INTEGER PROGRAMMING[1]

by

V. Chandru[2] & M. R. Rao[3]

March 1998

Please address all correspondence to :

Dr. M. R. Rao
Indian Institute of Management Bangalore
Bannerghatta Road
Bangalore 560 076
India

Fax: (080) 6644050

---

[1]*To appear as a chapter of the Handbook of Algorithms edited by M.J. Atallah, CRC Press (1998)*

[2]*CS & Automation, Indian Institute of Science, Bangalore-560 012, India. chandru@csa.iisc.ernet.in*

[3]*Indian Institute of Management Bangalore, Bangalore 560 076, India. mrao@iimb.ernet.in*

INTEGER PROGRAMMING[4]

VIJAY CHANDRU, Indian Institute of Science, Bangalore 560 012, India.
M.R. RAO, Indian Institute of Management, Bangalore 560 076, India.
JANUARY, 1998

Abstract

Integer programming is an expressive framework for modeling and solving discrete optimization problems that arise in a variety of contexts in the engineering sciences. Integer programming representations work with implicit algebraic constraints (linear equations and inequalities on integer valued variables) to capture the feasible set of alternatives, and linear objective functions (to minimize or maximize over the feasible set) that specify the criterion for defining optimality. This algebraic approach permits certain natural extensions of the powerful methodologies of linear programming to be brought to bear on combinatorial optimization and on fundamental algorithmic questions in the geometry of numbers.

1 Introduction

In 1957 the Higgins lecturer of mathematics at Princeton, Ralph Gomory, announced that he would lecture on solving linear programs in integers. The immediate reaction he received was: "But that's impossible!". This was his first indication that others had thought about the problem [57]. Gomory went on to work on the foundations of the subject of integer programming as a scientist at IBM from 1959 until 1970 (when he took over as Director for Research). From cutting planes and the polyhedral combinatorics of corner polyhedra to group knapsack relaxations, the approaches that Gomory developed remain striking to researchers even today.

There were other pioneers in integer programming who collectively played a similar role in developing techniques for linear programming in Boolean or 0-1 variables. These efforts were directed at combinatorial optimization problems such as Routing, Scheduling, Layout and Network Design. These are generic examples of combinatorial optimization problems that often arise in computer engineering and decision support.

Unfortunately, almost all interesting generic classes of integer programming problems are $\mathcal{NP}$-Hard. The scale at which these problems arise in applications and the explosive exponential complex-

[4]Readers unfamiliar with linear programming methodology are strongly encouraged to consult the chapter on linear programming in this handbook.

ity of the search spaces preclude the use of simplistic enumeration and search techniques. Despite the worst-case intractability of integer programming, in practice we are able to solve many large problems and often enough with off-the-shelf software. Effective software for integer programming is usually problem specific and based on sophisticated algorithms that combine approximation methods with search schemes and that exploit mathematical (and not just syntactic) structure in the problem at hand.

An abstract formulation of combinatorial optimization is

$$(CO) \quad \min\{f(I) : I \in \mathcal{I}\}$$

where $\mathcal{I}$ is a collection of subsets of a finite ground set $E = \{e_1, e_2, \ldots, e_n\}$ and f is a criterion (objective) function that maps 2^E (the power set of E) to the reals. The most general form of an integer linear program is

$$(MILP) \quad \min_{\mathbf{x} \in \Re^n} \{\mathbf{cx} : A\mathbf{x} \geq \mathbf{b},\ x_j \text{ integer } \forall\, j \in J\}$$

which seeks to minimize a linear function of the decision vector $\mathbf{x}$ subject to linear inequality constraints and the requirement that a subset of the decision variables are integer valued. This model captures many variants. If $J = \{1, 2, \ldots, n\}$, we say that the integer program is *pure* and *mixed* otherwise. Linear equations and bounds on the variables can be easily accomodated in the inequality constraints. Notice that by adding in inequalities of the form $0 \leq x_j \leq 1$ for a $j \in J$ we have forced x_j to take value 0 or 1. It is such Boolean variables that help capture combinatorial optimization problems as special cases of (MILP).

The next section contains preliminaries on linear inequalities, polyhedra, linear programming and an overview of the complexity of integer programming. These are the tools we will need to analyze and solve integer programs. Section 3, is the testimony on how integer programs model combinatorial optimization problems. In addition to working a number of examples of such integer programming formulations, we shall also review formal representation theories of (Boolean) mixed integer linear programs.

With any mixed integer program we associate a `linear programming relaxation` obtained by simply ignoring the integrality restrictions on the variables. The point being, of course, that we have polynomial-time (and practical) algorithms for solving linear programs (see Chapter **Editor: Please Insert the Number of the Chapter on Linear Programming** of this handbook). Thus the linear programming relaxation of (MILP) is given by

$$(LP) \quad \min_{\mathbf{x} \in \Re^n} \{\mathbf{cx} : A\mathbf{x} \geq \mathbf{b}\}$$

The thesis underlying the integer linear programming approaches is that this linear programming relaxation retains enough of the structure of the combinatorial optimization problem, to be a

useful weak representation. In Section 4 we shall take a closer look at this thesis in that we shall encounter special structures for which this relaxation is "tight". For general integer programs, there are several alternate schemes for generating linear programming relaxations with varying qualities of approximation. A general technique for improving the quality of the linear programming relaxation is through the generation of valid inequalities or cutting planes.

The computational art of integer programming rests on useful interplays between search methodologies and algebraic relaxations. The paradigms of Branch & Bound and Branch & Cut are the two enormously effective partial enumeration schemes that have evolved at this interface. These will be discussed in Section 5. It may be noted that all general purpose integer programming software available today uses one or both of these paradigms.

A general principle, is that we often need to disaggregate integer formulations to obtain higher quality linear programming relaxations. To solve such huge linear programs we need specialized techniques of large-scale linear programming. These aspects are described in the chapter (Chapter -**Editor: Please Fill In**) on linear programming in this handbook. The reader should note that the focus in this chapter is on solving hard combinatorial optimization problems. We catalog several special structures in integer programs that lead to tight linear programming relaxations (Section 6) and hence to polynomial-time algorithms. These include structures such as network flows, matching and matroid optimization problems. Many hard problems actually have pieces of these nice structures embedded in them. Successful implementations of combinatorial optimization have always used insights from special structures to devise strategies for hard problems.

The inherent complexity of integer linear programming has led to a long-standing research program in approximation methods for these problems. Linear programming relaxation and Lagrangean relaxation (Section 6) are two general approximation schemes that have been the real workhorses of computational practice. Primal-Dual strategies and Semi-Definite relaxations (Section 7) are two recent entrants that appear to be very promising.

Pure integer programming with variables that take arbitrary integer values is a natural extension of diophantine equations in number theory. Such problems arise in the context of cryptography, dependence analysis in programs, the geometry of numbers and Presburgher arithmetic. Section 8 covers this aspect of integer programming.

We conclude the chapter with brief comments on future prospects in combinatorial optimization from the algebraic modeling perspective.

2 Preliminaries

2.1 Polyhedral Preliminaries

Polyhedral combinatorics is the study of embeddings of combinatorial structures in Euclidean space and their algebraic representations. We will make extensive use of some standard terminology from polyhedral theory. Definitions of terms not given in the brief review below can be found in [95,124].

A (convex) *polyhedron* in $\Re^n$ can be algebraically defined in two ways. The first and more straightforward definition is the *implicit* representation of a polyhedron in $\Re^n$ as the solution set to a finite system of linear inequalities in n variables. A single linear inequality $\mathbf{a}\mathbf{x} \leq a_0$; $\mathbf{a} \neq 0$ defines a *half-space* of $\Re^n$. Therefore geometrically a polyhedron is the intersection set of a finite number of half-spaces.

A *polytope* is a bounded polyhedron. Every polytope is the convex closure of a finite set of points. Given a set of points whose convex combinations generate a polytope we have an explicit or *parametric* algebraic representation of it. A *polyhedral cone* is the solution set of a system of homogeneous linear inequalities. Every (polyhedral) cone is the conical or positive closure of a finite set of vectors. These generators of the cone provide a parametric representation of the cone. And finally a polyhedron can be alternately defined as the Minkowski sum of a polytope and a cone. Moving from one representation of any of these polyhedral objects to another defines the essence of the computational burden of polyhedral combinatorics. This is particularly true if we are interested in "minimal" representations.

A set of points $\mathbf{x}^1, \ldots, \mathbf{x}^m$ is *affinely independent* if the unique solution of $\sum_{i=1}^m \lambda_i \mathbf{x}^i = \mathbf{0}$, $\sum_{i=1}^m \lambda_i = 0$ is $\lambda_i = 0$ for $i = 1, \ldots, m$. Note that the maximum number of affinely independent points in $\Re^n$ is $n+1$. A polyhedron P is of *dimension* k, $\dim P = k$, if the maximum number of affinely independent points in P is $k+1$. A polyhedron $P \subseteq \Re^n$ of dimension n is called *full-dimensional.*

An inequality $\mathbf{a}\mathbf{x} \leq a_0$ is called *valid* for a polyhedron P if it is satisfied by all $\mathbf{x}$ in P. It is called *supporting* if in addition there is an $\bar{\mathbf{x}}$ in P that satisfies $\mathbf{a}\bar{\mathbf{x}} = a_0$. A *face* of the polyhedron is the set of all $\mathbf{x}$ in P that also satisfy a valid inequality as an equality. In general, many valid inequalities might represent the same face. Faces other than P itself are called *proper.* A *facet* of P is a maximal nonempty and proper face. A facet is then a face of P with a dimension of $\dim P - 1$. A face of dimension zero, i.e. a point $\mathbf{v}$ in P that is a face by itself, is called an *extreme point* of P. The extreme points are the elements of P that cannot be expressed as a strict convex combination of two distinct points in P. For a full-dimensional polyhedron, the valid inequality representing a facet is unique up to multiplication by a positive scalar, and facet-inducing inequalities give a minimal implicit representation of the polyhedron. Extreme points, on the other hand, give rise to minimal parametric representations of polytopes.

The two fundamental problems of linear programming (which are polynomially equivalent) are:

- **Solvability** This is the problem of checking if a system of linear constraints on real (rational) variables is solvable or not. Geometrically, we have to check if a polyhedron, defined by such constraints, is nonempty.
- **Optimization** This is the problem (LP) of optimizing a linear objective function over a polyhedron described by a system of linear constraints.

Building on polarity in cones and polyhedra, duality in linear programming is a fundamental concept which is related to both the complexity of linear programming and to the design of algorithms for solvability and optimization. Here we will state the main duality results for optimization. If we take the *primal* linear program to be

$$(P) \quad \min_{\mathbf{x}\in\Re^n} \{\mathbf{cx} : A\mathbf{x} \geq \mathbf{b}\}$$

there is an associated *dual* linear program

$$(D) \quad \max_{\mathbf{y}\in\Re^m} \{\mathbf{b}^T\mathbf{y} : A^T y = \mathbf{c}^T, \mathbf{y} \geq \mathbf{0}\}$$

and the two problems satisfy

1. For any $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ feasible in (P) and (D) (i.e. they satisfy the respective constraints), we have $\mathbf{c}\hat{\mathbf{x}} \geq \mathbf{b}^T\hat{\mathbf{y}}$ **(weak duality)**. Consequently, (P) has a finite optimal solution if and only if (D) does.

2. $\mathbf{x}^*$ and $\mathbf{y}^*$ are a pair of optimal solutions for (P) and (D) respectively, if and only if $\mathbf{x}^*$ and $\mathbf{y}^*$ are feasible in (P) and (D) (i.e. they satisfy the respective constraints) **and** $\mathbf{cx}^* = \mathbf{b}^T\mathbf{y}^*$ **(strong duality)**.

3. $\mathbf{x}^*$ and $\mathbf{y}^*$ are a pair of optimal solutions for (P) and (D) respectively, if and only if $\mathbf{x}^*$ and $\mathbf{y}^*$ are feasible in (P) and (D) (i.e. they satisfy the respective constraints) **and** $(A\mathbf{x}^* - \mathbf{b})^T\mathbf{y}^* = 0$ **(complementary slackness)**.

The strong duality condition gives us a good stopping criterion for optimization algorithms. The complementary slackness condition, on the other hand gives us a constructive tool for moving from dual to primal solutions and vice-versa. The weak duality condition gives us a technique for obtaining lower bounds for minimization problems and upper bounds for maximization problems.

Note that the properties above have been stated for linear programs in a particular form. The reader should be able to check, that if for example the primal is of the form

$$(P') \quad \min_{\mathbf{x}\in\Re^n} \{\mathbf{cx} : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$$

then the corresponding dual will have the form

$$(D') \quad \max_{\mathbf{y} \in \Re^m} \{\mathbf{b}^T\mathbf{y} : A^T\mathbf{y} \leq \mathbf{c}^T\}$$

The tricks needed for seeing this is that any equation can be written as two inequalities, an unrestricted variable can be substituted by the difference of two non-negatively constrained variables and an inequality can be treated as an equality by adding a non-negatively constrained variable to the lesser side. Using these tricks, the reader could also check that duality in linear programming is involutory (i.e. the dual of the dual is the primal).

2.2 Linear Diophantine Systems

Let us first examine the simple case of solving a single equation with integer (rational) coefficients and integer variables.

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b \tag{1}$$

A classical technique is to use the Euclidean algorithm to eliminate variables. Consider the first iteration in which we compute a_{12}, and the integers δ_1 and δ_2 where

$$a_{12} = gcd(a_1, a_2) = \delta_1 a_1 + \delta_2 a_2$$

Now we have a reduced equation that is equivalent to equation (1).

$$a_{12}x_{12} + a_3x_3 + \cdots + a_nx_n = b \tag{2}$$

It is apparent that integer solutions to equation (2) are linear projections of integer solutions to equation (1). However, it is not a simple elimination of a variable as happens in the case of equations over reals. It is a projection to a space whose dimension is one less than the dimension we began with. The solution scheme reduces the equation to a univariate equation and then inverts the projection maps. All of this can be accomplished in polynomial time since the Euclidean algorithm is "good".

Solving a system of linear diophantine equations $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \in Z^n$ now only requires a matrix version of the simple scheme described above. An integer matrix, of full row rank, is said to be in *Hermite Normal Form* if it has the appearance $[L|0]$ where L is nonsingular and lower triangular with nonnegative entries satisfying the condition that the largest entry of each row is on the main diagonal. A classical result (cf. [113]) is that any integer matrix A with full row rank has a *unique* Hermite Normal Form HNF(A) = $AK = [L|0]$ where K is a square unimodular matrix (an integer matrix with determinant ± 1).

The matrix K encodes the composite effect of the elementary column operations on the matrix A needed to bring it to normal form. The elementary operations are largely defined by repeated

invocation of the Euclidean algorithm in addition to column swaps and subtractions. Polynomial-time computability of Hermite normal forms of integer matrices was first proved by Kannan and Bachem [75] using delicate and complicated analysis of the problems of intermediate swell. Subsequently, a much easier argument based on modulo arithmetic was given by Domich, Kannan and Trotter [36]. As consequences, we have that:

- *Linear Diophantine Systems can be solved in polynomial time. Assuming A has been preprocessed to have full row rank, to solve* $Ax = b, x \in Z^n$ *we first obtain* $HNF(A) = AK = [L|0]$. *The input system has a solution if and only if* $L^{-1}b$ *is integral and if so, a solution is given by* $x = K \begin{pmatrix} L^{-1}b \\ 0 \end{pmatrix}$.
- KRONECKER: (cf. [113]) $Ax = b, x \in Z^n$ *has no solution if and only if there exists a* $y \in \Re^m$ *such that* y^tA *is integral and* y^tb *is not. A certificate of unsolvability is always available from the construction described above.*
- *The solutions to a Linear Diophantine System are finitely generated. In fact, a set of generators can be found in polynomial time.* $\{x \in Z^n | Ax = b\} = \{x^0 + \sum_{i=1}^{m} \lambda_j x^j | \lambda_j \in Z\}$, $x^0 = K \begin{pmatrix} L^{-1}b \\ 0 \end{pmatrix}$ *and* $[x^1, x^2, \cdots, x^m] = K(0, I)^t$.

In summary, linear diophantine systems are a lot like linear systems over the reals (rationals). The basic theory and complexity results for variable elimination in both constraint domains are similar. This comfortable situation changes when we move on to linear *inequalities* over the integers or equivalently to non-negative solutions to linear diophantine systems.

2.3 Computational Complexity of Integer Programming

Any algorithm for integer programming is a universal polynomial-time decision procedure on a nondeterministic Turing machine. This statement is credible not only because the solvable systems of linear inequalities (with rational coefficients) over integer-valued variables describe an $\mathcal{NP}$-complete language but also because integer programs are expressive enough to capture most decision problems in $\mathcal{NP}$ via straightforward reductions. This expressiveness derives from our ability to embed sentential inference and combinatorial structures with equal ease in integer programs. We will see ample demonstration of this expressiveness in the next section. This relationship of integer programming with $\mathcal{NP}$ is akin to the relationship of linear programming with $\mathcal{P}$.

COMPLEXITY OF LINEAR INEQUALITIES:

From our earlier discussion of polyhedra, we have the following algebraic characterization of extreme points of polyhedra.

Theorem 2.1 *Given a polyhedron P, defined by* $\{x : Ax \leq b\}$, *where A is* $m \times n$, x^i *is an extreme point of P if and only if it is a face of P satisfying* $A^i x^i = b^i$ *where* $((A^i), (b^i))$ *is a submatrix of* (A, b) *and the rank of* A^i *equals n.*

Corollary: The decision problem of verifying the membership of an input string (A,b) in the language $\mathcal{L}_I = \{(A, b) : \exists x$ *such that* $Ax \leq b\}$ belongs to $\mathcal{NP}$.

Proof: It follows from the theorem that every extreme point of the polyhedron $Q = \{x : Ax \leq b\}$ is the solution of an $(n \times n)$ linear system whose coefficients come from (A, b). Therefore we can guess a polynomial length string representing an extreme point and check its membership in Q in polynomial time. □

A consequence of Farkas Lemma [45] (duality in linear programming) is that the decision problem of testing membership of input (A, b) in the language

$$\mathcal{L}_I = \{(A, b) : \exists\, x \text{ such that } Ax \leq b\}$$

is in $\mathcal{NP} \bigcap co\mathcal{NP}$. That $\mathcal{L}_I \in \mathcal{P}$, follows from algorithms for linear programming [81,77]. This is as far down the polynomial hierarchy that we can go since $\mathcal{L}_I$ is known to be $\mathcal{P}$-complete (that is, complete for $\mathcal{P}$ with respect to log-space reductions.

