Computer Science
and Scientific Computing

DISCRETE OPTIMIZATION

R. Gary Parker
Ronald L. Rardin

Discrete Optimization

R. Gary Parker

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia

Ronald L. Rardin

School of Industrial Engineering
Purdue University
West Lafayette, Indiana

ACADEMIC
Harcourt Brace Jov

Boston Orlando San Diego
New York Austin London Sydney
Tokyo Toronto

Preface

From the outset, our aim in pursuing this project has been to produce a book that would span, in a single volume, the host of fundamental issues and algorithmic strategies that have emerged as the general-purpose core in the discipline of discrete optimization. With roots in mathematical programming, computer science, and combinatorial mathematics, this field now attracts students, researchers, and practitioners alike having a variety of backgrounds and interests. While this diversity and its concomitant effect on fundamental results makes the task of unification arduous, we firmly believe that it also has been a source of substantial richness and has contributed directly to the ultimate maturation of the discipline.

Following a brief introduction, we begin in Chapter 2 with complexity theory. Here, we provide an overview of results that have proven central to the development of the intellectual framework of discrete optimization. Computations and supporting theory of generic approaches relevant to various discrete model contexts follow. For problems falling into the polynomial-time solvable category, we treat both matroid (Chapter 3) and linear programming-based (Chapter 4) procedures. In Chapters 5 and 6, we take up approaches that are inherently exponential in character, including in-depth coverage of enumerative (branch and bound) as well as cutting/polyhedral methods. We conclude (Chapter 7) with a generic treatment of common nonexact approaches.

We also would like to call attention to the inclusion of three appendices covering basic results from the theory of convex sets and polytopes, graphs, and linear programming. Our principal purpose in adding this material is to alleviate some potential frustration felt by those having uneven backgrounds in the aforementioned areas. Still others may find the coverage useful as a refresher.

Throughout this undertaking, we have tried to remain steadfast in our goal of making accessible to students, instructors, and researchers the substantial array of elegant results that have emerged in this field in the past quarter century. Algorithms are presented in computational format and demonstrated throughout with examples. Still, we have made every effort to maintain what we believe to be a suitable and consistent treatment of underlying theory. We have compiled lists of exercises for each chapter that span the range from routine applications of algorithms to new research results, with the more challenging problems appropriately marked with a star. In short, we have attempted to produce a book possessing both pedagogical value as a graduate text and reference value in discrete optimization research.

The material in this volume has been used at Georgia Tech and Purdue in both Masters- and PhD-level courses carrying such titles as "Integer Programming," "Combinatorial Optimization," and "Discrete Deterministic Models in Operations Research." Accordingly, we have found the modular organizational structure within chapters to be nicely suited for tailoring courses to fit varying student backgrounds and topical objectives. It also facilitates the capability of "entering" the book at specific sections of interest.

As a final comment regarding style we should mention the convention we have adopted pertaining to references. Naturally, we have attempted to be accurate, providing what we believe to be the principal citing(s) relative to given concepts, algorithms, theorems, and the like. On the other hand, we have not sought to be comprehensive in the sense of a literature review, choosing, instead, to compile a fairly extensive bibliography. Even so, there will no doubt be some references we have overlooked. Our only defense in this regard is an apology to those authors of works omitted and a hope that what we have included sufficiently supports the coverage comprising this book.

A large group of individuals have contributed both directly and indirectly to this probject. Some have used various parts of the book (or at least some manuscript version) in their courses at other universities. Some have read the manuscript and offered valuable comments on important matters regarding style, content, and clarity. Still others graciously provided us the benefit of their substantial expertise in a host of technical issues that

necessarily arise in a book of this scope. A very partial list of names would include J. Kennington, M. Karwan, R. Bulfin, V. Chandru, C. Tovey, J. Vande Vate, J. Jarvis, M. Bazaraa, and R. Jeroslow.

