Chapter 2

POLYHEDRAL COMBINATORICS

Robert D. Carr
Discrete Mathematics and Algorithms Department, Sandia National Laboratories
bobcarr@cs.sandia.gov

Goran Konjevod
Computer Science and Engineering Department, Arizona State University
goran@asu.edu

Abstract Polyhedral combinatorics is a rich mathematical subject motivated by integer and linear programming. While not exhaustive, this survey covers a variety of interesting topics, so let's get right to it!

Keywords: combinatorial optimization, integer programming, linear programming, polyhedron, relaxation, separation, duality, compact optimization, projection, lifting, dynamic programming, total dual integrality, integrality gap, approximation algorithms

Introduction

There exist several excellent books [15, 24, 29, 35, 37] and survey articles [14, 33, 36] on combinatorial optimization and polyhedral combinatorics. We do not see our article as a competitor to these fine works. Instead, in the first half of our Chapter, we focus on the very foundations of polyhedral theory, attempting to present important concepts and problems as early as possible. This material is for the most part classical and we do not trace the original sources. In the second half (Sections 5–7), we present more advanced material in the hope of drawing attention to what we believe are some valuable and useful, but less well explored, directions in the theory and applications of polyhedral combinatorics. We also point out a few open questions that we feel have not been given the consideration they deserve.

2.1 Combinatorial Optimization and Polyhedral Combinatorics

Consider an undirected graph $G = (V, E)$ (i.e. where each edge is a set of 2 vertices (endpoints) in no specified order). A *cycle* $C = (V(C), E(C))$ in G is a subgraph of G (i.e. a graph with $V(C) \subseteq V$ and $E(C) \subseteq E$) that is connected, and in which every vertex has degree 2. If $V(C) = V$, then C is said to be a *Hamilton cycle* in G. We define the *traveling salesman problem* (TSP) as follows. Given an undirected graph $G = (V, E)$ and a cost function $c : E \to \mathbf{R}$, find a minimum cost Hamilton cycle in G. The TSP is an example of a *combinatorial optimization problem.* In fact, it is an NP-hard optimization problem [31], although some success has been achieved both in approximating the TSP and in solving it exactly [19].

More generally, a *combinatorial optimization problem* is by definition a set, and we refer to its elements as *instances.* Each instance further specifies a finite set of objects called *feasible solutions*, and with each solution there is an associated *cost* (usually an integer or rational number). Solving an instance of a combinatorial optimization problem means finding a feasible solution of minimum (in the case of a *minimization* problem)—or maximum (in the case of a *maximization* problem)—cost. The solution to a combinatorial optimization problem is an algorithm that solves each instance.

A first step in solving a combinatorial optimization problem instance is the choice of representation for the set of feasible solutions. Practitioners have settled on representing this set as a set of vectors in a finite-dimensional space over $\mathbf{R}$. The dimension depends on the instance, and the set of component indices typically has a combinatorial meaning special to the problem, for example, the edge-set E of a graph $G = (V, E)$ that gives rise to the instance (Section 2.2). However, we sometimes just use the index set $[n] = \{1, 2, \ldots, n\}$.

There often are numerous ways to represent feasible solutions as vectors in the spirit of the previous paragraph. Which is the best one? It is often most desirable to choose a vector representation so that the cost function on the (finite) set of feasible solutions can be extended to a linear function on $\mathbf{R}^I$ (where I is an index set) that agrees with the cost of each feasible solution when evaluated at the vector representing the solution.

In Theorems 2.1 and 2.2, we give just one concrete reason to strive for such a linear cost function. But first, some definitions (for those missing from this paragraph, see Section 2.2.2). Let $S \subset \mathbf{R}^I$ be a set of vectors (such as the set of all vectors that represent the (finitely many) feasible solutions to an instance of a combinatorial optimization problem, in some vector representation of the problem instance). A *convex combination* of a finite subset $\{v^i \mid i \in I\}$ of vectors in S is a linear combination $\sum_{i \in I} \lambda_i v^i$ of these vectors such that the scalar multiplier λ_i is nonnegative for each i ($\lambda_i \geq 0$) and

$$\sum_{i \in I} \lambda_i = 1.$$

Geometrically, the set of all convex combinations of two vectors forms the line segment having these two vectors as endpoints. A *convex set* of vectors is a set (infinite unless trivial) of vectors that is closed with respect to taking convex combinations. The *convex hull* of S (conv(S)) is the smallest (with respect to inclusion) convex set that contains S. For a finite S, it can be shown that conv(S) is a bounded set that can be obtained by intersecting a finite number of closed halfspaces. The latter condition defines the notion of a *polyhedron.* The first condition (that the polyhedron is also bounded) makes it a *polytope* (see Theorem 2.31). Polyhedra can be specified algebraically, by listing their defining halfspaces using inequalities as in (2.1).

Now, the theorems that justify our mathematical structures.

THEOREM 2.1 *If the cost function is linear, each minimizer over S is also a minimizer over* conv(S)*.*

THEOREM 2.2 *(Khachiyan, see the reference [15].) If the cost function is linear, finding a minimizer over a polyhedron is a polynomially solvable problem as a function of the size of the input, namely the number of bits in the algebraic description of the polyhedron (in particular, this will also depend on the number of closed halfspaces defining the polyhedron and the dimension of the space the polyhedron lies in).*

Proof: Follows from an analysis of the *ellipsoid method* or an *interior point method.* □

We refer to any problem of minimizing (maximizing) a linear function over a polyhedron as a *linear programming* problem, or just a *linear program.* Often we just use the abbreviation *LP*.

Consider a combinatorial optimization problem defined by a graph $G = (V, E)$ and a cost function on E. We define the size of this input to be $|V| + |E|$ and denote it by $|G|$. Denote the set of feasible solutions for an instance where the input graph is G by S_G. The following possibility, of much interest to polyhedral combinatorics, can occur. It is usually the case that $|S_G|$ grows as an exponential function of $|G|$. In this case, it may be hard to tell whether this combinatorial optimization problem is polynomially solvable. However, it is possible here that both the number of closed halfspaces defining $\text{conv}(S_G)$ and the dimension of the space that the convex hull $\text{conv}(S_G)$ lies in grow only as polynomial functions of $|G|$. Hence, if we can find this set of closed halfspaces in polynomial time, the above theorems allow us to immediately conclude that our optimization problem is polynomially solvable! This polyhedral approach may not be the most efficient way of solving our problem (and admittedly, there are annoying technical difficulties to overcome when the minimizer found for $\text{conv}(S_G)$ is not in S_G), but it is a sure-fire backup if more combinatorial approaches are not successful. We will find out later that polynomial solvability is also guaranteed if there exist polynomially many (as a function of $|G|$) sufficiently well-behaved *families* of halfspaces defining $\text{conv}(S_G)$. This involves the concept of *separation*, which we discuss in Section 2.3.2.

2.2 Polytopes in Combinatorial Optimization

Let us recall the traveling salesman problem of the previous section. When the input graph is undirected, we will refer to the *symmetric* traveling salesman problem (STSP), in contrast to the *asymmetric* traveling salesman problem (ATSP) that results when the input graph is directed. (For more on polytopes and polyhedra related to STSP and ATSP see the surveys by Naddef [28] and Balas and Fischetti [1].) It is convenient to assume that the input graph for the STSP is a complete graph $K_n = (V_n, E_n)$ on n vertices for some $n \geq 3$. Along with the complete graph assumption, we usually assume that the cost function satisfies the *triangle inequality* (i.e. defines a *metric* on V_n).

Recall that we wish to have a vector representation of Hamilton cycles so that our cost function is linear. If H is a Hamilton cycle, then the cost $c(H) := \sum_{e \in E(H)} c_e$. So, we cleverly represent each Hamilton cycle H in K_n by its *edge incidence vector* $\chi^{E(H)}$ defined by

$$\chi_e^{E(H)} := \begin{cases} 1 & e \in E(H), \\ 0 & e \in E_n \setminus E(H). \end{cases}$$

Note that now $c(H) = c \cdot \chi^{E(H)}$. Hence, we can extend our cost function from edge incidence vectors of Hamilton cycles to a linear function on the entire space $\mathbf{R}^{E_n}$ by simply defining $c(x) := c \cdot x$ for each $x \in \mathbf{R}^{E_n}$.

The convex hull of all edge incidence vectors of Hamilton cycles in K_n is a polytope called $STSP(n)$. We noted earlier that optimizing a linear cost function over $STSP(n)$ can be done in polynomial time in the number of closed halfspaces defining the polytope, the dimension of the space and the maximum number of bits needed to represent any coefficient in the algebraic description of the halfspaces (the size of the *explicit description*), using linear programming. Unfortunately, every known explicit description of $STSP(n)$ is exponential in size with respect to n. Note that in our representation the vector representing any Hamilton cycle has only integer values, in fact only values in $\{0, 1\}$. This suggests the new problems of *integer programming* (IP) and *0-1 programming.* For each of these problems, analyzing the geometry of polytopes is of value. We introduce some geometric concepts related to polyhedra and polytopes, starting with the notion of a polyhedron's dimension.

2.2.1 Affine Spaces and Dimension

Let $S = \{v^i \mid i \in I\}$ be a set of points in $\mathbf{R}^I$, where I is a finite index set. An *affine combination* on S is a linear combination $\sum_{i \in I} \lambda_i v^i$, where the scalar multipliers λ_i satisfy

$$\sum_{i \in I} \lambda_i = 1.$$

An *affine set (space)* is a set which is closed with respect to taking affine combinations. The *affine hull* $\mathrm{aff}(S)$ of a set S is the smallest (with respect to inclusion) affine set containing S.

We now give two definitions of *affine independence.*

Definition 2.3 *A finite set S is affinely independent iff for any point which can be expressed as an affine combination of points in S, this expression is unique.*

For the second definition, we introduce the notion of an *affine$_0$ combination* on S, which is a linear combination $\sum_{i \in I} \lambda_i v^i$ where the scalar multipliers λ_i satisfy

$$\sum_{i \in I} \lambda_i = 0.$$

Definition 2.4 *A finite set S is affinely independent iff the only affine$_0$ combination of points in S that gives the* 0 *(the origin) is that where the scalar multipliers satisfy $\lambda_i = 0$ for all $i \in I$.*

Definition 2.5 *A set B is an affine basis of an affine space U if it is affinely independent and* $\mathrm{aff}(B) = U$.

Theorem 2.6 *The affine hull* $\mathrm{aff}(S)$ *of a set S of points in a finite-dimensional space $\mathbf{R}^I$ has an affine basis of cardinality at most $|I| + 1$. Furthermore, every affine basis of* $\mathrm{aff}(S)$ *has the same cardinality.*

Finally, we can define the *dimension* of a polyhedron P.

Definition 2.7 *The dimension* $\dim(P)$ *of a polyhedron* P *(or any convex set) is defined to be* $|B|-1$*, where* B *is any affine basis of* $\mathrm{aff}(P)$*.*

For example, consider the one-dimensional case. Any two distinct points $v^1, v^2 \in \mathbf{R}^I$ define a one-dimensional affine space $\mathrm{aff}\{v^1, v^2\}$ called the *line* containing v^1, v^2.

If we replace "affine" in Definition 2.4 by "linear," and "points" by "vectors," we get the definition of *linear independence*:

Definition 2.8 *A set of vectors* $V = \{v^1, \ldots, v^k\}$ *is linearly independent, if the only linear combination of vectors in* V *that gives the* 0 *vector is that where the scalar multipliers satisfy* $\lambda_i = 0$ *for all* $i \in I$*.*

Note that linear independence could also be defined using a property analogous to that of Definition 2.3. Either affine or linear independence can be used for a development of the notion of dimension:

Theorem 2.9 *A set* $\{x^1, \ldots, x^k, x^{k+1}\}$ *is affinely independent iff the set* $\{x^1 - x^{k+1}, \ldots, x^k - x^{k+1}\}$ *is linearly independent.*

As in the definition of an affine space, a *linear (vector) space* is closed with respect to linear combinations. Let $\mathrm{span}(B)$ denote the smallest set containing all linear combinations of elements of B. There is also a notion of basis for linear spaces:

Definition 2.10 *A set* B *is a linear basis of a linear space* U *if it is linearly independent and* $\mathrm{span}(B) = U$*.*

Theorem 2.11 *The linear space* $\mathrm{span}(S)$*, for a set* S *of vectors in a finite-dimensional space* $\mathbf{R}^I$ *has a linear basis of cardinality at most* $|I|$*. Furthermore, every linear basis of* $\mathrm{span}(S)$ *has the same cardinality.*

Affine independence turns out to be more useful in polyhedral combinatorics, because it is invariant with respect to translation. This is not so with linear independence. Another way to see this distinction is to understand that linear (in)dependence is defined for vectors, and affine is for points. The distinction between points and vectors, however, is often blurred because we use the same representation ($|I|$-tuples of real numbers) for both.

2.2.2 Halfspaces and Polyhedra

A simple but important family of convex sets is that of *halfspaces.* We say that a set H is a halfspace in $\mathbf{R}^I$, if both H and its complement H^c are convex and nonempty. An equivalent definition can be given algebraically: a (*closed*) halfspace is a set of the form

$$\{x \in \mathbf{R}^I \mid a \cdot x \geq a_0\}, \tag{2.1}$$

where $a \in \mathbf{R}^I$ and $a_0 \in \mathbf{R}$. Every halfspace is either closed or open and every open halfspace is defined analogously but with a strict inequality instead. (Closed and open halfspaces are respectively closed and open with respect to the standard Euclidean topology on $\mathbf{R}^I$.) This algebraic representation is unique up to a positive multiple of a and a_0. The boundary of a (closed or open) halfspace is called a *hyperplane*, and can be described uniquely up to a non-zero multiple of (a, a_0) as the set

$$\{x \in \mathbf{R}^I \mid a \cdot x = a_0\}.$$

We denote the (topological) *boundary* of a halfspace H by ∂H. It is obvious which hyperplane forms the boundary of a given halfspace.

