Metadata of the chapter that will be visualized online

Chapter Title	Integer and Combinatorial Optimization	
Copyright Year	2012	
Copyright Holder	Springer Science+Business Media, LLC	
Corresponding Author	Family Name	**Hoffman**
	Particle	
	Given Name	**Karla L.**
	Suffix	
	Division/Department	Systems Engineering and Operations Research Department, School of Information Technology and Engineering
	Organization/University	George Mason University
	City	Fairfax
	State	VA
	Postcode	22030
	Country	USA
	Email	khoffman@gmu.edu
	URL	http://iris.gmu.edu/~khoffman
Author	Family Name	**Ralphs**
	Particle	
	Given Name	**Ted K.**
	Suffix	
	Division/Department	Department of Industrial and Systems Engineering
	Organization/University	Lehigh University
	City	Bethlehem
	State	PA
	Postcode	18015
	Country	USA
	Email	ted@lehigh.edu
	URL	http://coral.ie.lehigh.edu/~ted

I

Integer and Combinatorial Optimization

Karla L. Hoffman[1] and Ted K. Ralphs[2]
[1]Systems Engineering and Operations Research Department, School of Information Technology and Engineering, George Mason University, Fairfax, VA, USA
[2]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

Introduction

Integer optimization problems are concerned with the efficient allocation of limited resources to meet a desired objective when some of the resources in question can only be divided into discrete parts. In such cases, the divisibility constraints on these resources, which may be people, machines, or other discrete inputs, may restrict the possible alternatives to a finite set. Nevertheless, there are usually too many alternatives to make complete enumeration a viable option for instances of realistic size. For example, an airline may need to determine crew schedules that minimize the total operating cost, an automotive manufacturer may want to determine the optimal mix of models to produce in order to maximize profit, or a flexible manufacturing facility may want to schedule production for a plant without knowing precisely what parts will be needed in future periods. In today's changing and competitive industrial environment, the difference between ad hoc planning methods and those that use sophisticated mathematical models to determine an optimal course of action can determine whether or not a company survives.

A common approach to modeling optimization problems with discrete decisions is to formulate them as mixed integer optimization problems. This entry focuses on problems in which the functions required to represent the objective and constraints are additive, i.e., linear functions. Such a problem is called a mixed integer linear optimization problem (MILP) and its general form is

$$\max \sum_{j \in B} c_j x_j + \sum_{j \in I} c_j x_j + \sum_{j \in C} c_j x_j \tag{1}$$

$$\text{subject to } \sum_{j \in B} a_{ij} x_j + \sum_{j \in I} a_{ij} x_j + \sum_{j \in C} a_{ij} x_j \begin{Bmatrix} \le \\ = \\ \ge \end{Bmatrix} b_i \quad \forall i \in M, \tag{2}$$

$$l_j \le x_j \le u_j \quad \forall j \in N = B \cup I \cup C, \tag{3}$$

$$x_j \in \{0, 1\} \quad \forall j \in B, \tag{4}$$

$$x_j \in \mathbb{Z} \quad \forall j \in I, \text{ and} \tag{5}$$

$$x_j \in \mathbb{R} \quad \forall j \in C. \tag{6}$$

A solution to (1)–(6) is a set of values assigned to the variables x_j, $j \in N$. The objective is to find a solution that maximizes the weighted sum (1), where the coefficients c_j, $j \in N$ are given. B is the set of indices of binary variables (those that can take on only values 0 or 1), I is the set of indices of integer variables (those that can take on any integer value), and C is the set of indices of continuous variables.

S. Gass, M. Fu (eds.), *Encyclopedia of Operations Research and Management Science*,
DOI 10.1007/978-1-4419-1153-7,

Comp. by: DMuthuKumar Stage: Galleys Chapter No.: 129 Title Name: EORMS
Date:9/6/12 Time:12:12:24 Page Number: 2

As indicated above, each of the first set of constraints (2) can be either an inequality constraint ("$\leq$" or "$\geq$") or an equality constraint ("$=$"). The data l_j and u_j are the lower- and upper-bound values, respectively, for variable x_j, $j \in N$.

This general class of problems has many important special cases. When $B = I = \emptyset$, we have what is known as a linear optimization problem (LP). If $C = I = \emptyset$, then the problem is referred to as a (pure) binary integer linear optimization problem (BILP). Finally, if $C = \emptyset$, the problem is called a (pure) integer linear optimization problem (ILP). Otherwise, the problem is simply a MILP. Throughout this discussion, we refer to the set of points satisfying (1)–(6) as $\mathcal{S}$, and the set of points satisfying all but the integrality restrictions (4)–(5) as $\mathcal{P}$. The problem of optimizing over $\mathcal{P}$ with the same objective function as the original MILP is called the LP relaxation and arises frequently in algorithms for solving MILPs.

A class of problems closely related to BILPs are the combinatorial optimization problems (COPs). A COP is defined by a ground set $\mathcal{E}$, a set $\mathcal{F}$ of subsets of ε that are called the feasible subsets, and a cost c_e associated with each element $e \in \mathcal{E}$. Each feasible subset $F \in \mathcal{F}$ has an associated (additive) cost taken to be $\Sigma_{e \in F}\, c_e$. The goal of a COP is find the subset $F \in \mathcal{F}$ of minimum cost. The set $\mathcal{F}$ can often be described as the set of solutions to a BILP by associating a binary variable x_e with each member e of the ground set, indicating whether or not to include it in the selected subset. For this reason, combinatorial optimization and integer optimization are closely related and COPs are sometimes informally treated as being a subclass of MILPs, though there are COPs that cannot be formulated as MILPs.

Solution of an MILP involves finding one or more best (optimal) solutions from the set $\mathcal{S}$. Such problems occur in almost all fields of management (e.g., finance, marketing, production, scheduling, inventory control, facility location and layout, supply chain management), as well as in many engineering disciplines (e.g., optimal design of transportation networks, integrated circuit design, design and analysis of data networks, production and distribution of electrical power, collection and management of solid waste, determination of minimum energy states for alloy construction, planning for energy resource problems, scheduling of lines in flexible manufacturing facilities, and design of experiments in crystallography).

This article gives a brief overview of the related fields of integer and combinatorial optimization. These fields have by now accumulated a rich history and a rich mathematical theory. Texts covering the theory of linear and integer linear optimization include those of Bertsimas and Weismantel (2005), Chvátal (1983), Nemhauser and Wolsey (1988), Parker and Rardin (1988), Schrijver (1986), and Wolsey (1998). Overviews of combinatorial optimization are provided by Papadimitriou and Steiglitz (1982) and Schrijver (2003). Jünger et al. (2010) have produced a marvelous and comprehensive volume containing an overview of both the history and current state of the art in integer and combinatorial optimization.

