Polyhedral techniques in combinatorial optimization

Citation for published version (APA):

Document status and date:
Published: 01/01/1995

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Memorandum COSOR 95-15

Polyhedral Techniques in Combinatorial Optimization

K. Aardal
C.P.M. van Hoesel

Eindhoven, May 1995
The Netherlands
Abstract

Combinatorial optimization problems arise in several areas ranging from management to mathematics and graph theory. Most combinatorial optimization problems are computationally hard due to the restriction that a subset of the variables have to take integral values. During the last two decades there has been a remarkable development in polyhedral techniques leading to an increase in the size of several problem types that can be solved by a factor hundred. The basic idea behind polyhedral techniques is to derive a good linear formulation of the set of solutions by identifying linear inequalities that can be proved to be necessary in the description of the convex hull of feasible solutions. The purpose of this article is to give an overview of the developments in polyhedral theory, starting with the pioneering work by Dantzig, Fulkerson and Johnson on the traveling salesman problem, and by Gomory on integer programming. We also discuss several computational aspects and implementation issues related to the use of polyhedral methods.
Combinatorial optimization deals with maximizing or minimizing a function subject to a set of constraints and subject to the restriction that some, or all, variables should be integers. Several problems that occur in management and planning situations can be formulated as combinatorial optimizations problems, such as the *lot sizing problem*, where we need to decide on which time periods to produce, and how much to produce in these periods to satisfy customers demand at minimal total production, storage and setup costs. Another well-known combinatorial optimization problem is the *traveling salesman problem* where we want to determine in which order a "salesman" has to visit a number of "cities" such that all cities are visited exactly once and such that the length of the tour is minimal. This problem is one of the most studied combinatorial optimization problems, not because of its importance in the planning of salesmen tours, but because of its numerous other applications, both in its own right and as a substructure of more complex models, and because it is notoriously difficult to solve. The combination of being easy to state, relatively easy to formulate as a mathematical programming problem, but computationally intractable is something a majority of combinatorial optimization problems have in common.

The computational intractability of most core combinatorial optimization problems has been theoretically indicated, i.e. it is possible to show that most of these problems belong to the class of *NP-hard problems*, see Karp (1972), and Garey and Johnson (1979). No algorithm with a worst-case running time bounded by a polynomial in the size of the input is known for any NP-hard problem, and it is strongly believed that no such algorithm exists. Therefore, to solve these problems we have to use an *enumerative algorithm*, such as dynamic programming or branch and bound, with a worst-case running time that is exponential in the size of the input. The computational hardness of most combinatorial optimization problems has inspired researchers to develop good formulations, and algorithms that are expected to reduce the size of the enumeration tree. To use information about the structure of the convex hull of feasible solutions, which is the basis for polyhedral techniques, has been one of the most successful approaches so far. The pioneering work in this direction was done by Dantzig, Fulkerson and Johnson (1954), who invented a method to solve the traveling salesman problem. They demonstrated the power of their technique on a 49-city instance, which was huge at that time.

The idea behind the Dantzig-Fulkerson-Johnson method is the following. Assume we want to solve the problem

\[ \min \{ cx \ subject \ to \ x \in S \}, \]  

(1)

where \( S \) is the set of feasible solutions, which in our case is the set of traveling salesman tours. Let \( S = P \cap \mathbb{Z}^n \), where \( P = \{ x \in \mathbb{R}^n : Ax \leq b \} \) and where \( Ax \leq b \) is a system of linear inequalities. Since \( S \) is difficult to characterize, we could solve the problem

\[ \min \{ cx \ subject \ to \ x \in P \} \]  

(2)

instead. Problem (2) is easy to solve, but since it is a relaxation of (1) it may give us a solution \( x^* \) that is not a tour. More precisely, the following two things can happen if we solve (2): either the optimal solution \( x^* \) is a tour, which means that \( x^* \) is also optimal for (1), or \( x^* \) is not a tour, in which case it is not feasible for (1). If the solution \( x^* \) is not feasible for (1) it lies outside the convex hull of \( S \) which means we can cut off \( x^* \) by identifying a hyperplane separating
An inequality that is satisfied by all feasible solutions is called a valid inequality. When Dantzig, Fulkerson and Johnson solved the relaxation (2) of their 49-city instance they indeed obtained a solution $x^*$ that was not a tour. By looking at the solution they identified a valid inequality that was violated by $x^*$, and added this inequality to the formulation. They solved the resulting linear programming problem and obtained again a solution that was not a tour. After repeating this process a few times a tour was obtained, and since only valid inequalities were added to the relaxation, they could conclude that the solution was optimal.

Even though many theoretical questions regarding the traveling salesman problem remained unsolved, the work of Dantzig, Fulkerson and Johnson was still a breakthrough as it provided a methodology that was actually not limited to solving traveling salesman problems, but could be applied to any combinatorial optimization problem. This new area of research on how to describe the convex hull of feasible solutions by linear inequalities was called polyhedral combinatorics. During the last decades polyhedral techniques have been used with considerable success to solve many previously unsolved instances of hard combinatorial optimization problems, and it is still the only method available for solving large instances of the traveling salesman problem. The purpose of this paper is to describe theoretical and computational aspects of polyhedral techniques and to partially survey the results that have been obtained by applying this approach.

A natural question that arises when studying the work by Dantzig, Fulkerson and Johnson is whether it is possible to develop a general scheme for identifying valid inequalities. This question was answered by Gomory (1958), (1960), (1963) who developed a cutting plane algorithm for general integer linear programming, and showed that the integer programming problem \( \min \{cx \mid x \in S\} \) can be solved by solving a finite sequence of linear programs. Chvátal (1973) proved that all inequalities necessary to describe the convex hull of integer solutions can be obtained by taking linear combinations of the original and previously generated linear inequalities and then applying a certain rounding scheme, provided that the integer solutions are bounded. Schrijver (1980) proved the more general result that it is possible to generate the convex hull of integer solutions by applying a finite set of operations on the polyhedron describing the integer solutions, if this polyhedron is rational, but not necessarily bounded. The results by Gomory, Chvátal, and Schrijver are discussed in Section 1. Here we will also address the following two questions: When can we expect to have a concise description of the convex hull of feasible solutions? How difficult is it to identify a violated inequality? These questions are strongly related to the computational complexity of the considered problem, i.e. the hardness of a problem type will catch up with us at some point, but we shall also see that certain aspects of the answers make it possible to hope that a bad situation can be turned into a rather promising one.

The results of Gomory, Chvátal and Schrijver were very important theoretically, but they did not provide direct tools for solving realistic instances within reasonable time. Researchers therefore begun to develop problem specific classes of inequalities that contain inequalities that can be proved to be necessary in the description of the convex hull of feasible solutions. Based on the various classes of valid inequalities it is then necessary to develop separation
algorithms, i.e. algorithms for identifying violated inequalities given the current solution \(z^*\). In Section 2 we begin by describing families of valid inequalities for some basic combinatorial optimization problems, and the corresponding separation problems. These inequalities are important as they are often useful when solving more complex problems as well, either directly, or as a starting point for developing new, more general families of inequalities. Moreover, they represent different arguments that can be used when developing valid inequalities. We shall also give a partial survey of polyhedral results for combinatorial optimization problems.

Next to the theoretical work of developing good classes of valid inequalities and algorithms for identifying violated inequalities, there is a whole range of implementation issues that have to be considered in order to make polyhedral methods work well. One such issue is preprocessing. Important elements of preprocessing are to reduce the size of the initial formulation by deleting unnecessary variables and constraints, and to reduce the size of the constraint coefficients to make the instance numerically more attractive. In the course of strengthening the relaxation by adding valid inequalities we may also want to delete some of the previously added inequalities to avoid that the formulation grows too much. We may also want to work with a partial set of variables to speed up computations. Dantzig, Fulkerson and Johnson were able to find the optimal solution by adding valid inequalities only. In general we do end up in the situation where the current solution \(z^*\) is not feasible and where we are unable to identify an inequality violated by \(z^*\). We then have to start a branch-and-bound phase. For the branch-and-bound algorithm we must decide precisely how to create new subproblems, or nodes, in the search tree, as well as a suitable search strategy. It is also possible to add inequalities in every node of the tree, in which case we need to keep track of where in the tree the various inequalities are valid. All these issues are discussed in Section 3. To illustrate the computational possibilities of polyhedral techniques we present computational results for some selected problem types in Section 4.

Even though polyhedral combinatorics has been the foremost tool for computing large instances of a vast collection of combinatorial optimization problems it is not the only technique available, and depending on the problem type it may be preferrable to choose a different method. We conclude our article by briefly mentioning alternative approaches to solving integer and combinatorial optimization problems.

1 Theoretical background

The integer linear programming problem (ILP) is defined as

\[
\min \{ cx : x \in S \}
\]

where \(S = P \cap \mathbb{Z}^n\) and \(P = \{ x \in \mathbb{R}^n : Ax \leq b \}\). We call \(P\) the linear formulation of ILP. A polyhedron \(P\) is rational if it can be determined by a rational system \(Ax \leq b\) of linear inequalities. The convex hull of the set \(S\) of feasible solutions, denoted \(\text{conv}(S)\), is the smallest convex set containing \(S\). A facet defining valid inequality is a valid inequality that is necessary to describe \(\text{conv}(S)\), i.e. it is the "strongest possible" valid inequality. In Figure 1 we give an example of sets \(P, S\) and \(\text{conv}(S)\).
If we know the linear description of $\text{conv}(S)$ we can solve the linear programming problem $\min \{cx : x \in \text{conv}(S)\}$ which is computationally easy. In this section we shall primarily address the issue of how difficult it is to obtain $\text{conv}(S)$. First we show that for rational polyhedra, and for not necessarily rational bounded polyhedra, we can generate $\text{conv}(S)$ algorithmically in a finite number of steps. In general however, there is no upper bound on the number of steps in terms of the dimension of $S$. We also demonstrate that it is very unlikely that $\text{conv}(S)$ of any NP-hard problem can be described by concise families of linear inequalities. Finally, we relate the complexity of the problem of finding a hyperplane separating a vector $x^*$ from $\text{conv}(S)$ or showing that $x^* \in S$, to the complexity of the optimization problem given $S$. In general these two problems are equally hard, but if we restrict the search of a separating hyperplane to a specific class, this problem might be polynomially solvable even if the underlying optimization problem is NP-hard.

### 1.1 Solving Integer Programming Problems by Linear Programming

What was needed to transform the procedure of Dantzig, Fulkerson and Johnson (1954) into an algorithm was a systematic procedure for generating valid inequalities that are violated by the current solution. Assume that we want to solve the variant of ILP where the integer vectors in $S$ are bounded and where all entries of the constraint matrix $A$ and the right-hand side vector $b$ are integers. Gomory (1958), (1960) and (1963) developed a cutting plane algorithm based on the simplex method, for solving integer linear problems on this form. This was the first algorithm developed for integer linear programming that could be proved to terminate in a finite number of iterations. The basic idea of Gomory's algorithm is similar to the approach of Dantzig, Fulkerson and Johnson, i.e. instead of solving ILP directly we solve the linear programming (LP) relaxation $\min \{cx : x \in P\}$ by the simplex method. If the optimal solution to LP is integral, then we are done, and otherwise we need to identify a valid inequality cutting off $x^*$. Gomory developed a technique for automatically identifying a violated valid inequality and proved that after adding a finite number of inequalities, called Gomory cutting planes, the optimal solution is obtained. We shall illustrate Gomory's technique by an example. Assume
we have solved the linear relaxation of an instance of ILP, as described above, by the simplex method, and that one of the rows of the tableau reads

\[ x_1 - \frac{1}{11} x_3 + \frac{2}{11} x_4 = \frac{36}{11} \]

where \( x_1 \) is a basic variable and variables \( x_3 \) and \( x_4 \) are non-basic, i.e. at the current solution \( x_1 = 36/11 \) and \( x_3 = x_4 = 0 \). We now split each coefficient in an integer and a fractional part by rounding down all coefficients. The integer terms are put in the left hand side of the equation and the fractional terms are put in the right hand side. Since all coefficients are rounded down, the fractional part of the variable coefficients in the right hand side becomes nonpositive,

\[ x_1 - x_3 - 3 = -\frac{10}{11} x_3 - \frac{2}{11} x_4 + \frac{3}{11}. \]

In any feasible solution to ILP, the left hand side should be integral. Moreover, all variables are nonnegative. Since the variables in the right hand side appear with non positive coefficients we can conclude that

\[ \frac{3}{11} - \frac{10}{11} x_3 - \frac{2}{11} x_4 \leq 0, \quad \text{and integer.} \quad (4) \]

We have argued that the inequality (4) is valid, i.e. it is not violated by any feasible integer solution. It is easy however to see that it does cut off the current fractional solution as \( x_3 = x_4 = 0 \). Let \( \lfloor x \rfloor \) denote the integer part of \( x \).