A *polyhedron* is defined as any intersection of a finite number of closed halfspaces, and thus is a closed convex set. The space $\mathbf{R}^I$ (intersection of 0 closed halfspaces), a closed halfspace (intersection of 1 closed halfspace, namely itself), and a hyperplane (intersection of the 2 distinct closed halfspaces that share a common boundary), are the simplest polyhedra. A bounded polyhedron is called a *polytope*. (The convex hull of a finite set of points is always a polytope, and conversely any polytope can be generated in this way: see Theorem 2.31.)

EXERCISE 2.12 *An alternative definition of a closed halfspace: a set H is a closed halfspace if and only if H is a closed convex set strictly contained in $\mathbf{R}^I$, the dimension of H is $|I|$, and its boundary intersects every line it does not contain in at most one point.*

EXERCISE 2.13 *The intersection of two polyhedra is a polyhedron.*

2.2.3 Faces and Facets

Consider a polyhedron $P \subseteq \mathbf{R}^I$. We say that a closed halfspace (inequality) is *valid* for P if P is contained in that closed halfspace. Similarly, a hyperplane (equation) can be *valid* for P. A halfspace H is *slack* for a set P if H is valid for P and the intersection of P with ∂H is empty.

Every closed halfspace H valid but not slack for a polyhedron P defines a (nonempty) *face* F of P, by

$$F := \partial H \cap P,$$

where ∂H is the boundary of H. Of course, F is also a polyhedron. A face of P that is neither empty nor P is said to be *proper*. A face F of P whose dimension satisfies

$$\dim(F) = \dim(P) - 1,$$

in other words, a maximal proper face, is said to be a *facet* of P. Finally, when a face of P consists of a single point x, this point is said to be an *extreme point* of P.

Corresponding naturally to each facet (face) are one or more *facet-defining* (*face-defining*) inequalities. The only case where this correspondence is unique is for facet-defining inequalities of a *full-dimensional* polyhedron P (i.e. $\dim(P) = |I|$). A polyhedron P that is not full-dimensional is said to be *flat*. The following theorem gives an interesting *certificate* that a polyhedron P is full-dimensional.

Theorem 2.14 *A polyhedron $P \neq \emptyset$ is full-dimensional iff one can find $x \in P$ such that every closed halfspace defining P is slack for x (i.e. the interior of P is non-empty, with x being in P's interior).*

A set of closed halfspaces whose intersection is $P \neq \emptyset$ forms a *description* of P. It turns out that there is a facet-defining inequality for every facet of P in any description of P. A *minimal description* consists only of facet-defining inequalities and a minimal set of inequalities (or equations if permitted) that define the affine hull $\text{aff}(P)$ of P. Given a minimal description of P, we refer to the set of hyperplanes (equations) that are valid for P as the *equality system* $P^=$ of P. That is,

$$P^= := \{(a, a_0) \in \mathbf{R}^I \times \mathbf{R} \mid a \cdot x = a_0 \text{ is valid for } P\}.$$

Consider the normal vectors for the hyperplanes of $P^=$, namely

$$N(P^=) := \{a \in \mathbf{R}^I \mid \exists a_0 \in \mathbf{R} \text{ s.t. } (a, a_0) \in P^=\}.$$

Let B be a linear basis for $N(P^=)$. The minimum number of equations needed to complete the description of P will then be $|B|$. We define $dim(P^=) := |B|$. An important theorem concerning the dimension of a polyhedron P follows.

Theorem 2.15 *If the polyhedron $P \neq \emptyset$ lies in the space $\mathbf{R}^I$, then*

$$\dim(P) = |I| - \dim(P^=).$$

A useful technique for producing a set of affinely independent points $\{x^1, \ldots, x^k\}$ of P is to find a set of valid inequalities (a^i, a_0^i) for $i \in [k-1]$ such that for each i,

$$\begin{array}{rcll} a^i \cdot x^i & > & a_0^i, & \\ a^i \cdot x^j & = & a_0^i, & \forall j > i. \end{array}$$

Theorem 2.16 *$\{x^1, \ldots, x^k\}$ is affinely independent iff for each $i \in [k-1]$, one can find (a^i, a_0^i) such that*

$$\begin{array}{rcll} a^i \cdot x^i & \neq & a_0^i, & \\ a^i \cdot x^j & = & a_0^i & \forall j > i. \end{array}$$

One can determine that a face F of P is a facet by producing $\dim(P)$ affinely independent points on F (showing F is at least a facet), and one point in $P \setminus F$ (showing F is at most a facet).

For a valid inequality $a \cdot x \geq a_0$, we say that it is *less than facet-defining* for P, if the hyperplane defined by the inequality intersects P in a nonmaximal proper face of P. We can use the following theorem to determine that an inequality $a \cdot x \geq a_0$ is less than facet-defining.

THEOREM 2.17 *The inequality $a \cdot x \geq a_0$ is less than facet-defining iff it can be derived from a (strictly) positive combination of valid inequalities $a^i \cdot x \geq a_0^i$, for $i = 1, 2$ and one can find $x^1, x^2 \in P$ such that*

$$\begin{array}{rcl} a^2 \cdot x^1 & > & a_0^2, \\ a^2 \cdot x^2 & = & a_0^2, \\ a \cdot x^2 & > & a_0. \end{array}$$

To understand the above, note that the first condition implies that the hyperplane $\{x \mid a^2 \cdot x = a_0^2\}$ does not contain P (that is, the second inequality is not valid as an equation). Together, the second and third condition imply that $a^1 \cdot x \geq a_0^1$ is not valid as an equation and the two valid inequalities $a^i \cdot x \geq a_0^i$, $i = 1, 2$, define distinct faces of P.

Finally, we give a certificate important for determining the dimension of an equality system for a polyhedron.

THEOREM 2.18 *A system of equations $a^i \cdot x = a_0^i$ for $i \in [k]$ is consistent and linearly independent iff one can find x^0 satisfying all k equations and x^i satisfying precisely all equations except equation i for all $i \in [k]$.*

2.2.4 Extreme Points

Extreme points, the proper faces of dimension 0, play a critical role when minimizing (maximizing) a linear cost function. Denote the set of extreme points of a polyhedron P by $\text{ext}(P)$. For any polyhedron P, $\text{ext}(P)$ is a finite set. For the following four theorems, assume that P is a polytope.

THEOREM 2.19 *Let $x \in P$. Then $x \in \text{ext}(P)$ iff it cannot be obtained as a convex combination of other points of P iff it cannot be obtained as a convex combination of other points of $\text{ext}(P)$ iff $\forall x^1, x^2 \in P$ s.t.*

$$\frac{1}{2}x^1 + \frac{1}{2}x^2 = x,$$

we have $x^1 = x^2 = x$.

THEOREM 2.20 *If there is a minimizer in P of a linear cost function, at least one of these minimizers is also in $\text{ext}(P)$. Moreover, any minimizer in P is a convex combination of the minimizers also in $\text{ext}(P)$.*

THEOREM 2.21 *Any point $x \in P$ can be expressed as a convex combination of at most* $\dim(P) + 1$ *points in* $\text{ext}(P)$.

THEOREM 2.22 *Any extreme point can be expressed (not necessarily in a unique fashion) as an intersection of* $\dim(P^{=})$ *valid hyperplanes with linearly independent normals and another* $\dim(P)$ *hyperplanes coming from facet-defining inequalities. Conversely, any such non-empty intersection yields a set consisting precisely of an extreme point.*

A set S is *in convex position* (or *convexly independent*) if no element in S can be obtained as a convex combination of the other elements in S. Suppose S is a finite set of solutions to a combinatorial optimization problem instance. We saw that forming the convex hull $\text{conv}(S)$ is useful for optimizing a linear cost function over S.

We have the following theorem about the extreme points of $\text{conv}(S)$.

THEOREM 2.23 $\text{ext}(\text{conv}(S)) \subseteq S$, *and* $\text{ext}(\text{conv}(S)) = S$ *if and only if S is in convex position.*

Hence, when minimizing (maximizing) over $\text{conv}(S)$ as a means of minimizing over S, one both wants and is (in principle) able to choose a minimizer that is an extreme point (and hence in S).

The extreme point picture will be completed in the next section, but for now we add these elementary theorems.

THEOREM 2.24 *Every extreme point of a polyhedron P is a unique minimizer over P of some linear cost function.*

Proof: Construct the cost function from the closed halfspace that defines the face consisting only of the given extreme point. □

Conversely, only an extreme point can be such a unique minimizer of a linear cost function. Also, we have the following deeper theorem, which should be compared to Theorem 2.22.

THEOREM 2.25 *If an extreme point x is a minimizer for P, it is also a minimizer for a cone Q (defined in Section 2.2.5) pointed at x and defined by a set of precisely* $\dim(P^{=})$ *valid hyperplanes and a particular set of* $\dim(P)$ *valid closed halfspaces whose boundaries contain x.*

Define the *relative interior* of a (flat or full-dimensional) polyhedron P to be the set of points $x \in P$ such that there exists a sphere $S \subset \mathbf{R}^I$ centered at x such that $S \cap \text{aff}(P) \subset P$. A point in the relative interior is topologically as far away from being an extreme point as possible. We then have

THEOREM 2.26 *If x is a minimizer for P and x is in the relative interior of P, then every point in P is a minimizer.*

The famous *simplex method* for solving a linear program has not yet been implemented so as to guarantee polynomial time solution performance [40], although it is fast in practice. However, a big advantage of the simplex method is that it only examines extreme point solutions, going from an extreme point to a neighboring extreme point without increasing the cost. Hence, the simplex method outputs an extreme point minimizer every time. In contrast, its polynomial time performance competitors, interior point methods that go through the interior of a polyhedron, and the ellipsoid method that zeros in on a polyhedron from the outside, have a somewhat harder time obtaining an extreme point minimizer.

2.2.5 Extreme Rays

A *conical combination* (or *non-negative combination*) is a linear combination where the scalar multipliers are all non-negative. A finite set S of vectors is *conically independent* if no vector in S can be expressed as a conical combination of the other vectors in S. Denote the set of all non-negative combinations of vectors in S by $\mathrm{cone}(S)$. Note that $0 \in \mathrm{cone}(S)$.

A (convex) *cone* is a set of vectors closed with respect to non-negative combinations. A *polyhedral cone* is a cone that is also a polyhedron. Note that the facet-defining inequalities for a (full-dimensional or flat) polyhedral cone are all of the form $a^i \cdot x \leq 0$.

THEOREM 2.27 *A cone C is polyhedral iff there is a conically independent finite set S such that* $\mathrm{cone}(S) = C$.

Such a finite set S is said to *generate* the cone C and (if minimal) is sometimes referred to as a *basis* for C. Suppose $x \in C$ has the property that when

$$\frac{1}{2}x^1 + \frac{1}{2}x^2 = x$$

for some $x^1, x^2 \in C$, it follows that x^1 and x^2 differ merely by a nonnegative scalar multiple. When this property holds, x is called an *extreme ray* of C. When C is a polyhedral cone that doesn't contain a line, then the set of extreme rays of C is its unique basis.

Given two convex sets $A, B \subseteq \mathbf{R}^I$, let $A+B = \{u+v \mid u \in A,\ v \in B\}$. The set $A + B$ is usually called the *Minkowski sum* of A and B and is also convex. This notation makes it easy to state some important representation theorems.

A *pointed cone* is any set of the form $a + C$, where a is a point in $\mathbf{R}^I$ and C a polyhedral cone that does not contain a line. Here, a is the unique extreme point of this cone. Similarly, a *pointed polyhedron* is a polyhedron that has at least one extreme point.

THEOREM 2.28 *A non-empty polyhedron P is pointed iff P does not contain a line.*

THEOREM 2.29 *A set $P \subset \mathbf{R}^I$ is a polyhedron iff one can find a polytope Q (unique if P is pointed) and a polyhedral cone C such that $P = Q + C$.*

An equivalent representation is described by the following theorem.

THEOREM 2.30 *Any (pointed) polyhedron P can be decomposed using a finite set S of (extreme) points and a finite set S^0 of (extreme) rays, i.e. written in the form*

$$P = \text{conv}(S) + \text{cone}(S^0).$$

This decomposition is unique up to positive multiples of the extreme rays only for polyhedra P that are pointed. On the other hand, the decomposition of Theorem 2.30 for the polyhedron formed by a single closed halfspace, which has no extreme points, is far from unique.

A consequence of these theorems, anticipated in the earlier sections is

THEOREM 2.31 *A set $S \subseteq R^I$ is a polytope iff it is the convex hull of a finite set of points in R^I.*

2.3 Traveling Salesman and Perfect Matching Polytopes

A combinatorial optimization problem related to the symmetric traveling salesman problem is the *perfect matching* problem. Consider an undirected graph $G = (V, E)$. For each $v \in V$, denote the set of edges in G incident to v by $\delta_G(v)$, or just $\delta(v)$. Similarly, for $\emptyset \subset S \subset V_n$, define $\delta(S) \subset E_n$ to be the set of those edges with exactly one endpoint in S. For $F \subset E_n$, we also use the notation $x(F)$ to denote $\sum_{e \in F} x_e$. A *matching* in G is a subgraph $M = (V(M), E(M))$ of G such that for each $v \in V(M)$,

$$|E(M) \cap \delta_G(v)| = 1.$$

A *perfect matching* in G is a matching M such that $V(M) = V$. The *perfect matching problem* is: given a graph $G = (V, E)$ and a cost function $c : E \to \mathbf{R}$ as input, find a minimum cost perfect matching. Unlike the STSP (under the assumption $P \neq NP$), the perfect matching problem is polynomially solvable (i.e. in P). However, as we will see later, there are reasons to suspect that in a certain sense, it is one of the hardest such problems.

