_{i \in C} x_i \leq (|C| - 1)/2.$$

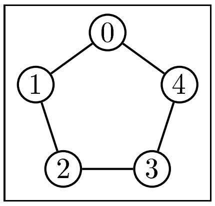


Figure 1.3: A 5-Cycle.

Padberg [1973a] showed that (the support graph of) an odd cycle inequality is *facet* defining if and only if the cycle is a hole; note that the "only if" part follows from minimal imperfection. Grötschel, Lovász & Schrijver [1988, Lemma 9.1.11] gave a polynomial time algorithm to *separate* odd cycle inequalities such that the stable set problem for *cycle (plus edge) perfect graphs*, that are also called *t-perfect* (t stands for trou, the French word for hole), is solvable in polynomial time. *Series parallel graphs* (graphs that do not contain a subdivision of K_4 as a minor) are one prominent class of cycle (plus edge) perfect graphs. This was conjectured by Chvátal [1975], who showed that the stable set problem for $w = \mathbf{1}$ can be solved in polynomial time, and proved by Boulala & Uhry, a short proof was given by Mahjoub [1985]. In fact, even more is true, and for a larger class: Gerards [1989] proved that the system of nonnegativity, edge, and odd cycle inequalities is TDI for graphs that do not contain an *odd* K_4, i.e., a subdivision of K_4 such that each face cycle is odd. This gives rise to a min cycle and edge covering-max node packing theorem. Perfect graphs, line graphs (see next paragraph), and Gerards's class seem to be the only instances where such a min-max result is known. A list of further cycle perfect graphs can be found in Grötschel, Lovász & Schrijver [1988].
Taking the union of clique and odd cycle inequalities, one obtains the class of *h-perfect* graphs, see again Grötschel, Lovász & Schrijver [1988] for more information.

Blossom Inequalities, Edmonds [1965]. The *matchings* in a graph $H = (V, E)$ are in one-to-one correspondence to the stable sets in the *line graph* $L(H) := (E, \{(ij, jk) \in E^2\})$ of H. Associated to such a linegraph $L(H)$ is the *blossom inequality*

$$\sum_{e \in E} x_e \leq \lfloor |V|/2 \rfloor.$$

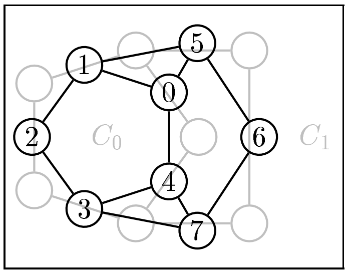


Figure 1.4: A Line Graph of a 2-Connected Hypomatchable Graph.

Edmonds & Pulleyblank [1974] showed that a blossom inequality is *facet* defining for $P_I(L(H))$ if and only if H is 2-connected and *hypomatchable*. (If we denote by $\nu(H)$ the maximum size of a matching in a graph H, this graph is hypomatchable if $\nu(H) = \nu(H - i)$ holds for all contractions $H - i$ of the graph H. It is known that a graph $H = (V, E)$ is 2-connected and hypomatchable if and only if it has an *open ear decomposition* $E = \dot\bigcup_{i=0}^{k} C_i$, where C_0 is an odd hole and each C_i is a path with an even number of nodes $v_i^1, \ldots, v_i^{2k_i}$ and distinct endnodes $v_i^1 \neq v_i^{2k_i}$, such that $V(C_i) \cap \dot\bigcup_{j=0}^{i-1} V(C_j) = \{v_i^1, v_i^{2k_i}\}$, see Lovász & Plummer [1986, Theorem 5.5.2] and Figure 1.4.) *Separation* of blossom inequalities is equivalent to a minimum *odd cut* problem, see Grötschel, Lovász & Schrijver [1988, page 256], for which Padberg & Rao [1982] gave a polynomial time algorithm. Edmonds [1965] has shown that the stable set polytope of a line graph is completely described by the nonnegativity, blossom, and the clique inequalities $\sum_{e \in \delta(i)} x_e \leq 1$ for all $i \in V$; this means that the class of *blossom (and clique) perfect graphs* subsumes the class of line graphs. These arguments yield a polynomial time algorithm for the stable set problem in line graphs (the matching problem in graphs) alternative to the celebrated combinatorial procedure of Edmonds [1965]. Finally, we mention that Cunningham & Marsh [1978] have shown that the above mentioned complete description of the set packing polytope of a line graph is even TDI, which results in a combinatorial min packing-max covering theorem for edges/blossoms and cliques in graphs.

Odd Antihole Inequalities, Nemhauser & Trotter [1973]. An *odd antihole* $\overline{C}$ is the complement of an odd hole, see Figure 1.5; the odd antiholes coincide with the circulants $C(2k+1,k)$. Associated to an odd antihole on $2k+1$ nodes is the *odd antihole inequality*

$$\sum_{i \in \overline{C}} x_i \leq 2.$$

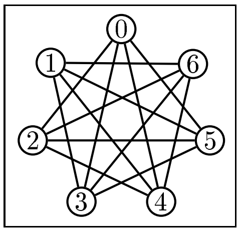


Figure 1.5: A 7-Antihole.

Odd antihole inequalities are *facet* defining by minimal imperfection. As far as we know, no combinatorial *separation* algorithm for these constraints is known, but the odd antihole inequalities are contained in a larger class of *matrix inequalities* with N_+-index 1, that can be separated in polynomial time, see Lovász & Schrijver [1991]; we will discuss the matrix inequalities in Subsection 1.8.4. These results imply that stable set problems for *antihole perfect graphs* can be solved in polynomial time.

Wheel Inequalities. A *wheel* in a graph $G=(V,E)$ is an odd cycle C plus an additional node $2k+1$ that is connected to all nodes of the cycle, see Figure 1.6. C is the *rim* of the wheel, node $2k+1$ is the *hub*, and the edges connecting the node $2k+1$ and i, $i=0,\ldots,2k$, are called *spokes*. For such a configuration we have the *wheel inequality*

$$kx_{2k+1} + \sum_{i=0}^{2k} x_i \leq k.$$

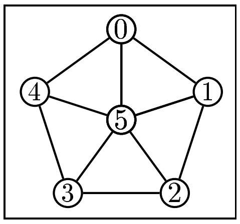


Figure 1.6: A 5-Wheel.

Note that wheel inequalities can have coefficients of arbitrary magnitude.
A wheel inequality can be obtained by a *sequential lifting* (see next subsection) of the hub into the odd cycle inequality for the rim. Trying all possible hubs, this yields a polynomial time *separation* algorithm for wheel inequalities. An alternative procedure, that reduces wheel separation to odd cycle separation, can be found in Grötschel, Lovász & Schrijver [1988, Theorem 9.5.6]. Hence, the stable set problem for *wheel perfect graphs* is solvable in polynomial time.
Generalizations of wheel inequalities that can be obtained by subdividing the edges of a wheel were studied by Barahona & Mahjoub [1994], who derive a class of K_4 inequalities (see the corresponding paragraph in this subsection), and by Cheng & Cunningham [1997], who give also a polynomial time separation algorithm for two such classes.

We finally refer the reader to Subsection 2.5.1 of this thesis, where we show that simple as well as generalized wheels belong to a (larger) class of "odd cycles of paths" of a combinatorial *rank relaxation* of the set packing polytope; the inequalities of this superclass can be separated in polynomial time. We remark that the wheel detection procedure of Grötschel, Lovász & Schrijver [1988, Theorem 9.5.6] is, with this terminology, *exactly* a routine to detect cycles of paths of length 2 with one hub endnode.

Antiweb and Web Inequalities, Trotter [1975]. *Antiweb* is a synonym for circulant, see Figure 1.7, and a *web* is the complement of an antiweb, see Figure 1.8. Obviously, every odd hole is an antiweb, and every odd antihole is a web. An odd antihole is also an antiweb, but the classes of antiwebs and webs do in general not coincide; in fact, Trotter [1975] proved that an antiweb is a web if and only if it is a hole or an antihole. The inequalities associated to $C(n,k)$ and $\overline{C}(n,k) := \overline{C(n,k)}$ are the *antiweb inequality* and the *web inequality*

$$\sum_{i \in C(n,k)} x_i \leq \lfloor n/k \rfloor \qquad \sum_{i \in \overline{C}(n,k)} x_i \leq k.$$

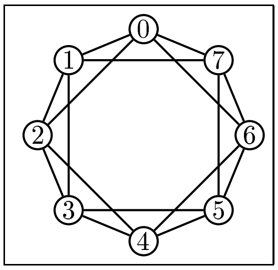


Figure 1.7: The Antiweb $C(8,3)$.

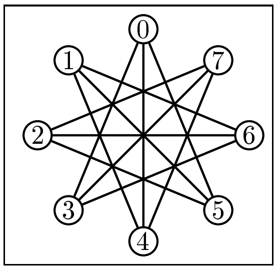


Figure 1.8: The Web $\overline{C}(8,3)$.

An antiweb $C(n,k)$ is *facet* defining if and only if either $k = n$ or k and n are relatively prime, a web $\overline{C}(n,k)$ defines a facet if and only if either $k = 1$ or k and n are relatively prime. As far as we known, no polynomial time *separation* algorithm for these classes themselves or any superclass is known.

Wedge and $\mathbf{K}_4$ Inequalities, Giles & Trotter [1979], Barahona & Mahjoub [1989]. To construct a *wedge*, one proceeds as follows: Take a 3-wheel K_4, subdivide its *spokes* (*not* the rim, and at least one subdivision must really add a node) such that each face cycle is odd, and take the complement; the resulting graph is a wedge, see Figure 1.9 for a complement of a wedge. If we subdivide the nodes of a wedge into the set of nodes $\mathcal{E}$ that have an even distance from the original rim nodes of the 3-wheel, and the set of remaining nodes $\mathcal{O}$, the *wedge inequality* states that

$$\sum_{i \in \mathcal{E}} x_i + \sum_{i \in \mathcal{O}} 2x_i \leq 3.$$

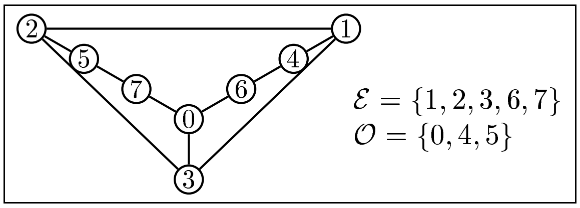


Figure 1.9: A Complement of a Wedge.

The wedges are *facet defining*, see Giles & Trotter [1979]. Nothing seems to be known about the *separation* of this class.
By construction, the complements of wedges (as in this thesis) are special subdivisions of K_4. *All* subdivisions of K_4 have been analyzed by Barahona & Mahjoub [1989]. It turns out that *complete descriptions* of the set packing polytopes associated to arbitrary subdivisions of K_4 can be obtained by means of 19 classes of K_4 *inequalities*. The *separation* of K_4 inequalities does not seem to have been investigated.

Chain Inequalities, Tesch [1994]. A $2k+1$-*chain* H is similar to the antiweb $C(2k+1,3)$; the difference is that the two chords $(0,2k-1)$ and $(1,2k)$ are replaced with the single edge $(1,2k-1)$, see Figure 1.10. This structure gives rise to an inequality for the set packing polytope. The *chain inequality* states that

$$\sum_{i \in H} x_i \leq \left\lfloor \frac{2k+2}{3} \right\rfloor .$$

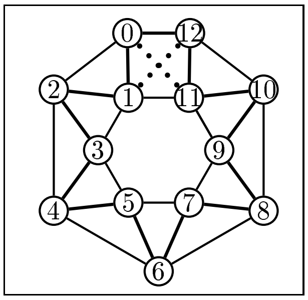


Figure 1.10: A 13-Chain.

A $2k+1$ chain is *facet* defining if and only if $k \bmod 3 = 0$. Nothing is known about the *separation* problem.

Composition of Circulant Inequalities, Giles & Trotter [1979]. A *composition of circulants* is constructed in the following way. Choose a positive integer k, let $n = 2k(k+2)+1$, set up the "inner circulant" $C = C(n, k+2)$ and the "outer circulant" $C' = C(n, k+1)$ with node sets $V = \{0, \ldots, n-1\}$ and $V' = \{0, \ldots, (n-1)'\}$, and add all edges $ii', \ldots, i(i+2k+1)'$ for all nodes $i \in V$, (indices taken modulo n). The graph that one obtains from an application of this procedure for any positive k is a composition of circulants that is denoted by $\mathcal{C}_k$, see Figure 1.11. Associated to such a structure is the *composition of circulants inequality*

$$(k+1) \sum_{i \in V} x_i + k \sum_{i' \in V'} x_{i'} \leq 2k(k+1).$$

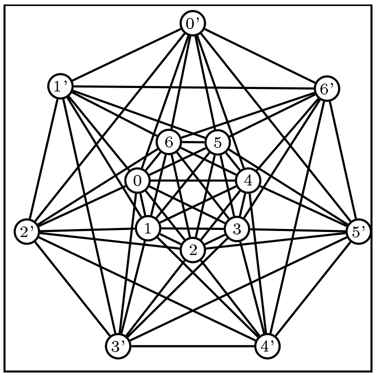


Figure 1.11: The Composition of Circulants $\mathcal{C}_1$.

It is known that composition of circulant inequalities are *facet* defining.

Further Inequalities. We close our list of facet defining inequalities for the set packing polytope with some pointers to further classes.

An *enumeration* of all facets of the set packing polytopes associated to certain *claw free* graphs (see next subsection) of up to 10 nodes has been done by by Euler & Le Verge [1996]. We finally refer the reader to Section 2.5 of this thesis, where we present a class of facet defining *cycle of cycles* inequality as an example of a method to derive facet defining inequalities from a "rank relaxation" of the set packing polytope.

*

We have seen that most of the facet defining graphs of our list appeared in *pairs* of graph and complement graph that give both rise to facets and one could thus be lead to believe that some (yet to be made precise) principle of this sort holds in general. Padberg [1977] offers some sufficient conditions in this direction but also points out that graphs like the line graph in Figure 1.4 have complements that are not facet producing (no facet defining inequality of the associated set packing polytope has full support).

Our discussion of facet defining graphs would not be complete without mentioning the *necessary and sufficient conditions* that have been derived for structures of this type. It is hard to come up with interesting characterizations of general constraints and the literature focusses on the already notoriously difficult class of *rank inequalities* or *canonical inequalities*, as they are also called. Denoting, us usual, the stability number or *rank* of a graph G by $\alpha(G)$, the rank inequality that is associated to G is

$$\sum_{i\in V} x_i \leq \alpha(G).$$

A *necessary* condition for a rank and more general for any inequality to define a facet is:

1.8.2 Observation (2-Connectedness of a Facet's Support)
Let G be a graph and $P_I(G)$ the associated set packing polytope. If $a^{\mathrm{T}}x \leq \alpha$ defines a facet of $P_I(G)$, its support graph $G[\operatorname{supp} a^{\mathrm{T}}]$ *is 2-(node-)connected.*

Observation 1.8.2, which is a special case of the more general Theorem 1.8.8 (to be discussed in the next section), is, as far as we know, the only general condition that is known; the criteria that follow apply to the rank case with all one coefficients.

We start with a *sufficient* condition of Chvátal [1975]. His criterion for facetial rank inequalities is based on the concept of *critical edges* in a graph $G = (V, E)$. Namely, an edge $ij \in E$ is called *critical* if its removal increases G's rank, i.e., if $\alpha(G - ij) = \alpha(G) + 1$. A graph G itself is called *critical*, if all of its edges are critical.

1.8.3 Theorem (Rank Inequalities from Critical Graphs, Chvátal [1975])
Let $G = (V, E)$ be a graph and E^* be the set of its critical edges. If the graph $G^* := (V, E^*)$ is connected, the rank inequality $\sum_{i\in V} x_i \leq \alpha(G)$ is facet defining.

The reader can verify that most of the rank inequalities in this section's list satisfy the criterion of Theorem 1.8.3 (in fact, most have even critical support graphs) but this condition is not necessary, see Balas & Zemel [1977] for a counterexample.

A set of further conditions, suggested by Balas & Zemel [1977], makes use of the notion of a *critical cutset* in a graph $G = (V, E)$, i.e., a cut $\delta(W)$ such such that $\alpha(G - \delta(W)) > \alpha(G)$. In words: A cut(set) is critical if its removal increases the rank.

1.8.4 Theorem (Critical Cutsets, Balas & Zemel [1977])
Let G be a graph and $P_I(G)$ be the associated set packing polytope. If the rank inequality for G defines a facet of $P_I(G)$, every cutset in G is critical.

Balas & Zemel [1977] give an example that shows that this condition is not sufficient. But it is possible to obtain a complete characterization of those rank facets that arise from facet defining *subgraphs* of a graph.

1.8.5 Theorem (Extension of Rank Facets, Balas & Zemel [1977])
Let G be a graph, $P_I(G)$ be the associated set packing polytope, $W \subseteq V$ some subset of nodes of G, and let the rank inequality $\sum_{i \in W} x_i \leq \alpha(G[W])$ be facet defining for $P_I(G[W])$. Then:

The rank inequality $\sum_{i \in W} x_i \leq \alpha(G[W])$ defines a facet for $P_I(G)$ if and only if the cutset $\delta(j)$ with respect to the graph $G[W \cup \{j\}]$ is not critical for every $j \notin W$.

It has been pointed out by Laurent [1989] that Theorems 1.8.3, 1.8.4, and 1.8.5 carry over to the more general context of rank facets of *set covering polytopes*, see also Section 1.9. For the notion of critical cutsets, this correspondence is as follows. If we interpret the stable sets in a graph G as the independent sets of an independence system (see Subsection 1.3), Theorem 1.8.4 says that V is *nonseparable*, while stating that all cutsets $\delta(j)$ with respect to the graphs $G[W \cup \{j\}]$ are not critical as in Theorem 1.8.5 is equivalent to W being *closed*.

1.8.2 Composition Procedures

In the preceding Subsection 1.8.1, we have studied and accumulated a list of *facet defining graphs*, that have a local relevance in the sense that they are facet defining for their associated set packing polytopes. In general, the given graph will rarely be of one of the special facet defining classes, but it is not only possible, but, as we known from the minor investigations of Section 1.6, *inevitable* that a given graph contains imperfect substructures of such types. Then, by down monotonicity, the associated inequalities carry over from the set packing polytopes of the subgraphs to the whole.
The procedure that we have just described is a simple example of a *constructive approach* to the study of the set packing polytope. The idea here is the following: Given valid/facet defining inequalities for one or several "small" graphs, *compose* valid/facet defining inequalities for a "bigger" graph. In this way, we can build on *analytic* classifications of facet defining graphs and *synthesize* global inequalities from elementary pieces.
In this subsection, we survey two *composition procedures* of this type: The *lifting* method and the study of the polyhedral consequences of *graph theoretic operations*.

Sequential Lifting, Padberg [1973a]. The *sequential lifting method*, that was introduced by Padberg [1973a] in connection with odd cycle inequalities, applied to arbitrary facets of set packing and set covering polyhedra by Nemhauser & Trotter [1973], and further extended to arbitrary 0/1 polytopes by Zemel [1978], provides a tool to build iteratively facets for the set packing polytope $P_I(G)$ associated to some graph G from facets of subpolytopes of the form $P_I(G[W])$.

1.8.6 Theorem (Sequential Lifting, Padberg [1973a], Nemhauser & Trotter [1973])
Let $G = (V, E)$ be a graph and $P_I(G)$ the associated set packing polytope. Let further $W = \{w_1, \ldots, w_k\} \subseteq V$ be some subset of nodes of G that is numbered in some arbitrary order, let $\overline{W} := V \setminus W$ be the complement of W, and let $a^{\mathrm{T}}x \leq \alpha$ be a facet defining inequality for $P_I(G[\overline{W}])$.

Determine numbers $\beta_i \in \mathbb{R}$ for $i = 1, \ldots, k$ by means of the recursion

$$\beta_i := \alpha - \max_{x \in P_I(G[\overline{W} \cup \{1,\ldots,i-1\}])} a^{\mathrm{T}}x + \sum_{j=1}^{i-1} \beta_j x_j, \qquad i = 1, \ldots, k. \tag{1.9}$$

Then:

The inequality $a^{\mathrm{T}}x + \sum_{i=1}^{k} \beta_i x_i \le \alpha$ is facet defining for $P_I(G)$.

The ordering of the nodes in W is called a *lifting sequence*, (1.9) is a *lifting problem*, the numbers β_i are the *lifting coefficients*,, the inequality $a^{\mathrm{T}}x + \sum_{i=1}^{k} \beta_i x_i \le \alpha$ is a *lifting* of the inequality $a^{\mathrm{T}}x \le \alpha$, and the whole procedure is referred to as "lifting the variables (or nodes) in W into the inequality $a^{\mathrm{T}}x \le \alpha$".

Some simple *properties* of the lifting process are the following. If we start with a nonnegative inequality (which we assume in the sequel), all lifting coefficients will be nonnegative as well and the right-hand side α of the original inequality is an *upper bound* on the value of each of them. Taking a *lower bound* for the value of some lifting coefficient is called a *heuristic lifting* step; if we do that one or several times, the resulting inequality will in general not be facet defining, but it will be valid. Next, note that different choices of the *lifting sequence* give rise to different liftings that have, however, an identical core $a^{\mathrm{T}}x \le \alpha$. We remark in this context that one can also consider the possibility to compute several or all lifting coefficients at once, an idea that is called *simultaneous lifting*, see again Zemel [1978].

We have already encountered a prominent *example* of a lifting: A wheel inequality can be obtained by lifting the hub into the odd cycle inequality that corresponds to the rim.

Sequential lifting is a powerful *conceptual* tool that offers an explanation for the appearance of facet defining inequalities of general set packing polytopes. Such inequalities frequently resemble the pure facet defining substructures as in Subsection 1.8.1, but with all kinds of additional protuberances; the aberrations can be understood as the results of sequential liftings. We remark that one does in general not obtain all facets of a set packing polytope $P_I(G)$ from sequential liftings of facets of subpolytopes, namely and by definition, when the graph G itself is facet producing; examples of facet producing graphs are odd holes.

Turning to the *algorithmic* side of lifting, we note that the lifting problem is again a set packing problem, one for each lifting coefficient. So lifting is *in principle* a difficult task. But the procedure is very flexible and offers many tuning switches, that can be used to reduce its complexity in rigorous and in heuristic ways. First, note that when the right-hand side α is *bounded*, the lifting problem can be solved by enumeration in *pseudo polynomial* time, i.e., time that is polynomial in the size of the data and the *value* of α. For instance, clique inequalities have a right-hand side of one and so will be all lifting coefficients; sequentially lifting a clique inequality is simply the process to extend the clique with additional nodes in the order of the lifting sequence until the clique is maximal with respect to set inclusion, and this is easy to do in polynomial time. In a similar fashion, one can come up with polynomial time lifting schemes for antihole inequalities etc. — all for a *fixed lifting sequence*. Second, there are many degrees of freedom for heuristic adjustments: One can switch from exact to heuristic lifting when the lifting problems become hard, stop at any point with a result in hand, make choices in an adaptive and dynamical way, etc. To put it short: Lifting is not the algorithmic panacea of facet generation, but it is a useful and flexible tool to *enhance* the quality of any given inequality. Some applications of lifting in a branch-and-cut code for set partitioning problems and some further discussion on computational and implementation issues can be found in Section 3.3 of this thesis.

The last aspect that we consider here is that the lifting method offers also an explanation for the difficulties that one encounters in classifying the facets of the set packing polytope: It is extremely easy to use the procedure to construct examples of arbitrarily *complex* inequalities with involved graphical structures and any sequence of coefficients. Does this mean that the attempt to understand the facial structure of set packing polytopes by analysis of small structures is useless? Maybe — but maybe things are not as bad. Padberg [1977] argues that small facet defining graphs may, in a "statistical" sense, give rise to reasonable fractional relaxations of general set packing polytopes. It is, however, known that there is no polynomial time *approximation algorithm* for set packing, see, e.g., Hougardy, Prömel & Steger [1994].

*

Graph Theoretic Operations. We consider in the following paragraphs composition procedures that are based on *graph theoretic operations*: Taking one or several graphs, possibly of special types, we glue these pieces together to obtain a new graph, possibly again of a special type. Studying the polyhedral consequences of such an operation, one tries to derive (i) analogous procedures for the composition of valid/facet defining inequalities or, more ambitious, (ii) complete descriptions for the set packing polytope of the composition from complete descriptions for the pieces.

Extensions. The first operation that we consider is the *extension* of a graph with additional nodes. *Sequential lifting* is an example of this doing when we reverse our point of view from "top-down" to "bottom-up": If we do not look at the seed graph $G[W]$ of Theorem 1.8.6 as a subgraph of a bigger graph that is given in advance, but as a graph of its own, the graph theoretic operation behind each lifting step turns out to be the addition of a single node. Adding bigger structures results in special simultaneous lifting procedures. As an example, we mention a procedure of Wolsey [1976] and Padberg [1977], who consider the extension of a graph G with a $K_{1,n}$: A single node is joined to every node of G with a path of length 2. Some aspects of this procedure are discussed in Subsection 1.8.2 of this thesis, and we mention here only that Padberg [1977] has shown that the method can not only be used to extend facet *defining* graphs, but to construct facet *producing* graphs (see this section's introduction) that give rise to facets with arbitrarily complex coefficients.

Substitutions. This is a second group of powerful graph theoretic manipulations: One takes a graph, selects some node or subgraph, substitutes another graph for this component, and joins the substitution to the whole in some way.
A first and important example of such a procedure is due to Chvátal [1975], who considered the replacement of a node v' of a graph $G' = (V', E')$ by a second graph $G'' = (V'', E'')$ (*node substitution*). The graph G that results from this operation is the union of $G' - v'$ and G'' with additional edges that join all nodes of G'' to all neighbors of v' in G'. Note that node substitution subsumes the *multiplication* or *replication* of a node to a clique of Fulkerson [1972] and Lovász [1971], which plays a role in the theory of perfect graphs. Further, substituting graphs G_1 and G_2 for the two nodes of an edge yields the *sum* (sometimes also called join, but we want to use this term later in another way), and substituting G'' for every node of G' the *lexicographic product* or *composition* of G' and G''. Node substitution has the following polyhedral consequences.

1.8.7 Theorem (Node Substitution, Chvátal [1975]) *Let G' and G'' be graphs and let $A'x' \leq b'$, $x' \geq 0$ and $A''x'' \leq b''$, $x'' \geq 0$ be complete descriptions of $P_I(G')$ and $P_I(G'')$. Let v' be a node of G' and G be the graph that results from substituting G'' for v'. Then:*

The system

$$\begin{aligned} a'_{iv'} \sum_{u'' \in V''} a''_{ju''} x_{u''} + \sum_{\substack{u' \in V' \\ u' \neq v'}} a'_{iu'} b''_j x_{u'} &\leq b'_i b''_j, \qquad \forall i, j \\ x &\geq 0 \end{aligned}$$

is a complete description of $P_I(G)$

Note that this system is of polynomial size with respect to the encoding length of the starting systems $A'x' \leq b'$, $x' \geq 0$ and $A''x'' \leq b''$, $x'' \geq 0$.

Other authors have considered similar operations. Wolsey [1976] obtains facet lifting results from studying the replacement of a node with a path of length 2 and of an edge with a path of length 3 (*edge subdivision*); different from Chvátal [1975]'s node substitution, these paths are *not* connected in a uniform way to the original graph. Some discussion of the first of these procedures can be found in Subsection 1.8.2 of this thesis.

Operations related to paths have also been considered by Barahona & Mahjoub [1989]. They transform facets using *subdivisions of stars*, i.e., simultaneous replacements of all edges that are incident to some fixed node with paths of length 2, and replacements of paths of length 3 with inner nodes of degree 2 by edges (*contraction of an odd path*, the reversal of edge subdivision).

Subdivisions of edges and stars are intimately related to the class of K_4 inequalities, see Subsection 1.8.1. Namely, Barahona & Mahjoub [1994] have shown that all nontrivial facets of $P_I(G)$ for such a graph arise from a 4-clique (K_4) inequality by repeated applications of these operations. The 19 types of inequalities that one can produce in this way form the class of K_4 inequalities.

Joins. The operations that we term here *joins* compose a new graph from two or more given graphs in a way that involves an *identification* of parts of the original graphs. Join operations often have the appealing property that they can not only be used for composition, but also for decomposition purposes, because the identification component is left as a fingerprint in the composition. If we can recognize these traces, we can recursively set up a *decomposition tree* that contains structural information about a graph.

The composition/decomposition principle that we have just outlined is the basis for a *graph theoretic approach* to the set packing problem. The idea of this approach is to develop algorithms that work as follows: A given graph is recursively decomposed into "basic" components (i.e., components that can not be decomposed further), the set packing problem is solved for each component, and the individual solutions are composed into an overall solution by going the decomposition tree up again.

To develop such an *algorithm*, we need the following ingredients: A join operation, an (efficient) procedure that can construct the associated decomposition tree for a (large) class of graphs, a method to solve the set packing problems at the leafs of the decomposition tree, and a way to compose an optimal stable set in a join from optimal stable sets in component graphs. The last of these four tasks is where *polyhedral investigations* of joins come into play. Namely, if the join operation is such that one can construct a complete description for the set packing polytope of a join from complete descriptions for its components, and such descriptions are known at the leafs, then such a system can also be constructed for the root and used to solve the original set packing problem.

Our first example of a join composes from two graphs $G' = (V', E')$ and $G'' = (V'', E'')$ their *union* $G' \cup G'' := (V' \cup V'', E' \cup E'')$. When the *intersection* $G' \cap G'' := (V' \cap V'', E' \cap E'')$

of the component graphs is a clique, this union is called a *clique identification*. Looking at the procedure from a decomposition point of view, clique identification offers a decomposition opportunity whenever we can identify in a graph a *node separator* that is a clique.

1.8.8 Theorem (Clique Identification, Chvátal [1975])
Let $G' \cup G''$ be a clique identification of two graphs G' and G'', and let $A'x' \leq b'$ and $A''x'' \leq b''$ be complete descriptions of the set packing polytopes $P_I(G')$ and $P_I(G'')$. Then:

The union of the systems $A'x' \leq b'$ and $A''x'' \leq b''$ is a complete description of $P_I(G' \cup G'')$.

Unions of graphs that intersect on a *coedge*, i.e., on two nonadjacent nodes, were studied by Barahona & Mahjoub [1994]. As in the case of clique identification, the set packing polytopes of *coedge identifications* can also be described completely if such knowledge is available for the components. This technique can be used to decompose a graph that has a coedge node separator.

Coedge identification/decomposition bears on the derivation of complete descriptions of set packing polytopes that are associated to W_4 *free graphs*, i.e., graphs that do not have a subdivision of a 4-wheel as a minor. It is known that such graphs can be decomposed into a number of components where complete descriptions are known (among them subdivisions of K_4). The decomposition uses only three types of node separators: Node and edge separators (cliques of size one and two) and coedge separators. Using Chvátal [1975]'s result on complete descriptions for clique identifications in the first two and their own result in the coedge case, Barahona & Mahjoub [1994] construct a polynomial sized *complete extended description* of the set packing polytope of a general W_4 free graph $G = (V, E)$. Here, the term "extended description" refers to a system that defines a polytope P in a high dimensional space that can be projected into $\mathbb{R}^V$ to obtain $P_I(G)$; extended descriptions take advantage of the observation that a projection of a polytope can have more facets than the polytope itself.

The last type of join that we want to mention is the *amalgamation* of two graphs of Burlet & Fonlupt [1994]. This concept subsumes the graph theoretic operations of node substitution and clique identification; it characterizes the class of *Meyniel graphs*. Burlet & Fonlupt [1994] show that one can obtain a complete description of the set packing polytope of the amalgam from complete descriptions for the components.

1.8.3 Polyhedral Results on Claw Free Graphs

We have collected in this subsection some results about set packing polyhedra that are associated to claw free graphs. Most of this material fits into other subsections of this survey, but the extent of the topic and some unique aspects seemed to suggest that a treatment in one place would be more appropriate.

Claw is a synonym for $K_{1,3}$, see Figure 1.12, and a *claw free graph* is one that does not contain such a structure.

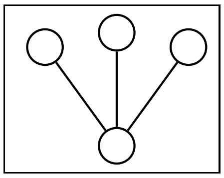


Figure 1.12: A Claw.

Claw free graphs stir the interest of the polyhedral community because the set packing problem for this class can be solved in polynomial time, see Minty [1980], but a complete polyhedral description is not known. The research objective is to determine this description.
Line graphs are claw free and it was initially suspected that the facets of the set packing polytopes of claw free graphs resemble the facets of the matching polytope and would not be too complicated; one early conjecture was, e.g., that the only coefficients on the left-hand side are 0, 1, and 2. Giles & Trotter [1979] were the first to point out that these polytopes are complex objects. They did not only prove the 0,1,2 conjecture false, but gave also examples of claw free graphs that produce complicated inequalities that contain, e.g., arbitrarily large coefficients. We have mentioned two such classes in Subsection 1.8.1: The compositions of circulants and the wedges (one can and must delete some edges in a wedge as defined in this thesis to make it claw free).
Some progress was made by Pulleyblank & Shepherd [1993] for a the more restrictive class of *distance claw free graphs*. These are graphs that do, for each node v, not only not contain a stable set of size 3 in the neighborhood of v, but they do also not contain such a stable set in the set of nodes that have distance 2 from v. Pulleyblank & Shepherd give a polynomial time (dynamic programming) algorithm for the set packing problem in distance claw free graphs and derive a polynomial sized complete extended description of the associated polytope.
Gallucio & Sassano [1993] take up the subject of general claw free graphs again and investigate the *rank facets* that are associated to such graphs. It turns out that there are only three types of rank facet *producing* claw free graphs: Cliques, line graphs of minimal 2-connected hypomatchable graphs, and the circulants $C(\alpha\omega+1,\omega)$. All rank facets can be obtained from these types either by sequential lifting or as *sums* of two such graphs.
We finally mention Euler & Le Verge [1996]'s list of complete descriptions of set packing polytopes of claw free graphs with up to 10 nodes.

1.8.4 Quadratic and Semidefinite Relaxations

Next to the search for facet defining and producing graphs, the study of *quadratic* and, intimately related, also of *semidefinite relaxations* is a second general technique to derive valid and facet defining inequalities for the set packing polytope. While the first concept has a *combinatorial* and in the first place *descriptive* flavour, the quadratic/semidefinite techniques are *algebraic* and, even better, *algorithmic* by their very nature, and they do not only apply to set packing, but to arbitrary 0/1 integer programs. And the method's wider scope and built-in separation machinery is not bought with a dilution of strength! Quite to the contrary, almost all of the explicitly known inequalities for set packing polytopes can be pinpointed in the quadratic/semidefinite setting as well and more: Superclasses of important types of constraints, most notably clique and antihole inequalities, can be separated in polynomial time. This implies, in particular, one of the most spectacular results in combinatorial optimization: The polynomial time solvability of the stable set problem in perfect graphs. There is only one price to pay for all of these achievements: The number of variables is squared.
We try to give in this subsection a *survey* over some basic aspects of quadratic and semidefinite techniques for the set packing problem. It goes without saying that we can not do more than scratching the surface of this fast developing field and we refer the reader to the book of Grötschel, Lovász & Schrijver [1988] and the article of Lovász & Schrijver [1991] and the references therein for a more comprehensive treatment. Our exposition is based on the latter publication and focusses on the special case of the set packing problem.

We start by introducing the concepts of a *quadratic relaxation*, and, as a particularly strong variant of such a model, of a *semidefinite relaxation* of the set packing problem in some graph $G = (V, E)$ with n nodes that will be numbered $V = \{1, \ldots, n\}$. The idea is to consider not the convex hull of the incidence vectors x of the stable sets in $\mathbb{R}^n$, but the convex hull of the *matrices* $xx^{\mathrm{T}} \in \mathbb{R}^{n \times n}$. We will see that this quadratic representation gives rise to two additional cut generation procedures that are not available in the linear case.
It is technically easier to study quadratic models that gives rise to *cones* instead of polyhedra and this is the reason to consider a *homogenization* of the set packing polytope that is constructed with the aid of an additional component x_0:

$$H_I(G) := \operatorname{cone}(\{1\} \times P_I) \subseteq \mathbb{R}^{\{0,\ldots,n\}}.$$

$P_I(G)$ can be retrieved from this object by an intersection with the hyperplane $x_0 = 1$. We introduce also a fractional relaxation of $H_I(G)$ that is obtained by replacing $P_I(G)$ with $P(G)$ and denoted by $H(G)$; we will assume here and elsewhere in this subsection that $P(G)$ is described (canonically) by the nonnegativity and the edge constraints (we assume also that there are no isolated nodes). We will work in this subsection only with the cone versions of the set packing polytope and its fractional relaxation and call $H_I(G)$ the *set packing cone* and $H(G)$ the *fractional set packing cone*. Going to quadratic space, we are interested in the *set packing matrix cone*

$$M_I(G) := \{xx^{\mathrm{T}} \in \mathbb{R}^{\{0,\ldots,n\} \times \{0,\ldots,n\}} \mid x \in H_I(G)\}.$$

The way to construct a *quadratic relaxation* of the set packing matrix cone is *not* just to replace $H_I(G)$ with $H(G)$ in the definition of the set packing matrix cone, which would yield a trivial quadratic relaxation. Instead, one can set up the following *stronger* relaxation.

$$\begin{array}{lll}
\text{(QSP)} & \text{(i)} & e_i^{\mathrm{T}} Y e_j = e_j^{\mathrm{T}} Y e_i & \forall i, j \\
& \text{(ii)} & e_i^{\mathrm{T}} Y e_0 = e_i^{\mathrm{T}} Y e_i & \forall i \\
& \text{(iii)} & u^{\mathrm{T}} Y e_i \geq 0 & \forall u \in H(G)^\circ,\ \forall i = 1, \ldots, n \\
& & u^{\mathrm{T}} Y f_i \geq 0 & \forall u \in H(G)^\circ,\ \forall i = 1, \ldots, n \\
& \text{(iv)} & Y \in \mathbb{R}^{\{0,\ldots,n\} \times \{0,\ldots,n\}}. &
\end{array}$$

Here, we denote by S° the *polar* of a set S and by f_i, $i = 1, \ldots, n$ the vectors of the form $f_i := e_0 - e_i$ (where e_i is the i-th unit vector). Their purpose is to serve as (the left-hand sides of) facets of the "homogenized unit cube" $U := \operatorname{cone}(\{1\} \times [0,1]^n)$, which contains $H_I(G)$. Associated to the system (QSP) is the *fractional set packing matrix cone $M(G)$* and *this* cone will serve as one relaxation of $M_I(G)$ in quadratic space.
Before we take a closer look at this object and at the system (QSP), let us quickly introduce another *infinite* set of linear inequalities, that strengthens the quadratic relaxation (QSP) to a *semidefinite relaxation* that we denote by (QSP$_+$).

$$\text{(v)} \quad u^{\mathrm{T}} Y u \geq 0 \qquad \forall u \in \mathbb{R}^{\{0,\ldots,n\}}$$

Associated to the system (QSP$_+$) is another fractional matrix cone $M_+(G)$.
(QSP) (i) states that the matrices that are solutions to the system (QSP) are symmetric, a property that surely holds for all 0/1 matrices xx^{T} with $x \in H_I(G) \cap \{0,1\}^{\{0,\ldots,n\}}$. (ii) states a type of "quadratic constraints": For matrices xx^T as above (ii) stipulates $x_i^2 = x_i$, $i = 0, \ldots, n$.

(iii) throws in what we know about $H(G)$. Again for matrices xx^{T}, we have that $u^{\mathrm{T}}x \geq 0$ for all left-hand sides $u \in H(G)^\circ$ of valid constraints for $H(G)$, and since $H(G) \subseteq U$, the same is true for the facets $v \in U^\circ$ of the homogenized unit cube U (which are exactly the vectors e_i and f_i), and this yields $u^{\mathrm{T}}xx^{\mathrm{T}}v \geq 0$. (iv) is the same as stating that the matrix Y is positive semidefinite, which clearly holds for all matrices of the form xx^{T}.
The quadratic constraints (QSP) (ii) and the semidefinite constraints (QSP$_+$) (v) are not available in the linear case and account for the greater strength of (QSP) and (QSP$_+$) in comparison to the trivial quadratic relaxation. One improvement is, e.g., the following. Consider the vector $u = -e_i - e_j + e_0 \in H(G)^\circ$ which is the left-hand side of the edge inequality $-x_i - x_j + x_0 \geq 0 \iff x_i + x_j \leq x_0$ for $H(G)$, and the vector $e_j \in U^\circ$ which is the left-hand side of the nonnegativity constraint $x_j \geq 0$ for U. Inserting these vectors in (iii) yields

$$u^{\mathrm{T}}Ye_j = (-e_i - e_j + e_0)^{\mathrm{T}}Y_{\cdot j} = -y_{ij} - y_{jj} + y_{0j} = -y_{ij} \geq 0,$$

and this implies $y_{ij} = 0$ for all $ij \in E$. This property does not hold for the trivial relaxation. But (QSP) as well as (QSP$_+$) are not only strong, they are also *algorithmically tractable*. In fact, (QSP) is of polynomial size and could be written down easily, a property that does not hold for (QSP$_+$), but one can solve the separation and the optimization problem for this system in polynomial time as well, see Grötschel, Lovász & Schrijver [1988].
Having these fine relaxations in quadratic space at hand, we go back to the original (homogenized) space by a simple *projection* to finally construct good relaxations of the (homogenized) set packing polytope, which inherit the superior descriptive and algorithmic properties of the matrix cones $M(G)$ and $M_+(G)$. These relaxations are the cones

$$\begin{aligned} N(G) &:= Me_0 &&= \{Ye_0 \in \mathbb{R}^{\{0,\ldots,n\}} \mid Y \in M(G)\} \\ N_+(G) &:= M_+e_0 &&= \{Ye_0 \in \mathbb{R}^{\{0,\ldots,n\}} \mid Y \in M_+(G)\} \end{aligned}$$

and any inequality that is valid for them is a *matrix inequality*. It follows once more from the general methodology of Grötschel, Lovász & Schrijver [1988] that the weak separation problem for matrix inequalities can be solved in polynomial time such that one can weakly optimize over $N(G)$ and $N_+(G)$ in polynomial time.

1.8.9 Theorem (N and N$_+$ Operator, Lovász & Schrijver [1991])
Let G be a graph, let $H_I(G)$ be the homogenization of its set packing polytope, and let $H(G)$ be the fractional (edge) relaxation of this homogenization. Then:

$$H_I(G) \subseteq N_+(G) \subseteq N(G) \subseteq H(G).$$

The weak separation and optimization problem for $N(G)$ and $N_+(G)$ can be solved in polynomial time.

We remark that the strong separation and optimization problems for $N(G)$ are also in $\mathcal{P}$.
One can now go one step further and iterate the construction of the matrix cones, obtaining tighter descriptions in every step. Inserting $N(G)$ into (QSP) (iii) in the place of $H(G)$, one obtains a second matrix cone $M^2(G)$, that is projected into $(n+1)$-space to become a cone $N^2(G)$, and so on; any valid inequality for such a relaxation $N^k(G)$ is called a *matrix inequality with N-index k*. Analogously, we can iterate the N_+-operator to obtain relaxations $N_+^k(G)$ and N_+ *matrix inequalities* for any natural index k. One can show that these relaxations get tighter and tighter, that one can solve the weak separation problem in polynomial time for any fixed k, and that the n-th relaxation coincides with $P_I(G)$.

1.8.10 Theorem (Iterated N and N_+ Operator, Lovász & Schrijver [1991])
Let $G = (V, E)$ be a graph with n nodes, let $H_I(G)$ be the homogenization of its set packing polytope, and let $H(G)$ be the fractional (edge) relaxation of this homogenization. Then:

$$
\begin{array}{lll}
(i) & N^{k+1}(G) \subseteq N^k(G) & \forall k \in \mathbb{N}. \\
(ii) & N_+^k(G) \subseteq N^k(G) & \forall k \in \mathbb{N}. \\
(iii) & N^n(G) = H_I(G). &
\end{array}
$$

The weak separation and optimization problem for $N^k(G)$ and $N_+^k(G)$ can be solved in polynomial time for every fixed natural number k.

Theorem 1.8.10 gives a wealth of polynomial time separable classes of inequalities for general set packing polytopes, namely, all matrix cuts with arbitrary but fixed N- or N_+-index. The *graphs* whose associated set packing polytopes can be described completely in this way are also said to have N- or N_+*-index k*. It turns out that a large number of graphs have bounded indices. We first state the results for the N-index.

1.8.11 Theorem (Graphs with Bounded N-Index, Lovász & Schrijver [1991])

(i) An odd cycle $C(2k+1, 2)$ has N-index 1.

(ii) A complete graph K_n has N-index $n-2$.

(iii) A perfect graph G has N-index $\omega(G) - 2$.

(iv) An odd antihole $C(2k+1, k)$ has N-index $2k$.

(v) A minimally imperfect graph G has N-index $\omega(G) - 1$.

It is more difficult to characterize graphs with bounded N_+-index, but (with an analogous definition) a number of *inequalities* are known to have small N_+-indices, in particular clique and odd antihole inequalities, which are hence contained in the polynomially separable superclass of matrix inequalities with N_+-index 1.

1.8.12 Theorem (Inequalities with Bounded N_+-Index, Lovász & Schrijver [1991])
Clique, odd cycle, wheel, and odd antihole inequalities have N_+-index 1.

As the *perfect graphs* are exactly those with perfect clique matrices, i.e., the graphs whose associated set packing polytopes can be described completely by means of clique inequalities, it follows that the set packing problem in perfect graphs can be solved in polynomial time, a spectacular result that was first proved by Grötschel, Lovász & Schrijver [1988].

1.8.13 Theorem (Set Packing Polytopes of Perfect Graphs, Grötschel, Lovász & Schrijver [1988], Lovász & Schrijver [1991])
Perfect graphs have N_+-index 1.

1.8.14 Theorem (Set Packing in Perfect Graphs, Grötschel, Lovász & Schrijver [1988])
The set packing problem in perfect graphs can be solved in polynomial time.

We finally relate the semidefinite relaxation $N_+(G)$ to the original approach of Grötschel, Lovász & Schrijver [1988]. They considered the semidefinite formulation (QSP_+), but with (iii) replaced by

$$(\mathrm{iii}') \quad e_i^{\mathrm{T}} Y e_j = 0 \qquad \forall ij \in E.$$

We denote this semidefinite system by (QSP'_+), and the associated matrix cone by $M'_+(G)$. The projection of this matrix cone into $(n+1)$-space yields an N'_+-cone, and this N'_+-cone's intersection with the hyperplane $x_0 = 1$ yields the convex, but in general not polyhedral set

$$\mathrm{TH}(G) := \big\{y \in \mathbb{R}^n \mid y_i = e_i^{\mathrm{T}} Y e_0,\ i = 1, \ldots, n,\ Y \in N'_+(G) \cap \{e_0^{\mathrm{T}} Y e_0 = 1\}\big\}.$$

Grötschel, Lovász & Schrijver [1988] have proved that $\mathrm{TH}(G)$ can be described completely by means of (nonnegativity and) *orthogonality inequalities*. Such an orthogonality inequality for a graph $G = (V, E)$ with nodes $V = \{1, \ldots, n\}$ involves an *orthonormal representation* of G, i.e., a set of vectors $v_i \in \mathbb{R}^n$ with $|v_i| = 1$, one for each node i of G, such that $v_i^{\mathrm{T}} v_j = 0$ holds for all $ij \notin E$, and an additional arbitrary vector $c \in \mathbb{R}^n$ with $|c| = 1$. The orthogonality inequality that corresponds to this data is

$$\sum_{i \in V} (c^{\mathrm{T}} v_i)^2 x_i \leq 1.$$

This class subsumes the clique inequalities by suitable choices of orthonormal representations.

1.8.15 Theorem (Orthogonality Inequalities, Grötschel, Lovász & Schrijver [1988])
For any graph G holds:

(i) Orthogonality inequalities can be separated in polynomial time.

(ii) $\mathrm{TH}(G)$ is completely described by nonnegativity and orthogonality inequalities.

(iii) G is perfect if and only if $\mathrm{TH}(G) = P_I(G)$.

1.8.16 Theorem (N_+-Index of Orthogonality Inequalities, Lovász & Schrijver [1991])
Orthogonality inequalities have N_+-index 1.

1.8.5 Adjacency

We summarize in this subsection some results on the *adjacency* of vertices of the set packing polytope and on the adjacency of integer vertices of its fractional relaxation. Such results bear on the development of *primal* algorithms for the set packing problem in a graph G.

For the purposes of this subsection, we can define a primal algorithm in terms of a *search graph* $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$, that has the set of all set packings of G as its nodes (and some set of edges). A primal algorithm uses the edges of $\mathfrak{G}$ to move from one set packing to another. In every move, the algorithm searches the neighbors of the current node for a set packing that has, with respect to some given objective, a better value than the current one; this neighborhood scan is called a *local search*. When an improving neighbor has been found, the algorithm moves there along an edge of the graph; this edge is an *improvement direction*. When there is no improvement direction, the algorithm is "trapped" in a *local optimum* and stops.

The *connection* between a primal algorithm of local search type and the adjacency relation on a set packing polytope $P_I(G)$ is that adjacency is a natural candidate to define the edge set of the search graph $\mathfrak{G}$. Namely, we let $\mathfrak{u}\mathfrak{v} \in \mathfrak{E}$ if and only if the incidence vectors of the set packings $\mathfrak{u}$ and $\mathfrak{v}$ are neighbors on $P_I(G)$, i.e., if they lie on a common 1-dimensional face of $P_I(G)$. Doing so produces a graph $\mathfrak{G}(P_I(G))$ which is called the *skeleton* of $P_I(G)$. Skeletons have a property that makes them attractive search graphs: Not only are they connected, but there is a path of improvement directions from any vertex to the global optimum.

Edmonds [1965] famous polynomial algorithm for set packing problems on *line graphs*, i.e., for matching problems, moves from one packing to another one by flipping nodes (in the line graph) of a connected structure that is called a *Hungarian tree*. For maximum cardinality

set packing problems on arbitrary graphs, Edmonds [1962] has derived a local optimality criterion that is also in terms of trees and characterizes all improvement directions: A set packing X in a graph $G = (V, E)$ is not of maximum cardinality if and only if the bipartite graph $(V, (X \times V \setminus X) \cap E)$ contains a tree $T = (W, F)$, such that $X \Delta W$ is a packing of larger cardinality than X. (Here, $X \Delta Y$ denotes the *symmetric difference* of two sets X and Y.) It follows from a result of Chvátal [1975] that we will state in a second, that Edmonds's matching algorithm does a local search on the skeleton of the set packing polytope that is associated to a line graph, and that his tree optimality criterion characterizes adjacent vertices in the skeleton of a general set packing polytope. In view of these similarities between the line graph/matching and the general case, it was hoped that matching like primal techniques could also be applied to general set packing problems. An attempt in this direction was undertaken by Nemhauser & Trotter [1975], who investigated Edmonds's criterion further and used it, supplemented with lower bounding LP techniques, for the development of a branch-and-bound algorithm that could solve maximum cardinality set packing problems on random graphs with up to 100 nodes.
The link between these results and polyhedral theory is the following result of Chvátal [1975] that characterizes the adjacent vertices of set packing polytopes completely. The theorem shows that the above mentioned optimality criteria characterize adjacency in the skeleton of the set packing polytope, and that the algorithms perform a local search in this structure.

1.8.17 Theorem (Adjacency on the Set Packing Polytope, Chvátal [1975])
Let $G = (V, E)$ be a graph, let $P_I(G)$ be the associated set packing polytope, and let x and y be the incidence vectors of two set packings X and Y in G, respectively. Then:

x and y are adjacent on $P_I(G)$ if and only if the graph $G[X \Delta Y]$ is connected.

Theorem 1.8.17 brings up the question if it is possible to use polyhedral information to perform a local search in the skeleton. One idea to do this was investigated in a series of papers by Balas & Padberg [1970, 1975, 1976] and is as follows. Consider the fractional set packing polytope $P(A)$ that is associated to a given 0/1 matrix A. Any vertex of $P_I(A)$ is also a vertex of $P(A)$, which means that it is possible to reach the integer optimum by searching through the skeleton $\mathfrak{G}(P(A))$ of the fractional relaxation. This is interesting because there is an effective and ready-to-use algorithm that does exactly this: The *simplex* method. In fact, nondegenerate pivots lead from one vertex to adjacent vertices, and, doing additional degenerate pivots, it is possible to reach from a given vertex all of its neighbors. In other words: The simplex algorithm performs a local search on the skeleton of the fractional set packing polytope with some additional degenerate steps. These statements were trivial, but point into an interesting direction: Is it perhaps possible to move with *all integer* pivots through the skeleton of the integer set packing polytope as well? It is, and $\mathfrak{G}(P_I(A))$ seems to have even extremely promising looking properties!

1.8.18 Theorem (Skeleton of Set Packing Polytopes, Balas & Padberg [1970, 1975])
Let A be an $m \times n$ 0/1 matrix, $P_I(A)$ the associated set packing polytope, $P(A)$ its fractional relaxation, and let $\mathfrak{G}_I := \mathfrak{G}(P_I(A))$ and $\mathfrak{G} := \mathfrak{G}(P(A))$ be the associated skeletons. Then:

(i) $\mathfrak{G}_I$ *is a subgraph of* $\mathfrak{G}$.
(ii) $\operatorname{diam} \mathfrak{G}_I \leq m/2$.
(iii) $|\delta(\mathfrak{v})| \geq n \quad \forall \mathfrak{v} \in V(\mathfrak{G}_I)$.

Here, $\operatorname{diam} \mathfrak{G}_I$ denotes the diameter of the graph $\mathfrak{G}_I$.

Theorem 1.8.18 (i) states that vertices are adjacent on $P_I(A)$ if and only if they are adjacent on $P(A)$, i.e., it is possible to reach the optimum integer solution by a sequence of all integer pivots. And not only is this possible: By (ii), one can reach the optimum from any given integer starting point in at most $m/2$ pivots! (The theorem makes only a statement about nondegenerate pivots, but one can sharpen this result.) We remark that Naddef [1989] has proved the *Hirsch conjecture* true for 0/1 polytopes; this can sometimes yield a smaller bound on the diameter than Theorem 1.8.18 (ii). Finally, (iii) hints to a difficulty: Each vertex has a very large number of neighbors, and this may render the local search difficult.
Balas & Padberg have developed and tested primal pivoting algorithms along these lines.

1.9 The Set Covering Polytope

We survey in this section *polyhedral* results on the set covering polytope $Q_I(A)$. Analogous to the set packing case, such investigations aim at the characterization of valid and facet defining inequalities and the development of methods to compute them efficiently. But the main *motivation* for this doing is different, namely, to unify polyhedral results that were obtained for various kinds of combinatorial optimization problems that can be stated as optimization problems over independence systems, a problem that is in a very direct way equivalent to the set covering problem. For example, the minimal cover inequalities of the knapsack polytope turn out to be so-called generalized clique inequalities, and the Möbius ladder inequalities of the acyclic subdigraph polytope can be seen as generalized cycle inequalities of an independence system/set covering polytope that is associated to an appropriately chosen independence system.
The *concepts* that guide these polyhedral investigations are essentially the same as in the set packing case: One considers facet defining submatrices of a given 0/1 matrix, tries to identify facet defining classes, and uses lifting procedures to make local constraints globally valid. The similarity in the approaches carries over to the *descriptive results*, and we will encounter familiar structures like cliques, cycles, etc. What misses in comparison to set packing are significant classes of polynomially solvable set covering problems, polynomially separable types of inequalities, and completely described cases. This *algorithmic* lack is apparently due to the ineffectiveness of graph theoretic approaches to set covering. In other words: The algorithmic theory of hypergraphs is way behind its graphical brother.
This section is organized as follows. The remainder of this introduction states some basic properties of the set covering polytope, most notably the relation to the independence system polytope. The only Subsection 1.9.1 gives a list of facet defining matrices and some results on rank facets.
The subsequent subsections resort to the following basic properties of the set covering polytope. Recall from Section 1.3 that a set covering problem with 0/1 matrix A is equivalent to an optimization problem over an *independence system* $\mathfrak{I}(A)$ via an affine transformation $y := \mathbf{1} - x$:

$$\begin{array}{lllcllll}
\text{(SCP)} & \min & w^{\mathrm{T}}(\mathbf{1}-y) & = w^{\mathrm{T}}\mathbf{1} - & \text{(ISP)} & \max & w^{\mathrm{T}}y \\
& & A(\mathbf{1}-y) \geq \mathbf{1} & & \text{(i)} & & Ay \leq (A-I)\mathbf{1} \\
& & (\mathbf{1}-y) \leq \mathbf{1} & & \text{(ii)} & & y \geq 0 \\
& & (\mathbf{1}-y) \geq 0 & & \text{(iii)} & & y \leq \mathbf{1} \\
& & (\mathbf{1}-y) \in \{0,1\}^n & & \text{(iv)} & & y \in \{0,1\}^n.
\end{array}$$

The independence system $\mathfrak{I}(A)$ has the set of column(indice)s of A as its ground set, and its *cycles* are the nonredundant rows of A.
The above relation implies that (SCP) and (ISP) are equivalent problems, and in a very direct way: x is a solution of (SCP) if and only of $\mathbf{1} - x$ is a solution of (ISP). This means in polyhedral terms that the associated polytopes satisfy

$$Q_I(A) = \mathbf{1} - P_{\text{ISP}}(A)$$

and we need to study only one of them. More precisely, we have the following.

1.9.1 Corollary (Set Covering and Independence System Polytope)
Let A be a 0/1 matrix and $Q_I(A)$ and $P_{\text{ISP}}(A)$ be the associated set covering and independence system polytope, respectively. Then:

$a^{\mathrm{T}}x \geq \alpha$ *is valid/a facet for* $Q_I(A) \iff a^{\mathrm{T}}x \leq a^{\mathrm{T}}\mathbf{1} - \alpha$ *is valid/a facet for* $P_{\text{ISP}}(A)$.