Applications

This section describes some classical integer and combinatorial optimization models to provide an overview of the diversity and versatility of this field.

Knapsack Problems

Suppose one wants to fill a knapsack that has a weight capacity limit of W with some combination of items from a list of n candidates, each with weight w_i and value v_i, in such a way that the value of the items packed into the knapsack is maximized. This problem has a single linear constraint (that the weight of the items selected not exceed W), a linear objective function (to maximize the sum of the values of the items in the knapsack), and the added restriction that each item either be in the knapsack or not—it is not possible to select a fractional portion of an item. For solution approaches specific to the knapsack problem, see Martello and Toth (1990).

Although this problem might seem too simplistic to have many practical applications, the knapsack problem arises in a surprisingly wide variety of fields. For example, one implementation of the public-key cryptography systems that are pervasive in security applications depends on the solution of knapsack problems to determine the cryptographic keys (Odlyzko 1990). The system depends on the fact that, despite their simplicity, some knapsack problems are extremely difficult to solve.

More importantly, however, the knapsack problem arises as a substructure in many other important combinatorial problems. For example, machine-scheduling problems involve restrictions on the capacities of the machines to be scheduled (in addition to other constraints). Such a problem involves assigning a set of jobs to a machine in such a way that the capacity constraint is not violated. It is easy to see that such a constraint is of the same form as that of a knapsack problem. Often, a component of the solution method for problems with knapsack constraints involves solving the knapsack problem itself, in isolation from the original problem (see Savelsbergh (1997)). Another important example in which knapsack problems arise is the capital budgeting problem. This problem involves finding a subset of the set of (possibly) thousands of capital projects under consideration that will yield the greatest return on investment, while satisfying specified financial, regulatory, and project relationship requirements (Markowitz and Manne 1957; Weingartner 1963). Here also, the budget constraint takes the same form as that of the knapsack problem.

Network and Graph Problems

Many optimization problems can be represented by a network, formally defined as a set of nodes and a set of arcs (unidirectional connections specified as ordered pairs of nodes) or edges (bidirectional connections specified as unordered pairs of nodes) connecting those nodes, along with auxiliary data such as costs and capacities on the arcs (the nodes and arcs together *without* the auxiliary data form a graph). Solving such network problems involves determining an optimal strategy for routing certain commodities through the network. This class of problems is thus known as network flow problems. Many practical problems arising from physical networks, such as city streets, highways, rail systems, communication networks, and integrated circuits, can be modeled as network flow problems. In addition, there are many problems that can be modeled as network flow problems even when there is no underlying physical network. For example, in the assignment problem, one wishes to assign people to jobs in a way that minimizes the cost of the assignment. This can be modeled as a network flow problem by creating a network in which one set of nodes represents the people to be assigned, and another set of nodes represents the possible jobs, with an arc connecting a person to a job if that person is capable of performing that job. A general survey of applications and solution procedures for network flow problems is given by Ahuja et al. (1993).

Space-time networks are often used in scheduling applications. Here, one wishes to meet specific demands at different points in time. To model this problem, different nodes represent the same entity at different points in time. An example of the many scheduling problems that can be represented as a space-time network is the airline fleet assignment problem, which requires that one assign specific planes to prescheduled flights at minimum cost (Abara 1989; Hane et al. 1995). Each flight must have one and only one plane assigned to it, and a plane can be assigned to a flight only if it is large enough to service that flight and only if it is on the ground at the appropriate airport, serviced and ready to depart when the flight is scheduled for takeoff. The nodes represent specific airports at various points in time and the arcs represent the flow of aircraft of a variety of types into and out of each airport. There are layover arcs that permit a plane to stay on the ground from one time period to the next, service arcs that force a plane to be out of duty for a specified amount of time, and connecting arcs that allow a plane to fly from one airport to another without passengers.

A variety of important combinatorial problems are graph based, but do not involve flows. Such graph-based combinatorial problems include the node-coloring problem, the objective of which is to determine the minimum number of colors needed to color each node of a graph in order that no pair of adjacent nodes (nodes connected by an edge) share the same color; the matching problem, the objective of which is to find a maximum weight collection of edges such that each node is incident to at most one edge; the maximum clique problem, the objective of which is to find the largest subgraph of the original graph such that every node is connected to every other node in the subgraph; and the minimum cut problem, the objective of which is to find a minimum weight collection of edges that (if removed) would disconnect a set of nodes s from a set of nodes t.

Although these graph-based combinatorial optimization problems might appear, at first glance, to be interesting only from a mathematical perspective and to have little application to the decision-making that occurs in management or engineering, their domain of application is extraordinarily broad. The four-color problem, e.g., which is the question of whether a map can be colored with four colors or less, is a special case of the node-coloring problem. The maximum clique problem has important implications in the growing field of social network analysis. The minimum cut problem is used in analyzing the properties of real-world networks, such as those arising in communications and logistics applications.

Location, Routing, and Scheduling Problems

Many network-based combinatorial problems involve finding a route through a given graph satisfying specific requirements. In the Chinese postman problem, one wishes to find a shortest walk (a connected sequence of arcs) through a network such that the walk starts and ends at the same node and traverses every arc at least once (Edmonds and Johnson 1973). This models the problem faced by a postal delivery worker attempting to minimize the number traversals of each road segment on a given postal route. If one instead requires that each node be visited exactly once, the problem becomes the notoriously difficult traveling salesman problem (Applegate et al. 2006). The traveling salesman problem has numerous applications within the routing and scheduling realm, as well as in other areas, such as genome sequencing (Avner 2001), the routing of SONET rings (Shah 1998), and the manufacturing of large-scale circuits (Barahona et al. 1988; Ravikumar 1996). The well-known vehicle routing problem is a generalization in which multiple vehicles must each follow optimal routes subject to capacity constraints in order to jointly service a set of customers (Golden et al. 2010).

A typical scheduling problem involves determining the optimal sequence in which to execute a set of jobs subject to certain constraints, such as a limited set of machines on which the jobs must be executed or a set of precedence constraints restricting the job order (see Applegate and Cook (1991)). The literature on scheduling problems is extremely rich and many variants of the basic problem have been suggested (Pinedo 2008). Location problems involve choosing the optimal set of locations from a set of candidates, perhaps represented as the nodes of a graph, subject to certain requirements, such as the satisfaction of given customer demands or the provision of emergency services to dispersed populations (Drezner and Hamacher 2004). Location, routing, and scheduling problems all arise in the design of logistics systems, i.e., systems linking production facilities to end-user demand points through the use of warehouses, transportation facilities, and retail outlets. Thus, it is easy to envision combinations of these classes of problems into even more complex combinatorial problems and much work has been in this direction.