Outline of Gomory's cutting plane algorithm.

1. Solve the linear relaxation of ILP with the simplex method. The current number of variables is \( k \). If the optimal solution \( x^* \) is integral, stop.

2. Choose a source row \( i_0 \) in the optimal tableau with a fractional basic variable. Row \( i_0 \) reads \( \bar{a}_{i_0,1} x_1 + \bar{a}_{i_0,2} x_2 + \ldots + \bar{a}_{i_0,k} x_k = \bar{b}_{i_0} \). Let \( a'_{ij} = \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor \), and \( b'_1 = \bar{b}_i - \lfloor \bar{b}_i \rfloor \).

3. Add the equation \( -a'_{i_0,1} x_1 - a'_{i_0,2} x_2 - \ldots - a'_{i_0,k} x_k + x_{k+1} = -b'_{i_0} \), where \( x_{k+1} \) is a slack variable, to the current linear formulation, and reoptimize using the dual simplex method. If the optimal solution \( x^* \) is integral, stop, otherwise \( k \leftarrow k + 1 \), go to 2.

In the outline above we have not specified how to choose the source row. To be able to prove that the algorithm terminates in a finite number of steps we have to make sure that certain technical conditions are satisfied. The technical details are omitted here but can be found in Gomory (1963) who gives two proofs of finiteness, and in Schrijver (1986), page 357.

**Theorem 1** Gomory (1963). *There exists an implementation of Gomory's cutting plane algorithm such that after a finite number of iterations either an optimal integer solution is found, or it is proved that \( S = 0 \).*

A recent discussion on Gomory cutting planes can be found in Balas et al. (1994) who incorporate the cutting plane algorithm in a branch-and-bound procedure and report on computational experience.

Chvátal (1973) studied the more general version of ILP, where the integer vectors of \( S \) are bounded and where the entries of \( A \) and \( b \) are real numbers. He showed that if one takes linear
combinations of the linear inequalities defining $P$ and then applies rounding, and repeats the procedure a finite number of times, $\text{conv}(S)$ is obtained. After each iteration of the procedure we get a new linear formulation containing more inequalities. We again illustrate the procedure by an example. Let $G = (V, E)$ be an undirected graph where $V$ is the set of vertices and $E$ is the set of edges. Let $\delta(v) = \{ e \in E : e \text{ is incident to } v \}$. A matching $M$ in a graph is a subset of edges such that each vertex is incident to at most one edge in $M$. The maximum cardinality matching problem can be formulated as the following integer linear programming problem.

$$\min \sum_{e \in E} x_e$$

subject to

1. $\sum_{e \in \delta(v)} x_e \leq 1$ for all $v \in V$ (6)
2. $0 \leq x_e \leq 1$ for all $e \in E$ (7)
3. $x_e$ integer for all $e \in E$ (8)

Let $U$ be any subset consisting of $k$ vertices, where $k \geq 3$ and odd, and let $E(U)$ be the set of edges with both endvertices in $U$. By adding inequalities (6) for all $v \in U$ we obtain

$$\sum_{e \in E(U)} x_e \leq |U|,$$

or equivalently

$$\sum_{e \in E(U)} x_e \leq \frac{|U|}{2}. \quad (9)$$

Since each $x_e$ is an integer, the left-hand side of (9) has to be integral. As $|U|$ is odd, the right-hand side of (9) is fractional, and hence we can round down the right-hand side of (9) giving the valid inequality

$$\sum_{e \in E(U)} x_e \leq \left\lfloor \frac{|U|}{2} \right\rfloor \quad (10)$$

which we call an odd-set constraint. It is easy to show that the odd-set constraints are necessary to describe the convex hull of matchings in $G$. We also note that there are exponentially many odd-set constraints as there are exponentially many ways of forming subsets $U$. We shall now give a more formal description of Chvátal’s procedure.

An inequality $\sum_{j=1}^m a_j x_j \leq b$ is said to belong to the elementary closure of a set $P$ of linear inequalities, denoted $e^1(P)$, if there are inequalities $\sum_{j=1}^m a_j x_j \leq b_i$ $i = 1, \ldots, m$ in $P$ and nonnegative real numbers $\lambda_1, \lambda_2, \ldots, \lambda_m$ such that

$$\sum_{i=1}^m \lambda_i a_{ij} = a_j \text{ with } a_j \text{ integer, } j = 1, \ldots, n,$$

and

$$\left\lfloor \sum_{i=1}^m \lambda_i b_i \right\rfloor \leq b.$$

For integer values of $k > 1$, $e^k(P)$ is defined recursively as $e^k(P) = e(P \cup e^{k-1}(P))$. The closure of $P$ is defined as $c(P) = \bigcup_{k=1}^\infty e^k(P)$. 

6
Theorem 2 Chvátal (1973). If $S$ is a bounded polyhedron, then $\text{conv}(S)$ can be obtained after a finite number $k$ of closure operations.

An interesting question is if $k$ can be bounded from above by a function of the dimension of $S$. Chvátal called the minimum number of closure operations $k$ required to obtain $\text{conv}(S)$, given a linear formulation $P$, the rank of $P$. If we return to the matching problem $(6)-(8)$, it was proved by Edmonds (1965) that the convex hull of the matching polytope is determined by inequalities $(6), (7)$ and $(10)$. As the odd-set constraints $(10)$ can be obtained by applying one closure operation on the linear formulation, the rank of the set of inequalities $(6)$ and $(7)$ is one. In general however, there is no upper bound on $k$ in terms of the dimension of $S$ as the following two-dimensional problem illustrates.

\[
\begin{align*}
\text{max} & \quad x_1 + x_2 \\
\text{s.t.} & \quad x_1 - tx_1 + x_2 + tx_1 + x_2 \leq 1 + t + 1 \\
& \quad x_1, x_2 \geq 0 \\
& \quad x_1, x_2 \text{ integer}
\end{align*}
\]

Only if $S = \emptyset$ there exists an upper bound on $k$ that is a function of the dimension of $P$. This was proved by Cook et al. (1987).

There is a clear relation between Chvátal’s closure operations and Gomory’s cutting planes in the sense that every Gomory cutting plane can be obtain by a series of closure operations and every inequality belonging to the elementary closure can be obtained as a Gomory cutting plane. It would be possible to prove Theorem 2 by using Gomory’s algorithm, but then one would first need to get rid of the inequalities $x_j \geq 0, j, \ldots, n$ and the assumption that the entries of $A$ and $b$ have to be integer. For further details, see Chvátal (1973).

Schrijver (1980) studied the version of ILP where $S$ is not necessarily bounded, and where $P$ is defined by a rational system of linear inequalities. The operations carried out on $P$ to obtain the convex hull of feasible solutions is quite different from the linear combination and rounding schemes developed by Gomory and Chvátal. The key component of Schrijver’s procedure is the formulation of a totally dual integral (TDI) system of inequalities. A rational system $Ax \leq b$ of linear inequalities is TDI if for all integer vectors $c$ such that $\max\{cx : Ax \leq b\}$ is finite, the dual, $\min\{yb : yA = c, y \geq 0\}$, has an integer optimal solution. Note that if $Ax \leq b$ is TDI, and if $b$ is integral, then $P = \{x : Ax \leq b\}$ is an integral polyhedron, i.e. all extreme points of $P$ are integral. TDI systems were introduced by Edmonds and Giles (1977).

Each iteration of Schrijver’s procedure consists of the following two steps.

1. Given a rational polyhedron $P$, find a TDI system $Ax \leq b$ defining $P$, with $A$ integral.
2. Round down the righthand side $b$. 

7
It has been proved by Giles and Pulleyblank (1979) and Schrijver (1981) that there exists a TDI system as in 1. for every rational polyhedron P, and that the TDI system is unique if P is full-dimensional. Finding such a TDI system can be done in finite time. After one iteration of the above procedure we get a polyhedron \( P^{(1)} \) strictly contained in P unless P is integral. Given the polyhedron \( P^{(1)} \) we repeat the steps 1. and 2. This continues until \( \text{conv}(S) \) is obtained.

**Theorem 3** Schrijver (1980). *For each rational polyhedron P, there exists a number k, such that after k iterations of Schrijver's procedure \( \text{conv}(S) \) is obtained.*

The results presented above are of significant theoretical importance as they give algorithmic ways of generating the convex hull of feasible solutions. All three approaches are finite, but from a practical point of view finite in most cases does not imply that computations can be done within reasonable time. One apparent question is whether for some problem classes it is possible to write down the linear description of the convex hull in terms of concise families of linear inequalities. If that is possible we could apply linear programming directly. This is the topic of the following subsection.

1.2 Concise Linear Descriptions

We mentioned in the previous subsection that the convex hull of matchings in a general undirected graph G is given by the defining inequalities (6), (7) and the exponential class of inequalities (10). Assume now that G is bipartite, i.e. that we can partition the set V of vertices into two sets \( V_1, V_2 \) such that all edges have one endvertex in \( V_1 \) and the other endvertex in \( V_2 \). For bipartite graphs the convex hull of matchings is described by the defining inequalities (6) and (7) only, which is a polynomial system of linear inequalities. This means that for bipartite graphs the integrality condition (8) is redundant. In contrast, there is no concise linear description known for the traveling salesman problem, even if we allow for exponential families of inequalities. The reason why the bipartite matching problem is so easy is that the constraint matrix is totally unimodular (TU). A matrix A is TU if each subdeterminant of A is equal to 0, 1 or -1.

**Theorem 4** If A is a TU matrix the polyhedron \( P = \{ x : Ax \leq b \} \) is integral for all integer vectors b for which P is not empty.

Seymour (1980) provided a complete characterization of TU matrices yielding a polynomial algorithm for testing whether a matrix is TU. For a thorough discussion on TU matrices we refer to Schrijver (1986), and Nemhauser and Wolsey (1988).

An observation that is interesting to make in this context is that the bipartite matching problem is polynomially solvable as its linear description is polynomial in the dimension of the problem. For the matching problem in general undirected graphs there is a polynomial combinatorial algorithm due to Edmonds (1965), but the traveling salesman problem is known to be NP-hard. The following theorem confirms that there is a natural link between the computational complexity of a class of problems and the possibility of providing concise
linear descriptions of the convex hull of feasible solutions. Before stating the result we need to introduce the following decision problems:

The lower-bound feasibility problem. An instance is given by integers $m, n$, an $m \times n$ matrix $A$, vectors $b$ and $c$, and a scalar $\delta$. The question is: $\exists x \in \mathbb{Z}^n : Ax \leq b, \ cx > \delta$?

The facet validity problem. An instance is given by the same input as for the lower-bound feasibility problem. The question is: Does $cx \leq \delta$ define a facet of $\text{conv}(\{x \in \mathbb{Z}^n : Ax \leq b\})$?

Note that if the lower-bound feasibility problem for a family of polyhedra is NP-complete then optimizing over the same family of polyhedra is NP-hard.