As in the STSP, we use the edge incidence vector of the underlying graph to represent perfect matchings, so that we obtain a linear cost function. We consider the special case where the input graph is a complete graph $K_n = (V_n, E_n)$ on n vertices. We denote the perfect matching polytope that results from this input by $PM(n)$. (Of course, just as is the case with the traveling salesman problem, there is one perfect matching polytope for each value of the parameter n. Most of the time, we speak simply of *the perfect matching polytope*, or *the traveling salesman polytope*, etc.)

2.3.1 Relaxations of Polyhedra

We would like a complete description of the facet-defining inequalities and valid equations for both $STSP(n)$ and $PM(n)$ (we assume in this section that n is even so that $PM(n)$ is non-trivial). For $STSP(n)$, this is all but hopeless unless $P = NP$. But, there is just such a complete description for $PM(n)$. However, these descriptions can get quite large and hard to work with. Hence, it serves our interest to obtain shorter and more convenient approximate descriptions for these polytopes. Such an approximate description can actually be a description of a polyhedron that approximates but contain the more difficult original polyhedron Z_n, and this new polyhedron P_n is then called a *relaxation* of Z_n. A polyhedron P_n is a *relaxation* of Z_n (e.g. $STSP(n)$ or $PM(n)$), if $P_n \supseteq Z_n$. Often, an advantage of a relaxation P_n is that a linear function can be optimized over P_n in polynomial time in terms of n, either because it has a polynomial number of facets and a polynomial-size dimension in terms of n, or because its facets have a nice combinatorial structure. Performing this optimization then gives one a lower bound to the original problem of minimizing over the more intractable polytope, say $STSP(n)$.

The description of a relaxation will often be a subset of a minimal description for the original polyhedron. An obvious set of valid equations for $STSP(n)$ and $PM(n)$ results from considering the number of edges incident to each node $v \in V_n$ in a Hamilton cycle or a perfect matching. Hence, for each $v \in V_n$, we have the *degree constraints*:

$$\begin{aligned} x(\delta(v)) &= 2 \quad (STSP(n)), \\ x(\delta(v)) &= 1 \quad (PM(n)). \end{aligned}$$

We also have inequalities valid for both $STSP(n)$ and $PM(n)$ that result directly from our choice of a vector representation for Hamilton cycles and perfect matchings. These are $0 \leq x_e \leq 1$ for each $e \in E_n$. Thus,

$$\begin{array}{ll} x(\delta(v)) = 2 & \forall v \in V_n, \\ 0 \leq x_e \leq 1 & \forall e \in E_n, \end{array} \tag{2.2}$$

describes a relaxation of $STSP(n)$. Similarly,

$$\begin{array}{ll} x(\delta(v)) = 1 & \forall v \in V_n, \\ 0 \leq x_e \leq 1 & \forall e \in E_n, \end{array} \tag{2.3}$$

describes a relaxation of $PM(n)$. The polytopes $STSP(n)$ and $PM(n)$ both have the property that the set of integer points they contain, namely $\mathbf{Z}^{E_n} \cap STSP(n)$ ($\mathbf{Z}^{E_n} \cap PM(n)$) is in one-to-one correspondence with the set of extreme points of these polytopes, which are in fact also the vector representations of all the Hamilton cycles in K_n (perfect matchings in K_n). If all the extreme points are integral, and the vector representations for all the combinatorial objects are precisely these extreme points, we call such a polytope an *integer polytope.* When its integer solutions occur only at its extreme points, we call this polytope a *fully integer polytope.*

For the integer polytope Z_n of a combinatorial optimization problem, it may happen that a relaxation P_n is an *integer programming (IP) formulation.* We give two equivalent definitions of an IP formulation that are appropriate even when Z_n is not a fully integer polytope.

Definition 2.32 *The relaxation P_n together with integrality constraints on all variables forms an integer programming formulation for the polytope Z_n iff*

$$\mathbf{Z}^I \cap P_n = \mathbf{Z}^I \cap Z_n.$$

The second definition is:

Definition 2.33 *The relaxation P_n (together with the integrality constraints) forms an IP formulation for the polytope Z_n, iff*

$$P_n \supseteq Z_n \supset P_n \cap \mathbf{Z}^I.$$

These definitions can be modified for the *mixed integer* case where only some of the *variables* are required to be integers. One can see that the constraint set (2.3) results in an IP formulation for $PM(n)$. However, the constraint set (2.2) does not result in an IP formulation for $STSP(n)$ because it does not prevent an integer solution consisting of an edge-incidence vector for 2 or more simultaneously occurring vertex-disjoint non-Hamiltonian cycles, called *subtours.* To prevent these subtours, we need a fairly large number (exponential with respect to n) of *subtour* *elimination constraints.* That is, for each $\emptyset \subset S \subset V_n$, we need a constraint

$$x(\delta(S)) \geq 2 \tag{2.4}$$

to prevent a subtour on the vertices in S from being a feasible solution. Although (2.3) describes an IP formulation for $PM(n)$, this relaxation can also be strengthened by preventing a subtour over every odd-cardinality set S consisting of edges of value 1/2. That is, for each $\emptyset \subset S \subset V_n$, $|S|$ odd, we add a constraint

$$x(\delta(S)) \geq 1 \tag{2.5}$$

to prevent the above mentioned fractional subtour on S. We call these *odd cut-set constraints.* It is notable that for even $|S|$ the corresponding fractional subtours (of 1/2's) actually are in the $PM(n)$ polytope.

We saw that adding (2.4) to further restrict the polytope P_n of (2.2), we obtain a new relaxation P'_n of $STSP(n)$ that is *tighter* than P_n, that is

$$P_n \supset P'_n \supseteq STSP(n).$$

In fact, P'_n is an IP formulation of $STSP(n)$, and is called the *subtour relaxation* or *subtour polytope* ($SEP(n)$) for $STSP(n)$. Similarly, we can obtain a tighter relaxation P_n^{PM} for $PM(n)$ by restricting the polytope of (2.3) by the constraints of (2.5). In fact, adding these constraints yields the complete description of $PM(n)$, i.e. $P_n^{PM} = PM(n)$ [34]. However, we cannot reasonably hope that $SEP(n) = STSP(n)$ unless $P = NP$.

2.3.2 Separation Algorithms

Informally, we refer to the subtour elimination inequalities of $STSP(n)$ and the odd cut-set inequalities of $PM(n)$ as *families* or *classes* of inequalities. We will now discuss the idea that a relaxation P_n consisting of a family of an exponential number of inequalities with respect to n can still be solved in polynomial time with respect to n if the family has sufficiently nice combinatorial properties. The key property that guarantees polynomial solvability is the existence of a so-called *separation algorithm* that runs in polynomial time with respect to n and its other input (a fractional point x^*).

Definition 2.34 *A separation algorithm for a class $\mathcal{A}$ of inequalities defining the polytope $P_n^{\mathcal{A}}$ takes as input a point $x^* \in \mathbf{R}^I$, and either outputs an inequality in $\mathcal{A}$ that is violated by x^* (showing $x^* \notin P_n^{\mathcal{A}}$) or an assurance that there are no such violated inequalities (showing $x^* \in P_n^{\mathcal{A}}$).*

Without giving a very formal treatment, the major result in the field is the following theorem that shows the equivalence of separation and optimization.

THEOREM 2.35 *(Grötschel, Lovász and Schrijver, see the reference [15].) One can minimize (maximize) over the polyhedron Z_n using any linear cost function c in time polynomial in terms of n and size(c) if and only if there is a polynomial-time separation algorithm for the family of inequalities in a description of Z_n.*

(The notion of *size* used in the statement of this theorem is defined as the number of bits in a machine representation of the object in question.) The algorithm used to optimize over Z_n given a separation algorithm involves the so-called *ellipsoid algorithm* for solving linear programs (Section 2.4.3).

It turns out that there are polynomial time separation algorithms for both the subtour elimination constraints of $SEP(n)$ and the odd cut-set constraints of $PM(n)$. Hence, we can optimize over both $SEP(n)$ and $PM(n)$ in polynomial time, in spite of the exponential size of their explicit polyhedral descriptions. This means in particular that, purely through polyhedral considerations, we know that the perfect matching problem can be solved in polynomial time.

By analogy to the ideas of complexity theory [31], one can define the notion of *polyhedral reduction*. Intuitively, a polyhedron A can be reduced to the polyhedron B, if by adding polynomially many new variables and inequalities, A can be restricted to equal B. A combinatorial optimization problem L can then be polyhedrally reduced to the problem M, if any polyhedral representation A of L can be reduced to a polyhedral representation B of M.

EXERCISE 2.36 *Given a graph $G = (V, E)$ and $T \subseteq V$, a subset $F \subseteq E$ is a T-join if T the set of odd-degree vertices in the graph induced by F. Show that the V-join and perfect matching problems are equivalent via polyhedral reductions. (One of the two reductions is easy.)*

The idea that the perfect matching problem is perhaps the hardest problem in P is can be expressed using the following informal conjectures.

CONJECTURE 2.37 *[43] Any polyhedral proof that $PM(n)$ is polynomially solvable requires the idea of separation.*

CONJECTURE 2.38 *Every polynomially solvable polyhedral problem Z has a polyhedral proof that it is polynomially solvable, and the idea of separation is needed in such a proof only if Z can be polyhedrally reduced to perfect matching.*

2.3.3 Polytope Representations and Algorithms

According to Theorem 2.31, every polytope can be defined either as the intersection of finitely many halfspaces (an *$\mathcal{H}$-polytope*) or as the convex hull of a finite set of points (a *$\mathcal{V}$-polytope*). It turns out that these two representations have quite different computational properties.

Consider the $\mathcal{H}$-polytope P defined by $\{x \mid Ax \geq b\}$, for $A \in \mathbf{R}^{I \times J}$, $b \in \mathbf{R}^I$. Given a point $x^0 \in \mathbf{R}^J$, deciding whether $x^0 \in P$ takes $O(|I||J|)$ elementary arithmetic operations. However, if P is a $\mathcal{V}$-polytope, there is no obvious simple separation algorithm for P.

The separation problem for the $\mathcal{V}$-polytope $P = \text{conv}\{v^i \mid i \in I\}$ can be solved using linear programming:

$$\begin{array}{lll} \text{minimize} & \lambda_{i'} & \\ \text{subject to} & & \\ & \sum_{i \in I} \lambda_i v^i & = & x^0 \\ & \sum_{i \in I} \lambda_i & = & 1 \\ & \lambda & \geq & 0, \end{array}$$

where i' is an arbitrary element of I (the objective function doesn't matter in this case, only whether the instance is feasible). The LP (2.3.3) has a feasible solution iff x^0 can be written as a convex combination of extreme points of P iff $x^0 \in P$. Therefore any linear programming algorithm provides a separation algorithm for $\mathcal{V}$-polytopes. Even for special cases, such as that of *cyclic polytopes* [44], it is not known how to solve the membership problem (given x, is $x \in P$?) in polynomial time, without using linear programming.

Symmetrically, the optimization problem for $\mathcal{V}$-polytopes is easy (just compute the objective value at each extreme point), but optimization for $\mathcal{H}$-polytopes is exactly linear programming.

Some basic questions concerning the two polytope representations are still open. For example, the *polyhedral verification*, or the *convex hull* problem is, given an $\mathcal{H}$-polytope P and a $\mathcal{V}$-polytope Q, to decide whether $P = Q$. Polynomial-time algorithms are known for simple and simplicial polytopes [20], but not for the general case.

2.4 Duality

Consider a linear programming (LP) problem instance whose *feasible region* is the polyhedron $P \subset \mathbf{R}^I$. One can form a matrix $A \in \mathbf{R}^{m \times I}$ and $b \in \mathbf{R}^m$ such that

$$P = \{x \in \mathbf{R}^I \mid A \cdot x \geq b\}.$$

This linear program can then be stated as

$$\begin{array}{ll} \text{minimize} & c \cdot x \\ \text{subject to} & \\ & A \cdot x \geq b. \end{array}$$

In this section we derive a natural notion of a *dual* linear program. Hence, we sometimes refer to the original LP as the *primal.* In the primal LP, the *variables* are indexed by a combinatorially meaningful set I. We may want to similarly index the *constraints*, (currently they are indexed by integers $1, \ldots, m$). We simply write J for the index set of the constraints, and when convenient associate with J a combinatorial interpretation.

One way to motivate the formulation of the dual linear program is by trying to derive valid inequalities for P through linear algebraic manipulations. For each $j \in J$, multiply the constraint indexed by j by some number $y_j \geq 0$, and add all these inequalities to get a new valid inequality. What we are doing here is choosing $y \in \mathbf{R}^J_+$, and deriving

$$(y \cdot A) \cdot x = y \cdot (A \cdot x) \geq y \cdot b. \tag{2.6}$$

If by this procedure we derived the inequality

$$c \cdot x \geq c_0,$$

where $(c, c_0) \in \mathbf{R}^I \times \mathbf{R}$, then clearly c_0 would be a lower bound for an optimal solution to the primal LP. The idea of the dual LP is to obtain the greatest such lower bound. Informally, the dual problem can be stated as:

$$\begin{array}{lll} \text{maximize } c_0 & & \\ \text{subject to} & & \\ & c \cdot x \geq c_0 & \\ & & \text{can be derived from } A \cdot x \geq b \\ & & \text{by linear-algebraic manipulations.} \end{array}$$

More formally, the dual LP is (see (2.6))

$$\begin{array}{llll} \text{maximize} & y \cdot b & & \\ \text{subject to} & & & \\ & y \cdot A & = & c \\ & y & \geq & 0. \end{array}$$

In the next section we derive the *strong duality* theorem, which states that in fact all valid inequalities for P can be derived by these linear-algebraic manipulations.

