Introduction

A *discrete-optimization problem* is a problem of maximizing a real-valued *objective function* c on a finite set of *feasible solutions* $\mathcal{S}$. Often the set $\mathcal{S}$ naturally arises as a subset of 2^E (the set of all subsets of E), for some finite *ground set* E, in which case we have a *combinatorial-optimization problem*. Of course, there is no *problem* because we can just enumerate all feasible solutions – but we seek to do better. Usually, the feasible solutions are *described* in some concise manner, rather than being explicitly listed. The challenge is to develop algorithms that are provably or practically better than enumerating all feasible solutions.

Applications of discrete-optimization problems arise in industry (e.g., manufacturing and distribution, telecommunication-network design and routing, airline crew scheduling) and in applied sciences (e.g., statistics, physics, and chemistry).

Besides the applications, discrete optimization has aspects that connect it with other areas of mathematics (e.g., algebra, analysis and continuous optimization, geometry, logic, numerical analysis, topology, and, of course, other subdisciplines of discrete mathematics such as graph theory, matroid theory, and enumerative combinatorics) as well as computer science. Thus research in discrete optimization is driven by mathematics as well as by applications.

It is almost always the case that the set of feasible solutions $\mathcal{S}$ is delivered to us *descriptively* rather than by an explicit list. For example, $\mathcal{S}$ might be the set of spanning trees of a connected graph. As a complete graph on n vertices has n^{n-2} spanning trees (a nontrivial fact discovered by Cayley), it may come as quite a surprise that finding a 'maximum-weight' spanning tree is about as difficult as sorting the $\binom{n}{2} = n(n-1)/2$ edge weights. As another example, $\mathcal{S}$ might be the set of 'traveling-salesperson's tours' through n points in some metric space. There are $(n-1)!/2$ (equivalence classes of) such tours (as we may call any of the n points the initial point of the tour, and we can reverse the ordering of the points to obtain another tour of the same total length). The problem of finding

a shortest traveling-salesperson's tour is a notoriously difficult problem; yet we will touch on techniques that enable us to find good solutions for instances that are significantly larger than brute-force enumeration would permit.

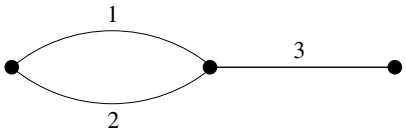
An algorithm is *theoretically efficient* for a class of problems if the number of computational steps required for solving instances of the problem is bounded by a polynomial in the number of bits required for encoding the problem (in which integers are encoded in base 2). We encode a rational number by encoding its integer numerator and denominator. This model does not permit the encoding of irrational numbers. To make all of this precise, we would need to carefully specify a model of computation (e.g., the Turing machine). Then, through notions of problem equivalence (e.g., polynomial-time reductions), we would define complexity classes (e.g., the class NP) and the idea of "completeness" for a complexity class. We will hardly touch on such issues in what follows, but a full appreciation of combinatorial optimization, from the point of view of "theoretical efficiency," requires such ideas.

The beacon of theoretical efficiency has its faults as an indicator of practical performance: (1) It is an asymptotic theory, (2) it is a worst-case theory, and (3) the order of the bounding polynomial may be quite high. Correspondingly, we note that (1) practical problems have some limited size, (2) practical instances may be quite different than worst-case instances, and (3) a high-order polynomial may grow too quickly in the limited range of problem sizes that are of practical concern. Still, this guiding light has shown the way to many practical methods.

For combinatorial-optimization problems, it will often be enlightening, and sometimes computationally effective, to embed our problem in $\mathbf{R}^E$ (real $|E|$-dimensional space with coordinates indexed by E). The natural method is as follows. We consider the convex hull $\mathcal{P}_{\mathcal{S}}$ of the set of characteristic vectors of sets in $\mathcal{S}$ – that is, the smallest convex set that contains these characteristic vectors. Next, we need to find a function $\tilde{c} : [0, 1]^E \mapsto \mathbf{R}$ such that, if $x(S)$ is the characteristic vector of a feasible set S, then $\tilde{c}(x(S)) = c(S)$. The success of such an approach depends, critically, on the form of the objective function. Concave functions are relatively easy to maximize (provided we have a description of $\mathcal{P}_{\mathcal{S}}$ as the solution set of linear inequalities), as in this case a local maximum is a global maximum. On the other hand, convex functions have the nice property that they are maximized by extreme points of a polytope – these extreme points are characteristic vectors of our feasible sets. For linear functions we have the best of both worlds. A *weight function* $c : 2^E \mapsto \mathbf{R}$ satisfies $c(S) = \sum_{e \in S} c(e)$, $\forall\ S \subset E$ [we take the liberty of writing $c(e)$ for $c(\{e\})$]. The weight function c naturally leads to the linear function $\tilde{c}$ defined by $\tilde{c}(x) = \sum_{e \in E} c(e)x_e$, $\forall\ x \in \mathbf{R}^E$; note that $c(S) = \tilde{c}(x(S))$. Most of the combinatorial-optimization problems that we will study involve optimizing weight functions. This does not mean

