OPTIMIZATION

Toshihide Ibaraki
Kyoto University, Kyoto, Japan

Keywords: discrete optimization, combinatorial optimization, integer programming, integer polyhedron, total unimodularity, branch-and-bound, branch-and-cut, network flows, approximation algorithms, metaheuristics, local search, simulated annealing, tabu search, genetic algorithms.

Contents

Summary

In this chapter, we discuss discrete optimization problems. Among them, we first consider those problems formulated as the integer programming problem, and describe their mathematical properties derivable from coefficient matrices. Then algorithms to solve discrete optimization problems are explained, ranging from exact enumerative algorithms such as branch-and-bound and branch-and-cut, to approximation algorithms. As approximation algorithms, we present some simple ones with theoretical performance guarantees as well as metaheuristic algorithms such as iterated local search, simulated annealing, tabu search and genetic algorithms, which are being applied to many practical problems.

1. Introduction

An optimization problem is generally described as follows:

$Q:$ minimize (or maximize) $f(x)$

$$\text{subject to} \quad x \in F, \tag{1}$$

where F is a *feasible region*, which is a subset of the underlying space X, and f is an *objective function*. A solution $x \in X$ is called *feasible* if it belongs to F, and a feasible solution $x \in F$ is called *optimal* if no feasible solution $y \in F$ satisfies $f(y) < f(x)$. (If the problem is one of maximization, the condition is replaced by $f(y) > f(x)$.) We call an optimization problem Q as a *discrete optimization problem* (or *combinatorial optimization problem*) if F and/or X are discrete in some sense. Typically sets F and X are considered to be discrete if they are finite or countably infinite; e.g, the set of n-dimensional integer vectors, the set of n-dimensional 0-1 vectors, the set of permutations of n objects, the set of all subsets of an underlying finite set, and the set of all mappings from a finite set to a finite set.

As we shall see in the subsequent discussion, a variety of problems of practical importance can be formulated as discrete optimization problems. However, many of them are known to be computationally intractable as they belong to the class of NP-hard problems or other classes of higher complexity. Thus main efforts have been invested in two directions; one is to identify specially structured problems which permit efficient algorithms, and the other is to develop approximation algorithms that can obtain reasonably good solutions (if not optimal) requiring practically reasonable computation time. There are also attempts to develop exact algorithms that run in practical time if the problem instances are not very large.

Example 1.1 (traveling salesman problem): Given n cities $1, 2, \ldots, n$ and distances d_{ij} from city i to j, $1 \le i \le n$, $1 \le j \le n$, we want to find a shortest tour that visits all cities (i.e., a Hamiltonian cycle). This problem is called the *traveling salesman problem*. In general, we allow $d_{ij} \ne d_{ji}$, but, in many applications, $d_{ij} = d_{ji}$ holds for all i and j. In the latter case, the problem is called the *symmetric traveling salesman problem*. As a tour can be represented as a permutation $\pi = (i_1, i_2, \ldots, i_n)$ of n cities, the problem asks to find a permutation π that minimizes the tour length $d(\pi) = \sum_{k=1}^{n-1} d_{i_k i_{k+1}} + d_{i_n i_1}$. This is one of the most well known discrete optimization problems, and is NP-hard [8, 5].

Example 1.2 (set covering problem): For a finite ground set $M = \{1, 2, \ldots, m\}$, we are given a family of its subsets $M_j \subseteq M$, $j = 1, 2, \ldots, n$, as well as their weights c_j. The set covering problem asks to find a subfamily of subsets $M_{j_1}, M_{j_2}, \ldots, M_{j_k}$ such that their union covers M (i.e., $\cup_{l=1}^{k} M_{j_l} = M$) and the sum of their weights $\sum_{l=1}^{k} c_{j_l}$ is minimized. It has many applications such as optimal crew scheduling for flights and finding optimum locations of service facilities. This problem is NP-hard [5].

Example 1.3 (maximum satisfiability problem): Given n 0-1 variables $x_1, x_2, \ldots, x_n$, a subset $C_i \subseteq \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n\}$ is called a *clause*, where x_j and $\bar{x}_j$ are *literals*, and $\bar{x}_j = 1 - x_j$ holds. We say that C_i is satisfied by a 0-1 vector x if either $x_j = 1$ holds for at least one literal $x_j \in C_i$ or $x_j = 0$ holds for at least one literal $\bar{x}_j \in C_i$. Given a set of *m* clauses $\mathcal{C} = \{C_i \mid i = 1, 2, \ldots, m\}$ and their weights $w_i (> 0),$ we are asked to find an *x* that maximizes the sum of weights w_i of the clauses C_i satisfied by *x*. This is called the *maximum satisfiability problem*, which is also known to be NP-hard [5].