R. Gary Parker
Ronald L. Rardin

Contents

1

Introduction to Discrete Optimization

Anyone who has ever made responsible decisions of any kind has certainly encountered discrete decisions; decisions among a finite set of mutually exclusive alternatives. Stark choices between building and not building, turning left versus turning right, or visiting city A instead of city B, simply cannot be escaped. Of course, discreteness of the decision space offers the advantage of concreteness and indeed, elementary graphs or similar illustrations can often naturally and intuitively represent the meaning of a particular choice. Discreteness however, also brings forth a heavy burden of dimensionality. If more than a few choices are to be made, the decision-maker confronts an incomprehensible expanse of cases, combinations and possibilities requiring his or her evaluation.

Since the 18th century, this intriguing paradox of problems, possessing intuitive simplicity of presentation coupled with mind-boggling complexity of solution, has combined with the reality that discrete decisions abound in all areas of management and engineering to attract researchers to the study of discrete problems. Interest has multiplied with the revolutionary development of computing machines during the second half of the 20th century. For some problems, elegant solution procedures have been discovered. For most, though, a host of properties and algorithms have been developed, and considerable progress has attended, but no really

complete resolution has yet appeared. In a few cases, problems with apparently simple form have yielded to almost no advances at all.

But why is this the case? Why, in fact, does a class of problems that in many cases appear to pose modest demands often create severe difficulties? Moreover, given the present state of affairs, what are the prospects for resolution and, in the interim, what are we to do regarding solutions to discrete models of practical, real-world problems? The response to these sorts of questions constitutes, in large measure, the substance of this book.

1.1 Discrete Optimization Defined

Our concern is *discrete optimization,* the analysis and solution of problems mathematically modeled as the minimization or maximization of a value measure over a feasible space involving mutually exclusive, logical constraints. Enforcement of such logical constraints can be viewed abstractly as the arrangement of given elements into sets. Thus, in their most abstract mathematical form, discrete optimization problems can be expressed as

$$\begin{aligned} &\text{min (or max)} && \alpha(T) \\ &\text{subject to} && T \in F \end{aligned}$$

where T is an arrangement, F is the collection of feasible arrangements, and $\alpha(T)$ measures the value of members of F.

The study of arrangements is at the heart of the definition of *combinatorics.* Consequently, the reader will observe that we wish to view discrete optimization as a branch of combinatorics. This stance is motivated by our desire to keep absolutely central to the notion of discrete optimization the element of unavoidable choice among mutually exclusive alternatives. Discreteness is at the core, not the perifery of the problems we shall discuss.

Certainly, discrete optimization does not encompass all of the vast field of combinatorics. One popular classification recognizes four forms of combinatorial problems, distinguished by whether the question is *existence* of specific arrangements, versus *exhibition* or *evaluation* of required arrangements, versus *enumeration* or *counting* of possible arrangements, versus *extremization* of some measure over arrangements. To this extent, we equate discrete optimization with the last, extremization, branch of combinatorics.

Much of what follows is concerned with defining and presenting known results about particular discrete optimization problems. We shall briefly introduce some here in order to illustrate the enormous diversity of model

forms that have been studied. The informed reader will also note that we have included in this introductory list problems spanning the field in terms of tractability. Some are extremely well solved, while others continue to frustrate researchers after two centuries of attention.

Traveling Salesman Problem. Given a graph (directed or undirected) with specified weights on the edges, determine a closed traversal that includes every vertex of the graph exactly once and that has minimum total edge weight.

Postman's Problem. For a given graph (directed or undirected) with specified weights on the edges, determine a traversal that includes every edge in the graph at least once and that has minimum total weight.

Knapsack Problem. Determine a set of integer values x_i, $i = 1, 2, \ldots, n$ that minimizes $f(x_1, x_2, \ldots, x_n)$ subject to the restriction $g(x_1, x_2, \ldots, x_n) \geq b$ where b is a parameter.