Packing, Partitioning, and Covering Problems

Many practical optimization problems involve choosing a set of activities that must either cover certain requirements or must be packed together so as to not exceed certain limits on the number of activities selected. The airline crew scheduling problem, e.g., is a covering problem in which one must choose a set of pairings (a set of flight legs that can be flown consecutively by a single crew) that cover all required routes (Hoffman and Padberg 1993; Vance et al. 1997). Alternatively, an example of a set packing problem is a combinatorial auction (Cramton et al. 2006). The problem is to select subsets of a given set of items that are up for auction in such a way that each item is included in at most one subset. This is the problem faced by an auctioneer in an auction in which bidders can bid on sets of items rather than just single items. If one requires that all items be sold, then the auctioneer's problem becomes a partitioning problem. There are a variety of languages that allow users to express the interrelationship among their bids. Such languages (e.g., "OR," "XOR," "ORofXOR," "XORofOR") create a somewhat different structure to the combinatorial problem.

In the above examples, the coefficients in constraints (2) are either zero or one and all variables are binary. The variables represent the choice of activities, while each constraint represents either a covering ("$\geq$"), packing ("$\leq$"), or partitioning ("$=$") requirement. In many cases, these problems can be easily interpreted by thinking of the rows as a set of items to be allocated or a set of activities to be undertaken and the columns as subset of those items/activities. The optimization problem is then to find the best collection of subsets of the activities/items (columns) in order to cover/partition/pack the row set. Surveys on set partitioning, covering, and packing are given in Balas and Padberg (1976), Borndörfer and Weismantel (2000), Hoffman and Padberg (1993), and Padberg (1979b).

Other Nonconvex Problems

The versatility of the integer optimization model (1)–(6) might best be exemplified by the fact that many nonlinear/nonconvex optimization problems can be reformulated as MILPs. For example, one reformulation technique for representing nonlinear functions is to find a piecewise linear approximation and to represent the function by adding a binary variable corresponding to each piece of the approximation. The simplest example of such a transformation is the fixed-charge problem in which the cost function has both a fixed charge for initiating a given activity, as well as marginal costs associated with continued operation. One example of a fixed-charge problem is the facility location problem in which one wishes to locate facilities in such a way that the combined cost of building the facility (a onetime fixed cost) and producing and shipping to customers (marginal costs based on the amount shipped and produced) is minimized (see Drezner and Hamacher (2004)). The fact that nothing can be produced in the facility unless the facility exists creates a discontinuity in the cost function. This function can be transformed to a linear function by the introduction of additional variables that take on only the values 0 or 1. Similar transformations allow one to model separable nonlinear functions as integer (linear) optimization problems.

Solution Methods

Solving integer optimization problems (finding an optimal solution), can be a difficult task. The difficulty arises from the fact that unlike (continuous) linear optimization problems, for which the feasible region is convex, the feasible regions of integer optimization problems consists of either a discrete set of points or, in the case of general MILP, a set of disjoint polyhedra. In solving a linear optimization problem, one can exploit the fact that, due to the convexity of the feasible region, any locally optimal solution is a global optimum. In finding global optima for integer optimization problems, on the other hand, one is required to prove that a particular solution dominates all others by arguments other than the calculus-based approaches of convex optimization. The situation is further complicated by the fact that the description of the feasible region is implicit. In other words, the formulation (1)–(6) does not provide a computationally useful geometric description of the set $\mathcal{S}$. A more useful description can be obtained in one of two ways described next.

The first approach is to apply the powerful machinery of polyhedral theory. Weyl (1935) established the fact that a polyhedron can either be defined as the intersection of finitely many half-spaces, i.e., as a set of points satisfying inequalities of the form (2) and (3), or as the convex hull of a finite set of extreme points plus the conical hull of a finite set of extreme rays. If the data describing the original problem formulation are rational numbers, then Weyl's theorem implies the existence of a finite system of linear inequalities describing the convex hull of $\mathcal{S}$, denoted by conv($\mathcal{S}$) (Nemhauser and Wolsey 1988). Optimization of a linear function over conv($\mathcal{S}$) is precisely equivalent to optimization over $\mathcal{S}$, but optimizing over conv($\mathcal{S}$) is a convex optimization problem. Thus, if it were possible to enumerate the set of inequalities in Weyl's description, one could solve the integer optimization problem using methods for convex optimization, in principle. The difficulty with this method, however, is that the number of linear inequalities required is too large to construct explicitly, so this does not lead directly to a practical method of solution.

A second approach is to describe the feasible set in terms of logical disjunction. For example, if $j \in B$, then either $x_j = 0$ or $x_j = 1$. This means that, in principle, the set $\mathcal{S}$ can be described by replacing constraints (4)–(5) with a set of appropriately chosen disjunctions. In fact, it is known that any MILP can be described as a set of linear inequalities of the form (2) and (3), plus a finite set of logical disjunctions (Balas 1998). Similarly, however, the number of such disjunctions would be too large to enumerate explicitly and so this does not lead directly to a practical method of solution either.

Although neither of the above methods for obtaining a more useful description of $\mathcal{S}$ leads directly to an efficient methodology because they both produce descriptions of impractical size, most solution techniques are nonetheless based on generating partial descriptions of $\mathcal{S}$ in one of the above forms (or a combination of both). The general outline of such a method is as follows:

1. Identify a (tractable) convex relaxation of the problem and solve it to either
   - Obtain a valid upper bound on the optimal solution value; or
   - Prove that the relaxation is infeasible or unbounded (and thus, the original MILP is also infeasible or unbounded)
2. If solving the relaxation produces a solution $\hat{x} \in \mathbb{R}^N$ that is feasible to the MILP, then this solution must also be optimal to the MILP.
3. Otherwise, either
   - Identify a logical disjunction satisfied by all members of $\mathcal{S}$, but not by $\hat{x}$ and add it to the description of $\mathcal{P}$ (more on how this is done below); or
   - Identify an implied linear constraint (called a valid inequality or a cutting plane) satisfied by all members of $\mathcal{S}$, but not by $\hat{x}$ and add it to the description of $\mathcal{P}$

In Step 1, the LP relaxation obtained by dropping the integrality conditions on the variables and optimizing over $\mathcal{P}$ is commonly used. Other possible relaxations include Lagrangian relaxations (Fisher 1981; Geoffrion 1974), semi-definite programming relaxations (Rendl 2010), and combinatorial relaxations, e.g., the one-tree relaxation for the traveling salesman problem Held and Karp (1970). This discussion initially considers use of the LP relaxation, since this is the simplest one and the one used in state-of-the-art software. Additional relaxations are considered in more detail in section "Advanced Procedures."