2. Integer Programming

2.1 Definitions

A discrete optimization problem in the following form is called the *integer programming problem*:

$$\begin{aligned}
&\text{minimize} && \sum_{j=1}^{n} c_j x_j \\
&\text{subject to} && \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, 2, \ldots, m \\
& && x_j \geq 0, \quad j = 1, 2, \ldots, n \\
& && x_j : \text{integers}, \quad j \in J,
\end{aligned} \qquad (2)$$

where *J* is a subset of $V = \{1, 2, \ldots, n\}$ and gives the index set of integer variables. If $J = V,$ this is the *all integer programming problem*, and otherwise (i.e., $J \subset V,$ where $\subset$ denotes the proper inclusion) the *mixed integer programming problem*. In many applications, integer variables are further restricted to be 0 or 1, and the resulting problems are respectively called the *all* 0-1 *programming problem* and *mixed* 0-1 *programming problem*, analogously to the above classification. In the special case of $J = \emptyset,$ the problem becomes the *linear programming problem*. An important difference between the integer programming problem and the linear programming problem is that, while the former is NP-hard, the latter can be solved in polynomial time by using interior point methods [9, 10].

It is clear to see that minimization in (2) can be changed to maximization by redefining $-f(x)$ as $f(x),$ and the inequalities $\geq$ in the constraint set can be changed to $\leq$ by appropriately multiplying –1 to coefficients a_{ij} and b_i. Similarly, equality constraints can be represented by combining inequality constraints. In the following, therefore, the objective function and constraints will be treated in convenient forms.

Example 2.1 (knapsack problem): The following integer programming problem with only one constraint of linear inequality is called the *knapsack problem*.

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^{n} c_j x_j \\ &\text{subject to} && \sum_{j=1}^{n} a_j x_j \le b \\ & && x_j : \text{nonnegative integers}, \quad j = 1, 2, \ldots, n, \end{aligned} \tag{3}$$

where a_j, c_j for all *j* and *b* are nonnegative integers. If x_j are further restricted to be 0 or 1, it is called the 0-1 *knapsack problem*.

Example 2.2 (set covering problem): The set covering problem defined in Example 1.2 can be formulated as the following integer programming problem.

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{n} c_j x_j \\ &\text{subject to} && \sum_{j=1}^{n} a_{ij} x_j \ge 1, \quad i = 1, 2, \ldots, m \\ & && x_j = 0 \text{ or } 1, \quad j = 1, 2, \ldots, n, \end{aligned} \tag{4}$$

where $a_{ij} = 1$ if $i \in M_j$, and 0 otherwise.

Example 2.3 (traveling salesman problem): The symmetric traveling salesman problem defined in Example 1.1 can be formulated as the following integer programming problem, where $V = \{1, 2, \ldots, n\}$ and 0-1 variables x_{ij} are introduced for all $i, j \in V$ with $i < j$. Note that a solution $x = \{x_{ij} \mid i < j\}$ of 0 -1 variables represents the set of edges (i, j) defined by $x_{ij} = 1$.

$$\begin{aligned} &\text{minimize} && \sum_{i<j} d_{ij} x_{ij} \\ &\text{subject to} && \sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \quad k = 1, 2, \ldots, n \\ & && \sum_{i<j,\ i,j\in W} x_{ij} \le |W| - 1, \quad \emptyset \ne W \subset V \\ & && x_{ij} = 0 \ or\ 1, \quad 1 \le i < j \le n. \end{aligned} \tag{5}$$

The first set of constraints says that, at each city *k*, two edges incident to *k* are selected. As this set of constraints allows as a solution, a set of disjoint cycles that together covers all cities, the second set of constraints is introduced to choose at most $|W| - 1$ edges in each *W*; thus prohibiting a short cycle contained in *W*.

In the following two subsections, we first consider special cases of the integer programming problem in which integer solutions are always obtained as a result of solving their linear programming relaxations, and then the general cases in which such nice property does not hold.