The *significance* of the set covering/independence system polytope for combinatorial optimization is that polyhedral results for $Q_I(A)/P_{\text{ISP}}(A)$ carry over to many combinatorial optimization problems. Namely, combinatorial optimization problems can often be interpreted as optimization problems over special independence systems and this means that their polytopes inherit all facets of the more general body. We will point out some relations of this type that have been observed in the literature next to the discussion of the corresponding classes of inequalities.
Some simple properties of the set covering polytope are collected in

1.9.2 Observation (Dimension, Up Monotonicity, Bounds, and Nonnegativity)
Let A be a 0/1 matrix that has at least 2 nonzero entries in each row and $Q_I(A)$ be the associated set covering polytope.

(i) $Q_I(A)$ *is full dimensional.*

(ii) $Q_I(A)$ *is up monotone, i.e.,* $x \in Q_I(A) \Longrightarrow y \in Q_I(A)$ *for all* $x \leq y \leq \mathbf{1}$.

(iii) The upper bound constraints $x_j \leq 1$ *induce facets of* $Q_I(A)$.

(iv) A nonnegativity constraint $x_j \geq 0$ *defines a facet of* $Q_I(A)$ *if and only if the minor* A'_j *that results from A by a contraction of column j has at least 2 nonzeros in each row.*

(v) If $a^{\mathrm{T}}x \geq \alpha$ *defines a facet of* $Q_I(A)$ *that is not one of the upper bound constraints* $x_j \leq 1$, *all coefficients of* $a^{\mathrm{T}}x \geq \alpha$ *are nonnegative.*

1.9.1 Facet Defining Matrices

The technique that is used in the literature to derive classes of valid and facet defining inequalities for the set covering polytope is the study of *submatrices* of a given $m \times n$ 0/1 matrix A, similar to the study of subgraphs of the conflict graph in the set packing case. Likewise, this approach is motivated by and related to the study of *minimally nonideal* matrix minors: The general theory of nonideal matrices guarantees the existence of certain “cores” of locally facet defining structures.
Let us get more precise. The derivation techniques for inequalities for set covering polytopes from *submatrices* are based on the up-monotonicity of $Q_I(A)$. Namely, if A_{IJ} is some arbitrary minor of A, where I is some set of row(indice)s of A, and J some set of column indices, and the nonnegative inequality $a^{\mathrm{T}}x \geq \alpha$ is valid for $Q_I(A_{I\cdot}) \cap \{x \in \mathbb{R}^n \mid x_j = 1 \quad \forall j \notin J\}$ and

has $a_j = 0$ for $j \notin J$, it is also valid for $Q_I(A)$. The simple extension technique that we have just described is not very satisfactory, but it points to the principle that substructures of A give rise to valid and facet defining inequalities. This motivates the concept of a *facet defining 0/1 matrix* in analogy to facet defining graphs for set packing problems: We say that the matrix A *defines* the facet $a^{\mathrm{T}}x \geq \alpha$ if this inequality is essential for $Q_I(A)$. A first research topic on set covering polytopes is now to undertake a classification of such facet defining matrices and the corresponding inequalities. The facet defining matrices will serve as candidates for minors of some given 0/1 matrix of interest. Having identified such a minor, we can set up a corresponding inequality and try to extend it to an inequality that is globally valid. The investigation of possibilities to do this extension in a systematic way leads to the study of *lifting* techniques. The lifting problems for set covering inequalities are slightly more complicated then in the set packing case, because one deals with additional columns and rows, but the general principle is the same; we refer the reader to Nemhauser & Trotter [1973], Sassano [1989], and Nobili & Sassano [1989] for examples of sequential and combinatorial simultaneous lifting procedures.
We give next a list of facet defining matrices for the set covering polytope.

Generalized Antiweb Inequalities, Laurent [1989], Sassano [1989]. For natural numbers $n \geq t \geq q$, a *generalized antiweb* $\mathcal{AW}(n,t,q)$ is a $n\binom{t-1}{q-1} \times n$ 0/1 matrix that has a row $\sum_{i \in Q} e_i^{\mathrm{T}}$ for each q-element subset Q of each set of t consecutive column indices $\{j, \ldots, j+t-1\}$ (indices taken modulo n), see Figure 1.13. Associated to this matrix is the *generalized antiweb inequality*

$$\sum_{j=1}^{n} x_j \geq \lceil n(t-q+1)/t \rceil.$$

$$\begin{pmatrix} 1 & 1 & . & . & . \\ 1 & . & 1 & . & . \\ . & 1 & 1 & . & . \\ . & 1 & . & 1 & . \\ . & . & 1 & 1 & . \\ . & . & 1 & . & 1 \\ . & . & . & 1 & 1 \\ 1 & . & . & 1 & . \\ 1 & . & . & . & 1 \\ . & 1 & . & . & 1 \end{pmatrix}$$

Figure 1.13: The Generalized Antiweb $\mathcal{AW}(5,3,2)$.

The generalized antiweb inequality is facet defining if and only if either $n = t$ or t does not divide $n(q-1)$, see Laurent [1989] and Sassano [1989].
Generalized antiwebs subsume a number of structures that have been investigated earlier: *Generalized cliques* ($n = t$) by Nemhauser & Trotter [1973], Sekiguchi [1983], and Euler, Jünger & Reinelt [1987], *generalized cycles* ($q = t$ and t does not divide n) by Sekiguchi [1983] and Euler, Jünger & Reinelt [1987], and *generalized antiholes*, ($n = qt + 1$) by Euler, Jünger & Reinelt [1987]. The last mentioned authors have also investigated some generalizations of their antiwebs, that arise from (i) duplicating columns of the matrix $\mathcal{AW}(n,t,q)$ any number of times and (ii) adding any number of additional columns with certain rather general properties like not having too many nonzero entries; see also Schulz [1996, Section 4.4] for some further extensions. They can show that these generalizations are also facet defining. An application to the independence system of acyclic arc sets in a complete digraph exhibits the classes of *k-fence* inequalities for the *acyclic subdigraph polytope* as generalized clique, and the *Möbius ladder* inequalities as generalized cycle inequalities, a further example is mentioned in Nobili & Sassano [1989]. Nemhauser & Trotter [1973] mention a relation to the *knapsack*

problem, where the class of *cover* inequalities turns out to correspond to the generalized clique inequalities of the associated independence system polytope, see also Padberg [1975b].
The antiwebs $\mathcal{AW}(2k+1,2,2)$, the *odd holes*, have been investigated further by Cornuéjols & Sassano [1989]. They study the effects of switching zeros in odd holes to ones and can completely characterize the cases where such operations do not destroy the validity and faceteness of the odd hole inequality.
Sassano [1989] and Nobili & Sassano [1989] give two further (and more complicated) classes of facet defining matrices that arise from certain operations on the antiwebs $\mathcal{AW}(n,q,q)$, one a lifting, the other a composition operation.

Generalized Web Inequalities, Sassano [1989]. Generalized webs are the complements of generalized antiwebs: For natural numbers $n \geq t \geq q$, the *generalized web* $\mathcal{W}(n,t,q)$ is a $\binom{n}{q} - n\binom{t-1}{q-1} \times n$ 0/1 matrix that has a row $\sum_{i \in Q} e_i^{\mathrm{T}}$ for each q-element subset Q of column indices such that Q is *not* contained in any of the sets $\{j, \ldots, j+t-1\}$ (indices taken modulo n) of t consecutive column indices, see Figure 1.14. Associated to such a web matrix is the *generalized web inequality*

$$\sum_{j=1}^{n} x_j \geq n - t,$$

$$\begin{pmatrix} 1 & . & . & 1 & . & . & . \\ 1 & . & . & . & 1 & . & . \\ . & 1 & . & . & 1 & . & . \\ . & 1 & . & . & . & 1 & . \\ . & . & 1 & . & . & 1 & . \\ . & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . & 1 \end{pmatrix}$$

Figure 1.14: The Generalized Web $\mathcal{W}(7,3,2)$.

which is facet defining if t does not divide n, see Nobili & Sassano [1989].

Further Inequalities. The inequalities that we have considered so far were all *rank inequalities*, i.e., they had all only 0/1 coefficients on their left-hand sides. We mention now two classes of facets with more general coefficients.
Nobili & Sassano [1989] have studied a class of inequalities from *compositions* of rank facets. Starting point is a matrix operation, the *complete bipartite composition*, that constructs from two 0/1 matrices A_1 and A_2 the new matrix

$$A_1 \circ A_2 := \begin{pmatrix} A_1 & E \\ E & A_2 \end{pmatrix}.$$

Here, E denotes a matrix with all one entries. Nobili & Sassano [1989] show that if the rank inequality $\mathbf{1}^{\mathrm{T}} x_1 \geq \alpha_1$ is valid for $Q_I(A_1)$ and the second rank inequality $\mathbf{1}^{\mathrm{T}} x_2 \geq \alpha_2$ is valid and, in addition, tight for $Q_I(A_2)$, the inequality

$$(\alpha_2 - 1)\mathbf{1}^{\mathrm{T}} x_1 + \mathbf{1}^{\mathrm{T}} x_2 \geq \alpha_2$$

defines a facet of $Q_I(A_1 \circ A_2)$.
Finally, we mention that Balas & Ng [1989a,b] have completely characterized those facets of the set covering polytope that have only coefficients of 0, 1, and 2 on the left-hand side.

*

A second branch of research on the set covering polytope is devoted to the study of *necessary* and *sufficient* conditions that make a valid inequality facet defining. Like in the set packing case, the literature focusses on the class of *rank inequalities.* To set up this class, consider an $m \times n$ 0/1 matrix A and define its *rank* as the number

$$\beta(A) := \min \ \mathbf{1}^{\mathrm{T}}x, \ Ax \geq \mathbf{1}, \ x \in \{0,1\}^n.$$

Then, the inequality

$$\sum_{j=1}^{n} x_j \geq \beta(A)$$

is the *rank inequality* that is associated to A. Rank inequalities are valid by definition, but there is no complete characterization of those matrices known that give rise to facet defining rank constraints. But a number of necessary and sufficient conditions have been derived that we survey next. It will turn out that deletion minors play an important role in this context, and, for the remainder of this subsection, we want to denote by $A_{\cdot J}$ the deletion minor of A that results from a deletion of all columns that are *not* contained in J, i.e., $A_{\cdot J}$ consists of the columns of A that have indices in J and those rows, whose support is contained entirely in J. This matrix is the "uncovered" part of A that remains when one sets all variables x_j, $j \notin J$, to 1.

The *necessary* conditions for a rank inequality to be facet defining can be given in terms of the notions of closedness and nonseparability. We say that a set J of column indices is *closed* if $\beta(A_{\cdot J\cup\{k\}}) > \beta(A_{\cdot J})$ holds for all columns $k \notin J$, i.e., if the addition of any k to J strictly increases the rank. J is *nonseparable* if $\beta(A_{\cdot J'}) + \beta(A_{\cdot J''}) < \beta(A_{\cdot J})$ holds for any partition $J = J' \uplus J''$ of J into sets J' and J'', i.e., a separation results in a loss of rank.

1.9.3 Observation (Necessary Conditions for Rank Facets)
Let A be an $m \times n$ 0/1 matrix, let J be any subset of column indices, and let $A_{\cdot J}$ be the minor that results from deleting from A the columns that are not in J. Then:

If the rank inequality $\sum_{j\in J} x_j \geq \beta(A)$ defines a facet of $Q_I(A)$, the set J must be closed and nonseparable.

There are some cases where the condition in Observation 1.9.3 is known to be also sufficient: When A is the circuit-node incidence matrix of a *matroid*, and when the independence system $\mathfrak{I}(A)$ that is associated to A is the set of solutions of a single knapsack problem, see Laurent [1989].

A *sufficient* condition for the faceteness of the rank inequality that is associated to an $m \times n$ 0/1 matrix A can be stated in terms of a *critical graph* $G = (V, E)$. This graph has the set of column(indice)s of A as its nodes and two nodes i and j are adjacent if and only if there exists a covering $\overline{x} \in Q_I(A) \cap \mathbb{Z}^n$ that satisfies $\mathbf{1}^{\mathrm{T}}\overline{x} = \beta(A)$ and such that the vector $\overline{x} - e_i + e_j$, which results from exchanging the elements i and j, is also a feasible covering.

1.9.4 Observation (Sufficient Condition for Rank Facets, Laurent [1989])
Let A be an $m \times n$ 0/1 matrix, let $Q_I(A)$ be the associated set covering polytope, and let G be the critical graph of A. Then:

If G is connected, the rank inequality $\sum_{j=1}^{n} x_j \geq \beta(A)$ defines a facet of $Q_I(A)$.

This observation generalizes a number of earlier results of Sekiguchi [1983], Euler, Jünger & Reinelt [1987], and Sassano [1989].

We close the discussion of rank inequalities for the set covering polytope with two approaches to the heuristic *separation* of cuts this type.

Rank Inequalities From K-Projection, Nobili & Sassano [1992]. Given an $m \times n$ 0/1 matrix A, a subset $J \subseteq \{1, \ldots, n\}$ of columns, its complement $\overline{J} = \{1, \ldots, n\} \setminus J$, and an integer $k \in \mathbb{Z}_+$, a *k-projection* of A with respect to J is a 0/1 matrix $A|_{x(J)=k}$ with $n - |J| = |\overline{J}|$ columns and the property that any of its covers can be extended to a cover of A that contains exactly k columns from the set J. This matrix is unique up to permutation of rows; in fact, $A|_{x(J)=k} = \operatorname{bl}\big((\operatorname{bl} A|_{A_{i\cdot}(J)=k})_{\cdot\overline{J}}\big)$, where $\operatorname{bl} A|_{A_{i\cdot}(J)=k}$ is the submatrix of $\operatorname{bl} A$ that has as its rows all the covers of A that contain exactly k columns from J. One can prove that $Q_I(A|_{x(J)=k})$ is the orthogonal projection of the "equality constrained" set covering polytope $\operatorname{conv}\{x \in \{0,1\}^n : Ax \geq \mathbf{1},\ x(J) = k\}$ onto the subspace $\mathbb{R}^{\overline{J}}$, hence the name k-projection. The operation has the property that $\beta(A) \leq \beta(A|_{x(J)=k}) + k$.
Under special circumstances, k-projections can be used to construct rank inequalities. Namely, suppose that the equation $\beta(A) = \beta(A|_{x(J)=k}) + k$ holds such that A is *k-projectable* with respect to J, as we say. In this case, we can write the rank inequality associated to A as

$$\sum_{j=1}^{n} x_j \geq \beta(A|_{x(J)=k}) + k = \beta(A),$$

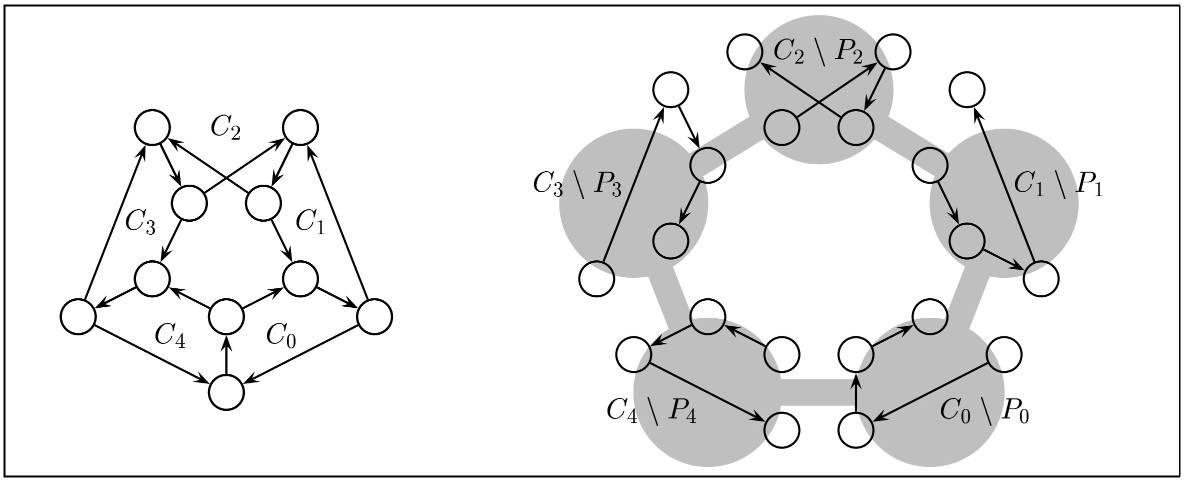
i.e., we can construct it from the rank inequality for $A|_{x(J)=k}$ which is simpler in the sense that it has a smaller right-hand side.
Nobili & Sassano [1992] suggest a *separation heuristic* for rank inequalities that is based on the iterative application of k-projections. They focus on the simplest case where $k = 1$ and J is the support of a row of the original matrix A, i.e., they always project with respect to one of the equations $A_{i\cdot}x = 1$. Projectability is established using two exponential sufficient criteria which are checked in a heuristic way. As the construction of the 1-projections is exponential as well, the authors resort to heuristically chosen submatrices at the cost of a weakening of the right-hand side. Projection is continued until the resulting matrix becomes so small that the covering number can be determined exactly. The separation routine, augmented by a clever lifting heuristic, has been successfully used in a branch-and-cut code for the solution of set covering problems from a library of randomly generated instances from the literature with several hundred rows and columns.

Conditional Cuts, Balas [1980], Balas & Ho [1980]. The cutting planes that we consider in this paragraph are special in the sense that they can very well (and are indeed supposed to) cut off parts of the set covering polytope under investigation: They are only valid under the condition that a solution that is better than the best currently known one exists, hence the name *conditional cut*.
A more precise description is the following. Suppose that an upper bound z_u on the optimum objective value of the set covering problem (SCP) is known and consider an arbitrary family $\mathfrak{W} \subseteq 2^n$ of column index sets $\mathfrak{v}$. If we can ensure that the disjunction $\bigvee_{\mathfrak{v}\in\mathfrak{W}} x(\mathfrak{v}) = 0$ holds for any solution x *with a better objective value than* z_u, the inequality

$$\sum_{j \in \bigcup_{\mathfrak{v}\in\mathfrak{W}} \operatorname{supp} A_{r(\mathfrak{v})\cdot} \setminus \mathfrak{v}} x_j \geq 1$$

is valid for all $x \in Q_I(A)$ such that $c^{\mathrm{T}}x < z_u$ and can be used as a cutting plane. Here, for each column set $\mathfrak{v}$, $A_{r(\mathfrak{v})\cdot}$ is an arbitrary row of A.

Conditional cuts are of "rank type"; the concept subsumes a number of earlier classes such as the *ordinal cuts* of Bowman & Starr [1973] and Bellmore & Ratliff [1971]'s cuts from *involutory bases*. Balas & Ho [1980] suggest a *separation heuristic* for conditional cuts that is based on LP duality arguments; the procedure has been applied with success in a branch-and-cut algorithm to set covering problems with up to 200 rows and 2,000 columns.



Chapter 2

Set Packing Relaxations

Summary. This chapter is about *set packing relaxations* of combinatorial optimization problems associated with acyclic digraphs and linear orderings, cuts and multicuts, multiple knapsacks, set coverings, and node packings themselves. Families of inequalities that are valid for such a relaxation and the associated separation routines carry over to the problems under investigation.

Acknowledgement. The results of this chapter are joint work with Robert Weismantel[1].

2.1 Introduction

This chapter is about *relaxations* of some combinatorial optimization problems in the form of a set packing problem and the use of such relaxations in a polyhedral approach.
Set packing problems are among the best studied combinatorial optimization problems with a beautiful theory connecting this area of research to Fulkerson's anti-blocking theory, the theory of perfect graphs, perfect and balanced matrices, semidefinite programming, and other fields, see the previous Chapter 1 of this thesis for a survey. Likewise, the *set packing polytope*, i.e., the convex hull of all node packings of a graph, plays a prominent role in polyhedral combinatorics not only because large classes of (facet defining) inequalities are known. Perhaps even more important, many of them can be separated in polynomial time, in particular odd cycle and orthogonality constraints, see Grötschel, Lovász & Schrijver [1988] and Lovász & Schrijver [1991].

[1]Otto-von-Guericke Universität Magdeburg, Fakultät für Mathematik, Institut für Mathematische Optimierung, Universitätsplatz 2, 39106 Magdeburg, Email robert.weismantel@mathematik.uni-magdeburg.de

Our aim in this chapter is to transfer some of these results to other combinatorial optimization problems. We show that the acyclic subdigraph and the linear ordering problem, the max cut, the k-multicut, and the clique partitioning problem, the multiple knapsack problem, the set covering problem, and the set packing problem itself have interesting *combinatorial relaxations* in form of a set packing problem. Families of inequalities that are valid for these relaxations and the associated separation routines carry over to the problems under investigation. The procedure is an application of a more general method to construct relaxations of combinatorial optimization problems by means of *affine transformations*.
The chapter contains seven sections in addition to this introduction. Section 2.2 describes our method to construct set packing relaxations. Section 2.3 is devoted to a study of the acyclic subdigraph and the linear ordering problem, see Grötschel, Jünger & Reinelt [1985a,b]. A main result here is that a class of Möbius ladders with dicycles of *arbitrary* lengths belongs to a (larger) class of odd cycles of an appropriate set packing relaxation; this superclass is polynomial time separable. Section 2.4 deals with set packing relaxations of the clique partitioning, the k-multicut, and the max cut problem, see Grötschel & Wakabayashi [1990] and Deza & Laurent [1997]. We introduce two types of "inequalities from odd cycles of triangles". The first of these classes contains the 2-chorded cycle inequalities, the second is related to circulant inequalities. Section 2.5 treats the set packing problem *itself*. We show, in particular, that the wheel inequalities of Barahona & Mahjoub [1994] and Cheng & Cunningham [1997] are odd cycle inequalities of a suitable set packing relaxation. We also introduce a new family of facet defining inequalities for the set packing polytope, the "cycle of cycles" inequalities. This class can be separated in polynomial time. Section 2.6 deals with the set covering problem. Again, we suggest a set packing relaxation in order to derive polynomially separable inequalities. We have *implemented* one version of such a separation procedure for use in a branch-and-cut code for set partitioning problems. Implementation details and computational experiences are reported in Section 3.3 of this thesis. Section 2.7 considers applications to the multiple knapsack problem, see Martello & Toth [1990] and Ferreira, Martin & Weismantel [1996]. The final Section 2.8 relates some results of the literature on set packing relaxations for 0/1 integer programming problems with nonnegative constraint matrices to our setting.
The following sections resort to some additional *notation* and two well known results for the *set packing* or *stable set problem*. Let

$$\text{(SSP)}\quad \begin{array}{rcl} \max\ w^{\mathrm{T}}x & & \\ Ax & \leq & \mathbf{1} \\ x & \in & \{0,1\}^V \end{array}$$

be an integer programming formulation of a set packing problem on a graph $G=(V,E)$ with nonnegative integer node weights $w \in \mathbb{Z}_+^V$, where $A = A(G) \in \{0,1\}^{E\times V}$ is the edge-node incidence matrix of G. Associated to (SSP) is the *set packing* or *stable set polytope* that we denote in this section by

$$P_{\mathrm{SSP}} = \mathrm{conv}\left\{\chi^S : S \text{ is a stable set in } G\right\} = \mathrm{conv}\left\{x \in \{0,1\}^n : Ax \leq \mathbf{1}\right\}$$

or, where convenient, also by $P_{\mathrm{SSP}}(G)$. For reasons that will become clear in the next section, we will actually not work with the stable set polytope P_{SSP} itself, but with its *anti-dominant*

$$\breve{P}_{\mathrm{SSP}} := P_{\mathrm{SSP}} - \mathbb{R}_+^V = \left\{x \in \mathbb{R}^V : \exists y \in P_{\mathrm{SSP}} : x \leq y\right\}.$$

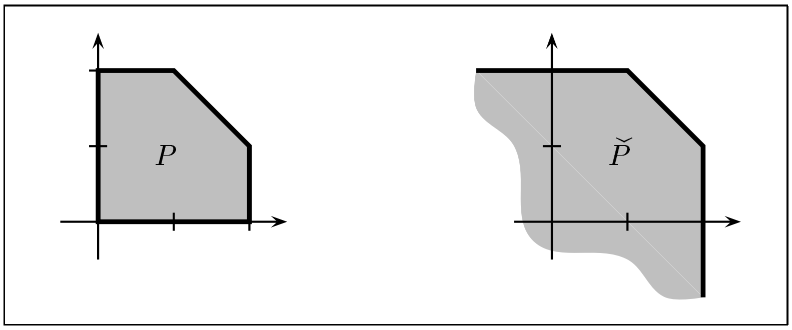


Figure 2.1: A Polyhedron and Its Anti-Dominant.

This construction allows to include vectors with arbitrary negative coordinates *without* destroying the polyhedral structure of P_{SSP}: Obviously, the valid inequalities for $\breve{P}_{\text{SSP}}$ are exactly the valid inequalities for P_{SSP} *with nonnegative coefficients*. Since the stable set polytope P_{SSP} is down monotone, its nontrivial constraints all have nonnegative coefficients, and we can thus work with $\breve{P}_{\text{SSP}}$ as well as with P_{SSP}. Figure 2.1 gives an illustration of a polytope and its anti-dominant.
We will need two results about $\breve{P}_{\text{SSP}}$ that are summarized in the following two theorems.

2.1.1 Theorem (Edge, Clique, and Odd Cycle Inequalities, Padberg [1973a])
Let $G = (V, E)$ be a graph and let $\breve{P}_{\text{SSP}}$ be the anti-dominant of the associated set packing polytope.

(i) If ij is an edge in G, the edge inequality $x_i + x_j \le 1$ *is valid for $\breve{P}_{\text{SSP}}$.*

(ii) If Q is a clique in G, the clique inequality

$$\sum_{i \in Q} x_i \le 1$$

is valid for $\breve{P}_{\text{SSP}}$; it is facet defining for $\breve{P}_{\text{SSP}}$ if and only if Q is a maximal clique (with respect to set inclusion).

(iii) If $C \subseteq V$ is the node set of an odd cycle in G, the odd cycle inequality

$$\sum_{i \in C} x_i \le (|C| - 1)/2$$

is valid for $\breve{P}_{\text{SSP}}$.

Separation of clique inequalities is $\mathcal{NP}$-hard, see Garey & Johnson [1979, Problem GT 19]. But the clique inequalities are contained in the class of *orthogonality inequalities*, see Grötschel, Lovász & Schrijver [1988], that can be separated in polynomial time. Odd cycle inequalities are polynomial time separable, see again Grötschel, Lovász & Schrijver [1988, Lemma 9.1.11].

2.1.2 Theorem (Orthogonality & Cycle Inequalities, Grötschel et al. [1988])
Let $G = (V, E)$ be a graph, $\breve{P}_{\text{SSP}}$ the anti-dominant of the associated set packing polytope, and $x \in \mathbb{Q}^V$. Suppose that $x_i + x_j \le 1$ holds for all edges $ij \in E$. Then:

(i) Orthogonality inequalities for $\breve{P}_{\text{SSP}}$ can be separated in polynomial time.

(ii) Odd cycle inequalities for $\breve{P}_{\text{SSP}}$ can be separated in polynomial time.

2.2 The Construction

Our aim in this section is to describe a *method to construct set packing relaxations* of combinatorial optimization problems. The setting is as follows. We are interested in some combinatorial optimization problem that is given by an integer programming formulation

$$\text{(IP)} \quad \max\ w^{\mathrm{T}}x \quad Ax \leq b,\ x \in \mathbb{Z}^n.$$

Here, $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$, and $w \in \mathbb{Z}^n$ are an integral matrix and vectors, respectively. The associated fractional and integer polyhedra are

$$P(A,b) := \{x \in \mathbb{R}^n \mid Ax \leq b\} \quad \text{and} \quad P_{\mathrm{IP}}(A,b) := \operatorname{conv}\{x \in \mathbb{Z}^n \mid Ax \leq b\}.$$

If the meaning is clear, we also write P for $P(A,b)$ and P_{IP} for $P_{\mathrm{IP}}(A,b)$.
Our method starts with an *affine function*

$$\pi : \mathbb{R}^n \to \mathbb{R}^{\overline{n}},\ x \mapsto \Pi x - \pi^0$$

given by a rational matrix $\Pi \in \mathbb{Q}^{\overline{n}\times n}$ and vector $\pi^0 \in \mathbb{Q}^{\overline{n}}$; note that the image space can be of *higher* dimension than the preimage. We call such functions *aggregation schemes* or simply *schemes*. A scheme is *integer* if it maps integer points to integer points, i.e., $\pi(\mathbb{Z}^n) \subseteq \mathbb{Z}^{\overline{n}}$, or, equivalently, if Π and π^0 are both integer, i.e., $\Pi \in \mathbb{Z}^{\overline{n}\times n}$ and $\pi \in \mathbb{Z}^{\overline{n}}$. Finally, the image $\pi(P)$ of a polyhedron P under the scheme π is called the *π-aggregate* or, if there is no danger of confusion, simply the *aggregate* of P.
Our motivation for studying aggregations is that they give rise to *valid inequalities* for some polyhedron P of interest. Namely, if $\overline{a}^{\mathrm{T}}\overline{x} \leq \overline{\alpha}$ is valid for an aggregate $\pi(P)$, the *expansion*

$$\overline{a}^{\mathrm{T}}\pi(x) \leq \overline{\alpha} \iff \overline{a}^{\mathrm{T}}\Pi x \leq \overline{\alpha} + \overline{a}^{\mathrm{T}}\pi^0$$

of this inequality is a constraint in $\mathbb{R}^n$ which is valid for the original polyhedron P.
The facial structure of an aggregate is, of course, in general as complicated as that of the original polyhedron. But it is often possible to find a *relaxation*

$$\overline{P} \supseteq \pi(P)$$

of the aggregate $\pi(P)$ that is of a well studied type. In this case, one can resort to known inequalities for the relaxation $\overline{P}$ to get an approximate description of the aggregate $\pi(P)$ and, via expansion, a description of a polyhedral relaxation of the original polyhedron P.
The crucial points in this procedure are the choice of the aggregation scheme and the identification of a suitable relaxation. The subsequent sections resort to the following method to construct *set packing relaxations*. Starting point is the observation that we are interested in *combinatorial programs*, i.e., 0/1 optimization problems (IP). Associated to such programs are *integer* polyhedra $P = P_{\mathrm{IP}}$. Restricting attention to likewise *integer* schemes (i.e., $\pi(\mathbb{Z}^n) \subseteq \mathbb{Z}^{\overline{n}}$, recall the above definition), the resulting aggregates are integer as well:

$$\pi(P_{\mathrm{IP}})_{\mathrm{IP}} = \pi(P_{\mathrm{IP}}) \qquad \forall \text{ (integer) } P_{\mathrm{IP}} \text{ and integer } \pi.$$

The next step is to construct a *conflict graph* $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$. To do this, we need a scheme that is bounded from above by one on the polyhedron P_{IP} of interest, i.e.,

$$\pi(x) \leq \mathbf{1} \qquad \forall x \in P_{\mathrm{IP}}.$$

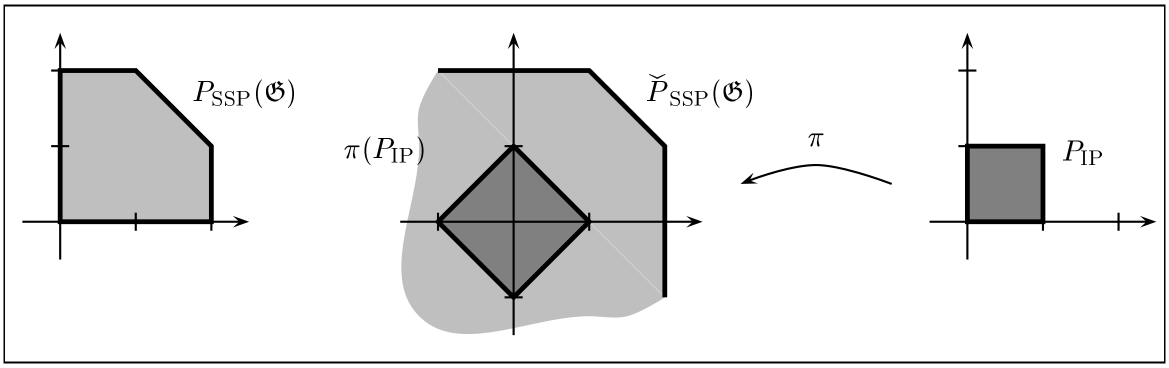


Figure 2.2: Constructing a Set Packing Relaxation.

Such schemes give rise to a conflict graph as follows. $\mathfrak{G}$ has a node for every component of the scheme, i.e., $\mathfrak{V} = \{1, \ldots, \overline{n}\}$. We draw an edge $\mathfrak{u}\mathfrak{v}$ between two nodes if π can not attain its maximum value of one in both components simultaneously:

$$\mathfrak{u}\mathfrak{v} \in \mathfrak{E} :\iff \pi_{\mathfrak{u}}(x) + \pi_{\mathfrak{v}}(x) \leq 1 \qquad \forall x \in P_{\mathrm{IP}}.$$

In this case, we say that $\mathfrak{u}$ and $\mathfrak{v}$ are *in conflict*. The anti-dominant $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ of the conflict graph is now a *set packing relaxation* of the π-aggregate $\pi(P_{\mathrm{IP}})$ in the sense that

$$\breve{P}_{\mathrm{SSP}}(\mathfrak{G}) \supseteq \pi(P_{\mathrm{IP}}).$$

Note that it is not possible to replace $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ with $P_{\mathrm{SSP}}(\mathfrak{G})$, because the scheme π can attain negative values, see Figure 2.2 for an illustration.
Once the set packing relaxation $\breve{P}_{\mathrm{SSP}}(\mathfrak{G}) \supseteq \pi(P)$ is found, inequalities and *separation* routines for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ carry over to the polyhedron P of interest. Given some point x to be tested for membership in P_{IP}, we simply (i) compute $\pi(x)$, (ii) solve the separation problem for $\pi(x)$ and $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$, and, if a separating hyperplane $\overline{a}^{\mathrm{T}}\overline{x} \leq \overline{\alpha}$ has been found, (iii) expand it. If all of these three steps are polynomial, this yields a polynomial time separation algorithm for a class of valid inequalities for P_{IP}, namely, for the expansions of all polynomial time separable and polynomial time expandable inequalities of $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$. Promising candidates for this are, in particular, the odd cycle and orthogonality constraints for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$.
The following sections present *examples* of set packing relaxations for a variety of combinatorial optimization problems.
We remark that for convenience of *notation*, we will occasionally consider paths, cycles, dipaths, dicycles, etc. as *sets* of nodes, edges, or arcs, and we will denote edges *as well as* arcs with the symbols ij and (i, j); the latter symbol will be used in cases like $(i, i+1)$.

2.3 The Acyclic Subdigraph and the Linear Ordering Problem

Our aim in this section is to construct a set packing relaxation of the acyclic subdigraph and the linear ordering problem in a space of exponential dimension. It will turn out that clique and odd cycle inequalities of this relaxation give rise to (and generalize) several classes of inequalities for the acyclic subdigraph and the linear ordering problem, namely, fence and Möbius ladder inequalities. We suggest Grötschel, Jünger & Reinelt [1985a] as a reference for the ASP, see also Goemans & Hall [1996, and references therein] for a recent study of known classes of inequalities, and Grötschel, Jünger & Reinelt [1985b] for the LOP.

The acyclic subdigraph and the linear ordering problem involve a complete digraph $D_n = (V, A)$ on n nodes with integer weights w_a on its arcs $a \in A$. An *acyclic arc set* in A contains no dicycle. The *acyclic subdigraph problem* (ASP) asks for an acyclic arc set with maximum weight on its arcs. Acyclic arc sets that contain, for any pair of nodes i and j, either the arc ij or the arc ji, are called *tournaments*. The *linear ordering problem* (LOP) is to find a tournament of maximum weight. IP formulations for the ASP and the LOP read as follows:

$$\begin{array}{lrll}
\max & \sum_{ij \in A} w_{ij} x_{ij} & & \\
\text{(ii)} & \sum_{ij \in C} x_{ij} & \le |C| - 1 & \forall \text{ dicycles } C \subseteq A \\
\text{(iii)} & x & \in \{0,1\}^A & \\
\end{array} \qquad \text{(ASP)}$$

$$\begin{array}{lrll}
\max & \sum_{ij \in A} w_{ij} x_{ij} & & \\
\text{(i)} & x_{ij} + x_{ji} & = 1 & \forall i, j \in V, i \ne j \\
\text{(ii)} & \sum_{ij \in C} x_{ij} & \le |C| - 1 & \forall \text{ dicycles } C \subseteq A \\
\text{(iii)} & x & \in \{0,1\}^A. & \\
\end{array} \qquad \text{(LOP)}$$

(ASP) is a relaxation of (LOP) and, what is more, the *linear ordering polytope* P_{LOP} is a face of the *acyclic subdigraph polytope* P_{ASP}. In particular, all inequalities that are valid for P_{ASP} are also valid for P_{LOP}. Two such classes of inequalities for both the ASP and the LOP are the *k-fence* and the *Möbius ladder inequalities*, see Grötschel, Jünger & Reinelt [1985a].

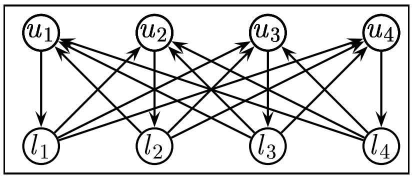


Figure 2.3: A 4-Fence.

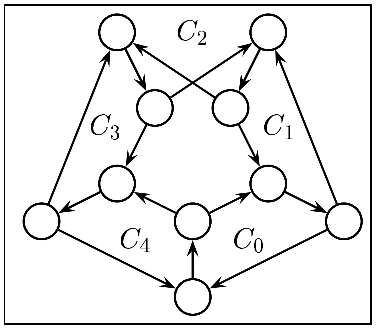


Figure 2.4: A Möbius Ladder of 5 Dicycles.

A (simple) *k-fence* involves two disjoint sets of "upper" and "lower" nodes $\{u_1, \ldots, u_k\}$ and $\{l_1, \ldots, l_k\}$ that are joined by a set of k *pales* $\{u_1 l_1, \ldots, u_k l_k\}$. All pales are oriented "downward". The k-fence is completed by adding all "upward" *pickets* $l_i u_j$ with the exception of the antiparallel pales. We remark that one can also allow that pales and pickets consist not only of a single arc, but of an entire dipath, thereby obtaining a larger class of general k-fences; for simplicity of exposition, however, we want to restrict ourselves here and elsewhere in this section to simple fences. Figure 2.3 shows a (simple) 4-fence.

A *Möbius ladder* consists of an odd number $2k+1$ of dicycles $C_0, \ldots, C_{2k}$ such that C_i and C_{i+1} (indices taken modulo $2k+1$) have a dipath P_i in common, see Figure 2.4; this time, we want to consider also the non simple case.

Fences and Möbius ladders give rise to valid inequalities for P_{ASP}: For a k-fence F_k and a Möbius ladder M of $2k+1$ dicycles we have

$$\sum_{ij \in F_k} x_{ij} \le k^2 - k + 1 \qquad \text{and} \qquad \sum_{i=0}^{2k} \sum_{ij \in C_i \setminus P_i} x_{ij} \le \left(\sum_{i=0}^{2k} |C_i \setminus P_i| \right) - (k+1).$$

Note that a Möbius ladder inequality as above has coefficients larger than one if an arc is contained in more than one of the dipaths $C_i \setminus P_i$. This is different from Grötschel, Jünger & Reinelt [1985a]'s (original) definition, where the coefficients take only values of zero and one and the Möbius ladder must meet a number of additional technical requirements to support a valid inequality. The two definitions of a Möbius ladder inequality coincide if and only if no two dipaths $C_i \setminus P_i$ have an arc in common (Möbius ladder without *arc repetition*).

We will show now that fences and Möbius ladders are cliques and odd cycles, respectively, in an (exponential) *conflict graph* $\mathfrak{G}(D_n) = (\mathfrak{V}, \mathfrak{E})$. $\mathfrak{G}$ has the set of all acyclic arc sets of D_n as its nodes. We draw an edge $\mathfrak{u}\mathfrak{v}$ between two acyclic arc-set nodes $\mathfrak{u}$ and $\mathfrak{v}$ if their union contains a dicycle. In this case, we say that $\mathfrak{u}$ *and* $\mathfrak{v}$ *are in conflict* because they can not be simultaneously contained in (the support of) a solution to (ASP).

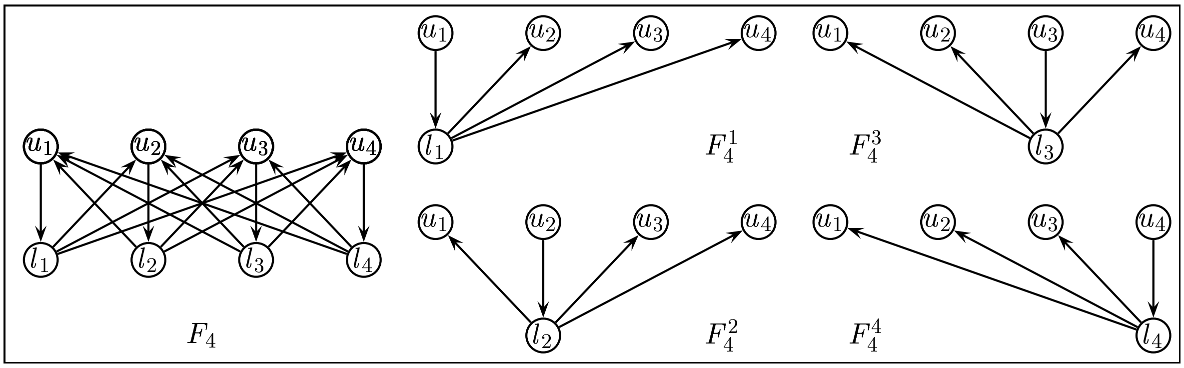


Figure 2.5: A Fence Clique.

It is now easy to identify the fences and Möbius ladders with cliques and odd cycles of $\mathfrak{G}$. To obtain a k-fence F_k, we look at the k acyclic arc sets F_k^i that consist of a pale $u_i l_i$ and the pickets $l_i u_j$ that go up from l_i for $i = 1, \ldots, k$. Any two such configurations F_k^i and F_k^j, $i \neq j$, are in conflict (they contain a dicycle). Hence, all of them together form a clique. Figure 2.5 illustrates this construction. Likewise, the Möbius ladders correspond to odd cycles of conflicting dipaths, namely, the dipaths $C_i \setminus P_i$, see Figure 2.6.

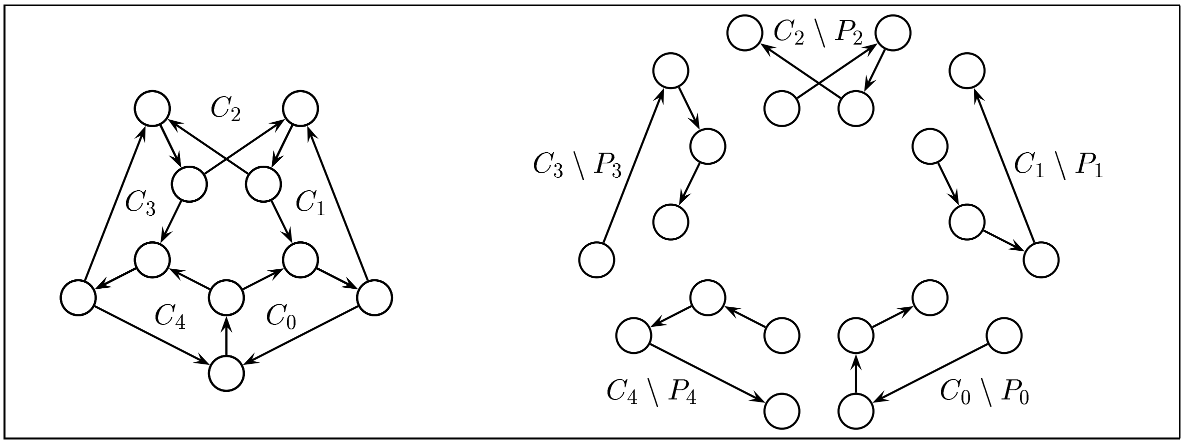


Figure 2.6: A Möbius Cycle of Dipaths.

The next step to obtain the fence and the Möbius ladder inequalities from the clique and odd cycle inequalities of the (anti-dominant of the) set packing polytope $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ associated to the conflict graph $\mathfrak{G}$ is to construct a set packing relaxation of the ASP. To this purpose, consider the aggregation scheme $\pi : \mathbb{R}^A \to \mathbb{R}^{\mathfrak{V}}$ defined as

$$\pi_{\mathfrak{v}}(x) := \sum_{ij\in\mathfrak{v}} x_{ij} - (|\mathfrak{v}| - 1) \qquad \forall \text{ acyclic arc sets } \mathfrak{v} \in \mathfrak{V}.$$

$\pi(x)$ is integral for all integral $x \in \mathbb{R}^A$. Moreover, for every incidence vector $x \in P_{\mathrm{ASP}}$ of an acyclic arc set $\mathrm{supp}(x) \subseteq A$ in D_n, we have that $\pi(x)$ attains its maximum value of one in component $\pi_{\mathfrak{v}}(x)$ if and only if $\mathfrak{v}$ is contained in $\mathrm{supp}(x)$. Since two conflicting acyclic arc sets can not simultaneously be contained in $\mathrm{supp}(x)$, we have that

$$\mathfrak{u}\mathfrak{v} \in \mathfrak{E} \iff \pi_{\mathfrak{u}}(x) + \pi_{\mathfrak{v}}(x) \le 1 \qquad \forall x \in P_{\mathrm{ASP}} \cap \mathbb{Z}^A$$

and, by convexity, also for all $x \in P_{\mathrm{ASP}}$. This argument proves that $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ is a set packing relaxation of P_{ASP}.

2.3.1 Lemma (Set Packing Relaxation of the ASP) $\pi\left(P_{\mathrm{ASP}}\right) \subseteq \breve{P}_{\mathrm{SSP}}\big(\mathfrak{G}(D_n)\big)$.

Note that it is not possible to replace $\breve{P}_{\mathrm{SSP}}$ with P_{SSP}, because the components of π can take negative values. More precisely, $\pi(x)$ is in general *not* the incidence vector of a stable set in $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$, but $\max\,\{0, \pi(x)\}$ is, with the maximum taken in every component (recall Figure 2.2).
Lemma 2.3.1 allows us to *expand* (see the definition on page 54) an inequality $\overline{a}^T\overline{x} \le \overline{\alpha}$ that is valid for $\breve{P}_{\mathrm{SSP}}$ into the inequality $\overline{a}^T\pi(x) \le \overline{\alpha}$ for P_{ASP}. Our next theorem states that the fence and Möbius ladder inequalities are expansions of clique and odd cycle inequalities, respectively.

2.3.2 Theorem (Fence and Möbius Ladder Inequalities)
Let D_n be the complete digraph on n nodes, P_{ASP} the corresponding acyclic subdigraph polytope, $\mathfrak{G}$ the conflict graph associated to D_n, and $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ the set packing relaxation of P_{ASP}.

(i) Every k-fence inequality for P_{ASP} is the expansion of a clique inequality for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$.

(ii) Every Möbius ladder inequality for P_{ASP} is the expansion of an odd cycle inequality for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$.

Proof.
(i) Let F_k be a k-fence. The acyclic arc sets F_k^i, $i = 1, \ldots, k$, defined on the previous page, form a clique in $\mathfrak{G}$, see the discussion on the previous page. An expansion of the corresponding clique inequality yields the desired k-fence inequality:

$$\begin{aligned}
&\sum_{i=1}^{k} \pi_{F_k^i}(x) \le 1 \\
\iff\ &\sum_{i=1}^{k} \Bigg(\sum_{ij\in F_k^i} x_{ij} - (|F_k^i| - 1) \Bigg) = \sum_{i=1}^{k} \Bigg(\sum_{ij\in F_k^i} x_{ij} - (k-1) \Bigg) = \sum_{ij\in F_k} x_{ij} - k^2 + k \le 1 \\
\iff\ &\sum_{ij\in F_k} x_{ij} \le k^2 - k + 1.
\end{aligned}$$

(ii) Let M be a Möbius ladder consisting of an odd number $2k+1$ of dicycles $C_0, \dots, C_{2k}$ such that C_i and C_{i+1} have a dipath P_i in common. The argument on page 57 showed that the dipaths $C_i \setminus P_i$ form an odd cycle of $2k+1$ acyclic arc sets in $\mathfrak{G}$. Expanding the corresponding odd cycle inequality for $\check{P}_{\mathrm{SSP}}(\mathfrak{G})$, one obtains the Möbius ladder inequality for M:

$$\begin{aligned}
& \sum_{i=0}^{2k} \pi_{C_i \setminus P_i}(x) \le k \\
\iff & \sum_{i=0}^{2k} \left(\sum_{ij \in C_i \setminus P_i} x_{ij} - (|C_i \setminus P_i| - 1) \right) \le k \\
\iff & \sum_{i=0}^{2k} \sum_{ij \in C_i \setminus P_i} x_{ij} \le \left(\sum_{i=0}^{2k} |C_i \setminus P_i| \right) - (k+1).
\end{aligned}$$

□

Fence and Möbius ladder inequalities have been discussed in a number of different frameworks in the literature. Euler, Jünger & Reinelt [1987] interpret fences and Möbius ladders without arc repetitions as *generalized cliques* and *generalized odd cycles* of an *independence system relaxation* of the ASP. Müller & Schulz [1995, 1996] give cutting plane proofs of fence and Möbius ladder inequalities in the context of *transitive packing*, see also Schulz [1996, Chapter 4]. Caprara & Fischetti [1996] give a derivation of Möbius ladder inequalities in terms of $\{0, \frac{1}{2}\}$ *Chvátal-Gomory cuts*. The last two constructions work for Möbius ladders with arc repetitions and yield a class that is "in the middle" between Grötschel, Jünger & Reinelt [1985a]'s Möbius ladder inequalities and our's, depending on the number of dipaths that contain a given repeated arc.

Separation of fence inequalities was shown to be $\mathcal{NP}$-hard by Müller [1993]. Looking at the *separation of Möbius ladder inequalities*, we notice that the construction that we presented to prove Theorem 2.3.2 (ii) yields a class of *odd cycle of dipath inequalities* that subsumes the Möbius ladder inequalities. Generalizing this class further by allowing the paths $C_i \setminus P_i$ to intersect themselves on nodes and/or arcs, i.e., by substituting in the definition of a Möbius ladder on page 56 *diwalk* for dipath and *closed diwalk* for dicycle, we obtain an even larger class of *odd cycle of diwalk inequalities* for the acyclic subdigraph polytope. Note that these inequalities do in general *not* correspond to odd cycles in the acyclic arc set conflict graph $\mathfrak{G}$, because diwalks may contain dicycles. This obstacle can be overcome by extending $\mathfrak{G}$ in a finite way (including certain relevant diwalks). At this point, however, we do not want to enter this formalism and defer the details of the extension to the proof of Theorem 2.3.3.

We can devise a polynomial time separation algorithm for odd cycle of diwalk inequalities, even though the number of diwalks is exponential and their length is arbitrary. The idea is to construct a most violated cycle of diwalks out of properly interlinked *longest* diwalks. Suppose that M is an odd cycle of diwalks (we want to denote these diwalks with a slight extension of notation by $C_i \setminus P_i$) that induces a violated inequality, and consider the diwalk P_i linking the two (successive) closed diwalks C_i and C_{i+1}. Rearranging, we can isolate the contribution of P_i in the constraint as

$$|P_i| - \sum_{ij \in P_i} x_{ij} < \sum_{j \ne i+1} \Big(\sum_{ij \in C_j \setminus P_j} x_{ij} - |C_j \setminus P_j| \Big) + \sum_{ij \in C_{i+1} \setminus (P_i \cup P_{i+1})} x_{ij} - |C_{i+1} \setminus (P_{i+1} \cup P_i)| + (k+1)$$

(Here, all sets are to be understood as multisets. Note also that we have $<$ because the constraint was, by assumption, violated.)

If we replace P_i by a diwalk P that is shorter with respect to the length function

$$|P| - \sum_{ij \in P} x_{ij} = \sum_{ij \in P} (1 - x_{ij}), \tag{2.1}$$

we get a more violated cycle of diwalks inequality. If we think of any closed diwalk C_i as being composed out of four diwalks, namely the diwalk $P_i^1 := P_i$, that C_i has in common with the succeeding closed diwalk C_{i+1}, the diwalk P_i^2 from P_i^1's head to the diwalk $P_i^3 := P_{i-1}$, that C_i has in common with the preceeding closed diwalk C_{i-1}, and the remaining diwalk P_i^4 from P_i^3's head to P_i^1's tail, the same argument holds for any of these diwalks. This observation allows us to show

2.3.3 Theorem (Polynomial Separability of Odd Cycle of Diwalk Inequalities)
Let D_n be the complete digraph on n nodes and P_{ASP} the associated acyclic subdigraph polytope. Suppose that $x \in \mathbb{Q}^A$ satisfies the constraints (ASP) (ii) and $0 \le x \le \mathbf{1}$. Then:

Odd cycle of diwalk inequalities can be separated in polynomial time.

Proof.
Using Dijkstra's algorithm, we can compute a shortest diwalk $P(u,v)$ with respect to the length (2.1) from any node u to any node v of D_n. We can assume these diwalks $P(u,v)$ w.l.o.g. to be dipaths and, in particular, to be of polynomial length. This yields a polynomial number of shortest diwalks of polynomial length and, moreover, (not every, but) *a* most violated cycle of diwalks will consist of a polynomial number of these shortest diwalks.
We can find a set of them forming an odd cycle of diwalks as follows. We think of all diwalks $P(u,v)$ as a possible common diwalk P_i of two successive closed diwalks C_i and C_{i+1} in a cycle of diwalks. To get the diwalks $C_i \setminus P_i$ as the pieces of the cycle, we compute for any two diwalks P_i and P_j the (2.1)-shortest diwalk $P_i\langle P_j\rangle$ that starts at P_i's head, contains P_j, and ends at P_i's tail. Such a diwalk $P_i\langle P_j\rangle$ will link (on P_j) properly with another diwalk $P_j\langle P_k\rangle$ to form a cycle of diwalks. Computation of the $P_i\langle P_j\rangle$ can be performed in polynomial time and yields, in particular, a polynomial number of $n(n-1)\big(n(n-1)-1\big) = O(n^4)$ diwalks of polynomial length.
We can now construct a graph that has these diwalks $P_i\langle P_j\rangle$ as its nodes with node weights equal to the walk lengths (2.1) and that has all edges of the form $(P_i\langle P_j\rangle, P_j\langle P_k\rangle)$. The node weight on an edge never exceeds one because x satisfies the dicycle inequalities (ASP) (ii), a most violated cycle of diwalks inequality corresponds to a most violated odd cycle inequality in the $P_i\langle P_j\rangle$-graph, and we can find a most violated odd cycle inequality there with the algorithm of Grötschel, Lovász & Schrijver [1988, Lemma 9.1.11]. □

2.3.4 Corollary (Separation of Möbius Ladder Inequalities)
A superclass of the Möbius ladder inequalities can be separated in polynomial time.

To discuss the results on Möbius ladder separation of the literature, we draw the reader's attention to a subtle difference between the ASP and the LOP. While the length of the dicycles in a facet defining Möbius ladder inequality (as defined in this paper) for the *acyclic subdigraph polytope* can be arbitrarily large, the constraint can only define a facet for the *linear ordering polytope* if the length of each dicycle is at most four, see Grötschel, Jünger & Reinelt [1985a]. For the LOP, one can thus restrict Corollary 2.3.4 to the case $|C_i| \le 4$ and then it also follows from Müller & Schulz [1995] and Caprara & Fischetti [1996]. For the ASP, Caprara & Fischetti [1996] showed polynomial time separability of Möbius ladder inequalities where all dicycles have at most constant length.

2.4 The Clique Partitioning, Multi-, and Max Cut Problem

In this section, we investigate set packing relaxations of combinatorial optimization problems in connection with cuts: The clique partitioning, the k-multicut, and the max cut problem. We will see that the 2-chorded cycle inequalities for the clique partitioning polytope can be seen as cycles of "lower triangle" inequalities. An analogous construction for cycles of "upper triangle" inequalities is related to the circulant inequalities for the max cut polytope. As a reference for the clique partitioning problem, we suggest Grötschel & Wakabayashi [1990], see also Oosten, Rutten & Spieksma [1995] for a recent report, for the multicut and the max cut problem Deza & Laurent [1997].
The three cut problems of this section come up on a complete graph $K_n = (V, E)$ on n nodes with integer weights $w : E \to \mathbb{Z}$ on the edges. The *clique partitioning problem* (CPP) is to find a partition of V into an arbitrary number k of cliques $V = C_1 \uplus \ldots \uplus C_k$ (where $\uplus$ denotes a union of disjoint sets), such that the sum of the weights of the edges that run between different cliques is maximal. In other words, we are trying to find a *multicut* $\delta(C_1 : \cdots : C_k)$ of maximum weight, where the number k of (non empty) members C_i of the *clique partition* $C_1 \uplus \ldots \uplus C_k$ is arbitrary. One obtains the *k-multicut problem* (k-MCP) from this formulation by restricting the number of cliques to be less than or equal to some given number k, and the *max cut problem* (MCP) by prescribing $k = 2$. Thus, any (max) cut is a k-multicut ($k \geq 2$), and any k-multicut comes from a clique partition. We remark that the CPP is often stated in an equivalent version to find a clique partition that minimizes the sum of the edge weights inside the cliques.
Integer programming formulations of the clique partitioning and the k-multicut problem read as follows ($x_{ij} = 1$ indicates that ij is in the multicut):

$$\begin{array}{lll}
\max & \sum_{ij \in E} w_{ij} x_{ij} & \\
\text{(ii)} & x_{ij} - x_{jk} - x_{ik} \leq 0 & \forall \{i,j,k\} \subseteq V \\
\text{(iii)} & x \in \{0,1\}^E & \\
\end{array}$$

$$\text{(CPP)}$$

$$\begin{array}{lll}
\max & \sum_{ij \in E} w_{ij} x_{ij} & \\
\text{(i)} & \sum_{ij \in E(W)} x_{ij} \leq |E(W)| - 1 & \forall W \subseteq V \\
 & & \text{with } |W| = k+1 \\
\text{(ii)} & x_{ij} - x_{jk} - x_{ik} \leq 0 & \forall \{i,j,k\} \subseteq V \\
\text{(iii)} & x \in \{0,1\}^E & \\
\end{array}$$

$$(k - \text{MCP})$$

Setting k to 2, inequalities (k-MCP) (i) turn out to be the "upper triangle" inequalities $x_{ij} + x_{jk} + x_{ik} \leq 2$ for all $\{i, j, k\} \subseteq V$ and (2-MCP) is an integer programming formulation for the max cut problem (we speak of upper triangle inequalities because their normal vectors are oriented "upward" such that the induced face is on the "upside" of the polytope). For $k = n$, on the other hand, (k-MCP) (i) becomes void and (n-MCP) coincides with (CPP). Hence, (CPP) is a relaxation of (k-MCP) which in turn is a relaxation of (MCP) and the associated polytopes P_{CPP}, $P_{k-\text{MCP}}$, and P_{MCP} satisfy

$$P_{\text{CPP}} \supseteq P_{k-\text{MCP}} \supseteq P_{\text{MCP}}.$$

In particular, any valid inequality for the clique partitioning polytope is also valid for the k-multicut and the max cut polytope. One such class is the family of *2-chorded cycle inequalities* of Grötschel & Wakabayashi [1990].

