
Annals of Operations Research 140, 125–161, 2005

Projection, Lifting and Extended Formulation in Integer and Combinatorial Optimization*

EGON BALAS† eb17@andrew.cmu.edu
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract. This is an overview of the significance and main uses of projection, lifting and extended formulation in integer and combinatorial optimization. Its first two sections deal with those basic properties of projection that make it such an effective and useful bridge between problem formulations in different spaces, i.e. different sets of variables. They discuss topics like projection and restriction, the integrality-preserving property of projection, the dimension of projected polyhedra, conditions for facets of a polyhedron to project into facets of its projections, and so on. The next two sections describe the use of projection for comparing the strength of different formulations of the same problem, and for proving the integrality of polyhedra by using extended formulations or lifting. Section 5 deals with disjunctive programming, or optimization over unions of polyhedra, whose most important incarnation are mixed 0-1 programs and their partial relaxations. It discusses the compact representation of the convex hull of a union of polyhedra through extended formulation, the connection between the projection of the latter and the polar of the convex hull, as well as the sequential convexification of facial disjunctive programs, among them mixed 0-1 programs, with the related concept of disjunctive rank. Section 6 reviews lift-and-project cuts, the construction of cut generating linear programs, and techniques for lifting and for strengthening disjunctive cuts. Section 7 discusses the recently discovered possibility of solving the higher dimensional cut generating linear program without explicitly constructing it, by a sequence of properly chosen pivots in the simplex tableau of the linear programming relaxation. Finally, section 8 deals with different ways of combining cuts with branch and bound, and briefly discusses computational experience with lift-and-project cuts.

Keywords: disjunctive programming, sequential convexification, 0-1 programming, lift-and-project

1. Basic properties of projection

Most combinatorial optimization problems tend to have several alternative formulations, some of which are easier to handle than others. Projection provides a connection between different formulations. Since most combinatorial optimization problems are solved by some combination of polyhedral methods and enumeration, and since the latter typically involves repeated solution of variants of the problem's linear programming relaxation, it is crucial to be able to compare the strength of the LP relaxations of different formulations of the same problem.

* This is an updated and extended version of the paper published in LNCS 2241, Springer, 2001 (as given in Balas, 2001).

† Research was supported by the National Science Foundation through grant #DMI-9802773 and by the Office of Naval Research through contract N00014-97-1-0196.

Given a polyhedron of the form

$$Q := \{(u, x) \in \mathbb{R}^p \times \mathbb{R}^q : Au + Bx \leq b\},$$

where A, B and b have m rows, the *projection of Q onto $\mathbb{R}^q$*, or *onto the x-space*, is defined as

$$\text{Proj}_x(Q) := \{x \in \mathbb{R}^q : \exists u \in \mathbb{R}^p : (u, x) \in Q\}.$$

Thus projecting the set Q onto $\mathbb{R}^q$ can be viewed as finding a matrix $C \in \mathbb{R}^{m \times q}$ and a vector $d \in \mathbb{R}^m$ such that

$$\text{Proj}_x(Q) = \{x \in \mathbb{R}^q : Cx \leq d\}.$$

This task can be accomplished as follows. Call the polyhedral cone

$$W := \{v : vA = 0, v \geq 0\}$$

the *projection cone* associated with $\text{Proj}_x(Q)$. Then we have

Theorem 1.1.

$$\text{Proj}_x(Q) = \{x \in \mathbb{R}^q : (vB)x \leq vb, v \in \text{extr } W\}$$

where extr W denotes the set of extreme rays of W.

Proof. W is a pointed cone (since $W \subseteq \mathbb{R}^m_+$), hence it is the conical hull of its extreme rays. Therefore the system of inequalities defining $\text{Proj}_x(Q)$ holds for all $x \in \text{extr } W$ if and only if it hold for all $x \in W$.

Now let $\bar{x} \in \text{Proj}_x(Q)$. Then there exists $\bar{u} \in \mathbb{R}^p$ such that $A\bar{u} + B\bar{x} \leq b$. Premultiplying this system with any $v \in W$, $\bar{x}$ satisfies $(vB)x \leq vb$ for all $v \in W$.

Conversely, suppose $\hat{x} \in \mathbb{R}^q$ satisfies $(vB)x \leq vb$ for all $v \in W$. Then there exists no $v \geq 0$ satisfying $vA = 0$ and $v(B\hat{x} - b) > 0$. But then, by Farkas' Lemma, there exists $\hat{u}$ satisfying $A\hat{u} \leq b - B\hat{x}$, i.e. $\hat{x} \in \text{Proj}_x(Q)$. □

Several comments are in order.

First, if some of the inequalities defining Q are replaced with equations, the corresponding components of v become unrestricted in sign. In such a case, if the projection cone W is not pointed, then it is not the convex hull of its extreme rays. Nevertheless, like any polyhedral cone, W is finitely generated, and any finite set $\widehat{W}$ of its generators can play the role previously assigned to extr W.

Second, the set Q may be defined, besides the system of inequalities involving the variables u to be projected out, by any additional constraints—not necessarily linear—not involving u: such constraints will become part of the projection, without any change.

In other words, for an arbitrary set $S \subseteq \mathbb{R}^q$, the projection of

$$Q := \{(u, x) \in \mathbb{R}^p \times \mathbb{R}^q : Au + Bx \leq b, x \in S\}$$

on $\mathbb{R}^q$ is

$$\text{Proj}_x(Q) := \{x \in \mathbb{R}^q : (vB)x \leq vb \text{ for all } v \in W, x \in S\}$$

where W is the projection cone introduced earlier.

Third, $\text{Proj}_x(Q)$ as given by Theorem 1.1 may have redundant inequalities, even when the latter come only from extreme rays of W; i.e., an extreme ray of W does not necessarily give rise to a facet of $\text{Proj}_x(Q)$. It would be nice to be able to identify the extreme rays of W which give rise to facets of $\text{Proj}_x(Q)$, but unfortunately this in general is not possible: whether an inequality $(vB)x \leq vb$ is or is not facet defining for $\text{Proj}_x(Q)$ depends on B and b as well as on v. We will return to this question later.

Fourth, we have the simple fact following from the definitions, that if $x \in \mathbb{R}^q$ is an extreme point of $\text{Proj}_x(Q)$, then there exists $u \in \mathbb{R}^p$ such that (u, x) is an extreme point of Q. This has the following important consequence:

Proposition 1.2. If Q is an integral polyhedron, then $\text{Proj}_x(Q)$ is an integral polyhedron.

Well known special cases

There are two well known algorithms whose iterative steps are special cases of projection:

If the matrix A in the definition of Q has a single column a, i.e., if

$$Q := \{(u_0, x) : au_0 + Bx \leq b\},$$

then

$$\text{extr } W = \{v^k : a_k = 0\} \cup \{v^{ij} : a_i \cdot a_j < 0\},$$

where

$$v^k := e_k, \quad v^{ij} := a_i e_j - a_j e_i,$$

with e_k the k-th unit vector, and we have one step of *Fourier Elimination* (see e.g. Stoer and Witzgall, 1970).

If Q is of the form

$$Q := \left\{(u, x, x_0) \;\middle|\; \begin{array}{ll} -cu - dx + x_0 \leq 0 \\ A'u + B'x \quad\quad \leq b' \end{array}\right\},$$

then

$$W = \{(v_0, v) : -v_0 c + vA' = 0; v_0, v \geq 0\},$$

$$\text{Proj}_{(x_0,x)}(Q) := \left\{(x_0, x) \;\middle|\; \begin{array}{r} x_0 + (vB' - d)x \leq vb', \ \forall v : (1, v) \in \text{extr } W \\ (vB')x \leq vb', \ \forall v : (0, v) \in \text{extr } W \end{array}\right\}$$

and we have the constraint set of the *Benders Master Problem* (see e.g. Nemhauser and Wolsey, 1986).

Projection and restriction

Projection is not to be confused with *restriction*, another operation that relates a higher dimensional object to one in a subspace. Referring to the same set $Q \subseteq \mathbb{R}^p \times \mathbb{R}^q$, its restriction to the subspace defined by $u = 0$ is

$$\{(u, x) \in Q : u = 0\} = \{x \in \mathbb{R}^q : (0, x) \in Q\}.$$

Notice that, unlike in the case of projection, the restriction of Q "to $\mathbb{R}^q$," or "to the x-space," is not well defined, since it may mean several things, of which the above is just one. Another one is, for some $t \in \mathbb{R}^p$,

$$\{(u, x) \in Q : u = t\} = \{x \in \mathbb{R}^q : (t, x) \in Q\}.$$

Clearly, the restriction of Q through some value assignment to each component of u is always a subset of $\text{Proj}_x(Q)$.

To illustrate the difference between projection and restriction, consider the set $Q := \{(x, u) \in \mathbb{R}^2 : x + u \geq 1, 0 \leq x \leq 2, 0 \leq u \leq 1\}$. Its projection on the x-space is $\{x \in \mathbb{R} : 0 \leq x \leq 2\}$, whereas its restriction to the subspace of $\mathbb{R}^2$ defined by $u = 0$ is $\{x \in \mathbb{R} : 1 \leq x \leq 2\}$, as shown in figure 1.

While both projection and restriction map polyhedra from a higher dimensional space to a lower dimensional one, sometimes we want to go into the opposite direction. The operation, in a sense reverse to projection, in which one goes from a given polyhedral

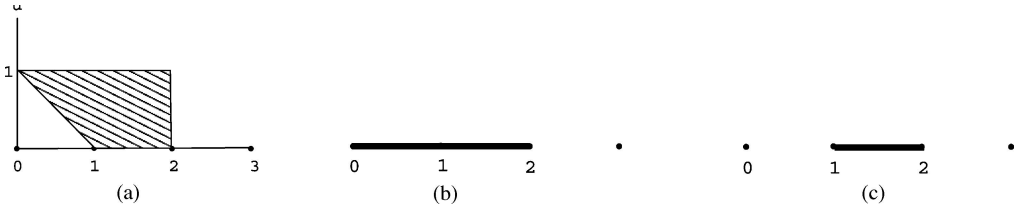

Figure 1. (a) Q, (b) $\text{Proj}_x Q$, and (c) $\{(x, u) \in Q : u = 0\}$.

formulation of a problem to a higher dimensional one, involving some new variables, is called *extended formulation*. Sometimes going to an extended formulation is referred to as lifting. For instance, many problems defined on graphs that are usually formulated in terms of arc variables, can also be formulated in the higher dimensional space of arc- and node-variables; and we will see examples where this is advantageous. Typically, extended formulations are not unique, and it takes some insight to recognize those which offer some advantage. One possible advantage might be that the extended formulation gives rise to a polyhedron with desirable properties.

The reverse of restriction of course is relaxation, i.e. the removal of the restricting constraint. But this operation becomes less trivial than it sounds when looked at in the following context. Suppose we know a valid inequality, $\alpha x \leq \alpha_0$, for a polyhedron $P \cap S$ that is a restriction to a subspace of a higher dimensional polyhedron P, and we would like to calculate the strongest (tightest) extension of the inequality, say $\alpha x + \beta y \leq \alpha_0$, to P. This is a well defined and important operation called *lifting*. It consists of calculating the missing coefficients β_j of the inequality $\alpha x + \beta y \leq \alpha_0$ valid for P. There is a well known procedure for doing this, called *sequential lifting* (see e.g. Nemhauser and Wolsey, 1986), which calculates the coefficients β_j one by one, and under certain conditions guarantees that if $\alpha x \leq \alpha_0$ defines a facet of $P \cap S$, then $\alpha x + \beta y \leq \alpha_0$ defines a facet of P.

Exercise 1.1 Consider the following variations of Q:

$Q_1 := \{(u, x) \in \mathbb{R}^p \times \mathbb{R}^q : Au + Bx = b\}$
$Q_2 := \{(u, x) \in \mathbb{R}^p \times \mathbb{R}^q : Au + Bx \leq b,\ u \geq 0\}$
$Q_3 = Q$ with $B = \binom{I}{0}$, where I is the $q \times q$ identity matrix and 0 is the $(m - q) \times q$ zero matrix (assume $m > q$).

For each Q_i, $i = 1, 2, 3$ write down the projection cone associated with $\text{Proj}_x(Q_i)$.

Exercise 1.2 Prove Proposition 1.2.