By recursively applying the basic strategy outlined above, a wide variety of convergent methods that generate partial descriptions of $\mathcal{S}$ can be obtained. These methods can be broadly classified as either implicit enumeration methods (employing the use of logical disjunction in Step 3) or cutting plane methods (based on the generation of valid inequalities in Step 3), though these are frequently combined into hybrid solution procedures in computational practice. In the next two sections, more details about these two classes of methods are given.

Enumerative Algorithms

The simplest approach to solving a pure integer optimization problem is to enumerate all finitely many possibilities (as long as the problem is bounded). However, due to the combinatorial explosion resulting from the fact that the size of the set $\mathcal{S}$ is generally exponential in the number of variables, only the smallest instances can be solved by such an approach. A more efficient approach is to only implicitly enumerate the possibilities by eliminating large classes of solutions using domination or feasibility arguments. Besides straightforward or implicit enumeration, the most commonly used enumerative approach is called branch and bound.

The branch-and-bound method was first proposed by Land and Doig (1960) and consists of generating disjunctions satisfied by points in $\mathcal{S}$ and using them to partition the feasible region into smaller subsets. Some variant of the technique is used by practically all state-of-the-art solvers. An LP-based branch-and-bound method consists of solving the LP relaxation as in Step 1 above to either obtain a solution and an associated upper bound or to prove infeasibility or unboundedness. If the generated solution $\hat{x} \in \mathbb{R}^N$ to the relaxation is infeasible to the original MILP, then $\hat{x}_j \notin \mathbb{Z}$ for some $j \in B \cup I$. However, $x_j \in \mathbb{Z}$ for all $x \in \mathcal{S}$. Therefore, the logical disjunction

$$x_j \leq \lfloor \hat{x}_j \rfloor \,\mathrm{OR}\, x_j \geq \lceil \hat{x}_j \rceil \qquad (7)$$

is satisfied by all $x \in \mathcal{S}$, but not by $\hat{x}$. In this case, one can impose the disjunction implicitly by branching, i.e., creating two subproblems, one associated with each of the terms of the disjunction (7).

The branch-and-bound method consists of applying this same method to each of the resulting subproblems recursively. Note that the optimal solution to a subproblem may or may not be the global optimal solution. Each time a new solution is found, it is checked to determine whether it is the best seen so far and if so, it is recorded and becomes the current incumbent. The true power of this method comes from the fact that if the upper bound obtained by solving the LP relaxation is smaller than the value of the current incumbent, the node can be discarded. Mitten (1970) provided the first description of a general algorithmic framework for branch and bound. Hoffman and

Padberg (1985) provided an overview of LP-based branch-and-bound techniques. Linderoth and Savelsbergh (1999) reported on a computational study of search strategies used within branch and bound.

Cutting Plane Algorithms

Gomory (1958, 1960) was the first to derive a cutting plane algorithm following the basic outline above for integer optimization problems. His algorithm can be viewed, in some sense, as a constructive proof of Weyl's theorem. Although Gomory's algorithm converges to an optimal solution in a finite number of steps (in the case of pure integer optimization problems), the convergence to an optimum may be extraordinarily slow due to the fact that these algebraically derived valid inequalities are weak—they may not even support conv($\mathcal{S}$) and are hence dominated by stronger (but undiscovered) valid inequalities. Since the smallest possible description of conv($\mathcal{S}$) is desired, one would like to generate only the strongest valid inequalities, i.e., those that are part of some minimal description of conv($\mathcal{S}$). Such inequalities are called facets. In general, knowing all facets of conv($\mathcal{S}$) is enough to solve the MILP (though this set would still be very large in most cases).

A general cutting plane approach relaxes the integrality restrictions on the variables and solves the resulting LP relaxation over the set $\mathcal{P}$. If the LP is unbounded or infeasible, so is the MILP. If the solution to the LP is integer, i.e., satisfies constraints (4) and (5), then one has solved the MILP. If not, then one solves a separation problem whose objective is to find a valid inequality that cuts off the fractional solution to the LP relaxation while assuring that all feasible integer points satisfy the inequality—i.e., an inequality that separates the fractional point from the polyhedron conv($\mathcal{S}$). Such an inequality is called a cut for short. The algorithm continues until termination in one of two ways: either an integer solution is found (the problem has been solved successfully) or the LP relaxation is infeasible and therefore the integer problem is infeasible.

For ILPs, there are versions of Gomory's method that yield cutting plane algorithms that will produce a solution in a finite number of iterations, at least with the use of exact rational arithmetic. In practice, however, the algorithm could terminate in a third way—it may not be possible to identify a new cut even though the optimal solution has not been found either due to numerical difficulties arising from accumulated round-off error or because procedures used to generate the cuts are unable to guarantee the generation of a violated inequality, even when one exists. If one terminates the cutting plane procedure because of this third possibility, then, in general, the process has still improved the original formulation and the bound resulting from solving the LP relaxation is closer to the optimal value. By then switching to an implicit enumeration strategy, one may still be able to solve the problem. This hybrid strategy, known as branch and cut, is discussed in the next section.

Advanced Procedures

Branch and Cut

The two basic methods described above can be hybridized into a single algorithm that combines the power of the polyhedral and disjunctive approaches. This method is called branch and cut. A rather sizable literature has sprung up around these methods. Papers describing the basic framework include those of Hoffman and Padberg (1991) and Padberg and Rinaldi (1991). Surveys of the computational issues and components of a modern branch-and-cut solver include Atamtürk and Savelsbergh (2005), Linderoth and Ralphs (2005), and Martin (2001). The major components of the algorithm consist of automatic reformulation and preprocessing procedures (see next section), heuristics that provide good feasible integer solutions, procedures for generating valid inequalities, and procedures for branching. All of these are embedded into a disjunctive search framework, as in the branch-and-bound approach. These components are combined so as to guarantee optimality of the solution obtained at the end of the calculation. The algorithm may also be stopped early to produce a feasible solution along with a bound on the relative distance of the current solution from optimality. This hybrid approach has evolved to be an extremely effective way of solving general MILPs. It is the basic approach taken by all state-of-the-art solvers for MILP.