Parallel Machine Scheduling. Given a set T of single operation tasks, each with processing time τ_j, $1 \leq j \leq |T|$, assign each task to exactly one of m machines so that the completion time of all tasks is minimized.

Vertex Coloring. Given an undirected graph, determine the minimum number of colors needed to color each vertex of the graph in order that no pair of adjacent vertices (vertices connected by an edge) share the same color.

Spanning Tree. Given an undirected graph with specified weights on the edges, determine a minimum total weight subset of edges that forms a connected, acyclic graph having at least one edge incident to every vertex.

Shortest Path. For a given graph (directed or undirected) with specified weights or lengths on the edges, find a minimum total length nonrepeating sequence of edges that connects two specified vertices and that conforms to any directions on edges.

Bin Packing. For a list of n weights, w_i, $1 \leq i \leq n$ and a set of bins, each with fixed capacity, say W, find a feasible assignment of weights to bins that minimizes the total number of bins used.

Matching. Given a list of items $i = 1, 2, \ldots, n$ and weights w_{ij} associated with pairing item i with item j, find a maximum total weight scheme for pairing items in the list so that each item is paired with one other at most.

Set Covering. Given a finite set S, a family of subsets $\{S_j \subseteq S: j \in J\}$ and costs, c_j, associated with the S_j, choose a minimum total cost collection of the subsets that includes every element of S at least once.

Maximum Flow. Given a graph (directed or undirected) and specified capacities on the edges, find a maximum edge flow between two specified vertices that conforms to capacities and has total flow into all other vertices equal to total flow out.

p-Median Problem. For a graph (directed or undirected) with specified weights on the edges, choose p vertices so that the sum of distances from all vertices to the closest of the chosen p is minimized.

Fixed Charge Problem. Given a feasible set S of nonnegative activity or traffic levels, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, unit costs v_j for employing x_j, and fixed costs f_j assessed whenever x_j is positive, choose a minimum total cost $\mathbf{x} \in S$.

1.2 Discrete Optimization and Integer Programming

All of the above discrete optimization problems, and every model we shall encounter, can be expressed in the form

$$
\begin{array}{lll}
\text{min (or max)} & \displaystyle\sum_{j=1}^{n} c_j x_j & \\
\text{s.t.} & \displaystyle\sum_{j=1}^{n} a_{ij} x_j \geq b_i & \text{for } i = 1, 2, \ldots, m \\
& x_j \geq 0 & \text{for } j = 1, 2, \ldots, n \\
& x_j \text{ integer} & \text{for } j \in I
\end{array} \tag{IP}
$$

Here the x_j are decision variables constrained by nonnegativity, m linear inequalities with coefficients a_{ij} and b_i, and the requirement that each x_j with $j \in I$ ($I \subseteq \{1, 2, \ldots, n\}$) be integral. We seek to minimize or maximize the sum of the x_j times given weights, c_j.

Problems in the form (*IP*) are known as linear integer programming problems, or more briefly, *integer programs.* If $I = \{1,2, \ldots ,n\}$ we call the problem a *pure integer program* and otherwise *mixed integer.* Aside from the obvious advantage of treating all problems in a single format, there can be considerable payoff in viewing discrete optimizations as integer programs. If the last, "x_j integer" constraints are dropped in (*IP*), we are left with a *linear program*—the best solved of all broad classes of optimization problems. General, and sometimes quite efficient, algorithms

can be derived for (*IP*) by exploiting this association with linear programming.

In the development to follow we shall present the more current of these general algorithms and emphasize how many procedures for specific discrete optimization problems can be interpreted as adaptions of general techniques. At the same time, it would be seriously oversimplifying matters to suggest that discrete optimization begins and ends with the study of general techniques for integer programming. Many discrete models can be forced into the (*IP*) format only at the cost of introducing immense numbers of variables and constraints. In such cases, general integer programming methods have little or no value, if for no other reason than that the underlying linear program is far beyond the capabilities of even the best of current algorithms. Furthermore, many discrete problems, including most of those that can be regarded as truly well-solved, have been conquered precisely because they have exploitable combinatorial structure, which is not present in general problems and thus not addressed by general algorithms.