**Lemma 5** If any NP-complete problem belongs to co-NP, then NP=co-NP.

**Theorem 6** Karp and Papadimitriou (1980). If lower-bound feasibility is NP-complete, and facet validity belongs to NP then NP=co-NP.

The way to prove Theorem 6 is to show that if facet validity belongs to NP, then lower-bound feasibility belongs to co-NP. Since lower-bound feasibility is NP-complete we can through Lemma 5 conclude that NP=co-NP. It is extremely unlikely that NP=co-NP, as this implies that all NP-complete problems have a compact certificate for the no-answer. Hence, if we believe that NP≠co-NP, and if $\min\{cx : x \in S\}$ is NP-hard then there are classes of facets of $\text{conv}(S)$ for which there is no short proof that they are facets.

### 1.3 Equivalence Between Optimization and Separation

We have seen that if a problem is NP-hard we cannot expect to have a concise linear description of the convex hull of feasible solution. Moreover, for the matching problem, which is polynomially solvable and which has a concise linear description of the convex hull of feasible solution, this description is exponential in the dimension of the problem. These observations do not necessarily have to be negative since what we primarily need is a good description of the area around the optimal solution. The question then is whether it is possible to identify a violated inequality whenever needed, i.e. if we can find a hyperplane separating a given fractional solution from the convex hull, or prove that no such hyperplane exists.

The separation problem for a family $FP$ of polyhedra. Given a polyhedron $P \in FP$, and a solution $x^*$, find an inequality $cx \leq \delta$, valid for $P$, satisfying $cx^* > \delta$, or prove that $x^* \in P$.

The optimization problem for a family $FP$ of polyhedra. Given is a polyhedron $P \in FP$. Assume that $P \neq \emptyset$ and that $P$ is bounded. Given a vector $c \in \mathbb{R}^n$, find a solution $x^0$ such that $cx^0 \leq cx$ for all $x \in P$.

**Theorem 7** Grötschel, Lovász and Schrijver (1981). There exists a polynomial time algorithm for the separation problem for a family $FP$ of polyhedra, if and only if there exists a polynomial time algorithm for the optimization problem for $FP$.

The theorem says that separation in general is equally hard as optimization but, as we shall see in the next section, when applying the polyhedral approach we develop specific families...
of valid inequalities for a given problem type, such as the odd-set constraints (10) developed for the matching problem.

The separation problem based on a family $F I$ of valid inequalities. Given a solution $x^*$, find an inequality $c x \leq \delta$ belonging to $F I$, satisfying $c x^* > \delta$, or prove that no such inequality in $F I$ exists.

The separation problem based on a family of valid inequalities may be polynomially solvable even if the underlying optimization problem is NP-hard. Moreover, even if a family of inequalities is NP-hard to separate we may still be able to separate it effectively using a heuristic. Good separation heuristics together with a good implementation of a preprocessing routine and a branch-and-bound scheme, form the basis for the success of the polyhedral approach.

2 Polyhedral Results for Selected Combinatorial Structures

The results presented in the previous section did provide very important theoretical answers, but no efficient computational tools. In the early seventies there was a renewed interest in developing general purpose integer programming solvers. Instead of Gomory's cutting plane method, which tended to be very time consuming, one developed facet defining inequalities and separation algorithms for various problem types and embedded the separation algorithms in a branch-and-bound framework. Since the added inequalities could be proved to be necessary to describe the convex hull of feasible solutions one could expect that they would be more effective than the Gomory cutting planes. Moreover, by developing facet defining inequalities and associated separation algorithms for some basic combinatorial structures that occur frequently in more general combinatorial optimization problems, and by implementing these algorithms in commercial software, it would possibly be very useful when solving a wide range of combinatorial problems. In the late seventies and in the eighties remarkable computational progress was made. Here we shall describe some classes of facet defining valid inequalities developed for a few basic, important, combinatorial optimization problems. The main purpose is to give an impression of how inequalities and separation algorithms are developed, and how they can be used, not only for the problem for which they are developed, but also for more general structures. We conclude the section by giving a list of different problem types for which polyhedral results are known, together with references. Since the space provided here is not enough for a complete survey, we recommend the following literature to the interested reader. The books by Schrijver (1986), and Nemhauser and Wolsey (1988) provide a broad theoretical foundation as well as many examples. The article by Jünger et al. (1994) contains a comprehensive survey of computational results obtained by using polyhedral techniques. The last developments on solving large traveling salesman problems is found in the article by Applegate et al. (1994).
2.1 Preliminaries

Here we introduce basic definitions that are needed to understand the terminology used in subsequent sections.

The set of linear combinations of a set of vectors \( x^1 \ldots x^K \subset \mathbb{R}^n \) is the linear space \( LS = \{ \sum_{k=1}^{K} \alpha_k x^k : \alpha \in \mathbb{R}^K \} \). If \( x^1 \ldots x^K \) form a minimal system, i.e., none of the vectors is a linear combination of the others, then the vectors \( x^1 \ldots x^K \) are called linearly independent. Equivalently, the vectors \( x^1 \ldots x^K \) are linearly independent if \( \alpha_k = 0, \forall k \) is the unique solution to the system \( \sum_{k=1}^{K} \alpha_k x^k = 0 \). The dimension of a linear space \( LS \), denoted by \( \dim(LS) \) is defined as the minimum number of linearly independent points in the space.

The set of affine combinations of the \( K + 1 \) points \( x^0, x^1 \ldots x^K \subset \mathbb{R}^n \) is called an affine space \( AS = \{ \sum_{k=0}^{K} \alpha_k x^k : \alpha \in \mathbb{R}^{K+1} ; \sum_{k=0}^{K} \alpha_k = 1 \} \). Thus, an affine space can be viewed as a linear space translated over a vector \( x^0 \): \( AS = \{ x^0 + \sum_{k=1}^{K} \beta_k (x^k - x^0) : \beta \in \mathbb{R}^K \} \). Hyperplanes in \( \mathbb{R}^n \) are affine spaces. If the set of points \( x^0 \ldots x^K \) is a minimal system, i.e., none of the points is an affine combination of the others, then the points \( x^0 \ldots x^K \) are called affinely independent. Equivalently, the points \( x^0 \ldots x^K \) are affinely independent if \( \alpha_k = 0, \forall k \) is the unique solution to the system \( \sum_{k=0}^{K} \alpha_k x^k = 0; \sum_{k=0}^{K} \alpha_k = 0 \). The dimension of an affine space, denoted by \( \dim(AS) \), is the number of affinely independent points minus 1. Thus, if the points \( x^0 \ldots x^K \) are affinely independent, the affine space defined by these points has dimension \( K \).

A polyhedron \( P \) is the set of points satisfying a system of finitely many linear inequalities, i.e., \( P = \{ x \in \mathbb{R}^n : Ax \leq b \} \). The dimension of \( P \), denoted \( \dim(P) \), is the dimension of the smallest affine space containing \( P \). A bounded polyhedron is called a polytope.

An inequality \( \pi x \leq \pi_0 \) is called valid for \( P \) if each point in \( P \) satisfies the inequality. The set \( F = \{ x \in P : \pi x = \pi_0 \} \) is called a face of \( P \) and the valid inequality \( \pi x \leq \pi_0 \) is said to define the face \( F \). A face is said to be proper if it is not empty and if it is properly contained in \( P \), i.e. if \( \emptyset \neq F \neq P \). The dimension of a proper face \( F \), \( \dim(F) \), is strictly smaller than the dimension of \( P \). If \( \dim(F) = \dim(P) - 1 \), i.e., if \( F \) is maximal, then \( F \) is called a facet. The facet defining inequalities are important since they are precisely the inequalities needed to define the convex hull of feasible solution in addition to the set of inequalities that are satisfied with equality by every feasible point.

2.2 The Vertex Packing Problem

A vertex packing is a subset \( V' \subset V \) of vertices in an undirected graph \( G = (V, E) \), such that no two vertices in \( V' \) are adjacent. Let \( x_v = 1 \) if \( v \in V' \) and let \( x_v = 0 \) otherwise. The integer programming formulation of the maximum cardinality vertex packing problem is given below.

\[
\begin{align*}
\max \quad & \sum_{v \in V} x_v \\
\text{s.t.} \quad & x_v + x_w \leq 1 \quad \text{for all } \{v, w\} \in E
\end{align*}
\]
The vertex packing problem is sometimes referred to as the independent set problem or as the stable set problem. Let \( X_{VP} \) be the set of feasible solutions to the vertex packing problem in the graph \( G \) and let \( \alpha(G) \) be the maximum cardinality of a vertex packing in \( G \). An edge is called critical if its removal from \( G \) produces a graph \( G' \) with \( \alpha(G') > \alpha(G) \). Chvátal (1975) derived the following general sufficient condition for an inequality to define a facet of \( \text{conv}(X_{VP}) \).

**Theorem 8** Chvátal (1975). Let \( E^* \) be the set of critical edges of \( G \). If the graph \( G^* = (V, E^*) \) is connected, then the inequality \( \sum_{j \in V} x_j \leq \alpha(G) \) defines a facet of \( \text{conv}(X_{VP}) \).

A clique in a graph \( G \) is a complete subgraph of \( G \), see Figure 2a). Since no two vertices in \( V' \) are allowed to be adjacent we could take any clique \( C \) in \( G \) and require that at most one vertex belonging to \( C \) should belong to the vertex packing \( V' \) giving the valid inequality

\[
\sum_{j \in C} x_j \leq 1. \tag{14}
\]

**Theorem 9** Padberg (1973). Let \( C \) be a clique in the graph \( G \). The inequality (14) defines a facet of \( \text{conv}(X_{VP}) \) if and only if \( C \) is maximal.

**Proof.** Sufficiency: The dimension of the vertex packing polytope is \( |V| \). Hence, to prove that (14) defines a facet of \( \text{conv}(X_{VP}) \) we need to find \( |V| \) affinely independent points that are tight for (14). Let \( C \) be a maximal clique. For every \( v \in C \) we take the vertex packing that contains only \( v \). For \( v \notin C \) we first choose a node \( w \in C \) that is not adjacent to \( v \). Since \( C \) is maximal such a node exists. We then take the vertex packing that contains both nodes \( v \) and \( w \). The \( |V| \) points given above are feasible and satisfy the clique inequality with equality. Thus, the inequality is facet-defining.

**Necessity:** If \( C \) is not maximal then there is a clique \( C' \) such that \( C \subset C' \). The clique inequality defined by \( C' \) dominates the inequality defined by \( C \).

Another class of valid inequalities for the vertex packing problem is the family of odd-hole inequalities. An odd hole \( H \) in a graph \( G \) is a chordless cycle consisting of an odd number of vertices, i.e. there are no edges of \( G \) connecting any nonconsecutive vertices in \( H \), see Figure 2b).
2b). Since the number of vertices in $H$ is odd, at most $|H|/2 = (|H| - 1)/2$ vertices in $H$ can belong to any vertex packing. Hence the following odd-hole inequality is valid,

$$\sum_{j \in H} x_j \leq \frac{|H| - 1}{2}. \quad (15)$$

Padberg showed that (15) defines a facet of $\text{conv}(X_{VPG} \cap \{x_j = 0 \text{ for all } j \notin H\})$, i.e. in general (15) defines a face of $\text{conv}(X_{VPG})$ of dimension less than $\dim(X_{VPG}) - 1$. The question is whether it is possible to increase the dimension of (15) such that (15) becomes a facet for $\text{conv}(X_{VPG})$. One way of increasing the dimension of a face is through sequential lifting (Padberg (1973) and Wolsey (1976)), which is illustrated in the following example. Consider the graph in Figure 3. Such a graph is called a wheel. The inequality $x_1 + x_2 + x_3 + x_4 + x_5 \leq 2$ defines a facet of $\text{conv}(X_{VPG} \cap \{x_6 = 0\})$. The question is whether there exists a constant $\alpha \geq 0$ such that $x_1 + x_2 + x_3 + x_4 + x_5 + \alpha x_6 \leq 2$ defines a facet of $\text{conv}(X_{VPG})$. If $x_6 = 0$, $\alpha$ can take any value, hence assume that $x_6 = 1$. If $x_6 = 1$ we must have $x_j = 0$, $j = 1, ..., 5$ since $x_6$ is adjacent to all other vertices. The maximal value of $\alpha$, such that the inequality remains valid, is $\alpha = 2$. In this example we had only one variable set to a fixed value, but in general we include one variable at the time, with a nonnegative coefficient, in the inequality. The following two theorems imply that if the inequality is facet defining in the reduced space, and if we “lift” in all variables sequentially with maximal coefficients, then the resulting inequality defines a facet in the full space.