If the primal LP includes the non-negativity constraints for all the variables, i.e. is

$$\begin{array}{llll} \text{minimize} & c \cdot x & & \\ \text{subject to} & & & \\ & A \cdot x & \geq & b \\ & x & \geq & 0, \end{array} \tag{2.7}$$

then valid inequalities of the form $c \cdot x \geq c_0$ can also be obtained by relaxing the equality constraint $y \cdot A = c$ to $y \cdot A \leq c$ ($\leq$ instead of merely $=$). Hence, the dual LP for this primal is

$$\begin{array}{llll} \text{maximize} & y \cdot b & & \\ \text{subject to} & & & \\ & y \cdot A & \leq & c \\ & y & \geq & 0. \end{array} \tag{2.8}$$

2.4.1 Properties of Duality

In the remainder of Section 2.4, we focus on the primal-dual pair (2.7) and (2.8). First, observe the following:

THEOREM 2.39 *The dual of the dual is again the primal.*

An easy consequence of the definition of the dual linear problem is the following *weak duality theorem.*

THEOREM 2.40 *If x is feasible for the primal and y is feasible for the dual, then*

$$c \cdot x \geq y \cdot b.$$

In fact, we can state something much stronger than this. A deep result in polyhedral theory says that the linear-algebraic manipulations described in the previous section can produce **all** of the valid inequalities of a polyhedron P. As a consequence, we have the *strong duality theorem.*

THEOREM 2.41 *Suppose the primal and dual polyhedra are non-empty. Then both polyhedra have optimal solutions (by weak duality). Moreover, if x^* is an optimal primal solution and y^* is an optimal dual solution, then*

$$c \cdot x^* = y^* \cdot b.$$

It should be noted that either or both of these polyhedra could be empty, and that if exactly one of these polyhedra is empty, then the objective function of the other is *unbounded.* Also, if either of these polyhedra has an optimal solution, then both of these polyhedra have optimal solutions, for which strong duality holds. This implies a very nice property of linear programming: an optimal solution to the dual provides a *certificate of optimality* for an optimal solution to the primal. That is, merely producing a feasible solution y^* to the dual proves the optimality of a feasible solution x^* to the primal, as long as their objective values are equal (that is, $c \cdot x^* = y^* \cdot b$). Some algorithms for the solution of linear programming problems, such as the simplex method, in fact naturally produce such dual certificates.

To prove the strong duality theorem, according to the previous discussion, it suffices to prove the following:

CLAIM 2.42 (STRONG DUALITY) *If (2.7) has an optimal solution x^*, then there exists a feasible dual solution y^* to (2.8) of equal objective value.*

Proof: Consider an extreme-point solution x^* that is optimal for (2.7). It clearly remains optimal even after all constraints that are not tight, that are either from A or are non-negativity constraints, have been removed from (2.7). Denote the system resulting from the removal of these non-tight constraints by

$$\begin{array}{rcl} A' \cdot x & \geq & b' \\ I' \cdot x & \geq & 0. \end{array}$$

That x^* is optimal can be seen to be equivalent to the fact that there is no improving feasible direction. That is, x^* is optimal for (2.7) if and only if there exists no $r \in \mathbf{R}^I$ such that $x^* + r$ is still feasible and $c \cdot (x^* + r) < c \cdot x^*$. (The vector r is called the *improving feasible direction.*) The problem of finding an improving feasible direction can be cast in terms of linear programming, namely

$$\begin{array}{lrcl} \text{minimize} & c \cdot r & & \\ \text{subject to} & & & \\ & A' \cdot r & \geq & 0 \\ & I' \cdot r & \geq & 0. \end{array} \tag{2.9}$$

Clearly, 0 is feasible for (2.9). Since x^* is optimal for the LP (2.7), 0 is the optimum for the "improving" LP (2.9).

Stack A' and I' to form the matrix $\overline{A}$. Denote the rows of $\overline{A}$ by $\{\overline{a}^1, \overline{a}^2, \ldots, \overline{a}^k\}$. Next we resort to the Farkas Lemma:

LEMMA 2.43 *The minimum of the LP* (2.9) *is* 0 *if and only if c is in the cone generated by $\{\overline{a}^1, \overline{a}^2, \ldots, \overline{a}^k\}$.*

Since x^* is optimal, the minimum for LP (2.9) is 0. Hence, by the Farkas Lemma, c is in the cone generated by $\{\overline{a}^1, \overline{a}^2, \ldots, \overline{a}^k\}$. Thus, there exist multipliers $\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_k \geq 0$ such that

$$c = \sum_{i=1}^{k} \overline{y}_i \overline{a}^i.$$

We now construct our dual certificate y^* from $\overline{y}$. First throw out any $\overline{y}_i$s that arise from the non-negativity constraints of I'. Renumber the rest of the $\overline{y}_i$s so that they correspond with the rows of A. We now define y^* by

$$y_i^* = \begin{cases} 0 & \text{if constraint } i \text{ of } A \text{ is not tight at } x^*, \\ \overline{y}_i & \text{otherwise.} \end{cases} \tag{2.10}$$

One can see that y^* is feasible for (2.8). From weak duality, we then have the chain

$$c \cdot x^* \geq (y^* \cdot A) \cdot x^* = y^* \cdot (A \cdot x^*) \geq y^* \cdot b. \tag{2.11}$$

It is easy to verify that the above chain of inequalities is in fact tight, so we actually have $c \cdot x^* = y^* \cdot b$, completing our proof. □

Proof of the Farkas Lemma: If c is in the cone of the vectors $\overline{a}^i$, then the inequality $c \cdot r \geq 0$ can be easily derived from the inequalities $\overline{a}^i \cdot r \geq 0$ (since c is then a non-negative combination of the vectors $\overline{a}^i$). But 0 is feasible for (2.9), so the minimum of (2.9) is then 0.

Now suppose c is not in the cone of the vectors $\overline{a}^i$. It is then geometrically intuitive that there is a separating hyperplane $r' \cdot a \geq r'_0$ passing through the origin (the $\vec{0}$-vector) such that all the vectors $\overline{a}^i$ (and thus the entire cone) are on one side (halfspace) of the hyperplane, and c is strictly on the other side. In fact, $r'_0 = 0$ since the origin lies on this hyperplane. We place the vector r' so that it is on the side of the hyperplane that contains the cone generated by the vectors $\overline{a}^i$. Then r' is feasible for (2.9), but $r' \cdot c < 0$, demonstrating that the minimum of (2.9) is less than 0. □

Let x^* be feasible for the primal and y^* feasible for the dual. Denote the entry in the j-th row and i-th column of the constraint matrix by $a_{j,i}$. Then, we have the *complementary slackness theorem*:

THEOREM 2.44 *The points x^* and y^* are optimal solutions for the primal and dual if and only if*

$$y_j^* (\sum_{i \in I} a_{j,i} x_i^* - b_j) = 0 \qquad \forall j \in J,$$

and

$$x_i^* (c_i - \sum_{j \in J} a_{j,i} y_j^*) = 0 \qquad \forall i \in I.$$

In other words, for x^*, y^* to be optimal solutions, for each $i \in I$, either the i-th primal variable is 0 or the dual constraint corresponding to this variable is *tight*. Moreover, for each $j \in J$, either the j-th dual variable is 0 or the primal constraint corresponding to this variable is tight.

EXERCISE 2.45 *Consider the linear program*

$$\begin{array}{lrcl} \textit{minimize} & c \cdot x & & \\ \textit{subject to} & & & \\ & A \cdot x & \geq & b \\ & I' \cdot x & \geq & 0, \end{array}$$

where the matrix $\begin{bmatrix} A \\ I' \end{bmatrix}$ *is square and* I' *consists of a subset of the set of rows of an identity matrix. Suppose* x^* *is the unique optimal solution to this linear program that satisfies all constraints at equality. Construct a dual solution* y^* *that certifies the optimality of* x^*.

EXERCISE 2.46 *(Due to R. Freund [35].) Given the linear program* $Ax \geq b$*, describe a linear program from whose optimal solution it can immediately be seen which of the inequalities are always satisfied at equality. (Hint: Solving a linear program with* $n^2 + n$ *variables gives the answer. Is there a better solution?)*

2.4.2 Primal, Dual Programs in the Same Space

We can obtain a nice picture of duality by placing the feasible solutions to the primal and the dual in the same space. In order to do this, let us modify the primal and dual linear programs by adding to them each other's variables and a single innocent constraint to both of them:

$$\begin{array}{lrcl} \text{minimize} & c \cdot x & & \\ \text{subject to} & & & \\ & A \cdot x & \geq & b \\ & x & \geq & 0 \\ & c \cdot x - y \cdot b & \leq & 0, \end{array}$$

$$\begin{array}{lrcl} \text{maximize} & y \cdot b & & \\ \text{subject to} & & & \\ & y \cdot A & \leq & c \\ & y & \geq & 0 \\ & c \cdot x - y \cdot b & \leq & 0. \end{array}$$

The y variables are in just one primal constraint in the revised primal LP. Hence, this constraint is easy to satisfy and does not affect the x variable values in any optimal primal solution. Likewise for the x variables in the revised dual LP.

Now comes some interesting geometry. Denote the optimal primal and dual objective function values by z_p and z_d, as if you were unaware of strong duality.

THEOREM 2.47 (WEAK DUALITY) *If the primal and dual linear programs are feasible, then the region*

$$\{(x, y) \in \mathbf{R}^{I \cup J} \mid 2z_p \geq c \cdot x + b \cdot y \geq 2z_d\},$$

and in particular the hyperplane

$$\{(x, y) \in \mathbf{R}^{I \cup J} \mid c \cdot x + b \cdot y = z_p + z_d\},$$

separates the revised primal and revised dual polyhedra.

THEOREM 2.48 (STRONG DUALITY) *If the primal and dual linear programs are feasible, then the intersection of the revised primal polyhedron and the revised dual polyhedron is non-empty.*

2.4.3 Duality and the Ellipsoid Method

The ellipsoid method takes as input a polytope P (or more generally a bounded convex body) that is either full-dimensional or empty, and gives as output either an $x \in P$ or an assurance that $P = \emptyset$. In the typical case where P may not be full-dimensional, with extra care, the ellipsoid method will still find an $x \in P$ or an assurance that $P = \emptyset$. With even more care, the case where P may be unbounded can be handled too.

The ellipsoid method, when combined with duality, provides an elegant (though not practical) way of solving linear programs. Finding optimal primal and dual solutions to the linear programs (2.7) and (2.8) is equivalent (see also Chvátal [7], Exercise 16.1) to finding a point in the polyhedron P defined by

$$P := \{(x, y) \in \mathbf{R}^{I \cup J} \mid A \cdot x \geq b,\ y \cdot A \leq c,\ x, y \geq 0,\ c \cdot x - y \cdot b \leq 0\}.$$

Admittedly, P is probably far from full-dimensional, and it may not be bounded either. But, consider the set of ray directions of P, given by

$$\begin{aligned} R \ := \ & \{(s, t) \in \mathbf{R}^{I \cup J} \mid A \cdot s \geq 0,\ t \cdot A \leq 0,\ s, t \geq 0,\ c \cdot s, y \cdot t = 0, \\ & s(I) + t(J) = 1\}. \end{aligned}$$

(For the theory of the derivation of this kind of result, see Nemhauser and Wolsey [29].) Since the polytope R gives the ray directions of P,

P is bounded if $R = \emptyset$. Conversely, if $R \neq \emptyset$, then P is unbounded or empty.

Suppose we are in the lucky case where $R = \emptyset$, and thus P is a polytope (the alternative is to artificially bound P). We also choose to ignore the complications arising from P being almost certainly flat. The ellipsoid method first finds an ellipsoid $\mathcal{E}$ that contains P and determines its center x. We run a procedure required by the ellipsoid algorithm (and called the *separation algorithm*) on x. The separation algorithm either determines that $x \in P$ or finds a closed halfspace that contains P but not x. The boundary of this halfspace is called a *separating hyperplane*. From this closed halfspace and $\mathcal{E}$, both of which contain P, the ellipsoid method determines a provably smaller ellipsoid $\mathcal{E}'$ that contains P. It then determines the center x' of $\mathcal{E}'$ and repeats the process. The sequence of ellipsoids shrinks quickly enough so that in a number of iterations polynomial in terms of $|I \cup J|$, we can find an $x \in P$ or determine that $P = \emptyset$, as required.

The alternative to combining duality with the ellipsoid method for solving linear programs is to use the ellipsoid method together with binary search on the objective function value. This is, in fact, a more powerful approach than the one discussed above, because it only requires a linear programming formulation and a separation algorithm, and runs in time polynomial in $|I|$, regardless of $|J|$. Thus the ellipsoid method (if used for the "primal" problem, with binary search) can optimize over polytopes like the matching polytope $PM(n)$ in spite of the fact that $PM(n)$ has exponentially many facets (in terms of n). The size of an explicit description in and of itself simply has no effect on the running time of the ellipsoid algorithm.

2.4.4 Sensitivity Analysis

Sensitivity analysis is the study of how much changing the input data or the variable values affects the optimal LP solutions.

2.4.4.1 Reduced Cost. Let x^* and y^* be optimal solutions to (2.7)-(2.8). The *reduced cost* c_e^{red} of a variable x_e is defined to be

$$c_e^{red} := c_e - (y^* \cdot A)_e.$$

Note that our dual constraints require that $c_e^{red} \geq 0$. Moreover $c_e^{red} > 0$ only when $x_e^* = 0$ by complementary slackness. The reduced cost is designed to put a lower bound on the increase of the cost of the primal solution when it changes from the solution x^*, where $x_e^* = 0$, to a feasible solution x^+, where $x_e^+ = 1$.