2. Dimensional aspects of projection

When we have a polyhedral representation of some combinatorial object, we would like it to be nonredundant. This makes us interested in identifying among the inequalities defining the polyhedron, say Q, those which are facet inducing, since the latter provide a minimal representation. Further, when we project Q onto a subspace, we would like to know whether the facets of Q project into facets of the projection. To be able to answer this and similar questions, we need to look at the relationship between the dimension of a polyhedron and the dimension of its projections. This was done in the paper (Balas and Oosten, 1998), on which most of this section is based.

Consider the polyhedron $Q := \{(u, x) \in \mathbb{R}^p \times \mathbb{R}^q : Au + Bx \leq b\} \neq \emptyset$, where A, B and b have m rows, and let us partition (A, B, b) into $(A^=, B^=, b^=)$ and $(A^\leq, B^\leq, b^\leq)$, where $A^=u + B^=x = b^=$ is the *equality subsystem* of Q, the set of equations corresponding to the inequalities satisfied at equality by every $(u, x) \in Q$. Let $r := \text{rank}(A^=, B^=) = \text{rank}(A^=, B^=, b^=)$, where the last equality follows from $Q \neq \emptyset$.

Let $\dim(Q)$ denote the dimension of Q. It is well known that $\dim(Q) = p + q - r$, and that Q is full-dimensional, i.e. $\dim(Q) = p + q$, if and only if the equality subsystem is vacuous. When this is the case, then $\dim(\text{Proj}_x(Q)) = q$, for otherwise $\text{Proj}_x(Q)$ has a nonempty equality subsystem that must also be valid for Q, contrary to the assumption that the latter is full dimensional.

Consider now the general situation, when Q is not necessarily full dimensional. Recall that $r = \text{rank}(A^=, B^=)$, and define $r^* := \text{rank}(A^=)$. Clearly, $0 \leq r^* \leq \min\{r, p\}$. It can then be shown that

Theorem 2.1 (Balas and Oosten, 1998)**.** $\dim(\text{Proj}_x(Q)) = \dim(Q) - p + r^*$.

It follows that $\dim(\text{Proj}_x(Q)) = \dim(Q)$ if and only if $r^* = p$, i.e., the projection operation is dimension-preserving if and only if the matrix $A^=$ is of full column rank.

When is the projection of a facet a facet of the projection?

We now turn to the question of what happens to the facets of Q under projection. Since a facet of a polyhedron is itself a polyhedron, the dimension of its projection can be deduced from Theorem 2.1.

Let $\alpha u + \beta x \leq \beta_0$ be a valid inequality for Q, and suppose

$$F := \{(u, x) \in Q : \alpha u + \beta x = \beta_0\}$$

is a facet of Q. Let $\binom{\alpha}{A}^= u + \binom{\beta}{B}^= x = \binom{\beta_0}{b}^=$ be the equality subsystem of F, and let $r_F := \text{rank}(\binom{\alpha}{A}^=, \binom{\beta}{B}^=)$. Notice that $r_F - r = 1$, since by assumption $\dim(F) = \dim(Q) - 1$. Further, denote $r_F^* := \text{rank}(\binom{\alpha}{A}^=)$. We may interpret $r_F^* - r^*$ as the difference between the number of dimensions "lost" in projecting Q (which is $p - r^*$) and in projecting F (which is $p - r_F^*$). From Theorem 2.1 we then have

Corollary 2.2. $\dim(\text{Proj}_x(F)) = \dim(\text{Proj}_x(Q)) - 1 + (r_F^* - r^*)$.

Indeed, $\dim(\text{Proj}_x(F)) = \dim(F) - p + r_F^* = \dim(Q) - 1 - p + r_F^*$, and substituting for $\dim(Q)$ its value given by Theorem 2.1 yields the Corollary.

We are now in a position to state what happens to the facets of Q under projection.

Corollary 2.3. Let F be a facet of Q. Then $\text{Proj}_x(F)$ is a facet of $\text{Proj}_x(Q)$ if and only if $r_F^* = r^*$.

Proof outline. From Corollary 2.2. it is clear that $\text{Proj}_x(F)$ has the dimension of a facet of $\text{Proj}_x(Q)$ if and only if $r^*_F = r^*$. However, this by itself does not amount to a proof, unless we can guarantee that $\text{Proj}_x(F)$ is a face of $\text{Proj}_x(Q)$, which is far from obvious: in general, the projection of a face need not be a face of the projection. Indeed, think of a 3-dimensional pyramid, and its projection on its base: neither the top vertex of the pyramid, nor the 3 edges incident with it, become faces of the projection as a result of the operation.

However, if $\text{Proj}_x(F)$ is a set of the form $\{x \in \text{Proj}_x(Q) : \beta x = \beta_0\}$ for some valid inequality $\beta x \leq \beta_0$ for $\text{Proj}_x(Q)$, then clearly $\text{Proj}_x(F)$ is a face of $\text{Proj}_x(Q)$. This is the case in our situation: since F is a facet of Q, it has a defining inequality $\alpha u + \beta x \leq \beta_0$. Now if $r^*_F = r^*$, then there exists a vector λ such that $\alpha = \lambda A^=$; hence subtracting $\lambda A^= u + \lambda B^= x = \lambda b^=$ from this defining inequality yields $(\beta - \lambda B^=)x \leq \beta_0 - \lambda b^=$, which can be written as $\beta' x \leq \beta'_0$. Thus $\text{Proj}_x(F) = \{x \in \text{Proj}_x(Q) : \beta' x = \beta'_0\}$, i.e. $\text{Proj}_x(F)$ is a face of $\text{Proj}_x(Q)$; hence a facet. □

Another consequence of Theorem 2.1. which is not hard to prove is this:

Corollary 2.4. Let $r^* = p$, i.e. let $A^=$ be of full column rank, and let $0 \leq d \leq \dim(Q) - 1$. Then every d-dimensional face of Q projects into a d-dimensional face of $\text{Proj}_x(Q)$.

Projection with a minimal system of inequalities

As discussed in Section 1, the inequalities of the system defining $\text{Proj}_x(Q)$ are not necessarily facet inducing, even though they are in 1-1 correspondence with the extreme rays of the projection cone W. In other words, the system $(vB)x \leq vb, \forall v \in \text{extr}\, W$, is not necessarily minimal. In fact, this system often contains a large number of redundant inequalities. It has been shown however (Balas, 1998), that if a certain linear transformation is applied to Q, which replaces the coefficient matrix B of x with the identity matrix I, the resulting polyhedron Q^0 has a projection cone W^0 that is pointed, and the projection of Q^0 is of the form

$$\text{Proj}_x(Q^0) = \{x \in \mathbb{R}^q : vx \leq v_0 \text{ for all } (v, v_0) \in \mathbb{R}^{m+1} \text{ such that } (v, w, v_0) \in \text{extr}\, W^0\},$$

with $\text{Proj}_x(Q^0) = \text{Proj}_x(Q)$. Here w is a set of auxiliary variables generated by the above transformation. Furthermore, we have the following

Proposition 2.5 (Balas, 1998)**.** Let $\text{Proj}_x(Q)$ be full dimensional. Then the inequality $vx \leq v_0$ defines a facet of $\text{Proj}_x(Q)$ if and only if (v, v_0) is an extreme ray of the cone $\text{Proj}_{(v,v_0)}(W^0)$.

The linear transformation that takes Q into Q^0 is of low complexity ($O(\max\{m, q\}^3)$, where B is $m \times q$). The need to project the cone W^0 onto the subspace (v, v_0)

arises only when B is not of full row rank. In the important case when B is of full row rank, we have the following stronger result:

Corollary 2.6 (Balas, 1998)**.** Let $\text{Proj}_x(Q)$ be full dimensional and $\text{rank}(B) = m$. Then the inequality $vx \leq v_0$ defines a facet of $\text{Proj}_x(Q)$ if and only if (v, v_0) is an extreme ray of W^0.

In many important cases the matrix B is of the form $B = \binom{I}{0}$, which voids the need for the transformation discussed above and leads to a projection cone whose extreme rays yield facet inducing inequalities for $\text{Proj}_x(Q)$. This is the case encountered, for instance, in the characterization of the perfectly matchable subgraph polytope of a graph to be discussed in Section 4; as well as in the projection used in the convex hull characterization of a disjunctive set discussed in Section 5.

Exercise 2.1 Give an example of a 3-dimensional polyhedron Q and one of its 2-dimensional projections $\text{Proj}(Q)$ such that $\text{Proj}(Q)$ has (a) fewer facets than Q; (b) more facets than Q; (c) the same number of facets as Q (visual representation will suffice).

3. Comparing different formulations

Combinatorial optimization problems tend to have several formulations, and choosing the most convenient one is often a nontrivial exercise. While the number of variables and constraints in the different formulations do have some importance, the most relevant criterion of the comparison is the strength of the linear programming relaxation. This is so because most solution procedures involve some branch and bound component, and the bounding usually comes from the LP relaxation: the tighter this relaxation, the stronger the bound.

Typically the comparisons are made between two problem formulations, say P and Q, such that P is expressed in terms of variables $x \in \mathbb{R}^n$, while Q is in terms of variables $(x, y) \in \mathbb{R}^n \times \mathbb{R}^q$. To compare the strength of the two LP relaxations, one has to express both P and Q in terms of the same variables, and so one uses projection to eliminate y and re-express Q in terms of x, i.e., as $\text{Proj}_x(Q)$. One then compares the LP relaxation of this latter set to that of P.

Before we give several examples of how this can be done, we wish to point out that projection can be used to compare two different formulations of the same problem even when they are expressed in completely different (nonoverlapping) sets of variables, provided the two sets are related by an affine transformation. Suppose, for instance, that we have two equivalent formulations of a problem whose feasible sets are P and Q:

$$P := \{x \in \mathbb{R}^n : Ax \leq a_0\}, \quad Q := \{y \in \mathbb{R}^q : By \leq b_0\}$$

where A is $m \times n$, B is $p \times q$, and

$$x = Ty + c \tag{1}$$

for some $T \subset \mathbb{R}^n \times \mathbb{R}^q$ and $c \in \mathbb{R}^n$.

Define

$$Q^+ := \left\{ (x, y) \in \mathbb{R}^{n+q} \;\middle|\; \begin{array}{r} x - Ty = c \\ By \leq b_0 \end{array} \right\}$$

and project Q^+ onto $\mathbb{R}^n$. Using the projection cone

$$W := \{(v, w) \in \mathbb{R}^{n+p} : -vT + wB = 0, w \geq 0\},$$

we obtain

$$\text{Proj}_x(Q^+) := \{x \in \mathbb{R}^n : vx \leq vc + wb_0, \forall (v, w) \in W\}.$$

Then to compare P with Q, we compare P with $\text{Proj}_x(Q^+)$.

The affine transformation (1) can also be used to derive a different analytical comparison between P and Q that does not make use of projection (Padberg and Sung, 1991).

The traveling salesman problem

Consider the constraint set of the TSP defined on the complete digraph G on $n+1$ nodes in two well known formulations:

Dantzig, Fulkerson, and Johnson (D-F-J) (1954)

$$\begin{array}{lll}
\sum (x_{ij} : j = 0, \ldots, n) = 1 & & i = 0, \ldots, n \\
\sum (x_{ij} : i = 0, \ldots, n) = 1 & & j = 0, \ldots, n \\
\sum (x_{ij} : i \in S, j \in S) \leq |S| - 1, & S \subseteq \{0, \ldots, n\}, & 2 \leq |S| \leq \dfrac{n+1}{2} \\
x_{ij} \in \{0, 1\}, & & i, j = 0, \ldots, n
\end{array}$$

Miller, Tucker, and Zemlin (M-T-Z) (1960)

$$\begin{array}{lll}
\sum (x_{ij} : j = 0, \ldots, n) = 1 & & i = 0, \ldots, n \\
\sum (x_{ij} : i = 0, \ldots, n) = 1 & & j = 0, \ldots, n \\
u_i - u_j + n x_{ij} \leq n - 1 & i, j = 1, \ldots, n, & i \neq j \\
x_{ij} \in \{0, 1\} & i, j = 0, \ldots, n &
\end{array}$$

The M-T-Z formulation introduces n node variables, but replaces the exponentially large set of subtour elimination constraints by $n(n-1)$ new constraints that achieve the same goal. However, projecting the constraint set of the M-T-Z formulation into the subspace of the arc variables by using the cone

$$W = \{v \mid vA = 0, \quad v \geq 0\}, \tag{2}$$

where A is the transpose of the node-arc incidence matrix of G, yields the inequalities

$$\sum(x_{ij} : (i, j) \in C) \leq \frac{n-1}{n}|C|$$

for every directed cycle C of G. These inequalities are strictly weaker than the corresponding subtour elimination inequalities of the D-F-J formulation, so the latter provides a tighter LP relaxation.