**Theorem 10** Wolsey (1976). Let $S \subseteq \{0, 1\}^n$. Suppose

$$\sum_{j=2}^{n} \pi_j x_j \leq \pi_0 \quad (16)$$

is valid for $S^0 = S \cap \{x \in \{0, 1\}^n : x_1 = 0\}$. If $S \cap \{x \in \{0, 1\}^n : x_1 = 1\} \neq \emptyset$, then

$$\alpha x_1 + \sum_{j=2}^{n} \pi_j x_j \leq \pi_0 \quad (17)$$

is valid for $S$ for any $\alpha \leq \pi_0 - \max_{S \cap \{x : x_1 = 1\}} \{\sum_{j=2}^{n} \pi_j x_j\}$. If (16) defines a face of $\text{conv}(S^0)$ of dimension $k$, and if $\alpha$ is chosen maximal, then (17) defines a face of $\text{conv}(S)$ of dimension $k + 1$. 

![Figure 3: A wheel.](image-url)
Theorem 11 Wolsey (1976). Let $S \subseteq \{0,1\}^n$. Suppose
\[
\sum_{j=2}^{n} \pi_j x_j \leq \pi_0
\]  \hfill (18)
is valid for $S^1 = S \cap \{x \in \{0,1\}^n : x_1 = 1\}$. If $S \cap \{x \in \{0,1\}^n : x_1 = 0\} \neq \emptyset$, then
\[
\beta x_1 + \sum_{j=2}^{n} \pi_j x_j \leq \pi_0 + \beta_1
\]  \hfill (19)
is valid for $S$ for any $\beta \geq \max_{S \cap \{x \in \{0,1\}^n : x_1 = 0\}} \sum_{j=2}^{n} \pi_j x_j - \pi_0$. If (18) defines a face of $\text{conv}(S^1)$ of dimension $k$, and if $\beta$ is chosen minimal, then (19) defines a face of $\text{conv}(S)$ of dimension $k + 1$.

Sequential lifting is sequence dependent, i.e. different lifting sequences give rise to different inequalities. Zemel (1978) proposed an alternative lifting procedure, called simultaneous lifting. As the name indicates, the coefficients of all variables that are to be lifted are considered simultaneously, yielding inequalities that cannot be obtained in general by sequential lifting. For more details on lifting procedures, see Nemhauser and Wolsey (1988).

The separation problem for clique inequalities consists of finding a maximum weight clique in a graph. This problem is NP-hard, and therefore we usually turn to heuristics for finding violated clique inequalities. The separation problem for odd-hole inequalities can be solved in polynomial time by applying a shortest path algorithm to a slightly adapted graph, see Hoffman and Padberg (1993).

2.3 The Traveling Salesman Problem

Consider an undirected complete graph $G = (V, E)$ with $n = |V|$. In the traveling salesman problem (TSP) we want to find a minimum length Hamiltonian cycle, i.e. a minimum length cycle containing each vertex exactly once. Let $x_e = 1$ if edge $e$ is belongs to the Hamiltonian cycle, and let $x_e = 0$ otherwise. Moreover, let $d_e$ denote the length of edge $e \in E$. Usually, the vertices of the graph are called cities, and the Hamiltonian cycle is called a tour.

\[
\min \sum_{e \in E} d_e x_e \hfill (20)
\]
s.t. \[
\sum_{e : v \in e} x_e = 2 \quad \text{for all } v \in V \hfill (21)
\]
\[
\sum_{e \subseteq S} x_e \leq |S| - 1 \quad \text{for all } S : \emptyset \neq S \neq V \hfill (22)
\]
\[
x_e \in \{0,1\} \quad \text{for all } e \in E \hfill (23)
\]

The formulation restricted to the constraints (21) and (23) is called the 2-matching relaxation of TSP and its solutions are referred to as 2-matchings. Such solutions may constitute
disjoint cycles, or *subtours*. Constraints (22), introduced by Dantzig et al. (1954), prevent subtours, and are therefore called *subtour elimination constraints*. Edmonds (1965) studied the polyhedral structure of the 2-matching problem, and obtained a complete linear description of the convex hull of feasible solutions by adding so-called *2-matching inequalities* to constraints (21) and $0 \leq x_e \leq 1$ for all $e \in E$. Since the 2-matching problem is a relaxation of TSP, the 2-matching inequalities are also valid for TSP. We illustrate these inequalities in the following example. Consider the fractional solution illustrated in Figure 4. The thick

![Figure 4: A fractional solution violating a 2-matching constraint.](image)

lines correspond to variables that have value 1 and the thin lines correspond to variables with value 0.5. Clearly, this solution satisfies the degree constraints (21). To separate this solution from the convex hull of 2-matchings we introduce the following inequality. Consider the set of vertices $H = \{1, 2, 3\}$. Let $E(H)$ be the set of edges with both endvertices in $H$, and let $E' = \{(1, 4), (2, 5), (3, 6)\}$, i.e., each edge in $E'$ has exactly one endvertex in $H$. Furthermore, let $x(F) = \sum_{e \in F} x_e$. From the set of edges $E(H) \cup E'$ at most four can belong to a 2-matching since otherwise at least one of the vertices in $H$ will have degree 3, which violates constraints (21). The cumulative value of the variables corresponding to edges in $E(H) \cup E'$ is 4.5. Hence, we can conclude that the inequality $x(E(H)) + x(E') \leq 4$ is violated by the solution described above. In general, a 2-matching constraint has the form

$$x(E(H)) + x(E') \leq |H| + \left\lfloor \frac{1}{2} |E'| \right\rfloor$$

where $H \subset V$ and where the edges in $E'$ have precisely one endvertex in $H$. Note that only 2-matching constraints with an odd number of edges in $E'$, can be facet-defining, since the inequalities otherwise are implied by the degree constraints.

*Comb inequalities* were introduced by Chvátal (1975) as a generalization of the 2-matching constraints. In the comb inequalities the edges in $E'$ are replaced by an odd number $s$ of disjoint vertex sets $T_1, \ldots, T_s$, called teeth, each having one vertex in common with the handle $H$. The comb inequality is written as

$$x(E(H)) + \sum_{j=1}^{s} x(E(T_j)) \leq |H| + \sum_{j=1}^{s} (|T_j| - 1) - \frac{1}{2} (s + 1). \quad (24)$$

15
The fractional solution illustrated in Figure 5 satisfies the 2-matching constraints and the subtour elimination constraints, but not the comb inequality defined by \( H = \{1, 5, 6, 7\} \), and \( T_1 = \{1, 2\}, T_2 = \{3, 4, 5, 6\}, T_3 = \{7, 8\} \). Again, thick lines correspond to variables with value 1 and thin lines to variables with value 0.5. Chvátal’s comb inequalities were generalized by Grötschel and Padberg (1979) who introduced structures where each tooth can have more than one vertex in common with the handle. The clique tree inequalities introduced by Grötschel and Pulleyblank (1986) are further generalization of combs in the sense that clique trees contain multiple handles, which are connected through the teeth. Many more exotic classes of inequalities have been derived to date, but the search for new classes is still vivid. A good overview of the current state-of-the-art is provided by Applegate et al. (1994). Goemans (1993) considers the quality of various classes of inequalities with respect to their induced relaxations.

The separation problem based on the subtour elimination constraints can be viewed as a minimum cut problem, which is polynomially solvable using max-flow algorithms. Separation of the 2-matching constraints is also polynomial, which was showed by Padberg and Grötschel (1985). Violated 2-matching constraints are however usually identified using a heuristic, since this is still effective and faster in practice. No exact polynomial time algorithm is known to date for solving the separation problem based on the comb inequalities, but there are fast heuristic methods available that perform quite well. For clique tree inequalities, not even good heuristics are known that will perform well in general. To illustrate the effectiveness of the polyhedral approach to solve TSP we provide detailed computational results in sections 3 and 4.

![Figure 5: Fractional solution violating a comb inequality.](image-url)
2.4 The Knapsack Problem

Let $N = \{1, \ldots, n\}$. The knapsack problem is formulated as

\[
\begin{align*}
\text{max} & \quad \sum_{j \in N} c_j x_j \\
\text{s.t.} & \quad \sum_{j \in N} a_j x_j \leq b \quad \text{for all } j \in N.
\end{align*}
\]

Assume that the vectors $c, a$ and the right-hand side $b$ are rational, and let $X_K$ denote the set of feasible solutions to the knapsack problem. We call a set $C$ a cover or a dependent set with respect to $N$ if $\sum_{j \in C} a_j > b$. A cover is minimal if $\sum_{j \in S} a_j \leq b$ for all $S \subseteq C$. If we choose all elements from the cover $C$, it is clear that the right-hand side of (26) is exceeded. Hence, the following knapsack cover inequality (Balas (1975), Hammer et al. (1975) and Wolsey (1975))

\[
\sum_{j \in C} x_j \leq |C| - 1
\]

is valid. A generalization of (28) is given by the family of $(1, k)$-configuration inequalities (Padberg (1980)). Let $C \subseteq N$, and $t \in N \setminus C$ be such that $\sum_{j \in C} a_j \leq b$ and such that $Q \cup \{t\}$ is a minimal cover for all $Q \subseteq C$ with $|Q| = k$. Let $T(r) \subseteq C$ vary over all subsets of cardinality $r$ of $C$, where $r$ is an integer satisfying $k \leq r \leq |C|$. The $(1, k)$-configuration inequality

\[
(r - k + 1)x_t + \sum_{j \in T(r)} x_j \leq r
\]

is valid for conv($X_K$), and if $k = |C|$ the cover inequalities (28) are obtained. The $(1, k)$-configuration inequalities are primarily designed to deal with elements $j$ of the knapsack having a large coefficient $a_j$.