Ideally, the cutting planes generated during the course of the algorithm would be facets of conv($\mathcal{S}$). In the early years of integer optimization, considerable research activity was focused on identifying part (or all) of the list of facets for specific combinatorial optimization problems by exploiting the special structure of conv($\mathcal{S}$) (Balas and Padberg 1972; Balas 1975; Bauer et al. 2002; Hammer et al. 1975; Nemhauser and Sigismondi 1992; Nemhauser and Trotter 1974; Nemhauser and Vance 1994; Padberg 1973, 1974, 1979a; Pochet and Wolsey 1991; Wolsey 1975, 1976). This led to a wide variety of problem-dependent algorithms that are nevertheless based on the underlying principle embodied in Weyl's theorem. An extensive survey of the use of these techniques in combinatorial optimization is given by Aardal and van Hoesel (1996a, b).

Research on integer optimization is increasingly focused on methods for generating inequalities based purely on the disjunctive structure of the problem and not on properties of a particular class of problems. Part of the reason for this is the need to be able to solve more general MILPs for which even the dimension of conv($\mathcal{S}$) is not known. With this approach, it is not possible to guarantee the generation of facets in every iteration, but theoretical advances have resulted in vast improvements in the ability to solve general unstructured integer optimization problems using off-the-shelf software. A survey of cutting plane methods for general MILPs is provided by (Cornuéjols 2008). Other papers on techniques for generating valid inequalities for general MILPs include Balas et al. (1993, 1996, 1999), Gu et al. (1998, 1999, 2000), Nemhauser and Wolsey (1990), Marchand and Wolsey (2001), and Wolsey (1990).

Equally as important as cutting plane generation techniques are branching schemes, though these methods have received far less attention in the literature. Branching methods are generally based on some method of estimating the impact of a given branching disjunction and trying to choose the best one according to certain criteria. Papers discussing branching methods include Achterberg et al. (2005), Fischetti and Lodi (2002), Karamanov and Cornuéjols (2009), and Owen and Mehrotra (2001).

There has been a surge in research on the use of heuristic methods within the branch-cut-cut framework in order to generate good solutions and improve bounds as the search progresses. Many search methods are based on limited versions of the same search procedures used to find globally optimal solutions. The development of such methods has led to marked improvements in the performance of exact algorithms (Balas and Martin 1980; Balas et al. 2004; Fischetti and Lodi 2002; Nediak and Eckstein 2001). In current state-of-the-art software, multiple heuristics are used because they are likely to produce feasible solutions more quickly than tree search, which helps both to eliminate unproductive subtrees and to calculate improved variable bounds that result in a tighter description of the problem. These heuristics include techniques for searching within the local neighborhood of a given linear feasible solutions for integer solutions using various forms of local search. Achtenberg and Berthold (2007), Danna et al. (2005), Fischetti et al. (2009), and Rothberg (2007) provide descriptions of heuristics built into current packages.

Automatic Reformulation

Before solving an integer optimization problem, the first step is that of formulation, in which a conceptual model is translated into the form (1)–(6). There are often different ways of mathematically representing the same problem, both because different systems of the form (1)–(6) may define precisely the same set $\mathcal{S}$ and because it may be possible to represent the same conceptual problem using different sets of variables. There are a number of different ways in which the conceptual model can be translated into a mathematical model, but the most common is to use an algebraic modeling language, such as AIMMS, AMPL (Fourer et al. 1993), GAMS (Brooke et al. 1988), MPL, or OPL Studio.

The time required to obtain an optimal solution to a large integer optimization problem usually depends strongly on the way it is formulated, so much research has been directed toward the effective automatic reformulation techniques. Unlike linear optimization problems, the number of variables and constraints representing an integer optimization problem may not be indicative of its difficulty. In this regard, it is sometimes advantageous to use a model with a larger number of integer variables, a larger number of constraints, or even both. Discussions of alternative formulation approaches are given in Guignard and Spielberg (1981) and Williams (1985), and a description of approaches to automatic reformulation or preprocessing is given in Anderson and Anderson (1995), Atamturk and Savelsbergh (2000), Brearley et al. (1975), Hoffman and Padberg (1991), Roy and Wolsey (1987), and Savelsbergh (1994).

A variety of difficult problems have been solved by reformulating them as either set-covering or set-partitioning problems having an extraordinary number of variables. Because for even small instances, such reformulations may be too large to solve directly, a technique known as column generation, which began with the seminal work of Gilmore and Gomory (1961) on the cutting stock problem, is employed. An overview of such transformation methods can be found in Barnhart et al. (1998). For specific implementations, for the vehicle routing problem, see Chabrier (2006), for the bandwidth packing problem, see Hoffman and Villa (2007) and Parker and Ryan (1995), for the generalized assignment problem, see Savelsbergh (1997), and for alternative column-generation strategies for solving the cutting stock problem see Vance et al. (1994). Bramel and Simchi-Levi (1997) have shown that the set-partitioning formulation for the vehicle routing problem with time windows is very effective in practice—that is, the relative gap between the fractional linear optimization solutions and the global integer solution is small. Similar results have been obtained for the bin-packing problem (Chan et al. 1998a) and for the machine-scheduling problem (Chan et al. 1998b).

Decomposition Methods

Relaxing the integrality restriction is not the only approach to relaxing the problem. An alternative approach to the solution to integer optimization problems is to relax a set of complicating constraints in order to obtain a more tractable model. This technique is effective when the problem to be solved is obtained by taking a well-solved base problem and adding constraints specific to a particular application. By capitalizing on the ability to solve the base problem, one can obtain bounds that are improved over those obtained by solving the LP relaxation. These bounding methods can then be used to drive a branch-and-bound algorithm, as described earlier. Such bounding methods are called constraint decomposition methods or simply decomposition methods, since they involve decomposing the set of constraints. By removing the complicating constraints from the constraint set, the resulting subproblem is frequently considerably easier to solve. The latter is a necessity for the approach to work because the subproblems must be solved repeatedly. The bound found by decomposition can be tighter than that found by linear optimization, but only at the expense of solving subproblems that are themselves integer optimization problems. Decomposition requires that one understand the structure of the problem being solved in order to then relax the constraints that are complicating.

The bound resulting from a particular decomposition can be computed using two different computational techniques—Dantzig-Wolfe decomposition (Dantzig and Wolfe 1960; Vanderbeck 2000) (column generation) and Lagrangian relaxation (Fisher 1981; Geoffrion 1974; Held and Karp 1970). In the former case, solutions to the base problem are generated dynamically and combined in an attempt to obtain a solution satisfying the complicating constraints. In the latter case, the complicating constraints are enforced implicitly by penalizing their violation in the objective function. Overviews of the theory and methodology behind decomposition methods and how they are used in integer programming can be found in Ralphs and Galati (2005) and Vanderbeck and Wolsey (2010). A related approach is that of Lagrangian decomposition (Guignard and Kim 1987), which consists of isolating sets of constraints so as to obtain multiple, separate, easy-to-solve subproblems. The dimension of the problem is increased by creating copies of variables that link the subsets and adding constraints that require these copies to have the same value as the original in any feasible solution. When these constraints are relaxed in a Lagrangian fashion, the problem decomposes into blocks that can be treated separately.