The set covering problem

Consider the set covering problem

$$\min\{cx|Ax \geq e, \quad x \in \{0, 1\}^n\} \tag{SC}$$

where $e = (1, \ldots, 1)$ and A is a matrix of 0's and 1's. Let M and N index the rows and columns of A, respectively, and

$$M_j = \{i \in M|a_{ij} = 1\}, \quad j \in N, \quad N_i = \{j \in N|a_{ij} = 1\}, \quad i \in M.$$

The following uncapacitated plant location (UPL) problem is known to be equivalent to (SC):

$$\begin{array}{lll} \min & \sum(c_j x_j : j \in N) & \\ & \sum(u_{ij} : j \in N_i) \geq 1, & i \in M \\ & -\sum(u_{ij} : i \in M_j) + |M_j|x_j \geq 0, & j \in N \\ & u_{ij} \geq 0,\ i \in M,\ j \in N;\ x_j \in \{0, 1\}, & j \in N \end{array}$$

If we now project this constraint set onto the x-space by using the cone

$$W = \{v|v_i - v_j \leq 0,\ j \in N_i,\ i \in M :\ v_i \geq 0,\ i \in M\},$$

we obtain the inequalities

$$\begin{array}{ll} \sum(|M_j|x_j : j \in N_S) \geq |S|, & \forall S \subseteq M; \\ x_j \geq 0, & j \in N \end{array}$$

(where $N_S := \cup(N_i : i \in S)$, each of which is dominated (strictly if $S \neq M$) by the sum of the inequalities of (SC) indexed by S. Hence the LP relaxation of the UPL formulation is weaker than that of (SC).

If, on the other hand, we use the so called "strong" formulation of the uncapacitated plant location problem, namely

$$\begin{aligned} \min \quad & \sum(c_j x_j : j \in N) \\ & \sum(u_{ij} : j \in N_i) \geq 1 \quad i \in M \\ & -u_{ij} + x_j \geq 0 \quad i \in M,\ j \in N; \\ & u_{ij} \geq 0,\ x_j \in \{0,1\}, \quad i \in M,\ j \in N; \end{aligned}$$

then the projected inequalities include

$$\begin{aligned} \sum(x_j : j \in N_i) \geq 1, \quad & i \in M; \\ x_j \geq 0 \qquad & j \in N \end{aligned}$$

thus yielding the same LP relaxation as that of (SC).

Nonlinear 0-1 programming

Consider the nonlinear inequality

$$\begin{aligned} \sum_{j \in N} a_j \Big(\pi_{i \in Q_j} x_i \Big) \leq b, \quad & a_j > 0,\ j \in N \\ x_i \in \{0,1\}, \quad & i \in Q_j,\ j \in N \end{aligned}$$

where π denotes product.

It is a well known linearization technique due to Fortet (1960) to replace this inequality by

$$\begin{aligned} \sum_{j \in N} a_j y_j \leq b & \\ -y_j + \sum_{i \in Q_j} x_i \leq |Q_j| - 1 \qquad & j \in N \\ y_j - x_i \leq 0 \quad i \in Q_j, \quad & j \in N \\ y_j \geq 0,\ x_i \in \{0,1\}, \qquad & i \in Q_j,\ j \in N \end{aligned}$$

When the number of nonlinear terms is small relative to the number of variables, this is perfectly satisfactory. However, often the number of terms is a high-degree polynomial in the number of variables, in which case it is desirable to eliminate the new variables y_j, $j \in N$.

The cone needed for projecting the above constraint set onto the x-space has for every $M \subseteq N$ an extreme direction vector

$$v_M = (1; w_M; 0),$$

where 1 is a scalar, 0 is the zero vector with $\sum(|Q_j| : j \in N)$ components, and $w_M \in \mathbb{R}^n$ is defined by

$$w_j = \begin{cases} a_j & \text{if } j \in M \\ 0 & \text{if } j \in N - M. \end{cases}$$

The corresponding projected inequalities are

$$\sum_{i \in Q_M} \left(\sum_{j : i \in Q_j} a_j \right) x_i \leq b - \sum_{j \in M} a_j (|Q_j| - 1), \;\; M \subseteq N,$$

where $Q_M := \cup(Q_j : j \in M)$.

But this is precisely the linearization of Balas and Mazzola (1984), arrived at by other means.

Exercise 3.1 Show that the extreme rays of the cone W defined in (2) are the incidence vectors of the directed cycles of G.

4. Proving the integrality of polyhedra

One of the important uses of projection in combinatorial optimization is to prove the integrality of certain polyhedra. It often happens that the LP relaxation of a certain formulation, say P, does not satisfy any of the known sufficient conditions for it to have the integrality property; but there exists a higher dimensional formulation whose LP relaxation, say Q, satisfies such a condition (for the relevant variables). In such a case all we need to do is to show that P is the projection of Q onto the subspace of the relevant variables. It then follows from Proposition 1.2. that P is integral.

We will illustrate the procedure on several examples.

Perfectly matchable subgraphs of a bipartite graph

Let $G = (V, E)$ be a bipartite graph with bipartition $V = V_1 \cup V_2$, let $G(W)$ denote the subgraph induced by $W \subseteq V$, and let X be the set of incidence vectors of vertex sets W such that $G(W)$ has a perfect matching.

The Perfectly Matchable Subgraph (PMS-) polytope of G is then conv(X), the convex hull of X. Its linear characterization (Balas and Pulleyblank, 1983) can be obtained by projection as follows.

Fact (from the Kőnig-Hall Theorem)

$G(W)$ has a perfect matching if and only if

$$|W \cap V_1| = |W \cap V_2|$$

and for every $S \subseteq W \cap V_1$

$$|S| \leq |N(S)|,$$

where

$$N(S) := \{j \in N \,|(i, j) \in E \text{ for some } i \in S\}.$$

Theorem 4.1 (Balas and Pulleyblank, 1983)**.** The PMS polytope of the bipartite graph G is defined by the system

$$\begin{aligned} 0 \leq x_i \leq 1 \quad & i \in V \\ x(V_1) - x(V_2) = 0 \quad & \\ x(S) - x(N(S)) \leq 0 \quad & S \subseteq V_1. \end{aligned} \tag{3}$$

If $0 \leq x_i \leq 1$ is replaced by $x_i \in \{0, 1\}$, the theorem is simply a restatement of the Kőnig-Hall condition in terms of incidence vectors. So the proof of the theorem amounts to showing that the polytope defined by (3) is integral.

Note that the coefficient matrix of (3) is not totally unimodular.

To prove the theorem, we restate the constraint set in terms of vertex and edge variables, x_i and u_{ij}, respectively, i.e., in a higher dimensional space. We then obtain the system

$$\begin{aligned} u(i, N(i)) - x_i = 0 \quad & i \in V_1 \\ u(N(j), j) - x_j = 0 \quad & j \in V_2 \\ x(V_1) - x(V_2) = 0 \quad & \\ u_{ij} \geq 0,\ (i, j) \in E;\ 0 \leq x_i \leq 1, \quad & i \in V \end{aligned} \tag{4}$$

whose coefficient matrix is totally unimodular. Here $u(i, N(i)) := \sum(u_{ij} : j \in N(i))$ and $u(N(j), j) := \sum(u_{ij} : i \in N(j))$. Thus the polyhedron defined by (4) is integral, and if it can be shown that its projection is the polyhedron defined by (3), this is proof that the latter is also integral. This is indeed the case (see Balas and Pulleyblank (1983) for a proof). Moreover, the projection yields the system (3) with the condition $S \subseteq V_1$ replaced by

$$S \subseteq V_1 \text{ such that } G(S \cup N(S)) \quad \text{and} \quad G((K_1 \setminus S) \cup (K_2 \setminus N(S))) \text{ are connected,}$$

where K is the component of G containing $S \cup N(S)$, and $K_i = K \cap V_i$, $i = 1, 2$.

This in turn allows one to weaken the "if" requirement of the Kőnig-Hall Theorem to "if $|S| \leq |N(S)|$ for all $S \subseteq V_1$ such that $G(S \cup N(S))$ and $G((K_1 \setminus S) \cup (K_2 \setminus N(S)))$ are connected."

Assignable subgraphs of a digraph

The well known *Assignment Problem* (AP) asks for assigning n people to n jobs. When represented on a digraph, an assignment, i.e. a solution to AP, consists of a collection of arcs spanning G that forms a node-disjoint union of cycles (a cycle decomposition). A digraph $G = (V, A)$ is *assignable* (admits a cycle decomposition) if the assignment problem on G has a solution. The Assignable Subgraph Polytope of a digraph is the convex hull of incidence vectors of node sets W such that $G(W)$ is assignable.

Let $\deg^+(v)$ and $\deg^-(v)$ denote the outdegree and indegree, respectively, of v and for $S \subset V$, let $\Gamma(S) := \{j \in V | (i, j) \in A \text{ for some } i \in S\}$.

Projection can again be used to prove the following

Theorem 4.3 (Balas, 1987)**.** The Assignable Subgraph Polytope of the digraph G is defined by the system

$$\begin{aligned} 0 \leq x_i \leq 1 \quad & i \in V \\ x(S \setminus \Gamma(S)) - x(\Gamma(S) \setminus S) \leq 0, \quad & S \subseteq V. \end{aligned}$$

Path decomposable subgraphs of an acyclic digraph

An acyclic digraph $G = (V, A)$ with two distinguished nodes, s and t, is said to admit an *s-t path decomposition* if there exists a collection of interior node disjoint s-t paths that cover all the nodes of G. The s-t Path Decomposable Subgraph Polytope of G is then the convex hull of incidence vectors of node sets $W \subseteq V - \{s, t\}$ such that $G(W \cup \{s, t\})$ admits an s-t path decomposition.

For $S \subseteq V$, define

$$\Gamma^*(S) := \begin{cases} (\Gamma(S) \setminus \{t\}) \cup \Gamma(s) & \text{if } t \in \Gamma(S) \\ \Gamma(S) & \text{if } t \notin \Gamma(S) \end{cases}$$

Projection can then be used to prove the following

Theorem 4.4 (Balas, 1987)**.** The s-t Path Decomposable Subgraph Polytope of the acyclic digraph $G = (V, A)$ is defined by the system

$$\begin{aligned} 0 \leq x_i \leq 1 \quad & i \in V \\ x(S \setminus \Gamma^*(S)) - x(\Gamma^*(S) \setminus S) \leq 0 \quad & S \subseteq V - \{s, t\} \end{aligned}$$

Perfectly matchable subgraphs of an arbitrary graph

For an arbitrary undirected graph $G = (V, E)$, the perfectly matchable subgraph (PMS-) polytope, defined as before, can also be characterized by projection, but this is a considerably more arduous task than in the case of a bipartite graph. The difficulty

partly stems from the fact that the projection cone in this case is not pointed and thus instead of the extreme rays one has to work with a finite set of generators. On the other hand, an interesting feature of the technique used in this case is that a complete set of generators did not have to be found; it was sufficient to identify a set of generators that produce all facet defining inequalities of the PMS-polytope.

For $W \subset V$, let $G(W)$ be the subgraph of G induced by W, let $c(W)$ be the number of components of $G(W)$, and let $N(W)$ be the set of neighbors of W, i.e. $N(W) := \{j \in V \backslash W) : (i, j) \in E \text{ for some } i \in W\}$.

Theorem 4.4 (Balas and Pulleyblank, 1989)**.** The PMS polytope of an arbitrary graph $G = (V, E)$ is defined by the system

$$\begin{aligned} &0 \leq x_i \leq 1 \quad i \in V \\ &x(S) - x(N(S)) \leq |S| - c(S) \end{aligned} \tag{5}$$

for all $S \subseteq V$ such that every component of $G(S)$ consists of a single node or else is a nonbipartite graph with an odd number of nodes.