COMPLEXITY OF LINEAR INEQUALITIES IN INTEGER VARIABLES:

The complexity of integer programming is polynomially equivalent to the complexity of the language

$$\mathcal{L}_{IP} = \{(A, b) : \exists x \in Z \textit{ such that } Ax \leq b\}$$

We are assuming that the input coefficients in A, b are integers (rationals). It is very easy to encode Boolean satisfiability as a special case of $\mathcal{L}_{IP}$ with appropriate choice of A, b as we shall see in the next section. Hence $\mathcal{L}_{IP}$ is $\mathcal{NP}$-hard.

It remains to argue that the decision problem of verifying the membership of an input string (A, b) in the language $\mathcal{L}_{IP}$ belongs to $\mathcal{NP}$. We will have to work a little harder since integer programs may have solutions involving numbers that are large in magnitude. Unlike the case of linear programming there is no extreme point characterization for integer programs. However, since linear diophantine systems are well behaved, we are able to salvage the following technical result that plays a similar role. Let α denote the largest integer in the integer matrices A, b and let $q = \max\{m, n\}$ where A is $m \times n$.

FINITE PRECISION LEMMA [103]: If B is an $r \times n$ submatrix of A with rank $B < n$ then $\exists$ a nonzero integral vector $z = (z_1, z_2, \cdots z_n)$ in the null space of B such that $|z_j| \leq (\alpha q)^{q+1}\ \forall j$.

Repeated use of this lemma shows that if $\{x \in Z : Ax \leq b\}$ is solvable then there is a polynomial size certificate of solvability and hence that $\mathcal{L}_{IP}$ belongs to $\mathcal{NP}$.

As with any $\mathcal{NP}$-hard problem, researchers have looked for special cases that are polynomial-time solvable. The following table is a summary of the important complexity classification results in integer programming that have been obtained to date.

	Input Data	Generic Problem	Complexity
	Solvability	**Does $\exists$ an x satisfying:**	
1.	$n, m, A, \mathbf{b}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, integer	NP-Complete [16,54,7
2.	$n, m, A, \mathbf{b}$	$A\mathbf{x} \leq \mathbf{b}; \mathbf{x} \in \{0,1\}^n$	NP-Complete [79,110
3.	$n, m, A, \mathbf{b}, d(>0)$	$A\mathbf{x} \equiv \mathbf{b} \pmod{d}; \mathbf{x} \geq \mathbf{0}$, integer	P [50,6]
4.	$n, m, A, \mathbf{b}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x}$ integer	P [50,6]
5.	$n, m, A, \mathbf{b}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$	P [81,77]
6.	$n, (m = 1), (A \geq 0), (\mathbf{b} \geq 0)$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, integer	NP-Complete [110]
7.	$n, (m = 2), (A \geq 0), (\mathbf{b} \geq 0)$	$a^1\mathbf{x} \geq b_1; a^2\mathbf{x} \leq b_2; \mathbf{x} \geq \mathbf{0}$, integer	NP-Complete [76]
8.	$(n = k), m, A, \mathbf{b}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, integer	P [87]
9.	$n, (m = k), A, \mathbf{b}$	$A\mathbf{x} \leq \mathbf{b}; \mathbf{x}$ integer	P [87]
	Optimization	**Find an x that maximizes cx subject to:**	
10.	$n, m, A, \mathbf{b}, \mathbf{c}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, integer	NP-Hard [54,79]
11.	$n, m, A, \mathbf{b}, \mathbf{c}$	$A\mathbf{x} \leq \mathbf{b}; \mathbf{x} \in \{0,1\}^n$	NP-Hard [79,110]
12.	$n, (m = 1), (A \geq 0), (\mathbf{b} \geq 0), \mathbf{c}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, integer	NP-Hard [110]
13.	$n, (m = 1), (A \geq 0), (\mathbf{b} \geq 0), \mathbf{c}$	$A\mathbf{x} \leq \mathbf{b}; \mathbf{x} \in \{0,1\}^n$	NP-Hard [110]
14.	$n, m, A, \mathbf{b}, \mathbf{c}, (d_i \geq 2 \forall i)$	$A\mathbf{x} \equiv \mathbf{b} \pmod{\mathbf{d}}; \mathbf{x} \geq \mathbf{0}$, integer	NP-Hard [20]
15.	$(n = k), m, A, \mathbf{b}, \mathbf{c}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$, integer	P [87]
16.	$n, (m = k), A, \mathbf{b}, \mathbf{c}$	$A\mathbf{x} \leq \mathbf{b}; \mathbf{x}$ integer	P [87]
17.	$n, m, A, \mathbf{b}, \mathbf{c}$	$A\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}$	P [81,77]
18.	$n, m,$ (A is graphic)$^{\#}, \mathbf{b}_1, \mathbf{b}_2, \mathbf{d}_1, \mathbf{d}_2$	$\mathbf{d}_1 \leq \mathbf{x} \leq \mathbf{d}_2, \mathbf{b}_1 \leq A\mathbf{x} \leq \mathbf{b}_2, \mathbf{x} \in Z$	P [44,101]

A is a graphic matrix if it has entries from $\{0, \pm 1, \pm 2\}$ such that the sum of absolute values of the entries in any column is at most 2.

Table 1: Summary of Complexity Results

3 Integer Programming Representations

We will first discuss several examples of combinatorial optimization problems and their formulation as integer programs. Then we will review a general representation theory for integer programs that gives a formal measure of the expressiveness of this algebraic approach. We conclude this section with a representation theorem due to Benders [10] which has been very useful in solving certain large-scale combinatorial optimization problems in practice.

3.1 Formulations

Formulating decision problems as integer or mixed integer programs is often considered an art form. However, there are a few basic principles which can be used by a novice to get started. As in all art forms though, principles can be violated to creative effect. We list below a number of example formulations, the first few of which may be viewed as principles for translating logical conditions into models.

1. DISCRETE CHOICE :

CONDITION	MODEL
$X \in \{s_1, s_2, ..., s_p\}$	$X = \sum_{j=1}^{p} s_j \, \delta_j$
	$\sum_{j=1}^{p} \delta_j = 1, \; \delta_j = 0 \; or \; 1 \; \forall j$

2. DICHOTOMY :

CONDITION	MODEL	
$g(\mathrm{x}) \geq 0$	$g(\mathrm{x}) \geq \delta \, \underline{\mathrm{g}}$	$\underline{\mathrm{g}}$ and $\underline{\mathrm{h}}$ are
or		finite lower
$h(\mathrm{x}) \geq 0$	$h(\mathrm{x}) \geq (1 - \delta) \, \underline{\mathrm{h}}$	bounds on g, h
or both	$\delta = 0 \; or \; 1$	*respectively.*

3. K-FOLD ALTERNATIVES :

condition	*model*
at least k of	$g_i(\mathrm{x}) \geq \delta_i \, \underline{\mathrm{g}}_i \;\; i = 1, ..., m$
$g_i(\mathrm{x}) \geq 0, i = 1, ..., m$	$\sum_{i=1}^{m} \delta_i \leq m - k$
must hold	$\delta_i = 0 \; or \; 1$

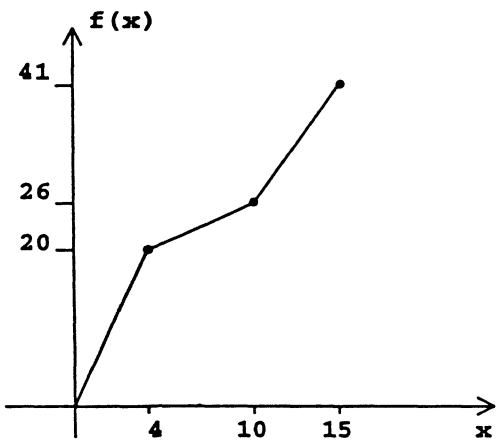
4. CONDITIONAL CONSTRAINTS :

condition	*model*	
$(f(\mathbf{x}) > 0 \Rightarrow g(\mathbf{x}) \geq 0)$	$g(\mathrm{x}) \geq \delta \mathrm{g}$	$\bar{f}/\mathrm{g}$ is an
$\Updownarrow$	$f(\mathrm{x}) \leq (1-\delta)\,\bar{f}$	upper/lower bound
$(f(\mathbf{x}) \leq 0$ *or* $g(\mathbf{x}) \geq 0$ *or both* $)$	$\delta = 0$ *or* 1	*on* f/g

5. FIXED CHARGE MODELS:

condition	*model*	
$f(x) = 0\ if\ x = 0$	$f(x) = Ky + cx$	
$f(x) = K + cx\ if\ x > 0$	$x \leq Uy$	U an upper bound
	$x \geq 0$	on x
	$y = 0$ *or* 1	

6. PIECEWISE LINEAR MODELS:



condition	*model*	
	$f(x) =$	$5\delta_1 + \delta_2 + 3\delta_3$
$\delta_2 > 0 \Rightarrow \delta_1 = 4$		$4W_1 \leq \delta_1 \leq 4$
$\delta_3 > 0 \Rightarrow \delta_2 = 6$		$6W_2 \leq \delta_2 \leq 6W_1$
		$0 \leq \delta_3 \leq 5W_2$
		$W_1, W_2 = 0\ or 1$

Figure 1: A Polyline

7. CAPACITATED PLANT LOCATION MODEL :

$i =$	$\{1, 2, ..., m\}$	possible locations for plants
$j =$	$\{1, 2, ..., n\}$	demand sites
$k_i =$	capacity of plant i, if opened	
$f_i =$	fixed cost of opening plant i	
$c_{ij} =$	per unit production cost at i plus transportion cost from i to j	
$d_j =$	demand at location j	

Choose plant locations so as to minimize total cost and meet all demands.

Formulation:

$$\begin{aligned} min \sum_i \sum_j c_{ij}\, x_{ij} \;+\; & \sum_i f_i\, y_i \\ s.t.\ \sum_i x_{ij} \;\geq\; & d_j \;\forall j \\ \sum_j x_{ij} \;\leq\; & k_i\, y_i \;\forall i \\ x_{ij} \;\geq\; & 0 \;\; \forall\, i,\, j \\ y_i \;=\; & 0 \; or \; 1 \;\forall i \end{aligned}$$

If the demand d_j is less than the capacity k_i for some ij combination, it is useful to add the constraint $x_{ij} \leq d_j y_i$ to improve the quality of the linear programming relaxation.

8. TRAVELING SALESMAN PROBLEM (ALTERNATE FORMULATIONS): A recurring theme in integer programming is that the same decision problem can be formulated in several different ways. Principles for sorting out the better ones have been the subject of some discourse [121]. We now illustrate this with the well known traveling salesman problem. Given a complete directed graph D(N,A) with distance c_{ij} of arc (i,j), we are to find the minimum length tour beginning at node 1 and visiting each node of D(N,A) exactly once before returning to the start node 1.

Formulation 1 :

$$\begin{aligned} \min \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} & \\ s.t.\ \textstyle\sum_{j=1}^{n} x_{ij} \;=\; & 1 \;\forall i \\ \textstyle\sum_{i=1}^{n} x_{ij} \;=\; & 1 \;\forall j \\ \textstyle\sum_{i\in\phi} \sum_{j\in\phi} x_{ij} \;\geq\; & 1 \;\forall\, \phi \subset N \\ x_{ij} \;=\; & 0 \; or \; 1 \;\; \forall\, i\, j \end{aligned}$$

Formulation 2 :

$$\begin{aligned} \min \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} & \\ s.t.\ \textstyle\sum_{j=1}^{n} x_{ij} \;=\; & 1 \;\forall i \\ \textstyle\sum_{i=1}^{n} x_{ij} \;=\; & 1 \;\forall j \\ \textstyle\sum_{j=1}^{n} y_{ji} - \sum_{j=2}^{n} y_{ij} \;=\; & 1 \;\forall i \neq 1 \\ y_{ij} \;\leq\; & (n-1)\, x_{ij} \;\; i = 1,2,...,n \\ & \qquad j = 2,...,n \\ x_{ij} \;=\; & 0 \; or \; 1 \;\; \forall ij \\ y_{ij} \;\geq\; & 0 \;\; \forall ij \end{aligned}$$

9. NONLINEAR FORMULATIONS: If we allow arbitrary nonlinear objective and constraint functions the general integer programming problem is closely related to Hilbert's tenth problem and is undecidable [69]. However, when the integer variables are restricted to $0-1$, the problem is

of the form

$$(NIP) \quad \min\{f(\mathrm{x}) \mid g_i(\mathrm{x}) \leq 0,\ i = 1, \ldots m\ , \mathrm{x} \in \{0,1\}^n\}$$

and we can capture a rich variety of decision problems (Modular Design, Cluster Analysis, Capital Budgeting under Uncertainty and Production Planning in Flexible Manufacturing Systems, to name a few). Restricting the constraints to linear assignment constraints while allowing quadratic cost functions yields the Quadratic Assignment Problem (QAP)

$$\begin{aligned} \min \textstyle\sum_{i \neq j} \sum_{k \neq l} c_{ijkl}\, x_{ik}\, x_{jl} & \\ s.t.\ \textstyle\sum_i x_{ik} &= 1 \ \forall k \\ \textstyle\sum_k x_{ik} &= 1 \ \forall i \\ x_{ik} &= 0 \text{ or } 1 \ \forall ik \end{aligned}$$

which includes the Travelling Salesman Problem and Plant Location Problems as special cases. Karmarkar [78] has advocated solving integer programs, by first formulating them as minimizing an indefinite quadratic function over a polyhedral region, and then solving the continuous model using interior point methods.

The other side of the coin is that these problems are extremely hard to solve and the most successful strategies to date for these problems are via linearization techniques and semi-definite relaxations.

10. COVERING AND PACKING PROBLEMS: A wide variety of location and scheduling problems can be formulated as set covering or set packing or set partitioning problems. The three different types of covering and packing problems can be succinctly stated as follows: Given

    (a) a finite set of elements $\mathcal{M} = \{1, 2, .., m\}$, and

    (b) a family F of subsets of $\mathcal{M}$ with each member F_j, $j = 1, 2, .., n$ having a profit (or cost) c_j associated with it,

    find a collection, S, of the members of F that maximizes the profit (or minimizes the cost) while ensuring that every element of $\mathcal{M}$ is in

    (P1): at most one member of S (set packing problem)

    (P2): at least one member of S (set covering problem)

    (P3): exactly one member of S (set partitioning problem)

    The, three problems (P1), (P2) and (P3) can be formulated as integer linear programs as follows:

Let A denote the $m \times n$ matrix where

$$A_{ij} = \begin{cases} 1 & \text{if element } i \in F_j \\ 0 & \text{otherwise} \end{cases}$$

The decision variables are x_j, $j = 1, 2, .., n$ where

$$x_j = \begin{cases} 1 & \text{if } F_j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

The set packing problem is

$$\begin{array}{lll} \text{(P1)} & \text{Max} & cx \\ & \text{Subject to} & Ax \leq e_m \\ & & x_j = 0 \text{ or } 1, \quad j = 1, 2, .., n \end{array}$$

where e_m is a m-dimensional column vector of 1s.

The set covering problem (P2) is (P1) with less than or equal to constraints replaced by greater than or equal to constraints and the objective is to minimize rather than maximize. The set partitioning problem (P3) is (P1) with the constraints written as equalities. The set partitioning problem can be converted to a set packing problem or a set covering problem (see [98]) using standard transformations. If the right hand side vector e_m is replaced by a non-negative integer vector b, (P1) is referred to as the generalised set packing problem.

The airline crew scheduling problem is a classic example of the set partitioning or the set covering problem. Each element of $\mathcal{M}$ corresponds to a flight segment. Each subset F_j corresponds to an acceptable set of flight segments of a crew. The problem is to cover at minimum cost, each flight segment exactly once. This is a set partitioning problem. If *dead heading* of crew is permitted, we have the set covering problem.

11. PACKING AND COVERING PROBLEMS IN A GRAPH: Suppose A is the node-edge incidence matrix of a graph. Now, (P1) is a weighted matching problem. If in addition, the right hand side vector e_m is replaced by a non-negative integer vector b, (P1) is referred to as a weighted b-matching problem. In this case, each variable x_j which is restricted to be an integer may have a positive upper bound of u_j. Problem (P2) is now referred to as the weighted edge covering problem. Note that by substituting for $x_j = 1 - y_j$, where $y_j = 0$ *or* 1, the weighted edge covering problem is transformed to a weighted b-matching problem in which the variables are restricted to be 0 or 1.

Suppose A is the edge-node incidence matrix of a graph. Now, (P1) is referred to as the weighted vertex packing problem and (P2) is referred to as the weighted vertex covering problem. It is

easy to see that the weighted vertex packing problem and the weighted vertex covering problem are equivalent in the sense that the complement of an optimal solution to one problem defines an optimal solution to the other. The *set packing* problem can be transformed to a weighted vertex packing problem in a graph G as follows:

> G contains a node for each x_j and an edge between nodes j and k exists if and only if the columns $A_{.j}$ and $A_{.k}$ are not orthogonal. G is called the *intersection graph* of A. Given G, the complement graph $\bar{G}$ has the same node set as G and there is an edge between nodes j and k in $\bar{G}$ if and only if there is no such corresponding edge in G. A clique in a graph is a subset, k, of nodes of G such that the subgraph induced by k is complete. Clearly, the weighted vertex packing problem in G is equivalent to finding a maximum weighted clique in $\bar{G}$.

12. SATISFIABILITY AND INFERENCE PROBLEMS: In propositional logic, a truth assignment is an assignment of "true" or "false" to each atomic proposition $x_1, x_2, ..x_n$. A literal is an atomic proposition x_j or its negation $\neg x_j$. For propositions in conjunctive normal form, a clause is a disjunction of literals and the proposition is a conjunction of clauses. A clause is obviously satisfied by a given truth assignment if at least one of its literals is true. The satisfiability problem consists of determining whether there exists a truth assignment to atomic propositions such that a set S of clauses is satisfied.

    Let T_i denote the set of atomic propositions such that if any one of them is assigned "true", the clause $i \in S$ is satisfied. Similarly let F_i denote the set of atomic propositions such that if any one of them is assigned "false", the clause $i \in S$ is satisfied.

    The decision variables are

$$x_j = \begin{cases} 1 & \text{if atomic proposition } j \text{ is assigned true} \\ 0 & \text{if atomic proposition } j \text{ is assigned false} \end{cases}$$

    The satisfiability problem is to find a feasible solution to

$$(P4) \quad \sum_{j \in T_i} x_j - \sum_{j \in F_i} x_j \geq 1 - |F_i| \quad i \in S$$

$$x_j = 0 \text{ or } 1 \text{ for } j = 1, 2, .., n$$

    By substituting $x_j = 1 - y_j$, where $y_j = 0$ or 1, for $j \in F_i$, (P4) is equivalent to the set covering problem

$$(P5) \qquad \text{Min} \quad \sum_{j=1}^{n}(x_j + y_j)$$

$$\text{subject to} \quad \sum_{j \in T_i} x_j + \sum_{j \in F_i} y_j \geq 1 \; i \in S$$

$$x_j + y_j \geq 1 \;\; j = 1, 2, .., n$$

$$x_j, y_j = 0 \text{ or } 1 \;\; j = 1, 2, .., n$$

Clearly (P4) is feasible, if and only if (P5) has an optimal objective function value equal to n.