A *2-chorded cycle* is an odd cycle C of K_n together with its set of 2-chords $\overline{C}$, see Figure 2.7. The associated inequality states that

$$\sum_{ij \in \overline{C}} x_{ij} - \sum_{ij \in C} x_{ij} \leq (|C| - 1)/2.$$

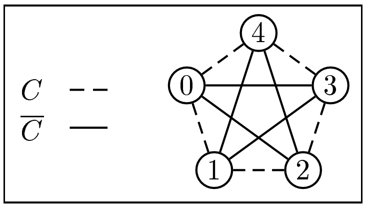


Figure 2.7: A 2-Chorded Cycle.

Müller [1996], and later Caprara & Fischetti [1996], proved that (superclasses of the) 2-chorded cycle inequalities can be separated in polynomial time, see also Müller & Schulz [1996]. We will show now that this class arises from odd cycle inequalities of a set packing relaxation of the clique partitioning (or k-multicut or max cut) problem. Our arguments will yield an alternative proof for the polynomial time separability of this class.
The relaxation involves a "lower triangle" *conflict graph* $\mathfrak{G}_\Delta(K_n) = (\mathfrak{V}_\Delta, \mathfrak{E}_\Delta)$. $\mathfrak{V}_\Delta$ consists of all *ordered* triples $(i, j, k) \in V^3$ of distinct nodes of K_n, the edges $\mathfrak{E}$ of $\mathfrak{G}$ are of the form $(i, j, k)(l, i, j)$, $(i, j, k)(l, j, i)$, $(i, j, k)(l, i, k)$, and $(i, j, k)(l, k, i)$ (the meaning of this definition will become clear in a second).

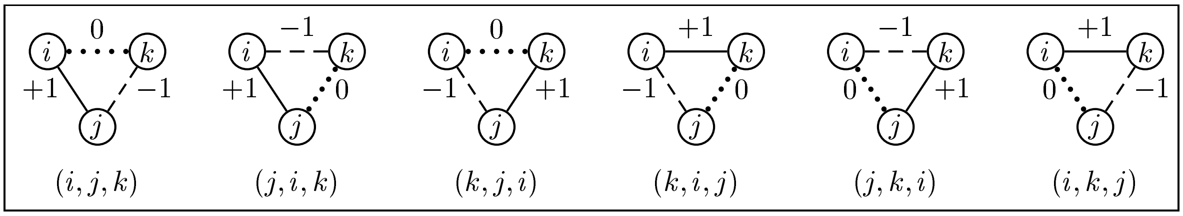


Figure 2.8: Labeling Lower Triangles.

To construct a set packing relaxation of the clique partitioning problem with this graph, we define an aggregation scheme $\pi_\Delta : \mathbb{R}^E \to \mathbb{R}^{\mathfrak{V}_\Delta}$ as

$$\pi_{\Delta_{(i,j,k)}}(x) := x_{ij} - x_{jk} \qquad \forall \text{ ordered triples } (i, j, k) \in \mathfrak{V}_\Delta.$$

$\pi_{\Delta_{(i,j,k)}}(x)$ is integral if $x \in \mathbb{R}^E$ is integral. Moreover, for every multicut $x \in P_{\mathrm{CPP}}$, the component $\pi_{\Delta_{(i,j,k)}}(x)$ attains its maximum value of one if and only if the nodes j and k belong to the same clique ($x_{jk} = 0$), but node i does not ($x_{ij} = x_{ik} = 1$). The reader may think of the triples (i, j, k) as "edge-labeled triangles" as shown in Figure 2.8. Enumerating all possible conflicts between these labeled triangles, it is easy to see that

$$\mathfrak{u}\mathfrak{v} \in \mathfrak{E}_\Delta \iff \pi_{\Delta_\mathfrak{u}}(x) + \pi_{\Delta_\mathfrak{v}}(x) \leq 1 \qquad \forall x \in P_{\mathrm{CPP}} \cap \mathbb{Z}^E$$

and thus for all $x \in P_{\mathrm{CPP}}$. In other words: $\mathfrak{E}_\Delta$ was *defined* in such a way that two triples are joined by an edge if and only if it is impossible that both triples attain their maximum value of one under π simultaneously. This argument shows that $P_{\mathrm{SSP}}(\mathfrak{G}_\Delta)$ is a "lower triangle" set packing relaxation of P_{CPP}:

2.4.1 Lemma (Set Packing Relaxation of the CPP) $\pi_\Delta(P_{\mathrm{CPP}}) \subseteq P_{\mathrm{SSP}}\big(\mathfrak{G}_\Delta(K_n)\big)$.

The construction is called a "lower triangle set packing relaxation" because one obtains the components $\pi_{(i,j,k)}(x) = x_{ij} - x_{jk} \leq 1$ of π from the lower triangle inequalities (CPP) (ii)

$$x_{ij} - x_{jk} - x_{ik} \leq 0 \iff x_{ij} - x_{jk} \leq x_{ik}$$

by setting $x_{ik} := 1$.

We are now ready to state our result that the 2-chorded cycle inequalities are expansions of odd cycle inequalities of $\breve{P}_{\text{SSP}}(\mathfrak{G}_\Delta)$.

2.4.2 Theorem (2-Chorded Cycle Inequalities)

Let K_n be the complete graph on n nodes, P_{CPP} the corresponding clique partitioning polytope, $\mathfrak{G}_\Delta$ the lower triangle conflict graph, and $\breve{P}_{\text{SSP}}(\mathfrak{G}_\Delta)$ the lower triangle set packing relaxation of P_{CPP}. Then:

Every 2-chorded cycle inequality for P_{CPP} is the expansion of an odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G}_\Delta)$.

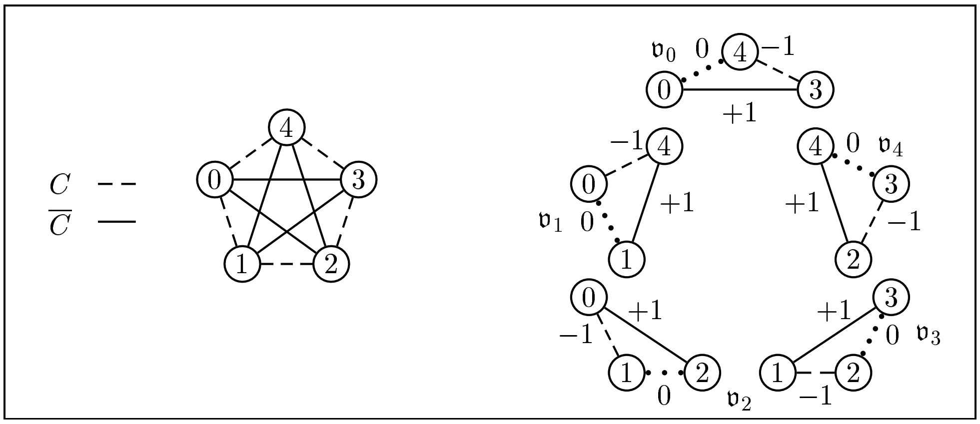


Figure 2.9: An Odd-Cycle of Lower Triangles.

Proof.

Let $C \cup \overline{C}$ be a 2-chorded cycle in K_n with node set $\{0, \ldots, 2k\}$. By definition, $C = \{ij : i = 0, \ldots, 2k,\ j = i+1\}$ and $\overline{C} = \{ij : i = 0, \ldots, 2k,\ j = i+2\}$ (where indices are taken modulo $2k+1$).

Consider the $2k+1$ triples $\mathfrak{v}_i := (i, i-2, i-1)$, $i = 0, \ldots, 2k$ (indices modulo $2k+1$). One verifies that $\mathfrak{v}_i\mathfrak{v}_{i+1} \in \mathfrak{E}$ represents a conflict and forms an edge of an odd cycle in $\mathfrak{G}_\Delta$, see Figure 2.9 for an example. The associated odd cycle inequality expands to the 2-chorded cycle inequality in question:

$$\sum_{i=0}^{2k} \pi_{\Delta_{(i,i-2,i-1)}}(x) = \sum_{i=0}^{2k} x_{(i,i-2)} - x_{(i-2,i-1)} = \sum_{ij \in \overline{C}} x_{ij} - \sum_{ij \in C} x_{ij} \leq (|C|-1)/2.$$

$\square$

Calling the expansions of odd cycle inequalities for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G}_\Delta)$ *inequalities from odd cycles of lower triangle inequalities* and noting $|\mathfrak{V}_\Delta| = O(n^3)$, we obtain

2.4.3 Corollary (Separation of Ineq's from Odd Cycles of Lower Triangle Ineq's)
Let K_n be the complete graph on n nodes and P_{CPP} the associated clique partitioning polytope. Suppose $x \in \mathbb{Q}^E$ satisfies the constraints (CPP) (ii) and $0 \leq x \leq \mathbf{1}$. Then:

Inequalities from odd cycles of lower triangle inequalities can be separated in polynomial time.

2.4.4 Corollary (Separation of 2-Chorded Cycle Inequalities)
A superclass of the 2-chorded cycle inequalities can be separated in polynomial time.

Note that the conflicts between successive triples $\mathfrak{v}_i = (i, i-2, i-1)$ and $\mathfrak{v}_{i+1} = (i+1, i-1, i)$ in a 2-chorded cycle stem from the common edge connecting nodes i and $i-1$ that has a coefficient of -1 in $\pi_{\mathfrak{v}_{i+1}}$ and 0 in $\pi_{\mathfrak{v}_i}$. But conflicts arise also from common edges with $+1$ and -1 coefficients. Thus, besides possible node/edge repetitions and the like, odd cycles of lower triangle inequalities give also rise to inequalities that do not correspond to 2-chorded cycle inequalities.
So far we have studied inequalities from pairwise conflicts of lower triangle inequalities. In the case of the max cut problem, the constraints (2-MCP) (i) form a class of "upper triangle inequalities"

$$x_{ij} + x_{jk} + x_{ik} \leq 2 \qquad \forall i, j, k \in V.$$

Analogous to the lower triangle case, we will now construct "inequalities from odd cycles of upper triangle inequalities" for the max cut polytope. These constraints are related to the $C(2k+1, 2)$*-circulant inequalities* of Poljak & Turzik [1992].
A $C(2k+1,2)$*-circulant* is identical to a 2-chorded cycle on an odd number of $2k+1$ nodes. We distinguish circulants $C(2k+1, 2)$ with *odd* k and with *even* k. The associated inequalities are

$$\sum_{ij \in C(2k+1,2)} x_{ij} \leq 3k+1, \qquad \text{if } k \bmod 2 = 1 \qquad \sum_{ij \in C(2k+1,2)} x_{ij} \leq 3k, \qquad \text{if } k \bmod 2 = 0.$$

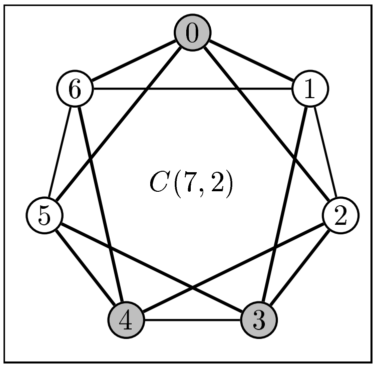


Figure 2.10: The Odd-k Circulant $C(7,2)$.

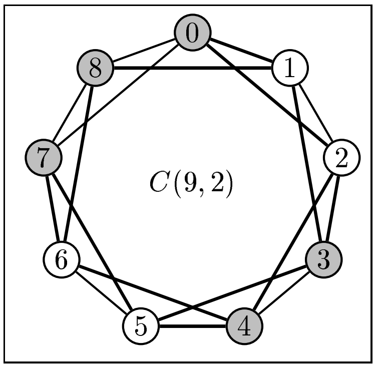


Figure 2.11: The Even-k Circulant $C(9,2)$.

Even-k circulant inequalities have been introduced by Poljak & Turzik [1992]. These constraints have a right-hand side of $3k$, whereas the odd case requires an increase of one in the right-hand side: Figure 2.10 shows the odd-k circulant $C(7, 2)$ ($k = 3$); the white and gray nodes form the shores of a cut with $10 = 3 \cdot 3 + 1 > 3 \cdot 3 = 9$ edges highlighted. Alternatingly putting pairs of successive nodes on the left and on the right shore of the cut except for the first node, one verifies that the right-hand side $3k + 1$ is best possible for odd k. Figure 2.11 shows a tight configuration for the case where k is even. (A rigorous proof for the validity of these constraints will follow from the upcoming discussion.)

We will show now that the circulant inequalities can be seen as "strengthened" odd cycle inequalities of an appropriate "upper triangle" set packing relaxation of the max cut problem; strengthened means that for even k, the right-hand side of the cycle of upper triangles inequality exceeds the right-hand side of the corresponding circulant inequality by one. Our considerations allow to design a polynomial time algorithm for separating inequalities from cycles of upper triangle inequalities. Poljak & Turzik [1992], on the other hand, have shown that separation of the *exact* class of $C(2k + 1, 2)$-circulant inequalities is $\mathcal{NP}$-hard.

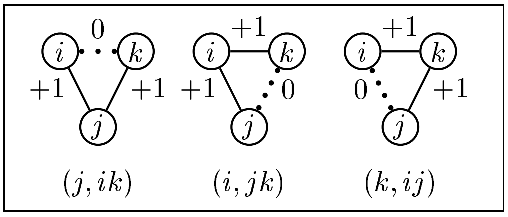


Figure 2.12: Labeling Upper Triangles.

As usual, the upper triangle set packing relaxation is based on an upper triangle conflict graph $\mathfrak{G}^{\Delta}(K_n) = (\mathfrak{V}^{\Delta}, \mathfrak{E}^{\Delta})$. This time, $\mathfrak{V}^{\Delta}$ consists of all 2-tuples $(i, jk) \in V \times E$ such that $i \notin jk$, while $\mathfrak{E}^{\Delta}$ is the set of all $(i, jk)(j, kl)$. To construct a set packing relaxation of P_{MCP} by means of this graph, we introduce the aggregation scheme $\pi^{\Delta} : \mathbb{R}^E \to \mathbb{R}^{\mathfrak{V}^{\Delta}}$ defined as

$$\pi^{\Delta}_{(i,jk)}(x) := x_{ij} + x_{ik} - 1.$$

One gets $\pi^{\Delta}_{(i,jk)}(x) \leq 1$ from the upper triangle inequality

$$x_{ij} + x_{ik} + x_{jk} \leq 2 \iff x_{ij} + x_{ik} - 1 \leq 1 - x_{jk}$$

by setting $x_{jk} := 0$, hence the name "upper triangle" conflict graph. This rearrangement also proves that $\pi^{\Delta}_{(i,jk)}(x)$ attains its maximum value of one if and only if node i is on one side of the cut, while nodes j and k are on the other. Again, one may think of the nodes i, j, and k as forming triangles with the edges labeled as indicated in Figure 2.12 and sees that

$$\mathfrak{u}\mathfrak{v} \in \mathfrak{E}^{\Delta} \iff \pi^{\Delta}_{\mathfrak{u}}(x) + \pi^{\Delta}_{\mathfrak{v}}(x) \leq 1 \qquad \forall x \in P_{\mathrm{MCP}} \cap \mathbb{Z}^E.$$

Thus, $\breve{P}_{\mathrm{SSP}}(\mathfrak{G}^{\Delta})$ is an "upper triangle" set packing relaxation of P_{MCP}.

2.4.5 Lemma (Set Packing Relaxation of the MCP) $\pi^{\Delta}(P_{\mathrm{MCP}}) \subseteq \breve{P}_{\mathrm{SSP}}\big(\mathfrak{G}^{\Delta}(K_n)\big)$.

This construction yields the circulant inequalities for $k \bmod 2 = 1$ as expansions of odd cycle inequalities for the upper triangle set packing relaxation of the max cut polytope. The case $k \bmod 2 = 0$ can be settled by *strengthening* the associated odd cycle inequality, i.e., one can *a posteriori* decrease the right-hand side by one and the inequality remains valid.

2.4.6 Theorem (Circulant Inequalities)
Let K_n be the complete graph on n nodes, P_{MCP} the corresponding max cut polytope, $\mathfrak{G}^\Delta$ the upper triangle conflict graph, and $\breve{P}_{\text{SSP}}(\mathfrak{G}^\Delta)$ the upper triangle set packing relaxation of P_{MCP}.

(i) Every odd-k circulant inequality for P_{MCP} is the expansion of an odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G}^\Delta)$.

(ii) Every even-k circulant inequality for P_{MCP} is the expansion of a strengthened odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G}^\Delta)$.

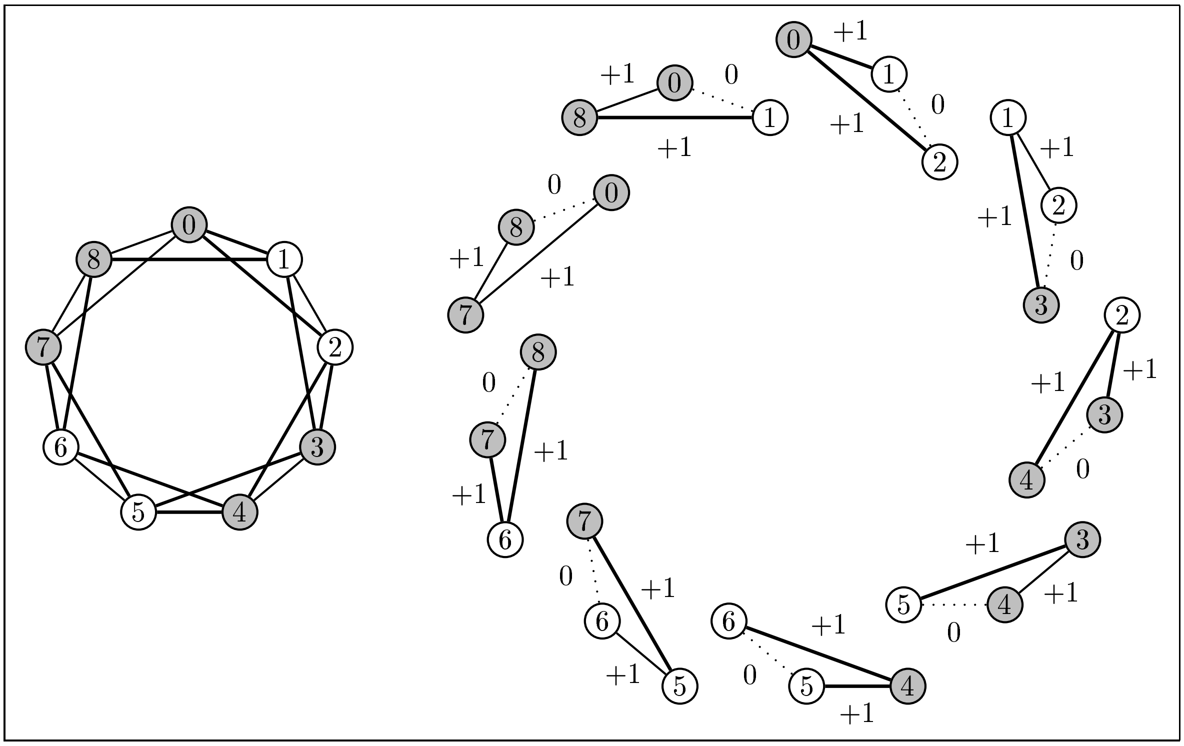


Figure 2.13: An Odd Cycle of Upper Triangles.

Proof.
(i) Let $C(2k+1,2)$ be an odd-k circulant with node set $\{0,\dots,2k+1\}$. Consider the $2k+1$ 2-tuples $\mathfrak{v}_i := \big(i,(i+1,i+2)\big)$ (with indices taken modulo $2k+1$). One verifies that the tuples $\mathfrak{v}_i$ and $\mathfrak{v}_{i+1}$ are in conflict, i.e., $\mathfrak{v}_i\mathfrak{v}_{i+1} \in \mathfrak{E}^\Delta$, and form an odd cycle in $\mathfrak{G}^\Delta$, see Figure 2.13. The associated odd cycle inequality expands to an odd-k circulant inequality:

$$\begin{aligned} & \sum_{i=0}^{2k} \pi^\Delta_{\big(i,(i+1,i+2)\big)}(x) \le k \\ \iff\ & \sum_{i=0}^{2k} (x_{(i,i+1)} + x_{(i,i+2)} - 1) \le k \\ \iff\ & \sum_{ij\in C(2k+1,2)} x_{ij} \le 3k+1. \end{aligned}$$

(ii) The even case is analogous to the odd case. To see that one can reduce the right-hand side of the odd cycle inequality by one and $\sum_{i=0}^{2k} \pi^{\Delta}_{(i,(i+1,i+2))}(x) \leq k-1$ is still valid, suppose this is not so and let $x \in P_{\text{MCP}}$ be the incidence vector of a cut that violates this constraint. Now, max $\{0, \pi^{\Delta}(x)\}$ is the incidence vector of a stable set in $\mathfrak{G}^{\Delta}$, and, clearly, this vector must be tight for the (unstrengthened) odd cycle inequality. This means that we have k 2-tuples with max $\{0, \pi^{\Delta}_{(i,(i+1,i+2))}(x)\} = 1$, and $k+1$ 2-tuples with max $\{0, \pi^{\Delta}_{(i,(i+1,i+2))}(x)\} = 0$, that are arranged in such a way that the two types appear alternatingly except for one time, where we have two "zeros" next to each other. Looking at a tuple with max $\{0, \pi^{\Delta}_{(i,(i+1,i+2))}(x)\} = 1$, we see that node i must be on one side of the cut while nodes $i+1$ and $i+2$ must be on the other. The next "one" max $\{0, \pi^{\Delta}_{(i+2,(i+3,i+4))}(x)\} = 1$ forces nodes $i+3$ and $i+4$ to be on the same side as i. Starting without loss of generality at $\pi^{\Delta}_{(0,(1,2))} = 1$ and continuing k times like this, all nodes of the circulant are assigned to one side of the cut or another in a unique way. When k is even, this results in nodes $2k-1$, $2k$, and 0 ending up on the same side such that $\pi^{\Delta}_{(2k-1,(2k,0))}(x)\} = -1$, see the right side of Figure 2.13 for an example; but then $\sum_{i=0}^{2k} \pi^{\Delta}_{(i,(i+1,i+2))}(x) \leq k-1$, a contradiction. □

Calling the expansion of an odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G}^{\Delta})$ an *inequality from an odd cycle of upper triangle inequalities*, we obtain

2.4.7 Corollary (Separation of Ineq's from Cycles of Upper Triangle Ineq's)
Let K_n be the complete graph on n nodes and P_{MCP} the associated max cut polytope. Suppose $x \in \mathbb{Q}^E$ satisfies the constraints (2-MCP) (i), (ii), and $0 \leq x \leq \mathbf{1}$. Then:

Inequalities from odd cycles of upper triangle inequalities can be separated in polynomial time.

2.4.8 Corollary (Separation of Circulant Inequalities)

(i) A superclass of odd-k $C(2k+1,2)$ circulant inequalities can be separated in polynomial time.

(ii) A superclass of even-k $C(2k+1,2)$ circulant inequalities with their right-hand sides increased by one can be separated in polynomial time.

We remark again that Poljak & Turzik [1992] have shown that it is $\mathcal{NP}$-complete to determine whether a graph contains a $C(2k+1,2)$ circulant and thus, separation of the *exact* class of $C(2k+1,2)$ circulant inequalities is $\mathcal{NP}$-hard.

2.5 The Set Packing Problem

We have demonstrated in the examples of the preceeding sections that certain combinatorial optimization problems have interesting set packing relaxations. Perhaps a bit surprising, we show now that the set packing problem *itself* also has interesting set packing relaxations! These considerations yield alternative derivations, generalizations, and separation techniques for several classes of wheel inequalities, including two classes introduced by Barahona & Mahjoub [1994] and Cheng & Cunningham [1997], as well as new classes such as, e.g., certain "cycle of cycles inequalities". A survey on results for the set packing problem can be found in Chapter 1 of this thesis.
The examples of this section are based on a "rank" set packing relaxation that we introduce now. Given a set packing problem (SSP) on a graph $G = (V, E)$, the associated conflict graph $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$ of the relaxation has the set $\mathfrak{V} := \{H : H \subseteq G\}$ of all (not necessarily node

induced) subgraphs of G as its nodes. In order to define the set of edges, we consider the aggregation scheme $\pi : \mathbb{R}^V \to \mathbb{R}^{\mathfrak{V}}$ defined as

$$\pi_H(x) = \sum_{i \in V(H)} x_i - (\alpha(H) - 1) \qquad \forall \text{ subgraphs } H \in \mathfrak{V} \text{ of } G,$$

where $\alpha(H)$ denotes the *rank*, i.e., the maximum cardinality of a stable set, of H. We draw an edge between two subgraphs H and W if there is no stable set in G such that its restrictions to H and W are simultaneously stable sets of maximum cardinality in H and W, i.e.,

$$HW \in \mathfrak{E} \iff \pi_H(x) + \pi_W(x) \leq 1 \qquad \forall x \in P_{\text{SSP}}(G) \cap \mathbb{Z}^V.$$

By definition, the rank conflict graph $\mathfrak{G}$ depends only on G and this is why we occasionally also denote it by $\mathfrak{G}(G)$. Well known arguments show that $\breve{P}_{\text{SSP}}(\mathfrak{G})$ is a set packing relaxation of P_{SSP} in the exponential space $\mathbb{R}^{\mathfrak{V}}$.

2.5.1 Lemma (Rank Set Packing Relaxation of the SSP) $\pi(P_{\text{SSP}}) \subseteq \breve{P}_{\text{SSP}}(\mathfrak{G})$.

2.5.1 Wheel Inequalities

One method to derive classes of polynomial time separable inequalities from the rank relaxation is to consider subgraphs of $\mathfrak{G}$ of polynomial size. A natural idea is to restrict the set of $\mathfrak{G}$'s nodes to

$$\mathfrak{V}_k := \{H : H \subseteq G : |V(H)| \leq k\},$$

the subgraphs H of G with bounded numbers of nodes $|V(H)| \leq k$ for some arbitrary, but fixed bound k. The smallest interesting case is $k = 2$, where H ($|V(H)| \leq 2$) is either empty, a singleton, an edge, or a coedge (complement of an edge). The odd cycle inequalities that one obtains from this restricted relaxation $\breve{P}_{\text{SSP}}(\mathfrak{G}[\mathfrak{V}_k])$ contain, among other classes, the *odd wheel inequalities* of the set packing polytope.

A $2k+1$-*wheel* is an odd cycle C of $2k+1$ nodes $\{0, \ldots, 2k\}$ plus an additional node $2k+1$ that is connected to all nodes of the cycle. C is the *rim* of the wheel, node $2k+1$ is the *hub*, and the edges connecting the node $2k+1$ and i, $i = 0, \ldots, 2k$, are called *spokes*. For such a configuration, we have that

$$kx_{2k+1} + \sum_{i=0}^{2k} x_i \leq k.$$

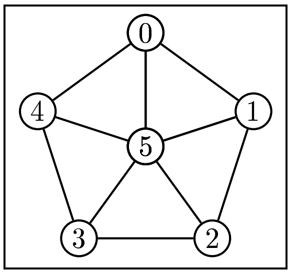


Figure 2.14: A 5-Wheel.

An odd wheel inequality can be obtained by a sequential lifting of the hub into the odd cycle inequality that corresponds to the rim. Trying all possible hubs, this yields a polynomial time *separation* algorithm for wheel inequalities. An alternative derivation is

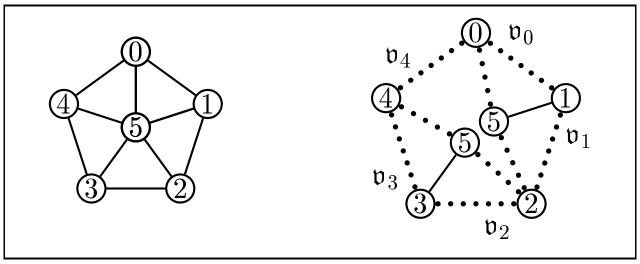


Figure 2.15: A Wheel and a Cycle of Nodes and Edges.

2.5.2 Theorem (Odd Wheel Inequalities)
Let $G = (V, E)$ be a graph, P_{SSP} the corresponding set packing polytope, $\mathfrak{G}$ the rank conflict graph, and $\breve{P}_{\text{SSP}}(\mathfrak{G})$ the rank set packing relaxation of P_{SSP}. Then:

Every odd wheel inequality for P_{SSP} is the expansion of an odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G}[\mathfrak{V}_2])$.

Proof.
Consider a $2k+1$ wheel with rim $C = \{0, \ldots, 2k\}$ and hub node $2k+1$. The subgraphs $\mathfrak{v}_i := G[\{i, 2k+1\}]$, $i = 1, 3, \ldots, 2k-1$, induced by the spokes with odd rim nodes, and the subgraphs $\mathfrak{v}_i = G[\{i\}]$, $i = 0, 2, \ldots, 2k$, induced by the even rim nodes, form an odd cycle in $\mathfrak{G}$, see Figure 2.15. Expanding the associated odd cycle inequality yields the wheel inequality:

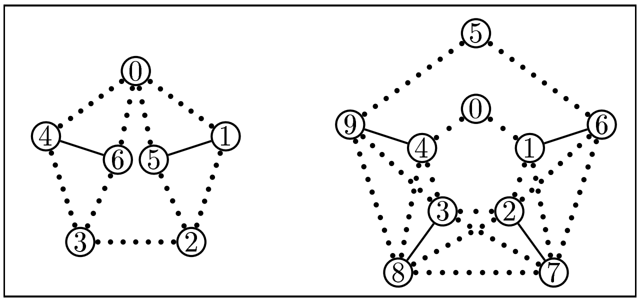
$$\sum_{i=0}^{2k} \pi_{\mathfrak{v}_i(x)} = \sum_{i=1,3,\ldots,2k-1} (x_i + x_{2k+1}) + \sum_{i=0,2,\ldots,2k} x_i = kx_{2k+1} + \sum_{i=0}^{2k} x_i \leq k. \qquad \square$$

2.5.3 Corollary (Separation of Ineq's from Odd Cycles of Nodes, Edges, Coedges)
Ineq's from odd cycles of nodes, edges, and coedges can be separated in polynomial time.

We show now two *examples* of cycles of nodes, edges, and coedges that give rise to *facetial* inequalities that do not correspond to odd wheels. The cycle on the left side of Figure 2.16 consists of the nodes 0, 2, and 3 and the edges $(1,5)$ and $(4,6)$, the one on the right of the edges $(1,6)$, $(2,7)$, $(3,8)$, and $(4,9)$ and the coedge $(0,5)$. The associated inequalities are

$$x_0 + (x_5 + x_1) + x_2 + x_3 + (x_6 + x_4) \leq 2 \iff \textstyle\sum_{i=0}^{6} x_i \leq 2$$
$$(x_5 + x_0 - 1) + (x_6 + x_1) + (x_7 + x_2) + (x_8 + x_3) + (x_9 + x_4) \leq 2 \iff \textstyle\sum_{i=0}^{9} x_i \leq 3.$$

$$\sum_{i=0}^{6} x_i \leq 2$$



$$\sum_{i=0}^{9} x_i \leq 3$$

Figure 2.16: Two Generalizations of Odd Wheel Inequalities.

Another generalization of odd wheel inequalities was given by Barahona & Mahjoub [1994] and Cheng & Cunningham [1997]. They introduce two classes of inequalities that have *subdivisions* of odd wheels as support graphs, where each face cycle must be odd, see Figure 2.17. Following Cheng & Cunningham [1997]'s terminology and denoting the set of end nodes of the *even spokes* (with an even number of *edges*) of an odd wheel W of this kind with some number $2k+1$ of faces by $\mathcal{E}$, the set of end nodes of the *odd spokes* (with an odd number of edges) by $\mathcal{O}$, and the hub by h, a *wheel inequality of type I* states that

$$kx_h + \sum_{i \in W-h} x_i + \sum_{i \in \mathcal{E}} x_i \le \frac{|W| + |\mathcal{E}|}{2} - 1. \tag{2.2}$$

A second variant of *wheel inequalities of type II* (associated to the same wheel) states that

$$(k+1)x_h + \sum_{i \in W-h} x_i + \sum_{i \in \mathcal{O}} x_i \le \frac{|W| + |\mathcal{O}| - 1}{2}. \tag{2.3}$$

We remark that these wheels do in general not arise from cycles of subgraphs of bounded size because they contain potentially very long paths.

2.5.4 Theorem (Odd Wheel Inequalities)
Let $G = (V, E)$ be a graph, P_{SSP} the corresponding set packing polytope, $\mathfrak{G}$ the rank conflict graph, and $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ the rank set packing relaxation of P_{SSP}. Then:

Every odd wheel inequality of type I and II for P_{SSP} is the expansion of an odd cycle inequality for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$.

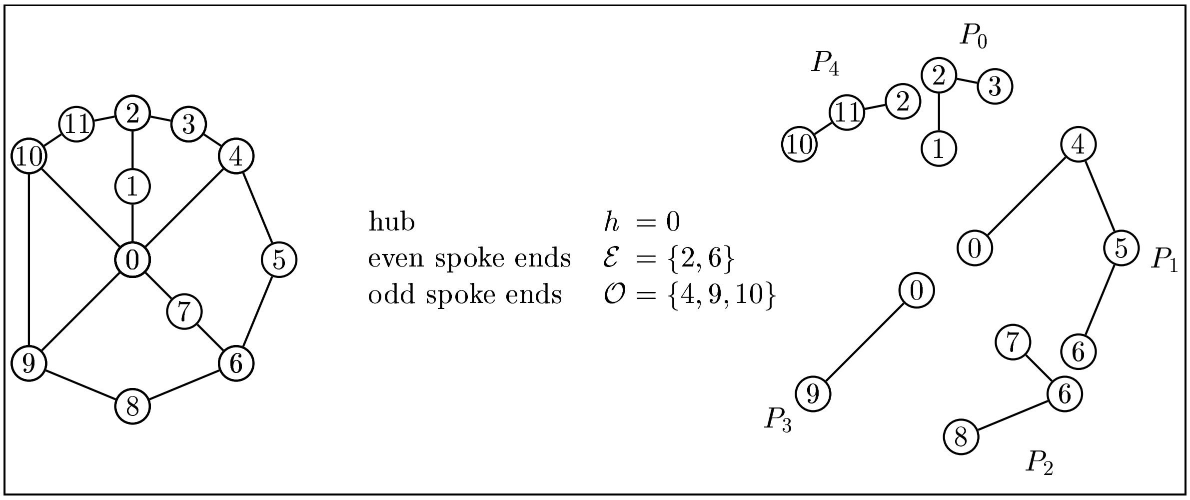


Figure 2.17: A 5-Wheel and a 5-Cycle of Paths of Type I.

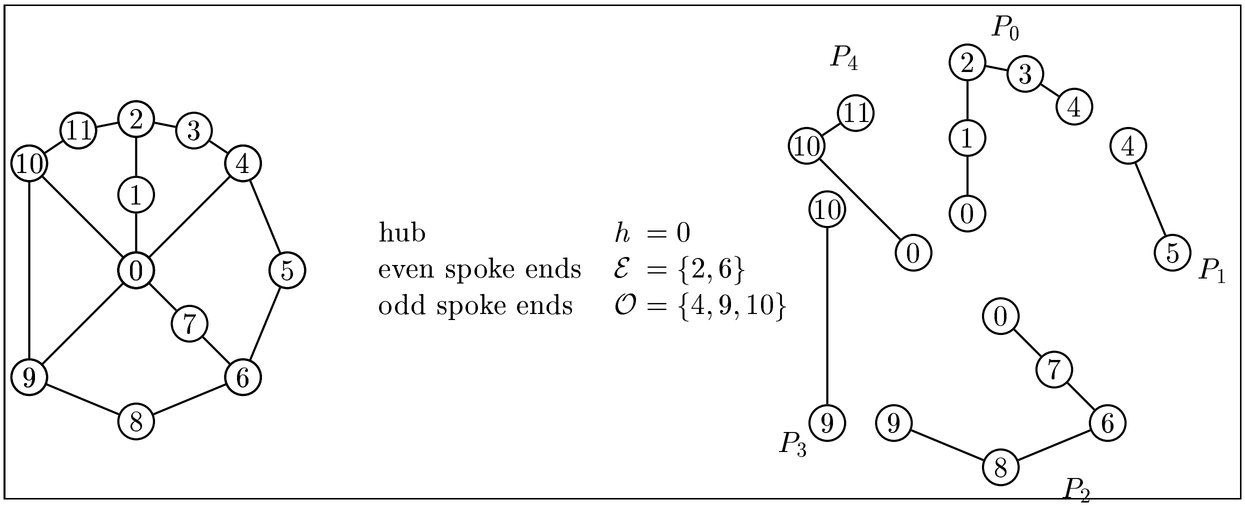
Proof.
(i) Wheel inequalities of type I.
The idea of the proof is to obtain the wheel inequality (2.2) of type I as a cycle of paths. Orienting a $2k+1$-wheel clockwise, it consists of $2k+1$ spoke paths S_i, $i = 0, \ldots, 2k$, and the same number of rim paths R_i such that R_i connects the ends of spokes S_i and S_{i+1} (indices in the proof are taken modulo $2k+1$). We can then compose the wheel from the paths

$$P_i := S_i \cup \left\{ \begin{array}{ll} R_i, & \text{if } S_{i+1} \text{ is even} \\ R_i \setminus S_{i+1}, & \text{if } S_{i+1} \text{ is odd} \end{array} \right\} \setminus \left\{ \begin{array}{ll} \emptyset, & \text{if } i \text{ is odd} \\ \{h\}, & \text{if } i \text{ is even,} \end{array} \right. \qquad i = 0, \ldots, 2k,$$

see Figure 2.17. By definition, a path P_i consists of the spoke S_i (plus minus the hub depending on i) and the *full* rim path R_i if the end node of the next spoke (in clockwise order) is even, or the rim path R_i *without* the end of the next spoke in case this spoke is odd. In this way, the even spoke-ends, having a coefficient of two in the wheel inequality, appear in two paths, the odd spoke-ends in one. Finally, the hub is removed from all paths with even index. It is not hard to see that any two successive paths P_i and P_{i+1} are in pairwise conflict: The subpaths $P_i \setminus \{h\}$ (with the hub removed) are all odd and in pairwise conflict, and the hub is in conflict with any of these subpaths. The odd cycle inequality corresponding to the paths P_i expands into the odd wheel inequality (2.2):

$$
\begin{aligned}
& \sum_{i=0}^{2k} \pi_{P_i}(x) \leq k \\
\iff & \sum_{i=0}^{k} \left(\sum_{j \in P_{2i}} x_j - (|P_{2i}| - 1)/2 \right) + \sum_{i=0}^{k-1} \left(\sum_{j \in P_{2i+1}} x_j - (|P_{2i+1}| - 2)/2 \right) \leq k \\
\iff & k x_h + \sum_{j \in W \setminus \{h\}} x_j + \sum_{j \in \mathcal{E}} x_j - \frac{|W| - 1 + k + |\mathcal{E}| - (k+1) - 2k}{2} \leq k \\
\iff & k x_h + \sum_{j \in W \setminus \{h\}} x_j + \sum_{j \in \mathcal{E}} x_j \leq \frac{|W| + |\mathcal{E}| - 2k - 2}{2} + k = \frac{|W| + |\mathcal{E}|}{2} - 1.
\end{aligned}
$$

Here, $|P_i|$ denotes the number of nodes in the path P_i.



hub $h = 0$
even spoke ends $\mathcal{E} = \{2, 6\}$
odd spoke ends $\mathcal{O} = \{4, 9, 10\}$

Figure 2.18: A 5-Wheel and a 5-Cycle of Paths of Type II.

(ii) Wheel inequalities of type II.
The wheel inequalities (2.3) of type II can be derived in much the same way as their relatives of type I. For the sake of completeness, we record the path decomposition

$$
P_i := S_i \cup \left\{ \begin{array}{ll} R_i, & \text{if } S_{i+1} \text{ is odd} \\ R_i \setminus S_{i+1}, & \text{if } S_{i+1} \text{ is even} \end{array} \right\} \setminus \left\{ \begin{array}{ll} \emptyset, & \text{if } i \text{ is even} \\ \{h\}, & \text{if } i \text{ is odd,} \end{array} \right. \qquad i = 0, \ldots, 2k.
$$

One can verify that, again, any two successive paths are in conflict. A final calculation to expand the resulting odd cycle inequality yields the wheel inequality (2.3) of type II:

$$\begin{aligned}
&\sum_{i=0}^{2k} \pi_{P_i}(x) \leq k \\
\iff\ &\sum_{i=0}^{k} \Bigg(\sum_{j\in P_{2i}} x_j - (|P_{2i}|-1)/2 \Bigg) + \sum_{i=0}^{k-1} \Bigg(\sum_{j\in P_{2i+1}} x_j - (|P_{2i+1}|-2)/2 \Bigg) \leq k \\
\iff\ &(k+1)x_h + \sum_{j\in W\setminus\{h\}} x_j + \sum_{j\in\mathcal{O}} x_j - \frac{|W|-1+(k+1)+|\mathcal{O}|-(k+1)-2k}{2} \leq k \\
\iff\ &(k+1)x_h + \sum_{j\in W\setminus\{h\}} x_j + \sum_{j\in\mathcal{O}} x_j \leq \frac{|W|+|\mathcal{O}|-2k-1}{2} + k = \frac{|W|+|\mathcal{O}|-1}{2}.
\end{aligned}$$

□

One can also derive polynomial time *separation* algorithms of much the same flavour as for the Möbius ladder inequalities; such procedures are given in Cheng & Cunningham [1997].

2.5.2 A New Family of Facets for the Set Packing Polytope

The rank relaxation of the set packing problem offers ample possibilities to define new classes of polynomially separable inequalities for the set packing problem. We discuss as one such example a *cycle of cycles* inequality; *cycle of cliques* inequalities and certain liftings of them are studied in Tesch [1994, Section 7.3].

The way to construct a cycle of cycles inequality is to link an odd number $2k+1$ of odd cycles $C_0, \ldots, C_{2k}$ to a circular structure such that any two successive cycles are in pairwise conflict, i.e., $\pi_{C_i}(x) + \pi_{C_{i+1}}(x) \leq 1$ (indices taken modulo $2k+1$).

One way to do this is to select from each cycle C_i three successive nodes $L_i \subseteq C_i$ that will serve as a part of the inter-cycle links yet to be formed. The *link* L_i has the property that $\pi_{C_i}(x) = 1$ implies that at least one of the nodes in L_i is contained in the stable set $\operatorname{supp}(x)$, i.e.,

$$\pi_{C_i}(x) = \sum_{j\in C_i} x_j - \big(|C_i|-1\big)/2 = 1 \Longrightarrow \sum_{j\in L_i} x_j \geq 1.$$

If we make sure that any two successive links L_i and L_{i+1} are joined by the edge set of the complete bipartite graph $K_{3,3}$, we have that

$$\pi_{C_i}(x) = 1 \Longrightarrow \sum_{j\in L_i} x_j \geq 1 \Longrightarrow \sum_{j\in L_{i+1}} x_j = 0 \Longrightarrow \pi_{C_{i+1}}(x) \leq 0 \qquad \forall x\, P_{\mathrm{SSP}}(G) \cap \mathbb{Z}^V$$

and, vice versa, that $\pi_{C_{i+1}}(x) = 1 \Longrightarrow \pi_{C_i}(x) \leq 0$ holds for all incidence vectors x of stable sets in G. But then, every two successive cycles C_i and C_{i+1} are in conflict, i.e., $\pi^{C_i}(x) + \pi^{C_{i+1}}(x) \leq 1$, and the cycles C_i form an odd cycle in $\mathfrak{G}$, see Figure 2.19 (links are colored gray).

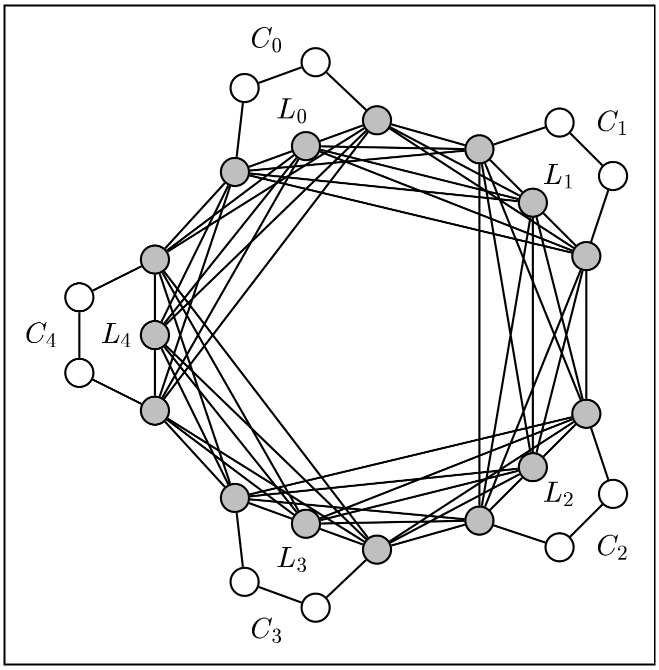


Figure 2.19: A 5-Cycle of 5-Cycles.

2.5.5 Theorem (Cycle of Cycles Inequality)
Let $G = (V, E)$ be a graph and P_{SSP} be the corresponding set packing polytope. Let C_i, $i = 0, \ldots, 2k$, be an odd cycle in G and $L_i \subseteq C_i$, $i = 0, \ldots, 2k$, a set of three successive nodes in C_i. Assume further that L_i and L_{i+1}, $i = 0, \ldots, 2k$, are joined by a complete $K_{3,3}$. Then:

The cycle of cycles inequality

$$\sum_{i=0}^{2k} \sum_{j \in C_i} x_j \le \left(\sum_{i=0}^{2k} (|C_i| - 1)/2 \right) - (k+1)$$

is valid for P_{SSP}.

Proof.

$$\begin{aligned} & \sum_{i=0}^{2k} \pi_{C_i}(x) \le k \\ \iff\; & \sum_{i=0}^{2k} \left(\sum_{j \in C_i} x_j - \big((|C_i| - 1)/2 - 1\big) \right) \le k \\ \iff\; & \sum_{j \in \cup C_i} x_j \le \left(\sum_{i=0}^{2k} \big((|C_i| - 1)/2 - 1\big) \right) + k = \left(\sum_{i=0}^{2k} (|C_i| - 1)/2 \right) - (k+1). \end{aligned}$$

□

2.5.6 Theorem (Separation of Cycle Of Cycles Inequalities)
Cycle of cycles inequalities can be separated in polynomial time.

Proof.
The number of potential links L_i is polynomial of order $O(|V|^3)$. We set up a *link graph* that has the links as its nodes; this device will, in a second, turn out to be a subgraph of $\mathfrak{G}$. Two

links are connected by an edge if and only if they are joined by a $K_{3,3}$. To assign weights to the links, we calculate for each link L_i the shortest even path P_i in G (with an even number of nodes) that connects the two endpoints of the link; here, shortest means shortest with respect to the length function $(1 - x_i - x_j)/2$ for all edges $ij \in E$. $L_i \cup P_i$ forms a shortest odd cycle C_i through L_i with respect to the length $\sum_{j \in C_i}(1 - x_j - x_{j+1})/2 = 1/2 - \pi_{C_i}(x)$, i.e., a longest odd cycle C_i through L_i with respect to the length $\pi_{C_i}(x)$. We set the weight of link L_i to the value $\pi_{C_i}(x)$, obtain the link graph as a subgraph of $\mathfrak{G}[\{C_i\}]$ (some edges that correspond to "non-link conflicts" are possibly missing), and detect a violated odd cycle inequality in the link graph if and only if a violated cycle of cycles inequality in G exists (eventually, we first have to separate the edge inequalities of the link graph). $\square$

A cycle of cycles inequality will in general not be *facet inducing*, for example, if one of the cycles has a chord that joins two non-link nodes. But one can come up with conditions that ensure this property. The most simple case is where the cycles C_i are holes, all node disjoint, and the only edges that run between different holes belong to the links, i.e., we have a "hole of holes". In this case, the cycle of cycles inequality is easily shown to be facet inducing using standard techniques, like noting that every edge in a hole of holes is *critical*.

2.5.7 Proposition (Facet Inducing Cycle of Cycles Inequalities)
If every cycle in a cycle of cycles inequality is a hole, and the only edges that run between different holes emerge from the links, then the cycle of cycles inequality is facet inducing for the set packing polytope $P_{\mathrm{SSP}}(G)$ associated to the support graph G of the inequality.

We want to give now an *alternative proof* for the faceteness of the hole of holes inequality. The technique that we are going to demonstrate works also for other constructions of this type. It is our aim to give an example how aggregation techniques, although *not designed for facetial investigations*, can sometimes lend themselves to results in this direction.

Proof (of Proposition 2.5.7).
The idea of the proof is to exploit the composition structure of a hole of holes $\mathfrak{C} = \{C_0, \ldots, C_{2k}\}$. To this purpose, it is convenient to consider $\mathfrak{C}$ sometimes as a hole in the conflict graph $\mathfrak{G}$, in which case we want to denote it by $\overline{\mathfrak{C}}$, and sometimes as a structure in the original graph G, and then we want to use the original notation $\mathfrak{C}$. The first step is to look at the inequality as a linear form in the image space of the aggregation, namely, as the odd cycle inequality

$$\sum_{\mathfrak{v}=C_0,\ldots,C_{2k}} \overline{x}_{\mathfrak{v}} \leq k \tag{2.4}$$

of the set packing polytope $P_{\mathrm{SSP}}(\overline{\mathfrak{C}})$ associated to the odd hole $\overline{\mathfrak{C}}$. As constraint (2.4) is a facet of $P_{\mathrm{SSP}}(\overline{\mathfrak{C}})$, there are $2k+1$ affinely independent incidence vectors $\overline{x}^r$, $r = 0, \ldots, 2k$, of set packings on the induced face. Likewise, each of the individual holes C_i has a set of $|C_i|$ affinely independent incidence vectors of set packings x^{is} in the graph C_i, $s = 1, \ldots, |C_i|$, that are tight for the odd cycle inequality associated to C_i, i.e.,

$$\textstyle\sum_{j=1}^{|C_i|} x_j^{is} = (|C_i| - 1)/2, \qquad i = 0, \ldots, 2k,\ s = 1, \ldots, |C_i|.$$

Moreover, there is an incidence vector x^{i0} of a set packing in C_i such that

$$\textstyle\sum_{j=1}^{|C_i|} x_j^{i0} = (|C_i| - 1)/2 - 1, \qquad i = 0, \ldots, 2k.$$

Note that

$$\textstyle\sum_{j=1}^{|C_i|} x_j^{is} - \big((|C_i|-1)/2-1\big) = 1, \qquad i=0,\ldots,2k,\ s=1,\ldots,|C_i|,$$

$$\textstyle\sum_{j=1}^{|C_i|} x_j^{i0} - \big((|C_i|-1)/2-1\big) = 0, \qquad i=0,\ldots,2k.$$

We will use the vectors x^{is} to expand the vectors $\overline{x}^r$ into a set of $\sum_{i=0}^{2k}|C_i|$ affinely independent incidence vectors of stable sets in $\mathfrak{C}$ that are tight for the hole of holes inequality in question. To this purpose, we can assume without loss of generality that

$$\overline{x}^r_{C_r} = 1, \qquad r=0,\ldots,2k,$$

i.e., the r-th component of the r-th (aggregated) incidence vector $\overline{x}^r$ is one. We now "blow up" each vector $\overline{x}^r$ into $|C_r|$ vectors $y^{rs}\in\mathbb{R}^{\mathfrak{C}}$, $s=1,\ldots,|C_r|$, defined as

$$y^{rs}_{C_i} = \begin{cases} x^{i0}, & \text{if } \overline{x}^r_{C_i}=0 \\ x^{i1}, & \text{if } \overline{x}^r_{C_i}=1 \text{ and } i\neq r \\ x^{is}, & \text{if } \overline{x}^r_{C_i}=1 \text{ and } i=r, \end{cases} \qquad i=0,\ldots,2k.$$

($y^{rs}_{C_i}$ indexes the subvector of $y^{rs}\in\mathbb{R}^{\mathfrak{C}}$ with all components that correspond to the hole C_i.) In other words: We take each incidence vector $\overline{x}^r$ and substitute for each of its components $\overline{x}^r_{C_i}$, $i=0,\ldots,2k$, a vector x^{is}: If $\overline{x}^r_{C_i}=0$, we take x^{i0}, if $\overline{x}^r_{C_i}=1$, we take x^{i1}. The only exception to this procedure is coordinate r, where we do not only substitute x^{r1}, but try all possibilities x^{rs}. In all cases, however, $\pi(y^{rs})=\overline{x}^r$ for all $s=1,\ldots,|C_r|$.
This results in a total of $\sum_{i=0}^{2k}|C_i|$ vectors y^{rs}. It is easy to see that these "expansions of stable sets by stable sets" are incidence vectors of stable sets in $\mathfrak{C}$ and that they are tight for the hole of holes inequality under consideration. We claim that they are also affinely independent. For suppose not; then there are multipliers λ_{rs}, not all zero, that sum up to zero such that $\sum_{rs}\lambda_{rs}y^{rs}=0$. But this implies that

$$\sum_{rs}\lambda_{rs}\pi(y^{rs}) = \sum_{rs}\lambda_{rs}\overline{x}^r = \sum_{r=0}^{2k}\left(\sum_{s=1}^{|C_r|}\lambda_{rs}\right)\overline{x}^r = 0,$$

and affine independence of the aggregated vector $\overline{x}^r$ yields

$$\sum_{s=1}^{|C_r|}\lambda_{rs} = 0, \qquad r=0,\ldots,2k. \tag{2.5}$$

Considering the rows of $\sum_{rs}\lambda_{rs}y^{rs}=0$ that correspond to the individual holes C_r, we obtain

$$\sum_{is}\lambda_{is}y^{is}_{C_r} = 0, \qquad r=0,\ldots,2k.$$

As for $i\neq r$ the vectors $y^{is}_{C_r}=x^{i1}$ are constant for all s, these equations simplify to

$$\sum_{is}\lambda_{is}y^{is}_{C_r} = \sum_{\substack{i=0,\ldots,2k\\ i\neq r}} x^{i1}\underbrace{\sum_{s=1}^{|C_i|}\lambda_{is}}_{=0,\text{ see (2.5)}} + \sum_{s=1}^{|C_r|}\lambda_{rs}y^{rs}_{C_r} = \sum_{s=1}^{|C_r|}\lambda_{rs}x^{rs} = 0, \qquad r=0,\ldots,2k$$

and imply $\lambda=0$, a contradiction. Thus, the incidence vectors y^{rs} were indeed affinely independent and the hole of holes inequality facet inducing for its support. □

2.5.3 Chain Inequalities

We have seen in the preceding subsections a variety of derivations of classes of inequalities from cycle and clique inequalities of appropriate set packing relaxations. To give an example of a different combinatorial type, we show in this subsection that a family of chain inequalities that were introduced by Tesch [1994] can be seen as strengthened (see page 65) expansions of Nemhauser & Trotter [1973]'s antiweb inequalities, see also Laurent [1989].
A $2k+1$-*chain* C is similar to a 2-chorded cycle with $2k+1$ nodes $0,\ldots,2k$; the difference is that the two chords $(0,2k-1)$ and $(1,2k)$ are replaced with the single edge $(1,2k-1)$, see Figure 2.20. An *antiweb* $C(k,t)$ is a graph on k nodes $0,\ldots,k-1$, such that any t successive nodes $i,i+1,\ldots,i+t-1$ form a clique, see Figure 2.21. Chains are very similar to 2-chorded cycles; these, in turn, coincide with the class of antiwebs $C(2k+1,3)$.
Chains and antiwebs give rise to inequalities for the set packing polytope. The *chain* and *antiweb inequalities* state that

$$\sum_{i\in C} x_i \le \left\lfloor \frac{2k+2}{3} \right\rfloor \qquad \sum_{i\in C(k,t)} x_i \le \left\lfloor \frac{k}{t} \right\rfloor .$$

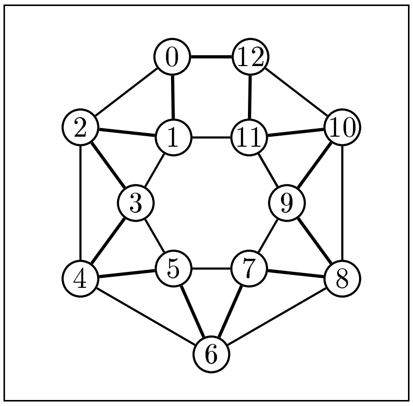


Figure 2.20: A 13-Chain.