THEOREM 2.49 *If x^+ is a feasible primal solution, and $e \in E$ such that $x_e^* = 0$, but $x_e^+ = 1$, then $c \cdot x^+ \geq c \cdot x^* + c_e^{red}$.*

Proof: Complementary slackness implies $c \cdot x^+ - y^* \cdot b \geq c_e^{red}$. □

The reduced cost is of use when the primal is a relaxation of a 0-1 program or integer program. In particular, if $c \cdot x^* + c_e^{red}$ (can be rounded up for integral c) exceeds the cost of a known feasible integer solution, then the variable indexed by e can never appear in any integer optimal solution. Hence, this variable can be permanently removed from all formulations.

Another use of reduced cost is when the (perhaps "exponentially many") x_e variables are not all explicitly present in the linear program, and one is using *variable generation* (or *column generation*) to add them into the formulation when this results in a lower cost solution. Variable generation is analogous to the ideas of separation, whether using the ellipsoid method or a simplex LP method.

Assume that $|I|$ grows exponentially ($O(2^n)$) with respect to n, but $|J|$ grows polynomially. If a variable x_e is not currently in the n-th formulation, then implicitly, $x_e^* = 0$. If $c_e^{red} < 0$, then there is a chance that adding x_e into the LP formulation will lower the cost of the resulting optimal solution. A valuable *pricing algorithm* (or *dual separation algorithm*) is one that in $O(\log^k |I|)$ steps, either finds an $e \in I$ such that $c_e^{red} < 0$, or an assurance that there is no such $e \in I$. In this latter case, the current solution x^* is in fact optimal for the entire problem! With such a pricing algorithm, the ellipsoid method guarantees that these LP problem instances can be solved in polynomial time with respect to n.

Finally, there is an analogous dual concept of reduced cost, which can be easily derived.

2.4.4.2 Changing Right Hand Sides.

Now we address how the optimal LP solutions are affected by changing the right hand side b or the cost c. We will address the case when b is altered only, since the case when c is altered can be easily derived from this. Suppose we add δ to the right hand side for $j \in J$. The dual solution y^* that was once optimal is still feasible. However, the dual objective function has changed from b to some b' (differing only in row index j). So,

$$b' \cdot y^* = b \cdot y^* + \delta y_j^* = c \cdot x^* + \delta y_j^*.$$

Since y^* is known only to be feasible for the new dual, the optimal solution to this dual can only be better. Since the dual problem is a maximization problem, the value of the new optimal dual solution is thus at least δy_j^* more than the value of the old optimal dual solution

y^*. Hence, the new optimal primal solution value is also at least δy_j^* more than that of the old optimal primal solution x^* (now potentially infeasible). In other words, the cost of the best primal solution has gone up by at least δy_j^*.

2.5 Compact Optimization and Separation

2.5.1 Compact Optimization

Consider as an example the following optimization problem: given the integer k and a linear objective function c, maximize $c \cdot x$ under the constraint that the L^1 norm $\|x\|_1$ of $x \in \mathbf{R}^I$ is at most k (by definition, $\|x\|_1 = \sum_{i \in I} |x_i|$). This optimization problem at first leads to a linear programming formulation with $2^{|I|}$ constraints, namely for every vector $a \in \{-1, 1\}^I$, we require that $a \cdot x \leq k$. Introducing $|I|$ new variables makes it possible to reduce the number of constraints to $2|I| + 1$ instead. For each $i \in I$, the variable y_i will represent the absolute value $|x_i|$ of the i-th component of x. This is achieved by imposing the constraints $y_i \geq x_i$ and $y_i \geq -x_i$. Then the $2^{|I|}$ original constraints are replaced by a single one:

$$\sum_{i \in I} y_i \leq k.$$

It turns out that introducing additional variables can make it easier to describe the polyhedron using linear inequalities. In this section, we denote the *additional variables* (those not needed to evaluate the objective function) by y, and the *decision variables* (those needed to evaluate the objective function) by x.

Suppose we have a linear objective representation for each instance of a combinatorial optimization problem. We informally define an approach to describing the convex hull of feasible solutions for such a linear objective representation. A *linear description* is an abstract mathematical description or algorithm that for each instance n, can quickly explicitly produce all facet-defining inequalities and equations valid for the linear objective representation of the n-th instance. Here, "quickly" is in terms of the number of these facet-defining inequalities and equations and the number of variables for an instance. Given a linear description and a particular instance n, we can thus explicitly produce the matrix $A_n \in \mathbf{R}^{J_n \times (I_n \cup I'_n)}$ and right hand side $b_n \in \mathbf{R}^{J_n}$ so that the convex hull P_n of feasible solutions is given by the x variables (Section 2.5.2) of

$$P_n = \{(x, y) \in \mathbf{R}^{I_n \cup I'_n} \mid A_n \begin{bmatrix} x \\ y \end{bmatrix} \geq b_n\}. \qquad (2.12)$$

A *compact linear description* is a linear description where the polyhedron lies in a space of polynomial (in terms of n) dimension and is the intersection of polynomially many (in terms of n) closed halfspaces. We call optimization using a compact linear description *compact optimization.* Since linear programming is polynomially solvable, a compact linear description guarantees that this combinatorial optimization problem is in P. Many combinatorial optimization problems in P have compact linear descriptions. A notorious possible exception to this (as we have already indicated in Section 2.3.2) is the perfect matching problem.

2.5.2 Projection and Lifting

Consider the polyhedron P_n defined by (2.12). Recall that the y variable values have no effect on the objective function value for our problem instance. In fact, the only role these y variables have is to make it easier to form a linear description for our problem.

Suppose one wanted to extract the feasible polyhedron $\mathrm{Proj}_x(P_n)$ that results from just the x decision variables when the y variables are removed from the vector representation. The process of doing this is called projection, and the resulting polyhedron $\mathrm{Proj}_x(P_n)$ is called the *projection* of the polyhedron P_n onto the space of the x variables. This new polyhedron is in fact a projection in a geometric sense, satisfying

$$\mathrm{Proj}_x(P_n) = \{x \in \mathbf{R}^{I_n} \mid \exists y \in \mathbf{R}^{I'_n} \text{ s.t. } (x,y) \in P_n\}.$$

A fundamental property of projection is that

$$\begin{array}{ll} \text{minimize} & c \cdot x \\ \text{subject to} & (x,y) \in P_n, \end{array}$$

and

$$\begin{array}{ll} \text{minimize} & c \cdot x \\ \text{subject to} & x \in \mathrm{Proj}_x(P_n) \end{array}$$

have the same minimizers (when the minimizers of P_n are projected onto the space of the x variables).

But what are the facet-defining closed halfspaces and hyperplanes in a description of $\mathrm{Proj}_x(P_n)$? Let us break up A_n by columns into A_x and A_y, the submatrices that act on the x variables and the y variables respectively. By strong duality, one can see that the valid inequalities for $\mathrm{Proj}_x(P_n)$ (involving only the x variables) are precisely

$$\begin{array}{lcl} (\lambda \cdot A_n) \cdot x & \geq & \lambda \cdot b_n, \\ \text{such that} & & \\ \lambda \cdot A_y & = & 0, \\ \lambda & \geq & 0, \\ \sum_{i \in J_n} \lambda_i & = & 1, \end{array} \tag{2.13}$$

where this last sum is an arbitrary scaling of each inequality by a positive multiple.

THEOREM 2.50 *The facet-defining inequalities and equations of the polyhedron* $\mathrm{Proj}_x(P_n)$ *correspond exactly with the extreme points of (2.13).*

The concept opposite to projection is that of *lifting*, where one adds more variables, inter-relating them to the original variables, to form a lifted linear description in a higher dimensional space. As already seen, lifting is useful in producing non-obvious compact linear descriptions for combinatorial optimization problems (when the obvious linear descriptions are exponential in terms of n).

We describe two examples of compact linear programs possible only thanks to the introduction of new variables and constraints.

Example: Parity polytope [43]. Let $PP(n)$ denote the convex hull of zero-one vectors over $\mathbf{R}^n$ with an odd number of ones, that is, $PP(n) = \mathrm{conv}\{x \in \{0,1\}^n \mid \sum_i x_i \text{ is odd}\}$. This polytope has exponentially many facets [18], but by introducing new variables the number of constraints can be reduced significantly. The idea is to write $x \in PP(n)$ as a convex combination $\sum_{k \text{ odd}} \lambda_k y^k$, where y^k is a convex combination of extreme points of $PP(n)$ with k ones. We represent $\lambda_k y^k$ by the variable z^k in the linear program, obtaining

$$\begin{array}{lrcll}
\text{minimize} & c \cdot x & & & \\
\text{subject to} & & & & \\
& \sum_{k \text{ odd}} \lambda_k & = & 1 & \\
& \sum_{k \text{ odd}} z_i^k & = & x_i & \forall i \\
& \sum_i z_i^k & = & k\lambda_k & \text{for odd } k \\
& 0 & \leq & z_i^k \leq \lambda_k & \forall i, k.
\end{array} \tag{2.14}$$

In fact, a similar formulation exists for any *symmetric 0-1 function* $f : \{0,1\}^n \to \{0,1\}$. (A function f is symmetric, if its value doesn't change when its arguments are reordered.) In the 0-1 case, the value of a symmetric function depends only on the number of non-zero coordinates in the argument. Thus in the formulation (2.14), whenever there is a reference to the set of odd integers, it should be replaced with the set I such that $f(x) = 1$ iff the number of nonzero coordinates of x belongs to I.

Example: Arborescences in directed graphs. Given a vertex r of a directed graph $G = (V, E)$, an *arborescence rooted at* r is a subgraph of G that contains a directed path from r to v for each $v \in V$, but no (undirected) cycles. If we represent arborescences in G by their incidence vectors, then the convex hull of all such incidence vectors is the set of solutions to

$$\begin{array}{lcll} x(\delta^+(S)) & \geq & 1 & \forall S \subset V \text{ such that } r \in S \\ x(E) & = & |V|-1 & \\ x & \geq & 0, & \end{array} \tag{2.15}$$

where $\delta^+(S)$ denotes the set of directed edges of G leaving S [10]. The constraints can be intuitively be thought of as requiring that every cut separating a vertex of G from the root r is crossed by at least one edge. This idea is the basis of an extended formulation. For each v, we require that the solution support a path from r to v. This path will be represented by a flow of value 1 from r to v.

$$\begin{array}{lrcll} \text{minimize} & & c \cdot x & & \\ \text{subject to} & & & & \\ & x_e & \geq & y^t_e & \forall e \in E,\ \forall t \in V \setminus \{r\} \\ & y^t(\delta^-(v)) & = & y^t(\delta^+(v)) & \forall t, v \in V \setminus \{r\},\ v \neq t \\ & y^t(\delta^-(t)) & = & 1 & \forall t \in V \setminus \{r\} \\ & y^t(\delta^-(r)) & = & 0 & \forall t \in V \setminus \{r\} \\ & y^t(\delta^+(r)) & = & 1 & \forall t \in V \setminus \{r\} \\ & x(E) & = & |V|-1 & \\ & y & \geq & 0 & \end{array} \tag{2.16}$$

Now by the max-flow-min-cut theorem [11], each of the cuts in the constraints of LP (2.15) has value at least 1 iff for every $v \neq r$, there is an r-v flow of value at least 1. In other words, the projection of LP (2.16) to the space spanned by the x-variables is exactly the r-arborescence polytope defined by LP (2.15). The number of constraints in the flow-based formulation is polynomial in the size of G. See Pulleyblank [33] for a discussion and references.

2.5.3 Compact Separation

Consider an optimization problem that can be phrased as

$$\begin{array}{ll} \text{minimize} & c \cdot x \\ \text{subject to} & \\ & x \in P^{\mathcal{A}}_n (\subset \mathbf{R}^{I_n}), \end{array} \tag{2.17}$$

such as we saw earlier (Definition 2.34) in Section 2.3.2. Here, n is an integer parameter indicating the size of the instance, and $P^{\mathcal{A}}_n$ is a polyhedron whose variables are indexed by I_n and constraints come from a class $\mathcal{A}$ of valid inequalities. Unfortunately, the number of inequalities describing $P^{\mathcal{A}}_n$ will usually grow exponentially with n. In this case, directly solving this problem through linear programming takes exponential time. But, as we saw in the last two sections, a compact linear description may still be possible. That is, if we introduce additional variables y, we are sometimes able to express the polyhedron $P_n^{\mathcal{A}}$ as:

$$P_n^{\mathcal{A}} = \text{Proj}_x\{(x,y) \in \mathbf{R}^{I_n \cup I'_n} \mid A_n \cdot \begin{bmatrix} x \\ y \end{bmatrix} \geq b_n\}, \tag{2.18}$$

where the total number of variables and constraints grows polynomially with respect to n, as was done in (2.12). This gives a polynomial-time solution to our problem using linear programming, showing that this problem is in $\mathcal{P}$.

Recall the separation problem defined in Section 2.3.2 for our problem (2.17). In this section, we learned that (2.17) is in $\mathcal{P}$ if and only if the separation problem for the class $\mathcal{A}$ of inequalities is polynomially solvable. This raises the following natural question:

QUESTION 2.51 *Can we use the polynomial solvability of the separation problem to construct a compact linear description for our problem?*

A partial answer to this question was given by Martin [26] and by Carr and Lancia [5].

ANSWER 2.52 *If (and only if) the separation problem can be solved by compact optimization then one can construct a compact linear description of our problem from a compact linear description of the separation problem.*

In other words, if we can phrase the separation algorithm as a compact linear program, then we can use this to optimize compactly.