For further details and a proof see Balas and Pulleyblank (1989).

Exercise 4.1 Discuss the connection between Theorem 4.4 and Tutte's condition for a graph to have a perfect matching.

5. Disjunctive programming

Disjunctive programming is optimization over unions of polyhedra. While polyhedra are convex sets, their unions of course are not. The name reflects the fact that the objects investigated by this theory can be viewed as the solution sets of systems of linear inequalities joined by the logical operations of conjunction, negation (taking of complement) and disjunction, where the nonconvexity is due to the presence of disjunctions. Pure and mixed integer programs, in particular pure and mixed 0-1 programs can be viewed as disjunctive programs; but the same is true of a host of other problems, like for instance the linear complementarity problem. It is clear that if, for instance, a linear program over a feasible set F is amended with the condition that variable x_j has to be an integer between 0 and k, which can be written as $(x_j = 0) \vee (x_j = 1) \ldots \vee (x_j = k)$ (with "$\vee$" the logical "or" symbol), then it becomes an optimization problem over a union of polyhedra $P_0 \cup P_1 \cup \cdots \cup P_k$, where $P_i := \{x \in F : x_j = i\}$ for $i = 0, 1, \ldots, k$. The main application of disjunctive programming has so far been to integer and, in particular, 0-1 programming, where it has served as the source of a rich family of cutting planes and in the early 90's has produced a computationally successful variant known as lift-and-project (Balas, Ceria, and Cornuéjols, 1993).

The foundations of disjunctive programming were laid in a July 1974 technical report, published 24 years later as an invited paper (Balas, 1998) with a foreword. For

additional work on disjunctive programming in the seventies and eighties see Balas (1979, 1985); Balas and Jeroslow (1980); Balas, Tama, and Tind (1989); Blair (1976, 1980); Jeroslow (1977, 1987, 1989); Sherali and Shetty (1980). In particular, Balas (1979) contains a detailed account of the origins of the disjunctive approach and the relationship of disjunctive cuts to Gomory's mixed integer cut, intersection cuts and others. Disjunctive programming received a new impetus in the early nineties from the work on matrix cones by Lovász and Schrijver (1991), see also Sherali and Adams (1990). The version that led to the computational breakthroughs of the nineties is described in the two papers by Balas, Ceria, and Cornuéjols (1993, 1996), the first of which discusses the cutting plane theory behind the approach, while the second deals with the branch-and-cut implementation and computational testing. Related recent developments are discussed in Balas et al (1996), Balas (1997), Balas and Perregaard (2003), Beaumont (1990), Ceria and Pataki (1998), Ceria and Soares (1997, 1999), Letchford (2001), Perregaard and Balas (2001), Stubbs and Mehrotra (1996), Turkay and Grossmann (1996) and Williams (1994).

A *disjunctive set*, i.e. the constraint set of a disjunctive program, can be expressed in many different forms, of which the following two extreme ones have special significance. Let

$$P_i := \{x \in \mathbb{R}^n : A^i x \geq b^i\}, \qquad i \in Q$$

be convex polyhedra, with Q a finite index set and (A^i, b^i) an $m_i \times (n+1)$ matrix, $i \in Q$, and let $P := \{x \in \mathbb{R}^n : Ax \geq b\}$ be the polyhedron defined by those inequalities (if any) common to all P_i, $i \in Q$. Then the disjunctive set $\bigcup_{i \in Q} P_i$ over which we wish to optimize some linear function can be expressed as

$$\left\{ x \in \mathbb{R}^n : \bigvee_{i \in Q} (A^i x \geq b^i) \right\}, \tag{6}$$

which is its *disjunctive normal form* (a disjunction whose terms do not contain further disjunctions). The same disjunctive set can also be expressed as

$$\left\{ x \in \mathbb{R}^n : Ax \geq b, \ \bigvee_{h \in Q_j} \left(d^h x \geq d_0^h\right), \ j = 1, \ldots, t \right\}, \tag{7}$$

which is its *conjunctive normal form* (a conjunction whose terms do not contain further conjunctions). Here (d^h, d_0^h) is a $(n+1)$-vector for $h \in Q_j$, all j, where each set Q_j contains exactly one inequality of each system $A^i x \geq b^i$, $i \in Q$, and t is the number of all sets Q_j with this property. Thus the connection between (6) and (7) is that each term $A^i x \geq b^i$ of (6) contains $Ax \geq b$ and exactly one inequality $d^h x \geq d_0^h$ of each disjunction of (7) indexed by Q_j for $j = 1, \ldots, t$, and that all distinct systems $A^i x \geq b^i$ with this property are present among the terms of (6).

The lift-and-project approach relies mainly on the following two basic ideas (results) of disjunctive programming, the first one of which is derived from the form (6), while the second from the form (7):

1. There is a compact representation of the convex hull of a union of polyhedra in a higher dimensional space, which in turn can be projected back into the original space. The first step of this operation, i.e. the higher dimensional representation, may be viewed as lifting (or extended formulation), while the second step is projection. As a result one obtains the convex hull in the original space.
2. A large class of disjunctive sets, called *facial*, can be convexified sequentially, i.e. their convex hull can be derived by imposing the disjunctions one at a time, generating each time the convex hull of the current set.

We will discuss these two ideas in turn.

The convex hull of a union of polyhedra

Theorem 5.1 (Balas, 1998)**.** Given polyhedra $P_i := \{x \in \mathbb{R}^n : A^i x \geq b^i\} \neq \emptyset$, $i \in Q$, the closed convex hull of $\bigcup_{i \in Q} P_i$ is the set of those $x \in \mathbb{R}^n$ for which there exist vectors $(y^i, y^i_0) \in \mathbb{R}^{n+1}$, $i \in Q$, satisfying

$$\begin{aligned} x - \sum (y^i : i \in Q) &= 0 \\ A^i y^i - b^i y^i_0 &\geq 0 \\ y^i_0 &\geq 0 \qquad i \in Q \\ \sum \left(y^i_0 : i \in Q\right) &= 1. \end{aligned} \tag{8}$$

In particular, denoting by $P_Q := \text{conv}(\bigcup_{i \in Q} P_i)$ the closed convex hull of $\bigcup_{i \in Q} P_i$ and by $\mathcal{P}$ the set of vectors $(x, \{y^i, y^i_0\}_{i \in Q})$ satisfying (8),

(i) if x^* is an extreme point of P_Q, then $(\bar{x}, \{\bar{y}^i, \bar{y}^i_0\}_{i \in Q})$ is an extreme point of $\mathcal{P}$, with $\bar{x} = x^*$, $(\bar{y}^k, \bar{y}^k_0) = (x^*, 1)$ for some $k \in Q$, and $(\bar{y}^i, \bar{y}^i_0) = (0, 0)$ for $i \in Q \backslash \{k\}$.

(ii) if $(\bar{x}, \{\bar{y}^i, \bar{y}^i_0\}_{i \in Q})$ is an extreme point of $\mathcal{P}$, then $\bar{y}^k = \bar{x}$ and $\bar{y}^k_0 = 1$ for some $k \in Q$, $(\bar{y}^i, \bar{y}^i_0) = (0, 0)$, $i \in Q \backslash \{k\}$, and $\bar{x}$ is an extreme point of P_Q.

Note that in this higher dimensional representation of P_Q, the number of variables and constraints is linear in the number $|Q|$ of polyhedra in the union, and so is the number of facets of $\mathcal{P}$. Note also that in any basic solution of the linear system (8), $y^i_0 \in \{0, 1\}$, $i \in Q$, automatically, without imposing this condition explicitly.

If we impose simultaneously *all* the integrality conditions of a mixed 0-1 program with p 0-1 variables, we have a disjunction with 2^p terms, one for every p-component 0-1 point. But if we impose only disjunctions that yield a set Q of manageable size,

then this representation becomes extremely useful (such an approach is facilitated by the sequential convexifiability of facial disjunctive sets, see below).

In the special case of a disjunction of the form $x_j \in \{0, 1\}$, when $|Q| = 2$ and

$$P_{j0} := \{x \in \mathbb{R}^n_+ : Ax \geq b,\ x_j = 0\},$$
$$P_{j1} := \{x \in \mathbb{R}^n_+ : Ax \geq b,\ x_j = 1\},$$

$P_Q := \text{conv}(P_{j0} \cup P_{j1})$ is the set of those $x \in \mathbb{R}^n$ for which there exist vectors (y, y_0), $(z, z_0) \in \mathbb{R}^{n+1}_+$ such that

$$\begin{aligned} x - y - z &= 0 \\ Ay - by_0 &\geq 0 \\ -y_j &= 0 \\ Az - bz_0 &\geq 0 \\ z_j - z_0 &= 0 \\ y_0 + z_0 &= 1 \end{aligned} \tag{8′}$$

Unlike the general system (8), the system (8′), in which $|Q| = 2$, is of quite manageable size.

Projection and polarity

In order to generate the convex hull P_Q, and more generally, to obtain valid inequalities (cutting planes) in the space of the original variables, we project $\mathcal{P}$ onto the x-space:

Theorem 5.2 (Balas, 1998)**.** $\text{Proj}_x(\mathcal{P}) = \{x \in \mathbb{R}^n : \alpha x \geq \beta \text{ for all } (\alpha, \beta) \in W_0\}$, where

$$W_0 := \{(\alpha, \beta) \in \mathbb{R}^{n+1} : \alpha = u^i A^i,\ \beta \leq u^i b^i \text{ for some } u^i \geq 0,\ i \in Q\}.$$

Note that W_0 is not the standard projection cone of $\mathcal{P}$, introduced in Section 1, which is (assuming each A^i is $m_i \times n$)

$$W := \{(\alpha, \beta, \{u^i\}_{i \in Q}) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^{m_1} \times \cdots \times \mathbb{R}^{m_{|Q|}} : \alpha - u^i A^i = 0, \beta - u^i b^i \leq 0, u^i \geq 0, i \in Q\}.$$

Instead, $W_0 = \text{Proj}_{(\alpha,\beta)}(W)$, i.e. W_0 is the projection of W onto the (α, β)-space. In fact, W_0 can be shown to be the *reverse polar cone* P^*_Q of P_Q, i.e. the cone of all valid inequalities for P_Q.

Theorem 5.3 (Balas, 1998)**.**

$$P_Q^* := \{(\alpha, \beta) \in \mathbb{R}^{n+1} : \alpha x \geq \beta \text{ for all } x \in P_Q\}$$

$$= \{(\alpha, \beta) \in \mathbb{R}^{n+1} : \alpha = u^i A^i,\ \beta \leq u^i b^i \text{ for some } u^i \geq 0,\ i \in Q\}.$$

To turn again to the special case of a disjunction of the form $x_j \in \{0, 1\}$, projecting the system (6) onto the x-space yields the polyhedron P_Q whose reverse polar cone is

$$\begin{aligned} P_Q^* = \{(\alpha, \beta) \in \mathbb{R}^{n+1} : \alpha &\geq uA - u_0 e_j \\ \alpha &\geq vA + v_0 e_j \\ \beta &\leq ub \\ \beta &\leq vb + v_0 \\ u, v &\geq 0\} \end{aligned}$$

(where e_j is the j-th unit vector).

One of the main advantages of the higher dimensional representation is that in projecting it onto the x-space we have a straightforward criterion to distinguish facets of P_Q from other valid inequalities:

Theorem 5.4 (Balas, 1998)**.** Assume P_Q is full dimensional. The inequality $\alpha x \geq \beta$ defines a facet of P_Q if and only if (α, β) is an extreme ray of the cone P_Q^*.

Next we turn to the class of disjunctive programs that are sequentially convexifiable.

Sequential convexification

A disjunctive set is called *facial* if every inequality in (7) induces a face of P, the polyhedron defined by the inequalities $Ax \geq b$ common to all terms of the disjunction. Zero-one programs (pure or mixed) are facial disjunctive programs, general integer programs are not. *Sequential convexifiability* is one of the basic properties that distinguish pure or mixed 0-1 programs from general integer programs.

Theorem 5.5 (Balas, 1998)**.** Let

$$D := \left\{ x \in \mathbb{R}^n : Ax \geq b,\ \bigvee_{h \in Q_j} \left(d^h x \geq d_0^h\right),\ j = 1, \ldots, t \right\},$$

where $|Q_j| \geq 1$ for $j = 1, \ldots, t$, and D is facial. Let $P_D := \text{conv}(D)$.