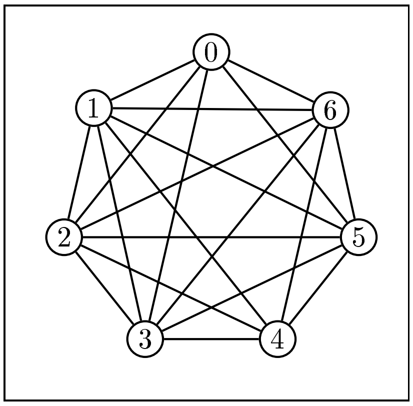


Figure 2.21: The Antiweb $C(7,3)$.

2.5.8 Theorem (Chain Inequalities)
Let C be $2k+1$-chain, P_{SSP} the corresponding set packing polytope, $\mathfrak{G}=(\mathfrak{V},\mathfrak{E})$ the rank conflict graph, and $\breve{P}_{\text{SSP}}(\mathfrak{G})$ the rank set packing relaxation of P_{SSP}. Then:

Every chain inequality for P_{SSP} is the expansion of a strengthened antiweb inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G})$.

Proof.
Consider in $\mathfrak{G}$ the $2k-1$ nodes

$$\mathfrak{v}_1 := G[\{1,2k\}] \quad \mathfrak{v}_i := G[\{i\}], \quad i=2,\ldots,2k-2, \quad \text{and} \quad \mathfrak{v}_{2k-1} := G[\{2k-1,0\}],$$

and let $\mathfrak{W} := \{\mathfrak{v}_1,\ldots,\mathfrak{v}_{2k-1}\}$. The reader verifies that $\mathfrak{W}$ induces an antiweb in $\mathfrak{G}$, more precisely,

$$\mathfrak{G}[\mathfrak{W}] = C(2k-1,3).$$

An expansion of the antiweb inequality corresponding to $\mathfrak{G}[\mathfrak{W}]$ yields

$$\begin{aligned}
&\sum_{i=1}^{2k-1} \pi_{\mathfrak{v}_i}(x) \le \left\lfloor \frac{2k-1}{3} \right\rfloor \\
\iff\quad & (x_1 + x_{2k} - 1) + \sum_{i=2}^{2k-2} x_i + (x_{2k-1} + x_0 - 1) \le \left\lfloor \frac{2k-1}{3} \right\rfloor \\
\iff\quad & \sum_{i=0}^{2k} x_i \le \left\lfloor \frac{2k+2}{3} \right\rfloor + 1.
\end{aligned}$$

A final strengthening of this inequality (reducing the right-hand side by one, see page 65) yields the desired chain inequality. The validity of the strengthening can be inferred in a similar way as in the proof of Theorem 2.4.6. □

2.5.4 Some Composition Procedures

While the examples of the preceding subsections had *analytic* flavour, we study in this subsection applications of set packing relaxations to *constructive* approaches to the stable set polytope. Our result is that certain composition procedures of the literature have a natural interpretation in terms of set packing relaxations.

The general principle behind *composition approaches* is the following: One considers some *graph theoretic operation* to construct a complex graph G from one or more simpler ones, and investigates the polyhedral consequences of this operation. Such consequences can be (i) to obtain analogous operations to construct valid or facet defining inequalities for G from known ones for the original graphs, or, in rare cases, (ii) to obtain a complete description of $\check{P}_{\text{SSP}}(G)$ from likewise complete descriptions of the anti-dominants of the set packing polytopes associated to the original graphs. A survey on composition methods for the set packing problem can be found in Section 2.5 of this thesis.

Operations of type (i) that give rise to facets are called *facet producing procedures* and we study three examples of this type in the remainder of this subsection (we investigate only their validity). The graph theoretic composition technique behind all of them is node *substitution* (in different variants): Given is some graph $\overline{G}$; replacing one or several nodes by graphs and the affected edges by appropriate sets of edges, one obtains a new graph G. The facet producing procedure associated to such a substitution translates valid/facet defining inequalities for $\check{P}_{\text{SSP}}(\overline{G})$ into valid/facet defining inequalities for $\check{P}_{\text{SSP}}(G)$.

This concept has an obvious relation to *expansion*. Namely, consider the expansion

$$\overline{a}^{\mathrm{T}}\overline{x} \le \overline{\alpha} \iff \overline{a}^{\mathrm{T}}\pi(x) \le \overline{\alpha}$$

of an inequality for the rank relaxation $\check{P}_{\text{SSP}}(\mathfrak{G})$ of some graph G: One obtains the support graph $G[\operatorname{supp}\overline{a}^{\mathrm{T}}\Pi]$ of the expansion from the support graph $\mathfrak{G}[\operatorname{supp}\overline{a}^{\mathrm{T}}]$ of the aggregated inequality by a sequence of node substitutions and identifications. Constructing inequalities in this way means thus to look at a given graph $\overline{G}$ as the conflict graph (or a subgraph of it) of some graph G that can be constructed from $\overline{G}$:

$$\text{construct } G \text{ such that } \qquad \overline{G} = \mathfrak{G}(G).$$

This technique —to *start* with the conflict graph and *construct* a suitable original graph— is our interpretation of composition in terms of aggregation.

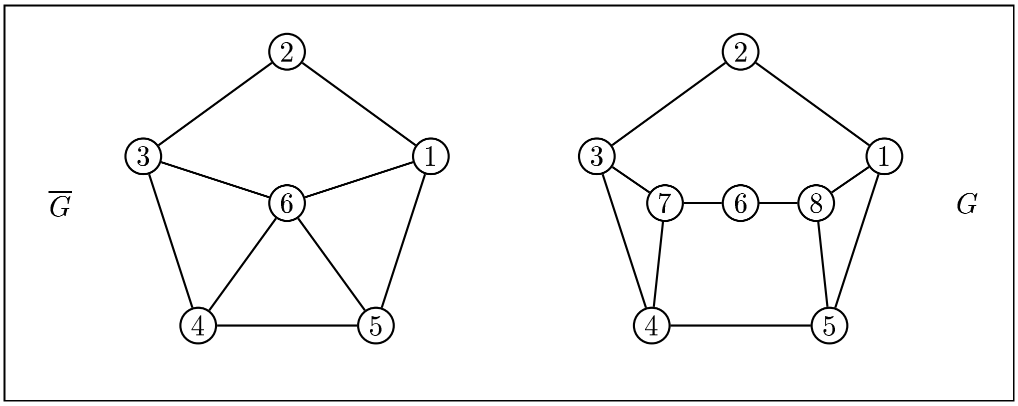


Figure 2.22: Applying a Composition Procedure.

We turn now to the examples and start with a procedure of Wolsey [1976]. Given a graph $\overline{G} = (\overline{V}, \overline{E})$ with nodes $\overline{V} = \{1, \ldots, n\}$, the operation constructs a new graph $G = (V, E)$ from $\overline{G}$ by replacing node n with a path $(n+1, n, n+2)$ involving two new nodes $n+1$ and $n+2$, such that $n+1$ is adjacent to some subset γ_1 of neighbors of the "old" node n, while $n+2$ is adjacent to the remaining neighbors. Figure 2.22 shows an example where node 6 of a graph is replaced by the path $(7, 6, 8)$ and the new node 7 is connected to the old neighbors $\gamma_1 = \{3, 4\}$ of 6. The procedure asserts that, if $\overline{a}^{\mathrm{T}}\overline{x} \leq \overline{\alpha}$ was a valid inequality for $\breve{P}_{\mathrm{SSP}}(\overline{G})$, the constraint

$$\overline{a}^{\mathrm{T}}x + \overline{a}_n x_{n+1} + \overline{a}_n x_{n+2} \leq \alpha + a_n \tag{2.6}$$

holds for $\breve{P}_{\mathrm{SSP}}(G)$. This inequality can be obtained from a rank relaxation of G that involves the aggregation scheme $\pi : \mathbb{R}^V \to \mathbb{R}^{\overline{V}}$ defined as

$$\pi_i(x) = \begin{cases} x_i, & i \neq n \\ x_{n+1} + x_n + x_{n+2} - 1, & i = n. \end{cases}$$

π maps each node onto itself except for the path $(n, n+1, n+2)$ which is aggregated into a single node. One easily checks

2.5.9 Lemma (Composition Procedure I) $\overline{G} = \mathfrak{G}(G)$.

An expansion of $\overline{a}^{\mathrm{T}}\overline{x} \leq \overline{\alpha}$ yields inequality (2.6). We remark that this argument does not show that this procedure translates facets into facets.

Our second example is due to Wolsey [1976] and Padberg [1977]. The procedure joins an additional node $2n+1$ to all nodes $1, \ldots, n$ of the given graph $\overline{G} = (\overline{V}, \overline{E})$, and the graph $G = (V, E)$ arises from this join by subdividing each of the new edges $(2n+1, i)$ with a node $n+i$. In this case, the inequality $\overline{a}^{\mathrm{T}}\overline{x} = \sum_{i=1}^{n} \overline{a}_i\overline{x}_i \leq \overline{\alpha}$ for $\breve{P}_{\mathrm{SSP}}(\overline{G})$ gives rise to the constraint

$$\sum_{i=1}^{n} \overline{a}_i(x_i + x_{n+i}) + \left(\sum_{i=1}^{n} \overline{a}_i - \overline{\alpha}\right) x_{2n+1} \leq \sum_{i=1}^{n} \overline{a}_i \tag{2.7}$$

for $\breve{P}_{\mathrm{SSP}}(G)$ (and is, in fact, even facet inducing, if $\overline{a}^{\mathrm{T}}\overline{x} \leq \overline{\alpha}$ was). Figure 2.23 shows an application of this technique to the graph $\overline{G}$ on the left side with nodes $1, \ldots, 5$ (adding the grey node 6 in the middle results in a certain graph $\overline{G}^*$ that will be explained in a second).

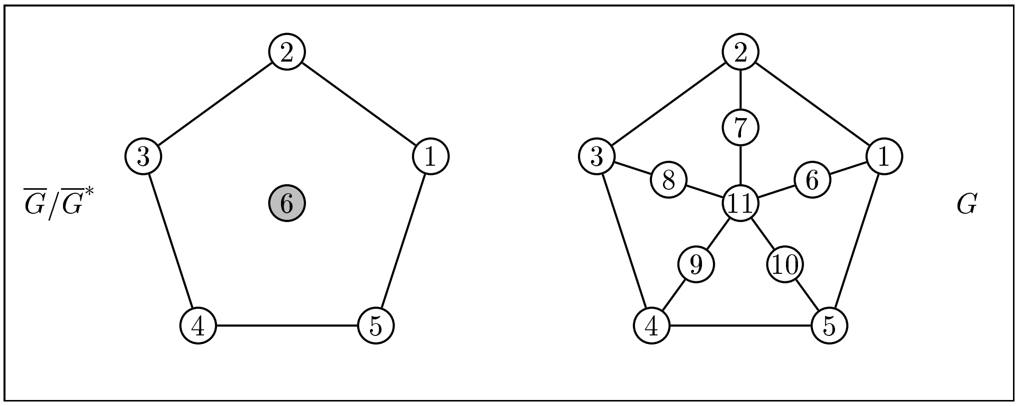


Figure 2.23: Another Composition Procedure.

To obtain inequality (2.7) from a rank relaxation, we consider the scheme $\pi : \mathbb{R}^V \to \mathbb{R}^{\overline{V}^*}$ defined as

$$\pi_i(x) := \begin{cases} x_i + x_{n+i} + x_{2n+1} - 1, & i = 1, \ldots, n \\ x_{2n+1}, & i = n+1, \end{cases}$$

where we have set $\overline{G}^* := (\overline{V}^*, \overline{E}^*) := (\overline{V} \cup \{n+1\}, \overline{E})$.

2.5.10 Lemma (Composition Procedure II) $\overline{G}^* = \mathfrak{G}(G)$.

In other words, the conflict graph $\overline{G}^*$ of G coincides with $\overline{G}$ *augmented* by an additional node $n+1$ that is not connected to any other node. Obviously, any inequality $\sum_{i=1}^n \overline{a}_i \overline{x}_i \leq \overline{\alpha}$ that is valid for $\breve{P}_{\text{SSP}}(\overline{G})$ is also valid for $\breve{P}_{\text{SSP}}(\overline{G}^*)$.

It is now not true that an expansion of the inequality $\sum_{i=1}^n \overline{a}_i \overline{x}_i \leq \overline{\alpha}$ for $\breve{P}_{\text{SSP}}(\overline{G}^*)$ yields the desired inequality (2.7), but we get it with one additional "strengthening type" argument. This argument is that if $\sum_{i=1}^n \overline{a}_i \overline{x}_i \leq \overline{\alpha}$ is valid for $\breve{P}_{\text{SSP}}(\overline{G}^*)$, the stronger inequality

$$\sum_{i=1}^n \overline{a}_i \overline{x}_i \leq \overline{x}_{n+1} \overline{\alpha}$$

is perhaps no longer valid for $\breve{P}_{\text{SSP}}(\overline{G}^*)$, but it is valid for $\pi(\breve{P}_{\text{SSP}}(G))$. An expansion of this inequality yields the inequality (2.7) of interest (but again no facetial result).

As an example of a much more general composition technique, we consider now the *substitution* of a node v of $\overline{G}$ by some graph $\overline{G}_v$, such that the resulting graph $G = (V, E)$ is the union of $\overline{G} - v$ and $\overline{G}_v$, with all nodes of $\overline{G}_v$ joined to all neighbors of v in $\overline{G}$. Substitutions of this type were studied by Chvátal [1975], who showed not only that if $\overline{a}^{\mathrm{T}}\overline{x} \leq \overline{\alpha}$ is a facet of $\breve{P}_{\text{SSP}}(\overline{G})$ and $\overline{b}^{\mathrm{T}}\overline{x} \leq \overline{\beta}$ a facet of $\breve{P}_{\text{SSP}}(\overline{G}_v)$, the inequality

$$\overline{a}_v \sum_{u \in V(\overline{G}_v)} \overline{b}_u x_u + \sum_{\substack{u \in \overline{V} \\ u \neq v}} \overline{a}_u \overline{\beta} x_u \leq \overline{\alpha}\overline{\beta} \tag{2.8}$$

is valid for $\breve{P}_{\text{SSP}}(G)$, but that all facets of $\breve{P}_{\text{SSP}}(G)$ are of this form. Note that this operation subsumes the famous *multiplication* of a node to a clique of Fulkerson [1972] and Lovász [1971], that plays an important role in studying the polyhedra associated to *perfect graphs*.

To derive the validity of inequalities (2.8) for fixed but arbitrary $v \in \overline{V}$ and $\overline{b}^{\mathrm{T}}\overline{x} \leq \overline{\beta}$ as above from a set packing relaxation, we consider the aggregation scheme $\pi : \mathbb{R}^V \to \mathbb{R}^{\overline{V}}$ given as

$$\pi_u(x) := \begin{cases} x_u, & u \neq v \\ \frac{\sum_{w \in V(\overline{G}_v)} \overline{b}_w x_w}{\overline{\beta}} = \frac{\overline{b}^{\mathrm{T}} x}{\overline{\beta}}, & u = v. \end{cases}$$

π is bounded by one in every component, integral in all coordinates different from v, but *not* integral in v and in particular *not* a rank aggregation (π is our only non-rank example in this section). But if $\overline{b}^{\mathrm{T}}\overline{x} \leq \overline{\beta}$ is a *support* of $\breve{P}_{\mathrm{SSP}}(\overline{G}_v)$, i.e., there is an incidence vector $\overline{x}$ of a stable set in $\overline{G}_v$ such that the inequality $\overline{b}^{\mathrm{T}}\overline{x} \leq \overline{\beta}$ is tight, the aggregate $\pi(\breve{P}_{\mathrm{SSP}}(G))$ has not only integer and thus zero-one vertices only, but, in fact

2.5.11 Lemma (Composition Procedure III) $\pi(\breve{P}_{\mathrm{SSP}}(G)) = \breve{P}_{\mathrm{SSP}}(\overline{G})$.

Once this relation is established, an expansion of the inequality $\overline{a}^{\mathrm{T}}\overline{\beta}\overline{x} \leq \overline{\alpha}\overline{\beta}$ yields Chvátal's inequality (2.8) (but not an equivalent result about complete descriptions).

Proof (of Lemma 2.5.11).
We prove first that $\pi(\breve{P}_{\mathrm{SSP}}(G))$ is integral. The proof is by contradiction, i.e., suppose $\pi(\breve{P}_{\mathrm{SSP}}(G))$ is not integral. Then there must be a non integer vertex $\pi(x^0)$, where x^0 is a vertex of $\breve{P}_{\mathrm{SSP}}(G)$. Note that $x^0 \geq 0$ and so must be $\pi(x^0)$. The only fractional component of $\pi(x^0)$ can be $\pi_v(x^0)$ and it must be nonnull. By assumption, there exist incidence vectors $\overline{y}^0$ and $\overline{y}^1$ of stable sets in $\overline{G}_v$ such that

$$\overline{b}^T\overline{y}^0 = 0 \qquad \text{and} \qquad \overline{b}^T\overline{y}^1 = \overline{\beta}.$$

The vectors x^1 and x^2 that arise from x^0 by replacing $x^0_{\overline{G}_v}$ with $\overline{y}^0$ and $\overline{y}^1$ are again incidence vectors of stable sets in G, because $x^0_{\overline{G}_v}$ is nonnull and the stable set $\operatorname{supp} x^0$ has a node in $\overline{G}_v$. But then

$$\pi(x^0) = (\overline{\beta} - \overline{b}^{\mathrm{T}}x^0)/\overline{\beta}\ \pi(x^1) + (\overline{b}^{\mathrm{T}}x^0)/\overline{\beta}\ \pi(x^2)$$

is not a vertex, a contradiction.
The last step to establish $\pi(\breve{P}_{\mathrm{SSP}}(G)) = \breve{P}_{\mathrm{SSP}}(\overline{G})$ is to note that $\pi_u(x) + \pi_w(x) \leq 1$ holds for any vertex $\pi(x) \in \{0,1\}^V$ of $\pi(\breve{P}_{\mathrm{SSP}}(G))$ if and only if $uw \in E(\overline{G})$. □

2.6 The Set Covering Problem

We propose in this section a set packing relaxation of the set covering problem that gives rise to polynomially separable classes of inequalities. This is important for two reasons: (i) Set covering deals with general independence systems, see Section 1.3 of this thesis, while many problems in combinatorial optimization arise from special independence systems; hence, the set covering results carry over. Unfortunately, however, (ii) very few classes of (polynomial time) separable inequalities for the set covering problem are known; we are only aware of the odd hole inequalities, see Subsection 1.9.1, Nobili & Sassano [1992]'s *k-projection* cuts, Balas [1980]'s *conditional cuts*, and certain classes of $\{0, \frac{1}{2}\}$ Chvátal-Gomory cuts, see Caprara & Fischetti [1996], see also Schulz [1996, Section 4.6] for some classes of this type.

The need to develop cutting planes for the set covering polytope was the starting point for our work on set packing relaxations. We have *implemented* one version of such a procedure for use in a branch-and-cut algorithm for set partitioning problems; details about and computational experience with this routine are reported in Chapter 3 of this thesis.
The *set covering problem* (SCP) is the integer program

$$\text{(SCP)} \quad \min\ w^{\mathrm{T}}x \quad Ax \geq \mathbf{1},\ x \in \mathbb{Z}_+^n,$$

where $A \in \{0,1\}^{m\times n}$ and $w \in \mathbb{Z}_+^n$. The associated polyhedron is denoted in this section by P_{SCP} or $P_{\mathrm{SCP}}(A)$. For a survey on set covering, see Chapter 1.
The *set packing relaxation* for (SCP) that we suggest is based on an exponential conflict graph $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$ that records pairwise conflicts of substructures of the matrix A. We take the set $\mathfrak{V} := 2^{\{1,\ldots,n\}}$ (where 2^S denotes the *powerset* of some set S) of all subsets of column(indice)s of A as the nodes of $\mathfrak{G}$ and define an aggregation scheme $\pi : \mathbb{R}^n \to \mathbb{R}^{\mathfrak{V}}$ as

$$\pi_J(x) := 1 - \sum_{j\in J} x_j \qquad \forall J \subseteq \{1,\ldots,n\}.$$

We draw an edge between two (not necessarily disjoint) sets I and J of columns when their union covers a row of A, or, equivalently, some variable in $I \cup J$ has to be set to one:

$$IJ \in \mathfrak{E} \iff \exists i : I \cup J \supseteq A_{i\cdot} \iff \pi_I(x) + \pi_J(x) \leq 1 \quad \forall x \in P_{\mathrm{SCP}} \cap \mathbb{Z}^n.$$

2.6.1 Lemma (Set Packing Relaxation of the SCP) $\pi(P_{\mathrm{SCP}}) \subseteq \check{P}_{\mathrm{SSP}}(\mathfrak{G})$.

This set packing relaxation has been considered by Sekiguchi [1983] in a special case. He studies 0/1 matrices A with the property that there is a partition $\mathfrak{W}$ of the column(indice)s $\dot{\bigcup}_{\mathfrak{v}\in\mathfrak{W}} \mathfrak{v} = \{1,\ldots,n\}$ into nonempty column sets $\mathfrak{v}$ such that (the support of) each row $A_{r\cdot}$ is the union of exactly two such column sets, i.e., $\forall A_{r\cdot} : \exists \mathfrak{u}, \mathfrak{v} \in \mathfrak{W} : \mathfrak{u} \neq \mathfrak{v} \ \wedge\ \operatorname{supp} A_{r\cdot} = \mathfrak{u} \,\dot\cup\, \mathfrak{v}$. Figure 2.24 shows an example of a 0/1 matrix that has such a *Sekiguchi partition*.

$$A = \begin{array}{c} \\ 1\\2\\3\\4\\5 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1&2&3&4&5&6&7&8&9 \end{array} \\ \left(\begin{array}{cc|c|cc|c|c|cc} 1&1&1&.&.&.&.&.&. \\ .&.&1&1&1&.&.&.&. \\ .&.&.&1&1&1&.&.&. \\ 1&1&.&.&.&1&.&.&. \\ .&.&.&.&.&.&1&1&1 \end{array}\right) \end{array} \qquad \begin{array}{r} \mathfrak{W} = \{1,2\} \,\dot\cup\, \{3\} \,\dot\cup\, \{4,5\} \,\dot\cup\, \{6\} \\ \dot\cup\, \{7\} \,\dot\cup\, \{8,9\} \end{array}$$

Figure 2.24: A Sekiguchi Partitionable 0/1 Matrix.

Using essentially the same technique as we did to prove Proposition 2.5.7 (about the faceteness of hole of holes inequalities), Sekiguchi [1983] shows that for a 0/1 matrix A that has a Sekiguchi partition $\mathfrak{W}$ it is not only true that

$$\pi(P_{\mathrm{SCP}}) = \check{P}_{\mathrm{SSP}}(\mathfrak{G}[\mathfrak{W}]),$$

but, even more, that the facets of P_{SCP} are *exactly* the expansions of the facets of $\check{P}_{\mathrm{SSP}}(\mathfrak{G}[\mathfrak{W}])$. We remark that Sekiguchi considers in his proof the, as we would say, aggregation scheme $\overline{\pi} : \mathbb{R}^n \to \mathbb{R}^{\mathfrak{W}}$ defined as

$$\overline{\pi}_{\mathfrak{v}}(x) := \sum_{i\in\mathfrak{v}} x_i \qquad \forall \mathfrak{v} \in \mathfrak{W},$$

that is "complementary" to ours in the sense that $\pi + \overline{\pi} \equiv \mathbf{1}$.

We mention the *odd hole inequalities* for the SCP, see, e.g., Cornuéjols & Sassano [1989], as one example for a class of inequalities that can be obtained from a set packing relaxation *in the sense of Sekiguchi.*
In this context of set covering, the term *odd hole* is commonly used to refer to the edge-node incidence matrix $A(2k+1,2) = A(C(2k+1,2)) \in \mathbb{R}^{(2k+1)\times(2k+1)}$ of the circulant $C(2k+1,2)$:

$$A(2k+1,2)_{ij} = \begin{cases} 1, & \text{if } j = i \text{ or } j = i+1 \pmod{2k+1}, \\ 0, & \text{else.} \end{cases}$$

The associated *odd hole inequality* asserts that

$$\sum_{i=1}^{2k+1} x_i \geq k+1$$

is valid for $P_{\mathrm{SCP}}(A(2k+1,2))$.

2.6.2 Proposition
Let $A(2k+1,2)$ be an odd hole, P_{SCP} the corresponding set covering polyhedron, $\mathfrak{G}$ the conflict graph associated to $A(2k+1,2)$, and $\breve{P}_{\mathrm{SSP}}(\mathfrak{G}[\mathfrak{W}])$ the Sekiguchi relaxation of P_{SCP}, where $\mathfrak{W} = \{\{i\} \mid i = 1,\ldots,2k+1\}$ is the (unique) Sekiguchi partition of $A(2k+1,2)$. Then:
The odd hole inequality associated to P_{SCP} is the expansion of an odd cycle inequality for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G}[\mathfrak{W}])$.
We omit the simple proof of this proposition and proceed with an example of an expanded cycle inequality that can not be obtained from a Sekiguchi relaxation; we call this larger class of (inequalities from expansions of) cycle inequalities for $\mathfrak{G}$ *aggregated cycle inequalities.*

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & . & . & . & . & . \\ . & . & . & 1 & 1 & 1 & . & . & . \\ . & . & . & 1 & 1 & 1 & 1 & . & . \\ . & . & . & . & . & . & 1 & 1 & 1 \\ 1 & 1 & . & . & . & . & . & 1 & 1 \end{pmatrix} \end{array}$$

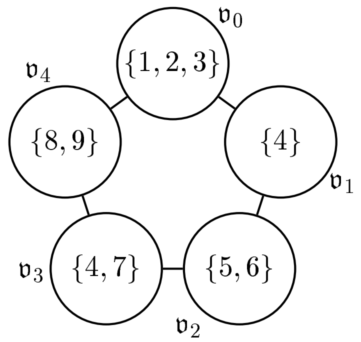


Figure 2.25: A Not Sekiguchi Partitionable 0/1 Matrix and an Aggregated 5-Cycle.

The matrix A on the left of Figure 2.25 gives rise to a 5-cycle $\mathfrak{C}$ in $\mathfrak{G}$ formed by the nodes $\mathfrak{v}_0 = \{1,2,3\}$, $\mathfrak{v}_1 = \{4\}$, $\mathfrak{v}_2 = \{5,6\}$, $\mathfrak{v}_3 = \{4,7\}$, and $\mathfrak{v}_4 = \{8,9\}$. A is not Sekiguchi partitionable, because row $A_{1\cdot}$ calls for the sets $\operatorname{supp} A_{1\cdot} \setminus \operatorname{supp} A_{5\cdot} = \{3,4\}$ and $\operatorname{supp} A_{1\cdot} \cap \operatorname{supp} A_{3\cdot} = \{4\}$ as elements of the partition, but these sets are not disjoint. An expansion of the odd cycle inequality corresponding to $\mathfrak{C}$ yields

$$\begin{aligned} & \sum_{i=0}^{4} \pi_{\mathfrak{v}_i}(x) \leq 2 \\ \iff \quad & (1 - x_1 - x_2 - x_3) + (1 - x_4) + (1 - x_5 - x_6) + (1 - x_4 - x_7) + (1 - x_8 - x_9) \leq 2 \\ \iff \quad & x_1 + x_2 + x_3 + 2x_4 + x_5 + x_6 + x_7 + x_8 + x_9 \geq 3. \end{aligned}$$

Turning back to general case and looking at the *separation* of inequalities for P_{SCP} from the set packing relaxation $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$, we can obtain polynomially separable classes by restricting attention to node induced subgraphs $\mathfrak{G}[\mathfrak{W}]$ of the conflict graph of polynomial size. A *heuristic* way to do this is to split the support of each row $A_{i\cdot}$ into two "equal sized halves"

$$\operatorname{supp} A_{i\cdot} =: I \cupdot \overline{I}$$

with respect to a given fractional covering x, i.e., we split such that $A_{iI}x_I \approx A_{i\overline{I}}x_{\overline{I}}$, and take $\mathfrak{W}$ as the set of these "halves":

$$\mathfrak{W} := \{I, \overline{I} \mid i = 1, \ldots, m\}.$$

The idea behind this procedure is to (i) obtain a "reasonable" (polynomial) number of $2m$ nodes (in fact, in our computations a lot of these always turned out to be identical) with values of $\pi_I(x), \pi_{\overline{I}}(x)$ close to $\frac{1}{2}$ that lead (ii) with some probability not only to a significant number of edges at all, but (hopefully) even to "tight edges" of the set packing relaxation, which in turn (iii) offers some potential to identify violated inequalities. We have *implemented* this procedure to separate aggregated cycle inequalities in a branch-and-cut code for set partitioning problems; for more implementation details and *computational experience* with this routine see Chapter 3 of this thesis.

Another separation idea that suggests itself would be to derive inequalities from *submatrices* of A. But in contrast to the set packing case, such inequalities are in general only valid for their row and/or column support. They have to be *lifted* to become globally valid and we do not know how to derive efficiently separable classes of inequalities in this way.

We close this subsection with an attempt to demonstrate the *flexibility* of the concept of the set packing relaxation $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ by stating a result of Balas & Ho [1980] on *cutting planes from conditional bounds* in "set packing relaxation terminology".

Balas & Ho assume that some upper bound z_u on the optimum objective value of the set covering program (SCP) is known. In this situation, they consider some family of variable index sets $\mathfrak{W} \subseteq \mathfrak{V}$ and investigate conditions that ensure that at least one of the corresponding aggregated variables $\pi_{\mathfrak{v}}(x)$, $\mathfrak{v} \in \mathfrak{W}$, has a value of one for any solution x *with a better objective value than* z_u. If this condition can be established (Balas & Ho [1980] suggest arguments and algorithms based on LP duality), the *conditional cut*

$$\sum_{i \in \bigcup_{\mathfrak{v} \in \mathfrak{W}} \operatorname{supp} A_{r(\mathfrak{v})\cdot} \setminus \mathfrak{v}} x_i \geq 1$$

can be used as a cutting plane. Here, for each column set $\mathfrak{v}$, $A_{r(\mathfrak{v})\cdot}$ is an arbitrary row of A. Note that conditional cuts are again of set covering type.

2.7 The Multiple Knapsack Problem

In this section, we investigate a set packing relaxation of the multiple knapsack problem in an exponential space. It will turn out that the validity of certain classes of extended cover and combined cover inequalities can be explained in terms of a single conflict of two "item-knapsack configurations". As references to the multiple knapsack problem we give Wolsey [1990], the textbook Martello & Toth [1990, Chapter 6], Ferreira, Martin & Weismantel [1996], see also the thesis of Ferreira [1994], and the survey article Aardal & Weismantel [1997].

The *multiple knapsack problem* (MKP) is the integer program

$$\begin{array}{llrcl}
\text{(MKP)} & \max & \sum_{i\in I}\sum_{k\in K} w_i x_{ik} & & \\
& \text{(i)} & \sum_{i\in I} a_i x_{ik} & \le & \alpha \quad \forall k \in K \\
& \text{(ii)} & \sum_{k\in K} x_{ik} & \le & 1 \quad \forall i \in I \\
& \text{(ii)} & x & \in & \{0,1\}^{I\times K}.
\end{array}$$

Here, $I = \{1, \ldots, n\}$ is a set of *items* of nonnegative integer weights and profits $a, w \in \mathbb{Z}_+^I$, that can be stored in a set K of *knapsacks* of capacity α each. Associated with the MKP is the multiple knapsack polytope P_{MKP}.
The set packing relaxation that we propose involves the following conflict graph $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$. $\mathfrak{G}$ has the set $\mathfrak{V} = 2^{I\times K}$ (where 2^S denotes the *powerset* of some set S) of all sets of possible "item-knapsack pairs" as its nodes. We will call such a set of item-knapsack pairs a(n item-knapsack) *configuration*. To define the edges of the conflict graph, we consider the aggregation scheme $\pi : \mathbb{R}^{I\times K} \to \mathbb{R}^{\mathfrak{V}}$ defined as

$$\pi_{\mathfrak{v}}(x) = \Big(\sum_{ik\in\mathfrak{v}} x_{ik}\Big) - \big(|I(\mathfrak{v})| - 1\big).$$

Here, $I(\mathfrak{v}) = \{i \in I : \exists k \in K : ik \in \mathfrak{v}\}$ denotes the set of items that appear somewhere in the configuration $\mathfrak{v}$. $\pi_{\mathfrak{v}}(x)$ is one for some solution x of (MKP) if and only if x assigns *all* items in $\mathfrak{v}$ to feasible knapsacks with respect to $\mathfrak{v}$, i.e., all items $i \in I(\mathfrak{v})$ of the configuration satisfy $x_{ik} = 1$ for some $(i,k) \in \mathfrak{v}$. Two configurations $\mathfrak{u}$ and $\mathfrak{v}$ are in conflict and we draw an edge between them if $\pi_{\mathfrak{u}}(x) + \pi_{\mathfrak{v}}(x) \le 1$ holds for all $x \in P_{\text{MKP}} \cap \mathbb{Z}^{I\times K}$, i.e., it is not possible to simultaneously assign all items in $\mathfrak{u}$ and $\mathfrak{v}$ to feasible knapsacks.

2.7.1 Lemma (Set Packing Relaxation of the MKP) $\pi(P_{\text{MKP}}) \subseteq \breve{P}_{\text{SSP}}(\mathfrak{G})$.

We show now that the classes of *extended cover inequalities* and *combined cover inequalities* of Ferreira, Martin & Weismantel [1996] arise from expansions of *edge inequalities* of this set packing relaxation; our discussion refers to Ferreira [1994]'s description of these inequalities. An *extended cover inequality* involves two configurations $\mathfrak{v}'$ and $\mathfrak{v}''$ of the form

$$\mathfrak{v}' = I' \times \{k_1, k_2\} \qquad \text{and} \qquad \mathfrak{v}'' = I'' \times \{k_2\}$$

with two knapsacks k_1 and k_2 and two sets of items I' and I''. In this situation, it is in general not true that $\pi_{\mathfrak{v}'}(x) + \pi_{\mathfrak{v}''}(x) \le 1$ holds, but one can look for combinatorial conditions that ensure this inequality. Ferreira [1994, page 74] assumes that

(i) I' forms a *cover* for knapsack k_1, i.e., the items in I' do not all fit into k_1, and that
(ii) $I'' \cup \{i\}$ forms a cover for knapsack k_2 for each item $i \in I'$.

(Actually, he assumes also $I' \cap I'' = \emptyset$.) (i) means that if $\pi_{\mathfrak{v}'}(x) = 1$, i.e., all items I' of the first configuration are assigned to the knapsacks k_1 and k_2, then at least one item of I' must be assigned to k_2, and then $\pi_{\mathfrak{v}''}(x) \le 0$ due to (ii). This implies the validity of the edge inequality $\pi_{\mathfrak{v}'}(x) + \pi_{\mathfrak{v}''}(x) \le 1$ that expands into the extended cover inequality

$$\sum_{i\in I'} x_{ik_1} + \sum_{i\in I'} x_{ik_2} + \sum_{i\in I''} x_{ik_2} \le |I'| + |I''| - 1.$$

Of similar flavour are the *combined cover inequalities.* This time, the configurations are

$$\mathfrak{v}' = (I_1 \times \{k_1\}) \cup (I_2 \times \{k_2\}) \cup (I' \times \{k_3\}) \qquad \text{and} \qquad \mathfrak{v}'' = I'' \times \{k_3\},$$

with three different knapsacks k_1, k_2, and k_3 and satisfying (confer Ferreira [1994, page 82])

(i) $I' = I_1 \cup I_2$ and $|I_1 \cap I_2| = 1$,

(ii) I_1 is a cover for k_1, I_2 a cover for k_2, and

(iii) $I'' \cup \{i\}$ is a cover for knapsack k_3 for each item $i \in I'$.

(Ferreira [1994] assumes again $I' \cap I'' = \emptyset$.) $\pi_{\mathfrak{v}'}(x) = 1$ and (i) together imply that at least one item from I' must be assigned to knapsack k_3, and then (ii) results in $\pi_{\mathfrak{v}''}(x) \leq 0$ as for the extended cover inequalities. Expanding the edge inequality $\pi_{\mathfrak{v}'}(x) + \pi_{\mathfrak{v}''}(x) \leq 1$, we obtain the combined cover inequality

$$\sum_{i \in I_1} x_{ik_1} + \sum_{i \in I_2} x_{ik_2} + \sum_{i \in I'} x_{ik_3} + \sum_{i \in I''} x_{ik_3} \leq |I'| + |I''| - 1.$$

The following theorem summarizes our results on extended and combined cover inequalities.

2.7.2 Theorem (Extended and Combined Cover Inequalities)
Let MKP be a multiple knapsack problem, P_{MKP} the associated multiple knapsack polytope, and $\breve{P}_{\text{SSP}}(\mathfrak{G})$ the set packing relaxation of P_{MKP}.

(i) Every extended cover inequality for P_{MKP} is the expansion of an edge inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G})$.

(ii) Every combined cover inequality for P_{MKP} is the expansion of an edge inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G})$.

2.8 The 0/1 Programming Problem with Nonnegative Data

We have seen in the previous sections examples of set packing relaxations for special combinatorial optimization problems. To give a perspective in a more general direction, we want to draw the reader's attention now to a set packing relaxation for a class of 0/1 integer programming problems that was suggested by Bixby & Lee [1993]. This construction assumes only nonnegativity of the constraint matrix, but no particular structure; it yields clique, odd cycle, etc. inequalities in the natural variables.

Set packing constraints of this type form one of the rare families of *structural cuts* for general integer programming problems, i.e., cuts that are derived by searching, detecting, and utilizing some *combinatorial structure* in an a priori unstructured constraint system. Set packing relaxations try to set up a conflict graph, the famous single knapsack relaxation of Crowder, Johnson & Padberg [1983] analyzes the diophantine structure of an individual row, Padberg, van Roy & Wolsey [1985]'s flow covers are based on combinatorial properties of fixed charge problems, and the feasible set inequalities of Martin & Weismantel [1997] come from intersections of several knapsacks. These classes of structural cuts are the first of only three types of tools to solve 0/1 integer programs by branch-and-cut. *Enumeration* is, unfortunately, the second and the third consists of *general cutting planes* for 0/1 integer programs: Gomory [1960]'s cuts, see Balas, Ceria, Cornuéjols & Natraj [1994] for an exciting recent renaissance, lift-and-project cuts, see Balas, Ceria & Cornuéjols [1993], or Lovász & Schrijver [1991]'s matrix cuts. To put it brief: There is significant interest in identifying further families of structural cuts for general integer programs.

One class of integer programs with an embedded set packing structure consists of 0/1 programs with *nonnegative constraint systems*

$$(\mathrm{IP}^+) \quad \max\ w^{\mathrm{T}}x \quad A^+x \leq b^+,\ x \in \{0,1\}^n.$$

Here, $A^+ \in \mathbb{Z}_+^{m\times n}$ and $b^+ \in \mathbb{Z}_+^m$ are a nonnegative integral matrix and right-hand side vector, and $w \in \mathbb{Z}^n$ is an integer objective; no further structural properties are assumed. The polytope associated to this program is denoted by $P_{\mathrm{IP}+}$.
Bixby & Lee [1993] propose for such programs the following "natural" set packing relaxation. The conflict graph $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$ of the relaxation has the column(indice)s of the matrix A^+, or, if we want, the variables, as its nodes, i.e., $\mathfrak{V} = \{1, \ldots, n\}$. The edges are defined in terms of the identity aggregation scheme $\pi : \mathbb{R}^n \to \mathbb{R}^n$ that has

$$\pi_i(x) := x_i, \qquad i = 1, \ldots, n.$$

There is an edge between two columns i and j if and only if not both of them can be contained in a solution to (IP^+) at the same time, i.e.,

$$ij \in \mathfrak{E} \iff A^+_{i\cdot} + A^+_{j\cdot} \not\leq b^+ \iff \pi_i(x) + \pi_j(x) \leq 1 \qquad \forall x \in P_{\mathrm{IP}+} \cap \mathbb{Z}^n.$$

2.8.1 Lemma (Set Packing Relaxation of IP$_+$, Bixby & Lee[1993]) $\pi(P_{\mathrm{IP}+}) \subseteq \breve{P}_{\mathrm{SSP}}(\mathfrak{G})$.

The natural set packing relaxation yields clique, odd cycle, and other set packing inequalities in the original variables. An extension of the natural set packing relaxation to the mixed integer case is currently studied by Atamturk, Nemhauser & Savelsbergh [1998].

Bibliography of Part 1

Aardal & Weismantel (1997). Polyhedral Combinatorics. In Dell'Amico, Maffioli & Martello [1997], chapter 3, pp. 31–44.

Atamturk, Nemhauser & Savelsbergh (1998). The Mixed Vertex Packing Problem. TLI Tech. Rep. 98-05[1], Georgia Inst. of Tech.

Balas (1980). Cutting Planes From Conditional Bounds: A New Approach to Set Covering. *Math. Prog. Study 12, 19–36.*

Balas, Ceria & Cornuéjols (1993). A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs. *Math. Prog. 58, 295–324.*

Balas, Ceria, Cornuéjols & Natraj (1994). Gomory Cuts Revisited. Mgmt. Sci. Res. Rep., GSIA, Carnegie Mellon University, Pittsburgh, PA.

Balas & Clausen (Eds.) (1995). *Integer Programming and Combinatorial Optimiziation*, Proc. of the 4th Int. IPCO Conf.

Balas, Cornuéjols & Kannan (Eds.) (1992). *Integer Programming and Combinatorial Optimiziation*, Proc. of the 2nd Int. IPCO Conf.

Balas & Ho (1980). Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization: A Computational Study. *Math. Prog. 12, 37–60.*

Balas & Ng (1989a). On the Set Covering Polytope: I. All the Facets with Coefficients in $\{0,1,2\}$. *Math. Prog. 43, 57–69.*

—— (1989b). On the Set Covering Polytope: II. Lifting the Facets with Coefficients in $\{0,1,2\}$. *Math. Prog. 45, 1–20.*

Balas & Padberg (1976). Set Partitioning: A Survey. *SIAM Rev. 18, 710–760.*

Balas & Padberg (1970). On the Set-Covering Problem. *Op. Res. 20, 1153–1161.*

—— (1975). On the Set-Covering Problem: II. An Algorithm for Set Partitioning. *Op. Res. 1, 74–90.*

Balas & Zemel (1977). Critical Cutsets of Graphs and Canonical Facets of Set-Packing Polytopes. *Math. of OR 2(1), 15–19.*

Balinski (1965). Integer Programming: Methods, Uses, Computation. *Mgmt. Sci. 12(3), 253–313.*

Barahona & Mahjoub (1994). Compositions of Graphs and Polyhedra II: Stable Sets. *SIAM J. on Disc. Math. 7, 359–371.*

Barahona & Mahjoub (1989). Compositions of Graphs and Polyhedra III: Graphs With No W_4 Minor. Rep. No. 89565-OR, Inst. für Ökonomie und OR, Rheinische Friedrich-Wilhelms-Univ. Bonn.

[1] Avail. at URL `http://akula.isye.gatech.edu/~atamturk/`

Bellman & Hall (Eds.) (1960). *Combinatorial Analysis*, Proc. of Symposia in Applied Mathematics, Providence, RI.

Bellmore & Ratliff (1971). Set Covering and Involutory Bases. *Mgmt. Sci. 18(3), 194–206.*

Berge (1961). Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. Wissenschaftliche Zeitschrift, Martin Luther Univ. Halle-Wittenberg.

—— (1971). Balanced Matrices. *Math. Prog. 2, 19–31.*

Birge & Murty (Eds.) (1994). *Mathematical Programming: State of the Art 1994*. Univ. of Michigan.

Bixby & Lee (1993). Solving a Truck Dispatching Scheduling Problem Using Branch-and-Cut. Tech. Rep. 93-37[2], Rice Univ. To appear in *Op. Res.*.

Bland, Huang & Trotter (1979). Graphical Properties Related to Minimal Imperfection. *Disc. Math. 27, 11–22.*

Boulala & Uhry (1979). Polytops des Independantes d'un Graph Serie-Parallele. *Disc. Math. 27, 225–243.*

Bowman & Starr (1973). Set Covering by Ordinal Cuts. I: Linear Objective Functions. Mgmt. Sci. Res. Rep. 321, Carnegie Mellon Univ., Schenley Park, Pittsburgh, PA 15213.

Burlet & Fonlupt (1994). Polyhedral Consequences of the Amalgam Operation. *Disc. Math. 130, 39–55.*

Caprara & Fischetti (1996). $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts. *Math. Prog. 74(3), 221–235.*

Ceria, Nobili & Sassano (1997). Set Covering Problem. In Dell'Amico, Maffioli & Martello [1997], chapter 23, pp. 415–428.

Cheng & Cunningham (1997). Wheel Inequalities for Stable Set Polytopes. *Math. Prog. 77(3), 389–421.*

Chvátal (1975). On Certain Polytopes Associated with Graphs. *J. Comb. Theory 18, 138–154.*

—— (1976). On the Strong Perfect Graph Conjecture. *J. Comb. Theory 20, 139–141.*

Conforti & Cornuéjols (1992). Balanced $0, \pm 1$ Matrices, Bicoloring and Total Dual Integrality. Mgmt. Sci. Res. Rep. 581, Carnegie Mellon Univ., Schenley Park, Pittsburgh, PA 15213.

Conforti, Cornuéjols, Kapoor & Vušković (1997). Perfect, Ideal and Balanced Matrices. In Dell'Amico, Maffioli & Martello [1997], chapter 6, pp. 81–94.

Conforti, Cornuéjols, Kapoor, Vušković & Rao (1994). Balanced Matrices. In Birge & Murty [1994], pp. 1–33.

Conforti, Cornuéjols & Rao (1991). Decomposition of Balanced 0,1 Matrices, Parts I–VII. Mgmt. Sci. Res. Rep. 569–575, Carnegie Mellon Univ., Schenley Park, Pittsburgh, PA 15213.

Cook & Seymour (Eds.) (1990). *Polyhedral Combinatorics, Proc. of the DIMACS Workshop, Morristown, NJ, 1989*, volume 1 of *DIMACS Ser. in Disc. Math. and Theoretical Comp. Sci.* Am. Math. Soc., Providence, R.I.

Cornuéjols & Novik (1989). Ideal 0,1 Matrices. Mgmt. Sci. Res. Rep. #537, Grad. School of Ind. Admin., Carnegie Mellon Univ., Pittsburgh, PA 15213, USA.

Cornuéjols & Sassano (1989). On the 0,1 Facets of the Set Covering Polytope. *Math. Prog. 43, 45–55.*

Crowder, Johnson & Padberg (1983). Solving Large-Scale Zero-One Linear Programming Problems. *Op. Res. 31, 803–834.*

[2]Avail. at URL `http://akula.isye.gatech.edu/~evakylee/`

Cunningham & Marsh (1978). A Primal Algorithm for Optimum Matching. *Math. Prog. Study 8, 50–72.*

Cunningham, McCormick & Queyranne (Eds.) (1996). *Integer Programming and Combinatorial Optimization*, Proc. of the 5th Int. IPCO Conf., Vancouver, British Columbia, Canada.

Dell'Amico, Maffioli & Martello (Eds.) (1997). *Annotated Bibliographies in Combinatorial Optimization.* John Wiley & Sons Ltd, Chichester.

Deza & Laurent (1997). *Geometry of Cuts and Metrics.* Springer Verlag, Berlin.

Edmonds (1962). Covers and Packings in a Family of Sets. *Bulletin of the Am. Math. Soc. 68, 29–32.*

—— (1965). Maximum Matching and a Polyhedron with $0, 1$ Vertices. *J. of Res. of the Nat. Bureau of Standards 69 B, 125–130.*

Edmonds & Pulleyblank (1974). Facets of 1-Matching Polyhedra. In C. Berge & D. Chuadhuri (Eds.), *Hypergraph Seminar,* pp. 214–242. Springer Verlag, Heidelberg.

Euler (1736). Solutio Problematis ad Geometriam Situs Pertinentis (Latin) [The Solution of Problems About the Geometry of the Site]. In König [1986], pp. 275–288.

Euler, Jünger & Reinelt (1987). Generalizations of Cliques, Odd Cycles and Anticycles and Their Relation to Independence System Polyhedra. *Math. of OR 12(3), 451–462.*

Euler & Le Verge (1996). Complete Linear Descriptions for Stable Set Polytopes of Small Claw-Free Graphs. Manuscript, Lab. d'Informatique de Brest.

Ferreira (1994). *On Combinatorial Optimization Problems Arising in Computer System Design.* PhD thesis[3], Tech. Univ. Berlin.

Ferreira, Martin & Weismantel (1996). Solving Multiple Knapsack Problems by Cutting Planes[4]. *SIAM J. on Opt. 6, 858 – 877.*

Ford, Jr. & Fulkerson (1962). *Flows in Networks.* Princeton, NJ: Princeton Univ. Press.

Fulkerson (1970). Blocking Polyhedra. In Harris [1970], pp. 93–112.

—— (1971). Blocking and Anti-Blocking Pairs of Polyhedra. *Math. Prog. 1, 168–194.*

—— (1972). Anti-blocking Polyhedra. *J. Comb. Theory 12, 50–71.*

—— (1973). On the Perfect Graph Theorem. In Hu & Robinson [1973], pp. 69–76.

Fulkerson, Hoffman & Oppenheim (1974). On Balanced Matrices. *Math. Prog. Study 1, 120–132.*

Gallucio & Sassano (1993). The Rank Facets of the Stable Set Polytope for Claw-Free Graphs. Tech. Rep. R. 370, Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30, 00185 Roma, Italy.

Garey & Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman and Company.

Garfinkel & Nemhauser (1972). *Integer Programming.* NY, London, Sydney, Toronto: John Wiley & Sons.

Gerards (1989). A Min-Max Relation for Stable Sets in Graphs with no Odd-K_4. *J. Comb. Theory 47, 330–348.*

Giles & Trotter (1979). On Stable Set Polyhedra for $K_{1,3}$-Free Graphs. *J. Comb. Theory 31(3), 313–326.*

[3] Avail. at URL `http://www.zib.de/ZIBbib/Publications/`

[4] Avail. at URL `http://www.zib.de/ZIBbib/Publications/` as ZIB Preprint SC 93-07.

Goemans & Hall (1996). The Strongest Facets of the Acyclic Subgraph Polytope are Unknown. In Cunningham, McCormick & Queyranne [1996], pp. 415–429.

Gomory (1960). Solving Linear Programming Problems in Integers. In Bellman & Hall [1960].

Grötschel, Jünger & Reinelt (1985b). Facets of the Linear Ordering Polytope. *Math. Prog. 33, 43–60.*

—— (1985a). On the Acyclic Subgraph Polytope. *Math. Prog. 33, 28–42.*

Grötschel, Lovász & Schrijver (1984). Polynomial Algorithms for Perfect Graphs. *Annals of Disc. Math. 21, 325–356.*

—— (1988). *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, Berlin.

Grötschel & Wakabayashi (1990). Facets of the Clique Partitioning Polytope. *Math. Prog. 47(3), 367–387.*

Harris (Ed.) (1970). *Graph Teory and Its Applications.* NY, London: Academic Press, London.

Hougardy, Prömel & Steger (1994). Probabilistically Checkable Proofs and Their Consequences for Approximation Algorithms. *Disc. Math. 136, 175–223.*

Hu & Robinson (Eds.) (1973). *Mathematical Programming.* NY: Academic Press, London.

König (1931). Graphok és Matrixok (Hungarian with a German Abstract) [Graphs and Matrices]. *Matematikai és Fizikai Lapok 38, 116–119.*

—— (1936). Theorie der Endlichen und Unendlichen Graphen (German) [Theory of Finite and Infinite Graphs]. In König [1986], pp. 7–274.

—— (1986). *Theorie der Endlichen und Unendlichen Graphen (German) [Theory of Finite and Infinite Graphs]*, volume 6 of *Teubner Archiv zur Mathematik.* BSB B.G. Teubner Verlagsgesellschaft Leipzig. Distributed by Springer Verlag Wien NY. Reprint.

Laurent (1989). A Generalization of Antiwebs to Independence Systems and Their Canonical Facets. *Math. Prog. 45, 97–108.*

Lehman (1979). On the Width-Length Inequality. *Mathematical Programming 17, 403–417.*

—— (1981). The Width-Length Inequality and Degenerate Projective Planes. In Cook & Seymour [1990], pp. 101–105.

Lovász (1971). Normal Hypergraphs and the Perfect Graph Conjecture. *Disc. Math. 2, 253–267.*

—— (1972). A Characterization of Perfect Graphs. *J. Comb. Theory 13, 95–98.*

—— (1983). Perfect Graphs, volume 2 of *Selected Topics in Graph Theory*, chapter 3, pp. 55–87. Academic Press, London.

Lovász & Plummer (1986). *Matching Theory*, volume 29 of *Annals of Disc. Math.*

Lovász & Schrijver (1991). Cones of Matrices and Set-Functions and 0-1 Optimization. *SIAM J. on Opt. 1, 166–190.*

Lütolf & Margot (1998). A Catalog of Minimally Nonideal Matrices. *Math. Methods of OR 47, 221–241.*

Mahjoub (1985). On the Stable Set Polytope of a Series Parallel Graph. *Math. Prog. 40, 53–57.*

Martello & Toth (1990). *Knapsack Problems.* John Wiley & Sons Ltd, Chichester.

Martin & Weismantel (1997). The Intersection of Knapsack Polyhedra and Extensions. Preprint SC 97-61[5], Konrad-Zuse-Zentrum Berlin.

[5]Avail. at URL `http://www.zib.de/ZIBbib/Publications/`

Minty (1980). On Maximal Independent Sets of Vertices in Claw-Free Graphs. *J. Comb. Theory 21, 212–223.*

Müller (1993). On the Transitive Acyclic Subdigraph Polytope. In Rinaldi & Wolsey [1993], pp. 463–477.

—— (1996). On the Partial Order Polytope of a Digraph. *Math. Prog. 73(1), 31–49.*

Müller & Schulz (1995). The Interval Order Polytope of a Digraph. In Balas & Clausen [1995], pp. 50–64.

—— (1996). Transitive Packing. In Cunningham, McCormick & Queyranne [1996], pp. 430–444.

Naddef (1989). The Hirsch Conjecture is True for $(0,1)$-Polytopes. *Math. Prog. 45, 109–110.*

Nemhauser & Trotter (1973). Properties of Vertex Packing and Independence System Polyhedra. *Math. Prog. 6, 48–61.*

—— (1975). Vertex Packings: Structural Properties Algorithms. *Math. Prog. 8, 232–248.*

Nobili & Sassano (1989). Facets and Lifting Procedures for the Set Covering Polytope. *Math. Prog. 45, 111–137.*

—— (1992). A Separation Routine for the Set Covering Polytope. In Balas, Cornuéjols & Kannan [1992], pp. 201 – 219.

Oosten, Rutten & Spieksma (1995). The Facial Structure of the Clique Partitioning Polytope. Rep. M 95-09[6], Univ. of Limburg.

Padberg (1973a). On the Facial Structure of Set Packing Polyhedra. *Math. Prog. 5, 199–215.*

—— (1973b). Perfect Zero-One Matrices. *Math. Prog. 6, 180–196.*

—— (1975a). Characterizations of Totally Unimodular, Balanced and Perfect Matrices. In Roy [1975], pp. 275–284.

—— (1975b). A Note on Zero-One Programming. *Op. Res. 23(4), 833–837.*

—— (1976). Almost Integral Polyhedra Related to Certain Combinatorial Optimization Problems. *Lin. Algebra and Its Applications 15, 69–88.*

—— (1977). On the Complexity of Set Packing Polyhedra. *Annals of Disc. Math. 1, 421–434.*

—— (1979). Covering, Packing, and Knapsack Problems. *Annals of Disc. Math. 4, 265–287.*

—— (1993). Lehman's Forbidden Minor Characterization of Ideal 0-1 Matrices. *Disc. Math. 111(1–3), 409–420.*

Padberg & Rao (1982). Odd Minimum Cut-Sets and b-Matchings. *Math. of OR 7, 67–80.*

Padberg, van Roy & Wolsey (1985). Valid Inequalities for Fixed Charge Problems. *Op. Res. 33, 842–861.*

Poljak & Turzik (1992). Max-Cut in Circulant Graphs. *Disc. Math. 108, 379 – 392.*

Pulleyblank (1979). Minimum Node Covers and 2-Bicritical Graphs. *Math. Prog. 17, 91–103.*

Pulleyblank & Shepherd (1993). Formulations for the Stable Set Polytope of a Claw-Free Graph. In Rinaldi & Wolsey [1993], pp. 267–279.

Rinaldi & Wolsey (Eds.) (1993). *Integer Programming and Combinatorial Optimiziation*, Proc. of the 3rd Int. IPCO Conf.

Roy (Ed.) (1975). *Combinatorial Programming.* D. Reidel Pub. Co., Dordrecht, Holland.

Sachs (1986). Kommentierender Anhang (German) [Commenting Appendix]. In König [1986], pp. 313–345.

[6] Avail. at URL `http://www.fdaw.unimaas.nl/math/`

Sassano (1989). On the Facial Structure of the Set Covering Polytope. *Math. Prog. 44, 181–202.*

Schrijver (1979). Fractional Packing and Covering. Math. Center Tracts 106, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.

Schrijver (1986). *Theory of Linear and Integer Programming.* John Wiley & Sons, Chichester.

Schulz (1996). *Polytopes and Scheduling.* PhD thesis[7], Tech. Univ. Berlin.

Sekiguchi (1983). A Note on Node Packing Polytopes on Hypergraphs. *OR Letters 2(5), 243–247.*

Seymour (1990). On Lehman's Width-Length Characterization. In Cook & Seymour [1990], pp. 107–117.

Tesch (1994). *Disposition von Anruf-Sammeltaxis.* Deutscher Univ. Verlag, Wiesbaden.