Most decomposition-based strategies involve decomposition of constraints, but there are cases in which it may make sense to decompose the variables. These techniques work well in the case when fixing some subset of the variables (the complicating variables) to specific values reduces the problem to one that is easy to solve. Benders' decomposition algorithm projects the problem into the space of these complicating variables and treats the remaining variables implicitly by adding so-called Benders cuts violated by solutions that do not have a feasible completion and adding a term to the objective function representing the cost of completion for any given set of value of the complicating variables (Benders 1962). For a survey on Benders cuts, see Hooker (2002).

Related Topics

There are a number of topics related to combinatorial and integer optimization that have not been covered here. One such topic is the complexity of integer optimization problems (Garey and Johnson 1979), an area of theoretical study that has increased our understanding of the implicit difficulty of integer optimization dramatically. Another important topic is that of heuristic solution approaches—that is, techniques for obtaining good but not necessarily optimal solutions to integer optimization problems quickly. In general, heuristics do not provide any guarantee as to the quality of the solutions they produce, but are very important in practice for a variety of reasons. Primarily, they may provide the only usable solution to very difficult optimization problems for which the current exact algorithms fail to produce one. Research into heuristic algorithms has applied techniques from the physical sciences to the approximate solution of combinatorial problems. For surveys of research in simulated annealing (based on the physical properties of heat), genetic algorithms (based on properties of natural mutation), and neural networks (models of brain function) see Hansen (1986), Goldberg (1989), and Zhang (2010), respectively. Glover and Laguna (1998) have generalized some of the attributes of these methods into a method called tabu search. Worst-case and probabilistic analysis of heuristics are discussed in Cornuejols et al. (1980), Karp (1976), and Kan (1986).

Another developing trend is the use of approaches from other disciplines in which optimization problems also arise. In some cases, multiple approaches can be used to handle difficult optimization problems by merging alternative strategies into a single algorithm (the so-called algorithm portfolio approach). As an example, constraint-logic programming was developed by computer scientists in order to work on problems of finding feasible solutions to a set of constraints. During the last decade, many of the advances of constraint-logic programming have been embedded into mathematical programming algorithms in order to handle some of the difficult challenges of combinatorial optimization such as those related to scheduling where there is often significant symmetry. For example, see Hooker (2007) and Rasmussen and Trick (2007) for some applications that use both Benders decomposition and constraint programming to handle difficult scheduling problems. For research that relates issues in computational logic to those associated with combinatorial optimization see McAloon and Tretkoff (1996).

See

- ▶ Air Traffic Management
- ▶ Airline Industry Operations Research
- ▶ Assignment Problem
- ▶ Bender's Decomposition
- ▶ Bin Packing
- ▶ Branch and Bound
- ▶ Capital Budgeting
- ▶ Chinese Postman Problem
- ▶ Combinatorial Auctions
- ▶ Combinatorial Explosion
- ▶ Combinatorics
- ▶ Facility Location
- ▶ Fathom
- ▶ Global Optimum
- ▶ Heuristics
- ▶ Lagrangian Function
- ▶ Linear Programming
- ▶ Local Optimum
- ▶ Networks
- ▶ Packing Problem
- ▶ Relaxed Problem
- ▶ Set-Covering Problem
- ▶ Set-Partitioning Problem
- ▶ Tabu Search
- ▶ Traveling Salesman Problem

References

Aardal, K., & van Hoesel, C. (1996a). Polyhedral techniques in combinatorial optimization I: Applications and computations. *Statistica Neerlandica, 50*, 3–26.

Aardal, K., & van Hoesel, C. (1996b). Polyhedral techniques in combinatorial optimization II: Applications and computations. *Statistica Neerlandica, 50*, 3–26.

Abara, J. (1989). Applying integer linear programming to the fleet assignment problem. *Interfaces, 19*, 20–28.

Achtenberg, T., & Berthold, T. (2007). Improving the feasibility pump. *Discrete Mathematics, 4*, 77–86.

Achterberg, T., Koch, T., & Martin, A. (2005). Branching rules revisited. *Operations Research Letters, 33*, 42–54.

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms, and applications*. Englewood Cliffs, NJ: Prentice Hall.

Anderson, E., & Anderson, K. (1995). Presolving in linear programming. *Mathematical Programming, 71*, 221–245.

Applegate, D., & Cook, W. (1991). A computational study of the job-shop scheduling problem. *INFORMS Journal on Computing, 3*, 149–156.

Applegate, D., Bixby, R., Chvátal, V., & Cook, W. (2006). *The traveling salesman problem: A computational study*. Princeton, NJ: Princeton University Press.

Atamturk, A., & Savelsbergh, M. (2000). Conflict graphs in solving integer programming problems. *European Journal of Operational Research, 121*, 40–55.

Atamtürk, A., & Savelsbergh, M. (2005). Integer-programming software systems. *Annals of Operations Research, 140*, 67–124.

Avner, P. (2001). A radiation hybrid transcript may of the mouse genome. *Nature Genetics, 29*, 194–200.

Balas, E. (1975). Facets of the knapsack polytope. *Mathematical Programming, 8*, 146–164.

Balas, E. (1998). Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics, 89*, 3–44.

Balas, E., & Martin, R. (1980). Pivot and complement: A heuristic for 0-1 programming. *Management Science, 26*, 86–96.

Balas, E., & Padberg, M. (1972). On the set-covering problem. *Operations Research, 20*, 1152–1161.

Balas, E., & Padberg, M. (1976). Set partitioning: A survey. *SIAM Review, 18*, 710–760.

Balas, E., Ceria, S., & Corneujols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming, 58*, 295–324.

Balas, E., Ceria, S., & Cornuejols, G. (1996). Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science, 42*, 1229–1246.

Balas, E., Ceria, S., Cornuejols, G., & Natraj, N. (1999). Gomory cuts revisited. *Operations Research Letters, 19*, 1–9.

Balas, E., Schmieta, S., & Wallace, C. (2004). Pivot and shift—A mixed integer programming heuristic. *Discrete Optimization, 1*, 3–12.

Barahona, F., Grötschel, M., Jünger, M., & Reinelt, G. (1988). An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research, 36*, 493–513.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch and price: Column generation for solving huge integer programs. *Operations Research, 46*, 316–329.

Bauer, P., Linderoth, J., & Savelsbergh, M. (2002). A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming, 9*, 307–348.

Benders, J. F. (1962). Partitioning procedures for solving mixed variable programming problems. *Numerische Mathematik, 4*, 238–252.