that we can easily solve all combinatorial-optimization problems involving the optimization of weight functions. The challenge in the approach that has been outlined is to find a useful description of $\mathcal{P}_{\mathcal{S}}$ by means of linear inequalities.

Next, we look at a concrete example. To visualize the geometry of the example, we are forced to use an instance with very few elements in the ground set. Our ground set $E := \{1, 2, 3\}$ corresponds to the set of edges of the following graph:



We define our set $\mathcal{S}$ of feasible sets to consist of subsets of E that are acyclic (i.e., contain no cycle). That is, $\mathcal{S}$ is the set of *forests* of the graph. Here

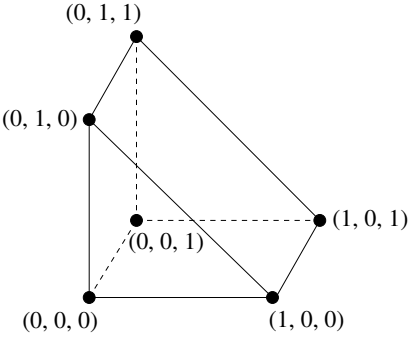
$$\mathcal{S} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 3\}, \{2, 3\}\}$$

(the only sets containing cycles are $\{1, 2\}$ and $\{1, 2, 3\}$).

We consider the characteristic vectors of sets in $\mathcal{S}$, namely,

$$\begin{gathered}(0, 0, 0),\\ (1, 0, 0),\\ (0, 1, 0),\\ (0, 0, 1),\\ (1, 0, 1),\\ (0, 1, 1).\end{gathered}$$

Next, we embed these points in $\mathbf{R}^E$, and we depict the convex hull $\mathcal{P}_{\mathcal{S}}$:



The polytope $\mathcal{P}_S$ is one-half of the unit cube. It is the solution set of the linear inequalities

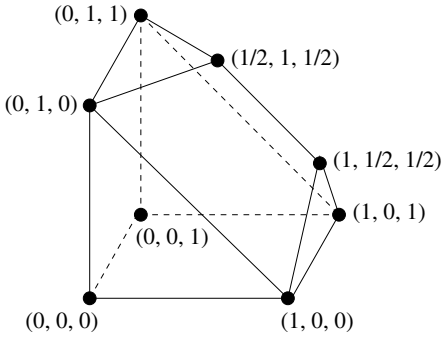
$$\begin{aligned} x_1 &\geq 0, \\ x_2 &\geq 0, \\ 1 \geq x_3 &\geq 0, \\ x_1 + x_2 &\leq 1\ . \end{aligned}$$

If, for example, we maximize the linear function $5x_1 + 4x_2 + x_3$ over the solutions to this inequality system, we get the optimal solution $x = (1, 0, 1)$, which is the characteristic vector of $\{1, 3\}$.

We may not be so fortunate if we model the points carelessly. For example, the set of linear inequalities

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

has precisely the same $0/1$ solutions as the inequality system that describes $\mathcal{P}_S$. It is easy to see that $(1, 1, 0)$ (the characteristic vector of $\{1, 2\}$) is the only $0/1$ vector excluded by $x_1 + x_2 - x_3 \leq 1$. Also, $(1, 1, 1)$ (the characteristic vector of $\{1, 2, 3\}$) is the only $0/1$ vector excluded by $x_1 + x_2 + x_3 \leq 2$. However, these inequalities describe a region that properly contains $\mathcal{P}_S$:



The difficulty with this latter set of inequalities is that there are linear functions having their unique maximum on the feasible region at a point with fractional components. For example, $5x_1 + 4x_2 + x_3$ (the objective function that we used earlier) has its unique maximum at $x = (1, 1/2, 1/2)$. So, if we do not work with the inequalities that describe the convex hull, we cannot

rely on linear programming to identify an optimal solution to the underlying combinatorial-optimization problem. Finally, we note that if we add $1/2$ of each of the inequalities

$$x_1 + x_2 - x_3 \leq 1,$$
$$x_1 + x_2 + x_3 \leq 2,$$

we get

$$x_1 + x_2 \leq 3/2.$$

Rounding down the right-hand-side constant, which we can do because the left-hand side will only take on integer values on 0/1 solutions, we recover the inequality

$$x_1 + x_2 \leq 1,$$

which is needed in the inequality description of $\mathcal{P}_S$.

Even if we have a description of the convex hull using linear inequalities, the situation is far more difficult for nonlinear maximization. For example, it is not hard to see that the function $2x_1 + x_2 - 3x_1x_2 + x_3$ is maximized on $\mathcal{P}_S$ by the point $x = (1, 0, 1)$. However, this function is not concave, so methods of nonlinear optimization that would seek a local minimum on the convex set $\mathcal{P}_S$ may fail to find the optimum. For example, the point $x = (0, 1, 1)$ is a strict local minimizer on $\mathcal{P}_S$. Therefore, it is hard to proceed from that point to the optimum by use of local methods of nonlinear programming. We can try to salvage something by transforming the objective function. The concave function $-3x_1^2 - 3x_2^2 - 3x_1x_2 + 5x_1 + 4x_2 + x_3$ takes on the same values at 0/1 vectors as the original function (we are just using the identity $x_j^2 = x_j$ when x_j is 0 or 1). This function has its unique maximum on $\mathcal{P}_S$ at $x = (2/3, 1/3, 1)$. However, this point is not a characteristic vector. Therefore, even though it is relatively easy to find this maximum by continuous local-search methods of nonlinear programming (maximizing a strictly concave function on a concave set is a situation in which finding a local maximizer is sufficient), the solution does not solve the underlying combinatorial-optimization problem. Finally, if we are clever enough to notice that the function $2x_1 + x_2 + x_3$ takes on the same values at *feasible* 0/1 vectors as the original function $2x_1 + x_2 - 3x_1x_2 + x_3$, then we can easily find $x = (1, 0, 1)$ as the solution of a linear program.

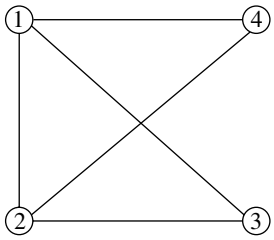
The important point to draw from this example is that continuous modeling must be done very carefully when variables are used to represent discrete choices in a combinatorial-optimization problem. This section closes with some Exercise and Problems that further develop this point.

Exercise (Comparing relaxations). The following three systems of inequalities have the same set of *integer-valued* solutions.

$$(I)\quad \begin{cases} x_1 + x_2 \le 1 & x_1 \ge 0 \\ x_1 + x_3 \le 1 & x_2 \ge 0 \\ x_1 + x_4 \le 1 & x_3 \ge 0 \\ x_2 + x_3 \le 1 & x_4 \ge 0 \\ x_2 + x_4 \le 1 & \end{cases}$$

$$(II)\quad \begin{cases} 2x_1 + 2x_2 + x_3 + x_4 \le 2 \\ 0 \le x_1 \le 1 \\ 0 \le x_2 \le 1 \\ 0 \le x_3 \le 1 \\ 0 \le x_4 \le 1 \end{cases} \qquad (III)\quad \begin{cases} x_1 + x_2 + x_3 \le 1 \\ x_1 + x_2 + x_4 \le 1 \\ x_1 \ge 0 \\ x_2 \ge 0 \\ x_3 \ge 0 \\ x_4 \ge 0 \end{cases}$$

In fact, the solutions to each system are the characteristic vectors of the "vertex packings" of the graph following – a *vertex packing* of G is just a set of vertices S with no edges of G between elements of S:



Compare how closely the three systems of inequalities approximate the set of integer-valued solutions in real space.