Given a set S of clauses and an additional clause $k \notin S$, the logical inference problem is to find out whether every truth assignment that satisfies all the clauses in S also satisfies the clause k. The logical inference problem is

$$(P6) \qquad \text{Min} \quad \sum_{j \in T_k} x_j - \sum_{j \in F_k} x_j$$

$$\text{subject to} \quad \sum_{j \in T_i} x_j - \sum_{j \in F_i} x_j \geq 1 - \mid F_i \mid \; i \in S$$

$$x_j = 0 \text{ or } 1 \;\; j = 1, 2, .., n$$

The clause k is implied by the set of clauses S, if and only if (P6) has an optimal objective function value greater than $- \mid F_k \mid$. It is also straightforward to express the MAX-SAT problem (i.e. find a truth assignment that maximizes the number of satisfied clauses in a given set S) as an integer linear program.

13. MULTI-PROCESSOR SCHEDULING: Given n jobs and m processors, the problem is to allocate each job to one and only one of the processors so as to minimize the make span time, i.e minimize the completion time of all the jobs. The processors may not be identical and hence job j if allocated to processor i requires p_{ij} units of time. The multi-processor scheduling problem is

$$(P7) \quad \text{Min} \quad T$$

$$\begin{aligned} \text{subject to} \quad & \textstyle\sum_{i=1}^{m} \quad x_{ij} = 1 \quad j = 1,2,..,n \\ & \textstyle\sum_{j=1}^{n} \quad p_{ij}x_{ij} - T \leq 0 \quad i = 1,2,..,m \\ & \qquad\qquad x_{ij} = 0 \text{ or } 1 \quad \forall\, ij \end{aligned}$$

Note that if all p_{ij} are integers, the optimal solution will be such that T is an integer.

3.2 Jeroslow's Representability Theorem

R. Jeroslow [70], building on joint work with J. K. Lowe [71], characterized subsets of n-space that can be represented as the feasible region of a mixed integer (Boolean) program. They proved that a set is the feasible region of some mixed integer/linear programming problem (MILP) if and only if it is the union of finitely many polyhedra having the same recession cone (defined below). Although this result is not widely known, it might well be regarded as the fundamental theorem of mixed integer modeling.

The basic idea of Jeroslow's results is that any set that can be represented in a mixed integer model can be represented in a disjunctive programming problem (i.e., a problem with either/or constraints). A *recession direction* for a set S in n-space is a vector $\mathbf{x}$ such that $\mathbf{s} + \alpha\mathbf{x} \in S$ for all $\mathbf{s} \in S$ and all $\alpha \geq 0$. The set of recession directions is denoted $rec(S)$. Consider the general mixed integer constraint set below.

$$\begin{aligned} & f(\mathbf{x}, \mathbf{y}, \lambda) \leq \mathbf{b} \qquad (3) \\ & \mathbf{x} \in \Re^n, \ \ \mathbf{y} \in \Re^p \\ & \lambda = (\lambda_1, \ldots, \lambda_k), \ \text{ with } \ \lambda_j \in \{0,1\} \ \text{ for } j = 1, \ldots, k \end{aligned}$$

Here $\mathbf{f}$ is vector-valued function, so that $\mathbf{f}(\mathbf{x}, \mathbf{y}, \lambda) \leq b$ represents a set of constraints. We say that a set $S \subset \Re^n$ is *represented* by (3) if,

$$\mathbf{x} \in S \ \text{ if and only if } (\mathbf{x}, \mathbf{y}, \lambda) \text{ satisfies (3) for some } \mathbf{y}, \lambda.$$

If $\mathbf{f}$ is a linear transformation, so that (3) is a MILP constraint set, we will say that S is *MILP representable*. The main result can now be stated.

Theorem 3.1 (Jeroslow, Lowe [71,70]) *A set in n-space is MILP representable if and only if it is the union of finitely many polyhedra having the same set of recession directions.*

3.3 Benders Representation

Any mixed integer linear program (MILP) can be reformulated so that there is only one continuous variable. This reformulation, due to Benders [10], will in general have an exponential number of constraints. Benders representation suggests an algorithm for mixed integer programming (known as Benders Decomposition in the literature because of its similarity to Dantzig-Wolfe Decomposition, cf. [95]) that uses dynamic activation of these rows (constraints) as and when required.

Consider the (MILP)

$$\max\{\mathbf{cx}+\mathbf{dy}: A\mathbf{x}+G\mathbf{y} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, y \geq \mathbf{0} \text{ and integer}\}$$

Suppose the integer variables $\mathbf{y}$ are fixed at some values, then the associated linear program is

$$(LP) \quad \max\{\mathbf{cx}: \mathbf{x} \in \mathcal{P} = \{\mathbf{x}: A\mathbf{x} \leq \mathbf{b}-G\mathbf{y}, \mathbf{x} \geq \mathbf{0}\}\}$$

and its dual is

$$(DLP) \quad \min\{\mathbf{w}(\mathbf{b}-G\mathbf{y}): \mathbf{w} \in Q = \{\mathbf{w}: \mathbf{w}A \geq \mathbf{c}, \mathbf{w} \geq 0\}\}$$

Let $\{\mathbf{w}^k\}$, $k = 1, 2, \cdots, K$ be the extreme points of Q and $\{\mathbf{u}^j\}$, $j = 1, 2, \cdots, J$ be the extreme rays of the recession cone of Q, $C_Q = \{\mathbf{u}: \mathbf{u}A \geq \mathbf{0}, \mathbf{u} \geq \mathbf{0}\}$. Note that if Q is nonempty, the $\{\mathbf{u}^j\}$ are all the extreme rays of Q.

From linear programming duality, we know that if Q is empty and $\mathbf{u}^j(\mathbf{b}-G\mathbf{y}) \geq 0, j = 1, 2, \cdots, J$ for some $\mathbf{y} \geq \mathbf{0}$ and integer then (LP) and consequently $(MILP)$ has an unbounded solution. If Q is non-empty and $\mathbf{u}^j(\mathbf{b}-G\mathbf{y}) \geq 0, j = 1, 2, \cdots, J$ for some $\mathbf{y} \geq \mathbf{0}$ and integer then (LP) has a finite optimum given by

$$\min_k\{\mathbf{w}^k(\mathbf{b}-G\mathbf{y})\}$$

Hence an equivalent formulation of $(MILP)$ is

$$\begin{aligned}
&max\ \alpha \\
\alpha &\leq \mathbf{dy} + \mathbf{w}^k(\mathbf{b}-G\mathbf{y}), \quad k = 1, 2, \cdots, K \\
\mathbf{u}^j(\mathbf{b}-G\mathbf{y}) &\geq 0, \quad j = 1, 2, \cdots, J \\
\mathbf{y} &\geq \mathbf{0} \text{ and integer} \\
\alpha &\quad \textit{unrestricted}
\end{aligned}$$

which has only one continuous variable α as promised.

3.4 Aggregation

An integer linear programming problem with only one constraint, other than upper bounds on the variables, is referred to as a Knapsack problem. An integer linear programming problem with m

constraints can be represented as a Knapsack problem. In this section, we show this representation for an integer linear program with bounded variables [96,53]. We show how two constraints can be aggregated into a single constraint. By repeated application of this aggregation, an integer linear program with m constraints can be represented as a Knapsack problem.

Consider the feasible set S defined by two constraints with integer coefficients and m non-negative integer variables with upper bounds, i.e.,

$$S = \{\mathrm{x} | \sum_{j=1}^{n} a_j x_j = d_1;\ \sum_{j=1}^{n} b_j x_j = d_2;\ 0 \leq x_j \leq u_j \text{ and } x_j \text{ integer}\}$$

Consider now the problem

$$(P) \quad z = \max\{|\sum_{j=1}^{n} a_j x_j - d_1| \ :\ 0 \leq x_j \leq u_j \text{ and } x_j \text{ integer},\ j = 1, 2, ..., n\}$$

This problem is easily solved by considering two solutions, one in which the variables x_j with positive coefficients are set to u_j while the other variables are set to zero and the other in which the variables x_j with negative coefficients are set to u_j while the other variables are set to zero.

Let α be an integer greater than z, the maximum objective value of (P).

It is easy to show that S is equivalent to

$$K = \{\mathrm{x} | \sum_{j=1}^{n} (a_j + \alpha b_j) x_j = d_1 + \alpha d_2;\ 0 \leq x_j \leq u_j \text{ and } x_j \text{ integer}\}$$

Note that if $\mathrm{x}^* \in S$, then clearly $\mathrm{x}^* \in K$. Suppose $\mathrm{x}^* \in K$. Now we show that $\sum_{j=1}^{n} b_j x_j = d_2$. Suppose not and that

$$\sum_{j=1}^{n} b_j x_j = d_2 + k \tag{4}$$

where k is some arbitrary integer, positive or negative.

Now multiplying (4) by α, and subtracting it from the equality constraint defining K, we have $\sum_{j=1}^{n} a_j x_j^* = d_1 - \alpha k$

But since $|\sum_{j=1}^{n} a_j x_j^* - d_1| < \alpha$, it follows that $k = 0$ and $\mathrm{x}^* \in S$.

4 Polyhedral Combinatorics

One of the main purposes of writing down an algebraic formulation of a combinatorial optimization problem as an integer program is to then examine the linear programming relaxation and understand how well it represents the discrete integer program [107]. There are somewhat special but rich classes of such formulations for which the linear programming relaxation is sharp or tight. These special structures are presented next.

4.1 Special Structures and Integral Polyhedra

A natural question of interest is whether the LP associated with an ILP has only integral extreme points. For instance, the linear programs associated with matching and edge covering polytopes in a bipartite graph have only integral vertices. Clearly, in such a situation, the ILP can be solved as LP. A polyhedron or a polytope is referred to as being integral if it is either empty or has only integral vertices.

Definition 4.1 *A* $0, \pm 1$ *matrix is totally unimodular if the determinant of every square submatrix is* 0 *or* ± 1.

Theorem 4.2 *Hoffman and Kruskal [64] Let* $A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix}$ *be a* $0, \pm 1$ *matrix and* $\mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix}$ *be a vector of appropriate dimensions. Then* A *is totally unimodular if and only if the polyhedron*

$$P(A, \mathbf{b}) = \{\mathbf{x} \ : \ A_1\mathbf{x} \leq \mathbf{b}_1; A_2\mathbf{x} \geq \mathbf{b}_2; A_3\mathbf{x} = \mathbf{b}_3; \mathbf{x} \geq \mathbf{0}\}$$

is integral for all integral vectors $\mathbf{b}$.

The constraint matrix associated with a network flow problem (see for instance [1]) is totally unimodular. Note that for a given integral $\mathbf{b}$, $P(A, \mathbf{b})$ may be integral even if A is not totally unimodular.

Definition 4.3 *A polyhedron defined by a system of linear constraints is totally dual integral(TDI) if for each objective function with integral coefficients, the dual linear program has an integral optimal solution whenever an optimal solution exists.*

Theorem 4.4 *Edmonds and Giles [43]*
If $P(A) = \{\mathbf{x} \ : \ A\mathbf{x} \leq \mathbf{b}\}$ *is TDI and* $\mathbf{b}$ *is integral, then* $P(A)$ *is integral.*

Hoffman and Kruskal [64] have in fact shown that the polyhedron $P(A, \mathbf{b})$ defined in Theorem 4.2 is TDI. This follows from Theorem 4.2 and the fact that A is totally unimodular if and only if A^T is totally unimodular.

Balanced matrices, first introduced by Berge [13] have important implications for packing and covering problems (see also [14]).

Definition 4.5 *A* $0, 1$ *matrix is balanced if it does not contain a square submatrix of odd order with two ones per row and column.*

Theorem 4.6 *Berge [13], Fulkerson, Hoffman and Oppenheim [52])*

Let A be a balanced $0,1$ matrix. Then the set packing, set covering and set partitioning polytopes associated with A are integral, i.e., the polytopes

$P(A) = \{x : x \geq 0; Ax \leq 1\}$

$Q(A) = \{x : 0 \leq x \leq 1; Ax \geq 1\}$ *and*

$R(A) = \{x : x \geq 0; Ax = 1\}$

are integral.

Let $A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix}$ be a balanced $0,1$ matrix. Fulkerson, Hoffman and Oppenheim [52] have shown that the polytope $P(A) = \{x : A_1x \leq 1; A_2x \geq 1; A_3x = 1; x \geq 0\}$ is TDI and by the theorem of Edmonds and Giles [43] it follows that $P(A)$ is integral.

Truemper [119] has extended the definition of balanced matrices to include $0, \pm 1$ matrices.

Definition 4.7 *A $0,\pm 1$ matrix is balanced if for every square submatrix with exactly two nonzero entries in each row and each column, the sum of the entries is a multiple of 4.*

Theorem 4.8 *Conforti and Cornuejols [26]*

Suppose A is a balanced $0,\pm 1$ matrix. Let $n(A)$ denote the column vector whose i^{th} component is the number of -1s in the i^{th} row of A. Then the polytopes

$P(A) = \{x : Ax \leq 1 - n(A); 0 \leq x \leq 1\}$

$Q(A) = \{x : Ax \geq 1 - n(A); 0 \leq x \leq 1\}$

$R(A) = \{x : Ax = 1 - n(A); 0 \leq x \leq 1\}$

are integral.

Note that a $0,\pm 1$ matrix A is balanced if and only if A^T is balanced. Moreover A is balanced (totally unimodular) if and only if every submatrix of A is balanced (totally unimodular). Thus if A is balanced (totally unimodular) it follows that Theorem 4.8 (Theorem 4.2) holds for every submatrix of A.

Totally unimodular matrices constitute a subclass of balanced matrices, i.e., a totally unimodular $0,\pm 1$ matrix is always balanced. This follows from a theorem of Camion [17] which states that a $0,\pm 1$ is totally unimodular if and only if for every square submatrix with an even number of nonzeros entries in each row and in each column, the sum of the entries equals a multiple of 4. The 4 x 4 matrix in Figure 2 illustrates the fact that a balanced matrix is not necessarily totally unimodular. Balanced $0,\pm 1$ matrix have implications for solving the satisfiability problem. If the given set of clauses defines a balanced $0,\pm 1$ matrix, then as shown by Conforti and Cornuejols [26], the satisfiability problem

is trivial to solve and the associated MAXSAT problem is solvable in polynomial time by linear programming. A survey of balanced matrices is in Conforti et al [29].

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 2. A Balanced Matrix and A Perfect Matrix

Definition 4.9 *A* $0,1$ *matrix* A *is perfect if the set packing polytope* $P(A) = \{\mathbf{x} : A\mathbf{x} \leq \mathbf{1}; \mathbf{x} \geq \mathbf{0}\}$ *is integral.*

The chromatic number of a graph is the minimum number of colors required to color the vertices of the graph so that no two vertices with the same color have an edge incident between them. A graph G is perfect if for every node induced subgraph H, the chromatic number of H equals the number of nodes in the maximum clique of H. The connections between the integrality of the set packing polytope and the notion of a perfect graph, as defined by Berge [11,12], are given in Fulkerson [51], Lovasz [89], Padberg [97], and Chvatal [25].

Theorem 4.10 *Fulkerson [51], Lovasz [89], Chvatal [25]*

Let A *be* $0,1$ *matrix whose columns correspond to the nodes of a graph* G *and whose rows are the incidence vectors of the maximal cliques of* G. *The graph* G *is perfect if and only if* A *is perfect.*

Let G_A denote the intersection graph associated with a given $0,1$ matrix A (see Section 3.1). Clearly, a row of A is the incidence vector of a clique in G_A. In order for A to be perfect, every maximal clique of G_A must be represented as a row of A because inequalities defined by maximal cliques are facet defining. Thus by Theorem 4.10, it follows that a $0,1$ matrix A is perfect if and only if the undominated (a row of A is dominated if its support is contained in the support of another row of A) rows of A form the clique-node incidence matrix of a perfect graph.

Balanced matrices with $0,1$ entries, constitute a sub-class of $0,1$ perfect matrices, i.e., if a $0,1$ matrix A is balanced, then A is perfect. The 4 x 3 matrix in Figure 2 is an example of a matrix which is perfect but not balanced.

Definition 4.11 *A* $0,1$ *matrix* A *is ideal if the set covering polytope*

$$Q(A) = \{\mathbf{x} : A\mathbf{x} \geq \mathbf{1}; \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$$

is integral.

Properties of ideal matrices are described by Lehman [85], Padberg [99] and Cornuejols and Novick [32]. The notion of a $0,1$ perfect (ideal) matrix has a natural extension to a $0,\pm 1$ perfect (ideal) matrix. Some results pertaining to $0,\pm 1$ ideal matrices are contained in Hooker [65] while some results pertaining to $0,\pm 1$ perfect matrices are given in Conforti, Cornuejols and De Francesco [27].

An interesting combinatorial problem is to check whether a given $0,\pm 1$ matrix is totally unimodular, balanced or perfect. Seymour's [116] characterization of totally unimodular matrices provides a polynomial time algorithm to test whether a given matrix $0,1$ matrix is totally unimodular. Conforti, Cornuejols and Rao [30] give a polynomial time algorithm to check whether a $0,1$ matrix is balanced. This has been extended by Conforti et al [28] to check in polynomial time whether a $0,\pm 1$ matrix is balanced. An open problem is that of checking in polynomial time whether a $0,1$ matrix is perfect. For linear matrices (a matrix is linear if it does not contain a 2 x 2 submatrix of all ones), this problem has been solved by Fonlupt and Zemirline [47] and Conforti and Rao [31].

4.2 Matroids

Matroids and submodular functions have been studied extensively, especially from the point of view of combinatorial optimization (see for instance Nemhauser & Wolsey [95]). Matroids have nice properties that lead to efficient algorithms for the associated optimization problems. One of the interesting examples of a matroid is the problem of finding a maximum or minimum weight spanning tree in a graph. Two different but equivalent definitions of a matroid are given first. A greedy algorithm to solve a linear optimization problem over a matroid is presented. The matroid intersection problem is then discussed briefly.

Definitions 4.12 *Let $N = \{1,2,\cdot,n\}$ be a finite set and let $\mathcal{F}$ be a set of subsets of N. Then $I = (N,\mathcal{F})$ is an independence system if $S_1 \in \mathcal{F}$ implies that $S_2 \in \mathcal{F}$ for all $S_2 \subseteq S_1$. Elements of $\mathcal{F}$ are called independent sets. A set $S \in \mathcal{F}$ is a maximal independent set if $S \cup \{j\} \notin \mathcal{F}$ for all $j \in N \backslash S$. A maximal independent set T is a maximum if $|T| \geq |S|$ for all $S \in \mathcal{F}$.*

The rank $r(Y)$ of a subset $Y \subseteq N$ is the cardinality of the maximum independent subset $X \subseteq Y$. Note that $r(\phi) = 0$, $r(X) \leq |X|$ for $X \subseteq N$ and the rank function is non-decreasing, i.e. $r(X) \leq r(Y)$ for $X \subseteq Y \subseteq N$.