When one can solve the separation problem by compact optimization, this is called *compact separation.* We first argue that compact separation is a plausible case. Consider (2.17) again. Denote by Q_n the set of points $(a, a_0) \in \mathbf{R}^{I_n} \times \mathbf{R}$ corresponding to valid inequalities $a \cdot x \geq a_0$ of $P_n^{\mathcal{A}}$ (with (a, a_0) scaled so that $-1 \leq (a, a_0) \leq 1$). It follows from the theory behind projection (described in Section 2.5.2) and duality that Q_n is a (bounded) polyhedron!

Hence, the separation problem for the class $\mathcal{A}$ of inequalities can be phrased as a linear program of the form

$$\begin{array}{ll} \text{minimize} & x^* \cdot a - a_0 \\ \text{subject to} & \\ & (a, a_0) \in Q_n, \end{array} \tag{2.19}$$

where, x^* is the fractional point we wish to separate from $P_n^{\mathcal{A}}$. There is a violated inequality if and only if the minimum objective function value is less than 0. In this case, if the minimizer is (a^*, a_0^*), then

$$a^* \cdot x \geq a_0^*$$

is a valid inequality for $P_n^{\mathcal{A}}$ that is violated by x^*.

Unfortunately, the polyhedron Q_n often has exponentially many constraints with respect to n. But it may be possible to obtain compact separation by examining either Q_n or the workings of a polynomial separation algorithm. We thus strive to obtain

$$Q_n = \mathrm{Proj}_{(a,a_0)}\{(a, a_0, w) \in \mathbf{R}^{I_n} \times \mathbf{R} \times \mathbf{R}^{J_n} \mid B_n\,[a\,a_0\,w]^{\tau} \geq d_n\}, \quad (2.20)$$

where there is a polynomial number of variables and constraints with respect to n.

We now state the main theorem of this section.

Theorem 2.53 *[26, 5] Compact optimization is possible for $P_n^{\mathcal{A}}$ if (and only if) compact separation is possible for the class $\mathcal{A}$ of valid inequalities of $P_n^{\mathcal{A}}$.*

Proof: We only give the "if" direction here. Let Q_n be the polyhedron of valid inequalities $a \cdot x \geq a_0$ of $P_n^{\mathcal{A}}$. Assuming compact separation is possible, let (2.21) be the LP that achieves this, with a polynomial number of variables and constraints with respect to n.

$$\begin{array}{ll} \text{minimize} & x^* \cdot a - a_0 \\ \text{subject to} & \\ & B_n \begin{bmatrix} a \\ a_0 \\ w \end{bmatrix} \geq d_n \end{array} \quad (2.21)$$

Here, x^* violates an inequality in $\mathcal{A}$ if and only if the minimum of (2.21) is less than 0. Hence, x^* satisfies all such inequalities iff the minimum of (2.21) is at least 0. Partition B_n by columns into $[B_n^a B_n^{a_0} B_n^w]$. Consider the dual linear program:

$$\begin{array}{ll} \text{maximize} & y \cdot d_n \\ \text{subject to} & \\ & y \cdot B_n^a = x^* \\ & y \cdot B_n^{a_0} = -1 \\ & y \cdot B_n^w = 0 \\ & y \geq 0. \end{array} \quad (2.22)$$

By strong duality, we now have that x^* satisfies all such inequalities iff the maximum of (2.22) is at least 0. When we remove the star (!) from x^*, we see that x satisfies all the inequalities of $\mathcal{A}$ iff the linear system

$$\begin{array}{l} y \cdot d_n \geq 0 \\ y \cdot B_n^a = x \\ y \cdot B_n^{a_0} = -1 \\ y \cdot B_n^w = 0 \\ y \geq 0 \end{array} \quad (2.23)$$

has a feasible solution. Thus, $P_n^{\mathcal{A}}$ is given by

$$P_n^{\mathcal{A}} = \mathrm{Proj}_x\{(x,y) \in \mathbf{R}^{I_n \cup J_n} \mid (x,y) \text{ satisfies (2.23)}\},$$

which is of the form (2.20). Hence, compact optimization of (2.17) is given by

$$\begin{array}{ll} \text{minimize} & c \cdot x \\ \text{subject to} & \\ & (x,y) \text{ satisfies (2.23)}, \end{array} \tag{2.24}$$

which concludes our proof. □

2.6 Integer Polyhedra, TDI Systems and Dynamic Programming

In this section we are concerned with problems formulated in terms of integer programs. In order to solve such a problem in polynomial time we may apply a linear programming algorithm to the linear relaxation of the integer program. This approach sometimes works because polyhedra P associated with several combinatorial optimization problems are integer polyhedra (have only integer extreme points). In such a case, any polynomial-time algorithm for linear programming that outputs an extreme point solution implies a polynomial-time algorithm for the problem. How can we tell whether a given polyhedron is integer? It is easy to see that the problem of determining whether a given polyhedron is integer is in Co-NP, but it is not known if it is in NP. However, in order to optimize a linear objective function over the polyhedron, we do not need to know that it is integer. Since the known polynomial-time algorithms for linear programming can be used to find an optimal extreme point, it follows that an integer programming problem can be solved in polynomial time when the polyhedron of its linear relaxation is integer.

There do exist several typical classes of integer polyhedra and they come with characterizations that allow one to prove their integrality, or at least, that an integer optimum can be found efficiently.

2.6.1 Total Unimodularity

An important special case of integer polyhedra is the class defined by *totally unimodular* matrices. In this case the matrix of the polyhedron nearly guarantees all by itself that the polyhedron is integer.

Definition 2.54 *A matrix A is totally unimodular, if for each integer vector b, the polyhedron $\{x \mid x \geq 0, Ax \leq b\}$ has only integer vertices.*

It can easily be seen that all the entries of a totally unimodular matrix belong to $\{0, -1, 1\}$.

There exist several other conditions that can be used as definitions of total unimodularity. We only list a few.

THEOREM 2.55 *For a matrix A with entries from $\{0, -1, 1\}$, the following conditions are equivalent:*

1 *For each integral vector b, the polyhedron $\{x \mid x \geq 0, Ax \leq b\}$ has only integral vertices.*

2 *The determinant of each square submatrix of A belongs to $\{0, -1, 1\}$.*

3 *Each subset of columns (rows) of A can be split into two parts A_1 and A_2 so that the difference of the sum of columns (rows) in A_1 and the sum of columns (rows) in A_2 has all entries in $\{0, -1, 1\}$.*

Total unimodularity is preserved under typical matrix operations such as transposition, multiplication of a row or column by -1, interchange or duplication of rows or columns, or appending a basis vector.

The typical example of a totally unimodular matrix is the vertex-edge incidence matrix of a bipartite graph. To prove the total unimodularity, split the rows (corresponding to vertices) into two sets according to a bipartition of the graph. Then for each column (edge), the difference between the sums of the rows in the two subsets is 0 (one endpoint in each set), and the third condition of Theorem 2.55 is satisfied. It is not difficult to prove the converse, either, namely that the vertex-edge incidence matrix of a nonbipartite graph is not totally unimodular.

2.6.2 Total Dual Integrality

As long as the right-hand side vector b is integral, a totally unimodular matrix A will define an integral polyhedron $Ax \leq b$. In order to generalize the class of integral polyhedra we consider, we may include the right-hand side in the specification of an instance. The system $Ax \leq b$ (with rational A and b) is *totally dual integral*, if it is feasible, and for every integral objective function c, the dual problem

$$\min\{y \cdot b \mid y \geq 0, \, yA = c\} \tag{2.25}$$

has an integer optimal solution y^*, when the dual is feasible.

THEOREM 2.56 *If $Ax \leq b$ is totally dual integral and b is integer, then $Ax \leq b$ defines an integer polyhedron.*

From Theorem 2.56 it follows that one can optimize efficiently over polyhedra defined by totally dual integral systems. However, a stronger result holds and this is possible even without requiring the integrality of b [35].

THEOREM 2.57 *If $Ax \leq b$ is a totally dual integral system and c an integral objective function, then it is possible to find an integral solution to the dual problem* (2.25) *in polynomial time.*

Totally dual integral systems do not behave as nicely as totally unimodular matrices when subjected to transformations. In fact, *any* rational system $Ax \leq b$ can be made totally dual integral by dividing both sides of the equation with an appropriately chosen (potentially large) integer k [35]. (Of course, if b was integral before, it is unlikely to remain integral upon dividing by such a k.) Another way of stating this property of rational systems is that regardless of the objective function (as long as it is integral), there is an integer k such that in the optimum solution to the dual problem (2.25) each entry is an integral multiple of $1/k$.

However, there is a simple and useful operation that does preserve total dual integrality.

THEOREM 2.58 *If $Ax \leq b$ is a totally dual integral system, then so is the system obtained from it by making any inequality an equality.*

2.6.3 Dynamic Programming and Integrality

In this section we describe a very general source of totally dual integral systems, integer polytopes and compact linear formulations of combinatorial optimization problems. The main source for this material is the paper of Martin et al. [27].

2.6.3.1 Discrete Dynamic Programming.

A combinatorial minimization problem (maximization can be handled analogously) can be solved using dynamic programming, if for each instance of the problem there exists a family of instances S (called the family of *subproblems*, or *states*) and a decomposition relation $D \subseteq 2^S \times S$ such that the following two conditions hold:

1 (*Optimality principle.*) For each $(A, I) \in D$, an optimal solution to I can be constructed from any collection that includes an optimal solution to every instance in A.

2 (*Acyclicity.*) There is an ordering $\sigma : S \to [m]$ (where $m = |S|$) compatible with the decomposition relation: given $I' \in S$, $A \subseteq S$ such that $(A, I') \in D$, the condition $\sigma(I) < \sigma(I')$ holds for all $I \in A$.

The dynamic programming algorithm solves the given problem I_m by solving recursively each of the subproblems I such that $I \in A$ for some

$A \subseteq S$ that satisfies $(A, I_m) \in D$. Then the algorithm builds an optimal solution to I_m from optimal solutions to the collection of subproblems A' that minimizes (over the selection of A) the cost $\sum_{I' \in A} c(I') + c(A, I_m)$. Here, $c(I)$ is the cost of an optimal solution to I and $c(A, I_m)$ denotes the cost of building a solution to I_m from the solutions of the subproblems in A.

A simple example of this paradigm in action is provided by the *shortest path problem*: given a graph $G = (V, E)$ with a nonnegative cost (or length) function $c : E \to R_+$ defined on the set of edges, and given two vertices s and t of G, find an s-t path of minimum cost. The collection of subproblems consists of pairs (v, i), where $v \in V$ and $i \in [n]$ $(n = |V|)$. Solutions to the subproblem (v, i) are all s-v paths no more than i edges long. Then for every pair of vertices u and v such that $uv \in E$ and for every $i \leq n$, there will be an element $(\{(u, i)\}, (v, i{+}1))$, in the relation D such that the optimal cost for $(v, i{+}1)$ is equal to the optimal cost for the pair (u, i) plus c_{uv}. Consequently, the associated cost $c(\{(u, i)\}, (v, i{+}1))$ equals the cost c_{uv} of the edge uv.

The shortest path problem possesses a special property, namely the relation D again defines a graph on the set of subproblems, and in fact, solving the dynamic programming formulation is equivalent to solving a shortest path problem on this *decision graph*. This has been used by several authors [21, 17, 9]. We follow the paper of Martin et al. [27] in describing a more general *hypergraph model*. In this context, the elements of the relation D will be called *decisions* or *hyperarcs*.

Another example is provided by the parity polytope. We gave a linear description of the parity polytope in Section 2.5.1, following Yannakakis [43]. Here is an alternative description motivated by dynamic programming.

Let $P_n = \{x \in \{0,1\}^n \mid \sum_i x_i \text{ is odd}\}$, as in Section 2.5.1. We introduce additional variables p_i, $i = 1, \ldots, n$. The idea is to have $p_{i'}$ express the parity of the partial sum $\sum_{i \leq i'} x_i$. The first constraint is

$$p_1 = x_1. \tag{2.26}$$

For each $i \in \{2, \ldots n-1\}$, there will be four constraints:

$$\begin{array}{rcl} p_{i+1} & \leq & p_i + x_{i+1} \\ p_{i+1} & \leq & 2 - (p_i + x_{i+1}) \\ p_{i+1} & \geq & p_i - x_{i+1} \\ p_{i+1} & \geq & x_{i+1} - p_i. \end{array} \tag{2.27}$$

Finally, to ensure the sum of all components of x is odd, the constraint

$$p_n = 1 \tag{2.28}$$

is needed. Upper and lower bounds complete the formulation:

$$0 \leq x, p \leq 1. \tag{2.29}$$

It is easy to interpret this formulation in terms of the dynamic programming paradigm and infer the integrality of the polyhedron described by the constraints. Note the small numbers of $2n$ variables and $6n-6$ constraints used, as opposed to roughly $n^2/2$ variables and constraints each in the formulation (2.14). However, this alternative formulation is not symmetric with respect to the variables. In fact, the asymmetry seems to have helped make this a smaller formulation. Yannakakis proves [43] that no symmetric linear programming formulation of polynomial size exists for the matching or traveling salesman polytopes and conjectures that asymmetry will not help there. Is he right?

2.6.3.2 Polyhedral formulations. To construct a polyhedral description of the general discrete dynamic programming problem, let S denote the set of states, ordered by $\sigma : S \to [m]$, and let D be the decomposition relation.