Define

$$P^0 (= P) := \{x \in \mathbb{R}^n : Ax \geq b\},$$

and for $j = 1, \ldots, t$,

$$P^j := \text{conv}\left(P^{j-1} \cap \left\{x : \bigvee_{h \in Q_j} \left(d^h x \geq d_0^h\right)\right\}\right).$$

Then

$$P^t = P_D.$$

While faciality is a sufficient condition for sequential convexifiability, it is not necessary. A necessary and sufficient condition is given in Balas, Tama, and Tind (1989). The most important class of facial disjunctive programs are mixed 0-1 programs, and for that case Theorem 5.5. asserts that if we denote

$$P_D := \text{conv}\,\{x \in \mathbb{R}^n_+ : Ax \geq b,\ x_j \in \{0, 1\},\ j = 1, \ldots, p\},$$
$$P^0 := \{x \in \mathbb{R}^n_+ : Ax \geq b\},$$

and define recursively for $j = 1, \ldots, p$

$$P^j := \text{conv}\,(P^{j-1} \cap \{x : x_j \in \{0, 1\}\}),$$

then

$$P^p = P_D.$$

Thus, in principle, a 0-1 program with p 0-1 variables can be solved in p steps. Here each step consists of imposing the 0-1 condition on one new variable and generating all the inequalities that define the convex hull of the set defined in this way.

Note that while pure and mixed 0-1 programs are facial disjunctive programs and therefore are sequentially convexifiable, general integer programs—whether pure or mixed—are not.

Disjunctive rank

A useful concept in cutting plane theory is that of *rank* with respect to a cut generating procedure. If $Ax \geq b$ is the linear programming relaxation of some integer program, every valid inequality can be derived by (repeatedly) taking nonnegative combinations of $Ax \geq b$ and rounding down the resulting inequality, as in $\lfloor \lambda A \rfloor x \geq \lfloor \lambda b \rfloor$ with $\lambda \geq 0$. The minimum number of times this procedure has to be iterated in order to obtain a given valid inequality (or an inequality that dominates it) is known as the *Chvatal rank* of that inequality (see e.g., Nemhauser and Wolsey, 1986).

Based on the sequential convexifiability of facial disjunctive programs, one can define the *disjunctive rank* of an inequality $\alpha x \geq \beta$ for a mixed 0-1 program as the smallest integer k for which there exists an ordering $\{i_1, \ldots, i_p\}$ of $\{1, \ldots, p\}$ such that $\alpha x \geq \beta$ is valid for P^k. In other words, an inequality is of rank k if it can be obtained by k, but not by fewer than k, applications of the recursive procedure defined above. Clearly,

the disjunctive rank of a cutting plane for 0-1 programs is bounded by the number of 0-1 variables. It is known that the number of 0-1 variables is not a valid bound for the Chvatal rank of an inequality.

The above definition of the disjunctive rank is based on using the disjunctions $x_j \in \{0, 1\}$, $j = 1, \ldots, p$. Tighter bounds can be derived by using stronger disjunctions. For instance, a 0-1 program whose constraints include the generalized upper bounds $\sum(x_j : j \in Q_i) = 1$, $i = 1, \ldots, t$, with $|Q_i| = |Q_j| = q$, $Q_i \cap Q_j = \emptyset$, $i, j \in \{1, \ldots, t\}$, and $|\bigcup_{i=1}^{t} Q_i| = p$, can be solved as a disjunctive program with the disjunctions

$$\bigvee_{j \in Q_i} (x_j = 1), \quad i = 1, \ldots, t\,(= p/q),$$

in which case the disjunctive rank of any cut is bounded by the number $t = p/q$ of GUB constraints.

Another derivation of the basic results

The two basic ingredients of our approach, the lifting/projection technique and sequential convexification, can also be derived by the following procedure (Balas, Ceria, and Cornuéjols, 1993). Define

$$P := \{x \in \mathbb{R}^n : \tilde{A}x \geq \tilde{b}\} \subseteq \mathbb{R}^n$$

and

$$P_D := \text{conv}\,(\{x \in P : x_j \in \{0, 1\},\ j = 1, \ldots, p\}),$$

with the inequalities $x \geq 0$ and $x_j \leq 1$, $j = 1, \ldots, p$, included in $\tilde{A}x \geq \tilde{b}$.

1. Select an index $j \in \{1, \ldots, p\}$. Multiply $\tilde{A}x \geq \tilde{b}$ with $1 - x_j$ and x_j to obtain the nonlinear system

$$\begin{aligned} (1 - x_j)(\tilde{A}x - \tilde{b}) &\geq 0 \\ x_j(\tilde{A}x - \tilde{b}) &\geq 0. \end{aligned} \tag{9}$$

2. Linearize (9) by substituting y_i for $x_i x_j$, $i = 1, \ldots, n$, $i \neq j$, and x_j for x_j^2.
3. Project the resulting polyhedron onto the x-space.

Theorem 5.6 (Balas, Ceria, and Cornuéjols, 1993)**.** The outcome of steps 1, 2, 3 is

$$\text{conv}\,(P \cap \{x : x_j \in \{0, 1\}\}).$$

Corollary 5.7 (Balas, Ceria, and Cornuéjols, 1993)**.** Repeating steps 1, 2, 3 for each $j \in \{1, \ldots, p\}$ in turn yields P_D.

The fact that this procedure is isomorphic to the one introduced earlier can be seen by examining the outcome of step 2. In fact, the linearized system resulting from step 2 is precisely (8′), the higher dimensional representation of the disjunctive set defined by the constraint $x_j \in \{0, 1\}$ (see Balas, Ceria, and Cornuéjols, 1993 for details).

The above 3-step procedure is a streamlined version of the matrix cone procedure of Lovász and Schrijver (1991). The latter involves in step 1 multiplication with $1 - x_j$ and x_j for every $j \in \{1, \ldots, p\}$ rather than just one. While obtaining P_D by the Lovász-Schrijver procedure still involves p iterations of steps 1, 2, 3, the added computational cost brings a reward: after each iteration, the coefficient matrix of the linearized system must be positive semidefinite, a condition that can be used in various ways to derive strong bounds or cuts (see Lovász and Schrijver, 1991; Balas et al, 1996).

Another similar procedure, due to Sherali and Adams (1990), is based on multiplication with every product of the form $(\pi_{j\in J_1} x_j)(\pi_{j\in J_2}(1 - x_j))$, where J_1 and J_2 are disjoint subsets of $\{1, \ldots, p\}$ such that $|J_1 \cup J_2| = t$ for some $1 \leq t \leq p$. Linearizing the resulting nonlinear system leads to a higher dimensional polyhedron whose strength (tightness) is intermediate between P and P_D, depending on the choice of t: for $t = p$, the polyhedron becomes identical to $\mathcal{P}$ defined by the system (8) for a 0-1 polytope.

Disjunctive cuts

Theorem 5.3 describes the set of valid inequalities for a union of polyhedra P_i as defined in Theorem 5.1. If we replace this definition by $P_i := \{x \in \mathbb{R}^n_+ : A^i x \geq b^i\}$, then the valid inequalities for $P_Q = \text{conv}(\bigcup_{i\in Q} P_i)$ are of the form $\alpha x \geq \beta$, where $(\alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}$ satisfies

$$\alpha \geq u^i A^i, \quad \beta \leq u^i b^i, \ i \in Q \tag{10}$$

for some $u^i \geq 0$, $i \in Q$.

Clearly, every inequality satisfying (10) is dominated by one whose coefficients satisfy

$$\alpha_j = \max_{i\in Q} u^i a^i_j, \quad \beta \leq \min_{i\in Q} u^i b^i \tag{11}$$

where a^i_j is the j-th column of A^i, $i \in Q$.

Since (10)—or, alternatively, (11)—defines all valid inequalities for a disjunctive program, *every* valid cutting plane for a disjunctive program can be obtained from (11) by choosing suitable multipliers u^i, $i \in Q$. In other words, any cutting plane for a combinatorial optimization problem that can be stated as a disjunctive program, can be brought to the form (10) or (11). Clearly, the strength of such a cut depends on the choice of the multipliers u^i, $i \in Q$.

We will illustrate this on an example.

Example. Consider the mixed integer program whose constraint set is

$$\begin{aligned} x_1 &= 0.2 + 0.4(-x_3) + 1.3(-x_4) - 0.01(-x_5) + 0.07(-x_6) \\ x_2 &= 0.9 - 0.3(-x_3) + 0.4(-x_4) - 0.04(-x_5) + 0.1(-x_6) \\ x_j &\geq 0,\ j = 1, \ldots, 6,\ x_j \text{ integer},\ j = 1, \ldots, 4. \end{aligned}$$

This problem is taken from the paper Johnson (1974), which also lists six cutting planes derived from the associated group problem:

$$\begin{aligned} 0.75x_3 + 0.875x_4 + 0.0125x_5 + 0.35x_6 &\geq 1, \\ 0.778x_3 + 0.444x_4 + 0.40x_5 + 0.111x_6 &\geq 1, \\ 0.333x_3 + 0.667x_4 + 0.033x_5 + 0.35x_6 &\geq 1, \\ 0.50x_3 + x_4 + 0.40x_5 + 0.25x_6 &\geq 1, \\ 0.444x_3 + 0.333x_4 + 0.055x_5 + 0.478x_6 &\geq 1, \\ 0.394x_3 + 0.636x_4 + 0.346x_5 + 0.155x_6 &\geq 1. \end{aligned}$$

The first two of these inequalities are the mixed-integer Gomory cuts derived from the row of x_1 and x_2 respectively. To show how they can be improved, we first derive them as they are. To do this, for a row of the form

$$x_i = a_{i0} + \sum_{j \in J} a_{ij}(-x_j),$$

with x_j integer-constrained for $j \in J_1 := \{1, \ldots, 4\}$, continuous for $j \in J_2 := \{5, 6\}$, we define $f_{ij} = a_{ij} - [a_{ij}]$, $j \in J_1 \cup \{0\}$, $\varphi_{i0} = f_{i0}$, and

$$\varphi_{ij} = \begin{cases} f_{ij}, & j \in J_1^+ = \{j \in J_1 | f_{i0} \geq f_{ij}\}, \\ f_{ij} - 1, & j \in J_1^- = \{j \in J_1 | f_{i0} < f_{ij}\}, \\ a_{ij}, & j \in J_2. \end{cases}$$

Then every x which satisfies the above equation and the integrality constraints on x_i and x_j, $j \in J_1 \cup \{i\}$, also satisfies the condition

$$y_i = \varphi_{i0} + \sum_{i \in J} \varphi_{ij}(-x_j), \quad y_i \text{ integer}.$$

For the two equations of the example, the resulting conditions are

$$\begin{aligned} y_1 &= 0.2 - 0.6(-x_3) - 0.7(-x_4) - 0.01(-x_5) + 0.07(-x_6), \quad y_1 \text{ integer}, \\ y_2 &= 0.9 + 0.7(-x_3) + 0.4(-x_4) - 0.04(-x_5) + 0.1(-x_6), \quad y_2 \text{ integer}. \end{aligned}$$

Since each y_i is integer-constrained, they have to satisfy the disjunction $y_i \leq 0 \vee y_i \geq 1$. Substituting $\varphi_{i0} + \sum_{j \in J} \varphi_{ij}(-x_j)$ for each y_i and applying the formula (11)

with multipliers $u_0^i = 1/\varphi_{i0}$ in the first term, and $u_0^i = 1/(1-\varphi_{i0})$ in the second term of each disjunction we obtain for $i = 1$ and $i = 2$ the two cuts

$$\frac{0.6}{0.8}x_3 + \frac{0.7}{0.8}x_4 + \frac{0.01}{0.8}x_5 + \frac{0.07}{0.2}x_6 \geq 1,$$

and

$$\frac{0.7}{0.9}x_3 + \frac{0.4}{0.9}x_4 + \frac{0.04}{0.1}x_5 + \frac{0.1}{0.9}x_6 \geq 1.$$

These are precisely the first two inequalities of the above list. Since all cuts discussed here are stated in the form ≥ 1, the smaller the j-th coefficient, the stronger is the cut in the direction j. We would thus like to reduce the size of the coefficients as much as possible.