A matroid $M = (N,\mathcal{F})$ is an independence system in which every maximal independent set is a maximum.

Examples 4.13 *Let $G = (V,E)$ be an undirected connected graph with V as the node set and E as the edge set.*

(i) Let $I = (E, \mathcal{F})$ where $F \in \mathcal{F}$ if $F \subseteq E$ is such that at most one edge in F is incident to each node of V, i.e. $F \in \mathcal{F}$ if F is a matching in G. Then $I = (E, \mathcal{F})$ is an independence system but not a matroid.

(ii) Let $M = (E, \mathcal{F})$ where $F \in \mathcal{F}$ if $F \subseteq E$ is such that $G_F = (V, F)$ is a forest i.e. G_F contains no cycles. Then $M = (E, \mathcal{F})$ is a matroid and maximal independent sets of M are spanning trees.

An alternate but equivalent definition of matroids is in terms of submodular functions.

Definitions 4.14 *Let N be a finite set. A real valued set function f defined on the subsets of N is submodular if $f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y)$ for $X, Y \subseteq N$.*

Examples 4.15 *Let $G = (V, E)$ be an undirected graph with V as the node set and E as the edge set. Let $c_{ij} \geq 0$ be the weight or capacity associated with edge $(ij) \in E$. For $S \subseteq V$, define the cut function $c(S) = \sum_{i \in S, j \in V \setminus S} c_{ij}$. The cut function defined on the subsets of V is submodular since $c(X) + c(Y) - c(X \cup Y) - c(X \cap Y) = \sum_{i \in X \setminus Y, j \in Y \setminus X} 2c_{ij} \geq 0$.*

Definitions 4.16 *A non-decreasing integer valued submodular function r defined on the subsets of N, is called a matroid rank function if $r(\phi) = 0$ and $r(\{j\}) \leq 1$ for $j \in N$. The pair (N, r) is called a matroid.*

A non-decreasing, integer-valued, submodular function f, defined on the subsets of N is called a polymatroid function if $f(\phi) = 0$. The pair (N, r) is called a polymatroid.

4.2.1 Matroid Optimization

In order to decide whether an optimization problem over a matroid is polynomially solvable or not, we need to first address the issue of representation of a matroid. If the matroid is given either by listing the independent sets or by its rank function, many of the associated linear optimization problems are trivial to solve. However, matroids associated with graphs are completely described by the graph and the condition for independence. For instance, the matroid in which the maximal independent sets are spanning trees, the graph $G = (V, E)$ and the independence condition of no cycles describes the matroid.

Most of the algorithms for matroid optimization problems require a test to determine whether a specified subset is independent or not. We assume the existence of an oracle or subroutine to do this checking in running time which is a polynomial function of $|N| = n$.

Maximum Weight Independent Set

Given a matroid $M = (N, \mathcal{F})$ and weights w_j for $j \in N$, the problem of finding a maximum weight independent set is $\max_{F \in \mathcal{F}} \left\{ \sum_{j \in F} w_j \right\}$. The greedy algorithm to solve this problem is as follows:

Procedure: Greedy

0. **Initialize:** Order the elements of N so that $w_i \geq w_{i+1}$, $i = 1, 2, \cdots, n-1$. Let $T = \phi$, $i = 1$.

1. If $w_i \leq 0$ or $i > n$, stop T is optimal, i.e. $x_j = 1$ for $j \in T$ and $x_j = 0$ for $j \notin T$. If $w_i > 0$ and $T \cup \{i\} \in \mathcal{F}$, add element i to T.

2. **Increment** i by 1 and return to step 1.

Edmonds [40, 41] derived a complete description of the *matroid polytope*, the convex hull of the characteristic vectors of independent sets of a matroid. While this description has a large (exponential) number of constraints, it permits the treatment of linear optimization problems on independent sets of matroids as linear programs. Cunningham [35] describes a polynomial algorithm to solve the *separation problem*[5] for the matroid polytope. The matroid polytope and the associated greedy algorithm have been extended to polymatroids [40,93].

The separation problem for a polymatroid is equivalent to the problem of minimizing a submodular function defined over the subsets of N, see Nemhauser and Wolsey [95]. A class of submodular functions that have some additional properties can be minimized in polynomial time by solving a maximum flow problem (Rhys [109], Picard and Ratliff [106]). The general submodular function can be minimized in polynomial time by the ellipsoid algorithm (Grötschel, Lovász and Schrijver [60]).

The uncapacitated plant location problem (see Section 6.1 can be reduced to maximizing a submodular function. Hence it follows that maximizing a submodular function is NP-hard.

4.2.2 Matroid Intersection

A matroid intersection problem involves finding an independent set contained in two or more matroids defined on the same set of elements.

Let $G = (V_1, V_2, E)$ be a bipartite graph. Let $M_i = (E, \mathcal{F}_i), i = 1, 2$ where $F \in \mathcal{F}_i$ if $F \subseteq E$ is such that no more than one edge of F is incident to each node in V_i. The set of matchings in

[5]The separation problem for a convex body K is to test if an input point x belongs to K and if it does not to produce a linear halfspace that separates x from K. It is known [60,80,100] that linear optimization over K is polynomially equivalent to separation from K.

G constitute the intersection of the two matroids $M_i, i = 1, 2$. The problem of finding a maximum weight independent set in the intersection of two matroids can be solved in polynomial time (Lawler 84], Edmonds [40,42], Frank [49]). The two (poly) matroid intersection polytope has been studied by Edmonds [42].

The problem of testing whether a graph contains a Hamiltonian path is NP-complete. Since this problem can be reduced to the problem of finding a maximum cordinality independent set in the intersection of three matroids, it follows that the matroid intersection problem involving three or more matroids is NP-hard.

4.3 Valid Inequalities, Facets and Cutting Plane Methods

In Section 4.1, we were concerned with conditions under which the packing and covering polytopes are integral. But in general these polytopes are not integral and additional inequalities are required to have a complete linear description of the convex hull of integer solutions. The existence of finitely many such linear inequalities is guaranteed by Weyl's Theorem [120].

Consider the feasible region of an ILP given by

$$P_I = \{\mathbf{x} \ : \ A\mathbf{x} \leq \mathbf{b}; \mathbf{x} \geq \mathbf{0} \text{ and integer}\} \tag{5}$$

Recall that an inequality $\mathbf{fx} \leq f_0$ is referred to as a valid inequality for P_I if $\mathbf{fx}^* \leq f_0$ for all $\mathbf{x}^* \in P_I$. A valid linear inequality for $P_I(A, \mathbf{b})$ is said to be facet defining if it intersects $P_I(A, \mathbf{b})$ in a face of dimension one less than the dimension of $P_I(A, \mathbf{b})$. In the example shown in Figure 3, the inequality $x_2 + x_3 \leq 1$ is a facet defining inequality of the integer hull.

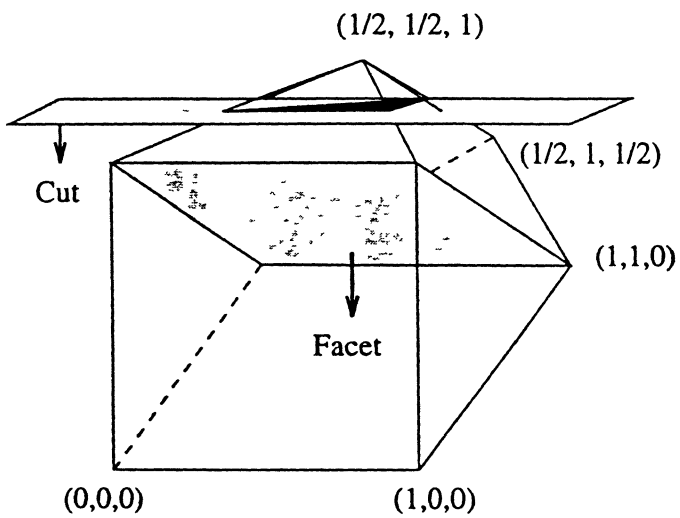


Figure 3: Relaxation, Cuts and Facets

The Relaxation

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 2 \\ x_1 - x_2 - x_3 &\geq -1 \\ 0 \leq x_1, x_2, x_3 &\leq 1 \end{aligned}$$

The Integer Hull

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 2 \\ x_1 - x_2 - x_3 &\geq -1 \\ x_2 + x_3 &\leq 1 \\ 0 \leq x_1, x_2, x_3 &\leq 1 \end{aligned}$$

Let $\mathbf{u} \geq 0$ be a row vector of appropriate size. Clearly $\mathbf{u}A\mathbf{x} \leq \mathbf{ub}$ holds for every $\mathbf{x}$ in P_I.

Let $(\mathbf{u}A)_j$ denote the j^{th} component of the row vector $\mathbf{u}A$ and $\lfloor(\mathbf{u}A)_j\rfloor$ denote the largest integer less than or equal to $(\mathbf{u}A)_j$. Now, since $\mathbf{x} \in P_I$ is a vector of non-negative integers, it follows that $\sum_j \lfloor(\mathbf{u}A)_j\rfloor x_j \leq \lfloor \mathbf{ub} \rfloor$ is a valid inequality for P_I. This scheme can be used to generate many valid inequalities by using different $\mathbf{u} \geq 0$. Any set of generated valid inequalities may be added to the constraints in (5) and the process of generating them may be repeated with the enhanced set of inequalities. This iterative procedure of generating valid inequalities is called Gomory-Chvátal (GC) Rounding. It is remarkable that this simple scheme is complete, i.e. every valid inequality of P_I can be generated by finite application of GC rounding [24,113].

The number of inequalities needed to describe the convex hull of P_I is usually exponential in the size of A. But to solve an optimization problem on P_I, one is only interested in obtaining a partial description of P_I that facilitates the identification of an integer solution and prove its optimality. This is the underlying basis of any cutting plane approach to combinatorial problems.

4.3.1 The Cutting Plane Method

Consider the optimization problem

$$\max\{\mathbf{cx} : \mathbf{x} \in P_I = \{\mathbf{x} : A\mathbf{x} \leq \mathbf{b};\ \mathbf{x} \geq \mathbf{0} \text{ and integer}\}\}$$

The generic cutting plane method as applied to this formulation is given below.

Procedure: Cutting Plane

1. Initialize $A' \leftarrow A$ and $\mathbf{b}' \leftarrow \mathbf{b}$.

2. Find an optimal solution $\bar{\mathbf{x}}$ to the linear program

$$\max\{\mathbf{cx} : A'\mathbf{x} \leq \mathbf{b}';\ \mathbf{x} \geq \mathbf{0}\}$$

   If $\bar{\mathbf{x}} \in P_I$, stop and return $\bar{\mathbf{x}}$.

3. Generate a valid inequality $\mathbf{fx} \leq f_0$ for P_I such that $f\bar{\mathbf{x}} > f_0$ (the inequality ``cuts'' $\bar{\mathbf{x}}$).

4. Add the inequality to the constraint system, update

$$A' \leftarrow \begin{pmatrix} A' \\ \mathbf{f} \end{pmatrix},\quad \mathbf{b}' \leftarrow \begin{pmatrix} \mathbf{b}' \\ f_0 \end{pmatrix}$$

   Go to step 2.

In Step 3 of the cutting plane method, we require a suitable application of the GC rounding scheme (or some alternate method of identifying a cutting plane). Notice that while the GC rounding scheme

will generate valid inequalities, the identification of one that cuts off the current solution to the linear programming relaxation is all that is needed. Gomory [56] provided just such a specialization of the rounding scheme that generates a cutting plane. While this met the theoretical challenge of designing a sound and complete cutting plane method for integer linear programming, it turned out to be a weak method in practice. Successful cutting plane methods, in use today, use considerable additional insights into the structure of facet-defining cutting planes. Using facet cuts makes a huge difference in the speed of convergence of these methods. Also, the idea of combining cutting plane methods with search methods has been found to have a lot of merit. These branch and cut methods will be discussed in the next section.

4.3.2 The b-Matching Problem

Consider the b-matching problem:

$$\max\{\mathbf{cx}: \ A\mathbf{x} \leq \mathbf{b}, \ \mathbf{x} \geq \mathbf{0} \ \text{ and } \ \text{integer}\} \tag{6}$$

where A is the node-edge incidence matrix of an undirected graph and $\mathbf{b}$ is a vector of positive integers. Let G be the undirected graph whose node-edge incidence matrix is given by A and let $W \subseteq V$ be any subset of nodes of G (i.e., subset of rows of A) such that $\mathrm{b}(W) = \sum_{i \in W} b_i$ is odd. Then the inequality

$$\mathrm{x}(W) = \sum_{e \in E(W)} x_e \leq \frac{1}{2}(\mathrm{b}(W) - 1) \tag{7}$$

is a valid inequality for integer solutions to (6) where $E(W) \subseteq E$ is the set of edges of G having both ends in W. Edmonds [39] has shown that the inequalities (6) and (7) define the integral b-matching polytope (see also [44]). Note that the number of inequalities (7) is exponential in the number of nodes of G. An instance of the successful application of the idea of using only a partial description of P_I is in the blossom algorithm for the matching problem, due to Edmonds [39].

An implication of the ellipsoid method for linear programming is that the linear program over P_I can be solved in polynomial time if and only if the associated separation problem can be solved in polynomial time (see Grotschel, Lovasz and Schrijver [60], Karp and Papadimitriou [80], and Padberg and Rao [100]). The separation problem for the b-matching problem with or without upper bounds was shown by Padberg and Rao [101], to be solvable in polynomial time. The procedure involves a minor modification of the algorithm of Gomory and Hu [59] for multiterminal networks. However, no polynomial (in the number of nodes of the graph) linear programming formulation of this separation problem is known. Martin [92] has shown that if the separation problem can be expressed as a compact linear program then so can the optimization problem. Hence an unresolved issue is whether there exists a polynomial size (compact) formulation for the b-matching problem. Yannakakis [123] has shown that, under a symmetry assumption, such a formulation is impossible.

4.3.3 Other Combinatorial Problems

Besides the matching problem several other combinatorial problems and their associated polytopes have been well studied and some families of facet defining inequalities have been identified. For instance the set packing, graph partitioning, plant location, maximum cut, travelling salesman and Steiner tree problems have been extensively studied from a polyhedral point of view (see for instance Nemhauser and Wolsey [95]).

These combinatorial problems belong to the class of NP-complete problems. In terms of a worst-case analysis, no polynomial time algorithms are known for these problems. Nevertheless, using a cutting plane approach with branch and bound or branch and cut (see Section 5), large instances of these problems have been successfully solved, see Crowder, Johnson and Padberg [34], for general $0-1$ problems, Barahona et al [7] for the maximum cut problem, Padberg and Rinaldi [102] for the traveling salesman problem and Chopra, Gorres and Rao [23] for the Steiner tree problem.

5 Partial Enumeration Methods

In many instances, to find an optimal solution to an integer linear programing problems (ILP), the structure of the problem is exploited together with some sort of partial enumeration. In this section, we review the branch and bound (B & B) and branch and cut (B & C) methods for solving an ILP.

5.1 Branch and Bound

The branch and bound (B & B) method is a systematic scheme for implicitly enumerating the finitely many feasible solutions to an ILP. Although, theoretically the size of the enumeration tree is exponential in the problem parameters, in most cases, the method eliminates a large number of feasible solutions. The key features of branch and bound method are

(i) **Selection/Removal** of one or more problems from a candidate list of problems.

(ii) **Relaxation** of the selected problem so as to obtain a lower bound (on a minimization problem) on the optimal objective function value for the selected problem.

(iii) **Fathoming**, if possible, of the selected problem.

(iv) **Branching Strategy:** If the selected problem is not fathomed, branching creates sub-problems which are added to the candidate list of problems.

The above four steps are repeated until the candidate list is empty. The B & B method sequentially examines problems that are added and removed from a candidate list of problems.

`Initialization:`

Initially the candidate list contains only the original ILP which is denoted as

$$(P) \qquad \min\{\mathbf{cx} : A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \text{ and integer}\}$$

Let $F(P)$ denote the feasible region of (P) and $z(P)$ denote the optimal objective function value of (P). For any $\bar{\mathbf{x}}$ in $F(P)$, let $z_P(\bar{\mathbf{x}}) = \mathbf{c}\bar{\mathbf{x}}$.

Frequently, heuristic procedures are first applied to get a good feasible solution to (P). The best solution known for (P) is referred to as the current incumbent solution. The corresponding objective function value is denoted as z_I. In most instances, the initial heuristic solution is not either optimal or at least not immediately certified to be optimal. So further analysis is required to ensure that an optimal solution to (P) is obtained. If no feasible solution to (P) is known, z_I is set to ∞.

`Selection/Removal:`

In each iterative step of B & B, a problem is selected and removed from the candidate list for further analysis. The selected problem is henceforth referred to as the candidate problem (CP). The algorithm terminates if there is no problem to select from the candidate list. Initially there is no issue of selection since the candidate list contains only the problem (P). However, as the algorithm proceeds, there would be many problems on the candidate list and a selection rule is required. Appropriate selection rules, also referred to as branching strategies, are discussed later. Conceptually, several problems may be simultaneously selected and removed from the candidate list. However, most sequential implementations of B & B select only one problem from the candidate list and this is assumed henceforth. Parallel aspects of B & B on $0-1$ integer linear programs are discussed in Cannon and Hoffman [18] and for the case of traveling salesman problems in [4].

The computational time required for the B & B algorithm depends crucially on the order in which the problems in the candidate list are examined. A number of clever heuristic rules may be employed in devising such strategies. Two general purpose selection strategies that are commonly used are

(A) Choose the problem that was added last to the candidate list. This last-in-first-out rule $(LIFO)$ is also called depth first search (DFS) since the selected candidate problem increases the depth of the active enumeration tree.

(B) Choose the problem on the candidate list that has the least lower bound. Ties may be broken by choosing the problem that was added last to the candidate list. This rule would require that a lower bound be obtained for each of the problems on the candidate list. In other words, when a problem is added to the candidate list, an associated lower bound should also be stored. This

may be accomplished by using ad-hoc rules or by solving a relaxation of each problem before it is added to the candidate list.

Rule (A) is known to empirically dominate the rule (B) when storage requirements for candidate list and computation time to solve (P) are taken into account. However, some analysis indicates that rule (B) can be shown to be superior if minimizing the number of candidate problems to be solved is the criterion (see Parker and Rardin [105]).