In general (28) is not facet defining, but as with the odd-hole inequalities (15) they can be lifted to become facets. One special case of a lifted cover inequality, where all lifting coefficients are equal to zero or one, is obtained by considering the extension $E(C)$ of a minimal cover $C$, where $E(C) = \{k \in N \setminus C : a_k \geq a_j, \text{ for all } j \in C\}$. The inequality $\sum_{j \in E(C)} x_j \leq |C| - 1$ is valid for conv($X_K$) and under certain conditions it also defines a facet of conv($X_K$). The most general form of the knapsack cover inequality is obtained by partitioning the set $N$ into the sets $(N', N \setminus N')$. Let $x_j = 0$ for all $j \in N \setminus N'$, and let $C'$ be a minimal cover with respect to $N'$. Moreover, let $x_j = 1$ for all $j \in N' \setminus C'$. By using the lifting results presented in Theorems 10 and 11, we can conclude that conv($X_K$) has a facet of the following form

\[
\sum_{j \in N \setminus N'} \alpha_j x_j + \sum_{j \in N \setminus C'} \beta_j x_j + \sum_{j \in C'} x_j \leq |C'| - 1 + \sum_{j \in N \setminus C'} \beta_j,
\]

where $\alpha_j \geq 0$ for all $j \in N \setminus N'$ and $\beta_j \geq 0$ for all $j \in N' \setminus C'$. Balas (1975) characterized the lifting coefficients $\alpha_j$ in the case where $N' \setminus C' = \emptyset$. 17
The separation problem based on the cover inequalities can again be viewed as a knapsack problem as we show below. Assume we are given the point \( x^* \). To find a cover inequality (28) violated by \( x^* \) we need to find a set \( C \) such that \( \sum_{j \in C} x_j^* > |C| - 1 \) and \( \sum_{j \in C} a_j > b \). Let \( z_j = 1 \) if \( j \in C \), and let \( z_j = 0 \) otherwise and assume without loss of generality that \( a_j, j \in N \) and \( b \) are integral. For (28) to be violated the \( z_j \)-variables have to satisfy the constraints

\[
\sum_{j \in N} x_j^* z_j > \left( \sum_{j \in N} x_j^* \right) - 1 \quad \text{and} \quad \sum_{j \in N} a_j z_j \geq b + 1.
\]

The first of the above constraints can be rewritten as \( \sum_{j \in N} (1 - x_j^*) z_j < 1 \), leading to the following formulation of the problem of finding the most violated cover inequality (28).

\[
\min \eta = \sum_{j \in N} (1 - x_j^*) z_j \quad \text{subject to} \quad \sum_{j \in N} a_j z_j \geq b + 1 \quad \forall j \in N
\]

A violated cover inequality is identified if and only if \( \eta < 1 \). To see that the separation problem (31)-(33) is equivalent to a knapsack problem we only need to complement the \( z_j \)-variables, i.e. substitute \( z_j \) by \( 1 - z_j \). Problem (31)-(33) is however often easier to solve than the original knapsack problem since, at a typical fractional solution \( x^* \), many variables take value zero or one. If \( x_j^* = 1 \) the coefficient of \( z_j \) in (31) is equal to zero and we can set \( z_j \) equal to one. Analogously, if \( x_j^* = 0 \) we set \( z_j \) is equal to zero. Therefore, typically few variables remain in the separation problem. Crowder et al. (1983) developed a heuristic for the separation problem and for choosing the sets \( N' \) and \( C' \). Once a minimal cover \( C' \) is generated it is also used in a heuristic for finding a violated (1, k)-configuration inequality. They implemented the algorithms and solved large 0-1 integer programming problems by automatically generating knapsack cover inequalities. Recent work on the knapsack polytope is done by Weismantel (1994).

### 2.5 The Single-Node Flow Problem

Consider a single node in a directed graph, and let \( N \) be the set of arcs entering the node. The outflow from the node is equal to \( b \). Let \( x_j \) be a continuous variable denoting the flow on arc \( j \), and let \( m_j \) be the capacity of arc \( j \). If arc \( j \) is open, then \( y_j = 1 \), otherwise \( y_j = 0 \). The following fixed charge single-node flow structure is a relaxation of many combinatorial flow models,

\[
\sum_{j \in N} x_j = b
\]

\[
0 \leq x_j \leq m_j y_j \quad \text{for all } j \in N
\]

\[
y_j \in \{0, 1\} \quad \text{for all } j \in N.
\]
Let $X_{FC}$ denote the set of feasible solutions to (34)-(36). A subset $J \subseteq N$ is called a \textit{flow cover} if $\sum_{j \in J} m_j = b + \lambda$ with $\lambda > 0$. If we have a cover $J$ and if we close one arc $k \in J$ then $\max\{x_j : j \in J \setminus k\} = \min\{b, \sum_{j \in J \setminus k} m_j\} = \min\{b, b - (m_k - \lambda)\} = b - (m_k - \lambda)^+$ yielding the valid inequality

$$\sum_{j \in J} x_j \leq b - \sum_{j \in J}(m_j - \lambda)^+(1 - y_j).$$  \hspace{1cm} (37)

\textbf{Theorem 12} Padberg, Van Roy and Wolsey (1985). \textit{The flow cover inequality (37) defines a facet of $\text{conv}(X_{FC})$ if and only if $\max_{j \in J}\{m_j\} > \lambda$.}

Let $z_j = 1$ if $j \in J$ and let $z_j = 0$ otherwise, and let $(x^*, y^*)$ denote a fractional point. For a given value of $\lambda$, the separation problem based on the family of flow cover inequalities (37) is formulated as follows.

$$\max \sum_{j \in N}[x_j^* + (m_j - \lambda)^+(1 - y_j^*)]z_j$$ \hspace{1cm} (38)

s.t. \hspace{1cm} $\sum_{j \in N} m_j z_j = b + \lambda$ \hspace{1cm} (39)

$z_j \in \{0, 1\}$ for all $j \in N$. \hspace{1cm} (40)

Once we have a set $J$ satisfying the condition of Theorem 12 we can extend the flow cover inequality by including flow from the arcs belonging to the set $L \subseteq (N \setminus J)$.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{single-node-fixed-charge-flow.png}
\caption{Single-node fixed charge flow.}
\end{figure}

Let $\bar{m}_l = \max\{\max_{j \in J}\{m_j\}, m_l\}$ for all $l \in L$. The following \textit{extended flow cover inequality} is valid for $\text{conv}(X_{FC})$,

$$\sum_{j \in J \cup L} x_j \leq b - \sum_{j \in J}(m_j - \lambda)^+(1 - y_j) + \sum_{j \in L}(\bar{m}_j - \lambda)y_j.$$ \hspace{1cm} (41)

Padberg et al. (1985) gave sufficient conditions for the extended flow cover inequality to define a facet of $\text{conv}(X_{FC})$. Aardal et al. (1993) showed that the separation problem based on the family of extended flow cover inequalities can be solved in polynomial time if $m_j = m$ for all $j \in N$.

Van Roy and Wolsey (1986) also studied the single-node flow model with both fixed charge inflow and outflow arcs as well as general uncapacitated fixed charge structures, for which they developed various families of facet defining valid inequalities. Separation heuristics for these inequalities are also discussed by Van Roy and Wolsey (1987).
2.6 An Application: The Facility Location Problem

Here we shall discuss how some of the inequalities presented above can be used, and extended, to solve facility location problems. The facility location problem is defined as follows. Let \( M = \{1, \ldots, m\} \) be the set of facilities, and let \( N = \{1, \ldots, n\} \) be the set of clients. Facility \( j \) has capacity \( m_j \), and client \( k \) has demand \( d_k \). The total demand of the clients in the set \( S \subseteq N \) is denoted by \( d(S) \). The fixed cost of opening facility \( j \) is equal to \( f_j \) and the cost of transporting one unit of goods from facility \( j \) to client \( k \) is equal to \( c_{jk} \). Let \( y_j = 1 \) if facility \( j \) is open and let \( y_j = 0 \) otherwise. The flow from facility \( j \) to client \( k \) is denoted by \( v_{jk} \). We want to determine which facility should be opened and how the flow should be distributed between the open facilities and the clients such that the sum of the fixed costs of opening the facilities, and the transportation costs is minimized, and such that all clients are served, and all capacity restrictions are satisfied. The mathematical formulation is given below.

\[
\begin{align*}
\text{min} & \quad \sum_{j \in M} f_j y_j + \sum_{j \in M} \sum_{k \in N} c_{jk} v_{jk} \\
\text{s.t.} & \quad \sum_{j \in M} v_{jk} = d_k \quad \text{for all } k \in N \quad (43) \\
& \quad \sum_{k \in N} v_{jk} \leq m_j y_j \quad \text{for all } j \in M \quad (44) \\
& \quad 0 \leq v_{jk} \leq d_k y_j \quad \text{for all } j \in M, \ k \in N \quad (45) \\
& \quad y_j \in \{0, 1\} \quad \text{for all } j \in M \quad (46)
\end{align*}
\]

2.6.1 The Uncapacitated Case

In the uncapacitated facility location (UFL) problem we have \( m_j = d(N) \) for all \( j \in M \). It is convenient to scale the flow by substituting the variables \( v_{jk} \) by the variables \( x_{jk} = v_{jk}/d_k \). The set of feasible solutions to UFL, \( X_{UFL} \), is given by the following sets of constraints.

\[
\begin{align*}
\sum_{j \in M} x_{jk} = 1 & \quad \text{for all } k \in N \quad (47) \\
0 \leq x_{jk} \leq y_j & \quad \text{for all } j \in M, \ k \in N \quad (48) \\
y_j \in \{0, 1\} & \quad \text{for all } j \in M \quad (49)
\end{align*}
\]

It is possible to require explicitly that \( x_{jk} \in \{0, 1\} \) since there is at least one optimal solution of UFL having this property. Moreover, we can change the equality sign in constraint set (47) to a less-than-or-equal-to sign if we make an appropriate change in the objective function (for more details see Cho et al. (1983)). Finally, by complementing the \( y_j \)-variables, i.e. by introducing \( y'_j = 1 - y_j \), we obtain the following vertex packing formulation of UFL.

\[
\sum_{j \in M} x_{jk} \leq 1 \quad \text{for all } k \in N \quad (50)
\]
\[ x_{jk} + y_j^k \leq 1 \quad \text{for all } j \in M, k \in N \tag{51} \]
\[ y_j^k, x_{jk} \in \{0, 1\} \quad \text{for all } j \in M, k \in N \tag{52} \]

Let \( X_{UFLVP} \) be the set of feasible solutions to (50)-(52). Given a vertex packing formulation of UFL, we can construct an associated undirected graph, called the intersection graph by introducing a vertex for every variable and an edge for every pair of nonorthogonal columns. To determine \( \text{conv}(X_{UFLVP}) \) is equivalent to determining the convex hull of vertex packings in the associated intersection graph. Hence, we can use all results described in Section 2.2 to derive valid inequalities for UFL. Due to the construction of the intersection graph all cliques in this graph are described by inequalities (50) and (51), and all odd holes are cycles where every third vertex is a \( y_j^k \)-vertex. Both Cornuéjols and Thizy (1982) and Cho et al. (1983) used the result by Chvatal given in Theorem 8 to find more general inequalities than the odd-hole inequalities. All these inequalities have a regular cyclic structure and all coefficients are equal to one for all variables except one example of a simultaneously lifted odd-hole inequality given by Cornuéjols and Thizy. Aardal and Van Hoesel (1995) discuss further use of simultaneous lifting to get new facets having different coefficients.

### 2.6.2 The Capacitated Case

By aggregating the flow from each depot we can easily see that a version of the knapsack as well as the single node flow structure form relaxations of the capacitated facility location (CFL) problem. Let \( v_j = \sum_{k \in N} v_{jk}. \) By using the aggregate flow variable \( v_j \) we can obtain the aggregate capacity and demand constraints

\[ 0 \leq v_j \leq m_j y_j \quad \text{for all } j \in M \tag{53} \]
\[ \sum_{j \in M} v_j = d(N). \tag{54} \]

If we combine constraints (53) and (54) with constraint (46) we obtain a single-node flow polynotope and a so-called surrogate knapsack polynotope \( X_{SK} = \{ y \in \{0, 1\} : \sum_{j \in M} m_j y_j \geq d(N) \} \). Complementing the \( y_j \)-variables, i.e. letting \( y_j' = 1 - y_j \) for all \( j \in M \) gives the knapsack polynotope \( \{ y' \in \{0, 1\} : \sum_{j \in M} m_j y_j' \leq \sum_{j \in M} m_j - d(N) \} \). Hence we can use both the knapsack cover inequalities as well as the flow cover inequalities when solving CFL. Both classes of inequalities can also be derived for subsets \( K \subseteq N \) of clients. Especially the cover inequalities have proved very useful computationally, as is illustrated further in Section 3.1.3. One way of generalizing the flow cover inequalities is by considering a subset of clients as well as subsets of arcs yielding the family of effective capacity inequalities (Aardal et al. (1993)). Let \( K_j \subseteq K \) for all \( j \in M \) and let \( \bar{m}_j = \min \{ m_j, d(K_j) \} \). Let \( J \) define a cover with respect to \( K \), i.e. \( \sum_{j \in J} \bar{m}_j = d(K) + \lambda \) with \( \lambda > 0 \). The effective capacity (EC) inequality

\[ \sum_{j \in J} \sum_{k \in K_j} v_{jk} \leq d(K) - \sum_{j \in J} (\bar{m}_j - \lambda)^+(1 - y_j) \tag{55} \]

is valid for \( \text{conv}(X_{CFL}) \). The facet defining EC inequalities were completely characterized by Aardal et al. (1993). To further generalize the EC inequalities consider the function \( f(J) \)
which is the maximum feasible flow from the depots in $J$ to the clients in $K$ on the arcs \(\{(j, k) : j \in J, k \in K_j\}\). By using maximum flow arguments we can show that $f(J) - f(J \setminus \{j\}) \geq (\hat{m}_j - \lambda)^+$. Hence the valid inequality

$$\sum_{j \in J} \sum_{k \in K_j} v_{jk} \leq f(J) - \sum_{j \in J} (f(J) - f(J \setminus \{j\}))(1 - y_j) \quad (56)$$

is at least as strong as the EC inequality (55). Inequalities (56) are called submodular inequalities since the function $f(J)$ is a submodular set function. Submodular inequalities were first considered by Wolsey (1989) and further developed for CFL by Aardal et al. (1993). Since there is no closed-form expression of $f(J)$ in general, it is hard to characterize the submodular facets. Aardal et al. completely characterized two subclasses for which $f(J) - f(J \setminus \{j\}) \geq (\hat{m}_j - \lambda)^+$ for at least one $j \in J$, namely the single-depot and the multi-depot inequalities. The separation problem based on the EC inequalities and the submodular inequalities are discussed by Aardal (1994).