Bertsimas, D., & Weismantel, R. (2005). *Optimization over integers*. Cambridge, MA: Dynamic Ideas.

Borndörfer, R., & Weismantel, R. (2000). Set packing relaxations of some integer programs. *Mathematical Programming, 88*, 425–450.

Bramel, J., & Simchi-Levi, D. (1997). On the effectiveness of set covering formulations for the vehicle routing problem with time windows. *Operations Research, 45*, 295–301.

Brearley, A., Mitra, G., & Williams, H. (1975). Analysis of mathematical programming problems prior to applying the simplex method. *Mathematical Programming, 8*, 54–83.

Brooke, A., Kendrick, D., & Meeraus, A. (1988). *GAMS, a user's guide*. Redwood City, CA: The Scientific Press.

Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers and Operations Research, 33*(10), 2972–2990.

Chan, L., Muriel, A., & Simchi-Levi, D. (1998a). Parallel machine scheduling, linear programming, and parameter list scheduling heuristics. *Operations Research, 46*, 729–741.

Chan, L., Simchi-Levi, D., & Bramel, J. (1998b). Worst-case analyses, linear programming and the bin-packing problem. *Mathematical Programming, 83*, 213–227.

Chvátal, V. (1983). *Linear programming*. New York: W. H. Freeman.

Cornuéjols, G. (2008). Valid inequalities for mixed integer linear programs. *Mathematical Programming B, 112*, 3–44.

Cornuejols, G., Nemhauser, G., & Wolsey, L. (1980). Worst-case and probabilistic analysis of algorithms for a location problem. *Operations Research, 28*, 847–858.

Cramton, P., Shoham, Y., & Steinberg, R. (2006). *Combinatorial auctions*. Cambridge, MA: MIT Press.

Danna, E., Rothberg, E., & LePape, C. (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming, 102*, 71–90.

Dantzig, G., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research, 8*, 101–111.

Drezner, Z., & Hamacher, H. (2004). *Facility location: Applications and theory*. Berlin: Springer.

Edmonds, J., & Johnson, E. L. (1973). Matching, Euler tours, and the Chinese postman. *Mathematical Programming, 5*, 88–124.

Fischetti, M., & Lodi, A. (2002). Local branching. *Mathematical Programming, 98*, 23–47.

Fischetti, M., Lodi, A., & Salvagnin, D. (2009). Just MIP It! In V. Maniezzo, T. Stuetzle, & S. Voss (Eds.), *MATHEURISTICS: Hybridizing metaheuristics and mathematical programming* (pp. 39–70). Berlin: Springer.

Fisher, M. L. (1981). The lagrangian method for solving integer programming problems. *Management Science, 27*, 1–18.

Fourer, R., Gay, D. M., & Kernighan, B. W. (1993). *AMPL: A modeling language for mathematical programming*. San Francisco: The Scientific Press.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W. H. Freeman.

Geoffrion, A. (1974). Lagrangian relaxation for integer programming. *Mathematical Programming Study, 2*, 82–114.

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting stock problem. *Operations Research, 9*, 849–859.

Glover, F., & Laguna, M. (1998). *Tabu search*. Berlin: Springer.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Golden, B., Raghavan, S., & Wasail, E. (2010). *The vehicle routing problem: Latest advances and new challenges*. Berlin: Springer.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Monthly, 64*, 275–278.

Gomory, R. E. (1960). *An algorithm for the mixed integer problem* (Tech. Rep. RM-2597). The RAND Corporation. Santa Monica, California.

Gu, Z., Nemhauser, G. L., & Savelsbergh, M. W. P. (1998). Cover inequalities for 0-1 linear programs: Computation. *INFORMS Journal on Computing, 10*, 427–437.

Gu, Z., Nemhauser, G. L., & Savelsbergh, M. W. P. (1999). Lifted flow covers for mixed 0-1 integer programs. *Mathematical Programming, 85*, 439–467.

Gu, Z., Nemhauser, G. L., & Savelsbergh, M. W. P. (2000). Sequence independent lifting. *Journal of Combinatorial Optimization, 4*, 109–129.

Guignard, M., & Kim, S. (1987). Lagrangian decomposition: A model yielding stronger lagrangian bounds. *Mathematical Programming, 39*, 215–228.

Guignard, M., & Spielberg, K. (1981). Logical reduction methods in zero-one programming: Minimal preferred inequalities. *Operations Research, 29*, 49–74.

Hammer, P. L., Johnson, E. L., & Peled, U. N. (1975). Facets of regular 0-1 polytopes. *Mathematical Programming, 8*, 179–206.

Hane, C., Barnhart, C., Johnson, E., Marsten, R., Nemhauser, G., & Sigismondi, G. (1995). The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming, 70*, 211–232.

Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming. *Proceedings of Congress on Numerical Methods in Combinatorial Optimization*, Italy.

Held, M., & Karp, R. M. (1970). The traveling salesman problem and minimum spanning trees. *Operations Research, 18*, 1138–1162.

Hoffman, K., & Padberg, M. (1985). LP-based combinatorial problem solving. *Annals of Operations Research, 4*, 145–194.

Hoffman, K. L., & Padberg, M. W. (1991). Improving LP-representations of zero-one linear programs for branch and cut. *ORSA Journal on Computing, 3*, 121–134.

Hoffman, K., & Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. *Management Science, 39*, 667–682.

Hoffman, K., & Villa, C. (2007). A column-generation and branch-and-cut approach to the bandwidth-packing problem. *Journal of Research of the National Institute of Standards and Technology, 111*, 161–185.

Hooker, J. (2002). Logic, optimization, and constraint programming. *INFORMS Journal on Computing, 14*, 295–321.

Hooker, J. (2007). Planning and scheduling by logic-based benders decomposition. *Operations Research, 55*, 588–602.

Jünger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleyblank, W., Reinelt, G. (2010). *Fifty years of integer programming: 1958–2008*. Berlin: Springer.

Kan, A. R. (1986). An introduction to the analysis of approximation algorithms. *Discrete Applied Mathematics, 14*, 111–134.

Karamanov, M., & Cornuéjols, G. (2009). Branching on general disjunctions. *Mathematical Programming*, 128, 403–406.

Karp, R. (1976). Probabilistic analysis of partitioning algorithms for the traveling salesman problem. In J. F. Traub (Ed.), *Algorithms and complexity: New directions and recent results* (pp. 1–19). New York: Academic.

Land, A. H., & Doig, A. G. (1960). An automatic method for solving discrete programming problems. *Econometrica, 28*, 497–520.