`Relaxation`

In order to analyze the selected candidate problem, (CP), a relaxation (CP_R) of (CP) is solved to obtain a lower bound $z(CP_R) \leq z(CP)$. (CP_R) is a relaxation of (CP) if

(i) $F(CP) \subseteq F(CP_R)$,

(ii) for $\bar{\mathbf{x}} \in F(CP)$, $z_{CP_R}(\bar{\mathbf{x}}) \leq z_{CP}(\bar{\mathbf{x}})$ and

(iii) for $\bar{\mathbf{x}}, \hat{\mathbf{x}} \in F(CP)$, $z_{CP_R}(\bar{\mathbf{x}}) \leq z_{CP_R}(\hat{\mathbf{x}})$ implies that $z_{CP}(\bar{\mathbf{x}}) \leq z_{CP}(\hat{\mathbf{x}})$.

Relaxations are needed because the candidate problems are typically hard to solve. The relaxations used most often are either linear programming or Lagrangean relaxations of (CP), see Section 6 for details. Sometimes, instead of solving a relaxation of (CP), a lower bound is obtained by using some ad-hoc rules such as penalty functions.

`Fathoming`

A candidate problem is fathomed if

(FC1) analysis of (CP_R) reveals that (CP) is infeasible. For instance if $F(CP_R) = \phi$, then $F(CP) = \phi$.

(FC2) analysis of (CP_R) reveals that (CP) has no feasible solution better than the current incumbent solution. For instance if $z(CP_R) \geq z_I$, then $z(CP) \geq z(CP_R) \geq z_I$.

(FC3) analysis of (CP_R) reveals an optimal solution of (CP). For instance, if $\mathbf{x}_R$ is optimal for (CP_R) and is feasible in (CP), then $(\mathbf{x}_R)$ is an optimal solution to (CP) and $z(CP) = z_{CP}(\mathbf{x}_R)$.

(FC4) analysis of (CP_R) reveals that (CP) is dominated by some other problem, say CP^*, in the candidate list. For instance if it can shown that $z(CP^*) \leq z(CP)$, then there is no need to analyze (CP) further.

If a candidate problem (CP) is fathomed using any of the criteria above, then further examination of (CP) or its descendants (sub-problems) obtained by separation is not required. If (FC3) holds, and $z(CP) < z_I$, the incumbent is updated as $\mathbf{x}_R$ and z_I is updated as $z(CP)$.

Separation/Branching: If the candidate problem (CP) is not fathomed, then CP is separated into several problems, say $(CP_1), (CP_2), \cdots, (CP_q)$ where $\bigcup_{t=1}^{q} F(CP_t) = F(CP)$ and typically

$$F(CP_i) \cap F(CP_j) = \phi \quad \forall i \neq j.$$

For instance a separation of (CP) into $(CP_i), i = 1, 2, \cdots, q$ is obtained by fixing a single variable, say x_j, to one of the q possible values of x_j in an optimal solution to (CP). The choice of the variable to fix depends upon the separation strategy which is also part of the branching strategy. After separation, the sub-problems are added to the candidate list. Each sub-problem (CP_t) is a restriction of (CP) since $F(CP_t) \subseteq F(CP)$. Consequently $z(CP) \leq z(CP_t)$ and $z(CP) = \min_t z(CP_t)$.

The various steps in the B & B algorithm are outlined below.

Procedure: B & B

0. **Initialize:** Given the problem (P), the incumbent value z_I is obtained by applying some heuristic (if a feasible solution to (P) is not available, set $z_I = +\infty$). Initialize the candidate list $C \leftarrow \{(P)\}$.

1. **Optimality:** If $C = \emptyset$ and $z_I = +\infty$, then (P) is infeasible, stop. Stop also if $C = \emptyset$ and $z_I < +\infty$, the incumbent is an optimal solution to (P).

2. **Selection:** Using some candidate selection rule, select and remove a problem $(CP) \in C$.

3. **Bound:** Obtain a lower bound for (CP) by either solving a relaxation (CP_R) of (CP) or by applying some ad-hoc rules. If (CP_R) is infeasible, return to Step 1. Else, let x_R be an optimal solution of (CP_R).

4. **Fathom:** If $z(CP_R) \geq z_I$, return to Step 1. Else if x_R is feasible in (CP) and $z(CP) < z_I$, set $z_I \leftarrow z(CP)$, update the incumbent as x_R and return to Step 1. Finally, if x_R is feasible in (CP) but $z(CP) \geq z_I$, return to Step 1.

5. **Separation:** Using some separation or branching rule, separate (CP) into $(CP_i), i = 1, 2, \cdots, q$ and set $C \leftarrow C \cup \{(CP_1), (CP_2), \cdots, (CP_q)\}$ and return to Step 1.

6. **End Procedure.**

Although the B & B method is easy to understand, the implementation of this scheme for a particular ILP is a nontrivial task requiring

(A) a relaxation strategy with efficient procedures for solving these relaxations.

(B) efficient data-structures for handling the rather complicated book-keeping of the candidate list

(C) clever strategies for selecting promising candidate problems and

(D) separation or branching strategies that could effectively prune the enumeration tree.

A key problem is that of devising a relaxation strategy (A), i.e. to find "good relaxations" which are significantly easier to solve than the original problems and tend to give sharp lower bounds. Since these two are conflicting, one has to find a reasonable trade-off.

5.1.1 Branch and Cut

In the last few years, the branch and cut (B & C) method has become popular for solving combinatorial optimization problems. As the name suggests, the B & C method incorporates the features of both the branch and bound method presented above and the cutting plane method presented in the previous section. The main difference between the B & C method and the general B & B scheme is in the bound step (Step 3).

A distinguishing feature of the B & C method is that the relaxation (CP_R) of the candidate problem (CP) is a linear programming problem and instead of merely solving (CP_R), an attempt is made to solve (CP) by using cutting planes to tighten the relaxation. If (CP_R) contains inequalities that are valid for (CP) but not for the given ILP, then the GC rounding procedure may generate inequalities that are valid for (CP) but not for the ILP. In the B & C method, the inequalities that are generated are always valid for the ILP, and hence can be used globally in the enumeration tree.

Another feature of the B & C method is that often heuristic methods are used to convert some of the fractional solutions, encountered during the cutting plane phase, into feasible solutions of the (CP) or more generally of the given ILP. Such feasible solutions naturally provide upper bounds for the ILP. Some of these upper bounds may be better than the previously identified best upper bound and if so, the current incumbent is updated accordingly.

We thus obtain the B & C method by replacing the bound step (Step 3) of the B & B method by Steps 3(a) and 3(b), and also by replacing the fathom step (Step 4) by Steps 4(a) and 4(b) given below.

3(a) **Bound:** Let (CP_R) be the LP relaxation of (CP). Attempt to solve (CP) by a cutting plane method which generates valid inequalities for (P). Update the constraint System of (P) and the incumbent as approriate.

Let $F\mathbf{x} \leq \mathbf{f}$ denote all the valid inequalities generated during this phase. Update the constraint system of (P) to include all the generated inequalities, i.e. set $A^T \leftarrow (A^T, F^T)$ and $\mathbf{b}^T \leftarrow (\mathbf{b}^T, \mathbf{f}^T)$. The constraints for all the problems in the candidate list are also to be updated.

During the cutting plane phase, apply heuristic methods to convert some of the identified fractional solutions into feasible solutions to (P). If a feasible solution, $\bar{\mathbf{x}}$, to (P), is obtained such that $\mathbf{c}\bar{\mathbf{x}} < z_I$, update the incumbent to $\bar{\mathbf{x}}$ and z_I to $c\bar{\mathbf{x}}$. Hence the remaining changes to B & B are as follows:

3(b) If (CP) is solved go to Step 4(a). Else, let $\hat{\mathbf{x}}$ be the solution obtained when the cutting plane phase is terminated, (we are unable to identify a valid inequality of (P) that is violated by $\hat{\mathbf{x}}$.) go to Step 4(b).

4(a) **Fathom by Optimality:** Let $\mathbf{x}^*$ be an optimal solution to (CP). If $z(CP) < z_I$, set $\mathbf{x}_I \leftarrow z(CP)$ and update the incumbent as $\mathbf{x}^*$. Return to Step 1.

4(b) **Fathom by Bound:** If $\mathbf{c}\hat{\mathbf{x}} \geq z_I$, return to Step 1. Else go to step 5.

The incorporation of a cutting plane phase into the B & B scheme involves several technicalities which require careful design and implementation of the B & C algorithm. Details of the state of the art in cutting plane algorithms including the B & C algorithm are reviewed in Junger, Reinelt and Thienel[72].

6 Relaxations

The effectiveness of a partial enumeration strategy like branch and bound is closely related to the quality of the relaxations used to generate the bounds and incumbents. We describe four general relaxation methods that together cover the most successful computational techniques for bound evaluation in the practice of partial enumeration for integer programming. These are linear programming relaxation, Lagrangean relaxation, group relaxation and semi-definite relaxation methods.

6.1 LP Relaxation

A linear programming relaxation is derived from an integer programming formulation by relaxing the integrality constraints. When there are alternate integer programming formulations, modeling

the same decision problem, it becomes necessary to have some criteria for selecting from among the candidate relaxations. We illustrate these ideas on the plant location model, a prototypical integer programming example.

Plant Location Problems

Given a set of customer locations $N = \{1, 2, .., n\}$ and a set of potential sites for plants $M = \{1, 2, .., m\}$, the plant location problem is to identify the sites where the plants are to be located so that the customers are served at a minimum cost. There is a fixed cost f_i of locating the plant at site i and the cost of serving customer j from site i is c_{ij}. The decision variables are

y_i is set to 1 if a plant is located at site i and to 0 otherwise.

x_{ij} is set to 1 if site i serves customer j and to 0 otherwise.

A formulation of the problem is

$$
\begin{array}{lll}
(P8) \quad \text{Min} & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i & \\
\text{subject to} & \sum_{i=1}^{m} x_{ij} = 1 & j = 1, 2, .., n \\
& x_{ij} - y_i \leq 0 & i = 1, 2, .., m;\ j = 1, 2, .., n \\
& y_i = 0 \text{ or } 1 & i = 1, 2, .., m \\
& x_{ij} = 0 \text{ or } 1 & i = 1, 2, .., m;\ j = 1, 2, .., n
\end{array}
$$

Note that the constraints $x_{ij} - y_i \leq 0$ are required to ensure that customer j may be served from site i only if a plant is located at site i. Note that the constraints $y_i = 0$ or 1, force an optimal solution in which $x_{ij} = 0$ or 1. Consequently the $x_{ij} = 0$ or 1 constraints may be replaced by non-negativity constraints $x_{ij} \geq 0$.

The linear programming relaxation associated with (P8) is obtained by replacing constraints $y_i = 0$ or 1, and $x_{ij} = 0$ or 1 by non-negativity contraints on x_{ij} and y_i. The upper bound constraints on y_i are not required provided $f_i \geq 0$, $i = 1, 2, .., m$. The upper bound constraints on x_{ij} are not required in view of constraints $\sum_{i=1}^{m} x_{ij} = 1$.

Remark:

It is frequently possible to formulate the same combinatorial problem as two or more different ILPs. Suppose we have two ILP formulations (F1) and (F2) of the given combinatorial problem with both (F1) and (F2) being minimizing problems. Formulation (F1) is said to be stronger than (F2) if (LP1), the the linear programming relaxation of (F1), always has an optimal objective function value which is greater than or equal to the optimal objective function value of (LP2) which is the linear programming relaxation of (F2).

It is possible to reduce the number of constraints in (P8) by replacing the constraints $x_{ij} - y_i \leq 0$ by an aggregate:

$$\sum_{j=1}^{n} x_{ij} - ny_i \leq 0 \;\; i = 1, 2, .., m$$

However, the disaggregated (P8) is a stronger formulation than the formulation obtained by aggregrating the constraints as above. By using standard transformations, (P8) can also be converted into a set packing problem.

6.2 Lagrangean Relaxation

This approach has been widely used for about two decades now in many practical applications. Lagrangean relaxation, like linear programming relaxation, provides bounds on the combinatorial optimization problem being relaxed (ie. lower bounds for minimization problems).

Lagrangean relaxation has been so successful because of a couple of distinctive features. As was noted earlier, in many hard combinatorial optimization problems, we usually have some "nice" tractable embedded subproblems which admit efficient algorithms. Lagrangean relaxation gives us a framework to "jerry-rig" an approximation scheme that uses these efficient algorithms for the subproblems as subroutines. A second observation is that it has been empirically observed that well-chosen Lagrangean relaxation strategies usually provide very tight bounds on the optimal objective value of integer programs. This is often used to great advantage within partial enumeration schemes to get very effective pruning tests for the search trees.

Practitioners also have found considerable success with designing heuristics for combinatorial optimization by starting with solutions from Lagrangean relaxations and constructing good feasible solutions via so-called "dual ascent" strategies. This may be thought of as the analogue of rounding strategies for linear programming relaxations (but with no performance guarantees - other than empirical ones).

Consider a representation of our combinatorial optimization problem in the form:

$$(P) \quad z = \min\{\mathbf{cx} : A\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \in X \subseteq \Re^n\}$$

Implicit in this representation is the assumption that the explicit constraints ($A\mathbf{x} \geq \mathbf{b}$) are *"small"* in number. For convenience let us also assume that that X can be replaced by a finite list $\{\mathbf{x}^1, \mathbf{x}^2, ...\mathbf{x}^T\}$.

The following definitions are with respect to (P)

- **Lagrangean** $L(\mathbf{u}, \mathbf{x}) = \mathbf{u}(A\mathbf{x} - \mathbf{b}) + \mathbf{cx}$ where $\mathbf{u}$ are the Lagrange multipliers.
- **tt Lagrangean Subproblem** $\min_{\mathbf{x} \in X}\{L(\mathbf{u}, \mathbf{x})\}$

- Lagrangean-Dual Function $\mathcal{L}(\mathbf{u}) = \min_{\mathbf{x}\in X}\{L(\mathbf{u},\mathbf{x})\}$
- Lagrangean-Dual Problem (D) $d = \max_{\mathbf{u}\geq 0}\{\mathcal{L}(\mathbf{u})\}$

It is easily shown that (D) satisfies a weak duality relationship with respect to (P), i.e., $z \geq d$. The discreteness of X also implies that $\mathcal{L}(\mathbf{u})$ is a piece-wise linear and concave function (see Shapiro [115]). In practice, the constraints X are chosen such that the evaluation of the Lagrangean Dual function $\mathcal{L}(\mathbf{u})$ is easily made.

An Example: Traveling Salesman Problem (TSP)

For an undirected graph G, with costs on each edge, the TSP is to find a minimum cost set H of edges of G such that it forms a Hamiltonian cycle of the graph. H is a Hamiltonian cycle of G if it is a simple cycle that spans all the vertices of G. Alternately H must satisfy:

1. exactly two edges of H are adjacent to each node, and
2. H forms a connected, spanning subgraph of G.

Held and Karp [63] used these observations to formulate a Lagrangean relaxation approach for TSP, that relaxes the degree constraints (1). Notice that the resulting subproblems are minimum spanning tree problems which can be easily solved.

The most commonly used general method of finding the optimal multipliers in Lagrangean relaxation is subgradient optimization (cf. Held et al. [62]). Subgradient optimization is the non-differentiable counterpart of steepest descent methods. Given a dual vector $\mathbf{u}^k$, the iterative rule for creating a sequence of solutions is given by:

$$\mathbf{u}^{k+1} = \mathbf{u}^k + t_k\,\gamma(\mathbf{u}^k)$$

where t_k is an appropriately chosen step size, and $\gamma(\mathbf{u}^k)$ is a subgradient of the dual function $\mathcal{L}$ at $\mathbf{u}^k$. Such a subgradient is easily generated by

$$\gamma(\mathbf{u}^k) = A\mathbf{x}^k - \mathbf{b}$$

where $\mathbf{x}^k$ is an optimal solution of $\min_{\mathbf{x}\in X}\{L(\mathbf{u}^k,\mathbf{x})\}$.

Subgradient optimization has proven effective in practice for a variety of problems. It is possible to choose the step sizes $\{t_k\}$ to guarantee convergence to the optimal solution. Unfortunately, the method is not finite, in that the optimal solution is attained only in the limit. Further, it is not a pure descent method. In practice, the method is heuristically terminated and the best solution in the generated sequence is recorded. In the context of non-differentiable optimization, the Ellipsoid Algorithm was devised by Shor [118] to overcome precisely some of these difficulties with the subgradient method.

The Ellipsoid Algorithm may be viewed as a scaled subgradient method in much the same way as variable metric methods may be viewed as scaled steepest descent methods (cf. [2]). And if we use the Ellipsoid method to solve the Lagrangean dual problem, we obtain the following as a consequence of the polynomial-time equivalence of optimization and separation.

Theorem 6.1 *The Lagrangean dual problem is polynomial-time solvable if and only if the Lagrangean subproblem is. Consequently, the Lagrangean dual problem is $\mathcal{NP}$-Hard if and only if the Lagrangean subproblem is.*

The theorem suggests that in practice, if we set up the Lagrangean relaxation so that the subproblem is tractable, then the search for optimal Lagrangean multipliers is also tractable.

6.3 Group Relaxations

A relaxation of the integer programming problem is obtained by dropping the non-negativity restrictions on some variables. Consider the integer linear program

$$(ILP) \quad \max\{\mathbf{cx} : A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \text{ and integer}\}$$

where the elements of A, b are integral.

Suppose the linear programming relaxation of (ILP) has a finite optimum, then an extreme-point optimal solution $\mathbf{x}^* = \begin{pmatrix} B^{-1}\mathbf{b} \\ \mathbf{0} \end{pmatrix}$ where B is a non-singular submatrix of A. Let $A = (B, N)$, $\mathbf{x} = \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{pmatrix}$, and $\mathbf{c} = (\mathbf{c}_B, \mathbf{c}_N)$. By dropping the non-negativity constraint on $\mathbf{x}_B$, substituting $\mathbf{x}_B = B^{-1}(\mathbf{b} - N\mathbf{x}_N)$, ignoring the constant term $\mathbf{c}_B B^{-1} b$ in the objective function, and changing the objective function from maximize to minimize, we obtain the following relaxation of (ILP).

$$(ILP_B) \quad \min\{(\mathbf{c}_B B^{-1} N - \mathbf{c}_N)\mathbf{x}_N : \mathbf{x}_B = B^{-1}(\mathbf{b} - N\mathbf{x}_N),\ \mathbf{x}_B \text{ integer},\ \mathbf{x}_N \geq \mathbf{0} \text{ and integer}\}$$

Now $\mathbf{x}_B = B^{-1}(\mathbf{b} - N\mathbf{x}_N)$ and $\mathbf{x}_B$ integer is equivalent to requiring

$$B^{-1}N\mathbf{x}_N \equiv B^{-1}\mathbf{b}(mod\ 1)$$

where the congruence is with respect to each element of the vector $B^{-1}\mathbf{b}$ taken modulo 1.