2.2 Total Unimodularity

Let us describe the constraints (2) without integrality constraint in the matrix form:

$$Ax \geq b, \;\; x \geq 0,$$

where *A* is an $m \times n$ matrix of coefficients a_{ij}, *b* is an *m*-vector of coefficients b_i and x is an *n*-vector of variables x_j. The set

$$P = \{x \mid Ax \geq b, x \geq 0\} \tag{6}$$

is the *polyhedron* defined by the above constraint. A polyhedron *P* is *integral* if all of its vertices are integer points. Such a polyhedron plays an important role in integer programming, since the linear programming relaxation obtained from (2) by ignoring the integrality constraint always gives an optimal solution that is integral. Thus, in this case, the linear programming relaxation solves the integer programming problem in polynomial time (by using a polynomial time algorithm for linear programming [9, 10]).

In order to characterize the above property, the following concept is introduced. A matrix *A* is called *totally unimodular* if every subdeterminant of *A* is 0, 1 or –1. This says, in particular, that each entry of A is 0, 1 or –1. Totally unimodular matrices arise in various applications. The following theorem is due to Hoffman and Kruskal [10].

Theorem 1 *A polyhedron* $\{x \mid Ax \geq b, x \geq 0\}$ *is integral for any integral vector b if and only if matrix A is totally unimodular.*

Example 2.4 (bipartite graphs): Let $G = (V, E)$ be an undirected graph, and let M be its incidence matrix; i.e., M is the 0-1 matrix with rows and columns indexed by the vertices and edges of *G*, respectively, where $M_{v,e} = 1$ holds if and only if *v* is an end vertex of edge *e*. Then it is known that M is totally unimodular if and only if *G* is bipartite.

Now, given an undirected graph $G = (V, E)$, the problem of finding the largest set V^* of independent vertices (i.e., no edge $e = (v, w) \in E$ satisfies $v, w \in V^*$) can be described as follows.

$$\begin{aligned} &\text{maximize} && \sum_{i \in V} y_i \\ &\text{subject to} && \sum_{i \in V} a_{ie} y_i \leq 1, \;\; e \in E \\ & && y_i = 0 \text{ or } 1, \;\; i \in V, \end{aligned} \tag{7}$$

where $a_{ie} = 1$ holds if i is an end vertex of $e \in E$, and 0 otherwise. Based on this formulation, the largest set $V^* = \{i \mid y_i^* = 1\}$ is obtained from its optimal solution y^*.

Similarly, the problem of finding the smallest set E^* of edges that cover all vertices (i.e., every vertex is incident to at least one edge in E^*) is described as follows by using the same coefficients a_{ie}.

$$\begin{array}{ll} \text{minimize} & \sum_{e \in E} x_e \\ \text{subject to} & \sum_{e \in E} a_{ie} x_e \geq 1, \ i \in V \\ & x_e = 0 \text{ or } 1, \ e \in E. \end{array} \tag{8}$$

If G is bipartite, the total unimodularity says that the linear programming relaxations of (7) and (8) (i.e., the 0-1 constraints are relaxed to $0 \leq y_i \leq 1$ and $0 \leq x_e \leq 1$, respectively) give integral optimal solutions of (7) and (8), respectively. Thus these integer programming problems can be solved in polynomial time. Furthermore, the duality theorem of linear programming tells that the optimum values of (7) and (8) coincide. This is known as König's theorem [2, 10] in graph theory.

Example 2.5 (directed graphs): Let $D = (V, E)$ be a directed graph, and let M be its incidence matrix; i.e., M is a matrix of 0, 1 and –1 with rows and columns indexed by the vertices and arcs of *D*, respectively, where $M_{v,a} = 1$ (resp., –1) holds if and only if arc *a* leaves (resp., enters) *v*. Note that a matrix is the incidence matrix of a directed graph if and only if each column contains exactly one entry of 1, exactly one entry of –1 and otherwise all 0 entries. Such matrices are always totally unimodular. The constraints using incidence matrices of directed graphs appear in network flow problems. In Section 4, two of such problems will be discussed.

Example 2.6 (network matrix): Let $D = (V, E)$ be a directed graph, and let $T = (V, E_0)$ be a directed tree in *D*. Let M be the matrix whose rows and columns are indexed by E_0 and E, respectively, and whose entries are defined as follows, where $a' \in E_0$, $a = (v, w) \in E$ and $\pi(v, w)$ denotes the unique path in *T* from *v* to *w*.

$$M_{a',a} = \begin{cases} 1, & \text{if } \pi(v,w) \text{ passes through } a' \text{ forwardly,} \\ -1, & \text{if } \pi(v,w) \text{ passes through } a' \text{ backwardly,} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The matrices M thus constructed are called *network matrices*. It is known that all network matrices are totally unimodular [10].