Now suppose that instead of $y_1 \leq 0 \vee y_1 \geq 1$, we use the disjunction

$$\{y_1 \leq 0\} \vee \left\{ \begin{array}{l} y_1 \geq 1 \\ x_1 \geq 0 \end{array} \right\},$$

which of course is also satisfied by every feasible x.

Then, applying formula (11) with multipliers 5, 5 and 15 for $y_1 \leq 0$, $y_1 \geq 1$ and $x_1 \geq 0$ respectively, we obtain the cut whose coefficients are

$$\max\left\{\frac{5\times(-0.6)}{5\times 0.2}, \frac{5\times 0.6 + 15\times(-0.4)}{5\times 0.8 + 15\times(-0.2)}\right\} = -3,$$

$$\max\left\{\frac{5\times(-0.7)}{5\times 0.2}, \frac{5\times 0.7 + 15\times(-1.3)}{5\times 0.8 + 15\times(-0.2)}\right\} = -3.5,$$

$$\max\left\{\frac{5\times(-0.01)}{5\times 0.2}, \frac{5\times 0.01 + 15\times 0.01}{5\times 0.8 + 15\times(-0.2)}\right\} = 0.2,$$

$$\max\left\{\frac{5\times 0.07}{5\times 0.2}, \frac{5\times(-0.07) + 15\times(-0.07)}{5\times 0.8 + 15\times(-0.2)}\right\} = 0.35$$

that is

$$-3x_3 - 3.5x_4 + 0.2x_5 + 0.35x_6 \geq 1.$$

The sum of coefficients on the left hand side has been reduced from 1.9875 to -5.95. This strengthening has been obtained by assigning a positive multiplier in (11) to the inequality $x_1 \geq 0$, which had a zero multiplier in the previous derivation.

Similarly, for the second cut, if instead of $y_2 \leq 0 \vee y_2 \geq 1$ we use the disjunction

$$\left\{ \begin{array}{l} y_2 \leq 0 \\ x_1 \geq 0 \end{array} \right\} \vee \{y_2 \geq 1\},$$

with multipliers 10, 40 and 10 for $y_2 \leq 0$, $x_1 \geq 0$ and $y_2 \geq 1$ respectively, we obtain the cut

$$-7x_2 - 4x_4 + 0.4x_5 - x_6 \geq 1.$$

Here the sum of left hand side coefficients has been reduced from 1.733 to -11.6. Again, the effect is obtained by assigning a positive multiplier to $x_1 \geq 0$, this time on the other side of the disjunction.

Exercise 5.1 Consider the example at the end of Section 5. Why is it that to improve the first cut, we introduce $x_1 \geq 0$ into the *second* term of the disjunction $(y_1 \leq 0) \vee (y_1 \geq 1)$, and to improve the second cut, we introduce $x_1 \geq 0$ into the *first* term of the disjunction $(y_2 \leq 0) \vee (y_2 \geq 1)$? Why not the other way? Give a necessary and/or sufficient condition for an inequality $x_i \geq 0$ (where x_i is a basic variable) to be usable in this role, i.e. to improve the cut from a given disjunction $y_1 \leq 0 \vee y_1 \geq 1$ when included in one of the terms of the disjunction.

6. Generating lift-and-project cuts

The implementation of the disjunctive programming approach into a practical 0-1 programming algorithm had to wait until the early '90s. It required not only the choice of a specific version of disjunctive cuts, but also a judicious combination of cutting with branching, made possible in turn by the discovery of an efficient procedure for lifting cuts generated in a subspace (for instance, at a node of the search tree) to be valid in the full space (i.e. throughout the search tree).

Deepest cuts

As mentioned earlier, if P_D is full dimensional, then facets of P_D correspond to extreme rays of the reverse polar cone P_D^*. To generate such extreme rays, for each 0-1 variable x_j that is fractional at the linear programming optimum, we solve a linear program over a normalized version of the cone P_D^* corresponding to the disjunction $x_j = 0 \vee x_j = 1$, with an objective function aimed at cutting off the linear programming optimum $\bar{x}$ by as much as possible. This "cut generating linear program" for the j-th variable is of the form

$$\begin{array}{rl} \min & \alpha\bar{x} - \beta \\ \text{s.t.} & \alpha - uA + u_0 e_j \geq 0 \\ & \alpha - vA - v_0 e_j \geq 0 \\ & -\beta + ub = 0 \\ & -\beta + vb + v_0 = 0 \\ & u, v \geq 0 \end{array} \qquad (\text{CGLP})_j$$

and (i) $\beta \in \{1, -1\}$, or (ii) $\sum_j |\alpha_j| \leq 1$. For details, see Balas, Ceria, and Cornuéjols (1993, 1996).

The normalization constraint (i) or (ii) has the purpose of turning the cone P_D^* into a polyhedron. Several other normalizations have been proposed later, with pro's and con's for each one, and their choice plays an important role in determining the optimum.

Solving $(\text{CGLP})_j$ yields a cut $\alpha x \geq \beta$, where

$$\alpha_k = \begin{cases} \max\{ua_k, va_k\} & k \in N \setminus \{j\} \\ \max\{ua_j - u_0, va_j + v_0\} & k = j, \end{cases} \tag{12}$$

with a_k the k-th column of A, and $\beta = \min\{ub, vb + v_0\}$. This cut maximizes the amount $\beta - \alpha\bar{x}$ by which $\bar{x}$ is cut off.

Cut lifting

A cutting plane derived at a node of the search tree defined by a subset $F_0 \cup F_1$ of the 0-1 variables, where F_0 and F_1 index those variables fixed at 0 and 1, respectively, is valid at that node and its descendants in the tree (where the variables in $F_0 \cup F_1$ remain fixed at their values). Such a cut can in principle be made valid at other nodes of the search tree, where the variables in $F_0 \cup F_1$ are no longer fixed, by calculating appropriate values for the coefficients of these variables—a procedure called lifting and mentioned in Section 1. However, calculating such coefficients is in general a daunting task, which may require the solution of an integer program for every coefficient. One important advantage of the cuts discussed here is that the multipliers u, u_0, v, v_0 obtained along with the cut vector (α, β) by solving $(\text{CGLP})_j$ can be used to calculate by closed form expressions the coefficients α_h of the variables $h \in F_0 \cup F_1$.

While this possibility of calculating efficiently the coefficients of variables absent from a given subproblem (i.e. fixed at certain values) is crucial for making it possible to generate cuts during a branch-and-bound process that are valid throughout the search tree, its significance goes well beyond this aspect. Indeed, most columns of A corresponding to nonbasic components of $\bar{x}$ typically play no role in determining the optimal solution of $(\text{CGLP})_j$ and can therefore be ignored. In other words, the cuts can be generated in a subspace involving only a subset of the variables, and then lifted to the full space. This is the procedure followed in Balas, Ceria, and Cornuéjols (1993, 1996), where the subspace used is that of the variables indexed by some $R \subset N$ such that R includes all the 0-1 variables that are fractional and all the continuous variables that are positive at the LP optimum. The lifting coefficients for the variables not in the subspace, which are all assumed w.l.o.g. to be at their lower bound, are then given by

$$\alpha_\ell := \max\{ua_\ell, va_\ell\}, \quad h \in N \setminus R$$

where u and v are the optimal vectors obtained by solving $(\text{CGLP})_j$.

These coefficients always yield a valid lifted inequality. If normalization (i) is used in $(\text{CGLP})_j$, the resulting lifted cut is exactly the same as the one that would have been obtained by applying $(\text{CGLP})_j$ to the problem in the full space. If other normalizations are used, the resulting cut may differ in some coefficients (lifting a cut does not in general have a unique outcome).

Cut strengthening

The cut $\alpha x \geq \beta$ derived from a disjunction of the form $x_j \in \{0, 1\}$ can be strengthened by using the integrality conditions on variables other than x_j, as shown in Balas and Jeroslow (1980) (see also Section 7 of Balas, 1979). Indeed, if x_k is such a variable, the coefficient

$$\alpha_k := \max\{ua_k, va_k\}$$

can be replaced by

$$\alpha'_k := \min\{ua_k + u_0\lceil m_k \rceil,\ va_k - v_0\lfloor m_k \rfloor\},$$

where

$$m_k := \frac{va_k - ua_k}{u_0 + v_0}.$$

For a proof of this statement, see Balas (1979) or Balas, Ceria, and Cornuéjols (1993). The strengthening "works," i.e. produces an actual change in the coefficient, if and only if either $ua_k > va_k + u_0$ or $va_k > ua_k + v_0$. Furthermore, the larger the difference $|ua_k - va_k|$, the more room there is for strengthening the coefficient in question.

This strengthening procedure can also be applied to cuts derived from disjunctions other than $x_j \in \{0, 1\}$, including disjunctions with more than two terms. In the latter case, however, the closed form expression for the values m_k used above has to be replaced by a procedure for calculating those values, whose complexity is linear in the number of terms in the disjunction (see Balas and Jeroslow, 1980 or Balas, 1979 for details).

The overall cut generating procedure

To summarize briefly the above discussion, the actual cut generating procedure is not just "lift and project," but rather RLPLS, an acronym for

- RESTRICT the problem to a subspace defined from the LP optimum, and choose a disjunction;
- LIFT the disjunctive set to describe its convex hull in a higher dimensional space;

- PROJECT the polyhedron describing the convex hull onto the original (restricted) space, generating cuts;
- LIFT the cuts into the original full space;
- STRENGTHEN the lifted cuts.

Exercise 6.1 Generalize the procedure described in the subsection "Deepest cuts" of Section 6 to the case where you want to use a stronger disjunction than $x_j = 0 \vee x_j = 1$, namely the one implied by imposing the 0-1 condition on two variables rather than just one: $x_i \in \{0, 1\}$, $x_j \in \{0, 1\}$:

(a) state the resulting (4-term) disjunction,

(b) formulate the corresponding $(\text{CGLP})_{ij}$,

(c) given an optimal solution to $(\text{CGLP})_{ij}$, state the associated cut and derive the expression corresponding to (12).

7. Solving the cut generating linear program in the (LP) simplex tableau

A breakthrough in generating lift-and-project cuts occurred in 2000, when a precise correspondence was established in Balas and Perregaard (2003) between feasible bases of the higher dimensional cut generating linear program (CGLP) and (not necessarily feasible) bases of the original linear programming relaxation (LP). This correspondence has made it possible to generate lift-and-project cuts corresponding to an optimal solution of the (CGLP) without explicitly constructing the latter, by working in the original (LP) tableau. In other words, the correspondence makes it possible to mimic the pivots of the (CGLP) through certain pivots of the (LP), and thereby solve the (CGLP) implicitly. Furthermore, as the (CGLP) is by construction highly degenerate, the correspondence between the bases of the two linear programs is many-to-one, i.e. many feasible bases of the (CGLP) correspond to a basis of the (LP). Thus solving the (CGLP) implicitly saves time not only because (LP) is a smaller linear program than (CGLP), but also because one pivot in the (LP) tableau corresponds to several pivots in the (CGLP) tableau.

This correspondence also gives an exact characterization of the connection between lift-and-project cuts and earlier cuts in the literature.

Consider the simplex tableau associated with the optimal solution $\bar{x}$ to (LP), and let

$$x_k = \bar{a}_{k_o} - \sum_{j \in J} \bar{a}_{k_j} s_j \tag{13}$$

be one of its rows, with x_k a 0-1 variable. Here J is the index set of nonbasic variables, $|J| = n$, and $0 < \bar{a}_{k0} < 1$. The simple disjunctive cut from the condition $x_k \leq 0 \vee x_k \geq 1$

applied to (13) is $\pi s_J \geq \pi_0$ where

$$\pi_0 = \bar{a}_{k0}(1 - \bar{a}_{k0}) \quad \text{and} \quad \pi_j = \max\{\bar{a}_{kj}(1 - \bar{a}_{k0}), -\bar{a}_{kj}\bar{a}_{k0}\} \quad j \in J.$$

If some of the variables s_j, say those indexed by $J_1 \subseteq J$, are integer-constrained, this cut can be strengthened to $\bar{\pi} s_J \geq \pi_0$ through the procedure described in the previous section, which in this case yields

$$\bar{\pi}_j = \begin{cases} \min\{(\bar{a}_{kj} - \lfloor \bar{a}_{kj} \rfloor)(1 - \bar{a}_{k0}), (\lceil \bar{a}_{kj} \rceil - \bar{a}_{kj})\bar{a}_{k0}\} & j \in J_1 \\ \pi_j & j \in J \setminus J_1 \end{cases}$$

This strengthened simple disjunctive cut is the same (Balas, 1979) as the mixed integer Gomory cut.