It is mainly to avoid such oversimplification that we have chosen to define discrete optimization as a branch of combinatorics rather than a division of mathematical programming, even though it is without doubt a part of both. There is also a more subtle reason. The integrality constraints of (*IP*) formulations, which are totally or partially relaxed in general algorithms, are exactly the combinatorial, disjunctive characteristics that make the problems discrete. If discrete optimization problems are too casually couched in the integer programming format, it is very easy to be lulled into believing that relaxed structure is unimportant. How much difference can it make that some variable takes on the value 7/10 instead of 0 or 1? In truly discrete problems, it makes all the difference. The phenomenon coded by a 0–1 variable may simply have no interpretation when the variable assumes a fractional value; rounding to either 0 or 1 is in effect guessing a solution that might very well have been obtained without any optimization at all. Even though the integer programming point of view is highly relevant to much of discrete optimization, we believe one is less likely to be mislead if he or she keeps primary focus on combinatorial aspects of the problems.

1.3 Why are Discrete Optimization Problems Difficult to Solve?

Anyone who has had even the most passing introduction to discrete optimization problems is at least subtly aware of their inherent difficulty. What is it about discrete problems that makes them difficult to solve?

It is turning out that trying to provide a satisfactory answer to this question is one of the most intriguing phases of theoretical investigation in discrete optimization. Considerable progress has been achieved, and we shall devote a whole chapter to it in this book. Even at this introductory point, however, it seems appropriate to provide some preliminary insight into the fundamental causes of the intransigence that surrounds many discrete problems.

Superficially, it is quite clear why discrete problems are difficult. Their feasible solution spaces are enormous in size, and they grow explosively with the number of discrete choices to be resolved. For example, a problem requiring a modest 200 independent, binary decisions has 2^{200}, or about 10^{60} solutions to consider. A case with 201 binary decisions has twice as many.

This immense and exponentially growing size of discrete solution spaces categorically rules out a complete enumeration of the solutions in all but the smallest of cases. Often, the enumeration of even a tiny fraction of the set of solutions is computationally untenable.

A cynic might observe that the simplest of continuous optimization problems has a truly infinite solution space, and some such problems have been well-solved since the era of Newton and Liebnitz. There must be more to problem complexity than mere cardinality of the solution space.

Size of the solution space is relevant only if the problem must be solved by enumerating all, or at least a significant fraction, of the solutions. Unfortunately, that is precisely the state of our present knowledge regarding most discrete optimization problems. Furthermore, there is strong reason to believe that quite fundamental limits of our mathematics and computing machines will leave most discrete problems permanently in this "enumeration required" category.

To escape such enumeration, one must be able to find short cuts. Many successful algorithms take advantage of proofs that optimal solutions occur within a very small subset of the feasible points, e.g., the extreme points, inflextion points, etc. Others depend heavily on formal characterizations of progress. They stop only when theory is available to show that, since no further progress of the designated kind can be achieved, the current solution is necessarily optimal.

Numerous simplifying results of these two types are available in discrete optimization. With the exception of the minority of problems that are well-solved, however, none of the available results appears even close to providing completely satisfactory algorithms. Although they may reduce the number of solutions that need to be considered, and/or provide a conclusive progress test, the problem or test left to be resolved is usually a difficult discrete optimization problem.

1.4 Progress in Discrete Optimization

There is a degree of gloom in the picture we have painted so far. As a consequence, even the most acclaimed of discrete optimizers is sometimes (at least briefly) overcome by pessimism about the prospects for the field.

We find the challenge of dealing, however partially, with such difficult but important problems, and of establishing the boundaries of tractability in discrete optimization more than a little exciting. Our principal motivation in preparing this book is to collect the immense knowledge that is available.