Problem (Uncapacitated facility location). The *uncapacitated facility-location problem* involves production and distribution of a single commodity at available facility locations, numbered $1, 2, \ldots, n$. Customers, numbered $1, 2, \ldots, m$ have demand for the commodity. A fixed-cost f_i is incurred if any (positive) production occurs at facility i. The profit (not accounting for the fixed costs) for satisfying the fraction x_{ij} of the demand of customer j from facility i is $c_{ij}x_{ij}$. The goal is to maximize net profit, subject to

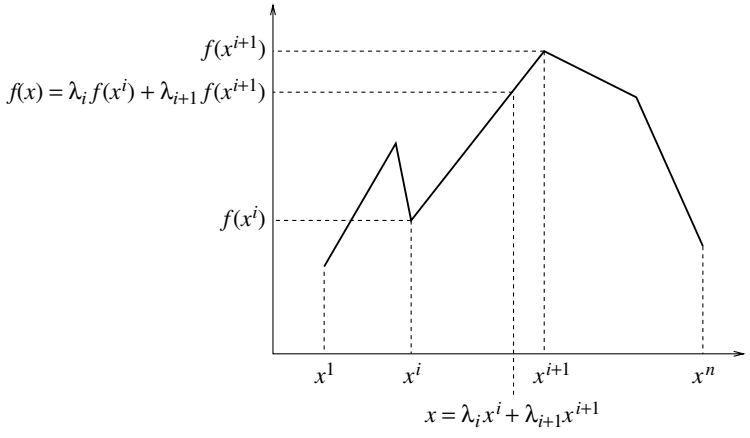
satisfying all customer demand exactly. We can formulate this problem as the program

$$\begin{aligned}
\max \; & -\sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \\
& \text{subject to:} \\
& \sum_{i=1}^{m} x_{ij} = 1, \text{ for } j = 1, 2, \ldots, n; \\
(*) \qquad & -n y_i + \sum_{j=1}^{n} x_{ij} \leq 0, \text{ for } i = 1, 2, \ldots, m; \\
& 0 \leq x_{ij} \leq 1, \text{ for } i = 1, 2, \ldots, m \text{ and} \\
& \qquad j = 1, 2, \ldots, n; \\
& 0 \leq y_i \leq 1 \text{ integer, for } i = 1, 2, \ldots, m.
\end{aligned}$$

Compare the strength of $(*)$ and

$$(**) \qquad -y_i + x_{ij} \leq 0, \text{ for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n.$$

Problem (Piecewise-linear functions). In practical instances of many optimization problems, key quantities, like costs, may not be well modeled as linear functions. In many instances, however, a piecewise-linear function is adequate. Let $x^1 < x^2 < \cdots < x^n$ be real numbers. We consider the piecewise-linear function $f : [x^1, x^n] \mapsto \mathbf{R}$ that we define by linearly interpolating f between the x^i. That is, if $x = \lambda_i x^i + \lambda_{i+1} x^{i+1}$, for some $\lambda_i, \lambda_{i+1} \geq 0$ with $\lambda_i + \lambda_{i+1} = 1$, then $f(x) := \lambda_i f(x^i) + \lambda_{i+1} f(x^{i+1})$:



The difficulty in formulating this with linear constraints is that the choice of i depends on x. Still, with 0/1 variables, we can make the choice. We employ the formulation

$$f(x) = \sum_{i=1}^{n} \lambda_i f(x^i);$$

$$\sum_{i=1}^{n} \lambda_i = 1; \qquad \sum_{i=1}^{n-1} y_i = 1;$$

$$\lambda_i \geq 0, \text{ for } i = 1, 2, \ldots, n; \qquad y_i \geq 0 \text{ integer, for } i = 1, 2, \ldots, n-1;$$

$(*)$ $y_i = 1 \Longrightarrow$ only λ_i and λ_{i+1} may be positive.

a. Explain why $(*)$ can be modeled by

$$(**) \quad \begin{cases} \lambda_1 \leq y_1 \\ \lambda_i \leq y_{i-1} + y_i, \text{ for } i = 2, 3, \ldots, n-1. \\ \lambda_n \leq y_{n-1} \end{cases}$$

b. Compare the strength of $(**)$ and

$$(***) \quad \begin{cases} \sum_{i=1}^{j} y_i \leq \sum_{i=1}^{j+1} \lambda_i, \text{ for } j = 1, 2, \ldots, n-2 \\ \sum_{i=j}^{n-1} y_i \leq \sum_{i=j}^{n} \lambda_i, \text{ for } j = 2, 3, \ldots, n-1 \end{cases}.$$