Now consider the cut generating linear program corresponding to variable x_k:

$$\begin{aligned} \min \quad & \alpha \bar{x} - \beta \\ & \alpha - u\tilde{A} + u_0 e_k = 0 \\ & \alpha - v\tilde{A} - v_0 e_k = 0 \\ & -\beta + u\tilde{b} = 0 \\ & -\beta + v\tilde{b} + v_0 = 0 \\ & ue + u_0 + ve + v_0 = 1 \\ & u, u_0, v, v_o \geq 0 \end{aligned} \qquad (\text{CGLP})_k$$

This formulation differs from the one used in the previous section by the fact that surplus variables have been introduced (which accounts for $\tilde{A}$, $\tilde{b}$ replacing A, b), and a different normalization is used (note that $\tilde{A}$, just like A, has n columns).

It can be shown that in any feasible basic solution to $(\text{CGLP})_k$ that yields an inequality not dominated by the constraints of (LP), u_0 and v_0 are both positive, and if the basic components of u and v are indexed by M_1 and M_2, respectively, then $M_1 \cap M_2 = \emptyset$, $|M_1 \cup M_2| = n$, and the $n \times n$ submatrix $\hat{A}$ of $\tilde{A}$ whose rows are indexed by $M_1 \cup M_2$ is nonsingular.

Now define $J := M_1 \cup M_2$ and consider the subsystem of $\tilde{A}x \geq \tilde{b}$ consisting of the constraints indexed by J, written in equality form, i.e. with the surplus variables explicitly shown:

$$\hat{A}x - s_J = \hat{b}$$

or

$$x = \hat{A}^{-1}\hat{b} + \hat{A}^{-1}s_J.$$

The equation corresponding to x_k (where $k \notin J$) is then of the form (13), with $\bar{a}_{k0} = e_k \hat{A}_k^{-1}\hat{b}$ and $\bar{a}_{kj} = -(\hat{A}^{-1})_{kj}$, and it can be shown that $0 < \bar{a}_{k0} < 1$.

The following two theorems (Balas and Perregaard, 2003) give the correspondence.

Theorem 7.1 (A). Let $\alpha x \geq \beta$ be the lift-and-project cut associated with a basic solution $(\alpha, \beta, u, u_0, v, v_0)$ to $(\text{CGLP})_k$, with $u_0, v_0 > 0$ and the basic components of u, v indexed by M_1, M_2, respectively.

Let $\pi s_J \geq \pi_0$ be the simple disjunctive cut from the disjunction $x_k \leq 0 \vee x_k \geq 1$ applied to (13) with $J := M_1 \cup M_2$. Then $\pi s_J \geq \pi_0$ is equivalent to $\alpha x \geq \beta$.

Theorem 7.1 (B). Let $\hat{A}$ be any $n \times n$ nonsingular submatrix of $\tilde{A}$, and $\hat{b}$ the corresponding subvector of $\tilde{b}$, such that $0 < e_k \hat{A}^{-1} \hat{b} < 1$, and let J be the row index set of $(\hat{A}, \hat{b})$.

Let $\pi s_J \geq \pi_0$ be the simple disjunctive cut obtained from the disjunction $x_k \leq 0 \vee x_k \geq 1$ applied to the expression of x_k in terms of the nonbasic variables indexed by J. Further, let (M_1, M_2) be any partition of J such that $j \in M_1$ if $\bar{a}_{kj} < 0$ and $j \in M_2$ if $\bar{a}_{kj} > 0$.

Let $\alpha x \geq \beta$ be the lift-and-project cut corresponding to the basic solution of $(\text{CGLP})_k$ in which $u_0, v_0 > 0$ and the basic components of u, v are indexed by M_1, M_2, respectively.

Then $\alpha x \geq \alpha_0$ is equivalent to $\pi s_j \geq \pi_0$.

Notice that the partition of J into M_1, M_2 is not unique: any j such that $\bar{a}_{kj} = 0$ may go either into M_1 or into M_2. This is what makes the correspondence between the bases many-to-one. However, the correspondence between the solutions, hence the cuts, is one-to-one.

The correspondence expressed in Theorems 7.1. A and B can be can be extended to the strengthened versions of these cuts. Indeed, these theorems remain valid if the inequalities $\alpha x \geq \beta$ and $\pi x \geq \pi_0$ are replaced by the strengthened lift-and-project cut, and the strengthened simple disjunctive cut or Gomory mixed integer cut, respectively.

Using the above correspondence, one can solve $(\text{CGLP})_k$, the cut generating linear program for variable x_k, implicitly, working only on the (LP) tableau. Every row of the LP tableau corresponds to a pair of columns of the $(\text{CGLP})_k$ tableau. To mimic the pivoting in the latter tableau we need to calculate its reduced costs. For every row i of the (LP) tableau, there are two columns (with variables u_i and v_i) of the $(\text{CGLP})_k$ tableau. If any of these two columns has a negative reduced cost, a pivot can be performed in row i of the LP tableau, corresponding to one or several objective-improving pivots in the $(\text{CGLP})_k$ tableau. For this purpose, an appropriate column of the (LP) tableau, say j, is identified, and the pivot is executed on entry $\bar{a}_{ij}$, resulting in a new (LP) tableau, generally neither optimal nor feasible. The procedure is then repeated, until there are no more negative reduced costs. At that point the solution to $(\text{CGLP})_k$ is optimal, and the simple disjunctive cut from row k of the (LP) tableau is equivalent to the optimal lift-and-project cut (see Balas and Perregaard (2003) for details of the algorithm).

Since the strengthened version of the simple disjunctive cut is the same as the mixed integer Gomory cut, the above procedure can be interpreted as an algorithm for strengthening (improving) a mixed integer Gomory (MIG) cut obtainable from a given row k of the optimal (LP) simplex tableau through a sequence of pivots in that tableau.

The first pivot replaces the source row k for the MIG cut with the source row k of the new tableau resulting from the pivot. This new tableau is in general infeasible, but its row k yields a MIG cut guaranteed to be stronger (more violated by the optimal LP solution $\bar{x}$) than the cut from the previous tableau would have been. Each subsequent pivot results in the replacement of the MIG cut from row k of the current tableau with a MIG cut from row k of a new tableau, with a guaranteed improvement of the cut.

8. Branch-and-cut and computational experience

No cutting plane approach known at this time can solve large, hard integer programs just by itself. Repeated cut generation tends to produce a flattening of the region of the polyhedron where the cuts are applied, as well as numerical instability which can only partly be mitigated by a tightening of the tolerance requirements. Therefore the successful use of cutting planes requires their combination with some enumerative scheme. One possibility is to generate cutting planes as long as that seems profitable, thereby creating a tighter linear programming relaxation than the one given originally, and then to solve the resulting problem by branch and bound. Another possibility, known as branch-and-cut, consists of branching and cutting intermittently; i.e., when the cut generating procedure "runs out of steam," move elsewhere in the feasible region by branching. This approach depends crucially on the ability to lift the cuts generated at different nodes of the search tree so as to make them valid everywhere.

The first successful implementation of lift-and-project for mixed 0-1 programming in a branch-and-cut framework was the MIPO (Mixed Integer Program Optimizer) code described in Balas, Ceria, and Cornuéjols (1996). The procedure it implements can be outlined as follows.

Nodes of the search tree (subproblems created by branching) are stored along with their optimal LP bases and associated bounds. Cuts that are generated are stored in a pool. The cut generation itself involves the RLPLS process described earlier. Furthermore, cuts are not generated at every node, but at every k-th node, where k is a cutting frequency parameter.

At any given iteration, a subproblem with best (weakest) lower bound is retrieved from storage and its optimal LP solution $\bar{x}$ is recreated. Next, all those cuts in the pool that are tight for $\bar{x}$ or violated by it, are added to the constraints and $\bar{x}$ is updated by reoptimizing the LP.

At this point, a choice is made between generating cuts or skipping that step and going instead to branching. The frequency of cutting is dictated by the parameter k that is problem dependent, and calculated after generating cuts at the root node. Its value is a function of several variables believed to characterize the usefulness of cutting planes for the given problem, primarily the average depth of the cuts obtained. In our experiments, the most frequently used value of k was 8.

If cuts are to be generated, this happens according to the RLPLS scheme described above. First, a subspace is chosen by retaining all the 0-1 variables fractional and all the

continuous variables positive at the LP optimum, and removing the remaining variables along with their lower and upper bounds. A lift and project cut is then generated from each disjunction $x_j \in \{0, 1\}$ for j such that $0 < \bar{x}_j < 1$ (this is called a round of cuts). Each cut is lifted and strengthened; and if it differs from earlier cuts sufficiently (the difference between cuts is measured by the angle between their normals), it is added to the pool; otherwise it is thrown away. After generating a round of cuts, the current LP is reoptimized again.

If cuts are not to be generated (or have already been generated), a fractional variable is chosen for branching on a disjunction of the form $x_j \in \{0, 1\}$; i.e., two new subproblems are created, their lower bounds are calculated, and they are stored. The branching variable is chosen as the one with largest (in absolute value) cost coefficient among those whose LP optimal value is closest to 0.5.

Computational results in branch-and-cut mode

The above described procedure was implemented in the code MIPO, described in detail in Balas, Ceria, and Cornuéjols (1996). This implementation of MIPO does not have its own linear programming routine; instead, it calls a simplex code whenever it has to solve or reoptimize an LP. In the experiments of Balas, Ceria, and Cornuéjols (1996) the LP solver used was that of CPLEX 2.1. The test bed consisted of 29 test problems from MIPLIB and other sources, ranging in size from about 30 to 9,000 0-1 variables, 0 to 500 continuous variables, and about 20 to 2,000 constraints. The large majority of these problems have a real world origin; they were contributed mostly by people who tried, not always successfully, to solve them. Of the MIPLIB problems, most of those not included into the testbed were omitted as too easily solved by straight branch-and-bound; two problems were excluded because their LP relaxation exceeded the dimensioning of the experiment. MIPO was compared with MINTO, OSL and CPLEXMIP 2.1 (the most advanced version available at the time of the experiments). The outcome (see Balas, Ceria, and Cornuéjols, 1996 for detailed results) is best summarized by showing the number of times a code ranked first, and second, both in terms of search tree nodes and in terms of computing time. This is done in Table 1.

Table 1

OSL		CPLEX		MINTO		MIPO	
First	Second	First	Second	First	Second	First	Second
Ranking by number of search tree nodes							
15	2	0	4	4	10	11	10
Ranking by CPU time							
2	3	6	6	10	3	12	14

In a sense, MIPO seemed to be the most robust among the four codes: it was the only one that managed to solve all 29 test problems, and it ranked first or second in computing time on 26 out of the 29 instances.

Other experiments with lift-and-project in an enumerative framework are reported on in Thienel (1995), where S. Thienel compares the performance of ABACUS, an object oriented branch-and-cut code, in two different modes of operation, one using lift-and-project cuts and the other using Gomory cuts; with the outcome that the version with lift-and-project cuts is considerably faster on all hard problems, where hard means requiring at least 10 minutes.

Little experimentation has taken place so far with cuts derived from stronger disjunctions than the 0-1 condition on a single variable. In Balas et al (1996) the MIPO procedure was run on maximum clique problems, where the higher dimensional formulation used to generate cuts was the one obtained by multiplying the constraint set with inequalities of the form $1 - \sum(x_j : j \in S) \geq 0$, $x_j \geq 0$, $j \in S$, where S is a stable set. This is the same as the higher dimensional formulation derived from the disjunction

$$(x_j = 0,\ j \in S) \vee \left(x_{j_1} = 1,\ x_j = 0,\ j \in S \backslash \{j_1\}\right) \\ \vee \cdots \vee \left(x_{j_s} = 1,\ x_j = 0,\ j \in S \backslash \{j_s\}\right),$$

where $s = |S|$. As this disjunction is more powerful than the standard one, the cuts obtained were stronger; but they were also more expensive to generate, and without some specialized code to solve the highly structured cut generating LP's of this formulation, the trade-off between the strength of the cuts and the cost of generating them favored the weaker cuts from the standard disjunction.