Linderoth, J., & Ralphs, T. (2005). Noncommercial software for mixed-integer linear programming. In J. Karlof (Ed.), *Integer programming: Theory and practice* (pp. 253–303). Boca Raton, FL: CRC Press.

Linderoth, J. T., & Savelsbergh, M. W. P. (1999). A computational study of search strategies in mixed integer programming. *INFORMS Journal on Computing, 11*, 173–187.

Marchand, H., & Wolsey, L. (2001). Aggregation and mixed integer rounding to solve MIPs. *Operations Research, 49*, 363–371.

Markowitz, H., & Manne, A. (1957). On the solution of discrete programming problems. *Econometrica, 2*, 84–110.

Martello, S., & Toth, P. (1990). *Knapsack problems*. New York: Wiley.

Martin, A. (2001). Computational issues for branch-and-cut algorithms. In M. Juenger & D. Naddef (Eds.), *Computational combinatorial optimization* (pp. 1–25). Berlin: Springer.

McAloon, K., & Tretkoff, C. (1996). *Optimization and computational logic*. New York: Wiley.

Mitten, L. (1970). Branch-and-bound methods: General formulation and properties. *Operations Research, 18*, 24–34.

Nediak, M., & Eckstein, J. (2001). *Pivot, cut, and dive: A heuristic for mixed 0-1 integer programming* (Tech. Rep. RUTCOR Research Report RRR 53-2001). Rutgers University, Newark, New Jersey.

Nemhauser, G. L., & Sigismondi, G. (1992). A strong cutting plane/branch-and-bound algorithm for node packing. *Journal of the Operational Research Society, 43*, 443–457.

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

Nemhauser, G., & Vance, P. (1994). Lifted cover facets of the 0-1 knapsack polytope with GUB constraints. *Operations Research Letters, 16*, 255–264.

Nemhauser, G., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. New York: Wiley.

Nemhauser, G., & Wolsey, L. (1990). A recursive procedure for generating all cuts for 0-1 mixed integer programs. *Mathematical Programming, 46*, 379–390.

Odlyzko, A. M. (1990). The rise and fall of knapsack cryptosystems. In C. Pomerance (Ed.), *Cryptology and* *computational number theory* (pp. 75–88). Ann Arbor: American Mathematical Society.

Owen, J., & Mehrotra, S. (2001). Experimental results on using general disjunctions in branch-and-bound for general-integer linear programs. *Computational Optimization and Applications, 20*(2).

Padberg, M. (1973). On the facial structure of set packing polyhedra. *Mathematical Programming, 5*, 199–215.

Padberg, M. (1974). Perfect zero-one matrices. *Mathematical Programming, 6*, 180–196.

Padberg, M. (1979a). Covering, packing and knapsack problems. *Annals of Discrete Mathematics, 4*, 265–287.

Padberg, M. W. (1979b). A note on 0-1 programming. *Operations Research, 23*, 833–837.

Padberg, M. W., & Rinaldi, G. (1991). A branch and cut algorithm for the solution of large scale traveling salesman problems. *SIAM Review, 33*, 60–100.

Papadimitriou, C., & Steiglitz, K. (1982). *Combinatorial optimization: Algorithms and complexity*. New Jersey: Prentice-Hall.

Parker, R., & Rardin, R. (1988). *Discrete optimization*. San Diego: Academic.

Parker, M., & Ryan, J. (1995). A column generation algorithm for bandwidth packing. *Telecommunication Systems, 2*, 185–196.

Pinedo, M. (2008). *Scheduling: Theory, algorithms, and systems*. Berlin: Springer.

Pochet, Y., & Wolsey, L. (1991). Solving multi-item lot sizing problems using strong cutting planes. *Management Science, 37*, 53–67.

Ralphs, T., & Galati, M. (2005). Decomposition in integer programming. In J. Karlof (Ed.), *Integer programming: Theory and practice* (pp. 57–110). Boca Raton, FL: CRC Press.

Rasmussen, R., & Trick, M. (2007). A benders approach to the constrained minimum break problem. *European Journal of Operational Research, 177*, 198–213.

Ravikumar, C. (1996). *Parallel methods for VLSI layout design*. Norwood, NJ: Ablex Publishing Corporation.

Rendl, F. (2010). Semidefinite relaxations for integer programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, & L. Wolsey (Eds.), *Fifty years of integer programming: 1958–2008* (pp. 687–726). Berlin: Springer.

Rothberg, E. (2007). An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing, 19*, 534–541.

Roy, T. J. V., & Wolsey, L. A. (1987). Solving mixed integer 0-1 programs by automatic reformulation. *Operations Research, 35*, 45–57.

Savelsbergh, M. W. P. (1994). Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing, 6*, 445–454.

Savelsbergh, M. W. P. (1997). A branch and price algorithm for the generalized assignment problem. *Operations Research, 45*, 831–841.

Schrijver, A. (1986). *Theory of linear and integer programming*. Chichester: Wiley.

Schrijver, A. (2003). *Combinatorial optimization: Polyhedra and efficiency*. Berlin: Springer.

Shah, R. (1998). *Optimization problems in SONET/WDM ring architecture*. Master's Essay, Rutgers University, Newark, NJ.

Vance, P. H., Barnhart, C., Johnson, E. L., & Nemhauser, G. L. (1994). Solving binary cutting stock problems by column generation and branch and bound. *Computational Optimization and Applications, 3*, 111–130.

Vance, P., Barnhart, C., Johnson, E., & Nemhauser, G. (1997). Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research, 45*, 188–200.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research, 48*, 111–128.

Vanderbeck, F., & Wolsey, L. (2010). Reformulation and decomposition of integer programs. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, & L. Wolsey (Eds.), *Fifty years of integer programming: 1958–2008* (pp. 431–504). Berlin: Springer.

Weingartner, H. (1963). *Mathematical programming and the analysis of capital budgeting problems*. Englewood Cliffs, NJ: Prentice Hall.

Weyl, H. (1935). Elementare theorie der konvexen polyheder. *Commentarii Mathematici Helvetici, 7*, 290–306.

Williams, H. (1985). *Model building in mathematical programming* (2nd ed.). New York: Wiley.

Wolsey, L. A. (1975). Faces for a linear inequality in 0-1 variables. *Mathematical Programming, 8*, 165–178.

Wolsey, L. A. (1976). Facets and strong valid inequalities for integer programs. *Operations Research, 24*, 367–372.

Wolsey, L. A. (1990). Valid inequalities for mixed integer programs with generalized and variable upper bound constraints. *Discrete Applied Mathematics, 25*, 251–261.

Wolsey, L. A. (1998). *Integer programming*. New York: Wiley.

Zhang, X. (2010). *Neural networks in optimization*. Berlin: Springer.