For each $(A, I) \in D$, let there be a binary variable $z_{A,I}$ whose value indicates whether the hyperarc (A, I) is a part of the (hyper)path to the optimum. Let $I_m \in S$ be the subproblem corresponding to the original instance. It must be solved by the dynamic program, hence the constraint

$$\sum_{\{A \mid (A,I_m)\in D\}} z_{A,I_m} = 1. \tag{2.30}$$

If some subproblem is used in the solution of another, then it must be solved too. This leads us to *flow conservation* constraints

$$\sum_{\{A \mid (A,I)\in D\}} z_{A,I} = \sum_{\{(A,J)\in D \mid I\in A\}} z_{A,J}, \quad \text{for all } I \neq I_m. \tag{2.31}$$

The polyhedral description of the dynamic programming problem is completed by the nonnegativity constraints

$$z_{A,I} \geq 0, \quad \text{for all } (A, I) \in D. \tag{2.32}$$

Now the complete formulation is

$$\begin{array}{ll} \text{minimize} & \sum_{(A,I)\in D} c(A,I) z_{A,I} \\ \text{subject to} & \\ & \text{(2.30)–(2.32).} \end{array} \tag{2.33}$$

To relate this to the dynamic programming algorithm, consider the dual formulation

$$\begin{array}{lll} \text{maximize} & v & \\ \text{subject to} & & \\ v - \sum_{J\in A} u_J & \leq c(A,I) & \text{for all } (A,I)\in D \\ & & \quad \text{such that } I = I_m \\ u_I - \sum_{J\in A} u_J & \leq c(A,I) & \text{for all } (A,I)\in D, \\ & & \quad \text{such that } I \neq I_m. \end{array} \tag{2.34}$$

Dual construction.
The dynamic programming procedure directly finds an optimal solution to the dual LP (2.34) by setting the dual variables u_I corresponding to the constraints (2.30)–(2.31) in the order defined by σ: once the value u^*_I has been defined for each I such that $\sigma(I) < i$, the procedure sets

$$u^*_{I_i} = \min_{A \mid (A,I_i)\in D} \{c(A,I_i) + \sum_{J\in A} u^*_J\},$$

and continues until reaching I_m.

Given u^*, the computation of the dynamic program can be reversed to recursively construct a primal solution z^* as follows.

Primal Construction:

(1) Initialize $z^* = 0$, $t = 0$, $A_0 = \{I_m\}$ (A_t is the set of subproblems whose solution is required at depth t).

(2) While $A_t \neq \emptyset$, do steps **(3)**–**(5)**.

(3) Pick a subproblem $I \in A_t$, remove it from A_t.

(4) Select a set A_{t+1} such that $u^*_I = c(A_{t+1}, I) + \sum_{J\in A_{t+1}} u^*_J$.

(5) If $t > 0$, set $t = t - 1$ and return to step **(2)** (backtrack).

THEOREM 2.59 *Consider u^* and z^* produced by the two constructions above. If the acyclic condition defined in Section 2.6.3.1 holds, then z^* and u^* are optimal solutions to* (2.33) *and* (2.34), *respectively, and z^* is an integer solution.*

Note that z^* is integer by the primal construction. Hence it is enough to prove that z^* is feasible for the primal (2.33), that u^* is feasible for the dual (2.34), and that the complementary slackness conditions (Theorem 2.44) are satisfied. We leave the details to the reader, but note that in the critical step **(4)** in the primal construction, the set A_{t+1} can always be found because of the way u^*_I has been defined during the dual construction. This same observation helps in establishing complementary slackness: a primal variable is only increased from its original zero value if the corresponding dual constraint is tight.

Theorem 2.59 implies also total dual integrality of the system defining the LP (2.33). However, it is possible that z^* contains some entries of value greater than 1, and so does not correspond to a dynamic programming solution. If constraints are added to LP (2.33), requiring that each entry of z is at most 1, then the LP may not have an integer optimum solution. Thus we must impose additional structure on the dynamic programming problem to ensure that zero-one solutions are extreme points. This structure can be expressed by associating with each state $I \in S$ a nonempty subset R_I of a finite *reference set* R. The partial order defined on the family of reference subsets induced by set inclusion must be compatible with the relation D:

$$R_I \subseteq R_J,$$

for all I, J such that there exists an A such that $I \in A$ and $(A, J) \in D$. The reference subsets must also satisfy the following condition:

$$R_I \cap R_{I'} = \emptyset \text{ for all } (A, J) \in D \text{ such that } I, I' \in A.$$

THEOREM 2.60 *For a discrete dynamic programming problem formulated over a set of states S with the decision variables $z_{A,I}$ for $(A, I) \in D$, where there exists a reference set R satisfying the conditions described above, the LP* (2.33) *describes the convex hull of zero-one solutions.*

A few final observations:

1. The resulting formulations are compact.
2. The resulting formulations are typically not symmetric.
3. Usually related to the second statement is the fact that the formulations are typically not over the most natural variable space. A good example is the k-median problem on a tree. There exist dynamic programming algorithms for this problem [39], but a polyhedral formulation in terms of natural vertex decision variables is not known [8]. Theoretically, it is possible to express the formulation in terms of natural variables by using a linear transformation and then a projection. However, this is often not practical.

2.7 Approximation Algorithms, Integrality Gaps and Polyhedral Embeddings

Many combinatorial optimization problems are NP-hard. In such a case, we cannot hope to have a complete polyhedral description that would be solvable in polynomial time and we must resort to relaxations (Section 2.3.1). In many cases, an optimal solution to the relaxation of a polytope can be used as a starting point in the derivation of a point belonging to the polytope. In a combinatorial optimization setting, the motivation is to try and use the optimal LP solution to find a good—though not necessarily optimal—solution to the problem in question. Such a procedure is usually called *LP rounding* because from an optimal (fractional) solution to the LP relaxation it produces an integral solution to the problem.

Consider a combinatorial minimization problem. Denote by $z^*(I)$ the cost of an optimal solution to the instance I. An algorithm A that produces a feasible solution of cost $z_A(I)$ such that

$$\frac{z_A(I)}{z^*(I)} \leq \alpha$$

for all instances I is called an *α-approximation algorithm* for the given problem. In other words, an approximation algorithm provides a guarantee on the quality of solutions it produces.

An approximation algorithm based on LP rounding usually (but not always [23]) implies a bound on the maximum ratio between the cost z^* of an optimal solution to the problem and the cost z^*_{LP} of an optimal solution to the relaxation. Given an LP relaxation of a combinatorial optimization problem, the value

$$\sup_I \frac{z^*(I)}{z^*_{LP}(I)}$$

is called the *integrality gap* of the relaxation. The integrality gap will be a constant in most of our examples, but in many cases it is a function of the instance size.

2.7.1 Integrality Gaps and Convex Combinations

A standard example of the integrality gap is provided by the *vertex cover* problem. Given a graph $G = (V, E)$ with a cost function $c : V \to \mathbf{R}_+$, a set of $U \subseteq V$ is a vertex cover of G, if for every edge $e \in E$, at least one endpoint of e belongs to U. The cost of the vertex cover U is defined as $c(U) = \sum_{v \in U} c_v$. Any minimum-cost vertex cover of G is an optimal solution to the vertex cover problem on G. The vertex cover problem is NP-hard and so we do not know a polyhedral description, but it turns out that the relaxation (2.35) is useful in designing an approximation algorithm.

$$\begin{array}{lrcll} \text{minimize} & \sum_{v\in V} c_v x_v & & & \\ \text{subject to} & & & & \\ & x_u + x_v & \geq & 1 & \forall uv \in E \\ & x & \geq & 0 & \end{array} \tag{2.35}$$

Let x^* be an optimal solution to the relaxation (2.35) and consider $2x^*$. Since $x_u + x_v \geq 1$ for every $uv \in E$, at least one of x_u and x_v is at least $1/2$. For each edge $e \in E$, choose an endpoint v_e of e such that $v_e \geq 1/2$. Let $U = \{v_e \mid e \in E\}$. Then define a 0-1 vector x^U by

$$x_v^U = \begin{cases} 1, & v \in U \\ 0, & v \notin U. \end{cases}$$

Now x^U is the incidence vector of a vertex cover, and since

$$x^U \leq \lfloor 2x^* \rfloor \leq 2x^*,$$

it follows that $2x^*$ dominates a convex combination (in this case a trivial combination, consisting only of x^U) of integral solutions. The cost of the solution x^U then satisfies

$$c \cdot x^U \leq 2c \cdot x^*.$$

One consequence of this argument is that the integrality gap of the LP (2.35) is at most 2. Another is that the algorithm that computes and outputs x^U is a 2-approximation algorithm.

How does one find a rounding procedure? The vertex cover case is simple enough that the rounding can be guessed directly and the guarantee proven easily, but in general the question of whether there is a rounding procedure may be very difficult. The procedure described for vertex cover can be generalized to all relaxations with a nonnegative cost function. In situations when there is more than one integral solution with a nonzero coefficient in the convex combination, to guarantee a bound on the integrality gap we use the additional observation that the cheapest among these solutions costs no more than the convex combination itself. In fact, the following can be shown [3, 6]:

THEOREM 2.61 *For a minimization problem with a nonnegative cost function, the integrality gap is bounded above by α if (and only if) the corresponding multiple αx^* of any extreme-point solution to the relaxation dominates (component-wise) a convex combination of integral solutions.*

Thus there is no loss of generality in considering only the rounding procedures of this form. In Sections 2.7.2 and 2.7.5 we show two more examples of this principle: those of metric STSP and edge-dominating set problems.

Integrality gaps of many combinatorial optimization problems have been studied in the literature. A notable example is the traveling salesman problem where the cost satisfies the triangle inequality (that is, defines a metric on the underlying graph), in both undirected and directed versions. For the undirected case (that is, metric STSP), the integrality gap is between 4/3 and 3/2 [13, 19]. In the directed (asymmetric) case the best currently known bounds are 4/3 and $\log n$ [6]. For the metric STSP, it is conjectured and widely believed that the gap is 4/3. Many of the integrality gap results have been motivated by the search for approximation algorithms [41].

2.7.2 An STSP Integrality Gap Bound

Closely related to Hamilton cycles in combinatorial optimization are *spanning Eulerian multigraphs.* An *Eulerian (multi)graph* is a graph $T = (V, E(T))$ where every vertex in V has even degree (and parallel edges are allowed in $E(T)$). Such a graph is *spanning* if every vertex in V is an endpoint of some edge of $E(T)$ (and thus at least 2 such edges). If the cost function satisfies the triangle inequality, then any such multigraph can be converted to a Hamilton cycle of no greater cost by taking a so-called *Euler tour* (visiting each edge in $E(T)$ exactly once), and *shortcutting*, creating a new edge to bypass each vertex $v \in V$ that has already been visited, thus replacing the two edges entering and leaving v on this subsequent occasion. This relationship between Eulerian graphs and Hamilton cycles yields the following result:

Theorem 2.62 *The integrality gap for the metric STSP is at most α if (and only if) for each extreme point x^* of the subtour relaxation, αx^* can be expressed as a convex combination of spanning Eulerian multigraphs.*

Consider the extreme point in Figure 2.7.2. By applying the only if part of Theorem 2.62, one can show that the integrality gap of metric STSP is at least 4/3 (an appropriate choice of a cost function will obtain this result as well). Wolsey [42], and later Shmoys and Williamson [38], show this gap to be at most 3/2. We now outline an elegant argument for this in the spirit of Theorem 2.62. Let $n = |V|$. Recall that SEP denotes the subtour polytope (Section 2.3.1).

Figure 2.1. Diagram of an STSP extreme point. Each of the three horizontal paths has k edges. The solid lines represent edges with value 1, dashed with value $\frac{1}{2}$.

LEMMA 2.63 *Given any extreme point x^* of SEP, $\frac{n-1}{n}x^*$ can be expressed as a convex combination of (incidence vectors of)* spanning trees *(i.e. acyclic, connected spanning graphs).*

Proof: Consider the spanning tree polytope [25]. It can be shown that $\frac{n-1}{n}x^*$ satisfies all the constraints. $\square$

LEMMA 2.64 *Given any extreme point x^* of SEP and any $T \subset V$ with $|T|$ even, $\frac{1}{2}x^*$ can be expressed as a convex combination of* T-joins *(multigraphs whose odd degree vertices are precisely those in T, see Exercise 2.36).*

Proof: Consider the T-join polyhedron [12]. It can be shown that $\frac{1}{2}x^*$ satisfies all the constraints. $\square$

THEOREM 2.65 *Given any extreme point x^* of SEP, $(\frac{3}{2} - \frac{1}{n})x^*$ can be expressed as a convex combination of spanning Eulerian multigraphs.*

Proof: Express $\frac{n-1}{n}x^*$ as a convex combination

$$\sum_i \lambda_i \chi^{E(T_i)} \tag{2.36}$$

of spanning trees, using Lemma 2.63. Consider one such tree T_i and define T_i^{odd} to be the odd degree nodes of this tree. If an incidence vector of a T_i^{odd}-join is added to $\chi^{E(T_i)}$, we obtain an incidence vector of an Eulerian graph. Hence, using Lemma 2.64, we can express $\chi^{E(T_i)} + \frac{1}{2}x^*$ as a convex combination

$$\sum_j \mu_{i,j} \chi^{E(K_{i,j})} \tag{2.37}$$

of spanning Eulerian multigraphs. But, $(\frac{n-1}{n} + \frac{1}{2})x^*$ can be rewritten using (2.36) as a convex combination

$$\sum_i \lambda_i (\chi^{E(T_i)} + \frac{1}{2}x^*).$$

Therefore, $(\frac{n-1}{n} + \frac{1}{2})x^*$ can be expressed as the convex combination

$$\sum_{i,j} \lambda_i \mu_{i,j} \chi^{E(K_{i,j})}$$

of spanning Eulerian multigraphs by using (2.37), as required. □

COROLLARY 2.66 *The integrality gap of metric STSP is at most* 3/2*.*

2.7.3 Half-integrality

The linear relaxation (2.35) of vertex cover in fact satisfies a strong, very interesting property.