Results in cut-and-branch mode

Bixby et al. (1999) report on their computational experience with a parallel branch-and-bound code, run on 51 test problems after generating several types of cuts at the root node. One of the cut types used was disjunctive or lift-and-project cuts, generated essentially as in Balas, Ceria, and Cornuéjols (1996) with normalization (ii), but without restriction to a subspace and without strengthening. Since deriving these cuts in the full space is expensive, the routine generating them was activated only for 4 of the hardest problems. Their addition to the problem constraints reduced the integrality gap by 58.7%, 94.4%, 99.9% and 94.4%, respectively.

In Ceria and Pataki (1998), computational results are reported with a disjunctive cut generator used in tandem with the CPLEX branch and bound code. Namely, the cut generator was used to produce 2 and 5 rounds of cuts from the 0-1 disjunctions for the 50 most promising variables fractional at the LP optimum, after which the resulting problem with the tightened LP relaxation was solved by the CPLEX 5.0 MIP code. This "cut-and-branch" procedure was tested on 18 of the hardest MIPLIB problems and the results were compared to those obtained by using CPLEX 5.0 without the cut generator.

The outcome of the comparison can be summarized as follows (see Table 2 of Ceria and Pataki (1998) for details).

- The total running time of the cut-and-branch procedure was less than the time without cuts for 14 of the 18 problems; while the opposite happened for the remaining 4 problems.
- Two of the problems solved by cut-and-branch in 8 and 3 minutes respectively could not be solved by CPLEX alone in 20 hours.
- For 6 problems the gain in time was more than 4-fold.

Very good results were obtained on two difficult problems outside the above set. One of them, *set1ch*, could not be solved by CPLEX alone, which after exhausting its memory limitations stopped with a solution about 15% away from the optimum. On the other hand, running the cutting plane generator for 10 rounds on this problem produced a lower bound within 1.4% of the integer optimum, and running CPLEX on the resulting tightened formulation solved the problem to optimality in 28 seconds.

The second difficult problem, *seymour*, was formulated by Paul Seymour in an attempt to find a minimal irreducible configuration in the proof of the four color theorem. It was not solved to optimality until very recently. The value of the LP relaxation is 403.84, and an integer solution of 423.00 was known. The best previously known lower bound of 412.76 had been obtained by running a parallel computer with 16 processors for about 60 hours, using about 1400 megabytes of memory. The cut-and-branch procedure applied to this case generated 10 rounds of 50 cuts in about 10.5 hours, and produced a lower bound of 413.16, using less than 50 megabytes of memory. Running CPLEX for another 10 hours on this tightened formulation then raised the bound to 414.20. More recently, using the same lift-and-project cuts but a more powerful computer, the problem was solved to optimality (Pataki, 2001).

Results when generating lift-and-project cuts from the LP simplex tableau

Early testing on two dozen MIPLIB instances (Balas and Perregaard, 2003) showed that solving the (CGLP) implicitly on the (LP) simplex tableau required 7–8 times fewer pivots than doing it directly. A much more thorough testing was performed subsequently by M. Perregaard and reported on at the August 2003 International Symposium on Mathematical Programming held in Copenhagen (Perregaard, 2003). Out of 98 test problems originating in MIPLIB and Hans Mittelmann's database, 95 were solved within a time limit of 30 minutes, using both approaches, with cuts generated only at the root node. The average run time per instance was 596 seconds with the original version which solves the CGLP explicitly, and 451 seconds with the new version which solves it implicitly, working in the LP simplex tableau. The number of pivots for generating cuts was reduced 15–20 times. With the new version, the time spent on cut generation was on the order of 5% of the total time to solve the problem by branch and bound.

On the other hand, this experiment also showed that in about 1/4 of all this instances (27 out of 98) pivoting did not improve the cuts, i.e. the MIG cuts turned out to be optimal lift-and-project cuts. In the remaining cases, the improvements obtained came from the first 8-10 pivots. Over the entire problem set, MIG cuts on the average closed 35.3% of the integrality gap, while lift-and-project cuts closed 38.8%. However, for some hard problems the difference was much more substantial.

Starting with the 2003C release of XPRESS, the lift-and-project cuts generated from the (LP) simplex tableau are part of the mixed integer solver. The cut generating module uses at most 10 improving pivots per cut, and generates at most 50 lift-and-project cuts per round. The number of rounds is optional, with the understanding that heavy use of cuts is only recommended on hard problems on which the standard procedure fails.

References

Balas, E. (1998). "Disjunctive Programming: Properties of the Convex Hull of Feasible Points." Invited paper, with a foreword by G. Cornuéjols and W. Pulleyblank, *Discrete Applied Mathematics* 89, 3–44. (Originally MSRR# 348, Carnegie Mellon University, July 1974).

Balas, E. (1979). "Disjunctive Programming." *Annals of Discrete Mathematics* 5, 3–51.

Balas, E. (1985). "Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems." *SIAM Journal on Algebraic and Discrete Methods* 6, 466–485.

Balas, E. (1987). "The Assignable Subgraph Polytope of a Directed Graph." *Congressus Numerantium* 60, 35–42.

Balas, E. (1997). "A Modified Lift-and-Project Procedure." *Mathematical Programming* 79, 19–31.

Balas, E. (1998). "Projection with a Minimal System of Inequalities." *Computational Optimization and Applications* 10, 189–193.

Balas, E. (2001). "Projection and Lifting in Combinatorial Optimization." In M. Jünger and D. Naddef (eds.), *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions,* LNCS 2241, Springer, pp. 26–56.

Balas, E., S. Ceria, and G. Cornuéjols. (1993). "A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs." *Mathematical Programming* 58, 295–324.

Balas, E., S. Ceria, and G. Cornuéjols. (1996). "Mixed 0-1 Programming by Lift-and-Project in a Branch-and-Cut Framework." *Management Science* 42, 1229–1246.

Balas, E., S. Ceria, G. Cornuéjols, and G. Pataki. (1996). "Polyhedral Methods for the Maximum Clique Problem." In D. Johnson and M. Trick (eds.), *Clique, Coloring and Satisfiability: The Second DIMACS Challenge.* The American Mathematical Society, Providence, RI, pp. 11–27.

Balas, E. and R.G. Jeroslow. (1980). "Strengthening Cuts for Mixed Integer Programs." *European Journal of Operational Research* 4, 224–234.

Balas, E. and J.B. Mazzola. (1984). "Nonlinear 0-1 Programming: I. Linearization Techniques, II. Dominance Relations and Algorithms." *Mathematical Programming* 30, 1–21 and 22–45.

Balas, E. and M. Oosten. (1998). "On the Dimension of Projected Polyhedra." *Discrete Applied Mathematics* 87, 1–9.

Balas, E. and M. Oosten. (2000). "The Cycle Polytope of a Directed Graph." *Networks* 36, 34–46.

Balas, E. and M. Perregaard. (2003). "A Precise Correspondence Between Lift-and-Project Cuts, Simple Disjunctive Cuts, and Mixed Integer Gomory Cuts for 0-1 Programming." *Mathematical Programming B* 94, 221–245.

Balas, E. and W.R. Pulleyblank. (1983). "The Perfectly Matchable Subgraph Polytope of a Bipartite Graph." *Networks* 13, 495–518.

Balas, E. and W.R. Pulleyblank. (1989). "The Perfectly Matchable Subgraph Polytope of an Arbitrary Graph." *Combinatorica* 9, 321–337.

Balas, E., J. Tama and J. Tind. (1989). "Sequential Convexification in Reverse Convex and Disjunctive Programming." *Mathematical Programming* 44, 337–350.

Beaumont, N. (1990). "An Algorithm for Disjunctive Programming." *European Journal of Operational Research* 48, 362–371.

Bixby, R., W. Cook, A. Cox, and E. Lee. (1999). "Computational Experience with Parallel Mixed Integer Programming in a Distributed Environment." *Annals of Operations Research* 90, 19–45.

Blair, C. (1976). "Two Rules for Deducing Valid Inequalities for 0-1 Programs." *SIAM Journal of Applied Mathematics* 31, 614–617.

Blair, C. (1980). "Facial Disjunctive Programs and Sequences of Cutting Planes." *Discrete Applied Mathematics* 2, 173–180.

Ceria, S. and G. Pataki. (1998). "Solving Integer and Disjunctive Programs by Lift-and-Project." In R.E. Bixby, E.A. Boyd, and R.Z. Rios-Mercado (eds.), *IPCO VI, Lecture Notes in Computer Science, 1412*. Springer, pp. 271–283.

Ceria, S. and J. Soares. (1997). "Disjunctive Cuts for Mixed 0-1 Programming: Duality and Lifting." GSB, Columbia University.

Ceria, S. and J. Soares. (1999). "Convex Programming for Disjunctive Convex Optimization." *Mathematical Programming* 86, 595–614.

Dantzig, G.B., R.D. Fulkerson, and S.M. Johnson. (1954). "Solution of a Large-Scale Traveling Salesman Problem." *Operations Research* 2, 393–410.

Fortet, R. (1960). "Applications de l'algèbre de Boole en recherche opérationnelle." *Revue Francaise de Recherche Opérationnelle* 4, 17–25.

Jeroslow, R.G. (1977). "Cutting Plane Theory: Disjunctive Methods." *Annals of Discrete Mathematics* 1, 293–330.

Jeroslow, R.G. (1987). "Representability in Mixed Integer Programming I: Characterization Results." *Discrete Applied Mathematics* 17, 223–243.

Jeroslow, R.G. (1989). "Logic Based Decision Support: Mixed Integer Model Formulation." *Annals of Discrete Mathematics* 40.

Johnson, E.L. (1974). "The Group Problem for Mixed-Integer Programming." *Mathematical Programming Study* 2, 137–179.

Letchford, A. (2001). "On Disjunctive Cuts for Combinatorial Optimization." *Journal of Combinatorial Optimization* 5, 299–317.

Lovász, L. and A. Schrijver. (1991). "Cones of Matrices and Set Functions and 0-1 Optimization." *SIAM Journal of Optimization* 1, 166–190.

Miller, C.E., A.W. Tucker, and R.A. Zemlin. (1960). "Integer Programming Formulations and Traveling Salesman Problems." *Journal of the ACM* 7, 326–329.

Nemhauser, G. and L. Wolsey. (1986). *Integer and Combinatorial Optimization*. Wiley.

Padberg, M. and T.-Y. Sung. (1991). "An Analytical Comparison of Different Formulations of the Traveling Salesman Problem." *Math. Programming* 52, 315–357.

Pataki, G. (2001). Personal communication, March.

Perregaard, M. (2003). "A Practical Implementation of Lift-and-Project Cuts. A computational exploration of lift-and-project with XPRESS-MP." Paper presented at the International Symposium on Mathematical Programming, August, Copenhagen.

Perregaard, M. and E. Balas. (2001). "Generating Cuts from Multiple-Term Disjunctions." In K. Aardal and B. Gerards (eds.), *Integer Programming and Combinatorial Optimization* (8th IPCO Conference, Utrecht, 2001), LCNS No. 2081, Springer, pp. 318–360.

Sherali, H. and W. Adams. (1990). "A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems." *SIAM Journal on Discrete Mathematics* 3, 411–430.

Sherali, H. and C. Shetty. (1980). *Optimization with Disjunctive Constraints.* Lecture Notes in Economics and Mathematical Systems, 181, Springer.

Stoer, J. and C. Witzgall. (1970). *Convexity and Optimization in Finite Dimensions I.* Springer.

Stubbs, R.A. and S. Mehrotra. (1996). "A Branch-and-Cut Method for 0-1 Mixed Convex Programming." Department of Industrial Engineering, Northwestern University.

Thienel, S. (1995). "ABACUS: A Branch-and-Cut System." Doctoral Dissertation, Faculty of Mathematics and The Natural Sciences, University of Cologne.

Turkay, M. and I.E. Grossmann. (1996). "Disjunctive Programming Techniques for the Optimization of Process Systems with Discontinuous Investment Costs-Multiple Size Regions." *Industrial Engineering Chemical Research* 35, 2611–2623.

Williams, H.P. (1994). "An Alternative Explanation of Disjunctive Formulations." *European Journal of Operational Research* 72, 200–203.