2.7 A List of Polyhedral Results for Combinatorial Problems

Here we provide a list of polyhedral results that are known for combinatorial optimization problems. If a recent survey of results for a specific problem class is known, we refer to the survey and not to the individual articles. Surveys are marked with an asterisk. Due to the vast literature, we do not claim the list to be complete.


3 Computational Aspects

Once specific classes of valid inequalities for a certain version of ILP have been developed we can implement the separation algorithms for these inequalities in the following cutting plane algorithm.

**Outline of the cutting plane algorithm.**

1. Initialize the linear programming relaxation LP of ILP.
2. Solve LP and let $x^*$ be the optimal solution. If $x^*$ is integral, stop, otherwise go to step 3.
3. Separation algorithms are run to identify inequalities violated by $x^*$. If one or more inequalities, or cuts, have been found add them to LP and go to step 2. If no violated inequality is found, stop.

![Figure 7: Basic cutting plane algorithm.](image-url)
If the algorithm terminates by finding an integral solution \( x^* \), then \( x^* \) is provably optimal. Otherwise, the final fractional solution provides a lower bound on the optimal value, if we assume that ILP is a minimization problem. Contrary to Gomory’s cutting plane algorithm we cannot guarantee that the algorithm terminates with the optimal solution to ILP since we in general consider only a subset of all classes of facet defining inequalities, and since the separation problems are often solved heuristically. Nevertheless, this technique has proved very effective for finding at least very strong lower bounds. A good lower bound decreases the expected size of a branch-and-bound tree if we need to obtain the optimal solution. To illustrate how the lower bound develops if we add valid inequalities sequentially, we consider a TSP instance of 120 cities from Grötschel (1980), which was solved to optimality after adding cutting planes only. The optimal solution was found after 13 calls to the LP-solver. The value of the LP relaxation, \( Z_{LP} \), and the number of added cuts at each iteration, are given in Table 1. In total 36 subtour elimination constraints, 25 2-matching constraints, and 35 comb constraints were added. As can be concluded from the table, it is good practice to generate and add many violated inequalities at each iteration, since in general the computing time needed to solve the linear programs increase modestly, but the lower bound converges to the optimal value much quicker.

In the remainder of this section we shall discuss how the basic cutting plane algorithm can be extended and embedded in a branch-and-bound framework. We also discuss several implementation issues. Each extension is illustrated by an example or by tables showing computational results. In the tables we use the following notation: \( z_{LP} \) denotes the value of the LP-relaxation, and \( z_{IP} \) and \( z_{MIP} \) denote the optimal value of the integer and the mixed-integer optimization problems respectively. By \% gap we mean the percentage duality gap, i.e. \( (z_{IP} - z_{LP})/z_{IP} \). The percentage duality gap closed, denoted \% gap closed is calculated as \( (z_{IP}^{\text{root}} - z_{LP})/(z_{IP} - z_{LP}) \), where \( z_{IP}^{\text{root}} \) is the value of the LP-relaxation after all violated inequalities that have been identified in the root node of the branch-and-bound tree have been added. The number of branch-and-bound nodes needed to verify the optimal solution is given in the column \( B&B \) nodes.

<table>
<thead>
<tr>
<th>iteration</th>
<th>( z_{LP} )</th>
<th>inequalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6,662.5</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>6,883.5</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>6,912.5</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>6,918.8</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>6,928.0</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>6,935.3</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>6,937.2</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>6,939.5</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>6,940.4</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>6,940.8</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>6,941.2</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>6,941.5</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>6,942.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: A cutting plane algorithm applied to a 120-city TSP.
3.1 Extending the Cutting Plane Algorithm

There are several ways to extend the basic cutting plane algorithm. We will describe the major additional techniques in the order in which they appear in an extended cutting plane algorithm.

3.1.1 Preprocessing

Preprocessing integer linear programs involves removing redundant constraints, tightening the constraint coefficients and righthand sides of the constraints, and fixing variables to certain values. This leads not only to better lower bounds provided by the linear relaxation, but also to a significant reduction in the size of the formulation, both with respect to the number of constraints and number of variables. An important factor is also that the instance becomes numerically more tractable if large coefficients are reduced. There are many preprocessing techniques described in the literature. For each technique, or combination of techniques, one needs to find the right balance between effectiveness and computing time. Here we shall present some simple methods that strengthen a linear program quickly. These methods are described by Savelsbergh (1994), and originally developed by Crowder et al. (1983) and Hoffman and Padberg (1991).

Consider the following subset of constraints from a mixed integer program, where $N^+$ is the subset of variables with positive coefficients, $N^-$ is the subset of variables with negative coefficients, and $N = N^+ \cup N^-$.\[ \begin{align*}
\sum_{j \in N^+} a_j x_j - \sum_{j \in N^-} a_j x_j & \leq b \\
l_j & \leq x_j \leq u_j 
\end{align*} \] (57)

A lower bound on the left-hand side of (57) is $LB = \sum_{j \in N^+} a_j l_j - \sum_{j \in N^-} a_j u_j$. If $LB > b$, then the problem is infeasible. An upper bound on the left-hand side of (57) is $UB = \sum_{j \in N^+} a_j u_j - \sum_{j \in N^-} a_j l_j$. If $UB \leq b$, then the constraint is redundant. It is also possible to tighten the bounds (58) on the variables by considering one variable at the time. Consider variable $x_k$, $k \in N^+$, and let $LB_k = \sum_{j \in N^+ \setminus \{k\}} a_j l_j - \sum_{j \in N^-} a_j u_j$. Clearly, every feasible solution satisfies $x_k \leq (b - LB_k)/a_k$. Hence, the upper bound $u_k$ can be decreased if $u_k > (b - LB_k)/a_k$. Analogous results can be obtained for the lower bound $l_k$.

An elegant preprocessing technique is “probing” on the variables, which means fixing variables temporarily. Probing techniques were introduced by Guignard and Spielberg (1981). By fixing a variable we may detect logical relations between variables that can be used to tighten, and reduce the size of the formulation as is demonstrated in the following example. Consider the following set of constraints with two binary variables $x_1$ and $x_2$, and two nonnegative real variables $y_1$ and $y_2$.\[ \begin{align*}
\sum_{j \in N^+} a_j x_j - \sum_{j \in N^-} a_j x_j & \leq b \\
l_j & \leq x_j \leq u_j 
\end{align*} \] (58)
\[
\begin{align*}
2y_1 + 3y_2 & \geq 12 \\
2y_1 + y_2 & \geq 15 \\
y_1 & \leq 10x_1 \\
y_2 & \leq 20x_2
\end{align*}
\]

We probe on \(x_1\) by setting \(x_1\) equal to zero. Then, by the third constraint, \(y_1\) has to be equal to zero as well, which, due to the second and fourth constraints, implies that \(y_2 \geq 15 \) and \(x_2 = 1\).

If we consider the first constraint we see that if \(x_1 = 0\) then we can increase the right-hand side to 45. If however \(x_1 = 1\) then the right-hand side has to be equal to 12. Hence, it is possible to add the term \((45 - 12)(1 - x_1)\) to the right-hand side of the first constraint that now becomes

\[y_1 + 3y_2 \geq 12 + 33(1 - x_1)\]

Implication inequalities derived from binary variables can also be used to obtain clique constraints. In the previous example we saw that \(x_1 = 0\) implies \(x_2 = 1\). Thus, we have \(x'_i + x'_i \leq 1\), where \(x'_i, i = 1, 2\) denotes the complement of the variable of \(x_i\). To find such clique inequalities we can construct an auxiliary graph that has one vertex for every variable and its complement. Two vertices are connected by an edge if the corresponding variables cannot both have value one. Consider the auxiliary graph shown in Figure 8.

![Figure 8: Auxiliary vertex packing graph.](#)

From the structure of the graph we conclude that \(x'_1\) has to be equal to zero. To see that this is true note that \(x'_2 = 1\) implies \(x_2 = 0\). If \(x_2 = 0\) then either \(x_3 = 0\) or \(x_3 = 1\). If \(x_3 = 0\), then \(x'_3 = 1\), which implies \(x'_4 = 0\), which in turn implies that \(x_1 = 1\). This is however not feasible since \(x_1\) is adjacent to \(x'_4\). A similar contradiction is obtained if we choose \(x_3 = 1\). This example shows that by investigating logical implication we may be able to fix variables and thereby reduce the problem size. Moreover, the cliques in the auxiliary graph do in general induce inequalities that are stronger than the inequalities in the original formulation.

The effectiveness of the various preprocessing techniques has been tested by Savelsbergh (1994) on a set of 10 mixed integer programming problems from the literature. Table 2 shows the improvement of the lower bound after preprocessing as well as in the number of branch-and-bound nodes needed to verify the optimal solution. Observe that the linear programming bound increases substantially for all problems, and that the size of the branch-and-bound tree decreases quite a lot for most instances. For two instances however, the number of branch-and-bound nodes of the preprocessed problem is larger than for the original problem.
phenomenon is not really well understood. For more details regarding preprocessing we refer to Crowder et al. (1983), Hoffman and Padberg (1991) and Dietrich and Escudero (1990).

<table>
<thead>
<tr>
<th>problem</th>
<th>$z_{LP}$ without preproc.</th>
<th>$z_{LP}$ with preproc.</th>
<th>$z_{MIP}$</th>
<th>B&amp;B nodes without prepr.</th>
<th>B&amp;B nodes with prepr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>egout</td>
<td>149.5</td>
<td>562.1</td>
<td>568.1</td>
<td>553</td>
<td>3</td>
</tr>
<tr>
<td>fixnet3</td>
<td>40717.0</td>
<td>50414.2</td>
<td>51973.0</td>
<td>131</td>
<td>5</td>
</tr>
<tr>
<td>fixnet4</td>
<td>4257.9</td>
<td>7703.4</td>
<td>8936.0</td>
<td>2561</td>
<td>1031</td>
</tr>
<tr>
<td>fixnet6</td>
<td>1200.8</td>
<td>3192.5</td>
<td>3983.0</td>
<td>4795</td>
<td>4305</td>
</tr>
<tr>
<td>khb05250</td>
<td>95919464.0</td>
<td>106750366.0</td>
<td>106940226.0</td>
<td>11483</td>
<td>13</td>
</tr>
<tr>
<td>gen</td>
<td>112130.0</td>
<td>112271.0</td>
<td>112313.0</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>att</td>
<td>125.9</td>
<td>149.1</td>
<td>160.2</td>
<td>6459</td>
<td>127</td>
</tr>
<tr>
<td>sample2</td>
<td>247.0</td>
<td>290.4</td>
<td>375.0</td>
<td>336</td>
<td>51</td>
</tr>
<tr>
<td>p0033</td>
<td>2520.8</td>
<td>2838.5</td>
<td>3089.0</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>lleu</td>
<td>834.6</td>
<td>947.9</td>
<td>1120.0</td>
<td>297</td>
<td>464</td>
</tr>
</tbody>
</table>

Table 2: Effect of preprocessing techniques.