Hence (ILP_B) can be written as

$$(ILP_G) \quad \min\{(\mathbf{c}_B B^{-1} N - \mathbf{c}_N)\mathbf{x}_N : B^{-1}N\mathbf{x}_N \equiv B^{-1}\mathbf{b}(mod\ 1),\ \mathbf{x}_N \geq \mathbf{0} \text{ and integer}\}$$

Since A and $\mathbf{b}$ are integer matrices, the fractional parts of elements of $B^{-1}N$ and $B^{-1}\mathbf{b}$ are of the form $\left(\frac{k}{detB}\right)$ where $k \in \{0, 1, \ldots, |detB| - 1\}$. The congruences in (ILP_G) are equivalent to working

in a product of cyclic groups. The structure of the product group is revealed by the Hermite Normal Form of B (see Section 2.2). Hence Hence (ILP_G) is referred to as a group (knapsack) problem and is solved by a dynamic programming algorithm [58].

6.4 Semi-Definite Relaxation

Semi-definite programs are linear optimization problems defined over a cone of positive semi-definite matrices. These are models that generalize linear programs and are specializations of convex programming models. There are theoretical and "practical" algorithms for solving semi-definite programs in polynomial time [3]. Lovász and Schrijver [91] suggest a general relaxation strategy for $0-1$ integer programming problems that obtains semi-definite relaxations. The first step is to consider a homogenized version of a $0-1$ integer program (solvability version).

$$F_I = \{\mathbf{x} \in \Re^{n+1} : A\mathbf{x} \geq 0,\ x_0 = 1,\ x_i \in \{0,1\} \text{ for } i = 1,2,\cdots,n\}$$

The problem is to check if F_I is non-empty. Note that any $0-1$ integer program can be put in this form by absorbing a general right-hand-side b as the negative of x_0 column of A. Now a linear programming relaxation of this integer program is given by:

$$\{\mathbf{x} \in \Re^{n+1} : A\mathbf{x} \geq 0,\ x_0 = 1,\ 0 \leq x_i \leq x_0 \text{ for } i = 1,2,\cdots,n\}$$

Next, we define two polyhedral cones.

$$\mathcal{K} = \{\mathbf{x} \in \Re^{n+1} : A\mathbf{x} \geq 0,\ 0 \leq x_i \leq x_0 \text{ for } i = 0,1,\cdots,n\}$$

$$\mathcal{K}_I = \text{Cone generated by } 0-1 \text{ vectors in } P_I$$

Lovász and Schrijver [91] show how we might construct a family of convex cones $\{\mathcal{C}\}$ such that $\mathcal{K}_I \subseteq \mathcal{C} \subseteq \mathcal{K}$ for each $\mathcal{C}$.

(i) Partition the cone constraints of $\mathcal{K}$ into $T_1 = \{A_1\mathbf{x} \geq 0\}$ and $T_2 = \{A_2\mathbf{x} \geq 0\}$, with the constraints $\{0 \leq x_i \leq x_0 \text{ for } i = 0,1,\cdots,n\}$ repeated in both (the overlap can be larger).

(ii) Multiply each constraint in T_1 with each constraint in T_2 to obtain a quadratic homogeneous constraint.

(iii) Replace each occurance of a quadratic term x_ix_j by a new variable X_{ij}. The quadratic constraints are now linear homogeneous constraints in X_{ij}.

(iv) Add the requirement that the $(n+1)\times(n+1)$ matrix X is symmetric and positive semi-definite.

(v) Add the constraints $X_{0i} = X_{ii}$ for $i = 1,2,\cdots,n$.

The system of constraints on the X_{ij} constructed in steps (iii), (iv) and (v) above define a cone $M_+(T_1, T_2)$ parametrized by the partition T_1, T_2. We finally project the cone $M_+(T_1, T_2)$ to $\Re^{n+1}$ as follows.

$$C_+(T_1, T_2) = \text{ Diagonals of matrices in } M_+(T_1, T_2)$$

These resulting cones $\{C_+(T_1, T_2)\}$ satisfy

$$\mathcal{K}_I \subseteq C_+(T_1, T_2) \subseteq \mathcal{K}$$

where the X_{ii} in $C_+(T_1, T_2)$ are interpreted as the original x_i in $\mathcal{K}$ and $\mathcal{K}_I$.

One semi-definite relaxation of the $0-1$ integer program F_I is just $C_+(T_1, T_2)$ along with the normalizing constraint $X_{00} = x_0 = 1$. For optimization versions of integer programming, we simply carry over the objective function. An amazing result obtained by Lovász and Schrijver [91] is that this relaxation when applied to the vertex packing polytope (see Section 3.4) is atleast as good as one obtained by adding "clique, odd hole, odd antihole and odd wheel" valid inequalities (see [61] for the definitions) to the linear programming relaxation of the set packing formulation. This illustrates the power of this approach to reveal structure and yet obtain a tractable relaxation. In particular, it also implies a polynomial-time algorithm for the vertex packing problem on perfect graphs (cf. [61]). Another remarkable success that semi-definite relaxations have registered is the recent result on finding an approximately maximum weight edge cutset of a graph. This result of Goemans and Williamson [55] will be described in the next section. While the jury is still out on the efficacy of semi-definite relaxation as a general strategy for integer programming, there is little doubt that it provides an exciting new weapon in the arsenal of integer programming methodologies.

7 Approximation with Performance Guarantees

The relaxation techniques' we encountered in the previous section are designed with the intent of obtaining a "good" lower (upper) bound on the optimum objective value for a minimization (maximization) problem. If in addition, we are able to construct a "good" feasible solution using a heuristic (possibly based on a relaxation technique) we can use the bound to quantify the suboptimality of the incumbent and hence the "quality of the approximation".

In the past few years, there has been significant progress in our understanding of performance guarantees for approximation of $\mathcal{NP}$-Hard combinatorial optimization problems (cf.[117]). A ρ-approximate algorithm for an optimization problem is an approximation algorithm that delivers a feasible solution with objective value within a factor of ρ of optimal (think of minimization problems and $\rho \geq 1$). For some combinatorial optimization problems, it is possible to *efficiently* find solutions that are arbitrarily close to optimal even though finding the true optimal is hard. If this were true

of most of the problems of interest we would be in good shape. However, the recent results of Arora et al. [5] indicate exactly the opposite conclusion.

A PTAS or polynomial-time approximation scheme for an optimization problem is a family of algorithms A_ρ, such that for each $\rho > 1$, A_ρ is a polynomial-time ρ-approximate algorithm. Despite concentrated effort spanning about two decades, the situation in the early 90's was that for many combinatorial optimization problems, we had no PTAS and no evidence to suggest the non-existence of such schemes either. This led Papadimitriou and Yannakakis [104] to define a new complexity class (using reductions that preserve approximate solutions) called MAXSNP and they identified several complete languages in this class. The work of Arora et al. completed this agenda by showing that, assuming $\mathcal{P} \neq \mathcal{NP}$, there is no PTAS for a MAXSNP-complete problem.

An implication of these theoretical developments is that for most combinatorial optimization problems, we have to be quite satisfied with performance guarantee factors $\rho \geq 1$ that are of some small fixed value. (There are problems, like the general traveling salesman problem, for which there are no ρ-approximate algorithms for any finite value of ρ - of course assuming $\mathcal{P} \neq \mathcal{NP}$). Thus one avenue of research is to go problem by problem and knock ρ down to its smallest possible value.

A different approach would be to look for other notions of "good approximations" based on probabilistic guarantees or empirical validation. A good example of the benefit to be gained from randomization is the problem of computing the volume of a convex body. Dyer and Frieze [37] have shown that this problem is $\#\mathcal{P}$-hard. Barany and Furedi'[8] provide evidence that no polynomial-time deterministic approximation method with relative error less than $(cn)^{\frac{n}{2}}$, where c is a constant, is likely to exist. However, Dyer et al. [38] have designed a fully-polynomial randomized approximation scheme (FPRAS) for this problem. The FPRAS of Dyer et al. citeDFK uses techniques from integer programming and the geometry of numbers (see Section 8).

LP Relaxation and Rounding

Consider the well known problem of finding the *smallest weight vertex cover* in a graph (see Section 3.1). So we are given a graph $G(V, E)$ and a nonnegative weight $w(v)$ for each vertex $v \in V$. We want to find the smallest total weight subset of vertices S such that each edge of G has at least one end in S (This problem is known to be MAXSNP-Hard). An integer programming formulation of this problem is given by

$$\min\{\sum_{v \in V} w(v)x(v) : x(u) + x(v) \geq 1,\ \forall (u,v) \in E,\quad x(v) \in \{0,1\}\ \forall v \in V\}$$

To obtain the linear programming relaxation we substitute the $x(v) \in \{0,1\}$ constraint with $x(v) \geq 0$ for each $v \in V$. Let $\mathbf{x}^*$ denote an optimal solution to this relaxation. Now let us round the fractional parts of $\mathbf{x}^*$ in the usual way, that is, values of 0.5 and up are rounded to 1 and smaller

values to 0. Let $\hat{x}$ be the 0–1 solution obtained. First note that $\hat{x}(v) \leq 2x^*(v)$ for each $v \in V$. Also, for each $(u,v) \in E$, since $x^*(u) + x^*(v) \geq 1$, at least one of $x(u)$ and $x(v)$ must be set to 1. Hence $\hat{x}$ is the incidence vector of a vertex cover of G whose total weight is within twice the total weight of the linear programming relaxation (which is a lower bound on the weight of the optimal vertex cover). Thus we have a 2-approximate algorithm for this problem which solves a linear programming relaxation and uses rounding to obtain a feasible solution.

The deterministic rounding of the fractional solution worked quite well for the vertex cover problem. One gets a lot more power from this approach by adding in randomization to the rounding step. Raghavan and Thompson [108] proposed the following obvious randomized rounding scheme. Given a $0-1$ integer program, solve its linear programming relaxation to obtain an optimal $\mathbf{x}^*$. Treat the $x_j^* \in [0,1]$ as probabilities , i.e. let Probability$\{x_j = 1\}= x_j^*$, to randomly round the fractional solution to a $0-1$ solution. Using Chernoff bounds on the tails of the Binomial distribution, they were able to show, for specific problems, that with high probability, this scheme produces integer solutions which are close to optimal. In certain problems, this rounding method may not always produce a feasible solution. In such cases, the expected values have to be computed as conditioned on feasible solutions produced by rounding. More complex (non-linear) randomized rounding schemes have been recently studied and have been found to be extremely effective. We will see an example of non-linear rounding in the context of semi-definite relaxations of the max-cut problem below.

Primal Dual Approximation

The linear programming relaxation of the vertex cover problem, we saw above, is given by

$$(P_{VC}) \quad \min\{\sum_{v \in V} w(v)x(v) : x(u) + x(v) \geq 1,\ \forall (u,v) \in E,\quad x(v) \geq 0\ \forall v \in V\}$$

and its dual is

$$(D_{VC}) \quad \max\{\sum_{(u,v) \in E} y(u,v) : \sum_{u|(u,v) \in E} y(u,v) \leq w(v),\ \forall v \in V,\quad y(u,v) \geq 0\ \forall (u,v) \in E\}$$

The primal-dual approximation approach would first obtain an optimal solution y^* to the dual problem (D_{VC}). Let $\hat{V} \subseteq V$ denote the set of vertices for which the dual constraints are tight, i.e.,

$$\hat{V} = \{v \in V : \sum_{u|(u,v) \in E} y^*(u,v) = w(v)\}$$

The approximate vertex cover is taken to be $\hat{V}$. It follows from complementary slackness that $\hat{V}$ is a vertex cover. Using the fact that each edge (u,v) is in the star of at most two vertices (u and v), it also follows that $\hat{V}$ is a 2-approximate solution to the minimum weight vertex cover problem.

In general, the primal-dual approximation strategy is to use a dual solution, to the linear programming relaxation, along with complementary slackness conditions as a heuristic to generate an

integer (primal) feasible solution which for many problems turns out to be a good approximation of the optimal solution to the original integer program.

It is a remarkable property of the vertex covering (and packing) problem that all extreme points of the linear programming relaxation have values 0, $\frac{1}{2}$ or 1 [94]. It follows that the deterministic rounding of the linear programming solution to (P_{VC}) constructs exactly the same approximate vertex cover as the primal-dual scheme described above. However, this is not true in general.

Semi-Definite Relaxation and Rounding

The idea of using semi-definite programming to approximately solve combinatorial optimization problems appears to have originated in the work of Lovász [90] on the Shannon capacity of graphs. Grötschel, Lovász and Schrijver [61] later used the same technique to compute a maximum stable set of vertices in perfect graphs via the ellipsoid method. As we saw in Section 7.4, Lovasz and Schrijver [91] have devised a general technique of semi-definite relaxations for general $0-1$ integer linear programs. We will present a lovely application of this methodology to approximate the the maximum weight cut of a graph (the maximum sum of weights of edges connecting across all strict partitions of the vertex set). This application of semi-definite relaxation for approximating MAXCUT is due to Goemans and Williamson [55].

We begin with a quadratic Boolean formulation of MAXCUT

$$\max\{\frac{1}{2}\sum_{(u,v)\in E} w(u,v)(1-x(u)x(v)): \quad x(v)\in\{-1,1\}\ \forall\ v\in V\}$$

where $G(V,E)$ is the graph and $w(u,v)$ is the non-negative weight on edge (u,v). Any $\{-1,1\}$ vector of $\mathbf{x}$ values provides a bipartition of the vertex set of G. The expression $(1-x(u)x(v))$ evaluates to 0 if u and v are on the same side of the bipartition and 2 otherwise. Thus, the optimization problem does indeed represent exactly the MAXCUT problem.

Next we reformulate the problem in the following way.

- We square the number of variables by allowing each $\mathbf{x}(v)$ to denote an n-vector of variables (where n is the number of vertices of the graph).
- The quadratic term $x(u)x(v)$ is replaced by $\mathbf{x}(u)\cdot\mathbf{x}(v)$ which is the inner product of the vectors.
- Instead of the $\{-1,1\}$ restriction on the $x(v)$, we use the Euclidean normalization $\|\mathbf{x}(v)\|=1$ on the $\mathbf{x}(v)$.

So we now have a problem

$$\max\{\frac{1}{2}\sum_{(u,v)\in E} w(u,v)(1-\mathbf{x}(u)\cdot\mathbf{x}(v)): \quad \|\mathbf{x}(v)\|=1\ \forall\ v\in V\}$$

which is a relaxation of the MAXCUT problem (note that if we force only the first component of the $\mathbf{x}(v)$'s to have nonzero value, we would just have the old formulation as a special case).

The final step is in noting that this reformulation is nothing but a semi-definite program. To see this we introduce $n \times n$ Gram matrix Y of the unit vectors $\mathbf{x}(v)$. So $Y = X^T X$ where $X = (\mathbf{x}(v) : v \in V)$. So the relaxation of MAXCUT can now be stated as a semi-definite program.

$$\max\{\frac{1}{2}\sum_{(u,v)\in E} w(u,v)(1 - Y_{(u,v)}) : \quad Y \succeq 0, \;\; Y_{(v,v)} = 1 \; \forall \; v \in V\}$$

Note that we are able to solve such semi-definite programs to an additive error ϵ in time polynomial in the input length and $\log \frac{1}{\epsilon}$ using either the Ellipsoid method or Interior Point methods (see [3] and Chapters **Editor: Please cross reference chapters on Linear programming, Approximation and Nonlinear Programming** of this handbook).

Let $\mathbf{x}^*$ denote the near optimal solution to the semi-definite programming relaxation of MAXCUT (convince yourself that $\mathbf{x}^*$ can be reconstructed from an optimal Y^* solution). Now we encounter the final trick of Goemans and Williamson. The approximate maximum weight cut is extracted from $\mathbf{x}^*$ by randomized rounding. We simply pick a random hyperplane H passing through the origin. All the $v \in V$ lying to one side of H get assigned to one side of the cut and the rest to the other. Goemans and Williamson observed the following inequality.

Lemma 7.1 *For $\mathbf{x}_1$ and $\mathbf{x}_2$, two random n-vectors of unit norm, let $x(1)$ and $x(2)$ be ± 1 values with opposing sign if H separates the two vectors and with same signs otherwise. Then $\tilde{E}(1 - \mathbf{x}_1{}^T\mathbf{x}_2) \leq 1.1393 \cdot \tilde{E}(1 - x(1)x(2))$ where $\tilde{E}$ denotes the expected value.*

By linearity of expectation, the lemma implies that the expected value of the cut produced by the rounding is at least 0.878 times the expected value of the semi-definite program. Using standard conditional probability techniques for derandomizing, Goemans and Williamson show that a deterministic polynomial-time approximation algorithm with the same margin of approximation can be realized. Hence we have a cut with value at least 0.878 of the maximum value.

8 Geometry of Numbers and Integer Programming

Given an integer program with a fixed number of variables (k) we seek a polynomial-time algorithm for solving them. Note that the complexity is allowed to be exponential in k, which is independent of the input length. Clearly if the integer program has all $(0,1)$ integer variables this is a trivial task, since complete enumeration works. However if we are given an "unbounded" integer program to begin with, the problem is no longer trivial.

8.1 Lattices, Short Vectors and Reduced Bases

Euclidean lattices are a simple generalization of the regular integer lattice $\mathcal{Z}^n$. A (point) *Lattice* is specified by $\{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$ a *basis* (where $\mathbf{b}_i$ are linearly independent n - dimensional rational vectors). The Lattice L is given by :

$$L = \{\mathbf{x} : \mathbf{x} = \sum_{i=1}^{n} z_i \mathbf{b}_i;\ \ z_i \in Z\ \forall i\}$$

$$B = (\mathbf{b}_1 \vdots \mathbf{b}_2 \vdots \cdots \vdots \mathbf{b}_n) \text{ a basis matrix of } L$$

Theorem 8.1 *$|detB|$ is an invariant property of the lattice L (ie for every basis matrix B_i of L we have invariant $d(L) = |detB_i|$).*

Note that

$$d(L) = \Pi_{i=1}^{n} |\mathbf{b}_i| \text{ where } |\mathbf{b}_i| \text{denotes the Euclidean Length of } \mathbf{b}_i$$

if and only if the basis vectors $\{\mathbf{b}_i\}$ are mutually orthogonal. A "sufficiently orthogonal" basis B (called a *reduced basis*) is one that satisfies a weaker relation

$$d(L) \leq c_n \Pi_{i=1}^{n} |\mathbf{b}_i| \text{ where } c_n \text{ is a constant that depends only on } n$$

One important use (from our perspective) of a reduced basis is that one of the basis vectors has to be "short". Note that there is sunstantive evidence that finding the shortest lattice vector is $\mathcal{NP}$-Hard [61]. Minkowski proved that in every lattice L there exists a vector of length no larger than $c\sqrt{n}\sqrt[n]{d(L)}$ (with c no larger than 0.32). This follows from the celebrated Minkowski's Convex Body Theorem which forms the centerpiece of the geometry of numbers [19].