There have been significant success stories in discrete optimization. Among these, one would necessarily include the landmark work on network flows of Ford and Fulkerson, the early work on cutting planes by Gomory, and the elegant results of Edmonds on optimum matchings. The past 10 to 15 years alone have brought forth some major developments including the initial efforts of Cook and later of Karp, which gave impetus to many of the results in complexity theory that now saturate the literature. During this period, the four color conjecture was converted to theorem status, a fairly complete theory of cutting planes emerged, Lagrangean techniques were successfully adapted to discrete optimization, and the ellipsoid and related methods provided a polynomial time solution scheme for linear programs.

But recent years have not been marked by theoretical developments alone. Recognition that general methods were not on the horizon focused attention on a host of special cases and examples of moderate size. A great deal of practical progress occurred. Genuinely large knapsack problems can now be treated efficiently (empirically speaking); certain scheduling problems have been conquered; and even the notorious traveling salesman problem has been humbled somewhat. Problems of sizes unheard of even a decade ago can now be rather routinely solved.

1.5 Organization of the Book

This book presents general theory and algorithms relevant to all parts of discrete optimization. We begin (in Chapter 2) on what has come to be the common language of discrete optimization—complexity theory. Included are the ideas of computability, worst-case analysis, and the notions of *P*, *NP*, and *NP*-Complete.

The remainder of this book is composed of five chapters addressed to the algorithmic categories delineated by complexity theory. In Chapters 3 and 4, we address polynomial procedures. Primary topics include matroids,

polynomial solution of linear programs, and their relation to combinatorial polyhedra. Chapter five covers the class of procedures commonly referred to as partial enumeration or branch and bound. Standard linear programming based methods are fully treated, along with newer Lagrangean dual ideas and Benders decomposition. In Chapter 6, polyhedral description or cutting methods are presented. All of the modern general theories of these approaches are discussed along with an introduction to the specialized facetial techniques that have proven useful on specific models. Finally, in Chapter 7, we turn to nonexact, heuristic procedures. After general discussions of how these algorithms may be evaluated, we define and investigate three broad classes: greedy, local search, and truncated exponential. We have also included three appendices in order to provide some background pertaining to fundamental results from linear programming, graph theory, convex sets and polytopes.

EXERCISES

1-1. Formulate each of the following problems defined in Section 1.1 in the (mixed-)integer format (*IP*) of Section 1.2. Then comment on the interpretation (if any) of a solution satisfying all constraints except the integrality requirements. Each formulation should employ a number of constraints and variables that grows as a low order polynomial (e.g., n^2, m^3) with the problem size.

(a) Traveling salesman problem (directed graph)
(b) Traveling salesman problem (undirected graph)
(c) Postman's problem (directed graph)
(d) Postman's problem (undirected graph)
(e) Knapsack problem ($f(x_1, x_2, \ldots, x_n) \triangleq \sum_{j=1}^{n} c_j x_j$, $g(x_1, x_2, \ldots, x_n) \triangleq \sum_{j=1}^{n} a_j x_j$ for integer c_j and a_j)
(f) Parallel machine scheduling
(g) Vertex coloring
(h) Spanning tree (directed graph, at most one inbound arc per vertex in the solution)
(i) Spanning tree (undirected graph)
(j) Shortest path (directed graph, weights nonnegative)
(k) Shortest path (undirected graph, weights nonnegative)
(l) Bin packing
(m) Matching
(n) Set covering

(o) Maximum flow (directed graph)
(p) Maximum flow (undirected graph)
(q) p-Median (directed graph)
(r) p-Median (undirected graph)
(s) Fixed charge problem ($S \triangleq \{\mathbf{x} \in \mathbb{R}^n: \mathbf{x} \geq \mathbf{0}, \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$ for integer matrix $\mathbf{A}$ and vector $\mathbf{b}$)

1-2. Formulate each of the following problems in the (mixed-)integer format (*IP*) of Section 1.2. Then, comment on the interpretation (if any) of a solution satisfying all constraints except the integrality requirements. Each formulation should employ a number of constraints and variables that grows as a low order polynomial (e.g., n^2,m^3) with the problem size.