3.1.2 Postprocessing the Linear Program

After the linear program is solved, either the optimal solution is found, or, more usually, a fractional solution $x^*$ is obtained, which provides a lower bound $z_{LP}$ on the optimal value $z_{IP}$. Suppose we know a feasible solution with value $z_F$. The value $z_F$ is an upper bound on $z_{IP}$, thus $z_{IP}$ is guaranteed to lie in the interval $[z_{LP}, z_F]$. Heuristics that use the fractional solution $x^*$ to create a feasible solution are known as primal heuristics. One example of a simple primal heuristic is rounding the fractional variables to feasible integer values. Besides providing a worst case distance between the lower bound and the optimal value, an upper bound can also be used to fix variables by reduced cost fixing, or more involved, by parametric analysis on a single variable.

3.1.3 Generating Generic Inequalities

Besides the problem specific classes of valid inequalities, we can try to find violated generic inequalities. Many capacitated problems contain knapsack type constraints, in which case we may try to find violated extended knapsack cover inequalities (30). Other generic classes of valid inequalities are clique inequalities (14), obtained from the auxiliary graph of the binary variables as shown in Figure 8, and flow cover inequalities (41), obtained from variable upper bound constraints. The capacitated facility location problem provides a good insight in what these generic inequalities might offer. Table 3 shows the improvement obtained by adding extended cover inequalities to the formulation given in Section 2.6. The first five instances have 8 facilities and 25 clients, whereas the last five instances have 33 facilities and 50 clients. For more details, see Aardal (1994).
Table 3: Result of adding knapsack cover inequalities to CFL.

3.2 Embedding the Cutting Plane Algorithm in a Branch and Bound Framework

3.2.1 The Algorithm

In the early days of polyhedral techniques problems were solved by applying a cutting plane algorithm, followed by a straightforward branching process. In the mid-eighties Grötschel et al. (1984) used a cutting plane algorithm in every node of the branch-and-bound tree to solve the linear ordering problem. Padberg and Rinaldi (1987) called this idea branch and cut.

Outline of the branch-and-cut algorithm.

1. Initialize a list $L$ of subproblems of the original problem. Repeat steps 2 and 3, until $L$ is empty.

2. Select a subproblem $S$ from $L$.

3. Consider the linear relaxation of $S$ and apply a cutting plane algorithm to the relaxation. If $S$ is not solved, create new subproblems by branching. Put the new subproblems in $L$.

Every subproblem in $L$ corresponds to a node in the branch-and-cut tree. The subproblems that still need to be investigated are called active. In order to avoid complete enumeration the search tree is pruned at subproblem $j$, i.e. no further subproblems are created at node $j$, if one of the following conditions hold: a) subproblem $j$ is infeasible, b) the optimal solution to the linear relaxation of subproblem $j$ is integral, or c) $z^j_{LP} \geq \bar{z}$, where $\bar{z}$ is the best known upper bound.

In the branch-and-cut algorithm we need to specify a search strategy and a branching strategy, i.e. how to select a subproblem from the list $L$, and how to create new subproblems. The most commonly used search strategies are depth-first search and breadth-first search. In depth-first search one of the subproblems created at the current node is investigated if the current node is not pruned, whereas in breadth-first search all nodes at the current level of the
I initialize $L$

is $L$ empty? yes
no

select subproblem → process subproblem → subproblem solved?

yes no

add subproblems to $L$ → branch

Figure 9: Branch-and-cut algorithm.

tree are investigated before any node at the level below. The most frequently used branching rules are to branch on a variable according to one, or a mix, of the following four criteria. Here we assume that the variables are binary.

1. Select the variable with value closest to 0.5.
2. Select the variable with value closest to 1.
3. Select the variable with highest objective coefficient.
4. Select a set $P$ of “promising” variables and compute for each variable in $P$ the lower bound that is obtained at the corresponding subproblem. Select the variable that yields the smallest lower bound.

Padberg and Rinaldi (1991) suggest a combination of 1. and 3. for the traveling salesman problem. Rule 2. is surprisingly effective in combination with a depth-first strategy. Rule 4., introduced by Applegate et al. (1994), has similarities with the “steepest-edge” idea used in the simplex method for linear programming when choosing the variable to enter the basis. Other strategies have been proposed by Balas and Toth (1985). Jünger et al. (1992) report on computational experience with some combinations of these rules. When branching on a constraint, usually a clique constraint, a new branch is created for each value that the left-hand side of the constraint can obtain. Clochard and Naddef (1993) suggest such a rule for the traveling salesman problem.

3.2.2 Implementation Issues

The various components of the extended cutting plane algorithm may not be very effective in each node of the branch-and-cut tree. Preprocessing for instance has much effect in the
root node of the tree since the original formulation of a problem usually contains a lot of redundancy. Similarly, it may be hard to find effective cutting planes in the subproblems further down in the tree. Hence, the major effort on separation is usually put in the root node. In an implementation of a branch-and-cut algorithm we can therefore introduce selection mechanisms for where in the tree certain components should be performed. Effectiveness versus computational effort should then be weighed against each other.

As mentioned above, the search tree can be pruned at a certain node if the lower bound obtained at that node exceeds the best known upper bound. In order to decrease the expected size of the search tree it is therefore crucial to compute a good upper bound by a primal heuristic before entering the branching phase.

Branch pausing, introduced by Padberg and Rinaldi (1991), is a strategy where subproblems with high lower bounds are temporarily ignored if the lower bounds are greater than a certain threshold value. The threshold value is an estimate of the optimal value of the problem. The advantage with branch pausing is that the expected size of the search tree gets smaller. If we choose to consider subproblems in the order of increasing value of the lower bounds the implementation however gets quite complicated since subsequently chosen subproblems are not necessarily related.

Maintaining the cutting planes is a rather difficult implementation issue. In early versions of branch-and-cut packages, one was only allowed to generate globally valid inequalities, i.e., inequalities that are valid for the original problem instance. These inequalities were maintained in a central pool, from which one could select violated inequalities for the current subproblem. The global cuts usually work well, but to use the full power of the branch-and-cut algorithm, one should also be able to generate inequalities that are locally valid. Balas et al. (1994) report on very good results using branch and cut with locally valid Gomory cuts. When solving large instances it becomes important to work with a formulation that is as small as possible. One important feature is therefore to be able to delete inequalities from the active formulation and store them in a pool. A detailed overview of general implementation ideas can be found in Jünger et al. (1994). Data structures and other implementation details specific for the traveling salesman problem can be found in Applegate et al. (1994). To conclude this section we show in Figure 10 the branch-and-cut tree of a 532-city traveling salesman problem solved by Padberg and Rinaldi (1987). This tree gives an indication of the development of the lower bound at different levels of the tree.

4 Computational Results for Selected Problems

To give an idea of how polyhedral techniques perform, and how large instances can be solved, we have selected a number of problem types for which computational results are reported in the literature. For a more extensive survey we refer to Jünger et al. (1994).
Figure 10: Branch-and-cut tree for the 532-city TSP.
4.1 The Vertex Packing Problem

Nemhauser and Sigismondi (1992) report on solving randomly generated instances of the maximum cardinality vertex packing problem. The sizes of the instances vary between 40 and 120 vertices, and for every size they consider different densities by changing the probability that an edge is in the graph between 0.1 and 0.9. The code used by the authors was limited in the sense that the cutting plane algorithm was run only in the root node, and that only primitive branching rules were available. In Table 4 we report the results for the 0.2 density instances.

<table>
<thead>
<tr>
<th>vertices</th>
<th>% gap</th>
<th>clique ineq.</th>
<th>odd-hole ineq.</th>
<th>% gap</th>
<th>B&amp;B nodes</th>
<th>LP-iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>7</td>
<td>86</td>
<td>0</td>
<td>100.0</td>
<td>1</td>
<td>41</td>
</tr>
<tr>
<td>60</td>
<td>13</td>
<td>203</td>
<td>36</td>
<td>92.3</td>
<td>16</td>
<td>1439</td>
</tr>
<tr>
<td>80</td>
<td>21</td>
<td>369</td>
<td>33</td>
<td>80.9</td>
<td>97</td>
<td>13352</td>
</tr>
<tr>
<td>90</td>
<td>15</td>
<td>222</td>
<td>13</td>
<td>86.7</td>
<td>58</td>
<td>3649</td>
</tr>
<tr>
<td>100</td>
<td>29</td>
<td>181</td>
<td>19</td>
<td>93.1</td>
<td>108</td>
<td>6631</td>
</tr>
<tr>
<td>110</td>
<td>35</td>
<td>781</td>
<td>5</td>
<td>77.1</td>
<td>394</td>
<td>84115</td>
</tr>
<tr>
<td>120</td>
<td>40</td>
<td>903</td>
<td>5</td>
<td>72.5</td>
<td>251</td>
<td>35194</td>
</tr>
</tbody>
</table>

Table 4: Results for the vertex packing problem.

In general the clique inequalities closed most of the duality gap, but for low-density graphs lifted odd-holes were also important. The test instances with medium density graphs were the most difficult ones to solve. For instance, some of the medium-density 120-vertex problems were not solved within 100000 LP iterations. It seems from this study that random vertex packing problems are difficult to solve by the polyhedral approach. If we consider structured vertex packing problems however, much larger instances can be tackled as the following two applications show.

4.1.1 Frequency Assignment

The frequency assignment problem is the problem of assigning frequencies to transmission links such that no interference occurs and such that the number of used frequencies is minimized. The frequency assigned to a specific link has to be chosen from a set depending on the link. To avoid interference we have restrictions on every pair \((i, j)\) of links that the frequencies assigned to these links should differ by at least a certain prespecified amount. The problem is modelled as a vertex packing problem using a binary variable for each feasible combination of a link and a frequency. In Table 5 we present computational results as reported by Aardal et al. (1995). The number of variables is approximately equal to forty times the number of links giving instances of between approximately 4000 and 18000 variables. By making heavily use of preprocessing, the number of variables is reduced by at least fifty percent. The “lower bound by branch and bound” reported in the table is obtained by partial branching, and the time reported is the time needed to verify optimum, or, in the case of the last instance, the
<table>
<thead>
<tr>
<th>links</th>
<th>lower bound by clique ineq.</th>
<th>lower bound by B&amp;B</th>
<th>best known feasible value</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>46</td>
</tr>
<tr>
<td>200</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>1925</td>
</tr>
<tr>
<td>340</td>
<td>20</td>
<td>22</td>
<td>22</td>
<td>6167</td>
</tr>
<tr>
<td>458</td>
<td>14</td>
<td>14</td>
<td>16</td>
<td>400</td>
</tr>
</tbody>
</table>

Table 5: Results for the frequency assignment problem.