Trotter (1975). A Class of Facet Producing Graphs for Vertex Packing Polyhedra. *Disc. Math. 12, 373–388.*

Wolsey (1976). Further Facet Generating Procedures for Vertex Packing Polytopes. *Math. Prog. 11, 158–163.*

—— (1990). Valid Inequalities for 0-1 Knapsacks and MIPs with Generalized Upper Bound Constraints. *Disc. Applied Math. 29, 251–261.*

Zemel (1978). Lifting the Facets of Zero-One Polytopes. *Math. Prog. 15, 268–277.*

[7] Avail. at URL `ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/`

Index of Part 1

H

I

J

K

L

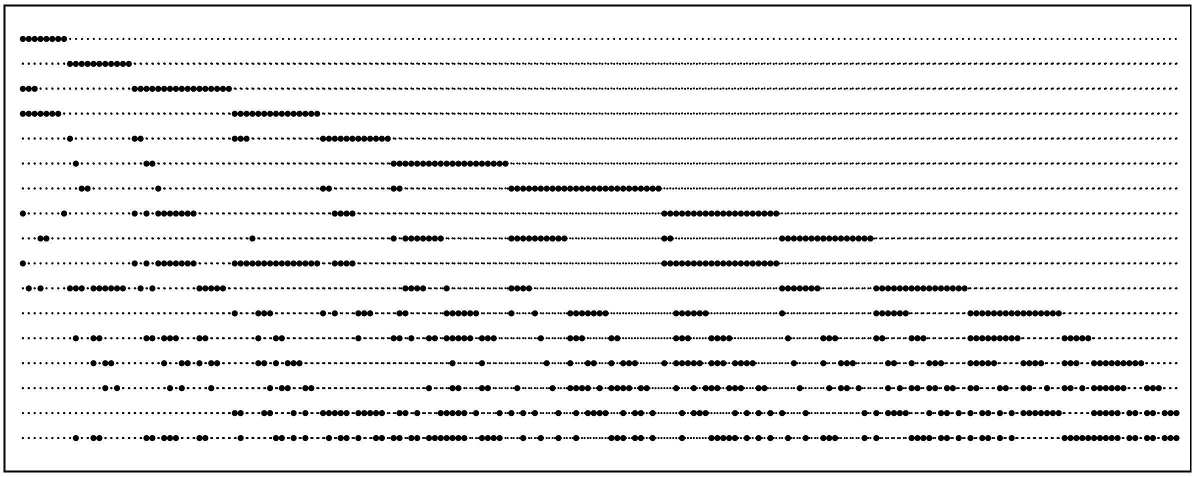
O

P

Q

Part II

Algorithmic Aspects



Chapter 3

An Algorithm for Set Partitioning

Summary. We document in this chapter the main features of a branch-and-cut algorithm for the solution of set partitioning problems. Computational results for a standard test set from the literature are reported.

Acknowledgement. We thank Robert E. Bixby[1] for many discussions about out-pivoting and for making this method available in the CPLEX callable library.

3.1 Introduction

This chapter is about the design and the implementation of a *branch-and-cut algorithm* for the solution of set partitioning problems: Our code `BC`. We assume for our exposition that the reader is familiar with the features of such a method. In particular, we do neither discuss the theoretical background of cutting plane algorithms, see Grötschel, Lovász & Schrijver [1988], nor the basic features and the terminology of branch-and-cut codes, see Nemhauser & Wolsey [1988], Padberg & Rinaldi [1991], Thienel [1995], and Caprara & Fischetti [1997]. In fact, our algorithm `BC` is to some extent a reimplementation of Hoffman & Padberg [1993]'s successful code `CREW_OPT`: The flowchart of `BC` coincides with the one of `CREW_OPT`, and the same applies to the primal heuristic, pool and searchtree management, and even to the data structures. Thus, we elucidate only those parts of our implementation where we see some contribution. This applies to the *mathematical core* of the algorithm: Separation and preprocessing.
Our *description* is intended to give enough information to allow a reimplementation of the routines in `BC`. We do, however, neither discuss software engineering and programming issues, nor do we report the computational tests that guided our design decisions.

[1]Robert E. Bixby, Dept. of Math., Univ. of Houston, TX 77204-3476, USA, Email `bixby@rice.edu`.

Our first contribution is a *separation* routine. We have implemented a new family of cutting planes for set partitioning problems: The aggregated cycle inequalities of Section 2.6. Recall that these inequalities stem from a set packing relaxation of the set covering problem. Together with Nobili & Sassano [1992]'s k-projection cuts and the odd hole inequalities for the set covering polytope, see page 47 in Subsection 1.9.1 of this thesis, that have reportedly been implemented by Hoffman & Padberg [1993], these cuts form one of the very few families of combinatorial inequalities for the set covering polytope that have been used in a branch-and-cut algorithm.

Our second contribution concerns *preprocessing*. We have extended some known techniques of the literature and explored their performance with probabilistic methods. We have also investigated the interplay of iterated use of preprocessing operations with a dual simplex algorithm. It turns out that much of the potential of preprocessing can only be realized after certain degeneracy issues have overcome. We have developed a novel pivoting technique that resolves this problem completely.

Pointers to other recent computational work on set partitioning problems are Atamturk, Nemhauser & Savelsbergh [1995] (Lagrangean relaxation with iterated preprocessing), Wedelin [1995] (Lagrangean relaxation with a perturbation technique), and Chu & Beasley [1995] (preprocessing and genetic algorithms).

This chapter is organized as follows. In Section 3.2 we discuss preprocessing. We give a list of preprocessing operations from the literature and perform a probabilistic analysis of their running time. The iterated application of such techniques in a simplex based branch-and-cut algorithm runs into an unexpected obstacle: Degeneracy problems prevent us from removing large redundant parts of the problem without destroying a dual feasible basis. We show how to overcome this problem. Separation procedures are treated in Section 3.3. We discuss implementation details of our routines for the detection of violated clique, cycle, and aggregated cycle inequalities. Computational results are presented in Section 4.7.

The subsequent sections resort to the following *notation*. We consider set partitioning problems of the form

$$\text{(SPP)} \quad \min \; w_0 + w^{\mathrm{T}}x \qquad Ax = \mathbf{1} \qquad x \in \{0,1\}^n,$$

where $A \in \{0,1\}^{m\times n}$ and $w \in \mathbb{Z}^n$ are an integer matrix and a nonnegative integer objective function, respectively. ρ is the density of the matrix A, λ is supposed to be the *maximum* number of nonzero entries in a column, and μ is the *average* number of entries in a column. $G = G(A)$ is the column intersection graph associated to A; this graph gives rise to terminology like "the neighbors $\gamma(j)$ of a column $A_{\cdot j}$" etc. The real number w_0, the *offset*, is a positive constant that plays a role in preprocessing. We denote by x^* an arbitrary but fixed optimal basic solution of the LP relaxation of (SPP), its objective value by z^*, the reduced costs by $\overline{w}$, and by F the set of fractional variables of x^*.

Associated to (SSP) are the set packing and set covering *relaxations*

$$\text{(SSP)} \quad \max \; w_0 + w^{\mathrm{T}}x \qquad Ax \leq \mathbf{1} \qquad x \in \{0,1\}^n \qquad\qquad \text{(SCP)} \quad \min \; w_0 + w^{\mathrm{T}}x \qquad Ax \geq \mathbf{1} \qquad x \in \{0,1\}^n.$$

It is well known that all of these problems are $\mathcal{NP}$-hard, see Garey & Johnson [1979]. We do not discuss further *complexity* issues here (see Emden-Weinert et al. [1996] for this topic).

3.2 Preprocessing

Preprocessing or *presolving* is the use of automatic simplification techniques for linear and integer programs. The techniques aim at "improving" a given IP formulation in the sense that some solution method works better. We are interested here in *preprocessing for branch-and-cut algorithms*. These algorithms have LP (re)optimizations as their computational bottleneck and their presolvers try to make this step more effective by (i) reducing the size of and (ii) "tightening" IP formulations, and by (iii) identifying "useful substructures". Here, tightness is a measure for the quality of an IP formulation: We say that (IP) is a tighter formulation than (IP′) if the integer solutions of both programs are the same, but the solution set of the LP relaxation of (IP) is contained in that of (IP′).
There are many ways to put (i)–(iii) into practice: Fixing of variables, removing redundant constraints, tightening bounds, reduced cost fixing, probing, and constraint classification are just a few popular examples of *reductions*, as preprocessing techniques are also called. *Surveys* on preprocessing for IPs can be found in Crowder, Johnson & Padberg [1983], the textbook Nemhauser & Wolsey [1988, Section I.1.6] and the references therein, Hoffman & Padberg [1991], and Suhl & Szymanski [1994], while Brearley, Mitra & Williams [1975], the lecture notes of Bixby [1994], and Andersen & Andersen [1995] review (closely related) LP presolving techniques. Most of these methods are *simple* — but amazingly *effective*, as the above publications' computational sections document.
Special problems offer additional potential for preprocessing, and *set partitioning* is one of the best studied classes of integer programs in this respect. The following subsections survey preprocessing techniques for set partitioning, discuss efficient implementations, analyse expected running times, and report some computational results.

3.2.1 Reductions

We give next a list of *reductions for set partitioning problems* that subsumes (as far as we know) all suggestions of the literature. Each technique describes, in principle, a transformation and a back transformation of a given set partitioning problem into another one and a correspondence of (optimal) solutions, but as the reductions are quite simple, we state them in a shortcut informal way as in Andersen & Andersen [1995]. These reductions will be discussed in more detail in Subsections 3.2.4–3.2.12.

P0 (Empty Column) $\exists j : A_{\cdot j} = 0$
If column j is empty, one can

(i) eliminate column j if $w_j \geq 0$
(ii) eliminate column j and add w_j to w_0 if $w_j < 0$.

P1 (Empty Row) $\exists r : A_{r\cdot} = 0$
If row r is empty, the problem is infeasible.

P2 (Row Singleton) $\exists r, j : A_{r\cdot} = e_j^{\mathrm{T}}$
If row r contains just one column j, one can "fix x_j to one", i.e.,

(i) eliminate column j and add w_j to w_0,
(ii) eliminate all columns $i \in \gamma(j)$ that are neighbors of column j, and
(iii) eliminate all rows $s \in \operatorname{supp} A_{\cdot j}$ that are covered by column j.

P3 (Dominated Column) $\exists i, J : A_{\cdot i} = \sum_{j \in J} A_{j\cdot} \ \wedge\ w_i \geq \sum_{j \in J} w_j, \quad i \notin J$
If column i is a combination of other columns $j \in J$ and $w_i \geq \sum_{j \in J} w_j$, one can eliminate the "dominated" column i.

P3′ (Duplicate Column) $\exists i, j : A_{\cdot i} = A_{\cdot j} \ \wedge\ w_i \geq w_j, \quad i \neq j$
If two columns $i \neq j$ are identical and $w_i \geq w_j$, one can eliminate column i.

P4 (Dominated Row) $\exists r, s : A_{r\cdot} \leq A_{s\cdot}, \quad r \neq s$
If row r is contained in row s, one can

(i) eliminate all columns $j \in \operatorname{supp}(A_{s\cdot} - A_{r\cdot})$ and
(ii) eliminate the "dominated" row r.

P4′ (Duplicate Row) $\exists r, s : A_{r\cdot} = A_{s\cdot}, \quad r \neq s$
If two rows $r \neq s$ are identical, one can eliminate the "duplicate" row r.

P5 (Row Clique) $\exists r, j : \operatorname{supp} A_{r\cdot} \subseteq \gamma(j) \ \wedge\ j \notin \operatorname{supp} A_{r\cdot}$
If all columns in row r are neighbors of a column j not in row r, one can eliminate column j.

P6 (Parallel Column) $\exists r, s, i, j : A_{r\cdot} - A_{s\cdot} = e_i^{\mathrm{T}} - e_j^{\mathrm{T}}, \quad r \neq s, i \neq j$
If two rows $r \neq s$ have a common support except for two elements $i \neq j$, one of them contained in row r and the other in row s, the variables $x_i = x_j$ are "parallel", and one can

(i) eliminate columns i and j if they are neighbors, i.e., $ij \in E(G)$,

or

(ii) merge column i and j into a "compound" column $A_{\cdot i} + A_{\cdot j}$ with cost $w_i + w_j$ otherwise.

P7 (Symmetric Difference) $\exists r, s, t, j : A_{r\cdot} \geq A_{s\cdot} - A_{t\cdot} + 2e_j^{\mathrm{T}}, \quad r \neq s, r \neq t, s \neq t$
If row r contains all columns that are in row s, but not in t, and some column j that is in row t, but not in s, one can eliminate column j.

P7′ (Symmetric Difference) $\exists r, s, t : \operatorname{supp} A_{r\cdot} \supseteq \operatorname{supp}(A_{s\cdot} - A_{t\cdot}), \quad r \neq s, r \neq t, s \neq t$
If row r covers the symmetric difference of rows s and t, one can

(i) eliminate all columns $j \in \operatorname{supp}(A_{s\cdot} - A_{t\cdot})$ in the symmetric difference and
(ii) eliminate row s.

P8 (Column Singleton) $\exists r, j : A_{\cdot j} = e_r \ \wedge\ (|\operatorname{supp} A_{r\cdot}| = 2 \ \vee\ \exists s : A_{r\cdot} - e_j^{\mathrm{T}} \leq A_{s\cdot})$
If column j is a unit column e_r and either row r is a "doubleton" (has only two nonzero elements) or row r, with a_{rj} set to zero, is covered by some other row s, one can

(i) substitute $x_j = 1 - \sum_{i \neq j} a_{ri} x_i$ in the objective to obtain $(w_0 + w_j) + (w^{\mathrm{T}} - w_j A_{r\cdot})x$,
(ii) eliminate column j, and
(iii) eliminate row r.

The next two reductions assume knowledge of an *upper bound* z_u

$$z_u \geq \min \ w^{\mathrm{T}} x \quad Ax = \mathbf{1} \quad x \in \{0,1\}^n$$

on the optimal objective value of the set partitioning problem and knowledge of LP information: P9 requires an optimal basis of the LP relaxation of (SPP) and associated data, in particular the objective value z^*, the solution x^*, and the reduced costs $\overline{w}$, P10 lower bounds z on the values of certain LPs. For these reasons, rules P9 and P10 are called *LP based*.

P9 (Reduced Cost Fixing) $\exists j : (x_j^* = 0 \ \wedge \ z_u \leq z^* + \overline{w}_j) \ \vee \ (x_j^* = 1 \ \wedge \ z_u \leq z^* - \overline{w}_j)$

(i) If $x_j^* = 0$ *and* $z_u \leq z^* + \overline{w}_j$*, one can eliminate column* j*.*

(ii) If $x_j^* = 1$ *and* $z_u \leq z^* - \overline{w}_j$*, one can fix* x_j *to one.*

P10 (Probing) $\exists j, z, \overline{x}_j : z_u \leq z \leq \min \ c^{\mathrm{T}}x, Ax = \mathbf{1}, 0 \leq x \leq \mathbf{1}, x_j = \overline{x}_j, \quad \overline{x}_j \in \{0,1\}$

(i) If $z_u \leq z \leq \min \ w^{\mathrm{T}}x, Ax = \mathbf{1}, 0 \leq x \leq \mathbf{1}, x_j = 0$*, one can fix* x_j *to one.*

(ii) If $z_u \leq z \leq \min \ w^{\mathrm{T}}x, Ax = \mathbf{1}, 0 \leq x \leq \mathbf{1}, x_j = 1$*, one can eliminate column* j*.*

Some of these reductions are "folklore" and do not have a genuine origin in the *literature*. But however that may be, P2–P4 appear in Balinski [1965, page 285] (in a set covering context), P1–P4 and P7 in Garfinkel & Nemhauser [1969], and P1–P5 in Balas & Padberg [1976]. P6 is due to Hoffman & Padberg [1993], substitution techniques like P8 are discussed by Bixby [1994]. P9 was introduced by Crowder, Johnson & Padberg [1983], probing techniques like P10 are mentioned in Suhl & Szymanski [1994], some related procedures of similar flavour in Beasley [1987] (for set covering problems).

Given all these reductions, the next point is to devise a good *strategy* for their application. As the application of one rule can result in additional possible simplifications for another rule, one usually applies reductions P0–P8 in a loop, doing another *pass* until no further simplifications can be achieved. P9 and P10 can be applied at any other point: Once the LP and/or bounding information is computed, these reductions are independent of the other rules.

Table 3.1 gives an impression of the *significance* of preprocessing for the solution of set partitioning problems. The figures in this table were obtained by preprocessing the Hoffman & Padberg [1993] `acs` test set of 55 set partitioning problems from `a`irline `c`rew `s`cheduling applications with the preprocessing routines of our code `BC`, that uses a subset of reductions P1–P10. The first column in Table 3.1 gives the name of the problem, and the next three columns its original size in terms of numbers of rows, columns, and matrix density ρ, i.e., the percentage of nonzero elements in the matrix. Applying some of the non LP-based preprocessing rules P1–P8, the problems are reduced as indicated in the three succeeding "Presolved" columns. The remainder of the table goes one step further: After solving the LP relaxation of the preprocessed problem and calling some primal heuristic, the problem is preprocessed again. This time, knowledge of the LP lower bound z^* and the upper bound z_u from the heuristic allows also the use of LP-based techniques, in this case reduced fixing (P9). The results of this second round of preprocessing, using a subset of reductions P1–P9, are reported in the "Presolved: LP-based" section of the table. The success of the LP based methods depends on the size of the gap between the heuristic upper bound z_u and the LP lower bound z^*, and this gap $(z_u - z^*)/z_u$, given as a percentage of the upper bound (the possible improvement of z_u), is reported in column "Gap". A value of $+\infty$ means that no valid solution is known. Sometimes, the LP relaxation is already integral and the problem is solved. In this case, indicated by the entry "LP" in the Gap column, it is not necessary to compute a further upper bound or perform a second round of preprocessing, hence the dashes in the corresponding preprocessing columns. One problem, `nw16`, was even solved in the first preprocessing phase such that not a single LP had to be solved; this outcome is indicated by the entry "PP" in column Gap. The final "Time" column gives the sum of the preprocessing times for, depending on the problem, one or two calls to the preprocessor in CPU seconds on a Sun Ultra Sparc 1 Model 170E.

Name	Original			Presolved			Gap	Presolved: LP-based			Time
	Rows	Cols	ρ	Rows	Cols	ρ	%	Rows	Cols	ρ	Sec.
nw41	17	197	0.22	17	177	0.22	4.23	12	15	0.20	0.01
nw32	19	294	0.24	17	241	0.27	3.64	13	42	0.25	0.01
nw40	19	404	0.27	19	336	0.27	4.96	13	36	0.24	0.03
nw08	24	434	0.22	19	349	0.28	LP	—	—	—	0.01
nw15	31	467	0.20	29	465	0.21	LP	—	—	—	0.01
nw21	25	577	0.25	25	426	0.24	9.91	19	51	0.22	0.03
nw22	23	619	0.24	23	531	0.24	0.60	17	18	0.21	0.00
nw12	27	626	0.20	25	451	0.15	LP	—	—	—	0.02
nw39	25	677	0.27	25	567	0.26	8.27	11	29	0.19	0.01
nw20	22	685	0.25	22	566	0.25	1.11	15	18	0.24	0.03
nw23	19	711	0.25	12	430	0.38	2.65	11	27	0.30	0.03
nw37	19	770	0.26	19	639	0.26	6.17	13	22	0.21	0.03
nw26	23	771	0.24	21	514	0.25	2.87	17	30	0.20	0.03
nw10	24	853	0.21	20	643	0.25	LP	—	—	—	0.02
nw34	20	899	0.28	20	750	0.28	3.18	15	20	0.25	0.01
nw28	18	1210	0.39	18	599	0.36	5.97	11	20	0.38	0.03
nw25	20	1217	0.30	20	844	0.30	11.41	20	101	0.24	0.02
nw38	23	1220	0.32	20	881	0.36	0.11	15	18	0.30	0.03
nw27	22	1355	0.32	22	926	0.31	4.51	13	19	0.26	0.03
nw24	19	1366	0.33	19	926	0.33	11.04	16	43	0.28	0.02
nn01	18	1072	0.25	17	983	0.26	8.30	17	983	0.26	0.04
nn02	23	1079	0.26	19	820	0.32	2.23	15	19	0.28	0.04
nw35	23	1709	0.27	23	1403	0.27	8.74	19	91	0.28	0.03
nw36	20	1783	0.37	20	1408	0.36	$+\infty$	20	1408	0.36	0.05
nw29	18	2540	0.31	18	2034	0.31	13.06	17	488	0.30	0.08
nw30	26	2653	0.30	26	1884	0.30	69.84	26	1884	0.30	0.07
nw31	26	2662	0.29	26	1823	0.29	2.23	20	34	0.26	0.06
nw19	40	2879	0.22	32	2134	0.22	LP	—	—	—	0.07
nw33	23	3068	0.31	23	2415	0.31	2.96	13	19	0.21	0.06
nw09	40	3103	0.16	33	2296	0.19	LP	—	—	—	0.07
nw07	36	5172	0.22	33	3104	0.23	LP	—	—	—	0.11
aa02	531	5198	0.01	361	3928	0.02	LP	—	—	—	0.24
nw06	50	6774	0.18	37	5936	0.20	18.98	37	883	0.18	0.28
aa06	646	7292	0.01	507	6064	0.01	0.25	419	892	0.01	1.18
kl01	55	7479	0.14	47	5957	0.13	1.00	44	1151	0.12	0.36
aa05	801	8308	0.01	533	6371	0.01	0.37	532	6354	0.01	1.00
aa03	825	8627	0.01	558	6970	0.01	0.43	558	6823	0.01	1.09
nw11	39	8820	0.17	28	5946	0.21	0.00	25	32	0.09	0.24
nw18	124	10757	0.07	81	7934	0.08	8.44	81	7598	0.08	0.74
us02	100	13635	0.14	44	8946	0.17	LP	—	—	—	0.50
nw13	51	16043	0.13	48	10901	0.12	0.21	46	100	0.06	0.45
us04	163	28016	0.07	98	4285	0.08	0.73	69	86	0.06	1.05
nw03	59	43749	0.14	53	38956	0.15	2.70	50	421	0.12	1.62
nw01	135	51975	0.06	135	50069	0.06	LP	—	—	—	0.77
us03	77	85552	0.18	50	23207	0.21	LP	—	—	—	3.32
nw02	145	87879	0.06	145	85258	0.06	LP	—	—	—	1.39
nw17	61	118607	0.14	54	78173	0.15	14.48	52	11332	0.16	4.84
nw14	73	123409	0.10	68	95169	0.10	LP	—	—	—	3.73
nw16	139	148633	0.07	0	1	0.00	PP	—	—	—	7.13
nw05	71	288507	0.10	58	202482	0.12	LP	—	—	—	9.00
kl02	71	36699	0.08	69	16542	0.08	2.16	62	3415	0.08	1.41
us01	145	1053137	0.09	86	351018	0.10	5.79	86	36201	0.10	57.95
nw04	36	87482	0.20	35	46189	0.20	9.47	35	15121	0.22	2.31
aa04	426	7195	0.02	343	6200	0.02	5.21	343	6189	0.02	1.62
aa01	823	8904	0.01	616	7625	0.01	3.46	616	7586	0.01	1.55
$\sum$ 55	6378	2305749	—	4736	1105692	—	—	3433	109619	—	104.86

Table 3.1: Preprocessing Airline Crew Scheduling Problems.

The figures in Table 3.1 show that already without LP-based information, the *problem size* can often be reduced substantially by rules P1–P8. Using additional LP and heuristic information leads often (but not always) to a further reduction in problem size of about an order of magnitude, and the preprocessing can be performed in a short time. Note that the matrix density is essentially unaffected by preprocessing, i.e., it is not true that "only very sparse or dense parts are removed". The extent of the reductions can be explained as a consequence of the generation of the `acs` problems: The instances are the output of an *automatic*, *heuristic*, and *randomized* column generation process that tends to produce redundant formulations for various reasons that we can not discuss here.

The second goal of preprocessing, namely, *tightening* of the formulation, could, however, not be achieved: In all cases (except `nw16`), the optimal objective values of the LP relaxations of the original and the preprocessed problem are identical. (The values are not reported in the table.) And in fact, (strictly) dominated columns, for example, can not become basic in an optimal solution anyway and neither does their identification provide information that is not also given by the LP solution, nor does elimination of dominated columns help in the sense that it leads to a different LP solution. One can check that similar conclusions hold also for most of the other preprocessing rules; only P7, P7′, and P10 can potentially fix variables to values that would not automatically be assigned to them by an optimal LP solution.

The last two paragraphs argued that the effect of preprocessing set partitioning problems is less a tighter LP relaxation than a reduction is problem size. There are three main *advantages* of solving smaller problems in a branch-and-cut context. First, a better use of the *cache*: If large contiguous parts of the problem data can be transferred into high-speed memory, list processing operations, like the computation of inner products, can be carried out much more efficiently. Note that some care has to be taken to profit from this effect; it is in particular not enough to fix or eliminate variables just by adjusting bounds, because this can result in useless data being not only transferred into and out of the cache, but also in "clogging" it. Instead, fixed columns must be purged from memory that is accessed for calculations in the cache. A second advantage is that a reduction in the number of *rows* results in a smaller *basis* and this speeds up the simplex algorithm. Third, it is of course also true that elimination reduces the number of arithmetic operations. Considering problem `us01`, for example, it is clear that pricing out 36,000 columns is much faster than pricing out one million, even if all of the redundant ones are fixed by bound adjustments. To illustrate these effects, we can compare the total time to solve the LP relaxations of the 55 original `acs` problems with the time needed to solve the presolved instances: Preprocessing halves LP time from 547.380 to 266.970 seconds, just as it halves the number of nonzeros, and this trend can safely be extrapolated. But simplex iterations, as expected in the light of the above discussion, are nearly unchanged: 13,277 with in comparison to 16,128 without preprocessing (using the dual simplex algorithm of CPLEX [1997], steepest edge pricing, and turning off the preprocessing capabilities of this code, again on a Sun Ultra Sparc 1 Model 170E). The numbers for problem `us01` are 116.840 seconds/296 iterations to 280.110 seconds/240 iterations, i.e., this problem (that makes up about half of the test set in terms of nonzeros) does not bias the results of the above comparison into a misleading direction. A rule of thumb for practical set partitioning solving is thus that after solving the first one or two LPs, the bulk of the data will have been eliminated, and the remainder of the computation deals with comparably small problems.

We close this introductory section with some general remarks on *algorithmic aspects*. At first glance, preprocessing appears to be completely trivial, because it is so easy to come up with polynomial time procedures for all rules except P3 and indeed, straightforward *implementa-*

tions work well for small and medium sized problems. The only essential issue is to keep an eye on exploiting the *sparsity* of the constraint matrix (e.g., by storing columns and rows as ordered lists of their nonzero entries) and this is the only implementation detail mentioned in most of the literature. For large scale problems, however, naive implementations will not work any more. For example, searching for duplicate columns by comparing all pairs is out of the question for problems with 100,000 or more columns, although this algorithm has a polynomial complexity of $O(n^2)$ operations (assuming that each column has at most some constant number of nonnull entries). Recent algorithms for large scale set partitioning problems of Hoffman & Padberg [1993] and Atamturk, Nemhauser & Savelsbergh [1995] thus (i) use only simple preprocessing rules that are (ii) implemented in a more sophisticated way. Both of these articles contain discussions on design and implementation of preprocessing routines.
The remainder of this section describes the design and implementation of the preprocessing module of our set partitioning solver `BC`. Subsection 3.2.2 contains some preliminaries on data structures. Subsections 3.2.4 to 3.2.12 investigate the individual preprocessing rules P1–P10. We describe and discuss our particular implementations and do a *probabilistic analysis* of the expected running times. Subsection 3.2.13 draws the readers attention to a conflict that comes up in repeated calls of preprocessing routines in a branch-and-cut framework: Elimination of variables and/or rows can destroy the dual feasibility of the basis. We argue that this phenomenon is a significant obstacle and develop a *pivoting* technique that overcomes the problem completely. The final Subsection 3.2.14 puts the pieces together and describes the global layout of the complete preprocessing module.

3.2.2 Data Structures

We will see in the discussions of individual routines in the following subsections that the whole task of preprocessing consists of doing various kinds of *loops* through the columns and rows of the constraint matrix, occasionally *deleting* some of the data. The data structures of the preprocessing module must allow to perform these basic operations efficiently and we discuss in this subsection some basic issues that come up in this context. These explanations are a preparation for the probabilistic analysis of the following subsection.

$$
\begin{array}{c c}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{pmatrix}
. & 1 & . & 1 & . & . \\
1 & . & 1 & . & . & 1 \\
1 & 1 & . & . & . & . \\
. & 1 & . & . & . & . \\
. & . & . & . & . & 1 \\
1 & . & . & 1 & . & . \\
. & 1 & . & 1 & . & .
\end{pmatrix}
\end{array}
\qquad
\begin{array}{l}
\texttt{beg[]} = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 3 & 7 & 8 & 11 & 11 \\ \hline\end{array} \\[1ex]
\texttt{cnt[]} = \begin{array}{|c|c|c|c|c|c|}\hline 3 & 4 & 1 & 3 & 0 & 2 \\ \hline\end{array} \\[1ex]
\texttt{ind[]} = \begin{array}{|c|c|c||c|c|c|c||c||c|c|c||||c|c|}\hline 1 & 2 & 5 & 0 & 2 & 3 & 6 & 1 & 0 & 5 & 6 & 1 & 4 \\ \hline\end{array}
\end{array}
$$

Figure 3.1: Storing Sparse 0/1 Matrices in Column Major Format.

We use a representation of the matrix in *row* and *column major format* as ordered and contiguous lists of the nonzero entries of the columns and rows. Figure 3.1 gives an example of column major format, row major format is obtained by storing the transposed matrix in column major format. The matrix in the example has 7 rows and 6 columns that are numbered starting from 0. Its 13 nonzero entries are stored by row index column wise, in ascending order, and contiguously in an array `ind[]`: The first three entries 1,2, and 5 give the row indices of the nonzero entries of column 0, the next four entries correspond to column 1, and so on; the empty column 4 has, of course, no entry. The arrays `cnt[]` and `beg[]` are used

to locate the data for a particular column in the ind[] array: cnt[i] gives the number of nonzero elements in column i, and beg[i] denotes the starting index for data of this column in the array ind. For more details, see, e.g., the manual CPLEX [1995].
Column major format allows fast *loops* through columns. As an example, consider the following C-type pseudocode to scan column i:

```
int nne;
register       int *colpnt = ind    + beg[i];
register const int *colend = colpnt + cnt[i];
for (; colpnt < colend; colpnt++) {
    nne = *colpnt;
    ... // some further operations
}
```

Note that this loop requires per nonzero just one comparison of two pointers that can be kept in registers, one increment of a pointer in a register, and one memory dereference, i.e., only three *operations*. The slowest of these is the dereference, but this operation can benefit from loading the ind[] array, or large contiguous parts of it, into the cache. Note that this doesn't work for pointer oriented data structures if data got fragmented in the computer's main memory, at least not if no additional precautions are taken. Note also that a pointer oriented structure requires at least one additional pointer dereference.
The structure also offers various kinds of possibilities to *eliminate* columns conveniently. The simplest method is to just set the cnt to zero. This technique results in some superfluous data in the cnt and beg arrays, and chunks of "dead" data in the ind array, with the already mentioned negative effects. At some point, it hence pays to re-store the matrix, eliminating garbage of this type; this can be done in time linear in the number of *remaining* nonzeros.
Column major format is, of course, unsuited for any kind of *row oriented* operations. To perform these efficiently as well, we store the matrix a second time in row major format. We like to point out that this is still more *memory efficient* than a pointer oriented representation, because only two entries are required for each nonzero (one ind entry for each nonzero in the row and one in the column representation).
We have to pay for the simplicity of row and column major format when it comes to keeping the two copies of the matrix *synchronized*: Eliminating columns with any one of the above mentioned methods renders the row representation invalid (and vice versa), and the only method to make them match again is to *transpose* the matrix, i.e., to set up the row representation from scratch. This can be done in time linear in the number of nonzeros in two passes through the matrix: The first pass determines the number of entries per row, and the second pass puts the elements in their places. To keep this bookkeeping effort at a minimum, it is of course advisable to first perform all column oriented operations, then transpose once, do row computations, transpose once, and so on. This strategy yields reasonable results: The first round of preprocessing in Table 3.1 spends 10.010 seconds out of a total of 80.830 in transposition and these numbers are also representative for later stages of the computation. This means that we pay a price of about 15% in computation time for using the simple row and column major format. It is not so easy to estimate how this compares to other possible data structures because of the effect of additional operations performed in a row or column scan and we have not implemented an alternative version, but we feel that the above considerations together with our computational findings justify the use of row and column major format for preprocessing purposes.

3.2.3 Probabilistic Analyses

We estimate in this subsection the *expected running time* of two basic list processing operations that we will use frequently in the sequel. These results will allow us to compute expected running times for the preprocessing rules of Subsection 3.2.1. Our results are summarized in the following Table 3.2. Here, each number gives the expected running time for the application of an entire *rule*, i.e., the value $O(n \log n)$ for rule P3$'$ gives the expected running time for removing *all* duplicate columns and so on. A — means that we have not analyzed the rule.

Operation		Expected Running Time
P0	Empty Columns	$O(n)$
P1	Empty Rows	$O(m)$
P2	Row Singletons	$O(mn)$ (amortized)
P3	Dominated Columns	—
P3$'$	Duplicate Columns	$O(n \log n)$
P4	Dominated Rows	$O(m\mu^2)$
P4$'$	Duplicate Rows	$O(m \log m)$
P5	Row Cliques	—
P5^M	Row Clique Heuristic	$O\big(M^2 n\mu^2 e^{-n\rho}(n\rho)^M/(1-\rho)^M\big)$
P6	Parallel Columns	—
P7	Symmetric Differences	—
P7$'$	Symmetric Differences	$O(n\mu^3)$
P8	Column Singletons	$O(n\mu^3)$
P9	Reduced Cost Fixing	$O(n)$
P10	Probing	—

Table 3.2: Estimating Running Times of Preprocessing Operations.

The first list processing operation that we consider is the *lexicographic comparison* of two random 0/1 sequences of infinite length, which is supposed to model a test whether two columns or rows of a random 0/1 matrix are identical or not. We will show that (under certain assumptions) this test takes constant expected time. The second operation is the *iterative intersection* of random 0/1 sequences of finite length. This time, we think of a situation where we want to find common rows in a set of columns or common columns in a set of rows. Again, it will turn out that the intersection of random 0/1 sequences becomes empty "fast".

Lexicographic Comparison of Two Random Infinite 0/1 Sequences. We compute in this paragraph the expected number of operations for a lexicographic comparison of two infinite random 0/1 sequences in a certain uniform probabilistic model. Our analysis will be based on the following *assumptions*:

(i) We look at infinite random sequences of zeros and ones, where the ones appear *independently* with some probability $\rho \in (0,1)$.

(ii) The sequences are stored in a *sparse* format as ordered lists of the indices of their nonnull entries.

(iii) Two sequences (from $\{0,1\}^\infty$) are *compared lexicographically* by scanning their index lists from the beginning, doing as many comparisons as there are common entries in the two index lists plus one additional comparison to detect the first difference.

We have already pointed out that we want to use this setting as a model for a lexicographic comparison of two columns or rows in a 0/1 matrix (of a set partitioning problem). In this context, assumptions (ii) and (iii) are canonical. (i) assumes identically and independently distributed ones in the sequences. Formally, such a sequence a belongs to a *probability space*

$$(\{0,1\}^\infty, \mathcal{A}, P)$$

that has as its groundset the set of all 0/1 sequences with an associated σ-algebra $\mathcal{A}$ and a probability distribution P such that the border distributions are binomial with parameter ρ, i.e., the two probabilities $P(a_i = 0) = 1 - \rho$ and $P(a_i = 1) = \rho$, $i \in \mathbb{N}$, exist and have the stated values. This is certainly unrealistic: The model results in low probabilities for the existence of duplicate columns and this obviously contradicts the computational findings of Table 3.1. But, for want of something better, we will nevertheless work with (i). Making the best of it, we can be happy about the technical advantage that this model has only one *parameter*, the probability ρ, which is to be identified with the matrix density. Our goal will be to obtain the expected number of operations to compare two 0/1 sequences lexicographically as a function of ρ. Considering *infinite* sequences for this purpose has the advantage that the analysis becomes independent of the number of rows or columns. As it takes certainly more time to compare two infinite sequences than two finite ones, this results in a model that is valid for lexicographic comparisons of rows *and* columns.

In this (not completely specified model) $(\{0,1\}^\infty, \mathcal{A}, P)$ consider the following *random experiment*: Choose two 0/1 sequences at random and perform a sparse lexicographic comparison according to (iii). Let the random variable $Y_\rho : \{0,1\}^\infty \times \{0,1\}^\infty \to \mathbb{N} \cup \{\infty\}$ denote the number of comparisons until the first two indices differ. Assumptions (i)–(iii) suggest that the probability that such a lexicographic comparison takes k comparisons of *individual* indices of nonzeros ($k \geq 1$) should be

$$P(Y_\rho = k) = \sum_{j=k}^{\infty} \binom{j-1}{k-1} \rho^{2(k-1)} (1-\rho)^{2\left((j-1)-(k-1)\right)} 2\rho(1-\rho), \qquad k \in \mathbb{N}. \tag{3.1}$$

In this expression, ρ^2 is the probability that a common nonzero appears at a random position in both 0/1 sequences, and $(1-\rho)^2$ is the probability for a common zero. There are $\binom{j-1}{k-1}$ possibilities to distribute $k-1$ common ones over the first $j-1$ positions in both sequences that account for the first $k-1$ comparisons, and

$$\binom{j-1}{k-1} \rho^{2(k-1)} (1-\rho)^{2\left((j-1)-(k-1)\right)}$$

is the corresponding probability. The final term $2\rho(1-\rho)$ is the probability for a difference in position j that is detected in the kth and last comparison. The following theorem *assumes* that the model $(\{0,1\}^\infty, \mathcal{A}, P)$ has property (3.1).

3.2.1 Lemma (Lexicographic Comparison of Two Random 0/1 Sequences)
Let $\rho \in (0,1)$ and $(\{0,1\}^\infty, \mathcal{A}, P)$ be a probability space. Let further Y_ρ be a random variable that counts the number of index comparisons in a lexicographic comparison of two random elements a, b from $(\{0,1\}^\infty, \mathcal{A}, P)$. If condition (3.1) holds, then:

$E(Y_\rho) = (2-\rho)/(2-2\rho) \approx 1.$

Proof. The term $P(Y_\rho = k)$ can be simplified to

$$
\begin{aligned}
P(Y_\rho = k) &= \sum_{j=k}^{\infty} \binom{j-1}{k-1} \rho^{2(k-1)} (1-\rho)^{2\left((j-1)-(k-1)\right)} 2\rho(1-\rho), \\
&= \rho^{2(k-1)} 2\rho(1-\rho) \sum_{j=k}^{\infty} \binom{j-1}{k-1} \big(1-\rho(2-\rho)\big)^{(j-k)} \\
&= \rho^{2(k-1)} 2\rho(1-\rho) \cdot \frac{1}{\big(\rho(2-\rho)\big)^k} \\
&= \frac{2(1-\rho)}{\rho} \left(\frac{\rho}{2-\rho}\right)^k \\
&= \frac{2-2\rho}{2-\rho} \left(\frac{\rho}{2-\rho}\right)^{k-1} \\
&= \varrho(1-\varrho)^{k-1},
\end{aligned}
$$

where $\varrho := (2-2\rho)/(2-\rho)$. In this calculation, the identity

$$
\frac{1}{\big(\rho(2-\rho)\big)^k} = \sum_{j=k}^{\infty} \binom{j-1}{k-1} \big(1-\rho(2-\rho)\big)^{(j-k)}
$$

arises from considering the Taylor series around $t_0 = 0$ of the function

$$
f : (-1,1) \to \mathbb{R}, t \mapsto (1-t)^{-k} = \sum_{j=0}^{\infty} \binom{j+k-1}{k-1} t^j = \sum_{j=k}^{\infty} \binom{j-1}{k-1} t^{(j-k)}
$$

at $t = 1-\rho(2-\rho)$.
Since $\varrho = (2-2\rho)/(2-\rho) \in (0,1)$ for $\rho \in (0,1)$, the function $p : \mathbb{N} \to [0,1],\ k \mapsto P(Y_\rho = k)$ is the density of the *geometric distribution* Geo_ϱ on $2^{\mathbb{N}}$ with parameter ϱ. If we consider, motivated by the above arguments, the term $\mathrm{Geo}_\varrho(\{k\}) = P(Y_\rho = k)$ as the probability that exactly k comparisons of indices are necessary to compare two infinite 0/1 sequences that are stored in sparse format, with the ones occurring independently at each position with probability ρ, the expected number of individual index comparisons is simply the expectation of this distribution

$$
E(Y_\rho) = E(\mathrm{Geo}_\varrho) = 1/\varrho = (2-\rho)/(2-2\rho).
$$

□

The number $E(Y_\rho) = (2-\rho)/(2-2\rho)$ tends fast to one as matrix density decreases

$$
E(Y_\rho) = \frac{2-\rho}{2-2\rho} = 1 + \frac{\rho}{2-2\rho} \xrightarrow[\rho \to 0]{} 1,
$$

even though the we are considering sequences of unbounded length: For a comparably high density of $\rho = 0.2$ (cf. Table 3.1) we would expect 1.125 comparisons, for $\rho = 0.1$ only $1.0\overline{5}$, and for $\rho = 0.01$ only $1.0\overline{05}$ comparisons.

Iterated Intersection of Finite Random 0/1 Sequences. The infinite sequence model of the previous paragraph is not suited for an analysis of iterated intersections of 0/1 sequences, because (under any reasonable assumptions) a finite number of such sequences will have an empty intersection with probability zero. Thus, we modify our model to deal with finite sequences: We look at the sequences as the result of m independent repetitions of a 0/1 experiment where the one has probability ρ. Formally, this model can be stated as

$$\left(\{0,1\}^m, \bigotimes_{i=1}^m 2^{\{0,1\}}, \mathrm{Bi}_{m,\rho}\right),$$

the m-fold product of the model $\left(\{0,1\}, 2^{\{0,1\}}, \mathrm{Bi}_{1,\rho}\right)$ that describes a single experiment (here, $\mathrm{Bi}_{m,\rho}$ denotes the *binomial distribution* with parameters m and ρ). This finite model is, of course, subject to the same criticism as its infinite brother.
To analyze the *sequence intersection algorithm*, consider the following *random experiment*. Initialize an index set R as $R := \{1, \dots, m\}$, draw one sequence from $\{0,1\}^m$ after the other at random, and update the set R by intersecting it with the sequence's support; this process is continued forever. Let a random variable $X_{m,\rho}$ count the number of sequence intersections until R becomes empty for the first time.

3.2.2 Lemma (Iterated Intersection of Random 0/1 Sequences)
Let $\rho \in (0,1)$ and $\left(\{0,1\}^m, \bigotimes_{i=1}^m 2^{\{0,1\}}, \mathrm{Bi}_{m,\rho}\right)$ be a probability space. Let further $X_{m,\rho}$ be a random variable that returns for an infinite number of randomly drawn sequences $a_1, a_2, \dots$ from $\left(\{0,1\}^m, \bigotimes_{i=1}^m 2^{\{0,1\}}, \mathrm{Bi}_{m,\rho}\right)$ the smallest number $k-1$ such that $\cap_{i=1}^k \operatorname{supp} a_i = \emptyset$. Then:
$E(X_{m,\rho}) \le m\rho/(1-\rho) \approx m\rho = \mu$.

Proof. The first step to compute $E(X_{m,\rho})$ is to note that the probability for k sequences to have a common one in some place is ρ^k, not to have a common one in some place is $1-\rho^k$, to have empty intersection in all places is $(1-\rho^k)^m$, and the probability for k sequences to have nonempty intersection in some of their m places is

$$P(X_{m,\rho} \ge k) = 1 - (1-\rho^k)^m, \quad k \in \mathbb{N}.$$

The expectation can now be computed as

$$\begin{aligned}
E(X_{m,\rho}) &= \sum_{k=1}^{\infty} k P(X_{m,\rho} = k) \\
&= \sum_{k=1}^{\infty} P(X_{m,\rho} \ge k) = \sum_{k=1}^{\infty} 1 - (1-\rho^k)^m = \sum_{k=1}^{\infty} 1^m - (1-\rho^k)^m \\
&\le \sum_{k=1}^{\infty} m 1^{m-1}\left(1-(1-\rho^k)\right) = \sum_{k=1}^{\infty} m\rho^k = m\left(\frac{1}{1-\rho} - 1\right) \\
&= m\rho/(1-\rho).
\end{aligned}$$

Here, the inequality

$$1^m - (1-\rho^k)^m \le m\, 1^{m-1}\left(1-(1-\rho^k)\right)$$

follows from applying the mean value theorem to the function $f : \mathbb{R} \to \mathbb{R}, t \mapsto t^m$. □

Considering the term $1/(1-\rho)$ as a constant, we arrive indeed at about $m\rho = \mu$ sequences that have to be intersected. Note that this number does not count the number of *operations* in the iterated sequence intersection algorithm, but the number of *intersections*.

3.2.4 Empty Columns, Empty Rows, and Row Singletons

Reductions P0, P1, and P2 are trivial and there is not much to say about their implementation. To find *empty columns* it is enough to go once through the objective and through the `cnt[]` array of the matrix's column representation, which can be done in $O(n)$ time. *Empty rows* are identified analogously in $O(m)$ time. The neighbors of a *row singleton* j are identified as follows: Scan column j; each nonzero entry identifies a row and all entries in this row, except for the singleton j itself, denote neighbors of the singleton and can be eliminated. Note that a nonzero in a row is used at most once to identify the neighbor of a row singleton, i.e., the routine has linear *amortized* running time over all passes.

3.2.5 Duplicate and Dominated Columns

Elimination of *duplicate columns* is a striking example for the effectiveness of even very simple preprocessing rules. Table 3.3, that gives statistics on the success of `BC`'s individual non LP based preprocessing subroutines when applied to the Hoffman & Padberg [1993] airline crew scheduling test set (in "Pass" many passes), shows the impact of this simple reduction. A quick glance at the table is enough to see that removing duplicate columns is the most significant preprocessing operation in terms of reduction in the number of nonzeros and columns (but not in rows, of course). One reason for this was already mentioned earlier: The `acs` problems, as many other "real world" set partitioning instances, were set up using automatic column generation procedures that produce the same columns more than once. A second reason is that identical columns can very well correspond to different activities: In airline crew scheduling, for example, two rotations may service the same flight legs, but on different routes at different costs.

The *implementations* of the literature seem to identify duplicate columns by comparisons of all pairs of columns enhanced by hashing techniques. Hoffman & Padberg [1993] compute a hash value for each column, *quicksort* the columns with respect to this criterion, and compare all pairs of columns with the same hash value. The hash value itself is the sum of the indices of the first and the last nonnull entry in a column. Atamturk, Nemhauser & Savelsbergh [1995] use the same algorithm, but a more sophisticated hash function: They assign a random number to each row and the hash value of a column is the sum of the random numbers corresponding to its nonnull entries. Both procedures do, in the worst case, a quadratic number of comparisons of two columns.

`BC`'s algorithm does an (expected) number of $O(n \log n)$ comparisons by simply (quick)sorting the columns *lexicographically*. We remark that this strategy is particularly easy to implement calling, e.g., the `C`-library's `qsort()`-function. In practice, one can slightly improve the running time by applying some linear time *presorting* operation to the columns using, e.g., some hashing technique. `BC` puts the columns into "buckets" according to the column `cnt` (see Subsection 3.2.2), i.e., the number of nonnull entries, and sorts the individual buckets as described above.

To estimate the *expected running time* of `BC`'s quicksorting procedure, we resort to Lemma 3.2.1 that states that in a certain uniform probabilistic model the expected number of operations to compare two random columns is *constant* (for bounded matrix density). Since uniform distribution of the sorted items in the partitions is an invariant of the quicksort algorithm, this results in an expected complexity of the complete procedure for removing duplicate columns of $O(n \log n)$ operations.

Name	Original		$P3'$	Pass	P2		$P4'$	P4	$P5^{16}$	$P7'$		P8
	Rows	Cols			Rows	Cols				Rows	Cols	
nw41	17	197	20	1	0	0	0	0	0	0	0	0
nw32	19	294	52	2	1	0	0	0	0	0	0	1
nw40	19	404	68	1	0	0	0	0	0	0	0	0
nw08	24	434	80	2	0	0	0	0	0	0	0	5
nw15	31	467	2	2	2	0	0	0	0	0	0	0
nw21	25	577	151	1	0	0	0	0	0	0	0	0
nw22	23	619	88	1	0	0	0	0	0	0	0	0
nw12	27	626	173	2	0	0	0	0	0	0	0	2
nw39	25	677	110	1	0	0	0	0	0	0	0	0
nw20	22	685	119	1	0	0	0	0	0	0	0	0
nw23	19	711	275	5	1	0	0	0	0	0	0	6
nw37	19	770	131	1	0	0	0	0	0	0	0	0
nw26	23	771	229	4	0	0	0	0	22	2	6	0
nw10	24	853	206	2	0	0	0	0	0	0	0	4
nw34	20	899	149	1	0	0	0	0	0	0	0	0
nw28	18	1210	385	2	0	0	0	0	226	0	0	0
nw25	20	1217	373	1	0	0	0	0	0	0	0	0
nw38	23	1220	336	2	0	0	0	0	0	0	0	3
nw27	22	1355	429	1	0	0	0	0	0	0	0	0
nw24	19	1366	440	1	0	0	0	0	0	0	0	0
nn01	18	1072	89	2	1	0	0	0	0	0	0	0
nn02	23	1079	255	3	0	0	0	0	0	0	0	4
nw35	23	1709	306	1	0	0	0	0	0	0	0	0
nw36	20	1783	375	1	0	0	0	0	0	0	0	0
nw29	18	2540	506	1	0	0	0	0	0	0	0	0
nw30	26	2653	769	1	0	0	0	0	0	0	0	0
nw31	26	2662	839	1	0	0	0	0	0	0	0	0
nw19	40	2879	737	2	0	0	0	0	0	0	0	8
nw33	23	3068	653	1	0	0	0	0	0	0	0	0
nw09	40	3103	800	2	0	0	0	0	0	0	0	7
nw07	36	5172	2065	2	0	0	0	0	0	0	0	3
aa02	531	5198	0	4	1	0	98	70	394	1	11	0
nw06	50	6774	825	2	0	0	0	0	0	0	0	13
aa06	646	7292	5	4	0	0	63	72	317	2	5	2
kl01	55	7479	676	3	0	0	4	4	0	0	0	0
aa05	801	8308	0	5	11	20	107	147	611	3	8	0
aa03	825	8627	1	4	17	52	106	138	366	4	69	2
nw11	39	8820	2863	2	0	0	0	0	0	0	0	11
nw18	124	10757	2779	2	0	0	0	0	1	0	0	43
us02	100	13635	2331	2	0	0	54	1	0	0	0	1
nw13	51	16043	5139	2	0	0	0	0	0	0	0	3
us04	163	28016	13005	4	0	0	36	25	4551	1	0	3
nw03	59	43749	4787	2	0	0	0	0	0	0	0	6
nw01	135	51975	1906	1	0	0	0	0	0	0	0	0
us03	77	85552	39362	4	0	0	14	13	0	0	0	0
nw02	145	87879	2621	1	0	0	0	0	0	0	0	0
nw17	61	118607	40427	2	0	0	0	0	0	0	0	7
nw14	73	123409	28235	2	0	0	0	0	0	0	0	5
nw16	139	148633	148494	2	1	0	0	0	0	0	0	138
nw05	71	288507	86012	2	0	0	0	0	0	0	0	13
kl02	71	36699	20157	2	0	0	2	0	0	0	0	0
us01	145	1053137	682495	2	0	0	58	1	0	0	0	0
nw04	36	87482	41292	2	0	0	0	0	0	0	0	1
aa04	426	7195	0	4	0	0	31	52	40	0	0	0
aa01	823	8904	9	4	0	0	70	134	214	2	8	1
$\sum$ 55	6378	2305749	1134631	115	35	72	643	657	6742	15	107	292

Table 3.3: Analyzing Preprocessing Rules.

Elimination of *dominated columns* is a generalization of removing duplicate columns: The latter reduction is the special case where the set J is restricted to contain just a single member. The main difficulty in implementing the general rule is, of course, to find this set J in an efficient way. The only known algorithm seems to be enumeration, which is out of the question even for medium sized problems. This and the already mentioned property of the LP relaxation to keep dominated columns at zero values in optimal solutions anyway can explain the apparent lack of implementations of this rule in the literature and the preprocessor of our algorithm `BC` does also *not* search for dominated columns. We remark that there are heuristic procedures for the set covering variant of the reduction, see Beasley [1987].

3.2.6 Duplicate and Dominated Rows

Removing *duplicate rows* of a 0/1 matrix is equivalent to removing duplicate columns from the transpose. Hence, the *implementation* of this operation is governed by exactly the same considerations as for the columns. The probabilistic *analysis* carries over as well, since it assumes only the independent random occurrence of ones in the matrix with probability equal to the density. The result is an expected number of $O(m \log m)$ index comparisons to remove all duplicate rows.

This favorable running time does not entirely show up in our *computations* for the `acs` test set. The reason is that these instances come ordered in a "staircase form" with sequences of consecutive ones in the rows, which increases the probability of common nonzeros in two rows. This does not fit with the analysis and leads to an increase in running time of the procedure.

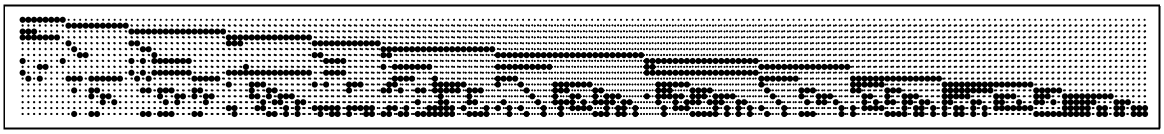


Figure 3.2: Bringing Set Partitioning Problem `nw41` (17×197) into Staircase Form.

A simple way out of this problem would be to permute rows and columns randomly, but staircase form also has its advantages elsewhere. Since removing duplicate rows is not a bottleneck operation, we opted to leave the matrices as they are and employ more elaborate *presorting* techniques instead. We use a two level hashing, first assigning the rows to "buckets" according to the number of nonzero entries (as we did for the columns) and then subdividing these buckets further into subbuckets of rows with the same sum of nonzero indices. The individual subbuckets are sorted using *shakersort* (an alternating *bubblesort* with almost linear expected running time for small arrays) for "small" buckets ($\leq$ 8 elements in our implementation) and quicksort else. With this tuning, removing duplicate rows takes about the same time as removing duplicate columns.

As was already pointed out earlier, removing rows from a set partitioning problem is particularly advantageous for branch-and-cut solvers, because it reduces the size of the LP basis which has a quadratic impact on parts of the LP time like factorization. For this reason, extending the removal of duplicate rows to *dominated rows* is of significant interest. Use of the latter reduction is reported by Hoffman & Padberg [1993] and Atamturk, Nemhauser & Savelsbergh [1995].

BC's procedure to remove dominated rows is based on the sequence intersection algorithm of Subsection 3.2.3. It exploits the simple observation that the set of rows that dominate some row $A_{r\cdot}$ can be expressed as the "intersection of the columns in this row"

$$\bigcap_{j \in \operatorname{supp} A_{r\cdot}} \operatorname{supp} A_{\cdot j}.$$

The method is simply to compute this set by intersecting the columns iteratively, stopping as soon as either r is the only row left in the set or when there is exactly one additional last candidate row, that is compared directly to r.

We will argue in the next paragraph that one can expect the stopping criterion in this procedure to apply after about μ intersections of columns. Considering about μ nonzero elements in each column and, if necessary, one more (see Subsection 3.2.3) in the final comparison of two columns, this results in an expected $O(\mu^2)$ operations per row. Doing this m times for m rows, we expect to remove all dominated rows in $O(m\mu^2)$, or, if one likes this better, in $O(m^3\rho^2)$ steps.

To estimate the number of column *intersections* until the stopping criterion applies, we will make use of Lemma 3.2.2: We claim that $E(X_{m-1,\rho})$ is an upper bound on the expected number of intersections in the column intersection algorithm. To see this, note that all columns considered in the algorithm have an entry in row r, but their remainders are distributed according to the model $\left(\{0,1\}^{m-1}, \bigotimes_{i=1}^{m-1} 2^{\{0,1\}}, \mathrm{Bi}_{m-1,\rho}\right)$ for one row less, namely, row r. In this $m-1$-row model, $E(X_{m-1,\rho})$ counts the number of intersections until no row is left, which corresponds, in the original m-row model, to the number of intersections until only row r is left. This ignores, of course, the possibility to stop earlier if $|R| \leq 2$ (in the m-row model), and hence $E(X_{m-1,\rho})$ is an upper bound on the number of columns considered by the algorithm.

3.2.7 Row Cliques

Elimination of columns that extend a *row clique* seems to have been used for computation by Chu & Beasley [1995]. Using the same rules as Hoffman & Padberg [1993] (P1,P2,P3′,P4′, and P6) otherwise, they report a slightly bigger reduction in problem size. A straightforward way to *implement* the rule would be to tentatively set each variable to one, all its neighbors to zero and check whether this contradicts some equation. This algorithm requires, however, one row scan for each nonzero element of the matrix. This is not acceptable, and as far as we know, nobody has suggested a better method to implement this reduction. But we will argue now and give some computational evidence that an exact implementation of this rule is not worth the effort.

We would expect from the analysis of the sequence intersection algorithm in Subsection 3.2.3 that the probability of a column to intersect many other columns (in a row) is extremely small such that the chances to eliminate such a column are at best questionable. This argument can be made more precise using the probabilistic model of Subsection 3.2.3. As we computed in the proof of Lemma 3.2.2 ibidem, the probability for k columns to intersect in some of their m rows is $P(X_{m,\rho} \geq k) = 1-(1-\rho^k)^m$. This probability, which increases with larger ρ values and larger m and decreases with larger k, respectively, is almost zero for the applications that we have in mind: Considering "unfavorable" settings like a rather high density of $\rho = 0.1$ and a comparably large number $m = 1000$ of rows (cf. Table 3.1), and a tiny row clique of just 8 columns, this number is $1-(1-10^{-8})^{1000} \leq 10^{-6}$.