Important questions concerning total unimodularity would be to clarify their structures and to find an algorithm to determine whether a given matrix is totally unimodular or not. For the first question, the following theorem contains some of the known characterizations.

Theorem 2 *Let A be a matrix consisting of entries* 0, 1 *and* –1. *Then the following conditions are equivalent.*

(i) *A is totally unimodular.*

(ii) *Each collection of columns of A can be split into two parts such that the sum of the columns in one part minus the sum of the columns in the other part is a vector of entries* 0, 1 *and* –1.

(iii) *Each nonsingular submatrix of A has a row with an odd number of nonzero entries.*

(iv) *The sum of entries in any square submatrix with even row sum and even column sum is divisible by 4.*

(v) *No square submatrix of A has determinant* +2 *or* –2.

In developing an algorithm for total unimodularity, the network matrix of Example 2.6 plays an important role, since a theorem of Seymour states that any totally unimodular matrix can be obtained from network matrices and other two special matrices by applying certain simple operations (details are omitted). Based on these, the following theorem was proved [10].

Theorem 3 *Given a matrix A of entries* 0,1 *and* -1, *it can be checked in polynomial time whether A is totally unimodular or not.*

Related to integrality of a polyhedron, various concepts have been introduced such as *balanced matrices*, *total dual integrality* (TDI) and *perfect matrices*. Discussion on these concepts can be found in textbooks such as [10].

-
-
-

Bibliography

[1] Aarts E. and Lenstra J.K. (eds.) (1997). *Local search in Combinatorial Optimization*, 512 pp. Joan Wiley, Chichester. [A collection of thirteen survey papers describing aspects of local search algorithms and their applications.]

[2] Ahuja R.A., Magnanti T.L. and Orlin J.B. (1993). *Network Flows: Theory, Algorithms, and Applications*, 846 pp. Prentice-Hall International, Englewood Cliffs. [A standard textbook on network flow theory including such problems as shortest path problem, and maximum and minimum cost network flow problems.]

[3] Ausiellio G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A. and Protasi M. (1999). *Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag. [A textbook describing complexity and approximation algorithms for combinatorial optimization problems, with a useful compendium of representative problems.]

[4] Bäck T., Fogel G.B. and Michalewicz Z. (1997). *Handbook of Evolutionary Computation*, Oxford University Press, New York. [A most updated handbook on evolutionary computation and genetic algorithms, containing both theory and applications.]

[5] Garey M.R. and Johnson D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 338 pp. W.H. Freeman and Company, San Francisco. [A standard textbook on the complexity theory, particularly on NP-completeness, with a comprehensive list of NP-complete problems.]

[6] Grötschel M., Lovász L. and Schrijver A. (1988). *Geometric Algorithms and Combinatorial Optimization*, 362 pp. Springer-Verlag, Berlin. [A research monograph containing innovative polynomial time algorithms that emanate from geometric ideas of ellipsoid method and basis reduction.]

[7] Ibaraki T. (1987). *Enumerative Approaches to Combinatorial Optimization, Part I and Part II,* 602 pp. Baltzer Scientific Publishing Company, Basel. [A textbook describing enumerative methods such as dynamic programming and branch-and-bound algorithms for combinatorial optimization problems.]

[8] Lawler E.L., Lenstra J.K., Rinnoy Kan A.H.G. and Shmoys D.B. (1985). *The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization*, 465 pp. John Wiley, Chichester. [A collection of articles describing history, theory and various type algorithms of the travelling salesman problem.]

[9] Nemhauser G.L. and Wolsey L.A. (1988). *Integer and Combinatorial Optimization*, 763 pp. John Wiley, New York. [A standard textbook on integer programming, putting emphasis on algorithms and applications.]

[10] Schrijver A. (1986). *Theory of Linear and Integer Programming*, 471 pp. John Wiley, Chichester. [A standard text- book on linear and integer programming, putting emphasis on theory and complexity results.]

Biographical Sketch

Toshihide Ibaraki was born on 29th September 1940. He received BS, MS and PhD degrees respectively in 1963. 1965, and 1970 all from Kyoto University. From 1967 February to 1969 February he was a Research Associate at the University of Illinois. He was with Kyoto University from 1969 May to 1973 March as a Research Associate, from 1973 April to 1983 March as an Associate Professor, before taking up the Professorship at Toyohashi Institute of Technology. He returned in 1985 October to Kyoto University as a Professor.