<table>
<thead>
<tr>
<th>original variables</th>
<th>constr.</th>
<th>preprocessed variables</th>
<th>constr.</th>
<th>( z_{LP} )</th>
<th>( \text{ineq.} )</th>
<th>B&amp;B ( \text{nodes} )</th>
<th>( z_{IP} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5198</td>
<td>531</td>
<td>3846</td>
<td>360</td>
<td>30494</td>
<td>0</td>
<td>1</td>
<td>30494</td>
</tr>
<tr>
<td>7292</td>
<td>646</td>
<td>5862</td>
<td>488</td>
<td>26977</td>
<td>74</td>
<td>1</td>
<td>27040</td>
</tr>
<tr>
<td>8308</td>
<td>801</td>
<td>6235</td>
<td>521</td>
<td>53736</td>
<td>345</td>
<td>5</td>
<td>53839</td>
</tr>
<tr>
<td>8627</td>
<td>825</td>
<td>6694</td>
<td>537</td>
<td>49616</td>
<td>37</td>
<td>1</td>
<td>49649</td>
</tr>
<tr>
<td>148633</td>
<td>139</td>
<td>138951</td>
<td>139</td>
<td>1181590</td>
<td>0</td>
<td>1</td>
<td>1181590</td>
</tr>
<tr>
<td>288507</td>
<td>71</td>
<td>202603</td>
<td>71</td>
<td>132878</td>
<td>0</td>
<td>1</td>
<td>132878</td>
</tr>
<tr>
<td>1053137</td>
<td>145</td>
<td>370642</td>
<td>90</td>
<td>9950</td>
<td>389</td>
<td>1</td>
<td>10022</td>
</tr>
</tbody>
</table>

Table 6: Results for the airline crew scheduling problem.

The time needed to find the feasible solution of value 16. The computations were carried out on a HP90000/720 work station.

4.1.2 The Set Partitioning Problem: Airline Crew Scheduling

Hoffman and Padberg (1993) report on solving huge set partitioning problems arising in airline crew scheduling problems. The cutting plane phase uses preprocessing techniques, and clique and lifted odd-hole inequalities. In the branch-and-cut phase a variable branching rule is used. From the reported results we have selected the instances with the largest number of variables and constraints. These results are presented in Table 6. Of the total time needed to solve the various problems, by far the longest time is spent on getting within the last percent of the optimal value. In Table 7 we show for three instances how much time it takes to get within one and two percent of the optimal value, as well as the time needed to verify optimality.

<table>
<thead>
<tr>
<th>variables</th>
<th>constraints</th>
<th>time 2% (s)</th>
<th>time 1% (s)</th>
<th>time opt (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>87482</td>
<td>36</td>
<td>225</td>
<td>298</td>
<td>2642</td>
</tr>
<tr>
<td>8904</td>
<td>823</td>
<td>375</td>
<td>375</td>
<td>14441</td>
</tr>
<tr>
<td>7195</td>
<td>426</td>
<td>868</td>
<td>7443</td>
<td>139337</td>
</tr>
</tbody>
</table>

Table 7: Time needed to get within certain percentages of the optimal value.
Table 8: Results for the traveling salesman problem.

4.2 The Traveling Salesman Problem

The literature on computational results for the traveling salesman problem is vast, and some of the results have already been shown in previous sections. To make the progress visual, we give in Table 8 a list of "world records" with respect to the size of the instances. It should be noted that there are still some small instances unsolved, which indicates that small does not necessarily imply easy, and that large is not synonymous with difficult. The instances we report on here are all Euclidean symmetric traveling salesman problems, and they arise from applications such as finding routes through actual cities, routing of drilling machines when manufacturing printed circuit boards, and x-ray crystallography. The instances can be found in the library NETLIB, see Reinelt (1991). Table 8 contains information on the number of "cities" \( n \) of the instances. For all instances a complete graph is assumed which means that the number of variables is equal to \( \frac{1}{2}n(n - 1) \). The data is obtained from the original articles, so later techniques may perform differently. For instance, for the 532-city instance we know of three different numbers reported for the total number of branch-and-cut nodes needed. To give an idea of the solution times, the 2392-city problem was solved in approximately 6 hours on a CYBER. As can be seen from Table 8, the lower bounds in the root node are very close to the optimal value which partly explains the success of cutting plane algorithms for the symmetric traveling salesman problem. When solving large instances a very advanced implementation is necessary, see Applegate et al. (1994).

4.3 General Zero-One Linear Programs

Crowder et al. (1983) present the first computational results for large-scale zero-one linear programs. The test problems are real life instances whithout any apparent structure. On a set of 10 instances they show the effects of simple preprocessing techniques, and knapsack cover and \((1, k)\)-configuration inequalities generated and added in the root node of the branch-and-bound tree. In the other nodes they use only reduced-cost fixing to eliminate variables. Their computational results are shown in Table 9.
In the last two decades there has been a remarkable development in polyhedral techniques leading to an increase in the size of many combinatorial problems that can be solved by a factor hundred. Most of the computational successes have occurred for zero-one combinatorial problems where the polytope is defined once the dimension is given, such as the traveling salesman problem. For more complex combinatorial optimization problems, and for general integer programming problems less progress has been made. Here we shall give a brief overview of other available solution techniques.

If the number of variables is large compared to the number of constraints column generation may in many cases be a good alternative. It can be viewed as a dual approach to polyhedral techniques in the sense that one aims at generating the extreme points of \( \text{conv}(S) \) rather than its facets. Instead of solving a separation problem to generate a violated inequality we need to solve the problem of finding a column, i.e. a feasible solution, that can improve the objective function. Column generation was introduced by Gilmore and Gomory (1961) to solve the cutting stock problem. Recent applications are presented by Savelsbergh (1993) and Vanderbeck and Wolsey (1994).

In Lagrangean relaxation we relax the problem by removing a subset of the constraints, different from the nonnegativity constraints. Violation of the relaxed constraints is penalized by including these constraints, with a nonnegative multiplier, in the objective function. The multipliers are then updated iteratively so as to maximize the lower bound obtained from the relaxed problem. To update the Lagrangean multipliers subgradient optimization is often used. Lagrangean relaxation was used successfully by Held and Karp (1970, 1971) to solve traveling salesman problems. For further details we refer to Geoffrion (1974), Held et al. (1974) and Fisher (1981).

Lovász and Schrijver (1991) considered 0-1 integer linear programming problems and proposed a procedure of increasing — or lifting — the dimension of the problem by introducing more variables and then projecting the extended formulation back onto the original space. From the projection step strong valid inequalities are obtained for the original problem. They

<table>
<thead>
<tr>
<th>original problem</th>
<th>preprocessing</th>
<th>cutting plane</th>
<th>B&amp;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars.</td>
<td>constr.</td>
<td>z&lt;sub&gt;LP&lt;/sub&gt;</td>
<td>nodes</td>
</tr>
<tr>
<td>33</td>
<td>16</td>
<td>2520.6</td>
<td>113</td>
</tr>
<tr>
<td>40</td>
<td>24</td>
<td>61796.5</td>
<td>21</td>
</tr>
<tr>
<td>201</td>
<td>134</td>
<td>6875.0</td>
<td>1116</td>
</tr>
<tr>
<td>282</td>
<td>242</td>
<td>176867.5</td>
<td>1862</td>
</tr>
<tr>
<td>291</td>
<td>253</td>
<td>1705.1</td>
<td>87</td>
</tr>
<tr>
<td>548</td>
<td>177</td>
<td>315.3</td>
<td>36</td>
</tr>
<tr>
<td>1550</td>
<td>94</td>
<td>1706.5</td>
<td>10</td>
</tr>
<tr>
<td>1939</td>
<td>109</td>
<td>2051.1</td>
<td>334</td>
</tr>
<tr>
<td>2655</td>
<td>147</td>
<td>6532.1</td>
<td>214</td>
</tr>
<tr>
<td>2756</td>
<td>756</td>
<td>2688.7</td>
<td>2392</td>
</tr>
</tbody>
</table>

Table 9: Results for general zero-one problems.
showed that by repeating this procedure a number of times equal to the number of variables in the original space, the convex hull of feasible solutions is obtained. At the lifting step the number of variables involved are squared and the number of constraints is increased by a factor two times the number of variables. Balas et al. (1993) developed this technique further and proved that it is sufficient to double the number of variables and constraints at the lifting step. They also related this technique to a convexification technique introduced by Balas (1979) and used this relation to develop a class of finitely converging cutting plane algorithms, called lift-and-project algorithms, for mixed 0-1 linear programming problems.

Cook et al. (1993) present an implementation of a generalized basis reduction algorithm for solving general integer programming problems. This method is based on the important theoretical result by H.W. Lenstra (1983) that the integer programming problem (3) is polynomially solvable if the number of variables is fixed. A central part of his results is to show that it is possible, in polynomial time, to find either an integral vector belonging to the bounded polyhedron \( P = \{ x \in \mathbb{R}^n : Ax \leq b \} \), or an integral direction \( d \in \mathbb{Z}^n \setminus \{0\} \) such that \( \max\{dx : x \in P\} - \min\{dx : x \in P\} \leq \gamma \) where \( \gamma \) depends on the dimension of \( P \) only. A direction \( d \) as described above is called flat. Instead of branching on variables as in conventional branch-and-bound techniques, the flat directions are used to branch on hyperplanes \( dx = t, x \in P \), where \( t \) is an integer varying between \( \lfloor \min\{dx : x \in P\} \rfloor \) and \( \lceil \max\{dx : x \in P\} \rceil \). Since the direction \( d \) is flat the number of subproblems created at each level of the search tree is limited by a constant depending only on \( n \). Moreover, we have no more than \( n \) levels in the tree. To find flat directions Cook et al. use the generalized basis reduction technique developed by Lovász and Scarf (1992).

One of the main drawbacks of polyhedral techniques, as described in Section 2, is that the separation problem based on several facet defining inequalities is hard to solve, or sometimes even hard to formulate. Boyd (1994) developed a cutting plane algorithm for general integer programming that is based on so-called Fenchel duality. The basic idea of Boyd’s method is to prove that a certain point \( \bar{x} \) belongs to \( \text{conv}(S) \) or to find a separating hyperplane, that is as far as possible from \( \bar{x} \). Such a separating hyperplane is referred to as a Fenchel cut. To find a Fenchel cut one needs to maximize a piecewise linear function on a nonlinear domain. Boyd suggests different relaxations of the nonlinear domain and reports on computational experience using these relaxations to solve the test problems of Crowder et al. (1983).

Natraj et al. (1994) used the theory of Gröbner bases to develop a solution method to solve a difficult scheduling problem. For a more general treatment of this technique we refer to Thomas (1992) and Sturmfels and Thomas (1994). The idea behind the approach by Natraj et al. is to walk from one integer solution to another in such a way that the objective function improves at every step. The directions used in this walk are specified by the Gröbner basis associated with the problem. A Gröbner basis can be viewed as a so-called test set of integral vectors \( x^1, \ldots, x^N \), depending on the constraint matrix only. These vectors have the property that a feasible solution \( x^* \) is optimal if and only if \( c(x^* + x^k) \geq cx^* \) whenever \( x^* + x^k, k = 1, \ldots, N \) is a feasible solution.
Acknowledgement

We would like to thank David Applegate for providing data on the traveling salesman problem reported in Section 4.2. Large parts of this article was written while the first author was visiting University of California at Berkeley. Financial support provided by the late Gene Lawler, and Umesh Vazirani through grant IRI-9120074 from NSF is greatly acknowledged.

References


E. BALAS (1975a) "Facets of the knapsack polytope", Mathematical Programming 8 146-164.


E. BALAS AND S.M. Ng (1989a) "On the set covering polytope: I. All the facets with coefficients in \{0, 1, 2\}" *Mathematical Programming* 43 57–69.

E. BALAS AND S.M. Ng (1989b) "On the set covering polytope: II. Lifting the facets with coefficients in \{0, 1, 2\}" *Mathematical Programming* 45 1–20.


M. Fischetti (1991b) "Facets of two Steiner arborescence polyhedra", Mathematical Programming 51 401-419.


41


P. Nobili and A. Sassano (1989) "Facets and lifting procedures for the set covering polytope", Mathematical Programming 45 111-147.


E. ZEMEL (1978) "Lifting the facets of zero-one polytopes" Mathematical Programming 15 268-277.