For this reason, and for the sake of speed, BC implements a *heuristic* version of P5 that considers only rows with at most some constant number M of entries (16, in our implementation). The rule is denoted by $\mathrm{P5}^M$ and the effect of $\mathrm{P5}^{16}$ on the Hoffman & Padberg [1993] test set can be seen from Table 3.3. To satisfy our curiosity, we have tested the full rule P5 as well: Applying P5 instead of $\mathrm{P5}^{16}$ resulted in 18 more rows and 8,140 more columns removed by the preprocessor, at the expense of several days of computation time.
BC's *implementation* of rule $\mathrm{P5}^M$ is based on the formula

$$\bigcap_{j\in\operatorname{supp} A_{r\cdot}} \gamma(j) = \bigcap_{j\in\operatorname{supp} A_{r\cdot}} \left(\bigcup_{s\in\operatorname{supp} A_{\cdot j}} \operatorname{supp} A_{s\cdot} \right).$$

Here, the neighbors of (all columns in) row r are determined by intersecting the neighbor sets $\gamma(j)$ of the individual columns and these are computed by scanning all corresponding rows. The complete routine determines, for some given matrix A, all rows with at most M nonzeros, and applies the above procedure to each of these rows.
To compute the *expected complexity* of this algorithm, we consider again the probabilistic model of Subsection 3.2.3, that looks at each row of the $m \times n$ 0/1 matrix A as the result of n independent 0/1 experiments with a probability of ρ for one. If the random variable $Y_{n,\rho}$ counts the number of nonzeros in a row, we would expect $\mathrm{P5}^M$ to take

$$O\big(mP(Y_{n,\rho} \le M) \cdot M \cdot mn\rho^2\big) = O\big(Mn\mu^2 P(Y_{n,\rho} \le M)\big)$$

operations. The first term $mP(Y_{n,\rho} \le M)$ in this expression is the expected number of rows with at most M nonzeros. Each of these M nonzeros (term two) corresponds to a column with $m\rho$ entries on average, and for each of these entries, we have to scan a row with about $n\rho$ entries.
Arguments in the next paragraph suggest that $P(Y_{n,\rho} \le M) \le (M+1)e^{-n\rho}(n\rho)^M/(1-\rho)^M$. This results in a total of

$$O\big(M^2 n\mu^2 e^{-n\rho}(n\rho)^M/(1-\rho)^M\big)$$

expected operations to perform $\mathrm{P5}^M$. For a numerical example, consider "unfavorable" parameters of $M = 16$, $m = 100$, $n = 1000$, $\rho = 0.1$; the result is less than 10^{-3}.
The upper bound on the probability $P(Y_{n,\rho} \le M)$ can be computed as follows:

$$\begin{aligned}
P(Y_{n,\rho} \le M) &= \sum_{i=0}^{M} \binom{n}{i} \rho^i (1-\rho)^{n-i} \\
&\le (1-\rho)^n \sum_{i=0}^{M} \left(\frac{n\rho}{1-\rho}\right)^i \\
&\le (1-\rho)^n (M+1) \left(\frac{n\rho}{1-\rho}\right)^M \qquad \text{assuming } n\rho/(1-\rho) > 1 \\
&= (M+1)(1-\mu/m)^{m\cdot n/m} \cdot \left(\frac{n\rho}{1-\rho}\right)^M \\
&\le (M+1)\; e^{-\mu n/m}(n\rho)^M/(1-\rho)^M \\
&= (M+1)\; e^{-n\rho}(n\rho)^M/(1-\rho)^M.
\end{aligned}$$

3.2.8 Parallel Columns

Elimination or merging of *parallel columns* has been used by Hoffman & Padberg [1993]. The rule requires some book keeping to be able to undo the merging of columns into compound columns once a solution has been found; note that repeated merging can result in compound columns that correspond to *sets* of original variables. BC does *not* implement this rule, and we do not analyze it here.

3.2.9 Symmetric Differences

The *symmetric difference* rule P7′ is particularly attractive, because it leads to the elimination of both rows and columns. An *implementation* based on checking all triples of rows is a disaster, but the column intersection technique of Subsection 3.2.3 can be used to design an efficient procedure. The algorithm in BC computes for each row s the column j with the smallest support. Now we distinguish two cases:

(i) Column j is supposed to be contained in the symmetric difference of row s and some, not yet known, row t. The only possible rows to cover the symmetric difference are the rows in $\operatorname{supp} A_{j\cdot} \setminus \{s\}$. For each such row r, the potential rows t are limited to the set

$$\bigcap_{i \in \operatorname{supp} A_{s\cdot} \setminus \operatorname{supp} A_{r\cdot}} \operatorname{supp} A_{\cdot i}$$

that agree with s on the columns that are not covered by r. This set is computed using iterative column intersection and each of the resulting candidates t is checked.

(ii) Column j is supposed to be contained in the intersection of row s with some row $t \neq s$; clearly, $s \in \operatorname{supp} A_{\cdot j}$. For each such row t, the symmetric difference $\operatorname{supp}(A_{s\cdot} - A_{t\cdot})$ is computed and a row r covering this difference, i.e., from the set

$$\bigcap_{i \in \operatorname{supp}(A_{s\cdot} - A_{t\cdot})} \operatorname{supp} A_{\cdot i}$$

is determined by means of iterative column intersection.

A heuristic estimate of the *running time* of this procedure is as follows. In case (i), we expect to consider a column with less than the mean of $\mu = m\rho$ nonzeros. For each nonzero, we apply the column intersection algorithm which takes $O(\mu^2)$ operations and yields (less than) $O(\mu)$ candidate rows t. For each of these candidate rows, we scan three rows which takes $O(n\rho)$ operations. We would thus expect that performing (i) once for each of m rows takes a total of $O\big(m \cdot \mu \cdot (\mu^2 + \mu n\rho)\big) = O\big(m\mu^3(1 + n/m)\big)$ operations. In case (ii), we consider the same column with expected μ nonzeros. For each of these entries, we scan two rows to compute a symmetric difference, which takes $O(n\rho)$ operations. Then the column intersection algorithm with $O(\mu^2)$ operations is applied, and, eventually, μ candidate rows checked, taking another $O(\mu n\rho)$ operations. This results in $O\big(m \cdot \mu \cdot (n\rho + \mu^2 + \mu n\rho)\big)$ overall operations, which is of the same order as in the first case. Thus, the total expected number of operations in this procedure is an acceptable

$$O\big((m+n)\mu^3\big) = O(n\mu^3).$$

The same technique could also be used to implement the generalization P7 of this rule, but, unfortunately, this was *not* done for the preprocessing module of BC.

3.2.10 Column Singletons

Elimination of rows and columns using *column singletons* is a special case of a more general *substitution* operation. This technique works as follows. Consider an integer program

$$\min\ w_0 + w^{\mathrm{T}}x \quad Ax = b,\ Cx \le d,\ l \le x \le u,\ x \in \mathbb{Z}^n$$

with an objective that contains a constant offset term w_0, some equations, inequalities, and lower and upper bounds (possibly $\pm\infty$) on the variables. Without loss of generality we can assume a_{11} to be positive and bring the first equation into the form

$$x_1 = b_1/a_{11} - \sum_{i\ge 2} a_{1i}/a_{11}x_i.$$

This equation can be used to eliminate x_1 by Gaussian elimination in the objective, the other constraints, and the bounds. The result of this operation is one variable (x_1) and one constraint ($A_1.x = b_1$) less, potential *fill* in the equations and inequalities, and a transformation of the two bounds $l_1 \le x_1 \le u_1$ and the integrality stipulation $x_1 \in \mathbb{Z}$ into the form

$$b_1 - a_{11}u_1 \le \sum_{i\ge 2} a_{1i}x_i \le b_1 - a_{11}l_1 \qquad \text{and} \qquad b_1/a_{11} - \sum_{i\ge 2} a_{1i}/a_{11}x_i \in \mathbb{Z}. \tag{3.2}$$

Sometimes these constraints will be redundant. One restrictive but relevant and easily detectable case is when the transformed integrality stipulation on the right of (3.2) holds because the equations $Ax = b$ have integer data, i.e., $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$, and $a_{11} = 1$, i.e., the pivot is one and there is no division in the Gaussian elimination, and when, in addition, the transformed bounds are redundant because

$$b_1 - u_1 \le \sum_{i\ge 2} \min\ \{a_{1i}l_i, a_{1i}u_i\} \le \sum_{i\ge 2} \max\ \{a_{1i}l_i, a_{1i}u_i\} \le b_1 - l_1. \tag{3.3}$$

Under these circumstances, the substitution results in a reduction in the number of rows and columns of the program and we speak of *preprocessing by substitution.* This technique is widely used in LP and MIP solvers, e.g., in CPLEX [1997]. To control fill, implementations generally restrict substitution to columns with few entries, like singleton columns, or to rows with few entries, like *doubleton* rows, see Bixby [1994].

An obstacle to the application of this rule to set partitioning problems is that the bound redundancy criterion (3.3) is computationally useless in this application. Namely, assuming that all fixed variables have already been removed earlier, condition (3.3) reads $0 \le |\operatorname{supp}(A_1. - e_1)| \le 1$. This can and will hold exactly for the trivial case of a doubleton row. Another criterion is thus needed for set partitioning problems, and

$$A_{s\cdot} \ge A_{r\cdot} - e_1$$

for another row $s \ne r$, as suggested in rule P8, is a suitable choice to guarantee (3.2).

Row s can be identified by column intersection, and this yields a running time of at most

$$O(m \cdot \mu^2 \cdot n\rho) = O(n\mu^3)$$

operations: At most m singletons can be eliminated, each candidate requires one application of the column intersection algorithm with $O(\mu^2)$ operations, and the resulting candidate row is checked and substituted into the objective in $O(n\rho)$ operations.

Note that the result of a sequence of substitutions is independent from the *elimination order*, but the amount of *work* is not. The three equation example in Figure 3.3 illustrates how this is meant. In the example, column singleton x_1 can be eliminated from equation (1). After the first row and column have been deleted, x_2 can be eliminated using equation (2), and finally x_3 from (3). Doing the substitutions in this order —x_1 from (1), x_2 from (2), and x_3 from (3)— produces no fill. Given the original matrix A and this ordered "substitution history" list, one can reproduce a P8-processed problem in time proportional to the number of nonzeros in the substitution equations by substituting these equations in the given order into the objective and by eliminating rows and columns. Using some other order leads to additional work. For example, substituting in Figure 3.3 for x_2 first using (2) produces two nonzeros in equation (1) at x_3 and x_4 and continuing to substitute in any order results in a worst possible fill.

$$\boxed{\begin{array}{llll} \mathbf{x_1} + x_2 & & = 1 & (1) \\ & \mathbf{x_2} + x_3 + x_4 & = 1 & (2) \\ & \quad\quad\; \mathbf{x_3} + x_4 & = 1 & (3) \end{array}}$$

Figure 3.3: Eliminating Column Singletons in the Right Order to Avoid Fill.

This phenomenon can become relevant in a *branch-and-cut* context on two occasions. First, when a solution to a preprocessed problem has been found and substitutions have to be *reversed*, this should be done in reverse order of the substitution history by computing the values of the substituted variables from the corresponding original equations; note that this does in general not work with some other elimination order. And second, when a preprocessed subproblem in the searchtree has to be *reproduced*. `BC` does not store the objectives of preprocessed subproblems, because this would require an array of $O(n)$ `double` variables at each node. Instead, the objective is recomputed from scratch each time. Doing this without making any substitutions in the matrix requires a zero fill substitution order. The consequence is to store the order of column singleton substitutions on a "history stack".

A final word has to be said about the impact of substitution on the objective function and the solution of the *LP relaxation*. Many set partitioning problems from scheduling applications have nonnegative objectives, and so do the `acs` problems. Substitution destroys this property by producing negative coefficients. Unexpectedly, the LP relaxations of problems that were preprocessed in this way become difficult to solve, probably because the start basis heuristics do not work satisfactory any more. But fortunately, there is a simple way out of this dilemma. The idea to counter the increase in LP time is to make the objective *positive* again by adding suitable multiples of the rows. `BC`'s procedure implements the formula

$$w_0 + w^{\mathrm{T}}x + \pi(Ax - \mathbf{1}),$$

where

$$\pi_r := \max{}_{j \in \operatorname{supp} A_{r\cdot}: w_j < 0} |w_j| / |\operatorname{supp} A_{\cdot j}|, \quad r = 1, \ldots, m.$$

The impact of rule P8 on the `acs` test can be read from Table 3.3: A nice success is that problem `nw16` can be *solved* by preprocessing with this rule. P8 has also proved valuable in dealing with set packing constraints (see the vehicle availability constraints in Chapter 4) in a set partitioning solver: Transforming such inequalities into equations introducing a slack variable produces column singletons that can potentially be eliminated.

3.2.11 Reduced Cost Fixing

Reduced cost fixing is another example of a strikingly simple and effective preprocessing operation. We draw the reader's attention to Table 3.1, where a comparison of the "Presolved" and "Presolved: LP-based" columns indicates that reduced cost fixing (based on the knowledge of a good upper bound from the primal heuristic) accounts for a reduction in the number of columns and nonzeros of one order of magnitude.

3.2.12 Probing

Probing (a rule that we have not completely specified) belongs to a group of *expensive* preprocessing operations in the sense that they require the exact or approximate solution of linear programs. There is additional information gained in this way that makes these operations powerful (P10 is, for example, stronger than P5), but there is of course a delicate trade-off between time spent in preprocessing and solving the actual problem.

An *implementation* of probing by tentatively setting variables to their bounds can be done with postoptimization techniques, using advanced basis information: Having an optimal basis at hand, one sets one variable at a time to one of its bounds and reoptimizes with the dual simplex method; after that, one reloads the original basis and continues in this way. This method has the disadvantage that there is no control on the amount of time spent in the individual LPs. Some control on the computational effort is gained by limiting the number of simplex iterations in the postoptimization process at the cost of replacing the optimal LP value with some lower bound. If the iteration limit allows only "few" iterations, this offers the additional possibility to avoid basis factorizations using an *eta file* technique: In each probe, the basis is updated adding columns to the eta file; when the iteration limit is exceeded (or the problem solved), the original basis is restored by simply deleting the eta file. This technique is implemented in CPLEX [1997], but despite all these efforts, probing is still expensive.

`BC` uses probing of variables in its default *strong branching strategy*, (cf. Bixby, personal communication): Some set of candidate variables for probing are determined (the 10 most fractional ones), each of these is probed 25 dual simplex iterations deep, and any possible fixings are carried out; the remaining bound information is used to guide the branching decision.

3.2.13 Pivoting

We have seen in the introduction to this section that preprocessing is an effective tool to reduce the size of a given set partitioning problem and that techniques of this sort can help to solve these IPs faster. There is no reason to believe that this does not also work in the same way for the subproblems created by a branch-and-bound algorithm. Rather to the contrary, one would expect *iterated preprocessing* on subproblems to be even more effective since subproblems contain additional fixings due to branching decisions and the lower bound is better. To exploit this information, one would like to preprocess not only the original problem formulation, but also subproblems *repeatedly* throughout the branch-and-bound tree.

LP-based methods, on the other hand, live on maintaining *dual feasibility* of the basis: Instead of solving an LP from scratch each time a variable has been fixed in a branching decision or a cutting plane has been added, the dual simplex method is called to reoptimize the LP starting from the *advanced basis* obtained in the preceding optimization step.

These two principles —repeated problem reduction and maintenance of a dual feasible basis— can get into conflict. Reductions that do not interfere with a dual feasible basis are:

(i) Eliminating *nonbasic* columns.
(ii) Eliminating *basic* rows, i.e., rows where the associated slack or artificial variable is basic.

(i) it does obviously neither affect the basis itself nor its dual feasibility. (ii) is possible since the multiplier (dual variable) associated to a basic row r is zero. But then the reduced costs $\overline{w}^{\mathrm{T}} = w^{\mathrm{T}} - \pi^{\mathrm{T}} A = w^{\mathrm{T}} - \sum_{s \neq r} \pi_s A_s$. are not affected by removing row r and, moreover, if $\hat{A}_B$ denotes the matrix that arises from the basis matrix A_B by deleting row r and column e_r, this reduced basis $\hat{A}_B$ is dual feasible for the reduced problem.
Rule (ii) can be slightly extended with a pivoting technique to

(iii) Eliminating rows with zero multipliers.

The method is to reduce (iii) to (ii) by performing a primal *pivot* on $a_{rj} = 1$, where row r with $\pi_r = 0$ is supposed to be eliminated and j the unit column corresponding to its slack/artificial; "primal pivot" means that the slack/artificial column e_r is entering the basis. As $\pi_r = \overline{w}_j = 0$, this pivot will be dual degenerate. We are interested here in the case where row r is known to be a (linearly) redundant equation; then, its artificial variable is zero in any feasible solution and the pivot will also be primal degenerate. This *in-pivoting* procedure was developed by Applegate, Bixby, Chvátal & Cook [1994] for the solution of large scale TSPs and is implemented in CPLEX V2.2 and higher versions.
One possible *strategy* for iterated preprocessing in a branch-and-cut algorithm is thus the following. Apply the preprocessor as often as you like and eliminate rows and columns using (i)–(iii), doing in-pivoting prior to the actual elimination of rows where necessary. If a basic column was eliminated or fixed to one by the preprocessor, change its bounds, but leave it in the formulation, and do also not remove rows with nonzero multipliers form the formulation, even if the preprocessor detected their redundancy. If too much "garbage" accumulates, eliminate everything, discard the (useless) basis, and optimize from scratch.
One might wonder whether it is at all possible that redundant rows can have nonzero multipliers. Do not all row elimination rules (except for the column singleton rule P8), after elimination of certain columns, result in sets of duplicate rows where at most one representative can have a nonzero multiplier? The following simple example shows that this is not so and why. Consider the set partitioning problem

$$\begin{array}{llccccccc} & \min & 2x1 & + & x2 & & & = & 0 \\ (\text{c1}) & & x1 & + & x2 & + & y1 & = & 1 \\ (\text{c2}) & & x1 & & & & + \; y2 & = & 1 \\ & & & & & & x1,\; x2 & \in & \{0,1\}. \end{array} \tag{3.4}$$

Here, the variables y_1 and y_2 denote the artificial variables of the constraints $c1$ and $c2$, respectively. The first two columns of the constraint matrix correspond to the variables x_1 and x_2 and constitute an optimal basis for (3.4); the corresponding simplex *tableau* reads

$$\begin{array}{llccccccc} & \min & & & - & y1 & - \; y2 & = & -2 \\ (\text{c1}) & & & x2 & + & y1 & - \; y2 & = & 0 \\ (\text{c2}) & & x1 & & & & + \; y2 & = & 1 \\ & & & & & & x1,\; x2 & \in & \{0,1\}. \end{array}$$

The values of the dual variables are both nonzero: $\pi_1 = \pi_2 = 1$.

Suppose that in this situation a preprocessor investigates formulation (3.4) and finds out that variable x_2 can be eliminated. (Consider the example as a part of a bigger problem and ignore the possibility to solve the problem by fixing also x_1 to one.) Eliminating x_2 (in the preprocessing data structures, not in the LP) results in two identical rows $c1$ and $c2$. Suppose the preprocessor finds this out as well and suggests to eliminate one of them. But whether we try to eliminate $c1$ or $c2$, neither of these suggestions is compatible with dual feasibility of the basis and we can not eliminate rows and columns that we know are redundant. Since linear dependent and much less duplicate rows can not be contained in a basis, there must be some *fixed variable* in the basis. Clearly, there must be alternative optimal bases that do not contain one, or some, or all fixed variables: We suffer from primal *degeneracy*.
The degeneracy phenomenon that we have just described does not only appear in theory, but is a major obstacle to the solution of set partitioning problems by branch-and-cut. Unexpectedly, it turns out that for the airline test set often almost half of the basis matrices consist of fixed variables, "blocking" the same number of rows from possible elimination. It is clear that a larger number of rows and a larger basis has a negative impact on LP time.
This problem can be overcome by a novel *out-pivoting* technique that forces fixed variables to leave the basis. The method is to perform one "dual pivot" with the fixed basic variable leaving the basis (allowing slacks/artificials to enter). As the leaving variable is fixed, this pivot is primal degenerate, but the dual solution changes, and the entering variable is determined in such a way that optimality is re-established, i.e., by a ratio test.
Out-pivoting is available in CPLEX release V5.0 and higher version. Its use to eliminate fixed variables from the basis allows for significant additional problem reductions while at the same time maintaining dual feasibility. We remark that although the method is best possible in the sense that it requires just a single dual pivot for each fixed basic variable, out-pivoting is not cheap. Table 3.4 shows that 7% of the total running time that our branch-and-cut algorithm `BC` needs to solve the airline test set is spent in out-pivoting (column Pvt under Timings). And the number of out-pivots *exceeds* the number of other pivots by a factor of about five!

3.2.14 The Preprocessor

Combining the routines of the previous subsections yields the *preprocessor* of our set partitioning solver `BC`. The module consists of 67 kilobytes of source code in 10,000 lines.
The module does not work on the LP itself, but on a (possibly smaller) auxiliary representation of the problem where reductions can be carried out no matter what the LP basis status is. The preprocessor is called for the first time prior to the solution of the first LP. All later invocations involve pivoting to maintain the dual feasibility of the basis. First, the basis is purged by pivoting out fixed variables (from previous invocations). Preprocessing starts with reduced cost fixing according to rule P9. Then the main preprocessing loop is entered that calls, in each pass, all the individual rules. First, a couple of column oriented reductions are carried out: P2 (row singletons) and P5[16] (row clique heuristic). Then the matrix is transposed, and row oriented operations follow: P4′ (duplicate rows), P4 (dominated rows), P7′ (symmetric difference), and P8 (column singletons). The matrix is transposed again for the next pass. This loop continues as long as some reduction was achieved. When no further reductions can be achieved, as many of the found ones as possible are transferred to the LP: Artificials of redundant rows are pivoted in and redundant nonbasic columns and redundant basic rows are eliminated from the LP. The reader can infer from Table 3.4 that the running time for this module is not a computational bottleneck for the entire branch-and-cut code.

3.3 Separation

"Branch-and-cut" — this term lists the two sources of power of the algorithms of this class. The second of these, the computation of *cutting planes*, aims at improving the quality of the current LP relaxation in the sense that the lower bound rises. If this can be achieved, it helps in fathoming nodes and fixing variables by preprocessing techniques, provides criteria for intelligent searchtree exploration, and, ideally, "pushes" the fractional solution toward an integral one. This, in turn, can be exploited for the development of heuristics by tracing histories of fractional variables etc., and there are certainly more of such *practitioner's* arguments in favor of cutting planes that are all based on the many algorithmically useful degrees of freedom in (as the name says) a *generic* branch-and-cut method. The *theoretical* justification for the use of cutting planes is perhaps even more convincing: By the general algorithmic results of Grötschel, Lovász & Schrijver [1988] we know that polynomial time separation allows for polynomial time optimization, and even if we give here the dual simplex algorithm's reoptimization capabilities (not to speak of the availability of suitable implementations) preference over the ellipsoid method's theoretical power, there is no reason to believe that not some of this favorable behaviour will show up in codes of the real world. And in fact, the number of implementations of this principle with successful *computational* experience is legion, see, e.g., Caprara & Fischetti [1997] for a survey.

The separation routines for *set partitioning* problems are based on the relation

$$P_I^=(A) = P_I(A) \cap Q_I(A)$$

between the set partitioning, the set packing, and the set covering polytope: To solve set partitioning problems, we can resort to cutting planes for the associated packing and covering polytopes. We have already pointed out in Section 1.2 why the polyhedral study of the latter two bodies is easier than the study of the first, and we have also listed in the Sections 1.8, 1.9, 2.5, and 2.6 many known types of valid and often even facet defining inequalities that qualify as candidates for cutting planes in a branch-and-cut code for set partitioning problems.

But not only these classes are available: *General cutting planes* suggest themselves as well: Gomory [1960] cuts, lift-and-project cuts, see Balas, Ceria & Cornuéjols [1993], or Martin & Weismantel [1997]'s feasible set cuts.

We have *selected* only a small number of them for our implementation: *Clique inequalities*, because they give facets, are easy to implement, numerically stable (only 0/1 coefficients), and sparse, *cycle inequalities* for the same reasons and because they can be separated exactly, and the *aggregated cycle inequalities* from the set packing relaxation of the set covering problem of Section 2.6 because we wanted to evaluate the computational usefulness of our aggregation technique. These cuts are all simple, but as the duality gaps in real world set partitioning problems are usually quite small, there is some justification for a strategy that opts for "whatever one can get in a short time".

We discuss in the following subsections the individual routines of our separation module. All of the procedures work with intersection graphs that we introduce in Subsection 3.3.1. Separation and lifting routines for inequalities from cliques are treated in Subsection 3.3.2, for cycles in Subsection 3.3.3, and for aggregated cycles in Subsection 3.3.4. A word on our strategies to call these routines can be found in the following Section 3.4.

3.3.1 The Fractional Intersection Graph

All of our separation routines will be combinatorial algorithms that work on *intersection graphs*. Namely, we look for our set packing inequalities on subgraphs of $G(A)$, the intersection graph of the set packing relaxation, and we identify aggregated cycle inequalities on subgraphs of the conflict graph $\mathfrak{G}(A)$ that is associated to the aggregation.

A quick calculation is enough to see that it is completely out of the question to set up $G(A)$ completely, and much less $\mathfrak{G}(A)$, and even if we could do this it is very unlikely that we could make any use of these gigabytes of information. But luckily, it follows from the nonnegativity of all nontrivial facets of set packing polytopes and the 2-connectedness of their support graphs, is well known, and was mentioned, for example, in Hoffman & Padberg [1993], that one can restrict attention to the "fractional parts"

$$G(A)[F] = G(A_{\cdot F}) \qquad \text{and} \qquad \mathfrak{G}(A)[\mathfrak{F}] = \mathfrak{G}(A_{\cdot F})$$

of these structures for separation purposes. These graphs are the *fractional intersection graph* and the *aggregated fractional intersection graph*, respectively. As there can be at most as many fractional variables as is the size of the basis as is the number of equations of the LP relaxation, this reduces, for "typical" real world set partitioning problems like the airline instances, the number of nodes from ten- to hundred thousands in $G(A)$ to some hundreds in $G(A_{\cdot F})$ by two to three orders of magnitude, and the number of edges even more. This is not so for the graph $\mathfrak{G}(A_{\cdot F})$, which is exponential by construction. We cope with this difficulty in a heuristic way by using only some subgraph of $\mathfrak{G}(A_{\cdot F})$. Note that the above mentioned 2-connectedness of the support graphs of facets makes it possible to restrict separation to individual 2-connected components of $G(A_{\cdot F})$ and of $\mathfrak{G}(A_{\cdot F})$.

Separating on the fractional variables only has the disadvantage that the resulting cutting planes have a very small support in comparison to the complete set of variables. One way to counter the stalling effects of "polishing" on a low dimensional face of the set partitioning polytope is to extend the support of cutting planes by *lifting*. Our overall separation *strategy* will be to reduce the effort to identify a violated inequality as much as possible by working on fractional intersection graphs, and we enhance the quality of whatever we were able to obtain in this first step *a posteriori* by a subsequent lifting step.

We turn now to the algorithmic *construction* of the fractional intersection graph. We treat only $G(A_{\cdot F})$ and do not discuss here how we set up a subgraph of $\mathfrak{G}(A_{\cdot F})$, because this is so intimately related to the separation of aggregated cycle inequalities that it is better discussed in this context in Subsection 3.3.4.

The *procedure* that we have implemented in `BC` sets up a new column intersection graph $G(A_{\cdot F})$ after the solution of every single LP, i.e., $G(A_{\cdot F})$ is constructed "on the fly", as Hoffman & Padberg [1993] say. Our routine uses two copies of the matrix $A_{\cdot F}$, one stored in column and the other in row major format. $A_{\cdot F}$ can be extracted from the column major representation of the "global" matrix A in time that is linear in the number of nonzeros of $A_{\cdot F}$. Next, we compute the neighbors of each column $j \in F$ by scanning its rows and store the result in a forward star adjacency list (see, e.g., Ahuja, Magnanti & Orlin [1989]). Under the assumptions of Subsection 3.2.3 we expect that this will take about $O(\mu\rho|F|^2)$ operations on average — fast enough to just forget about. We do not use a procedure to decompose $G(A_{\cdot F})$ into two connected components.

3.3.2 Clique Inequalities

We have already mentioned in the introduction of this section why we use *clique inequalities* as cutting planes in our branch-and-cut code `BC`: This class yields facets, it is easy to come up with separation and lifting heuristics, and such inequalities are sparse and pose no numerical difficulties. One must admit, however, that these appealing properties are strictly speaking outmatched by the unsatisfactory theoretical behaviour of these simple cutting planes. Clique separation is not only *$\mathcal{NP}$-hard*, see Garey & Johnson [1979], but, even worse, this class is contained in polynomial separable superclasses like orthogonality inequalities or matrix cuts. One could argue somewhat around the first difficulty, namely, we have implemented an exact clique separation routine as well and found that, even without any tuning, our heuristics already found nearly every violated clique inequality there was, and it is a little thing to tune the heuristic routines such that containment becomes equality. But we feel nevertheless that the above arguments show that it is not the right way to compensate the conceptual weakness in clique inequality separation by additional computational effort.
Our branch-and-cut code `BC` goes thus to the other extreme and concentrates on the computational advantages of heuristic clique detection by using only simple separation and lifting routines. We compute violated inequalities with a *row lifting* and a *greedy* heuristic, and a "semiheuristic" (the meaning of this term will become clear in the description of this method) *recursive smallest last* (RSL) procedure, and we lift the cutting planes that they return with tailor made procedures that fit with the separation routine's "philosophy" (these statements have been evaluated in computational experiments). These separation and lifting routines are described in the next paragraphs.

Row Lifting, Hoffman & Padberg [1993]. The idea of this separation routine is to exploit the knowledge of those cliques that are already encoded in the rows of the matrix A to design a very *fast* procedure. The details are as follows.
One considers each row A_{rF} of the matrix $A_{\cdot F}$ (that consists of the columns of A with fractional variables in the current solution x^*) in turn; note that the sum over the fractionals in a row is either zero (there are no fractional variables because some variable has a value of one) or one:

$$A_{rF}x_F^* \in \{0,1\} \qquad \forall r = 1, \ldots, m.$$

In the latter case, this row induces a minimal clique $Q := \operatorname{supp} A_{rF}$ such that the clique inequality $\sum_{j \in Q} x_j \leq 1$ is tight for the current LP solution x^*. If one additional fractional variable can be lifted (sequentially) into Q, a violated clique inequality is detected. Lifting more fractional variables increases violation, and one can lift some additional variables with zero values in the end as well to extend the cut's support. Hence, the procedure has three steps: (i) Determining the "core" clique $Q = \operatorname{supp} A_{rF}$, (ii) sequential lifting of fractional variables into the core, and (iii) supplementary sequential liftings of zero variables.
Here are some *implementation* issues. While (i) is clear, one can come up with numerous strategies for the lifting steps (ii) and (iii). The method that we have implemented in `BC` opts for *speed*, because we do not expect to find many additional neighbors of (a part of) a matrix row clique, that is usually of substantial size — a philosophy that fits with the idea behind the row lifting method and that is supported by the probabilistic results of the previous Section 3.2 and by our computational experiments. For each of the steps (ii) and (iii), we set up a list of candidate variables that we arrange in a fixed lifting order, and this

candidate sequence is used for every row. In step (ii), the candidate set consists of some constant number k_F of the fractional variables with the largest x^*-values (we use $k_F = 20$ in our implementation) which are tried greedily in order of decreasing x^* values, and another constant number k_L of zero variables (we use $k_L = 10$) for step (iii) that we simply select at random.

Turning to the expected *running time*, we note that one sequential lifting of a variable x_j can be done by checking whether all variables in the current clique Q are neighbors of j. Using a forward star representation of $G(A_{\cdot F})$ this takes $O(|\gamma(j)|) \leq O(|F|)$ steps (where $|F|$ denotes the number of fractional variables in the current LP solution x^*). Doing this $k_F + k_L \leq O(|F|)$ times for m rows results in a total of $O(m|F|^2)$ operations for this routine — which is as fast as one could possibly hope.

The apparent disadvantage of the method is, however, that the cutting planes that one computes with such a technique do, by construction, resemble much subsets of rows with a small extension here and there. Generally speaking, the row lifting clique separation routine is a good starting method in the initial phase of a branch-and-cut run and yields reasonable results there; it is less useful in later stages of the computation.

Greedy Algorithm. The greedy method is certainly the most obvious and simple to implement separation strategy that one can come up with and our branch-and-cut algorithm `BC` also uses a clique detection method of this type.

Our routine is *implemented* in the following way. The greedy criterion is to go for a most violated clique inequality and it makes sense to do so by considering the fractional variables in order of decreasing x^* values (where x^* denotes the current fractional solution):

$$x^*_{\sigma_1} \geq x^*_{\sigma_2} \geq \cdots \geq x^*_{\sigma_{|F|}}, \qquad \text{where } \{\sigma_1, \ldots, \sigma_{|F|}\} = F.$$

Our greedy does now $|F|$ trials, one for each fractional "seed" variable x^*_j. In trial j, we initialize a clique $Q := \{\sigma_j\}$, that will (hopefully) be grown into the support of a violated clique inequality, and try to lift into Q all variables $x_{\sigma_{j+1}}, \ldots, x_{\sigma_{|F|}}$ of smaller x^* value in this order. The motivation behind this is to give variables with small x^* values also a "chance" to foster a violated clique. We do not restrict the number of fractional lifting candidates this time, because we expect for familiar reasons that the cliques that we can compute in this way will not be very large. Note that this is *different* from row lifting, where we start a priori with a "large" clique. This inspires the different lifting philosophy that we should "at least lift such small cliques reasonably", to put it nonchalantly. But how can we get a large extension when all our probabilistic analyses and computational experience indicates that we can not obtain it sequentially? Our idea is to use the large cliques that we already know and to do a *simultaneous* lifting with matrix rows, similar to the row lifting separation routine. Namely, we do the following: Given some fractional clique Q, we determine its common neighbors $\gamma(Q) := \cap_{j \in Q} \gamma(j)$ (note that it is not clever to compute this for a large clique, but no problem for a small one!) and then we look for the largest intersection $\gamma(Q) \cap \operatorname{supp} A_{r\cdot}$ of this set with the support of some row r; this set is added to Q.

Looking at *running times*, we have again that one sequential lifting of a fractional variable takes $O(|F|)$ operations. Lifting at most $|F|$ variables in the greedy clique growing phase results in $O(|F|^2)$ steps. The common neighbors of at most $|F|$ members of such a clique can be determined in $O(|F| \cdot m\rho \cdot n\rho)$ steps using the matrix A's row representation (not the complete intersection graph $G(A)$ which we did not set up!), and the maximum intersection of this set with a matrix row in $O(m\rho \cdot n\rho)$ steps, which is smaller. Assuming $O(|F| \cdot m\rho \cdot n\rho) \geq O(|F|^2)$

and doing all of these steps $|F|$ times, once for each of the seed variables, amounts to a total of $O(|F|^2 mn\rho^2)$ expected steps — which does not look very good. But our analysis is a very conservative estimate, because the expensive simultaneous lifting step is only called when a violated inequality is found, which (unfortunately!) is not the case for every starting candidate. The method can be tuned further using obvious break criteria based on the "tail sums"

$$x^*_{\sigma_j} + \cdots + x^*_{\sigma_{|F|}}, \qquad j = 1, \ldots, |F|,$$

that make the routine bail out whenever there is no more chance of finding a violated clique inequality. With this and other improvements of this type, one obtains a separation procedure that displays a reasonable behaviour in computational practice.

Recursive Smallest First. One of the most popular branch-and-bound approaches to the maximum weight clique problem is based on the recursion

$$\max_{Q \text{ clique in } G} x^*(Q) = \max \,\{x^*_j + \max_{Q \text{ clique in } G[\gamma(j)]} x^*(Q), \max_{Q \text{ clique in } G-j} x^*(Q)\}. \tag{3.5}$$

Here, G is some graph with node weights x^* and j one of its nodes. The first successful implementation of (3.5) is, as far as we know, due to Carraghan & Pardalos [1990] and since then this branching rule has turned into the progenitor of a large family of algorithms that differ by node selection and clever bounding criteria that try to reuse information that is computed once as often as possible.

Recursive smallest first (RSF) is one member of this class. It uses the special branching strategy to select in each step a node j that attains the minimum degree in the current graph. The idea is obviously that one of the two subproblems, namely, the one on the neighbors of j, i.e., on the graph $G[\gamma(j)]$, will hopefully be "small" and can be fathomed or solved fast. For fathoming, we can develop simple criteria in terms of sums of node weights of the current graph. And the subproblem can surely be solved fast if the number of nodes in the current graph is small, say, smaller than some constant k. When such circumstances supervene in every subdivision step, the RSF algorithm solves the maximum weight clique problem to *proven* optimality in time that is polynomial of order k. The worst case running time is exponential, however.

The observations of the previous paragraph suggest a simple way to combine, under favorable conditions, the advantage of RSF —a certificate of optimality— with a polynomially bounded running time. The idea is to turn the algorithm *dynamically* into a *heuristic* whenever we are about to walk into the complexity trap. Namely, we pursue the following strategy: We use in principle the generic RSF algorithm as described above, but whenever the current graph has more than k nodes and our fathoming criteria fail, we solve the associated subproblem heuristically. We call such a hybrid method with both exact branch-and-bound and heuristic components a *semiheuristic*. A scheme of this type has the advantages that it (i) is able to exploit some structural properties of the graph, namely, to reduce it systematically by cutting off low degree parts, (ii) it allows to control the tradeoff between exactness and speed by tuning the parameter k, and (iii) it sometimes even proves optimality of the result.

In our *implementation* of the RSF method, we set the parameter $k := 16$. When the current graph has less than this number of nodes, we determine the maximum clique by complete enumeration. The heuristic that we apply in subproblems that involve graphs with more than k nodes is the greedy procedure that we have described in the previous paragraph. To find a

node with smallest degree in each branching step, we store the nodes of the graph in a binary heap that is sorted with respect to node degrees. For familiar reasons, we do not expect the RSF algorithm to return a large clique. In this vein, RSF has the flavour of an improved greedy algorithm. Therefore we apply the same strategies to lift variables that have a value of zero in the current LP solution. Hoffman & Padberg [1993] describe a similar implementation of the RSF method.
The *running time* of RSF is $O(|F|^k)$: The time to compute lower bounds is $O(|F|^2)$, the greedy heuristic takes $O(|F|^3)$, enumeration takes $O(|F|^k)$, and the heap updates require $O(|F|^2 \log |F|)$ operations; finally, lifting results in $O(|F| m\rho n\rho)$ steps. This gives a total running time of $O(|F|^k + |F|mn\rho^2)$, which we assume to be of order $O(|F|^k)$.
To evaluate the quality of the RSF method we have implemented an *exact* branch-and-bound algorithm for the maximum clique problem as well. It turned out that, even without any tuning, RSF almost always produced a largest clique. Our computational experiments showed that the choice $k = 16$ was the optimal tradeoff between speed and quality. In fact, with $k = 16$, RSF produces always the largest clique on the airline test problems. For this reason, and because of the arguments mentioned in the introduction of this section, we do not use the exact branch-and-bound algorithm for clique separation, although this method is implemented.

3.3.3 Cycle Inequalities

Cycle inequalities are the second separation ingredient in our branch-and-cut algorithm. Like the clique inequalities, cuts of this type have small support, and they tend to have a nice numerical behaviour (only 0/1 coefficients in unlifted versions). An additional bonus is that they can be separated in polynomial time with the *GLS algorithm* of Grötschel, Lovász & Schrijver [1988]. We use this cycle detection algorithm in our branch-and-cut algorithm.
The GLS algorithm works on a bipartite *auxiliary graph* $B := B(G(A_{\cdot F}))$ that is constructed from the fractional intersection graph $G(A_{\cdot F}) = (V, E)$ as follows. The nodes of B are two copies V' and V'' of V. There is an edge $u'v''$ in B if and only if uv is an edge of G. To each such edge $u'v''$ we associate the weight $w_{uv} = 1 - x^*_u - x^*_v$, where x^* is the current fractional LP solution. Note that $0 \le w \le \mathbf{1}$.
The main steps of the procedure are as follows. One computes for each node $u' \in V'$ the shortest path P_u in B to its pendant u''. Each such path P_u, interpreted as a set of nodes, corresponds to an odd cycle C_u in G through u, possibly with node repetitions. The weight of C_u is

$$w(C_u) = w(P_u) = \textstyle\sum_{u'v'' \in \mathcal{P}_u} (1 - x^*_u - x^*_v) = |P_u| - 2x^*(P_u) = |C_u| - 2x^*(C_u).$$

Hence

$$w(C_u) < 1 \iff |C_u| - 1 < 2x^*(C_u) \iff (|C_u| - 1)/2 < x^*(C_u).$$

Thus, a path P_u in B with weight less than one corresponds to a violated odd cycle inequality. Conversely, a shortest odd cycle through a node u corresponds to the path P_u. This proves that the GLS algorithm solves the separation problem for cycle inequalities in polynomial time.
Our *implementation* of the GLS algorithm computes the shortest paths P_u using Dijkstra's algorithm. When the distance labels of the nodes are kept in a binary heap, this results in a *running time* of $O(|F|^2 \log |F| + |F||E|) = O(|F|^2 \log |F| + |F|^3)$; here, $|E|$ is the number of edges in the fractional intersection graph.

We use a number of implementation *tricks* to make this method work in practice. First note that it is not necessary to set up the auxiliary graph explicitly because adjacencies in B can be read off from the neighbor lists of $G(A_{\cdot F})$. The only place where the auxiliary graph shows up explicitly is the heap, where we have to store a distance label for each of the two copies of a node. Second, one can exploit the special form of the distance function $1 - x_u^* - x_j^*$ for computing expressions of the form

$$\texttt{dist}[v] = \min \ \{\texttt{dist}[v], \texttt{dist}[u] + (1 - x_u^* - x_v^*)\}$$

that come up in the relabeling step. The three arithmetic operations that are required to compute the term $\texttt{dist}[u] + (1 - x_u^* - x_v^*)$ for every neighbor v of u can be reduced to one by a precomputation of the term $\texttt{dist}[u] + 1 - x_u^*$. A minor speed up can be achieved by turning `double` x^* values into `int`egers (this saves about 10% of the running time). Third, Dijkstra's algorithm is a dynamic program. As we are interested in paths of length smaller than one only, we can fathom a node as soon as its distance label $\texttt{dist}[v]$ attains a value of one or more. Fourth, note that the generic GLS algorithm computes the shortest path P_u for every node $u \in G(A_{\cdot F})$. Once this path P_u is computed for a particular node u, this node can be deleted from the graph without loosing the exactness of the method. This is correct because a most violated cycle inequality passes through the node u or not. In the first case, the path P_u yields such a most violated inequality. In the second case, u is not relevant and can therefore be removed. Note that this elimination strategy has the additional advantage that it tends to produce violated cycle inequalities with disjoint support. It also paves the way for a fifth implementation trick that is based on a special ordering of the starting nodes for which we call Dijkstra's algorithm. We order the nodes with respect to decreasing x^* value

$$1 > x_{\sigma_1}^* \geq x_{\sigma_2}^* \geq \cdots \geq x_{\sigma_{|F|}}^* > 0.$$

If we denote by G_i the graph $G(A_{\cdot F})[\{\sigma_i, \ldots, \sigma_{|F|}\}]$ obtained from $G(A_{\cdot F})$ by deleting the nodes $\{\sigma_1, \ldots, \sigma_{i-1}\}$ (the starting nodes for the previous $i-1$ calls of Dijkstra's algorithm), all edge distances in G_i satisfy

$$w_{uv} = 1 - x_u^* - x_v^* \geq 1 - 2x_{\sigma_i}^*.$$

Any odd cycle C in G_i must contain at least three such edges and we have for its weight

$$w(C) = |C| - 2\textstyle\sum_{v \in C} x_v^* \geq |C| - 2|C|x_{\sigma_i}^*.$$

The last value in this sequence exceeds 1 if and only if

$$(1 - 1/|C|)/2 = 1/2(1 - 1/|C|) \geq x_{\sigma_i}^*.$$

This will be the case if $x_{\sigma_i}^* \leq 1/3$, i.e., we can stop computation as soon as the maximum x^*-value drops below 1/3. We compute with the GLS algorithm and these tricks paths P_u that correspond to odd closed walks in $G(A_{\cdot F})$ and extract from these a cycle without node repetitions.

Lifting odd cycle inequalities is a bit more complicated than lifting clique inequalities.

Let us first turn to *sequential lifting*. Note that it is not difficult to lift a constant number of variables into a cycle. We tried an implementation that does this for the first two variables,

such that, in each of the two steps, a maximum additional violation of x_j^* times the lifting coefficient is achieved. More fractional variables were lifted heuristically. This sequential method turned out to be slow, taking more time than the separation of the pure cycles. Moreover, it did not produce many nonzero lifting coefficients.
Therefore, a *simultaneous* lifting method was implemented. This method identifies for each edge ij in the cycle C a row $r = r_{ij}$ in the matrix $A_{\cdot F}$ such that $\{i, j\} \subseteq \operatorname{supp} A_{r_{ij}\cdot}$ (breaking ties arbitrarily). These rows are used to compute a Chvátal-Gomory cut that can be seen as a lifting of the cycle inequality that corresponds to C: We add up the rows $A_{r_{ij}\cdot}$, divide by two, and round the coefficients down. Exploiting sparsity, this method can be implemented in $O(|C|n\rho)$ time and exhibits a satisfactory computational behaviour.
One final issue on cycle separation is that it is possible that a violated inequality can result from a lifting of a pure cycle inequality which is not tight. We exploit this heuristically in our routine by increasing the "target length" of the paths in the GLS algorithm form one to some larger value in a dynamic and adaptive fashion depending on the number of cycle inequalities found in the previous call.

3.3.4 Aggregated Cycle Inequalities

The third class of inequalities that we try to separate are the *aggregated cycle inequalities* of Section 2.6. Recall that these inequalities stem from a set packing relaxation of the set covering problem.
Set packing inequalities tend to have the disadvantage of "smearing" the values of the LP solution over their support. This tends to increase the number of fractional variables with small values, which has all kinds of negative impacts on the solution process. To counter these effects, one would like to use cutting planes for the set covering polytope that gather some x^* value on their support and prevent the LP solution from dilution. Unfortunately, little algorithmically useful knowledge about such cutting planes is available. This was our motivation for the development of the aggregated cycle inequalities.
Aggregated cycle inequalities are separated with the *implementation* of the GLS algorithm that we have described in the previous subsection. The only difference is that the input graph is a (small) subgraph $\mathfrak{G}' = (\mathfrak{V}', \mathfrak{E}')$ of the aggregated fractional intersection graph $\mathfrak{G}(A_{\cdot F})$, which is of exponential size. The selection is guided by the desire to find a subgraph of "reasonable" polynomial size and with *many* edges $\mathfrak{uv}$ with *small* weights $1 - \pi_{\mathfrak{u}}(x^*) - \pi_{\mathfrak{v}}(x^*) = w_{\mathfrak{uv}}$. Such edges make it likely that cycles in $\mathfrak{G}$ give rise to violated aggregated cycle inequalities. We do not *lift* aggregated cycle inequalities.
Our heuristic to generate the subgraph $\mathfrak{G}'$ is the following. We generate two nodes I and $\overline{I}$ for each row A_{iF} of the matrix $A_{\cdot F}$. Namely, we subdivide the support of each row A_{iF} into two "equal sized halves"

$$\operatorname{supp} A_{iF} =: I \cup \overline{I}$$

with respect to a given fractional LP solution x^*, i.e., we split (in some way) such that $A_{iI}x_I^* \approx A_{i\overline{I}}x_{\overline{I}}^*$ and take $\mathfrak{V}$ as the set of these "halves":

$$\mathfrak{V} := \{I, \overline{I} \mid i = 1, \ldots, m\}.$$

Two such nodes $\mathfrak{u}$ and $\mathfrak{v}$ are in conflict if their union contains some row of the matrix $A_{\cdot F}$. These conflicts define the edges of the graph $\mathfrak{G}$.

3.4 Computational Results

We report in this section on *computational experiences* with our branch-and-cut code `BC`. We intend to investigate the following *questions*:

(i) *Performance.* What is the performance of `BC` on a standard test set of set partitioning problems from the literature: The `acs` test set of Hoffman & Padberg [1993].

(ii) *Branching versus Cutting.* Do cutting planes make a significant contribution to the solution of the problems in our test set?

(iii) *Aggregated Cycle Inequalities.* What is the effect of the aggregated cycle inequalities?

We have chosen the *airline crew scheduling problems* of Hoffman & Padberg [1993] as our *test set* (see this reference for a thorough discussion of these instances) because they are *publicly available*[2] and well known to the community. This makes it possible to compare our results with those of the literature, see, e.g., Hoffman & Padberg [1993], Atamturk, Nemhauser & Savelsbergh [1995], and Chu & Beasley [1995].
According to the *guidelines* of Crowder, Dembo & Mulvey [1979] and Jackson, Boggs, Nash & Powell [1991] for reporting about computational experiments, we state that all test runs were made on a Sun Ultra Sparc 2 Model 200E workstation with 512 MB of main memory, running SunOS 5.5, that our branch-and-cut code `BC` was written in ANSI `C` compiled with the Sun `cc` compiler and switches `-fast -xO5`, and that we have used the CPLEX [1997] Callable Library V5.0 as our LP solver.
The results of the following computational experiments are documented in *tables* that have the following format. Column 1 gives the name of the problem, columns 2-4 its size in terms of numbers of rows, columns, and nonzeros. These sizes are reduced by an initial preprocessing to the numbers that appear in the next three columns. Columns 8 and 9 report solution values. $\overline{z}$ is the value of the best solution that the algorithm has computed. The —s in the succeeding "Gap" column indicate that all of the problems have been solved to proven optimality. The following 5 columns give details about the branch-and-cut computation. We list, from left to right, the number of in- and out-pivots (Pvt) that are performed by the preprocessor, the number of cutting planes (Cut) added, the number of simplex iterations to solve the LPs (Itn), the number of LPs solved (LP), and the number of branch-and-bound nodes (B&B). Running times (as a percentage of the total time) for these routines are contained in columns 15–19: Problem reduction (PP), pivoting (Pvt), separation (Cut), LP-solution (LP), and primal heuristic (Heu). The last column gives the total running time in CPU seconds.
If not explicitly stated otherwise, all of our computations use the following *default parameter* settings and strategies for our code `BC`. We use a *best first search* on the branch-and-bound tree, the branching rule is *strong branching* (cf. Bixby, personal communication), i.e., we select a set of fractional candidate variables close to 0.5 (we try 10 candidates), fix them tentatively to 0 and 1, and perform a couple of dual simplex iterations with these fixings (we do 25 iterations). The variable that yields the largest increase in the smaller of the corresponding two lower bound values is the branching variable. Our primal heuristic is a *plunging* method that iteratively rounds fractional variables to the nearest integer and reoptimizes the linear program (we round to 1.0 all variables with values above 0.8 or, if no such variable exists, the one with the largest value, breaking ties arbitrarily). This heuristic is called once after the solution of the initial LP relaxation and once at each node of the searchtree. The default

[2] Anonymous `ftp` from `happy.gmu.edu:/pub/acs`

strategy for *separation* is to call the row clique lifting routine, the greedy clique detection, the RSF semiheuristic, and the GLS cycle algorithm. All of these procedures are called after each individual LP. Among the violated inequalities that we have found, we select the most violated ones up to a threshold that depends on the size of the LP and the number of cuts found. In each iteration, cuts with positive slack (of more than 0.1) are removed from the present LP. To avoid tailing off, we use an *early branching strategy* that stops the cutting plane phase if the duality gap does not decrease significantly from one iteration to the next (to 0.75 within any four successive iterations). Like the separation routines, the *preprocessor* is invoked after each solution of an LP. The LPs themselves are solved with the *dual simplex algorithm* and *steepest edge pricing*.

We have performed three computational *experiments* to answer the questions (i)–(iii). Our *Experiment 1* applies `BC` with the default strategy to the `acs` test set. In *Experiment 2*, we also separate aggregated cycle inequalities, all other parameter settings are identical. For *Experiment 3*, we turn off the cut generation module of `BC` completely, i.e., we apply branch-and-bound with preprocessing. Our *results* are summarized in Tables 3.4–3.6.

The statistics in these tables have quite some similarities and not only at first glance. We will, in fact, argue in our *analysis* that the outcome of the three experiments is essentially the same except for three "hard instances", namely, `nw04`, `aa04`, and `aa01`; the other problems fall into a number of categories of readily solvable instances. Our discussion will try to explain the differences in the computational behavior of the instances in terms of two measures of problem difficulty: Response to and/or size after *preprocessing* and the initial duality *gap*. Note that these are *a priori criteria*, i.e., they are available prior to the solution of the problem and can be used to predict expected solution efforts. We remark that we found these indicators satisfactory not only for the `acs` problems, but also for two sets of "Telebus clustering and chaining instances" (of different characteristics) from a vehicle scheduling application, confer Section 4.7 for a discussion of computational results for these instances.

A first similarity is that the initial *preprocessing* does not depend on the different parameter settings of the experiments, i.e., the reductions are always the same, see the "Preprocessed" columns 5–7 in Tables 3.4–3.6. We have already given more detailed statistics on the initial preprocessing step in Table 3.1. Taking another look at this data, we see that the first 27 instances up to `nw31` are reduced to very small problems with less than 30 rows; all of these simple instances can be solved in well under a second with all strategies.

Many of the remaining 28 problems are also fairly small and/or display minimal initial duality *gaps* already after the first invocation of the primal heuristic and without adding any cutting planes, see column "Gap" in Table 3.1. In fact, all but 9 of the instances 28–55 have a duality gap of 1.0% or less. One would hope that the solutions of the initial LP relaxations of these problems are close to integrality, i.e., they have only few fractional variables (one can not see this from the tables), and this is indeed the case: 11 of the instances 28–55 have integral LP solutions, the remaining fractional solutions are rounded to optimal ones at the root node in all but 9 cases by `BC`'s simple plunging heuristic (this data is also not in the tables). It is thus not surprising that those 19 of instances 28–55 with gap $\leq$ 1.0% can be solved in about 30 seconds with all strategies. Note that the solution statistics for the "hardest" of these 19 problems, instances `aa06`, `kl01`, `aa05`, and `aa03`, see the "Branch-and-Cut" columns in Table 3.4, fit with our difficulty indicators in terms of size and gap: The difficulty of the three `aa` instances is due to a large number of rows which leads to large bases and a relatively large number of pivots in the LP solution process, see column "Itn" in Table 3.4, while `kl01` displays the largest initial duality gap of 1.0%, see column "Gap" in Table 3.1.

The remaining nine instances nw06, nw18, nw03, nw17, kl02, us01, nw04, aa04, and aa01 are the ones that require the use of cutting planes, see column "Cut" in Table 3.4, several LPs, see column "LP", and some branch-and-bound, see column "B&B". The first five of these can again be solved fast in about 30 CPU seconds no matter if many or no cutting planes at all are used. This behavior is due to the fast decrease of the duality gap in the root section of the searchtree: In Experiment 1, e.g., the optimum is not found in the first rounding of the solution of the initial LP relaxation, but it comes up rapidly in trial 4, 2, 2, 4, and 2, respectively, (recall that the plunging heuristic is called once after the solution of the initial LP and once at each node, i.e., a 2 means that the optimum is found rounding the second LP at the root node, while 4 refers to the first LP at node number 3). Comparing these numbers with the size of the searchtree in column "B&B" reveals that the problems were solved immediately after this happened.
The analysis of the previous paragraph applies also to problem us01: The optimum is found at the root node with the second call to the heuristic, and then the problem is essentially finished in all three experiments. us01 is not a hard problem, but a large one, accounting for about 35% of both nonzeros and columns of the entire test set, and it just takes some 4 minutes to process all this data: The initial LP alone takes about 2 minutes.
We are thus indeed left with only three instances where the different use of cutting planes in our experiments can make a difference: nw04, aa04, and aa01. Note that these problems account for 363 out of a total of 444 branch-and-bound nodes in Experiment 1 (similar statements hold for the other experiments), for 1,131 out of 1,355 LPs, for 54,307 out of 64,361 dual simplex iterations, for 293,737 out of 312,587 in- and out-pivots, and for 619.99 out of 1006.89 CPU seconds, i.e., the performance of our algorithm BC on these four problems determines the outcome of our computational experiments completely. We would, however, like to stress that the hitherto treated "simple instances" *are formulations of real world problems* and that the ability to solve airline crew scheduling problems to proven optimality in such short times is one of the most remarkable successes in operations research. To put it in a pointed way: It is the computational well-behaviour that makes set partitioning models so useful. As even the hard problems in the acs test set can be solved in about 5 minutes with the default strategy, we answer question (i) about the performance of BC on the acs problems with a confirmation of Hoffman & Padberg [1993]'s conclusion that "it is possible to solve very large set-partitioning problems to proven optimality" and that "by using the [branch-and-cut] technology described above and solving *larger* set-partitioning problems exactly ... than is done today, the airline industry could see immediate and substantial dollar savings in their crew costs".
The three hard instances themselves fall again into two different categories, namely, instance nw04 on the one and aa04 and aa01 on the other hand. The difference between them is that nw04 has few rows and many columns, while the aa problems have the opposite property. We will give now a number of heuristic arguments that suggest that set partitioning problems with *many rows* tend to be more difficult for a branch-and-cut algorithm than problems with many columns. In fact, there are only two occasions where BC examines the complete set of columns: In the pricing step of the dual simplex algorithm and in the preprocessing. But these steps take linear or log-linear time only. The more expensive modules work on data structures whose size depends on the number m of rows: Refactorization works on a matrix of size $O(m^2)$ and has quadratic running time, separation works on a fractional intersection graph of the same size and has at least the same order of running time, and we expect the primal heuristic to perform $O(m)$ rounding steps requiring the same number of LP reoptimizations.