(a) (Uncapacitated Facilities Location). Given a set of candidate facility locations $i = 1, 2, \ldots, m$, a set of customer demand points $j = 1, 2, \ldots, n$, positive construction costs f_i for constructing facility i, and nonnegative transportation cost v_{ij} for supplying demand j from facility i, find a minimum total cost collection of facilities to build such that each demand is allocated to one open facility.
(b) (Capacitated Facilities Location). Same as above except each candidate facility has an associated capacity, s_i, and each customer has an associated demand, d_j. Total demand of customers (partially or fully) assigned to a facility cannot exceed its capacity.
(c) (Assignment). Given n objects, n locations, and weights w_{ij} of assigning object i to location j, find a minimum total weight assignment of objects to locations (each object and location assigned once).
(d) (Generalized Assignment). Given m objects and n locations, weights w_{ij} of assigning object i to location j, capacities u_j of locations $j = 1, 2, \ldots, n$, and positive integer sizes a_i of objects $i = 1, 2, \ldots, m$, find a minimum total weight assignment of objects to locations that conforms to capacities (each object assigned once, total size assigned to a location $\leq$ capacity).
(e) (Edge Covering). Given an undirected graph with nonnegative weights w_{ij} on edges $e = (i, j)$, find a minimum total weight collection of edges such that each vertex is incident to at least one edge of the collection.
(f) (Tardiness Machine Scheduling). Given a set of tasks $j = 1, \ldots, n$, associated positive integer times τ_j required by the tasks on a single machine, due dates d_j for tasks and a collection $<$ of precedence pairs (i, j) indicating that task i must complete before task j can begin, find a sequencing of tasks on the machine that minimizes total due date violation.

1-3. Formulate each of the following problems (treated further in Section 2.3.2) in the (mixed-)integer format (*IP*) of Section 1.2. Then comment on the interpretation (if any) of a solution satisfying all constraints except the integrality requirements. Each formulation should employ a number of constraints and variables that grows as a low order polynomial (e.g., n^2, m^3) with the problem size.

(a) (Max Clique). Given an undirected graph, find the maximum k such that the graph contains a clique (vertex induced complete subgraph) of size k.

(b) (Vertex Cover). Given an undirected graph, and vertex weights, w_i, find the minimum weight collection of vertices such that every edge is incident to at least one vertex in the collection.

(c) (Set Packing). Given a finite set S, a family of subsets $\{S_j \subseteq S: j \in J\}$ and values v_j associated with each subset S_j, find a maximum total value collection of the subsets that includes every element of S at most once.

(d) (Set Partitioning). Given a finite set S, a family of subsets $\{S_j \subseteq S: j \in J\}$ and weights w_j associated with each subset S_j, find a minimum total weight collection of the subsets that forms a partition of S (every element of S occurs exactly once).

(e) (Steiner Tree). Given an undirected graph, a subset of distinguished vertices, S, and nonnegative edge weights, w_e, find a minimum weight subset of edges that forms a tree (connected, cycle free subgraph) having edges incident to every vertex of S.

(f) (Partition). Given positive integers $a_1, a_2, \ldots, a_n$ and corresponding weights $w_1, w_2, \ldots, w_n$, find a partition of $\{1, 2, \ldots, n\}$ into two subsets of equal a_j sum, such that the first has maximum total w_j weight.

(g) (Minimum Cut). Given a directed graph, two distinguished vertices $s \neq t$, and positive weights w_{ij} on arcs, find a minimum total weight cut separating s and t (arc subset including at least one element of every s-to-t path).