THEOREM 2.67 *[30] Every extreme point of the polyhedron defined by the LP* (2.35) *has coordinates that all belong to* $\{0, 1/2, 1\}$*, that is, the polyhedron is half-integral.*

Proof: Let x^* be an optimal solution to the LP (2.35) and define

$$\begin{array}{lcl} U_{-1} & = & \{v \in V \mid 0 < x^*_v < 1/2\}, \\ U_1 & = & \{v \in V \mid 1/2 < x^*_v < 1\}. \end{array}$$

Then modify the values of x^*_v for those v contained in either U_{-1} or U_1: define $y^1_v = x^*_v + k\epsilon$ and $y^2_v = x^*_v - k\epsilon$ if $v \in U_k$, for $k \in \{-1, 1\}$. Define $y^1_v = y^2_v = x_v$ for $v \notin U_{-1} \cup U_1$. Then $x^* = (y^1 + y^2)/2$. If either U_{-1} or U_1 is nonempty, ϵ may be chosen small enough that both y^1 and y^2 are feasible, and thus $U_{-1} = U_1 = \emptyset$ for an extreme point x^*. □

Half-integrality is common in combinatorial optimization. We give another example in Section 2.7.5 and refer the reader to Hochbaum's survey [16] for additional results.

2.7.4 Rounding Reductions

Occasionally, approximation algorithms may be composed by reductions that preserve approximation guarantees. For example, suppose we can transform an instance I_A of problem A to an instance I_B of problem B so that there is a one-to-one correspondence between solutions of I_A and those of I_B such that if s_A is a solution to I_A and s_B the corresponding solution to I_B, their costs satisfy $\alpha c(s_A) \geq c(s_B)$. Then a β-approximation algorithm for B implies an $\alpha\beta$ approximation algorithm for A.

In Section 2.7.5 we present such a reduction that uses linear relaxations and rounding. The problems involved are the *weighted edge-dominating set* (EDS) and *edge-cover* (EC). The algorithm starts with an optimal solution of a relaxation of EDS and multiplies it by 2. The resulting point can be interpreted as a solution to a relaxation of EC, and after multiplication by 4/3 expressed (after reducing the values of some of the variables) as a convex combination of integral solutions.

Other similar results exist [2, 22]. The integrality gap or the approximation guarantee is sometimes the product of those achieved by the intermediate steps (Section 2.7.5) and sometimes the sum (Section 2.7.2). Even better results are sometimes possible with careful analysis [4].

2.7.5 Weighted edge-dominating sets

Given a graph $G = (V, E)$ with a cost function $c : E \to \mathbf{R}_+$ defined on the edge set, an *edge-dominating set* of G is any subset $F \subseteq E$ of edges such that for every $e \in E$, at least one edge adjacent to e is contained in F. The cost of an edge-dominating set is defined as the sum of the costs of its edges. An optimal solution to the edge-dominating set (EDS) instance defined by G is then a minimum-cost edge-dominating set.

For edges $e, f \in E$, write $e \sim f$ if e and f are adjacent. A simple linear relaxation for EDS is as follows.

$$\begin{array}{lrcll} \text{minimize} & \sum_{e\in E} c_e x_e & & & \\ \text{subject to} & & & & \\ & \sum_{f\sim e} x_f & \geq & 1 & \forall e \in E \\ & 0 \leq x & \leq & 1 & \end{array} \tag{2.38}$$

Let x^* be an optimal solution to (2.38). Then for each edge $uv \in E$,

$$\max\{2x^*(\delta(u)), 2x^*(\delta(v))\} \geq 1 + x^*_{uv}.$$

For each edge $e \in E$, let v_e be the endpoint that achieves the maximum above. Then let $U = \{v_e \mid e \in E\}$, and $y_e = \min\{2x^*_e, 1\}$ for all $e \in E$. The point y is now an element of the polyhedron

$$\begin{array}{rcll} 0 \leq y_e & \leq & 1 & \forall e \in E \\ y(\delta(v)) & \geq & 1 & \forall v \in U. \end{array} \tag{2.39}$$

An important observation is that any integral solution to (2.39) is also an edge-dominating set for G. We shall prove that an extreme point optimal solution to (2.39) can be efficiently rounded to an edge-dominating set while increasing the cost by a factor at most 4/3, implying an 8/3-approximation algorithm (and integrality gap) for EDS.

THEOREM 2.68 *Let y^* be an extreme point of the polyhedron* (2.39). *Then y^* is half-integral and the edges of value* 1/2 *induce disjoint odd cycles in G.*

The proof of Theorem 2.68 is longer, but based on the same ideas as that of Theorem 2.67 (half-integrality of vertex cover).

A final fact needed to justify the performance guarantee:

LEMMA 2.69 *Let C be an odd cycle of edges with value $1/2$. Let y_C denote the vector of fractional edge-values of C. Then $(4/3)y_C$ dominates a convex combination of edge-covers of C.*

Proof: Let $C = (V_C, E_C)$, where $V_C = \{v_1, \ldots, v_k\}$ and $E_C = \{v_1v_2, v_2v_3, \ldots, v_kv_1\}$. Define integral edge-covers $D_1, \ldots, D_k$ as follows. For each i, D_i will have $(k+1)/2$ edges. For $i < k$, let D_i contain e_i and e_{i+1}. Then add $(k-3)/2$ edges to D_i so that e_i and e_{i+1} are the only adjacent pair of edges in D_i. In D_k, the adjacent edges are e_k and e_1.

Let y^i denote the incidence vector of D_i. Then

$$y^0 = \frac{1}{k}(y^1 + y^2 + \cdots + y^k)$$

is a convex combination of edge-covers, and $y^0_i = (k+1)/(2k)$ for all i. Since $4/3 \geq (k+1)/k$ for all $k \geq 3$ (and the shortest odd cycle is a triangle), it follows that $(4/3)y_C \geq y^0$.

Note that, because the sum of coefficients in any convex combination is 1, it follows that at least one of the edge-covers $D_1, \ldots, D_k$ has cost no more than $4/3$ times the cost of y_C.

Algorithm:

(1) Solve the LP-relaxation (2.38). Let x^{LP} be an optimal solution.

(2) Let $U = \emptyset$. For each edge $e = uv \in E$, if $x^{LP}(\delta(u)) \geq 1/2$, add u to U, otherwise add v to U.

(3) Solve the relaxed edge-cover linear program, where the constraints $y(\delta(u)) \geq 1$ are included only for $u \in U$, and the objective function is $\min \sum_{e\in E} c_e y_e$. Find an optimal solution y^* which is an extreme point.

(4) Let $D = \{e \in E \mid y^*_e = 1\}$. Find the least cost edge-cover of each odd cycle in y^*, and add its edges to D.

(5) Output D.

Note that D is an edge-dominating set because the edge-covering constraints for U require at least one endpoint of each edge in E to be covered (dominated) by an edge in D.

THEOREM 2.70 *The edge-dominating set D that the above algorithm outputs has cost no more than $8/3$ times the cost of an optimal edge-dominating set.*

THEOREM 2.71 *For any extreme point x^{LP} of* (2.38)*,* $(8/3)x^{LP}$ *dominates a convex combination of edge-dominating sets. Thus the integrality gap of the LP* (2.38) *is at most* 8/3*.*

A more careful analysis shows that the integrality gap of the relaxation (2.38) is in fact at most 2.1. In fact, the integrality gap is *exactly* 2.1 [4]: consider the complete graph K_{5n} on $5n$ vertices and partition it into n disjoint copies of K_5, named $G_1, \ldots, G_n$. Let $c_e = 1$ for each edge e induced by some G_i, and $c_e = 11n$ for every other edge e. Now any integral edge-dominating set has cost at least $3n - 1$, but on the other hand assigning $x_e = 1/7$ to all the edges of G_i for each i results in a feasible solution of cost $10n/7$.

It is possible to strengthen the linear relaxation (2.38) by adding inequalities for *hypomatchable sets* [32]. The integrality gap of the resulting relaxation is 2, and an algorithm with approximation guarantee 2 follows as well. Further improvements would imply improvements also for vertex cover, which is an outstanding open problem.

As a final remark, we note that the algorithm described above, as well as many others based on the same paradigm, can actually be implemented without explicitly solving the initial LP relaxation, and instead based on (more efficient) algorithms for approximating the "intermediate problem" (vertex cover).

Acknowledgment

This chapter arose partially from notes for a course taught at the University of New Mexico. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. Department of Energy under Contract DE-AC04-94AL85000.

References

[1] E. Balas and M. Fischetti. Polyhedral theory for the asymmetric traveling salesman problem. In *The traveling salesman problem and its variations*, pages 117–168. Kluwer, 2002.

[2] D. Bienstock, M. Goemans, D. Simchi-Levi, and David Williamson. A note on the prize-collecting traveling salesman problem. *Mathematical Programming, Series A*, 59(3):413–420, 1993.

[3] S. Boyd and R. D. Carr. A new bound for the radio between the 2-matching problem and its linear relaxation. *Mathematical Programming, Series A*, 86(3):499–514, 1999.

[4] R. D. Carr, T. Fujito, G. Konjevod, and O. Parekh. A $2\frac{1}{10}$ approximation algorithm for a generalization of the weighted edge-dominating set problem. *Journal of Combinatorial Optimization*, 5:317–326, 2001.

[5] R. D. Carr and G. Lancia. Compact vs. exponential-size LP relaxations. *Operations Research Letters*, 30(1):57–65, 2002.

[6] R. D. Carr and S. Vempala. Towards a 4/3-approximation for the asymmetric traveling salesman problem. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 116–125, 2000.

[7] V. Chvátal. *Linear programming.* Freeman, 1983.

[8] S. de Vries, M. E. Posner, and R. Vohra. Polyhedral properties of the K-median problem on a tree. No 1367 in Discussion Papers, Northwestern University, Center for Mathematical Studies in Economics and Management Science, 42 pages, 2003.

[9] E. V. Denardo. *Dynamic programming: theory and practice.* Prentice-Hall, 1978.

[10] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967.

[11] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[12] A. M. H. Gerards. Matching. In *Network models*, volume 7 of *Handbooks in Operations Research*, pages 135–224. Elsevier, 1995.

[13] M. Goemans. Worst-case comparison of valid inequalities for the TSP. *Mathematical Programming*, 69:335–349, 1995.

[14] M. Grötschel and L. Lovász. Combinatorial optimization. In *Handbook of combinatorics*, volume 2, pages 1541–1597. Elsevier, 1995.

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

[16] D. S. Hochbaum. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*, pages 94–143. PWS, 1997.

[17] T. Ibaraki. Solvable classes of discrete dynamic programming. *Journal of Mathematical Analysis and Applications*, 43:642–693, 1973.

[18] R. G. Jeroslow. On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11:119–124, 1975.

[19] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In *Network models*, volume 7 of *Handbooks in Operations Research*, pages 225–329. Elsevier, 1995.

[20] V. Kaibel and M. Pfetsch. Some algorithmic problems in polytope theory. In *Algebra, Geometry and Software Systems*, pages 23–47. Springer Verlag, 2003. arXiv:math.CO/0202204v1.

[21] R. M. Karp and M. Held. Finite state processes and dynamic programming. *SIAM Journal of Applied Mathematics*, 15:693–718, 1967.

[22] J. Könemann, G. Konjevod, O. Parekh, and A. Sinha. Improved approximation algorithms for tour and tree covers. In *Proceedings of APPROX 2000*, volume 1913 of *Lecture Notes in Computer Science*, pages 184–193, 2000.

[23] G. Konjevod, R. Ravi, and A. Srinivasan. Approximation algorithms for the covering Steiner problem. *Random Structures & Algorithms*, 20:465–482, 2002.

[24] B. Korte and J. Vygen. *Combinatorial optimization. Theory and algorithms.* Springer Verlag, 2000.

[25] T. L. Magnanti and L. A. Wolsey. Optimal trees. In *Network models*, volume 7 of *Handbooks in Operations Research*, pages 503–615. Elsevier, 1995.

[26] R. K. Martin. Using separations algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10(3):119–128, 1991.

[27] R. K. Martin, R. Rardin, and B. A. Campbell. Polyhedral characterizations of discrete dynamic programming. *Operations Research*, 38(1):127–138, 1990.

[28] D. Naddef. Polyhedral theory and branch-and-cut algorithms for the symmetric TSP. In *The traveling salesman problem and its variations*, pages 29–116. Kluwer, 2002.

[29] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.

[30] G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6:48–61, 1974.

[31] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[32] O. Parekh. Edge-dominating and hypomatchable sets. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 287–291, 2002.

[33] W. R. Pulleyblank. Polyhedral combinatorics. In *Optimization*, volume 1 of *Handbooks in Operations Research*, pages 371–446. Elsevier, 1989.

[34] W. R. Pulleyblank. Matchings and extensions. In *Handbook of combinatorics*, volume 1, pages 179–232. Elsevier, 1995.

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

[36] A. Schrijver. Polyhedral combinatorics. In *Handbook of combinatorics*, volume 2, pages 1649–1704. Elsevier, 1995.

[37] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer Verlag, 2003.

[38] D. B. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. *Information Processing Letters*, 35:281–285, 1990.

[39] A. Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19(2):59–64, 1996.

[40] T. Terlaky and S. Zhang. Pivot rules for linear programming: a survey on recent theoretical developments. *Annals of Operations Research*, 46:203–233, 1993.

[41] V. V. Vazirani. *Approximation algorithms.* Springer Verlag, 2001.

[42] L. A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Study*, 13:121–134, 1980.

[43] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991.

[44] G. Ziegler. *Lectures on Polytopes.* Springer, 1995.