In light of these arguments, it is not surprising that `nw04` can be solved five to six times faster than `aa04` and `aa01`. In fact, the solution time for `nw04` is at most 85 seconds with any strategy such that one could even question the classification of `nw04` as a hard instance. Looking at the solution statistics in the "Branch-and-Cut" columns 11–15 of Tables 3.4–3.6, however, shows that `nw04` has the same complexity as the `aa` problems: Its solution requires a number of nodes, LPs, simplex iterations, and cutting planes that is in the same order of magnitude as the figures for the `aa` problems. The three hard problems have in common that the initial LP solution does not immediately reveal the optimum nor a proof of optimality, and that the solution takes some algorithmic effort. The smaller running time for `nw04` is solely due to the smaller amount of computation at the individual nodes of the searchtree.

Recalling how the "simple instances" `nw06`, `nw18`, `nw03`, `nw17`, `kl02`, and `us01` could be solved easily once the optimal solution was found, one might wonder if the hard problems are difficult because `BC`'s simple plunging heuristic is unable to find good solutions? To answer this question, we have run our code with the optimal solution as an additional input. It turns out that primal knowledge does not make the problems much easier: For the default strategy, e.g., we still needed 75/121/101 nodes, 247/392/313 LPs, 3,569/18,780/21,936 dual simplex iterations, 709/1,333/1,067 cuts, and 38.110/181.900/289.320 CPU seconds to solve `nw04/aa04/aa01`, respectively (the decrease in the running times of `nw04` and `aa04` is mainly due to a more effective reduced cost fixing, while `aa01` takes, in fact, even longer to solve!).

Closing the gap from the dual side thus seems to be what makes instances `nw04`, `aa04`, and `aa01` hard. Here is where cutting planes come into play and where the different separation strategies in Experiments 1–3 make a difference. We first turn to question (ii) about the significance of cutting planes for the solution process. Comparing the results of Experiment 1 in Table 3.4 with the default strategy to the outcome of the branch-and-bound Experiment 3 in Table 3.6 gives the disappointing result that the negligence of the cuts is not punished with an increase in running time. There is only a redistribution away from cut generation and the LP to the other modules of `BC`. Hence, our timing statistics give no arguments in favor of cutting planes. The "Branch-and-Cut" parts of Tables 3.4 and 3.6, however, provide some justification for the use of cutting planes: Cuts reduce the size of the searchtree from 203/441/131 nodes in Experiment 1 to only 85/181/97 in Experiment 3, and similar albeit smaller reductions apply to the number of LPs, dual simplex iterations, and in- and out-pivots. These findings do certainly not speak against the use of cutting planes in computational set partitioning.

Experiment 2 was designed to investigate another step in this direction: Do the aggregated cycle inequalities of Section 2.6 yield a computational advantage? The answer to question (iii) is similar to our findings for question (ii). Comparing the results of Experiment 1 in Table 3.4 with the statistics on Experiment 2 in Table 3.5 displays an increase in running time by a factor of three when aggregated cycle inequalities are used. This outcome is, however, solely due to the experimental status of our aggregated cycle separation routine: An examination of the "Cut" column in the "Timings" section of Table 3.5 shows that about 70% of the running time is spent in this module. The "Branch-and-Cut" statistics show some encouraging effects of aggregated cycle separation: The searchtrees are reduced from 85/181/97 nodes to 49/133/111 nodes, and similar savings can be observed for the number of LPs and dual simplex iterations. We feel that these results indicate some potential for aggregated cycle inequalities and strongly believe that cuts of such aggregation types are valuable for solving hard integer programming problems (not only set partitioning problems). The separation module itself leaves ample room for improvement and this is one of the issues of future research.

	Original Problem			Preprocessed			Solutions		Branch-and-Cut					Timings/%					Total
Name	Rows	Cols	NNEs	Rows	Cols	NNEs	$\overline{z}$	Gap/%	Pvt	Cut	Itn	LP	B&B	PP	Pvt	Cut	LP	Heu	Time/Sec
nw41	17	197	740	17	177	672	11307	—	34	0	14	3	1	50	0	0	50	0	0.02
nw32	19	294	1357	17	241	1116	14877	—	74	0	23	6	3	0	33	0	0	0	0.03
nw40	19	404	2069	19	336	1715	10809	—	71	11	19	4	1	0	33	0	0	0	0.03
nw08	24	434	2332	19	349	1844	35894	—	0	0	25	1	1	50	0	0	50	0	0.02
nw15	31	467	2830	29	465	2820	67743	—	0	0	17	1	1	50	0	0	50	0	0.02
nw21	25	577	3591	25	426	2591	7408	—	76	6	16	3	1	60	0	20	20	0	0.05
nw22	23	619	3399	23	531	2958	6984	—	46	0	22	3	1	50	0	0	25	25	0.04
nw12	27	626	3380	25	451	1653	14118	—	0	0	41	1	1	50	0	0	50	0	0.02
nw39	25	677	4494	25	567	3725	10080	—	51	0	15	5	3	20	20	0	20	20	0.05
nw20	22	685	3722	22	566	3112	16812	—	66	0	25	3	1	50	25	0	25	0	0.04
nw23	19	711	3350	12	430	1937	12534	—	37	0	25	3	1	50	17	0	0	0	0.06
nw37	19	770	3778	19	639	3143	10068	—	39	2	19	3	1	17	17	0	0	0	0.06
nw26	23	771	4215	21	514	2722	6796	—	43	3	24	3	1	20	20	0	20	0	0.05
nw10	24	853	4336	20	643	3153	68271	—	0	0	28	1	1	0	0	0	0	0	0.02
nw34	20	899	5045	20	750	4224	10488	—	40	2	24	3	1	25	0	0	25	0	0.04
nw28	18	1210	8553	18	599	3898	8298	—	31	0	17	3	1	33	0	0	67	0	0.03
nw25	20	1217	7341	20	844	5090	5960	—	60	13	30	4	1	33	0	0	33	11	0.09
nw38	23	1220	9071	20	881	6400	5558	—	40	0	26	3	1	33	0	11	33	0	0.09
nw27	22	1355	9395	22	926	6266	9933	—	44	0	16	3	1	17	17	0	17	0	0.06
nw24	19	1366	8617	19	926	5844	6314	—	60	2	21	4	1	43	43	0	29	14	0.07
nn01	18	1072	4859	17	983	4412	8904	—	68	3	25	3	1	38	12	0	12	12	0.08
nn02	23	1079	6533	19	820	4938	7656	—	38	0	25	3	1	25	0	0	12	0	0.08
nw35	23	1709	10494	23	1403	8718	7216	—	47	2	19	3	1	20	10	0	30	10	0.10
nw36	20	1783	13160	20	1408	10176	7314	—	212	26	103	17	5	16	8	8	22	10	0.49
nw29	18	2540	14193	18	2034	11345	4274	—	178	14	80	11	3	16	14	8	24	5	0.37
nw30	26	2653	20436	26	1884	14603	3942	—	130	7	30	7	3	7	5	4	18	2	0.56
nw31	26	2662	19977	26	1823	13846	8038	—	52	2	25	3	1	29	0	0	29	0	0.14
nw19	40	2879	25193	32	2134	14968	10898	—	0	0	48	1	1	31	0	0	38	0	0.13
nw33	23	3068	21704	23	2415	17081	6678	—	60	3	23	4	1	29	0	0	24	6	0.17
nw09	40	3103	20111	33	2296	14065	67760	—	0	0	54	1	1	36	0	0	36	0	0.14
nw07	36	5172	41187	33	3104	23808	5476	—	0	0	42	1	1	35	0	0	40	0	0.20
aa02	531	5198	36359	361	3928	21822	30494	—	0	0	805	1	1	8	0	0	88	0	1.99
nw06	50	6774	61555	37	5936	44730	7810	—	259	5	109	7	3	22	8	2	37	3	1.02
aa06	646	7292	51728	507	6064	36671	27040	—	3902	64	1580	11	3	11	8	2	68	5	8.76
kl01	55	7479	56242	47	5957	37709	1086	—	694	32	176	11	3	18	12	2	27	12	1.79
aa05	801	8308	65953	533	6371	37503	53839	—	6997	129	2141	23	7	9	8	3	59	13	13.62
aa03	825	8627	70806	558	6970	43800	49649	—	2741	21	1916	4	1	7	6	0	79	5	12.47
nw11	39	8820	57250	28	5946	34614	116256	—	59	2	50	3	1	34	9	0	25	4	0.56
nw18	124	10757	91028	81	7934	51304	340160	—	490	12	208	3	1	20	11	4	41	8	2.19
us02	100	13635	192716	44	8946	66594	5965	—	0	0	118	1	1	27	0	0	56	0	1.31
nw13	51	16043	104541	48	10901	62356	50146	—	98	0	103	5	3	27	8	0	33	5	1.24
us04	163	28016	297538	98	4285	33597	17854	—	198	0	139	3	1	48	3	0	18	4	1.63
nw03	59	43749	363939	53	38956	318977	24492	—	157	5	130	4	1	18	5	0	48	9	7.34
nw01	135	51975	410894	135	50069	396507	114852	—	0	0	131	1	1	23	0	0	46	0	2.70
us03	77	85552	1211929	50	23207	238665	5338	—	0	0	75	1	1	47	0	0	29	0	5.50
nw02	145	87879	721736	145	85258	701959	102903	—	0	0	147	1	1	21	0	0	50	0	5.61
nw17	61	118607	1010039	54	78173	647646	11115	—	596	21	206	9	3	15	8	1	45	4	29.03
nw14	73	123409	904910	68	95169	651188	61844	—	0	0	223	1	1	19	0	0	71	0	19.23
nw16	139	148633	1501820	0	1	0	1181590	—	0	0	0	0	0	81	0	0	0	0	7.11
nw05	71	288507	2063641	58	202482	1404493	132878	—	0	0	172	1	1	24	0	0	62	0	30.72
kl02	71	36699	212536	69	16542	95192	219	—	1031	35	289	8	1	19	9	1	44	11	6.05
us01	145	1053137	13636541	86	351018	3161451	1003600	—	1068	31	415	12	3	21	6	0	47	6	228.68
nw04	36	87482	636666	35	46189	331225	16862	—	6209	775	4647	282	85	8	5	5	25	15	57.04
aa04	426	7195	52121	343	6200	38201	26374	—	168936	1677	30049	555	181	8	8	12	29	15	319.19
aa01	823	8904	72965	616	7625	48627	56137	—	117555	944	19611	294	97	7	9	10	36	14	238.76
55	6378	2305749	24174915	4736	1105692	8707674	3916911	—	312587	3849	64361	1355	444	13	7	7	39	11	1006.89

Table 3.4: Solving Set Partitioning Problems by Branch-and-Cut: Default Strategy.

	Original Problem			Preprocessed			Solutions		Branch-and-Cut					Timings/%					Total
Name	Rows	Cols	NNEs	Rows	Cols	NNEs	$\overline{z}$	Gap/%	Pvt	Cut	Itn	LP	B&B	PP	Pvt	Cut	LP	Heu	Time/Sec
nw41	17	197	740	17	177	672	11307	—	34	0	14	3	1	0	0	0	0	50	0.02
nw32	19	294	1357	17	241	1116	14877	—	74	0	23	6	3	0	20	40	0	20	0.05
nw40	19	404	2069	19	336	1715	10809	—	57	6	18	3	1	25	0	25	0	0	0.04
nw08	24	434	2332	19	349	1844	35894	—	0	0	25	1	1	100	0	0	0	0	0.01
nw15	31	467	2830	29	465	2820	67743	—	0	0	17	1	1	100	0	0	0	0	0.01
nw21	25	577	3591	25	426	2591	7408	—	76	6	16	3	1	40	0	0	20	0	0.05
nw22	23	619	3399	23	531	2958	6984	—	46	0	22	3	1	25	0	0	25	0	0.04
nw12	27	626	3380	25	451	1653	14118	—	0	0	41	1	1	50	0	0	50	0	0.02
nw39	25	677	4494	25	567	3725	10080	—	51	0	15	5	3	40	0	20	20	0	0.05
nw20	22	685	3722	22	566	3112	16812	—	66	0	25	3	1	33	17	0	17	17	0.06
nw23	19	711	3350	12	430	1937	12534	—	37	0	25	3	1	50	33	0	0	0	0.06
nw37	19	770	3778	19	639	3143	10068	—	39	2	19	3	1	0	0	0	40	0	0.05
nw26	23	771	4215	21	514	2722	6796	—	43	3	24	3	1	40	0	20	20	0	0.05
nw10	24	853	4336	20	643	3153	68271	—	0	0	28	1	1	0	0	0	50	0	0.02
nw34	20	899	5045	20	750	4224	10488	—	40	2	24	3	1	20	40	0	20	20	0.05
nw28	18	1210	8553	18	599	3898	8298	—	31	0	17	3	1	17	0	17	17	0	0.06
nw25	20	1217	7341	20	844	5090	5960	—	40	7	26	3	1	29	14	14	14	14	0.07
nw38	23	1220	9071	20	881	6400	5558	—	40	0	26	3	1	25	12	12	25	12	0.08
nw27	22	1355	9395	22	926	6266	9933	—	44	0	16	3	1	29	14	0	29	14	0.07
nw24	19	1366	8617	19	926	5844	6314	—	60	2	22	4	1	50	25	0	25	12	0.08
nn01	18	1072	4859	17	983	4412	8904	—	68	3	25	3	1	22	22	11	11	22	0.09
nn02	23	1079	6533	19	820	4938	7656	—	38	0	25	3	1	38	12	0	38	12	0.08
nw35	23	1709	10494	23	1403	8718	7216	—	47	2	19	3	1	40	20	0	20	10	0.10
nw36	20	1783	13160	20	1408	10176	7314	—	179	17	93	15	5	11	8	9	23	8	0.53
nw29	18	2540	14193	18	2034	11345	4274	—	242	24	70	15	5	12	12	15	21	4	0.52
nw30	26	2653	20436	26	1884	14603	3942	—	104	7	29	4	1	17	14	14	17	3	0.29
nw31	26	2662	19977	26	1823	13846	8038	—	52	2	25	3	1	29	0	7	29	0	0.14
nw19	40	2879	25193	32	2134	14968	10898	—	0	0	48	1	1	31	0	0	38	0	0.13
nw33	23	3068	21704	23	2415	17081	6678	—	60	3	23	4	1	28	17	0	22	6	0.18
nw09	40	3103	20111	33	2296	14065	67760	—	0	0	54	1	1	36	0	0	36	0	0.14
nw07	36	5172	41187	33	3104	23808	5476	—	0	0	42	1	1	38	0	0	38	0	0.21
aa02	531	5198	36359	361	3928	21822	30494	—	0	0	805	1	1	9	0	0	87	0	1.97
nw06	50	6774	61555	37	5936	44730	7810	—	222	7	101	7	3	21	6	5	34	2	1.08
aa06	646	7292	51728	507	6064	36671	27040	—	3675	52	1569	8	1	9	7	21	57	3	11.13
kl01	55	7479	56242	47	5957	37709	1086	—	530	44	166	10	3	17	11	13	24	10	2.05
aa05	801	8308	65953	533	6371	37503	53839	—	6030	115	2045	23	7	7	6	20	49	11	15.94
aa03	825	8627	70806	558	6970	43800	49649	—	2740	20	1902	4	1	7	6	3	77	5	12.77
nw11	39	8820	57250	28	5946	34614	116256	—	59	2	50	3	1	30	9	0	25	5	0.57
nw18	124	10757	91028	81	7934	51304	340160	—	490	12	208	3	1	19	10	7	40	8	2.26
us02	100	13635	192716	44	8946	66594	5965	—	0	0	118	1	1	28	0	0	55	0	1.28
nw13	51	16043	104541	48	10901	62356	50146	—	98	0	103	5	3	27	6	0	34	5	1.24
us04	163	28016	297538	98	4285	33597	17854	—	198	0	139	3	1	48	3	1	18	4	1.61
nw03	59	43749	363939	53	38956	318977	24492	—	157	5	133	4	1	18	6	0	48	9	7.19
nw01	135	51975	410894	135	50069	396507	114852	—	0	0	131	1	1	23	0	0	46	0	2.60
us03	77	85552	1211929	50	23207	238665	5338	—	0	0	75	1	1	47	0	0	30	0	5.38
nw02	145	87879	721736	145	85258	701959	102903	—	0	0	147	1	1	22	0	0	50	0	5.15
nw17	61	118607	1010039	54	78173	647646	11115	—	678	26	234	9	3	14	8	4	41	4	30.37
nw14	73	123409	904910	68	95169	651188	61844	—	0	0	223	1	1	17	0	0	74	0	17.90
nw16	139	148633	1501820	0	1	0	1181590	—	0	0	0	0	0	81	0	0	0	0	7.00
nw05	71	288507	2063641	58	202482	1404493	132878	—	0	0	172	1	1	25	0	0	62	0	29.87
kl02	71	36699	212536	69	16542	95192	219	—	1402	67	373	14	3	15	9	8	36	9	8.11
us01	145	1053137	13636541	86	351018	3161451	1003600	—	1523	71	454	18	5	19	6	3	43	6	247.38
nw04	36	87482	636666	35	46189	331225	16862	—	3572	543	3014	164	49	4	3	30	17	10	84.67
aa04	426	7195	52121	343	6200	38201	26374	—	145213	1467	22638	457	133	2	2	81	6	3	1170.24
aa01	823	8904	72965	616	7625	48627	56137	—	150784	1027	25132	342	111	2	3	70	12	5	1108.96
55	6378	2305749	24174915	4736	1105692	8707674	3916911	—	318939	3544	60828	1192	376	5	3	63	15	4	2780.07

Table 3.5: Solving Set Partitioning Problems by Branch-and-Cut: Separating Aggregated Cycle Inequalities.

	Original Problem			Preprocessed			Solutions		Branch-and-Cut					Timings/%					Total
Name	Rows	Cols	NNEs	Rows	Cols	NNEs	$\overline{z}$	Gap/%	Pvt	Cut	Itn	LP	B&B	PP	Pvt	Cut	LP	Heu	Time/Sec
nw41	17	197	740	17	177	672	11307	—	34	0	14	3	1	33	0	0	0	0	0.03
nw32	19	294	1357	17	241	1116	14877	—	74	0	23	6	3	20	0	0	0	40	0.05
nw40	19	404	2069	19	336	1715	10809	—	71	0	19	5	3	0	0	0	0	0	0.04
nw08	24	434	2332	19	349	1844	35894	—	0	0	25	1	1	50	0	0	50	0	0.02
nw15	31	467	2830	29	465	2820	67743	—	0	0	17	1	1	50	0	0	50	0	0.02
nw21	25	577	3591	25	426	2591	7408	—	76	0	21	5	3	17	0	0	33	17	0.06
nw22	23	619	3399	23	531	2958	6984	—	46	0	22	3	1	25	0	0	50	0	0.04
nw12	27	626	3380	25	451	1653	14118	—	0	0	41	1	1	50	0	0	50	0	0.02
nw39	25	677	4494	25	567	3725	10080	—	51	0	15	5	3	25	0	0	50	0	0.04
nw20	22	685	3722	22	566	3112	16812	—	66	0	25	3	1	17	0	0	17	17	0.06
nw23	19	711	3350	12	430	1937	12534	—	37	0	25	3	1	17	33	0	17	0	0.06
nw37	19	770	3778	19	639	3143	10068	—	39	0	19	5	3	50	25	0	25	0	0.04
nw26	23	771	4215	21	514	2722	6796	—	43	0	23	5	3	20	20	0	20	0	0.05
nw10	24	853	4336	20	643	3153	68271	—	0	0	28	1	1	33	0	0	0	0	0.03
nw34	20	899	5045	20	750	4224	10488	—	40	0	26	5	3	17	33	0	17	17	0.06
nw28	18	1210	8553	18	599	3898	8298	—	31	0	17	3	1	20	0	0	20	0	0.05
nw25	20	1217	7341	20	844	5090	5960	—	80	0	32	5	3	20	10	0	20	20	0.10
nw38	23	1220	9071	20	881	6400	5558	—	40	0	26	3	1	25	12	0	12	12	0.08
nw27	22	1355	9395	22	926	6266	9933	—	44	0	16	3	1	29	0	0	14	14	0.07
nw24	19	1366	8617	19	926	5844	6314	—	76	0	23	6	3	33	33	0	22	11	0.09
nn01	18	1072	4859	17	983	4412	8904	—	101	0	29	5	3	25	17	0	8	17	0.12
nn02	23	1079	6533	19	820	4938	7656	—	38	0	25	3	1	43	0	0	43	14	0.07
nw35	23	1709	10494	23	1403	8718	7216	—	47	0	18	5	3	40	20	0	20	10	0.10
nw36	20	1783	13160	20	1408	10176	7314	—	126	0	91	12	7	10	10	0	22	8	0.40
nw29	18	2540	14193	18	2034	11345	4274	—	235	0	96	18	11	14	11	0	25	11	0.44
nw30	26	2653	20436	26	1884	14603	3942	—	182	0	38	8	5	9	9	0	15	7	0.67
nw31	26	2662	19977	26	1823	13846	8038	—	52	0	25	3	1	21	7	0	29	7	0.14
nw19	40	2879	25193	32	2134	14968	10898	—	0	0	48	1	1	38	0	0	31	0	0.13
nw33	23	3068	21704	23	2415	17081	6678	—	60	0	23	6	3	21	0	0	26	5	0.19
nw09	40	3103	20111	33	2296	14065	67760	—	0	0	54	1	1	36	0	0	36	0	0.14
nw07	36	5172	41187	33	3104	23808	5476	—	0	0	42	1	1	35	0	0	30	0	0.20
aa02	531	5198	36359	361	3928	21822	30494	—	0	0	805	1	1	9	0	0	87	0	1.96
nw06	50	6774	61555	37	5936	44730	7810	—	560	0	120	14	9	19	9	0	27	5	1.34
aa06	646	7292	51728	507	6064	36671	27040	—	4378	0	1621	15	9	11	10	0	63	5	9.83
kl01	55	7479	56242	47	5957	37709	1086	—	848	0	210	22	13	18	11	0	22	11	2.18
aa05	801	8308	65953	533	6371	37503	53839	—	6761	0	2271	27	15	8	7	0	52	16	15.79
aa03	825	8627	70806	558	6970	43800	49649	—	4709	0	2049	9	5	7	9	0	68	5	14.71
nw11	39	8820	57250	28	5946	34614	116256	—	59	0	49	4	1	32	8	0	24	5	0.59
nw18	124	10757	91028	81	7934	51304	340160	—	968	0	227	6	3	14	9	0	21	7	4.95
us02	100	13635	192716	44	8946	66594	5965	—	0	0	118	1	1	28	0	0	56	0	1.33
nw13	51	16043	104541	48	10901	62356	50146	—	98	0	103	5	3	27	7	0	34	5	1.23
us04	163	28016	297538	98	4285	33597	17854	—	198	0	139	3	1	49	2	0	17	4	1.60
nw03	59	43749	363939	53	38956	318977	24492	—	157	0	154	6	3	18	5	0	48	9	7.41
nw01	135	51975	410894	135	50069	396507	114852	—	0	0	131	1	1	23	0	0	46	0	2.70
us03	77	85552	1211929	50	23207	238665	5338	—	0	0	75	1	1	48	0	0	29	0	5.42
nw02	145	87879	721736	145	85258	701959	102903	—	0	0	147	1	1	22	0	0	49	0	5.12
nw17	61	118607	1010039	54	78173	647646	11115	—	756	0	214	11	5	16	8	0	46	5	25.76
nw14	73	123409	904910	68	95169	651188	61844	—	0	0	223	1	1	17	0	0	74	0	18.41
nw16	139	148633	1501820	0	1	0	1181590	—	0	0	0	0	0	81	0	0	0	0	7.13
nw05	71	288507	2063641	58	202482	1404493	132878	—	0	0	172	1	1	24	0	0	63	0	30.89
kl02	71	36699	212536	69	16542	95192	219	—	3999	0	773	137	79	13	7	0	28	7	11.87
us01	145	1053137	13636541	86	351018	3161451	1003600	—	1208	0	382	13	7	21	6	0	46	6	230.56
nw04	36	87482	636666	35	46189	331225	16862	—	3909	0	1672	310	203	9	4	0	20	22	32.92
aa04	426	7195	52121	343	6200	38201	26374	—	228717	0	31882	714	441	11	9	0	21	16	341.99
aa01	823	8904	72965	616	7625	48627	56137	—	85521	0	15780	208	131	7	10	0	33	16	213.51
55	6378	2305749	24174915	4736	1105692	8707674	3916911	—	344535	0	60263	1646	1010	14	7	0	35	12	992.81

Table 3.6: Solving Set Partitioning Problems by Branch-and-Bound.

Bibliography of Part 2

Ahuja, Magnanti & Orlin (1989). Network Flows. In Nemhauser, Rinnooy Kan & Todd [1989], chapter IV, pp. 211–369.

Andersen & Andersen (1995). Presolving in Linear Programming. *Math. Prog. 71, 221–245.*

Applegate, Bixby, Chvátal & Cook (1994). Large Scale Combinatorial Optimization. Talk at the 15th Int. Symp. on Math. Prog., Ann Arbor, MI.

Atamturk, Nemhauser & Savelsbergh (1995). A Combined Lagrangian, Linear Programming and Implication Heuristic for Large-Scale Set Partitioning Problems. Tech. Rep. LEC - 95-07, Georgia Inst. of Tech.

Balas, Ceria & Cornuéjols (1993). A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs. *Math. Prog. 58, 295–324.*

Balas & Padberg (1976). Set Partitioning: A Survey. *SIAM Rev. 18, 710–760.*

Balinski (1965). Integer Programming: Methods, Uses, Computation. *Mgmt. Sci. 12(3), 253–313.*

Beasley (1987). An Algorithm for Set Covering Problem. *Europ. J. on OR 31, 85–93.*

Bellman & Hall (Eds.) (1960). *Combinatorial Analysis*, Proc. of Symposia in Applied Mathematics, Providence, RI.

Bixby (1994). Lecture Notes for CAAM 571: Combinatorial Optimization and Integer Programming, Spring 1994, Rice Univ., Houston, TX.

Brearley, Mitra & Williams (1975). Analysis of Mathemtical Programming Problems Prior to Applying the Simplex Method. *Math. Prog. 8(1), 54–83.*

Caprara & Fischetti (1997). Branch-and-Cut Algorithms. In Dell'Amico, Maffioli & Martello [1997], chapter 4, pp. 45–63.

Carraghan & Pardalos (1990). An exact Algorithm for the Maximum Weight Clique Problem. *Operations Research Letters 9, 375–382.*

Chu & Beasley (1995). Constraint Handling in Genetic Algorithms: The Set Partitioning Problem. Working paper[1], Imperial College, London, UK.

CPLEX (1995). *Using the CPLEX Callable Library*[2]. Suite 279, 930 Tahoe Blvd., Bldg 802, Incline Village, NV 89451, USA: CPLEX Optimization, Inc.

CPLEX (1997). *Using the CPLEX Callable Library.* 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA: ILOG CPLEX Division. Information available at URL http://www.cplex.com.

Crowder, Dembo & Mulvey (1979). On Reporting Computational Experiments with Mathematical Software. *ACM Transactions on Math. Software 5(2), 193–203.*

[1]Avail. at URL `http://mscmga.ms.ic.ac.uk/jeb/jeb.html`

[2]Inf. avail. at URL `http://www.cplex.com`

Crowder, Johnson & Padberg (1983). Solving Large-Scale Zero-One Linear Programming Problems. *Op. Res. 31, 803–834.*

Dell'Amico, Maffioli & Martello (Eds.) (1997). *Annotated Bibliographies in Combinatorial Optimization.* John Wiley & Sons Ltd, Chichester.

Emden-Weinert, Hougardy, Kreuter, Proemel & Steger (1996). *Einführung in Graphen und Algorithmen*[3].

Garey & Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman and Company.

Garfinkel & Nemhauser (1969). The Set Partitioning Problem: Set Covering with Equality Constraints. *Op. Res. 17(5), 848–856.*

Gomory (1960). Solving Linear Programming Problems in Integers. In Bellman & Hall [1960].

Grötschel, Lovász & Schrijver (1988). *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, Berlin.

Hoffman & Padberg (1991). Improving LP-Representations of Zero-One Linear Programs for Branch-and-Cut. *ORSA J. on Comp. 3(2), 121–134.*

Hoffman & Padberg (1993). Solving Airline Crew-Scheduling Problems by Branch-And-Cut. *Mgmt. Sci. 39, 657–682.*

Jackson, Boggs, Nash & Powell (1991). Guidelines for Reporting Results of Computational Experiments. Report of the Ad Hoc Committee. *Math. Prog. 49, 413–425.*

Martin & Weismantel (1997). The Intersection of Knapsack Polyhedra and Extensions. Preprint SC 97-61[4], Konrad-Zuse-Zentrum Berlin.

Nemhauser, Rinnooy Kan & Todd (Eds.) (1989). *Optimization*, volume 1 of *Handbooks in OR and Mgmt. Sci.* Elsevier Sci. B.V., Amsterdam.

Nemhauser & Wolsey (1988). *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc.

Nobili & Sassano (1992). A Separation Routine for the Set Covering Polytope. pp. 201 – 219.

Padberg & Rinaldi (1991). A Branch and Cut Algorithm for the Resolution of Large–Scale Symmetric Traveling Salesman Problems. *SIAM Review 33, 60–100.*

Suhl & Szymanski (1994). Supernode Processing of Mixed-Integer Models. *Computational Optimization and Applications 3, 317–331.*

Thienel (1995). *ABACUS A Branch-And-CUt System.* PhD thesis, Univ. zu Köln.

Wedelin (1995). An algorithm for large scale 0-1 integer programming with application to airline crew scheduling. *Ann. Oper. Res. 57, 283–301.*

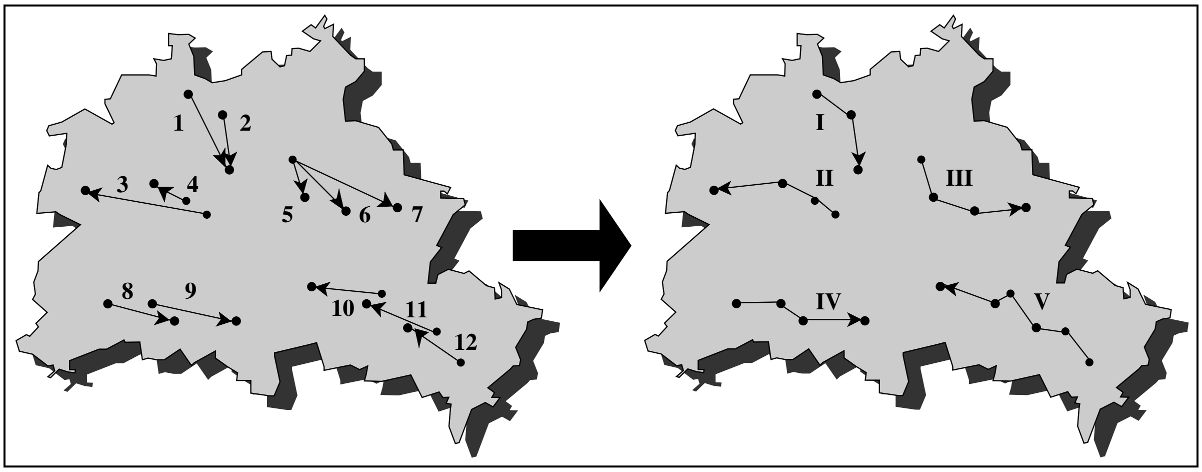
[3]Avail. at URL `http://www.informatik.hu-berlin.de/Institut/struktur/algorithmen/ga/`

[4]Avail. at URL `http://www.zib.de/ZIBbib/Publications/`

Index of Part 2

Part III

Application Aspects



Chapter 4

Vehicle Scheduling at Telebus

Summary. This chapter is about *set partitioning* methods for *vehicle scheduling in dial-a-ride systems.* We consider the optimization of Berlin's *Telebus* for handicapped people that services 1,500 requests per day with a fleet of 100 mini buses. Our scheduling system is in use since June 3, 1995 and resulted in *improved service* and significant *cost savings.*

Acknowledgement. The results of this chapter are joint work with Fridolin Klostermeier[1] and Christian Küttner[1].

Cooperation. The Telebus project was done at the Konrad-Zuse-Zentrum Berlin (ZIB)[2] in cooperation with the Berliner Zentralausschuß für Soziale Aufgaben e.V. (BZA)[3], the Berliner Senatsverwaltung für Soziales (SenSoz)[4], and the Intranetz Gesellschaft für Informationslogistik mbH[1].

Support. The Telebus project was supported from July 1, 1993 to December 31, 1996 by the Berliner Senatsverwaltung für Wissenschaft, Forschung und Kultur (SenWiFoKult)[5].

4.1 Introduction

This chapter is about *set partitioning* methods for *vehicle scheduling in dial-a-ride systems* and their application at Berlin's *Telebus* for handicapped people.

---

[1]Intranetz GmbH, Bergstr. 22, 10115 Berlin, Germany, URL `http://www.intranetz.de`
[2]Konrad-Zuse-Zentrum Berlin, Takustr. 7, 14195 Berlin, Germany, URL `http://www.zib.de`
[3]Berliner Zentralausschuß für Soziale Aufgaben e.V., Esplanade 17, 13187 Berlin, Germany
[4]Senatsverwaltung für Soziales, An der Urania 12, Abt. IX B, 10787 Berlin, Germany
[5]Senatsverwaltung für Wissenschaft, Forschung & Kultur, Abt. III B, Brunnenstr. 188-190, 10119 Berlin, Germany

Dial-a-ride systems give rise to challenging optimization problems that involve strategic as well as operational planning, uncertainty and on-line aspects, decisions in space and time, complicated feasibility constraints and multiple objectives, “soft” data, “fuzzy” rules, and applications of large scale. This colorful manifold of topics is matched by the wide variety of methods that have been developed to solve the planning questions of this area: Dynamic programming algorithms, network models, set partitioning/set covering and other integer programming approaches, and all kinds of combinatorial heuristics, single and multi-phased, cluster-first schedule-second and vice versa, etc. For surveys on *dial-a-ride problems* and solution methods we refer the reader to Desrosiers, Dumas, Solomon & Soumis [1995] and the thesis of Sol [1994, Chapter 1], see also Hamer [1997] for a recent description of a modern large-scale dial-a-ride system for handicapped people in Toronto and the thesis of Tesch [1994] (German) for the example of the German city of Passau.
We discuss in this chapter the application of some of these optimization methods to vehicle scheduling in a specific dial-a-ride system: Berlin’s *Telebus* for handicapped people. Our approach is based on a *decomposition* of this dial-a-ride problem into a “clustering” and a “chaining” step. Both of these steps lead to *set partitioning* problems that we attack with heuristic and branch-and-cut methods. These procedures form an optimization module that is the heart of a computer system that integrates and automates the complete operation of the Telebus center. This system is in use since June 3, 1995 and lead to improvements in service, cost reductions, and increased productivity of the center.
This chapter is organized in seven sections in addition to this introduction. Section 4.2 describes Berlin’s Telebus transportation system for handicapped people and our project to optimize the operation of this service. The core of the project was the development of mathematical optimization methods and software to solve the vehicle scheduling problem that comes up at Telebus; this particular dial-a-ride problem is discussed in Section 4.3. Section 4.4 introduces our two-phase clustering and chaining solution approach and the associated set partitioning models. The approach involves cluster and tour generation steps that are discussed in Sections 4.5 and 4.6. Computational experiences with our vehicle scheduling method are discussed in Section 4.7 and some possible future perspectives in the final Section 4.8.

Figure 4.1: A Telebus Picks Up a Customer.

4.2 Telebus

Accessibility of the public transportation system for *mobility disabled people* has become an important political goal for many municipalities: They introduce low-floor buses, install lifts in subway stations, etc. But many handicapped and elderly people still have problems because they need additional help, the next station is too far away, or the line they want to use is not yet accessible. Berlin, like many other cities, offers to these people a special transportation service: The so-called *Telebus* provides (i) *door-to-door transportation* and (ii) *assistance* at the pick-up and the drop-off point. The system is operated by the Berliner Zentralausschuß für Soziale Aufgaben e.V. (BZA), an association of charitable organizations, and financed by the Berliner Senatsverwaltung für Soziales (SenSoz), the city of Berlin's Department for Social Affairs. Figure 4.1 on page 158 shows a Telebus that picks up a customer.

Telebus is a *dial-a-ride system*: Every entitled user (currently about 25,000 people) can order up to 50 rides per month through the BZA's telephone center. If the order is placed one day in advance, Telebus guarantees to service the ride as requested, later "spontaneous" requests are serviced as possible. The advance orders, about 1,500 during the week and 1,000 on weekends, are collected and scheduled into a fleet of mini-buses that Telebus rents on demand from service providers like charitable organizations and commercial companies. These buses pick up the customers at the requested time (modulo a certain tolerance) and transport him/her to the desired destination; if required, the crew provides assistance to leave the apartment, enter the vehicle, etc. This service is available every day from 5:00 am in the morning to 1:00 am in the night. Figures 4.2 and 4.3 illustrate operation and organization of the Telebus system.

	order	customer
	telephone center	BZA
	bus scheduling	BZA
	bus renting	BZA
	transportation	service provider
	financing, goals	SenSoz

Figure 4.2: Operation of the Telebus System.

Telebus was established in 1981 and ever since then the number of customers and requests has been rapidly increasing. Figure 4.4 on page 161 gives an impression of the dramatic *history* of Telebus in this period of time; the numbers for the years up to 1993 are taken from T 336 of the report of the Rechnungshof von Berlin (Berlin's audit division) for the year 1994, the other data was provided by the BZA. We see first that there is a constant growth in the number of *entitled users*. But not all registered persons drive: The number of *customers*, i.e., persons that order rides, was basically constant and started to increase only after the reunification of Germany in 1990; the delay until 1992 is due to the initial lack of private telephones in the East. *Costs* got out of control in 1988, when a *taxi voucher system* was introduced that allowed for a certain *number* of spontaneous rides with taxis in addition to the bus rides.

Figure 4.3: Organization of the Telebus System.

When costs topped 30 million DM in 1991, drastic service reductions were taken to stop this trend: The voucher system was replaced by a *taxi account system* that limits taxi rides to 300 DM per person and month. But with new demand from East Berlin and a doubled area of service, costs were almost immediately up at 30 million DM again. What could be done?

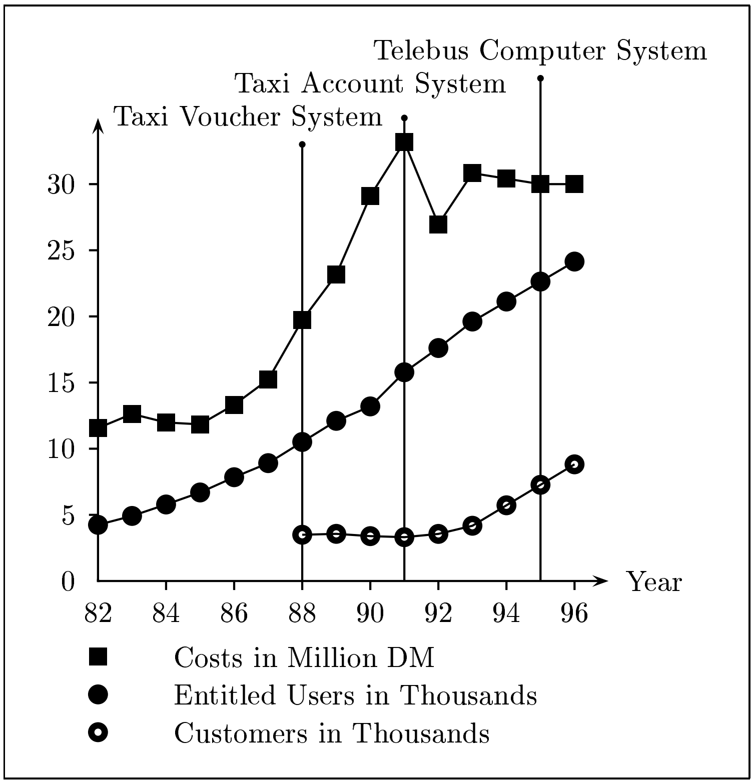


Figure 4.4: Increasing Usage and Costs of Telebus.

The best way to control costs without reducing the service was a better *vehicle scheduling* to service more requests for the same amount of money. The scheduling was traditionally done *manually* by experienced planners who could work out a feasible *bus plot* in about 16 man-hours. Now it became clear that this method was no longer appropriate to cope with rising demand and cost pressure. The core scheduling problem of the BZA could only be solved with modern computer hard- and software and the *Telebus project*, a cooperation involving the ZIB, the BZA, and the SenSoz (Intranetz joined later, see next paragraph), was started to develop it. The Telebus dial-a-ride problem, the methods that we use for its solution, and our computational experiences are what we are going to discuss in the subsequent sections of this chapter.

The project developed a broader scope. It soon turned out that a mathematical vehicle scheduling tool alone was not enough and the project evolved quickly into the development of a comprehensive *Telebus computer system*, that integrates and automates the complete operation of the BZA: Reservation, confirmation, and cancellation, vehicle scheduling, radio telephony, accounting, controlling, and statistics. The system consists of a tool box of software modules and runs on a network of 20 MacIntosh PCs; it is in operation since June 3, 1995. Design and installation of the Telebus computer system lead further to a *reorganization* of the center and the whole Telebus service with issues that ranged from a new bus renting

"mix" and "mode" to scheduling training of BZA staff. Fridolin Klostermeier and Christian Küttner, in particular, worked with great personal dedication for more than a year in the Telebus center, drove on Telebuses, etc. Finally, they even set up their own company, the Intranetz GmbH, that has scheduling systems for dial-a-ride systems as one of its business areas. More information on the *consulting aspect* of the Telebus project can be found in the articles Borndörfer et al. [1996, 1997a,b], and the thesis of Klostermeier & Küttner [1993] (all these publications are in German).
All these measures together —negotiations with vehicle providers, reorganization of center and service, the new computer system, and improved vehicle scheduling— resulted in

(i) Improvements in service: A reduction of the *advance call-in period* period from three days in 1992 to one day and increased *punctuality* of the schedule in comparison to the results of manual planning.

(ii) Cost reductions: Today, about 30% more requests can be serviced with the same resources as in 1992, see Figure 4.5 for a comparison of a month in 1994 before and in 1996 after the Telebus computer system went into operation.

(iii) More productivity in the Telebus center.

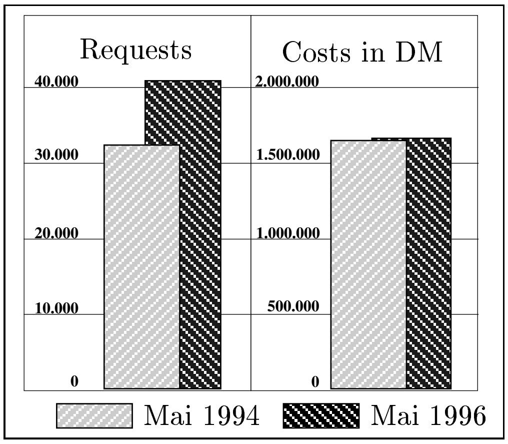


Figure 4.5: Results of the Telebus Project.

4.3 The Vehicle Scheduling Problem

The most important task at Telebus is the daily construction of the *vehicle schedule*, which determines the two most important objectives of the service: *Operation costs* and *customer satisfaction*. This vehicle scheduling problem is a *dial-a-ride problem* that can be stated in an informal way as follows:

(DARP) Given the customer requests, rent a suitable set of available vehicles and schedule all requests into them such that a number of constraints on the feasibility of vehicle tours are satisfied and operation costs are minimum.

In the remainder of this section, we discuss the Telebus DARP in detail.

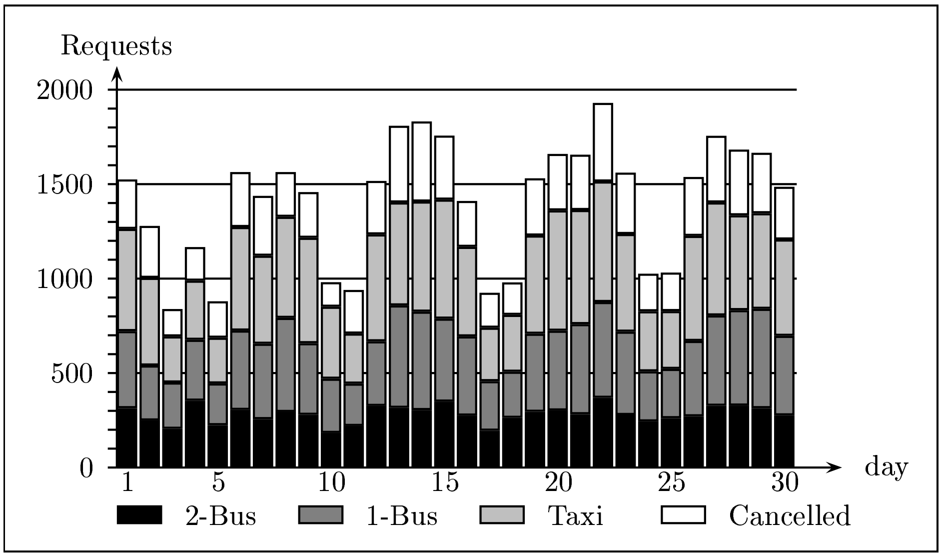


Figure 4.6: Telebus Requests in June 1995.

4.3.1 Pieces of Work

The basis for vehicle scheduling are the *vehicles*. Vehicles always come together with a crew for a possible shift of operation; following the terminology of Desrosiers, Dumas & Soumis [1988], we call such a part or all of a workday, during which a crew and a vehicle is *available* to service requests, a *piece of work*. The supply of pieces of work is determined by the vehicle providers who offer about 100 pieces of work of different types. The available pieces of work are known in advance and Telebus can rent them on a daily, weekly, or monthly basis (long term renting can be cheaper).
The following data is associated to a piece of work w:

	(i)	v_w	vehicle type: Teletaxi, 1-bus/2-bus small/large
	(ii)	$c(v_w) = (c^{\mathrm{c}}, c^{\mathrm{w}}, c^{\mathrm{nf}}, c^{\mathrm{f}}, c^{\mathrm{a}})(v_w)$	capacity:
(W)			total # of customers and wheelchairs
			# of non-folding/folding wheelchairs
			# of ambulatories
	(iii)	$G(w)$	group: vehicle type, depot, type of shift

The *vehicle types* are five: *Teletaxis* (taxis that are rented like buses), *1-buses* (with one driver), that can be small or large, and *2-buses*, also small or large. The vehicle type is important for deciding whether a request can be serviced by a particular piece of work: Teletaxis can transport only *ambulatories* and customers with *folding wheelchairs*, *non-folding wheelchairs* require a bus, and *staircase assistance* a 2-bus, see Figure 4.6 for statistics on Telebus requests which show a typical weekly distribution pattern.
Each vehicle has a *capacity*, that depends on the type: It can transport $c^{\mathrm{f}}(v_w)$ persons in folding and $c^{\mathrm{nf}}(v_w)$ persons in non-folding wheelchairs, but at most $c^{\mathrm{w}}(v_w)$ wheelchairs at the same time, plus $c^{\mathrm{a}}(v_w)$ ambulatories, in total at most $c^{\mathrm{c}}(v_w)$ customers. Teletaxis have a capacity of $c(v_w) = (3, 1, 0, 1, 3)$, i.e., they can service up to three customers at the same time and one of them can have a folding wheelchair. The small buses have a capacity of $c(v_w) = (5, 2, 2, 2, 3)$, large buses have $c(v_w) = (7, 3, 3, 3, 4)$; note that this allows to account for the (Telebus) rule that persons in folding wheelchairs travel in buses in their wheelchair.

Finally, the set W of all pieces of work falls into disjoint groups $W = \dot{\bigcup} G$. A *group* G is a set of pieces of work that are considered to be indistinguishable for the purpose of vehicle scheduling, i.e., the pieces of work of the same group have vehicles of the same type, which are stationed at the same *depot*, and they can be rented for identical *shifts* of operation; possible shifts are 8.5 and 10.5 hours fixed length, and early, late, and certain flexible shifts of variable duration. The groups will become important for the construction of vehicle tours, namely, we will require group specific parameter settings or even group specific algorithms to come up with tours that can be serviced by the pieces of a work of a given group.

4.3.2 Requests

The pieces of work will be used to service some number m of transportation *requests*.

	(i)	v_i^+, v_i^-	pick-up and drop-off event
	(ii)	$p(v_i^+), p(v_i^-)$	pick-up and drop-off location
	(iii)	$T(v_i^+) = [\underline{t}(v_i^+), \overline{t}(v_i^+)]$	pick-up time window
		$T(v_i^-) = [\underline{t}(v_i^-), \overline{t}(v_i^-)]$	drop-off time window
(R)	(iv)	$t^{++}(v_i^+), t^{++}(v_i^-)$	pick-up and drop-off service time
	(v)	$W(v_i^+), W(v_i^-)$	set of feasible pieces of work
	(vi)	$a(v_i^+) = (a^{\mathrm{c}}, a^{\mathrm{w}}, a^{\mathrm{nf}}, a^{\mathrm{f}}, a^{\mathrm{a}})(v_i^+)$,	total # of customers and wheelchairs
		$a(v_i^-) = (a^{\mathrm{c}}, a^{\mathrm{w}}, a^{\mathrm{nf}}, a^{\mathrm{f}}, a^{\mathrm{a}})(v_i^-)$	# of non-folding and folding wheelchairs
			# of ambulatories

Associated to each request i is a *pick-up node* v_i^+ and a *drop-off node* v_i^- that corresponds to the pick-up and drop-off *event* of a request; these nodes will be part of a space-time *transition network* that will be defined in the next section.
The pick-up and the drop-off nodes are mapped to *locations* or points $p(v_i^+)$ and $p(v_i^-)$ in a *road network* of Berlin (different from the transition network) that is shown in Figure 4.7. We estimate *travelling times and distances* by average values that are stored on the 2,510 edges of this network and use this data to compute shortest routes between the 828 nodes.
In addition to this spatial information, a request bears temporal data that is measured in units of 5 minutes of *Telebus time*. The customer communicates a *desired pick-up time* $t^\star(v_i^+)$ (or a desired drop-off time which is treated analogously) that gives rise to a window of feasible *pick-up times* $T(v_i^+)$. The pick-up time window is computed according to Telebus specific rules that try to find a compromise between punctual service and more degrees of freedom for the vehicle scheduling process. Currently, most requests have

$$T(v_i^+) = t^\star(v_i^+) + [-3, 3] \text{ (units of Telebus time)},$$

i.e., the vehicle is allowed to arrive up to 15 minutes early or late. Similar but more complex rules are used to determine a feasible *drop-off time window* $T(v_i^-)$: Here, the shortest possible travelling time and a maximum detour time play a role. Finally, some *service time* $t^{++}(v_i^+)$ and $t^{++}(v_i^-)$ is needed at the pick-up and the drop-off location.
The required assistance, the wheelchair type, etc. determine the set of *feasible pieces of work* $W(v_i^+) = W(v_i^-)$ that can or must be used to service the request; this set consists of all suitable groups.
$a(v_i^+)$ and $a(v_i^-)$ give the total number of customers and wheelchairs, the number of folding and non-folding wheelchairs, and the number of ambulatories that enter and leave the vehicle in the pick-up and drop-off event, respectively.

Figure 4.7: Highways and Major Roads in Berlin.

4.3.3 Constraints

Given the available pieces of work and the requests, a *schedule* of feasible vehicle *tours* has to be determined that satisfies a number of *constraints*. Following Desrosiers, Dumas & Soumis [1988], we distinguish the following types of constraints for feasibility:

	(i)	*visiting*	each pick-up and drop-off event has to be serviced exactly once
	(ii)	*pairing*	pick-up and drop-off of a request is serviced by the same vehicle
	(iii)	*precedence*	each customer must be picked up before dropped off
	(iv)	*time window*	pick-up and drop-off events must be serviced in time
(F)	(v)	*no stop*	it is not allowed to stop and wait with a customer on board
	(vi)	*capacity*	the vehicle capacity must not be exceeded
	(vii)	*depot*	the vehicle must return to its depot
	(viii)	*shift*	each piece of work must conform to its type of shift
	(ix)	*availability*	one can not use more pieces of work or others than available

Shift constraints arise from renting contracts and labour regulations for bus drivers. At Telebus, pieces of work have to be of certain fixed or maximum lengths and/or have to begin and end in certain time intervals, the exact parameters depend on the type of shift. Such types are, for example, 8.5 or 10.5 hour shifts between 5:00 am and 1:00 am and flexible shifts of variable length. Labour regulations prescribe maximum driving hours and obligatory breaks: A break of 30 minutes has to be taken between the fourth and sixth hour of a shift.
The meaning of the other constraints is self explanatory.

4.3.4 Objectives

The main *objective* of the DARP is to minimize *operation costs*, i.e., the costs for renting pieces of work from the service providers. *Customer satisfaction* is another important goal; it is treated by means of the time windows. Finally, Telebus uses some auxiliary objectives that reflect *security* issues. These criteria try to prefer "safe" tours to "risky" or "packed" ones in an attempt to safeguard against emergency situations and unpredictable events like cancellations, spontaneous requests, traffic jams, vehicle breakdowns, etc.

4.4 Solution Approach

In this section, we discuss our solution approach to the Telebus dial-a-ride problem. Starting from a network formulation of the DARP, we decompose the problem into a clustering and a chaining step. Both steps lead to set partitioning problems.

4.4.1 Transition Network

The basis for our solution approach is a formulation of the DARP in terms of a *transition network* $D = (V, A)$. The transition network is a space-time digraph, see Figure 4.8 for an illustration of the following construction.

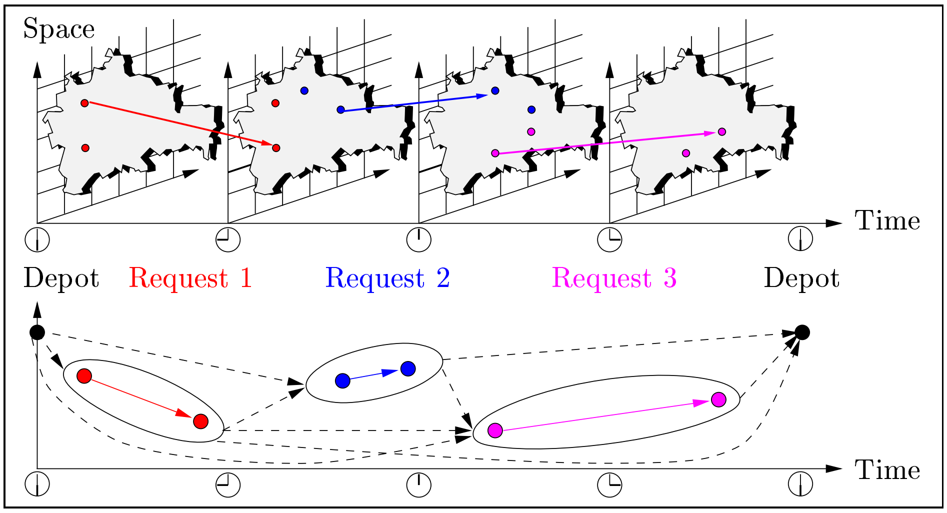


Figure 4.8: Constructing a Transition Network.

The transition network's set of *nodes* $V = V^+ \cup V^- \cup V^{G+} \cup V^{G-} \cup V^{\&}$ consists of all pick-up events $V^+ := \{v_i^+\}$, all drop-off events $V^- := \{v_i^-\}$, *tour start nodes* $V^{G+} := \{v_t^{G+}\}$ and *tour end nodes* $V^{G-} := \{v_t^{G-}\}$ for each group G of pieces of work and each possible point of Telebus time $t = 60, \ldots, 300$ ($60 * 5$ minutes is 5:00 am and $300 * 5$ minutes is 1:00 am on the next day), and *break nodes* $V^{\&} := \{v_t^{\&}\}$ for all Telebus times. We set[6]

$$\begin{aligned}
T(v_t^{G+}) :=& \quad T(v_t^{G-}) := T(v_t^{\&}) := \{t\},\\
t^{++}(v_t^{G+}) :=& \ t^{++}(v_t^{G-}) := 0 \qquad \text{and} \quad t^{++}(v_t^{\&}) := 6,\\
a(v_t^{G+}) :=& \quad a(v_t^{G-}) := a(v_t^{\&}) := 0, \quad \text{and}\\
W(v_t^{G+}) :=& \quad W(v_t^{G-}) := G \qquad \text{and} \quad W(v_t^{\&}) := W.
\end{aligned}$$

The *arcs* of the network are defined to reflect the local feasibility of possible vehicle tours. We draw an *event arc* uv between two event nodes u and v if

$$\underline{t}(u) + t^{++}(u) + t_{uv} \le \overline{t}(v),$$

that is, if it is possible to arrive at u, service u, drive to v, and arrive there in time. Here, we denote by t_{uv} the shortest time to get from location $p(u)$ to $p(v)$ in the road network.

[6] We use here the same symbol t for indices and variables, but we hope the notation is nevertheless clear.