0

Polytopes and Linear Programming

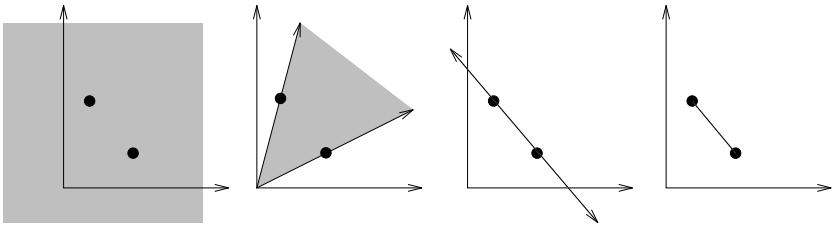
In this chapter, we review many of the main ideas and results concerning polytopes and linear programming. These ideas and results are prerequisite to much of combinatorial optimization.

0.1 Finite Systems of Linear Inequalities

Let x^k, $k \in N$, be a finite set of points in $\mathbf{R}^n$. Any point $x \in \mathbf{R}^n$ of the form $x = \sum_{k \in N} \lambda_k x^k$, with $\lambda_k \in \mathbf{R}$, is a *linear combination* of the x^k. If, in addition, we have all $\lambda_k \geq 0$, then the combination is *conical*. If $\sum_{k \in N} \lambda_k = 1$, then the combination is *affine*. Combinations that are both conical and affine are *convex*. The points $x^k \in \mathbf{R}^n, k \in N$, are *linearly independent* if $\sum_{k \in N} \lambda_k x^k = 0$ implies $\lambda_k = 0 \, \forall \, k \in N$. The points $x^k \in \mathbf{R}^n, k \in N$, are *affinely independent* if $\sum_{k \in N} \lambda_k x^k = 0, \sum_{k \in N} \lambda_k = 0$ implies $\lambda_k = 0 \, \forall \, k \in N$. Equivalently, the points $x^k \in \mathbf{R}^n, k \in N$, are *affinely independent* if the points $\binom{x^k}{1} \in \mathbf{R}^{n+1}, k \in N$, are *linearly independent*.

A set $X \subset \mathbf{R}^n$ is a *subspace/cone/affine set/convex set* if it is closed under (finite) linear/conical/affine/convex combinations. The *linear span/conical hull/affine span/convex hull* of X, denoted $\text{sp}(X)/\text{cone}(X)/\text{aff}(X)/\text{conv}(X)$, is the set of all (finite) linear/conical/affine/convex combinations of points in X. Equivalently, and this needs a proof, $\text{sp}(X)/\text{cone}(X)/\text{aff}(X)/\text{conv}(X)$ is the intersection of all subspaces/cones/affine sets/convex sets containing X.

In the following small example, X consists of two linearly independent points:



sp(X) cone(X) aff(X) conv(X)

A *polytope* is conv(X) for some finite set $X \subset \mathbf{R}^n$. A polytope can also be described as the solution set of a finite system of linear inequalities. This result is called Weyl's Theorem (for polytopes). A precise statement of the theorem is made and its proof is provided, after a very useful elimination method is described for linear inequalities.

Fourier–Motzkin Elimination is the analog of Gauss–Jordan Elimination, but for linear *in*equalities. Consider the linear-inequality system

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \ , \ \text{for } i = 1, 2, \ldots, m.$$

Select a variable x_k for elimination. Partition $\{1, 2, \ldots, m\}$ based on the signs of the numbers a_{ik}, namely,

$$\begin{aligned} S_+ &:= \{i \ : \ a_{ik} > 0\}, \\ S_- &:= \{i \ : \ a_{ik} < 0\}, \\ S_0 &:= \{i \ : \ a_{ik} = 0\}. \end{aligned}$$

The new inequality system consists of

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \text{ for } i \in S_0,$$

together with the inequalities

$$-a_{lj}\left(\sum_{j=1}^{n} a_{ij}x_j \leq b_i\right) + a_{ij}\left(\sum_{j=1}^{n} a_{lj}x_j \leq b_l\right),$$

for all pairs of $i \in S_+$ and $l \in S_-$. It is easy to check that

1. the new system of linear inequalities does not involve x_k;
2. each inequality of the new system is a nonnegative linear combination of the inequalities of the original system;