Theorem 8.2 Minkowski's Convex Body Theorem: *If $K \subseteq \Re^n$ is a convex body that is centrally symmetric with respect to the origin, and $L \subseteq \Re^n$ is is a lattice such that $vol(K) \geq 2^n d(L)$ then K contains a lattice point different from the origin.*

However, no one has been successful thus far in designing an efficient algorithm (polynomial-time) for constructing the short lattice vector guaranteed by Minkowski. This is where the concept of a reduced basis comes to the rescue. We are able to construct a reduced basis in polynomial time and extract a short vector from it. To illustrate the concepts we will now discuss an algorithm due to Gauss (cf. [6]) that proves the theorem for the special case of planar lattices ($n = 2$). It is called the 60^o Algorithm because it produces a reduced basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ such that the acute angle between the two basis vectors is atleast 60^o.

Procedure: 60° Algorithm

Input: Basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ with $|\mathbf{b}_1| \geq |\mathbf{b}_2|$.

Output: A reduced basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ with at least 60° angle between the basis vectors.

0. repeat until $|\mathbf{b}_1| < |\mathbf{b}_2|$

1. swap $\mathbf{b}_1$ and $\mathbf{b}_2$

2. $\mathbf{b}_2 \leftarrow (\mathbf{b}_2 - m\mathbf{b}_1)$ and $m = \left[\frac{\mathbf{b}_2^T \mathbf{b}_1}{\mathbf{b}_1^T \mathbf{b}_1}\right] \in Z$.
   Here $[\alpha]$ denotes the integer nearest to α.

3. end

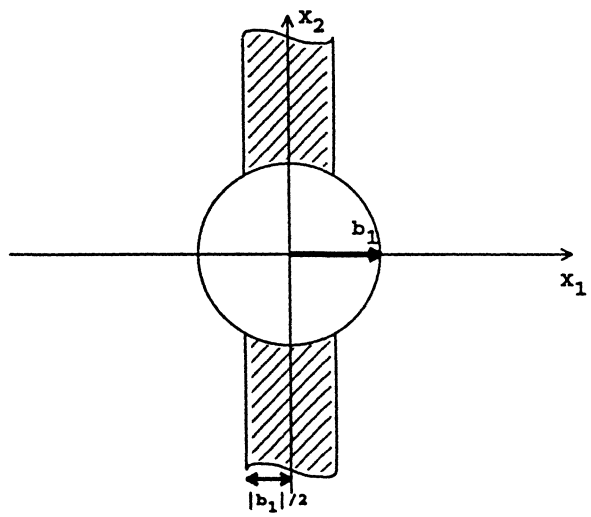


Figure 4: A Reduced Basis

REMARKS:

(i) In each iteration the projection of $(\mathbf{b}_2 - m\mathbf{b}_1)$ onto the direction of $\mathbf{b}_1$ is of length atmost $|\mathbf{b}_1|/2$.

(ii) When the algorithm stops $\mathbf{b}_2$ must lie in one of the shaded areas at the top or at the bottom of Figure 4 (since $|\mathbf{b}_2| > |\mathbf{b}_1|$ and since the projection of $\mathbf{b}_2$ must fall within the two vertical lines about $\mathbf{b}_1$).

(iii) The length of $\mathbf{b}_1$ strictly decreases in each iteration. Hence the algorithm is finite. That it is polynomial time takes a little more argument [82].

(iv) The short vector $\mathbf{b}_1$ produced by the algorithm satisfies $|\mathbf{b}_1| \leq (1.075)(d(L))^{\frac{1}{2}}$.

The only known polynomial-time algorithm for constructing a reduced basis in an arbitrary dimensional lattice [86] is not much more complicated than the 60^o algorithm. However, the proof of polynomiality is quite technical (see [6] for an exposition).

8.2 Lattice Points in a Triangle

Consider a triangle T in the Euclidean plane defined by the inequalities

$$a_{11}x_1 + a_{12}x_2 \leq d_1$$

$$a_{21}x_1 + a_{22}x_2 \leq d_2$$

$$a_{31}x_1 + a_{32}x_2 \leq d_3$$

The problem is to check if there is a lattice point $(x_1, x_2) \in Z^2$ satisfying the three inequalities. The reader should verify that checking all possible lattice points within some bounding box of T leads to an exponential algorithm for skinny and long triangles. There are many ways of realizing a polynomial-time search algorithm [73,46] for two-variable integer programs. Following Lenstra [88] we describe one that is a specialization of his [87] powerful algorithm for integer programming (searching for a lattice point in a polyhedron).

First we use a nonsingular linear transform τ that makes T equilateral (round). The transform sends the integer lattice Z^2 to a generic two-dimensional lattice L. Next we construct a reduced basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ for L using the 60^o algorithm. Let β_1 denote the length of the short vector $\mathbf{b}_1$. If l denotes the length of a side of the equilateral triangle $\tau(T)$ we can conclude that T is guaranteed to contain a lattice point if $\frac{l}{\beta_1} \geq \sqrt{6}$. Else $l < \sqrt{6}\beta_1$ and we can argue that just a few (no more than 3) of the lattice lines $\{\text{Affine Hull}\{\mathbf{b}_1\} + \mathbf{k}\mathbf{b}_2\}_{\mathbf{k}\in\mathbf{Z}}$ can intersect T. We recurse to the lower dimensional segments (lattice lines intersecting with T) and search each segment for a lattice point. Hence this scheme provides a simple polynomial-time algorithm for searching for a lattice point in a triangle.

8.3 Lattice Points in Polyhedra

In 1979, H.W. Lenstra Jr. announced that he had a polynomial-time algorithm for integer programming problems with a fixed number of variables. The final published version [87] of his algorithm resembles the procedure described above for the case of a triangle and a planar lattice. As we have noted before, integer programming is equivalent to searching a polyhedron specified by a system of linear inequalities for an integer lattice point. Lenstra's algorithm then proceeds as follows.

- First "round" the polyhedron using a linear transformation. This step can be executed in polynomial time (even for varying n) using any polynomial-time algorithm for linear programming.
- Find a reduced basis of the transformed lattice. This again can be done in polynomial time even for varying n.
- Use the reduced basis to conclude that a lattice point must lie inside or recurse to several lower dimensional integer programs by slicing up the polyhedron using a "long" vector of the basis to ensure that the number of slices depends only on n.

A slightly different approach based on Minkowski's convex body theorem was designed by Kannan [74] to obtain an $O(n^{\frac{9n}{2}}L$ algorithm for integer programming (where L is the length of the input). The following two theorems are straightforward consequences of these polynomial-time algorithms for integer programming with a fixed number of variables.

Theorem 8.3 (Fixed Number of Constraints) *Checking the solvability of*

$$A\mathbf{x} \leq \mathbf{b};\ \mathbf{x} \in Z^n$$

where A is $(m \times n)$ is solvable in polynomial time if m is held fixed.

Theorem 8.4 (Mixed Integer Programming) *Checking the solvability of*

$$A\mathbf{x} \leq \mathbf{b};\ x_j \in Z \text{ for } j = 1, \cdots, k;\ x_j \in \Re \text{ for } j = k+1, \cdots, n$$

where A is $(m \times n)$ is solvable in polynomial time if $\min\{m, k\}$ is held fixed.

A related but more difficult question is that of counting the number of feasible solutions to an integer program or equivalently counting the number of lattice points in a polytope. Building on results described above, Barvinok [9] was able to show the following.

Theorem 8.5 (Counting Lattice Points) *Counting the size of the set*

$$\{\mathbf{x} \in Z^k : A\mathbf{x} \leq \mathbf{b}\}$$

is solvable in polynomial time if k is held fixed.

8.4 An Application in Cryptography

In 1982, Adi Shamir [114] pointed out that Lenstra's algorithm could be used to devise a polynomial-time algorithm for cracking the basic Merkle-Hellman Cryptosystem (a knapsack based public-key cryptosystem). In such a cryptosystem the message to be sent is a (0-1) string $\bar{\mathbf{x}} \in \{0,1\}^n$. The

message is sent as an instance of the (0,1) knapsack problem which asks for an $x \in \{0,1\}^n$ satisfying $\sum_{i=1}^{n} a_i x_i = a_0$. The knapsack problem is an $\mathcal{NP}$-hard optimization problem (we saw in Section 3.4 that any integer program can be aggregated to a knapsack). However, if $\{a_i\}$ form a super-increasing sequence i.e., $a_i > \sum_{j=1}^{i-1} a_j \ \forall i$ the knapsack problem can be solved in time $O(n)$ by a simple algorithm:

Procedure: **Best-Fit**

0. $i \leftarrow n$

1. If $a_i \leq a_0$, $\bar{x}_i \leftarrow 1$, $a_0 \leftarrow a_0 - a_i$

2. $i \leftarrow (i-1)$

3. If $a_0 = 0$ stop and return the solution x

4. If $i = 1$ stop the knapsack is infeasible.

5. repeat 1.

However, an eavesdropper could solve this problem as well and hence the $\{a_i\}$ have to be encrypted. The disguise is chosen through two "secret" numbers M and U such that $M > \sum_{i=1}^{n} a_i$ and U is relatively prime to M (ie $gcd(U, M) = 1$). Instead of $\{a_i\}$, the sequence $\{\tilde{a}_i = Ua_i (\text{mod } M)\}_{i=1,2,\cdots,n}$ is published and $\tilde{a}_0 = Ua_0 (\text{mod } M)$ is transmitted.

Any receiver who *"knows" U* and M can easily reconvert $\{\tilde{a}_i\}$ to $\{a_i\}$ and apply the best fit algorithm to obtain the message $\bar{x}$. To find a_i given $\tilde{a}_i$, U and M, he runs the Euclidean Algorithm on (U, M) to obtain $1 = PU + QM$. Hence P is the inverse multiplier of U since $PU \equiv 1 \,(\text{mod } M)$. Using the identity $a_i \equiv P\tilde{a}_i \,(\text{mod } M) \ \forall i = 0, 1, \cdots, n$, the intended receiver can now use the best fit method to decode the message. An eavesdropper knows only the $\{\tilde{a}_i\}$ and is therefore supposedly unable to decrypt the message.

The objective of Shamir's cryptanalyst (code breaker) is to find a $\hat{P}$ and in $\hat{M}$ (Positive integer) such that :

$$\hat{a}_i \equiv \hat{P}\tilde{a}_i \text{ mod } \hat{M} \ \forall i = 0, 1, \cdots, n$$

(*) $\{\hat{a}_i\}_{i=1,\cdots,n}$ is a super increasing sequence.

$$\sum_{i=1}^{n} \hat{a}_i < \hat{M}$$

It can be shown that for all pairs $(\hat{P}, \hat{M})$ such that $(\hat{P}/\hat{M})$ is "sufficiently"close to (P/M) will satisfy (*). Using standard techniques from diophantine approximation it would be possible to guess P and M from the estimate $(\hat{P}/\hat{M})$. However, the first problem is to get the estimate of (P/M). This is where integer programming helps.

Since $a_i \equiv P\tilde{a}_i \pmod{M}$ for $i = 1, 2, \cdots, n$ we have $a_i P\tilde{a}_i - y_i M$ for some integers $y_1, y_2, \cdots, y_n$. Dividing by $\tilde{a}_i M$ we have $\left(\frac{a_i}{\tilde{a}_i M}\right) = \frac{P}{M} - \frac{y_i}{\tilde{a}_i}$ for $i = 1, 2, \cdots, n$. For small $i (1, 2, 3, \cdots, t)$ the LHS ≈ 0 since $\sum_{i=1}^{n} a_i < M$ and the $\{a_i\}_{i=1,\cdots,n}$ are super-increasing. Thus $(y_1/\tilde{a}_1), (y_2/\tilde{a}_2), \cdots, (y_t/\tilde{a}_t)$ are "close to" (P/M) and hence to each other. Therefore a natural approach to estimating (P/M) would be to solve the integer program:

(IP)
$$\boxed{\begin{array}{c} \text{Find } y_1, y_2, \cdots, y_t \in \mathcal{Z} \text{ such that :} \\ \underline{\epsilon_i} \leq \tilde{a}_i y_1 - \tilde{a}_1 y_i \leq \overline{\epsilon_i} \text{ for } i = 2, 3, \cdots, t \\ 0 < y_i < \tilde{a}_i \text{ for } i = 1, 2, \cdots, t \end{array}}$$

This t variable integer program provides an estimate of (P/M), whence Diophantine Approximation methods can be used to find $\hat{P}$ and $\hat{M}$. If we denote by d a density parameter of the instance where, $d = \frac{\log a_o}{n \log 2}$, the above scheme works correctly (with probability one as $n \to \infty$) if t is chosen to be $(\lfloor d \rfloor + 2)$ [114,83]. Moreover, the scheme is polynomial-time in n for fixed d. The probabilistic performance is not a handicap since as Shamir points out: "... a cryptosystem becomes useless when most of its keys can be efficiently cryptanalyzed" [114].

9 Prospects in Integer Programming

The current emphasis in software design for integer programming is in the development of shells (for example CPLEX [33], MINTO [111] and OSL [68]) wherein a general purpose solver like Branch & Cut is the driving engine. Problem specific code for generation of cuts and facets can be easily interfaced with the engine. We believe that this trend will eventually lead to the creation of general purpose problem solving languages for combinatorial optimization akin to AMPL [48] for linear and nonlinear programming.

A promising line of research is the development of an empirical science of algorithms for combinatorial optimization [66]. Computational testing has always been an important aspect of research on the efficiency of algorithms for integer programming. However, the standards of test designs and empirical analysis have not been uniformly applied. We believe that there will be important strides in this aspect of integer programming, and more generally of algorithms of all kinds. It may be useful to stop looking at algorithmics as purely a deductive science, and start looking for advances through repeated application of "hypothesize and test" paradigms [67], i.e. through empirical science.

The integration of logic-based methodologies and mathematical programming approaches is evidenced in the recent emergence of constraint logic programming (CLP) systems [112,15] and logico-mathematical programming [70,21]. In CLP, we see a structure of Prolog-like programming language in which some of the predicates are constraint predicates whose truth values are determined by

the solvability of constraints in a wide range of algebraic and combinatorial settings. The solution scheme is simply a clever orchestration of constraint solvers in these various domains and the role of conductor is played by SLD-Resolution. The clean semantics of logic programming is preserved in CLP. A bonus is that the output language is symbolic and expressive. An orthogonal approach to CLP is to use constraint programming methods to solve inference problems in logic. Imbeddings of logics in mixed integer programming sets were proposed by Williams [122] and Jeroslow [70]. Efficient algorithms have been developed for inference problems in many types and fragments of logic, ranging from Boolean to Predicate to Belief logics [22].

A persistent theme in the integer programming approach to combinatorial optimization, as we have seen, is that the representation (formulation), of the problem, deeply affects the efficacy of the solution methodology. A proper choice of formulation can therefore make the difference between a successful solution of an optimization problem and the more common perception that the problem is insoluble and one must be satisfied with the best that heuristics can provide. Formulation of integer programs has been treated more as an art form than a science by the mathematical programming community (exceptions are Jeroslow [70] and Williams[121]). We believe that progress in representation theory can have an important influence on future of integer programming as a broad-based problem solving methodology.

10 Defining Terms

- `Polyhedron`: The set of solutions to a finite system of linear inequalities on real-valued variables. Equivalently, the intersection of a finite number of linear half-spaces in $\Re^n$.
- `Extreme Point`: A corner point of a polyhedron.
- `Linear Program`: Optimization of a linear function subject to linear equality and inequality constraints.
- `Mixed Integer Linear Program`: A linear program with the added constraint that some of the decision variables are integer valued.
- `Packing and Covering`: Given a finite collection of subsets of a finite ground set, to find an optimal subcollection that are pairwise disjoint (packing) or whose union covers the ground set (covering).
- `Integer Polyhedron`: A polyhedron, all of whose extreme points are integer valued.

- `Cutting Plane`: A valid inequality for an integer polyhedron that separates the polyhedron from a given point outside it.
- `Relaxation`: An enlargement of the feasible region of an optimization problem. Typically, the relaxation is considerably easier to solve than the original optimization problem.
- `Fathoming`: Pruning a search tree.
- ρ-`Approximation`: An approximation method that delivers a feasible solution with objective value within a factor ρ of the optimal value of a combinatorial optimization problem.
- `Lattice`: A point lattice generated by taking integer linear combinations of a set of basis vectors.
- `Reduced Basis`: A basis for a lattice that is nearly orthogonal.
- `Knapsack Problem`: An integer linear program with a single linear constraint other than the trivial bounds and integrality constraints on the variables.

References

[1] R.K.Ahuja, T.L.Magnati, and J.B.Orlin, *Network flows: theory, algorithms and applications*, Prentice Hall, 1993.

[2] M.Akgul, Topics in Relaxation and Ellipsoidal Methods, Research notes in Mathematics, *Pitman Publishing Ltd.*, (1984).

[3] F.Alizadeh, Interior point methods in semidefinite programming with applications to combinatorial optimization, *SIAM J. on Optimization* 5 (1995) 13-51.

[4] D.Applegate, R.E.Bixby, V.Chvátal, and W.Cook, Finding cuts in large TSP's, *Technical Report*, AT&T Bell Laboratories, August 1994.

[5] S.Arora, C.Lund, R.Motwani, M.Sudan, and M.Szegedy, Proof verification and hardness of approximation problems, in *Proceedings of the* 33^{rd} *IEEE Symposium on Foundations of Computer Science*, 1992, pp. 14-23.

[6] A. Bachem and R. Kannan, Lattices and the Basis Reduction Algorithm , Technical Report, Computer Science, *Carnegie Mellon University* (1984).

[7] F.Barahona, M.Jünger, and G.Reinelt, Experiments in quadratic $0-1$ programming, in *Mathematical Programming* 44, 1989, 127-137.

[8] I.Barany and Z.Furedi, Computing the volume is difficult, *Proceedings of 18th Symposium on Theory of Computing*, ACM Press (1986) 442-447.

[9] A.Barvinok, Computing the volume, counting integral points in polyhedra when the dimension is fixed, in *Proceedings of the* 34^{th} *IEEE Conference on the Foundations of Computer Science (FOCS)* IEEE Press, (1993) 566-572.

[10] J.F.Benders, Partitioning procedures for solving mixed-variables programming problems, *Numerische Mathematik* 4 (1962), pp. 238-252.

[11] C.Berge,"Farbung von Graphen deren samtliche bzw. deren ungerade Kreise starr sind (Zusammenfassung)", Wissenschaftliche Zeitschrift, Martin Luther Universitat Halle-Wittenberg, Mathematisch-Naturwiseenschaftliche Reihe(1961) 114 - 115.

[12] C.Berge,"Sur certains hypergraphes generalisant les graphes bipartites," Combinatorial Theory and its Applications I (P.Erdos, A.Renyi and V.Sos eds.), Colloq. Math. Soc. Janos Bolyai,4,North Holland, Amsterdam(1970) 119-133.