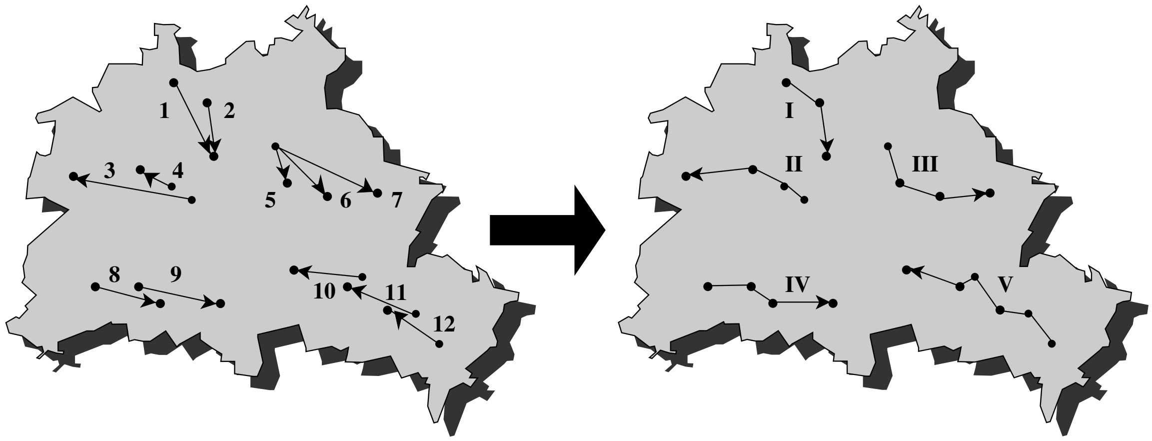
Analogously, we introduce *tour start arcs* from the tour start nodes to the event nodes, *tour end arcs* from the event nodes to the tour end nodes, and *break start arcs* and *break end arcs* from the event to the break nodes and vice versa. More precisely, we draw a tour start arc $v_t^{G+}v$ from a tour start node $u = v_t^{G+}$ to a pick-up node v if

$$t + t_{uv} \leq \bar{t}(v) \quad \text{and} \quad W(v) \supseteq G,$$

where t_{uv} is the time to get from the location of the depot associated to the group G of pieces of work to the event location $p(v)$. Tour end, break start, and break end arcs are defined in the same way, only that the break arcs get zero duration $t_{vv_t^\&} := t_{v_t^\& v} := 0$.
With this terminology, we can state the DARP in terms of the transition network. *Feasible vehicle tours* correspond to such *dipaths* in D that satisfy the constraints (F) (ii)–(viii) as stated in Subsection 4.3.3 on page 165; we assume here that a break is taken by visiting a break node. A *feasible vehicle schedule* is a collection of feasible vehicle tours that satisfies the remaining constraints (F) (i) and (ix) as well. The DARP is the problem to find a best possible schedule with respect to some (not yet precisely defined) objective function.

4.4.2 Decomposition

The construction of feasible vehicle tours in the transition network as dipaths subject to additional constraints is, although simple in principle, difficult in practice because of the many constraints (F). We use a *decomposition approach* to cope with this difficulty in a heuristic way. The method focusses on local feasibility in a first step. When validity of the local constraints is ensured, we deal with the remaining global restrictions in a second step.



I	Collection with Common Drop-off	IV	Concatenation
II	Insertion	V	Continued Concatenation
III	Collection with Common Pick-up		

Figure 4.9: Clusters at Telebus.

The decomposition is based on the concept of a *cluster* or, as schedulers at the BZA say, a “Verknüpfung”. A cluster is a dipath in the transition network that qualifies as a segment of a vehicle tour in the sense that it satisfies the “local” constraints (F) (ii)–(vi): Pairing, precedence, time windows, no stop, and capacity. Figure 4.9 shows a number of typical clusters at Telebus: *Collections*, *insertions*, simple and continued *concatenations*.

We denote a cluster-dipath $C = (v^1, \dots, v^k)$ as a sequence of visited nodes. In doing so, we want to adopt the convention that a cluster contains only pick-up and drop-off nodes, i.e., no tour start, tour end, or break nodes. We also stipulate that a cluster contains a node only once.

A cluster C satisfies the pairing constraints if it contains for every pick-up node the corresponding drop-off node and vice versa. The precedence constraints hold if every pick-up node of the cluster precedes its drop-off counterpart. We say that the capacity constraints are valid for C if there exists a piece of work w such that the capacity $c(v_w)$ of the associated vehicle type is always at least as large as the *load* $a_i(C)$ at each node v^i of the cluster:

$$c(v_w) \geq a_i(C) := \textstyle\sum_{j=1}^{i} a(v^j), \quad i = 1, \dots, k.$$

The time window and the no stop constraints hold if the recursion

$$\begin{aligned} T_1(C) &:= T(v^1) \\ T_{i+1}(C) &:= \big(T_i(C) + t^{++}(v^i) + t_{v^i v^{i+1}}\big) \cap T(v^{i+1}), \quad i = 1, \dots, k-1, \end{aligned} \tag{4.1}$$

that computes the *feasible time windows at the cluster nodes*, terminates with $T_k(C) \neq \emptyset$. (Here, we denote by $[a, b] + t$ the interval $[a + t, b + t]$.) In this case, it is possible to start the service of the cluster at the first node v^1 at a feasible time in $T_1(C)$, service v^1, go *immediately* (no stop) to the next node, arrive there at a feasible time $T_2(C)$, service v^2, and so on until the vehicle arrives at the last node v^k in the feasible time interval $T_k(C)$.

Before we discuss the use of clusters for vehicle scheduling, let us record the data that we associate with a cluster:

(C)	(i)	$C := (v^1, \dots, v^k)$	cluster as sequence of visited nodes
	(ii)	$t^{++}(C)$	cluster service time
	(iii)	$T(C)$	cluster start time window
	(iv)	$W(C)$	set of feasible pieces of work

By the no stop constraints and recursion (4.1), the *service time of a cluster* is *constant*:

$$t^{++}(C) := \textstyle\sum_{i=1}^{k} t^{++}(v^i) + \sum_{i=1}^{k-1} t_{v^i v^{i+1}}.$$

This results in a *cluster start time window* of possible times to begin the service of a cluster:

$$T(C) := [\underline{t}(C), \overline{t}(C)] := T_k(C) - t^{++}(C).$$

Finally, there is the set

$$W(C) := \{w \in \textstyle\bigcap W(v^i) \mid a_i(C) \leq c(v_w), \quad i = 1, \dots, k\}$$

of pieces of work that can possibly service C.

Clusters are useful for vehicle scheduling, because they can serve as the building blocks of vehicle tours: We can *chain* clusters to feasible tours just as we constructed clusters from the individual requests. As the clusters already satisfy the local constraints (F) (ii)-(vi), the chaining can concentrate on the remaining global conditions (F) (i) and (vii)–(ix); the only local constraints that appear again are the time window constraints (F) (iv) that transform into the cluster start time windows. This largely *independent treatment* of local and global constraints is one of the main benefits of request clustering.

These observations suggest the following two step *decomposition approach* to the DARP:

(i) *Clustering Step:* Construct a set of feasible clusters.

(ii) *Chaining Step:* Chain clusters to a set of tours that constitute a feasible schedule.

This generic clustering/chaining method is the vehicle scheduling procedure that we use for the solution of the DARP.

4.4.3 Set Partitioning

A refinement of the two steps of the clustering/chaining vehicle scheduling method leads to clustering and chaining *set partitioning problems* of identical structure.

The objective of the *clustering* step is to construct a set of clusters that is "good" in the sense that it provides a "reasonable" input for the chaining phase. In the best case, the clustering algorithm would yield a set of clusters that can be chained to an optimal solution of the DARP. While this is of course a hopeless criterion, one can look for computable *necessary conditions* that an optimal set of clusters must satisfy. If such conditions can be found, they can be used as a measure for the quality of a set of clusters and as a guide to construct it.

One way to derive a necessary condition is to note that any feasible schedule decomposes in a canonical way: The maximal tour segments such that the vehicle is always loaded form a set of minimal clusters with respect to set inclusion (interpreting, for the moment, clusters as sets of nodes) and this *minimal cluster decomposition* of a schedule is unique. Then, a necessary condition for the global optimality of a schedule is that its cluster decomposition is also *locally* optimal in the sense that the objective can not be improved by rescheduling the service of individual clusters.

Assuming that a local objective value can be associated to and computed for an individual cluster, we can approximate the global objective value of the schedule by the sum of the local cluster objectives. Applying this simplification to the DARP results in the following optimization problem over clusters:

(CLUSTER)	Given the customer requests, find a set of clusters such that each request is contained in exactly one cluster and the sum of the cluster objectives is minimal.

We use CLUSTER as our formulation of the clustering step; this model aims at inputs for the chaining phase that are optimal in a heuristic but *well-defined* sense.

Popular local optimization criteria for clustering are the *internal travelling time* (ITT) of a vehicle in a cluster, the *internal travelling distance* (ITD), and mixtures of these. Clusters with small ITT or ITD aim at a good vehicle usage in terms of transported customers per kilometer or per minute. One would expect that a minimal ITT or ITD clustering makes use of large vehicle capacities by transporting several customers at once where possible, see again Figure 4.9 for examples in this direction. In other words: Minimal ITT or ITD clustering yields "reasonable" results that planners accept in practice.

CLUSTER can be formulated as a *set partitioning problem*

$$\text{(SPP)} \quad \min\ c^{\mathrm{T}}x \quad Ax = \mathbf{1},\ x \in \{0,1\}^n,$$

where A is the $m \times n$ incidence matrix of requests versus clusters and $c \in \mathbb{R}^n$ is the vector of cluster objectives.

Having decided for a set of clusters, we can treat the *chaining* step in exactly the same way as we just did with the clustering step. Approximating (or even expressing) the objective value of the DARP as a sum of objectives of individual tours, the DARP for *fixed clusters* simplifies to (becomes) the following optimization problem over tours:

(CHAIN) Given a clustering, find a set of vehicle tours such that each cluster is contained in exactly one tour and the sum of the tour objectives is minimal.

We use CHAIN as our formulation for the chaining step; natural objectives associated to tours are operation costs for vehicles and/or customer satisfaction criteria like accumulated waiting time.
CHAIN can also be modelled as a set partitioning problem but needs one additional thought. In the simplest case, the matrix A records the incidences of clusters versus tours, and c is the vector of tour costs; this model is correct if there are no *availability* constraints. The presence of availability constraints leads to additional equations and variables, but the enlarged model is again of set partitioning type. Namely, availability constraints for a piece of work w prescribe that one can only choose at most one of the (incidence vectors of) tours $A_{J(w)}$. that correspond to w:

$$\sum_{j \in J(w)} x_j \leq 1.$$

Adding a slack variable and appending the new row and column to (SPP) results again in a set partitioning problem.

4.4.4 A Vehicle Scheduling Algorithm

We are now ready to state the *vehicle scheduling algorithm* that we propose for the solution of the DARP. The algorithm is a refinement of the generic clustering/chaining method of Subsection 4.4.2 in terms of the clustering problem CLUSTER and in terms of a *subproblem* of the chaining problem CHAIN:

(i) *Cluster Generation:* Enumerate *all possible* feasible clusters. Set up its clustering SPP.

(ii) *Clustering:* Solve the clustering SPP.

(iii) *Tour Generation:* Enumerate a *subset* of all feasible tours. Set up its chaining SPP.

(iv) *Chaining:* Solve the chaining SPP.

Here, the term "setting up the clustering SPP for a set of clusters" means to construct the request-cluster incidence matrix A and to compute the cluster cost vector c to set up a *clustering set partitioning problem* for the given set of clusters. The analogous expressions are used in the chaining case, but here we enumerate only some *subset* of all feasible tours to construct only a *submatrix* of the complete cluster-tour incidence matrix and a *subvector* of the complete tour cost vector. The resulting *chaining set partitioning problem* is hence a *subproblem* of the complete chaining SPP. The reason for this simplification is that it is out of the question to set up the complete chaining SPP: The number of possible tours is in the zillions (where zillion is an incredibly large number). Restricting the chaining SPP to some subset of "promising" tours is our heuristic way of dealing with this difficulty.
We use the branch-and-cut algorithm `BC`, see Chapter 3 of this thesis, to solve the clustering and chaining set partitioning problems. How we do the cluster and tour generation is described in the following Sections 4.5 and 4.6.

4.4.5 Related Literature

Dial-a-ride problems and solution approaches similar to our's have been discussed in a number of publications in the *literature.* In Bodin & Golden [1981]'s vehicle routing and scheduling classification scheme, the Telebus DARP qualifies as a *subscriber dial-a-ride routing and scheduling problem* and our method is a *cluster-first schedule-second* algorithm, with some of the clustering transferred to the scheduling-chaining phase. For survey articles on dial-a-ride and, more general, vehicle routing and scheduling problems we give the classic Assad, Ball, Bodin & Golden [1983, see Sections 4.8–4.10 for DARPs], the thesis of Sol [1994, Chapter 1], Desrosiers, Dumas, Solomon & Soumis [1995, see Chapter 6 for DARPs], and the annotated bibliography of Laporte [1997], and we suggest Barnhart et al. [1994] and the literature synopsis of Soumis [1997] for references on column generation techniques.
The termini "clustering" and "chaining" stem from Cullen, Jarvis & Ratliff [1981], who develop a set partitioning based two-phase clustering/chaining vehicle routing algorithm. Their approach differs from the one we give here in the use of *column generation techniques* and a possible overlap of clusters in the chaining phase. Ioachim, Desrosiers, Dumas & Solomon [1991], based on earlier work of Desrosiers, Dumas & Soumis [1988], report about clustering algorithms for vehicle routing problems in handicapped people's transport using column generation and a problem decomposition into *time slices.* Tesch [1994] develops a set partitioning method that optimizes over a fixed set of heuristically generated columns to solve dial-a-ride problems that come up in the German city of Passau. We finally mention Sol [1994] as a recent reference for the use of column generation techniques for pick-up and delivery problems.

4.5 Cluster Generation

We discuss in this section the algorithm that we suggest for cluster generation: Recursive enumeration of all dipaths in the transition network $D = (V, A)$ that correspond to feasible clusters by *depth first search.*
The procedure works with sequences of nodes that are extended in all possible ways until they eventually form clusters. Such a sequence $S = (v^1, \dots, v^k) \in V^*$ is called a *state*, where V^* denotes the set of all finite sequences of elements of V. Not every state can be extended to a cluster; necessary conditions for *feasibility* of a state S are

(i)	*event sequence*	$S \in (V^+ \times V^-)^*$	S contains only event nodes
(ii)	*no loop*	$v^i \neq v^j \quad \forall i \neq j$	S contains each node at most once
(iii)	*precedence*	$\forall v^i = v_p^+, v^j = v_p^- : i < j$	pick-ups precede drop-offs
(iv)	*time windows, no stop*	$T(S) \neq \emptyset$	the state time window is nonempty
(v)	*capacity, availability*	$W(S) \neq \emptyset$	there is a feasible piece of work

Here, we use the expressions *time window of a state* and its *set of feasible pieces of work* in analogy to the terms for clusters of the same name, see the definitions on page 168. A state is a cluster or *terminal* if it is feasible and the constraint for

(vi)	*pairing*	$\exists v^j = v_p^- \quad \forall v^i = v_p^+$	there is a drop-off for each pick-up

holds. Finally, a new state $S' = (v^1, \dots, v^{k+1})$ is produced from $S = (v^1, \dots, v^k)$ by appending a node v^{k+1}; this *transition* is denoted by

$$S' = S \leftarrow v^{k+1}.$$

To state our depth first search cluster generation algorithm in `C`-type pseudocode, we introduce predicates `infeasible` : $V^* \mapsto \{0,1\}$ and `terminal` : $V^* \mapsto \{0,1\}$ for feasibility and terminality of states. `tail` : $S^* \mapsto V$ is a function that returns the terminal node v^k of a state, and the procedure `output` is supposed to compute the cost and the request-state incidence vector of its argument and to store the result somewhere.

```
void dfs (state S, digraph D=(V,A)))
{
    if (infeasible (S)) return;
    if (terminal (S)) output (S);

    for all u ∈ γ+(tail(S))
        dfs (S ← u, D);
}

void cluster (digraph D=(V,A))
{
    for all v_i^+ ∈ V
        dfs ((v_i^+), D);
}
```

Figure 4.10: Enumerating Clusters by Depth First Search.

The complete cluster generation procedure `cluster` is given in Figure 4.10. Here, $\gamma^+(v)$ denotes the set of endnodes of arcs in D that go out from v.

The running time of `cluster` can be improved by strengthening the predicate `infeasible` by further *state elimination* criteria. For example, S is infeasible when it contains an unserviced pick-up v_p^+ that can not be dropped off in time regardless how S is extended:

(vii) *timeout* $\exists v^i = v_p^+ : \underline{t}(S) + t_{v^k v_p^-} > \overline{t}(v_p^-)$ $\quad$ v_p^+ can not be dropped in time

`cluster`, with the `infeasible` predicate strengthened by (vii), is the cluster generation algorithm that we use for vehicle scheduling at Telebus. To make this method work in practice, one needs, of course, efficient data structures, recursive updates of the predicates, and many other ingredients; the reader can find the implementation details in the thesis of Klostermeier & Küttner [1993] (German). For a typical Telebus DARP with 1,500 requests, our depth first search procedure enumerates the *complete set* of all possible clusters in a couple of minutes. Depending on the values of the time window, lateness, detour, and some other BZA parameters for cluster feasibility, this usually results in 100,000, sometimes up to 250,000, feasible clusters for input into the clustering set partitioning model.

We remark that similar results are *not* reported for comparable clustering problems in the literature. For instance, Ioachim, Desrosiers, Dumas & Solomon [1991] develop a *multilabel shortest path algorithms* for cluster generation problems that come up in the optimization of Toronto's Wheel-Trans Service. Although this dynamic program uses elaborate state space elimination criteria, special initialization strategies and data structures, and sophisticated preprocessing techniques to reduce the size of the transition network, it is in this case not possible to enumerate all feasible clusters.

Two Telebus specific factors may be responsible for the different outcome in our case. One is the combination of average service, driving, and detour times: As a rule of thumb, a transportation service takes in Berlin 5 minutes for pick-up, 20 minutes of driving, and another 5 minutes for drop-off. When the maximum detour time is 15 minutes, one will already be happy to pick up a single additional customer *en route*. Second, not every technically feasible cluster is accepted by BZA schedulers. To safeguards against accumulating delays etc., they often impose additional restrictions and forbid continued concatenations above a maximum length. These two factors limit the number of feasible clusters to a computable quantity.

4.6 Tour Generation

The topic of this section are tour generation algorithms that chain clusters to feasible vehicle tours. Starting from a simplified network formulation of the chaining problem, we develop a recursive depth first search tour enumeration algorithm and a number of tour generation heuristics. Some of these heuristics can also be used as stand-alone vehicle scheduling tools.

4.6.1 Chaining Network

The tour generation algorithms of this section work on a *chaining (transition) network* $\overline{D} = (\overline{V}, \overline{A})$ that one obtains from the transition network $D = (V, A)$ by a contraction of clusters. To give a more precise description of this construction, let $\dot{\bigcup} C^i = V^+ \cup V^-$ be a clustering of requests. $\overline{D}$ is set up from the transition network D in two steps. We (i) delete for each cluster $C = (v^1, \ldots, v^k)$ all entering and leaving arcs except the ones that enter the first node v^1 and the ones that leave the last node v^k, i.e., we delete all arcs from $\delta(C) \setminus \big(\delta^-(v^1) \cup \delta^+(v^k)\big)$. When this has been done, we (ii) contract each cluster into a single (super)node that we denote with the same symbol C. Note that we inherit in this way the definitions for the service time of a cluster(-node) $t^{++}(C)$, its start time interval $T(C)$, and its set $W(C)$ of feasible pieces of work.

4.6.2 Tour Enumeration

Feasible vehicle tours correspond to dipaths in the chaining network that satisfy the constraints (F). Such dipaths can be enumerated in much the same way as the cluster-dipaths in the transition network by a *depth first search* procedure. Using identical terminology and analogous definitions as for the cluster generation, a *state* $S = (\overline{v}^1, \ldots, \overline{v}^k) \in \overline{V}^*$ is *feasible* when the following conditions hold:

(i)	*tour start*	$\exists G, t : \overline{v}^1 = v_t^{G+}$	start at a tour start node
(ii)	*no loop*	$\overline{v}^i \neq \overline{v}^j \quad \forall i \neq j$	S contains each node at most once
(iii)	*time windows*	$T(S) \neq \emptyset$	the state time window is nonempty
(iv)	*availability*	$W(S) \neq \emptyset$	there is a feasible piece of work
(v)	*shift*	$\exists \overline{v}^i = v_t^{\&} : t \in \underline{t}(\overline{v}^1) + [48, 66]$	break during 4th–6th hour of shift
		$\underline{t}(\overline{v}^k) - \underline{t}(\overline{v}^1) \leq \overline{t}(G)$	maximum shift length respected
		$\ldots$	etc.

Here, we denote by $\overline{t}(G)$ the maximum duration of a piece of work of group G. The only difference to cluster generation is the update of the time window, that must allow for *waiting* (stops) between the service of two clusters:

$$T_{i+1}(S) := \big(T_i(S) + t^{++}(S) + t_{\overline{v}^i \overline{v}^{i+1}} + \mathbb{R}_+\big) \cap T(\overline{v}^{i+1}), \quad i = 1, \ldots, k-1.$$

A state is a tour or *terminal* if it is feasible and the

(vi)	*depot*	$\exists G, t^+, t^- : \overline{v}^1 = v_{t^+}^{G+}, \overline{v}^k = v_{t^-}^{G-}$	tour start and end at the same depot

constraint holds. Continuing with the dipath enumeration exactly as we did for the cluster generation in Section 4.5, we arrive at a very similar depth first search tour enumeration routine `chain`, see Figure 4.11 on the next page for a `C`-type pseudocode listing.

```
void dfs (state S, digraph D̄=(V̄,Ā))
{
    if (infeasible (S)) return;
    if (terminal (S)) output (S);

    for all u ∈ γ⁺(tail(S))
        dfs (S ← u, D̄);
}

void chain (digraph D̄=(V̄,Ā))
{
    for all v_t^{G+} ∈ V̄
        dfs ((v_t^{G+}), D̄);
}
```

Figure 4.11: Enumerating Tours by Depth First Search.

Computational practice, however, turns out to be completely different. While there were only some hundred thousand feasible clusters, the *number of tours* is zillions! The reason for this change is not the additional tour start, tour end, and break nodes (there are many, but not too many of these), but the possibility to wait between the service of two clusters. This degree of freedom, that is not available for cluster generation, leads to an enormous increase in the number of eligible clusters to extend a state: Looking one hour in the future, any cluster qualifies as a possible follow-on. Unlike in clustering, tour state extension does not have a local character and, although the `chain` routine works as fast as the `cluster` generator, there is no point in attempting a complete enumeration of all feasible tours.

The way that we deal with this difficulty is by reducing the number of arcs in the chaining network heuristically. One of our strategies is, for example, to keep only a constant number of outgoing arcs at each cluster node that are selected by local criteria, the "k best successors". While such methods are likely to produce individual efficient tours in some number, there is no reason other than pure luck to believe that the right set of unavoidable "garbage collection tours", that complete a good schedule, will also be produced in this way. We are aware of this fact and mention this unsatisfactory arc selection as a weak point in our vehicle scheduling algorithm. What we can do, however, is to produce, in some minutes of computation time, several hundred thousands of tours as input for the chaining set partitioning problem, see again Klostermeier & Küttner [1993] for more implementation aspects.

Löbel [1997] has dealt in his thesis with a similar arc selection problem in the context of *multiple depot vehicle scheduling*. He has developed a *Lagrangean pricing* technique that resolves this issue for his extremely large scale problems completely. It is perhaps possible to use this technique, based on a suitable *multi commodity flow relaxation* of the DARP, to obtain better chaining results and we have in fact performed some preliminary computational experiments in this direction. These computations indicated, in our opinion, a significant potential for this approach. We remark that a multi commodity flow relaxation gives also rise to *lower bounds* for the complete chaining problem (CHAIN).

4.6.3 Heuristics

The chaining transition network $\overline{D}$ can also serve as a basis for all kinds of combinatorial vehicle scheduling *heuristics* to produce individual tours or to produce complete schedules. Heuristic scheduling is a particularly attractive method of tour generation because it provides not only "reasonable" *input* for the chaining set partitioning problem, but also *primal solutions* and *upper bounds*. We give here a list of heuristics that we have developed for Telebus, a more detailed description can be found in Klostermeier & Küttner [1993].

Our first method is designed for the construction of individual tours.

K Best Neighbors. We have already mentioned the idea of the k *best neighbors* heuristic: Applying the depth first search algorithm `chain` to a reduced version of the chaining network where at most k arcs have been selected from each set $\delta^+(\overline{v})$ of arcs that go out from a node. We use in our implementation local criteria like proximity to select the arcs that lead to a node's "k best neighbors".

The following heuristics produce complete vehicle schedules.

Tour-by-Tour Greedy. This heuristic produces the tours of a complete vehicle schedule iteratively one by one. Starting from some tour start node, the tour is extended by "best fitting" follow-on clusters (including breaks) in a greedy way until a tour end node is reached. The serviced clusters are removed from the chaining network, the next tour is started, and so on. The tour-by-tour greedy heuristic tends to produce "good" tours in the beginning and worse later when only far-out or otherwise unattractive clusters are left.

Time Sweep. This method uses some linear *order* on the clusters, the "time". The planning process constructs all tours of a complete schedule simultaneously. In each step of the time sweep, the next cluster with respect to the given order is assigned to the best tour with respect to some local criterion, until all clusters are scheduled into tours. The orders that we use are the natural ones from morning to evening and vice versa, and a "peaks first" variant that tries to smooth the morning and afternoon *demand peak* by scheduling this demand first.

Hybrid. The tour-by-tour and the time sweep heuristic can be seen as the extreme representatives of a class of vehicle scheduling heuristics that vary from the construction of individual tours to a simultaneous construction of all tours by assigning clusters to tours in some order. Hybrid belongs to such a class of mixtures of these two procedures: It does a time sweep, but it adds not only one follow-on cluster to a tour, but some sequence of several clusters.

Assignment. This method belongs to the same class as the hybrid heuristic, but it aims at some global overview. The assignment heuristic subdivides the planning horizon into time slices (we use a length of 30 minutes) that are considered in the natural order. In each step, a best assignment (with respect to some local criterion) of all clusters in the next time slice to the current set of partial tours is computed, starting new tours if necessary.

BZA. A set of other methods imitates the traditional *hand-planning* methods of the BZA. First, the request clusters are grouped according to time and space such that the clusters in one group start in the same hour and city district (or similar criteria). Doing a time sweep from morning to evening, one constructs tours with an eye on the distribution of clusters and vehicles in the city districts. In the starting phase of the Telebus project, these methods were particularly important to build up confidence in computerized scheduling, because they can be used to produce vehicle schedules of "familiar" type.

The heuristic vehicle scheduling methods that we have just described already produce, in a few minutes, schedules that have significantly lower operation costs than the results of a manual planning. And they do not use "a posteriori changes of scheduling rules" (that is, they do not produce infeasible tours!), which lead to a quantum leap in punctuality of the schedule. Klostermeier & Küttner [1993] give a detailed account of these improvements.
We use the heuristic methods of this subsection as a stand-alone scheduling tool and, in combination with the enumeration routine `chain` of the previous subsection, to set up chaining set partitioning problems with up to 100,000 columns.

4.7 Computational Results

We report in this section on computational experiences with our vehicle scheduling system: The cluster and tour generation modules and the heuristics of Sections 4.5 and 4.6 and the branch-and-cut solver `BC`, that is described in Chapter 3 of this thesis. Our aim is to investigate two complexes of *questions:*

(i) *Performance.* What is the performance of our vehicle scheduling system on Telebus instances? Can we solve the clustering and chaining set partitioning instances?

(ii) *Vehicle Scheduling.* Does our system result in a better vehicle scheduling? Does clustering reduce the internal travelling time (ITT)/internal travelling distance (ITD)? Does the chaining set partitioning model yield better results than the heuristics?

Our *test set* consists of 14 typical Telebus DARPs: 7 from the week of April 15–21, 1996 (instances v0415–v0421[7] (clustering) and t0415–t0421[7] (chaining)) and another 7 for the week of September 16–22, 1996 (instances v1616–v1622[7] (clustering) and t1716–t1722[7] (chaining)). April 20/September 21 and April 21/September 22 were Saturdays and Sundays, respectively. The two weeks differ in the adjustment of feasibility parameters for clusters and tours. Generally speaking, the *April* instances represent a restrictive scenario with continued concatenations limited to a maximum length of only three, small detour times, etc. The *September* problems were produced in a liberal setting with more degrees of freedom; the maximum concatenation length was, e.g., doubled to six.

We have run our vehicle scheduling system on these problems and report in the following three subsections about the *results.* We give statistics on solving the clustering and chaining set partitioning problems, and we investigate the relevance of our integer programming approach for vehicle scheduling at Telebus. We do not give detailed statistics for cluster and tour generation, because these steps are not a computational bottleneck; the interested reader can find such data in Klostermeier & Küttner [1993].

Following the *guidelines* of Crowder, Dembo & Mulvey [1979] and Jackson, Boggs, Nash & Powell [1991] for reporting about computational experiments, we state that all test runs were made on a Sun Ultra Sparc 1 Model 170E workstation with 448 MB of main memory, running SunOS 5.5, that our branch-and-cut code `BC` was written in ANSI `C` compiled with the Sun `cc` compiler and switches `-fast -xO5`, and that we have used the CPLEX [1995] Callable Library V4.0 as our LP solver.

Our computational results are listed in tables that have the following *format.* Column 1 gives the name of the problem, columns 2–4 its size in terms of numbers of rows, columns, and nonzeros, and columns 5–7 the size after an initial preprocessing. The next two columns give solution values. $\overline{z}$ reports the value of the best solution that could be found. This number is a proven optimum when the duality gap is zero, which is indicated by a —. Otherwise, we are left with a nonzero duality gap $(\overline{z}-\underline{z})/\overline{z}$, where $\underline{z}$ is the value of the global lower bound. The following five columns give statistics on the branch-and-cut algorithm. There are, from left to right, the number of in- and out-pivots (Pvt)[8], cutting planes (Cut), simplex iterations to solve the LPs (Itn), LPs solved (LP), and the number of branch-and-bound nodes (B&B). The next five columns give timings: The percentage of the total running time spent in problem reduction (PP), pivoting (Pvt), separation (Cut), LP-solution (LP), and the heuristic (Heu). The last column gives the total running time in CPU seconds.

[7]Available at URL `http://www.zib.de/borndoerfer`

[8]Confer Chapter 3 for an explanation of this concept.

4.7.1 Clustering

Table 4.1 lists our clustering results. The first seven rows of this table correspond to the April instances, the next fourteen to the September instances which were solved twice: We used a time limit of 7,200 seconds to produce the results in rows 8–14 and 120 seconds in rows 15–21. We can see from column 2 of the table that the DARP *instances* that we are considering here involve 1,500 or more requests during the week (Tuesday is usually a peak), and significantly less requests on weekends. These numbers were typical for Telebus in 1996. The requests were clustered in all possible ways and this resulted in the number of clusters that is reported in column 3. The restrictive parameter settings for April lead to a rather small number of feasible clusters, only about four times the number of requests (v0417 is an exceptional instance that contains extraordinary large collective requests). More planning freedom in September lead to a 5-fold increase in the number of feasible clusters. We note that the average April cluster contains three requests, while the number for September is four.

As the number of feasible clusters for April is very small, one would expect that clusters do not overlap much and that there are often not many choices to assign requests to clusters. The statistics on *preprocessing* in columns 5–7 of rows 1–7 show that this is indeed so. The extremely large reduction in the number of rows indicates that, in particular, many requests can only be assigned in a single way to a cluster (either to a single possible cluster or in exactly the same way as some other request). The results for September are different, see rows 8–15. We observe also significant and encouraging problem reductions, but not to the same extreme extent as for the April problems.

The trends that we observed in the preprocessing step continue in the *branch-and-cut* phase. Largely orthogonal columns and few rows in the April instances translate into simple LPs with more or less integral solutions. The problems could be solved with a few LPs, cutting planes, and branch-and-bound nodes, two even at the root of the searchtree, see columns 11–15 in rows 1–7. Iterated preprocessing played a major role in these computations, as can be seen from the large number of pivots in column Pvt (this is a measure for successful preprocessing, see Subsection 3.2.13); note that the code spent about half of the total running time in problem reduction (sums of Timing columns PP and Pvt). All in all, about three minutes of CPU time were always sufficient to solve the easy April problems to proven optimality. The situation is different for the September data. The problems are larger, and substantial overlap in the clusters results in highly fractional LPs. Significant computational effort and extensive branching is required to solve the September problems, see columns 11–15 of rows 8–14; in fact, three instances could not be solved completely within 7,200 seconds. But the remaining duality gaps are so small that any practitioner at the BZA is perfectly happy with the solutions. And these results can even be obtained much faster: Setting the time limit to only 120 seconds yields already solutions of very good quality, see column Gap in rows 15–21.

The objective that we used in the April and September clustering set partitioning problems was a mixture of ITD and a penalty that discourages the clustering of taxi requests; servicing all but the most “clusterable” taxi requests with individual taxi rides was BZA policy at that time. Figure 4.12 compares on its left side the number of requests and the number of clusters that were obtained by optimizing this mixed criterion for the September data. Note that the number of taxi clusters (that contain only taxi requests) is largely identical to the original number of taxi requests, i.e., the taxi requests were essentially left unclustered. The observed reductions are thus solely due to the clustering of bus requests. The right side gives an impression of the reduction of ITD that can be achieved with a clustering of this type.

	Original Problem			Preprocessed			Solutions		Branch-and-Cut					Timings/%					Total
Name	Rows	Cols	NNEs	Rows	Cols	NNEs	$\overline{z}$	Gap/%	Pvt	Cut	Itn	LP	B&B	PP	Pvt	Cut	LP	Heu	Time[a]
v0415	1518	7684	20668	598	4536	10988	2429415	—	12774	70	755	36	9	32	32	4	12	3	5.68
v0416	1771	19020	58453	812	11225	33991	2725602	—	325151	1305	4677	1970	643	19	28	4	15	3	120.53
v0417	1765	143317	531820	715	55769	206131	2611518	—	61309	294	1360	171	41	35	21	8	10	3	174.07
v0418	1765	8306	20748	742	4957	11177	2845425	—	12203	81	941	25	7	29	31	6	15	3	5.72
v0419	1626	15709	52867	650	7852	25052	2590326	—	4106	55	801	4	1	29	17	11	17	5	3.99
v0420	958	4099	10240	417	2593	6124	1696889	—	2538	47	511	4	1	28	23	8	18	5	1.31
v0421	952	1814	3119	286	1134	1437	1853951	—	2304	34	317	9	3	32	18	4	18	3	0.72
v1616	1439	67441	244727	1230	52926	199724	1006460	—	1295605	11123	177084	4811	1605	6	30	8	41	8	4219.41
v1617	1619	113655	432278	1409	85457	336147	1102586	0.02	15257970	16169	67051	15661	3571	22	46	6	10	3	7200.61[b]
v1618	1603	146715	545337	1396	90973	349947	1154458	0.13	2418105	5549	70533	1461	296	11	27	16	21	9	7222.28[b]
v1619	1612	105822	401097	1424	85696	336068	1156338	0.02	5774346	9040	124824	4203	880	15	40	15	12	10	7205.74[b]
v1620	1560	115729	444445	1365	89512	353689	1140604	—	7460098	20801	111073	19230	8161	19	29	14	11	2	5526.43
v1621	938	24772	76971	807	16683	54208	825563	—	12214	130	1415	13	5	23	20	16	22	3	13.79
v1622	859	13773	41656	736	11059	35304	793445	—	13325	99	1147	14	3	27	29	10	18	3	9.69
v1616	1439	67441	244727	1230	52926	199724	1006460	0.02	83501	828	6193	70	11	19	26	17	22	4	125.06[b]
v1617	1619	113655	432278	1409	85457	336147	1103036	0.11	48690	426	2972	20	4	14	22	27	20	3	137.51[b]
v1618	1603	146715	545337	1396	90973	349947	1156417	0.37	38494	436	2976	15	3	12	18	32	22	2	130.19[b]
v1619	1612	105822	401097	1424	85696	336068	1157851	0.22	48584	528	3228	15	3	13	17	35	21	3	146.06[b]
v1620	1560	115729	444445	1365	89512	353689	1142159	0.17	35910	377	2940	15	3	12	20	24	29	3	133.33[b]
v1621	938	24772	76971	807	16683	54208	825563	—	12214	130	1415	13	5	22	20	17	22	3	13.82
v1622	859	13773	41656	736	11059	35304	793445	—	13325	99	1147	14	3	26	29	10	19	3	9.40
21	29615	1375763	5070937	20954	952678	3625074	31117511	—	32932766	67621	583360	47774	15258	15	34	12	18	6	32405.34

[a]Time in CPU seconds on a Sun Ultra Sparc 1 Model 170E.
[b]Total time exceeds time limit because the last LP is solved before stopping.

Table 4.1: Solving Clustering Set Partitioning Problems.

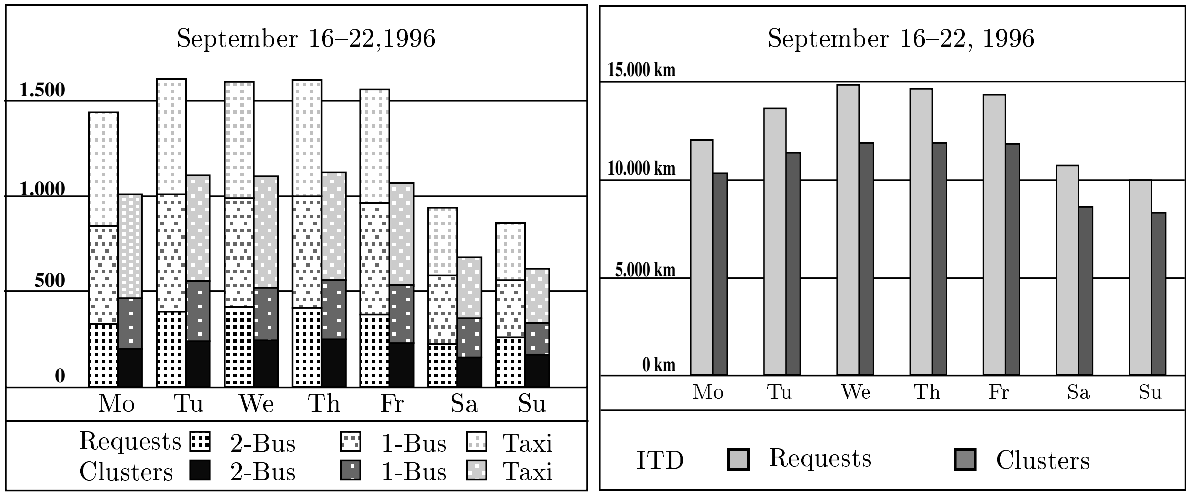


Figure 4.12: Reducing Internal Travelling Distance by Clustering.

4.7.2 Chaining

We have used the best clusterings that we computed in the tests of the previous subsection to set up two sets of chaining problems. Table 4.2 lists our results for these problems: Rows 1–7 correspond to the April instances, rows 8–14 are for the September chaining problems.

The *instances* for April contain redundant data, namely, identical rows for every request in a cluster, i.e., tours are stored by requests (not by clusters); thus, these problems have the same number of rows as their clustering relatives. This is not so for the September instances which are stored by clusters. In addition, we have also already removed from these instances all clusters that correspond to individual taxi rides: These clusters have to be serviced exactly in this way (with an individual taxi ride) and would give rise to row singletons. The number of rows in the September instances is thus exactly the sum of the heights of the columns for 2-bus and 1-bus clusters in Figure 4.12.

The picture for tour optimization has the same flavour as the clustering: A small number of tours was produced for April, more potential is present in the September data, where the average tour services between four and five clusters. Thinking about the possible success of *preprocessing*, one would guess that tours, which extend over a long period of time and a large area of service, have a significantly larger overlap than clusters, which have a local character in space and time. Hence, it is potentially much more difficult to find out about possible reductions. The real situation is even worse. Mostly only duplicate tours are eliminated in the preprocessing step. The chaining problems contain these duplicates in large numbers, because our tour generation procedures tend to produce, unfortunately, the same "locally promising" tours many times. The large reduction in the number of rows for the April instances is solely due to the removal of the duplicates that represent each cluster several times and to the detection of row singletons that correspond to individual taxi rides. These redundancies were already eliminated during the generation of the September problems, and not a single further row could be removed there.

Small reductions in preprocessing are a good indicator for the computational *hardness* of a set partitioning problem, and the chaining instances turn out to be very hard indeed. Although the problems are at best medium scale, we can solve none of them to proven optimality with our branch-and-cut code, and the duality gaps are disappointingly large in comparison

	Original Problem			Preprocessed			Solutions		Branch-and-Cut					Timings/%					Total
Name	Rows	Cols	NNEs	Rows	Cols	NNEs	$\overline{z}$	Gap/%	Pvt	Cut	Itn	LP	B&B	PP	Pvt	Cut	LP	Heu	Time[a]
t0415	1518	7254	48867	870	3312	20592	5590096	7.63	1291268	2029	93675	724	167	5	11	14	17	53	7218.94[b]
t0416	1771	9345	62703	974	3298	19692	6130217	4.05	1334745	2163	92796	641	144	5	11	14	17	54	7207.46[b]
t0417	1765	7894	54885	897	3774	24186	6043157	6.39	1614510	994	51439	316	71	6	17	6	11	60	7310.58[b]
t0418	1765	8676	66604	999	4071	29368	6550898	5.58	629332	1066	67551	399	87	3	9	12	21	56	7239.54[b]
t0419	1626	9362	64745	904	3287	19990	5916956	3.85	1891101	1235	57831	429	100	6	16	9	11	59	7251.57[b]
t0420	958	4583	27781	562	1872	10271	4276444	5.61	3989264	4440	135766	1507	362	10	16	11	11	52	7208.44[b]
t0421	952	4016	24214	557	1691	9015	4354411	5.54	4238861	4581	134126	1594	375	10	16	12	11	51	7213.44[b]
t1716	467	56865	249149	467	11952	61110	161636	24.27	1230592	886	39379	296	69	2	7	3	11	76	7212.95[b]
t1717	551	73885	325689	551	16428	85108	184692	26.61	1021307	592	27888	183	41	2	7	2	10	77	7331.93[b]
t1718	523	67796	305064	523	16310	83984	162992	22.06	982755	606	28048	203	44	2	6	3	10	78	7238.72[b]
t1719	556	72520	317391	556	15846	83893	187677	25.76	992993	404	22565	169	37	2	6	2	8	80	7281.77[b]
t1720	538	69134	310512	538	16195	84194	172752	26.35	899591	688	30360	187	38	2	6	3	11	77	7349.28[b]
t1721	357	36039	148848	357	9043	44106	127424	17.83	1965867	1921	69853	765	174	3	9	5	12	69	7243.42[b]
t1722	338	36630	133096	338	6581	33691	122472	17.96	1880505	2133	82286	856	188	3	9	6	13	68	7223.21[b]
14	13685	463999	2139548	9093	113660	609200	39981824	—	23962691	23738	933563	8269	1897	4	10	7	12	65	101531.25

[a] Time in CPU seconds on a Sun Ultra Sparc 1 Model 170E.
[b] Total time exceeds time limit because the last LP is solved before stopping.

Table 4.2: Solving Chaining Set Partitioning Problems.

to results for similar applications, most notably airline crew scheduling. For the September problems, we do not even get close to optimality. Looking at the pivoting column Pvt, we see that substantial reductions are achieved in the tree search, but this does obviously not suffice. In fact, the LPs do not only have completely fractional solutions, they are also difficult to solve: The average number of simplex pivots is well above 100, and, what one can not see from the table, the basis factorizations fill up more and more the longer the algorithm runs. Another problem is the primal LP plunging heuristic, that does not work well: An integral solution has about 100 variables at value one for 100 tours in a schedule, and if the LP solution is not strongly biased to an integral one or decomposes after a few decisions (as in clustering), it is not a good idea to search for such a solution by iteratively fixing variables.

As these results are as they are, we must unfortunately speculate now why this is so. We see three points. (i) Our current column generation process produces a set of vehicle schedules plus some variations of tours using greedy criteria. This works well as a heuristic, but it does not result in a large combinatorial variety of tours and there is no reason to believe that such tours can be combined in many ways to complete schedules. Rather the contrary seems to be the case: One can observe that the heuristics in `BC` nearly always fail in the chaining problems. If the set partitioning problems that we produce have *by construction* only few feasible solutions, it is not surprising that a branch-and-cut algorithm gets into trouble. We remark that one can not compensate this flaw with simple minded tricks like, e.g., adding tripper tours (unit columns), because these invariably lead to schedules of unacceptable costs. (ii) There are some reasons why Telebus DARPs might result in set partitioning problems that are difficult per se. In comparison to the Hoffman & Padberg [1993] airline test set, where our algorithm `BC` works well, see Chapter 3, the Telebus chaining SPPs have more rows, and the solutions have a much larger support. (iii) And maybe there is a structural difference between airline crew and bus scheduling: Marsten & Shepardson [1981] also report the computational hardness of set partitioning problems from bus (driver) scheduling applications in Helsinki.

4.7.3 Vehicle Scheduling

We can not solve the chaining SPPs of Telebus DARPs to optimality, but the approximate solutions that we can obtain are still valuable for vehicle scheduling.

Table 4.3 on the following page gives a *comparison* of different vehicle scheduling methods for the September DARPs. Column 1 lists the name of the instance, column 2 the day of the week, and column 3 the number of requests. The next three columns show the results of a *heuristic* vehicle scheduling that used the cluster and tour generators of Sections 4.5 and 4.6 as stand-alone optimization modules: There are, from left to right, the number of clusters in a heuristic clustering, its ITD, and the costs of a heuristic vehicle schedule computed from this clustering. We compare these numbers with the results of two *set partitioning* approaches. Skipping column 7 for the moment, we see in columns 8 and 9 the clustering results of Figure 4.12. Using this optimized clustering as input for the chaining heuristics results in vehicle schedules with *costs* that are reported in column 7. The final column 10 lists the costs of the vehicle schedules that we computed in Subsection 4.7.2.

These *results* indicate *substantial potential savings*. In our tests, the set partitioning clustering yields 10% less clusters than a heuristic clustering and about the same improvement in ITD. Heuristic vehicle scheduling based on such a clustering can save 5,000 DM of operation costs *per day* in comparison to the purely heuristic approach. Set partitioning based chaining can reduce costs by another 5,000 DM per day.

Name	Day	Requests	Heuristics				Set Partitioning		
			Clusters		Tours		Clusters		Tours
		#	#	ITD/km	DM	DM	#	ITD/km	DM
1616	Mo	1439	1167	10909	66525	60831	1011	10248	55792
1617	Tu	1619	1266	11870	71450	67792	1106	11291	62696
1618	We	1603	1253	12701	74851	68166	1107	11813	61119
1619	Th	1612	1276	12697	74059	68271	1121	11821	64863
1620	Fr	1560	1242	12630	71944	63345	1080	11757	61532
1621	Sa	938	748	9413	45842	47736[a]	676	8561	41638
1622	Su	859	703	8850	42782	44486[a]	620	8243	38803
$\sum$	7	9630	7655	79070	447453	420627	6721	73734	386443

[a]Scheduling anomaly due to heuristic chaining.

Table 4.3: Comparing Vehicle Schedules.

4.8 Perspectives

Telebus is an example that *mathematical programming techniques* can make a significant contribution to the solution of large-scale *transportation problems* of the real world. We mention here three further perspectives and refer the reader to Borndörfer, Grötschel & Löbel [1998] and the references therein for a broader treatment of optimization and transportation.

Telebus. We have pointed out in Section 4.2 that mathematical vehicle scheduling methods, as one factor, have translated into cost reductions and improvements in service at Telebus. And the optimization potential at Telebus is not yet depleted: At present, the BZA utilizes only the *heuristic* modules of our scheduling system. We have seen in Subsection 4.7.3 that integer programming allows for further cost reductions that have to be put into practice.

Computer Aided Scheduling. Automatic scheduling paves the way for a systematic *scenario analysis* not only at the BZA. The scheduler of the future will use software planning tools based on advanced mathematical methods to simulate, analyze, and anticipate the implications of changing operation conditions and variations in contractual obligations. Computer aided design (CAD) has replaced the drawing board, computer aided manufacturing (CAM) controls the factories — *computer aided scheduling* (CAS) for logistic systems is just another step in this direction.

Paratransit. Berlin's Telebus system of today is only a remainder of a comprehensive *paratransit* concept that was developed as a part of the seventies' efforts to revitalize the public transportation sector. The idea to reduce costs and simultaneously improve and extend service in times and areas of low traffic with *demand responsive systems* was convincing and immediately tested in a number of pilot schemes, in Germany in Friedrichshafen, in Wunstorf near Hannover, and, with a slightly different scope, in Berlin, see Figure 4.13 for the dimensions that were initially projected for Telebus. But some years later, most of these systems had either disappeared or turned into special purpose systems. And there can be no doubt that, for instance, the handicapped only used a system with advance call-in periods of initially three days because there was no other choice. The main reason for the lack of success of dial-a-ride systems seems to have been *scheduling problems*: After initial enthusiasm in every single one of these projects, the systems were virtually "killed by their own success" beyond a critical size.

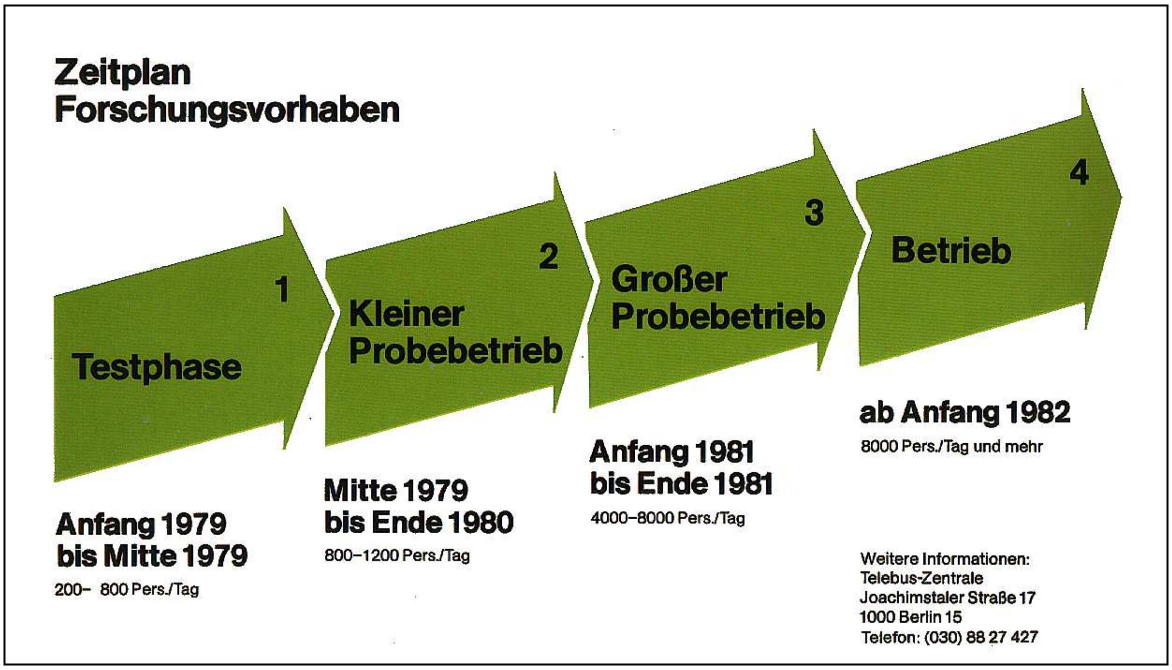


Figure 4.13: From a Telebus Project Flyer.

But *right now* the situation is changing and old reasons have kindled new interest in para-transit, see, e.g., Südmersen [1997] (German) for some examples of ongoing projects. Why? The driving force behind renewed popularity of demand responsive systems and many other developments is the upcoming *deregularization* of the European public transportation sector according to Article 90 of the *Maastricht II* treaty of the European Union, see Meyer [1997] (German) for some background information and a survey of the current situation of public transport in the EU. This law gives a new chance to dial-a-ride type systems. The future will show if mathematical programming techniques can help to take it.

Bibliography of Part 3

Assad, Ball, Bodin & Golden (1983). Routing and scheduling of vehicles and crews. *Computers & Operations Research 10(2), 63–211.*

Ball, Magnanti, Monma & Nemhauser (Eds.) (1995). *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science.* Elsevier Sci. B.V., Amsterdam.

Barnhart, Johnson, Nemhauser, Savelsbergh & Vance (1994). Branch-and-Price: Column Generation for Solving Huge Integer Programs. In Birge & Murty [1994], pp. 186–207.

Birge & Murty (Eds.) (1994). *Mathematical Programming: State of the Art 1994.* Univ. of Michigan.

Bodin & Golden (1981). Classification in Vehicle Routing and Scheduling. *Networks 11, 97–108.*

Borndörfer, Grötschel, Herzog, Klostermeier, Konsek & Küttner (1996). Kürzen muß nicht Kahlschlag heißen — Das Beispiel Telebus-Behindertenfahrdienst Berlin. Preprint SC 96-41[1], Konrad-Zuse-Zentrum Berlin. To appear in *VOP.*

Borndörfer, Grötschel, Klostermeier & Küttner (1997a). Berliner Telebus bietet Mobilität für Behinderte. *Der Nahverkehr 1–2/97, 20–22.* (ZIB Preprint SC 96-40[2]).

—— (1997b). Optimierung des Berliner Behindertenfahrdienstes. *DMV Mitteilungen 2, 38–43.* (ZIB Preprint SC 97-10[3]).

Borndörfer, Grötschel & Löbel (1998). Optimization of Transportation Systems. Preprint SC 98-09[4], Konrad-Zuse-Zentrum Berlin. To appear in *ACTA FORUM ENGELBERG 98.*

CPLEX (1995). *Using the CPLEX Callable Library*[5]. Suite 279, 930 Tahoe Blvd., Bldg 802, Incline Village, NV 89451, USA: CPLEX Optimization, Inc.

Crowder, Dembo & Mulvey (1979). On Reporting Computational Experiments with Mathematical Software. *ACM Transactions on Math. Software 5(2), 193–203.*

Cullen, Jarvis & Ratliff (1981). Set Partitioning Based Heuristics for Interactive Routing. *Networks 11, 125–143.*

Daduna & Wren (Eds.) (1988). *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag.

Dell'Amico, Maffioli & Martello (Eds.) (1997). *Annotated Bibliographies in Combinatorial Optimization.* John Wiley & Sons Ltd, Chichester.

Desrosiers, Dumas, Solomon & Soumis (1995). Time Constrained Routing and Scheduling. In Ball, Magnanti, Monma & Nemhauser [1995], chapter 2, pp. 35–139.

[1] Avail. at URL `http://www.zib.de/ZIBbib/Publications/`
[2] Avail. at URL `http://www.zib.de/ZIBbib/Publications/`
[3] Avail. at URL `http://www.zib.de/ZIBbib/Publications/`
[4] Avail. at URL `http://www.zib.de/ZIBbib/Publications/`
[5] Inf. avail. at URL `http://www.cplex.com`

Desrosiers, Dumas & Soumis (1988). The Multiple Vehicle DIAL-A-RIDE Problem. In Daduna & Wren [1988].

Hamer (1997). Fully Automated Paratransit Scheduling for Large Multi-Contractor Operations. In *Preprints of the 7th Int. Workshop on Comp.-Aided Sched. of Public Transp.*

Hoffman & Padberg (1993). Solving Airline Crew-Scheduling Problems by Branch-And-Cut. *Mgmt. Sci. 39, 657–682.*

Ioachim, Desrosiers, Dumas & Solomon (1991). A Request Clustering Algorithm in Door-to-Door Transportation. Tech. Rep. G-91-50, École des Hautes Études Commerciales de Montréal, Cahiers du GERAD.

Jackson, Boggs, Nash & Powell (1991). Guidelines for Reporting Results of Computational Experiments. Report of the Ad Hoc Committee. *Math. Prog. 49, 413–425.*

Klostermeier & Küttner (1993). Kostengünstige Disposition von Telebussen. Master's thesis, Tech. Univ. Berlin.

Laporte (1997). Vehicle Routing. In Dell'Amico, Maffioli & Martello [1997], chapter 14, pp. 223–240.

Löbel (1997). *Optimal Vehicle Scheduling in Public Transit.* PhD thesis, Tech. Univ. Berlin.

Marsten & Shepardson (1981). Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications. *Networks 11, 165–177.*

Meyer (1997). Regionalisierung und Wettbewerb. *Der Nahverkehr 5, 14–18.*

Sol (1994). *Column Generation Techniques for Pickup and Delivery Problems.* PhD thesis, Tech. Univ. Eindhoven.

Soumis (1997). Decomposition and Column Generation. In Dell'Amico, Maffioli & Martello [1997], chapter 8, pp. 115–126.

Südmersen (1997). Auf Anruf SammelTaxi. *Der Nahverkehr 9, 46–49.*

Tesch (1994). *Disposition von Anruf-Sammeltaxis.* Deutscher Univ. Verlag, Wiesbaden.

Index of Part 3

O

P

R

S

T

V

W

Z

Index of Symbols

O

P

Q

R

S

T

Curriculum Vitae

Ralf Borndörfer

geboren am 20. August 1967 in Münster

1973–1977	Grundschule in Oerlinghausen
1977–1986	Hans-Ehrenberg-Gymnasium in Bielefeld-Sennestadt
1986–1987	Grundwehrdienst bei der Panzerbrigade 21 in Augustdorf
1987–1991	Studium der Wirtschaftsmathematik an der Universität Augsburg
Dez. 1991	Diplom in Wirtschaftsmathematik mit Schwerpunkt Optimierung Betreuer: Prof. Dr. Martin Grötschel
1992–1997	Wissenschaftlicher Mitarbeiter an der Technischen Universität Berlin Betreuer: Prof. Dr. Martin Grötschel
seit April 1997	Wissenschaftlicher Mitarbeiter am Konrad-Zuse-Zentrum Berlin

Curriculum Vitae

Ralf Borndörfer

born on August 20, 1967 in Münster

1973–1977	Elementary School in Oerlinghausen
1977–1986	Hans-Ehrenberg-Gymnasium in Bielefeld-Sennestadt
1986–1987	Military Service at the Armored Brigade 21 in Augustdorf
1987–1991	Studies of Mathematics with Economics at the University of Augsburg
Dec. 1991	Master's Degree in Mathematics with Economics Supervisor: Prof. Dr. Martin Grötschel
1992–1997	Teaching Assistant at the Technical University of Berlin Supervisor: Prof. Dr. Martin Grötschel
since April 1997	Scientific Assistant at the Konrad-Zuse-Zentrum Berlin