[13] C.Berge,"Balanced matrices,"Mathematical Programming, **2** (1972), 19-31.

[14] C.Berge and M.Las Vergnas,"Sur un theoreme du type Konig pour hypergraphes," International Conference on Combinatorial Mathematics, Annals of the New York Academy of Sciences,175(1970),32-40.

[15] A.Borning (Editor), *Principles and Practice of Constraint Programming*, LNCS Volume 874, Springer-Verlag, 1994.

[16] I.Borosh, and L.B. Treybig, Bounds on positive solutions of linear diophantine equations, *Proc. Amer. Math. Soc.*, **55**,(1976) p.299.

[17] P.Camion,"Characterization of totally unimodular matrices," Proceedings of the American Mathematical Society, 16(1965),1068-1073.

[18] T.L.Cannon and K.L.Hoffman, Large-scale zero-one linear programming on distributed workstations, *Annals of Operations Research* **22** (1990), 181-217.

[19] J.W.S.Cassels, *An introduction to the geometry of numbers*, Springer Verlag (1971).

[20] V.Chandru, Complexity of the supergroup approach to integer programming, *Ph.D. Thesis*, Operations Research Center, M.I.T., (1982).

[21] V.Chandru and J.N.Hooker, "Extended Horn sets in propositional logic, " *JACM*, 38(1991), 205-221.

[22] V.Chandru and J.N.Hooker, *Optimization Methods for Logical Inference*, to be published by Wiley Interscience, 1998.

[23] S.Chopra, E.R.Gorres and M.R.Rao, Solving Steiner tree problems by Branch and Cut, *ORSA Journal of Computing*, **3**, (1992) 149-156.

[24] V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems, *Discrete Mathematics* 4 (1973) 305-337.

[25] V.Chvatal,"On certain polytopes associated with graphs," Journal of Combinatorial Theory B,18(1975),138-154.

[26] M.Conforti and G.Cornuejols,"A class of logical inference problems solvable by linear programming," FOCS,33(1992),670-675.

[27] M.Conforti, G.Cornuejols, and C.De Francesco, "Perfect 0, ± 1 matrices, "preprint, Carnegie Mellon University, 1993.

[28] M.Conforti, G.Cornuejols, A.Kapoor, and K.Vuskovic, " Balanced $0,\pm 1$ matrices," Parts I-II,preprints, Carnegie Mellon University, 1994.

[29] M.Conforti, G.Cornuejols, A.Kapoor, K.Vuskovic, and M.R.Rao, Balanced Matrices, in *Mathematical Programming, State of the Art 1994* (J.R.Birge and K.G.Murty eds.), University of Michigan, 1994.

[30] M.Conforti, G.Cornuejols, and M.R.Rao, " Decomposition of balanced 0,1 matrices," Parts I-VII, preprints, Carneigie Mellon University,1991.

[31] M.Conforti and M.R.Rao, "Testing balancedness and perfection of linear matrices," Mathematical Programming,61(1993) 1-18.

[32] G.Cornuejols and B.Novick, "Ideal 0,1 matrices," Journal of Combinatorial Theory,60(1994),145-157.

[33] CPLEX *Using the CPLEX callable Library and CPLEX mixed integer library*, CPLEX Optimization, Inc., 1993.

[34] H.Crowder, E.L.Johnson and M.W.Padberg, Solving large scale 0-1 linear programming problems, *Operations Research*, **31**, (1983) 803-832.

[35] W.H. Cunningham, Testing membership in matroid polyhedra, *Journal of Combinatorial Theory* **36B**, (1984) 161-188.

[36] P.D.Domich, R. Kannan and L.E. Trotter, Hermite normal form computation using modulo determinant arithmetic, *Mathematics of Operations Research*, **12**, (1987), 50-59.

[37] M.Dyer and A.Frieze, On the complexity of computing the volume of a polyhedron, *SIAM J. on Computing* **17** (1988) 967-974.

[38] M.Dyer, A.Frieze, and R.Kannan, A random polynomial time algorithm for approximating the volume of convex bodies, *Proceedings of 21st Symposium on Theory of Computing*, ACM Press (1989)375-381.

[39] J.Edmonds, Maximum matching and a polyhedron with 0-1 vertices, *Journal of Research of the National Bureau of Standards*, 69B, (1965) 125-130.

[40] J. Edmonds, Submodular functions, matroids and certain polyhedra, in *Combinatorial Structures and their Applications*, edited by R. Guy et al., Gordon Breach, (1970) 69-87.

[41] J.Edmonds, Matroids and the greedy algorithm, *Mathematical Programming*, (1971) 127-136.

[42] J. Edmonds, Matroid intersection, *Annals of Discrete Mathematics* 4, (1979) 39-49.

[43] J.Edmonds and R.Giles,"A min-max relation for submodular functions on graphs," Annals of Discrete Mathematics,1(1977),185-204.

[44] J.Edmonds and E.L.Johnson, Matching well solved class of integer linear programs, in *Combinatorial structure and their applications* (R.Guy ed), Gordon and Breach, New York (1970).

[45] Gy. Farkas, A Fourier-féle mechanikai elv alkalmazásai, (in Hungarian), *Mathematikai és Természettudományi Értesitõ* **12** (1894) 457-472.

[46] S.D.Feit, A fast algorithm for the two-variable integer programming problem, *JACM* **31** (1984) 99-113.

[47] J.Fonlupt and A.Zemirline, A polynomial recognition algorithm for $K_4 \setminus e$-free perfect graphs, *Research Report*, University of Grenoble (1981).

[48] R.Fourer, D.M.Gay, and B.W.Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Scientific Press, 1993.

[49] A. Frank, A weighted matroid intersection theorem, *Journal of Algorithms* **2**, (1981) 328-336.

[50] M.A.Frumkin, Polynomial-time algorithms in the theory of linear diophentine equations, in (M. Karpinski, ed.), Fundamentals of Computation Theory, *Lecture Notes in Computer Science*, **56**,(1977) Springer-Verlag.

[51] D.R.Fulkerson, The perfect graph conjecture and the pluperfect graph theorem, in *Proceedings of the Second Chapel Hill Conference on Combinatorial Mathematics and its Applications*, edited by R.C.Bose, et al., 1970, pp. 171-175.

[52] D.R.Fulkerson, A.Hoffman, and R.Oppenheim, On balanced matrices, *Mathematical Programming Study*, 1 (1974), 120-132.

[53] R.Garfinkel and G.L. Nemhauser, *Integer Programming*, Wiley, (1972).

[54] J.v.z.Gathen and M. Sieveking , Linear integer inequalities are NP-complete, *SIAM J. of Computing*,(1976).

[55] M.X.Goemans and D.P.Williamson, .878 approximation algorithms MAX CUT and MAX 2SAT, in *Proceedings of ACM STOC*, (1994) 422-431..

[56] R.E.Gomory, Outline of an algorithm for integer solutions to linear programs, *Bulletin of the American Mathematical Society* 64 (1958) pp. 275-278.

[57] R.E.Gomory, Early integer programming, in *History of Mathematical Programming* edited by J.K. Lenstra, et al., North Holland (1991).

[58] R.E.Gomory, On the relation between integer and noninteger solutions to linear programs, *Proceedings of the National Academy of Sciences of the United States of America* **53** (1965) 260-265.

[59] R.E.Gomory and T.C.Hu, Multi-terminal network flows, *SIAM Journal of Applied Mathematics*, **9**, (1961) 551-556.

[60] M.Grötschel, L.Lovász and A.Schrijver, The ellipsoid method and its consequences in Combinatorial optimization, *Combinatorica*, 1, (1982) 169-197.

[61] M.Grötschel, L.Lovász, and A.Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

[62] M.Held, P.Wolfe and H.P.Crowder, Validation of Subgradient Optimization. *Math. Programming*, 6, (1974) 62-88.

[63] M.Held, and R.M. Karp, The Travelling-Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18, (1970) 1138-1162; Part II. *Math. Programming*, 1, (1971) 6-25.

[64] A.J.Hoffman and J.K.Kruskal, Integral boundary points of convex polyhedra, *Linear Inequalities and Related Systems*, H.W.Kuhn and A.W.Tucker(editors), Princeton University Press, 1956, 223-246.

[65] J.N.Hooker, Resolution and the integrality of satisfiability polytopes, preprint, GSIA, Carnegie Mellon University, 1992.

[66] J.N.Hooker, Towards and empirical science of algorithms, *Operations Research*, (1993).

[67] J.N.Hooker and V.Vinay, Branching rules for satisfiability, in *Automated Reasoning*, 1995, pp.

[68] IBM, *Optimization Subroutine Library - Guide and Reference (Release 2)*, Third Edition, 1991.

[69] R.G.Jeroslow, There cannot be any algorithm for integer programming with quadratic constraints, *Operations Research*, **21** (1973) 221-224.

[70] R.G.Jeroslow, Logic-Based Decision Support: Mixed Integer Model Formulation, Annals of Discrete Mathematics, Volume 40, North Holland, 1989.

[71] R.G.Jeroslow and J. K. Lowe, Modeling with integer variables, *Mathematical Programming Studies* **22** (1984) 167-184.

[72] M.Jünger, G.Reinelt, and S.Thienel, Practical problem solving with cutting plane algorithms, in *Combinatorial Optimization: Papers from the DIMACS Special Year*, W.Cook, L.Lovász, and P.Seymour (Editors), Series in Discrete Mathematics and Theoretical Computer Science, Volume 20, AMS, (1995) 111-152.

[73] R.Kannan, A Polynomial algorithm for the Two-variable Integer programming Problem , *JACM* 27 (1980).

[74] R.Kannan, Minkowski's convex body theorem and integer programming, *Mathematics of Operations Research* **12** (1987) 415-440.

[75] R.Kannan and A. Bachem, Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix, *SIAM J. of Computing*, 8, (1979),

[76] Kannan, R. and C.L. Monma (1978), On the computational complexity of integer programming problems, in *Lecture Notes in Economics and Mathematical Systems 157* (R. Henn, B. Korte and W. Oettle, eds.), Springer-Verlag.

[77] N.K.Karmarkar, A New Polynomial-Time Algorithm for Linear Programming, *Combinatorica, 4*, (1984) 373-395.

[78] N.K.Karmarkar, An interior-point approach to NP-complete problems - Part I, in *Contemporary Mathematics*, Volume 114, 1990, pp. 297-308.

[79] Karp, R. (1972). Reducibilities among combinatorial problems, in *Complexity of Computer Computations* (R.E. Miller and J.W. Thatcher, eds.), Plenum Press, pp. 85-103.

[80] R.M.Karp and C.H.Papadimitriou, On Linear Characterizations of Combinatorial Optimization Problems, *SIAM Journal on Computing*, 11 (1982), 620-632.

[81] L.G. Khachiyan, A polynomial algorithm in linear programming, *Doklady Akademiia Nauk SSSR* 244:5 (1979), pp. 1093-1096, translated into English in *Soviet Mathematics Doklady*, 20:1 (1979), pp. 191-194.

[82] J.C.Lagarias, Worst-case complexity bounds for algorithms in the theory of integral quadratic forms, *Journal of Algorithms* 1 (1980) 142-186.

[83] J.C.Lagarias, Knapsack public key cryptosystems and diophantine approximation, *Advances in Cryptology*, Proceedings of CRYPTO 83, Plenum Press (1983) 3-23.

[84] E.L. Lawler, Matroid intersection algorithms, *Mathematical Programming* 9, (1975) 31-56.

[85] A.Lehman, On the width-length inequality, mimeographic notes(1965), *Mathematical Programming*, 17 (1979), 403-417.

[86] A.K.Lenstra, H.W. Lenstra Jr and L.Lovász, Factoring Polynomials with Rational Coefficients , Report 82-05, *University of Amsterdam* (1982).

[87] H.W. Lenstra Jr. Integer Programming with a Fixed Number of Variables , *Mathematics of Operations Research* 8 (1983) 538-548.

[88] H.W. Lenstra Jr, Integer Programming and Cryptography , *The Mathematical Intelligencer*, Vol.6, 1984.

[89] L.Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Mathematics*, 2 (1972), 253-267.

[90] L.Lovász, On the Shannon capacity of a graph, *IEEE Transactions on Information Theory*, 25 (1979), pp. 1-7.

[91] L.Lovász and A.Schrijver, Cones of matrices and set functions, *SIAM Journal on Optimization* 1, (1991), pp. 166-190.

[92] R.K.Martin, Using separation algorithms to generate mixed integer model reformulations, *Operations Research Letters*, 10, (1991) 119-128.

[93] C.J.H. McDiarmid, Rado's theorem for polymatroids, *Proceedings of the Cambridge Philosophical Society* 78, (1975) 263-281.

[94] G.L.Nemhauser and L.E.Trotter Jr., Properties of vertex packing and independence system polyhedra, *Mathematical Programming* 6 (1974) 48-61.

[95] G.L.Nemhauser and L.A.Wolsey, *Integer and Combinatorial Optimization*, John Wiley, 1988.

[96] M.W.Padberg, Equivalent knapsack-type formulations of bounded integer linear programs: an alternative approach, *Naval Research Logistics Quarterly*, 19, (1972), 699-708.

[97] M.W.Padberg, Perfect zero-one matrices, *Mathematical Programming* 6, (1974) 180-196.

[98] M.W.Padberg, Covering, packing and knapsack problems, *Annals of Discrete Mathematics*, 4, (1979), 265-287.

[99] M.W.Padberg, Lehman's forbidden minor characterization of ideal 0,1 matrices, *Discrete Mathematics*, 111 (1993), 409-420.

[100] M.W.Padberg and M.R.Rao, The Russian method for linear inequalities, Part III, Bounded integer programming, Preprint, New York University, (1981).

[101] M.W.Padberg and M.R.Rao, Odd minimum cut-sets and b-matching, *Mathematics of Operations Research*, 7, (1982) 67-80.

[102] M.W.Padberg and G.Rinaldi, A branch and cut algorithm for the resolution of large scale symmetric travelling salesman problems, *SIAM Review*, 33, (1991) 60-100.

[103] C.H.Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall (1982).

[104] C.H.Papadimitriou and M.Yannakakis, Optimization, approximation, and complexity classes, in *Journal of Computer and Systems Sciences* 43, 1991, pp. 425-440.

[105] G.Parker, and R.L.Rardin, *Discrete Optimization*, John Wiley, 1988.

[106] J.C. Picard and H.D. Ratliff, Minimum cuts and related problems, *Networks* 5, (1975) 357-370.

[107] W.R.Pulleyblank, Polyhedral Combinatorics, in *Handbooks in Operations Research and Management Science (Volume 1: Optimization)*, edited by G.L. Nemhauser, A.H.G. Rinooy Kan, and M.J. Todd, North Holland (1989), 371-446.

[108] P.Raghavan and C.D.Thompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs, *Combinatorica*, 7, pp. 365-374.

[109] J.M.W. Rhys, A selection problem of shared fixed costs and network flows, *Management Science*, **17**, (1970) 200-207.

[110] S. Sahni, Computationally related problems, *SIAM J. of Computing*, **3**,(1974).

[111] M.W.P.Savelsbergh, G.S.Sigosmondi, and G.L.Nemhauser, MINTO, a Mixed INTeger Optimizer, *Operations Research Letters* **15**, (1994) 47-58.

[112] V.Saraswat, and P. Van Hentenryck (Editors), *Principles and Practice of Constraint Programming*, MIT Press, 1995.

[113] A.Schrijver, *Theory of Linear and Integer Programming*, John Wiley, 1986.

[114] A. Shamir, A Polynomial-time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem, *Proceedings of the Symposium on the Foundations of Computer Science*, IEEE Press (1982).

[115] J.F.Shapiro, A Survey of Lagrangean Techniques for Discrete Optimization. *Annals of Discrete Mathematics* **5**, (1979), 113-138.

[116] P.Seymour, Decompositions of regular matroids, *Journal of Combinatorial Theory* B, 28 (1980), 305-359.

[117] D.B.Shmoys, Computing near-optimal solutions to combinatorial optimization problems, in *Combinatorial Optimization: Papers from the DIMACS Special Year*, W.Cook, L.Lovász, and P.Seymour (Editors), Series in Discrete Mathematics and Theoretical Computer Science, Volume 20, AMS, 1995, 355-398.

[118] N.Z.Shor, Convergence Rate of the Gradient Descent Method with Dilation of the Space, *Cybernetics, 6*, (1970).

[119] K.Truemper, Alpha-balanced graphs and matrices and GF(3)-representability of matroids, *Journal of Combinatorial Theory* B, 55 (1992), 302-335.

[120] H.Weyl, Elemetere Theorie der konvexen polyerer, Comm. Math. Helv., Vol. I, (1935) 3-18, (English translation in *Annals of Mathematics Studies*, **24**, Princeton, 1950).

[121] H.P.Williams, Experiments in the Formulation of Integer Programming Problems, *Mathematical Programming Study*, vol 2, (1974).

[122] H.P.Williams, Linear and integer programming applied to the propositional calculus, *International Journal of Systems Research and Information Science* 2 (1987) 81-100.

[123] M.Yannakakis, Expressing Combinatorial optimization problems by linear programs, in *Proceedings of ACM Symposium of Theory of Computing*, (1988) 223-228.

[124] M.Ziegler, *Convex Polytopes*, Springer Verlag, 1995.

11 Other Sources

JOURNALS:

Research publications in integer programming are dispersed over a large range of journals. The following is a partial list which emphasize the algorithmic aspects.

Mathematical Programming

Mathematics of Operations Research

Operations Research

Discrete Mathematics

Discrete Applied Mathematics

Journal of Combinatorial Theory (Series B)

INFORMS Journal on Computing

Operations Research Letters

SIAM Journal on Computing

SIAM Journal on Discrete Mathematics

Journal of Algorithms

Algorithmica

Combinatorica

NEWSLETTERS:

Integer programming professionals frequently use the following newsletters to communicate with each other:

- INFORMS Today (earlier OR/MS Today) published by The Institute for Operations Research and Management Science (INFORMS).
- INFORMS CSTS Newsletter published by the INFORMS computer science technical section.
- Optima published by the Mathematical Programming Society.

CONFERENCE PROCEEDINGS:

The International Symposium on Mathematical Programming (ISMP) is held once every three years and is sponsored by the Mathematical Programming Society. The most recent ISMP was held

in August 1997 in Lausanne, Switzerland. A conference on Integer Programming and Combinatorial Optimization (IPCO) is held on years when the symposium is not. Some important results in integer programming are also announced in the general conferences on algorithms and complexity (for example: SODA (SIAM), STOC (ACM) and FOCS (IEEE)). The annual meeting of the Computer Science Technical Section (CSTS) of the INFORMS held each January (partial proceedings published by Kluwer Press) is an important source for recent results in the computational aspects of integer programming.