Integer and Combinatorial Optimization: The Cut Generation LP

John E. Mitchell

Department of Mathematical Sciences
RPI, Troy, NY 12180 USA

March 2019
General disjunctions

Let

\[ Q_j = \{ x \in \mathbb{R}^n : A^i x \leq b^i \} \quad \text{for } j \in 1, \ldots, q. \]

Note that any bounds on \( x \) are included in the constraints \( A^i x \leq b^i \), for notational purposes.

Our feasible region is

\[ P := \bigcup_{j=1,\ldots,q} Q_j, \]

the union of polyhedra.

For example, we could look at a disjunction on a binary variable.

More generally, we could look at different \( Q^i \) corresponding to different allocations to subsets of the variables.
Convex hull

The convex hull of $P$ can be written in a lifted space as

$$
\text{conv}(P) = \left\{ x \in \mathbb{R}^n : \begin{array}{l}
    x - \sum_{j=1}^{q} z^j = 0 \\
    A^j z^j - b^j z^j_0 \leq 0 \text{ for } j = 1, \ldots, q \\
    z^j_0 \geq 0 \text{ for } j = 1, \ldots, q \\
    \sum_{j=1}^{q} z^j_0 = 1
\end{array} \right\}
$$

for some $z^j \in \mathbb{R}^n$, $z^j_0 \in \mathbb{R}$, for $j = 1, \ldots, q$.

The $z^j_0$ variables are the weights in the convex combination for each set $Q_j$.

The points $\frac{1}{z^j_0} z^j$ are feasible in each $Q^j$ for positive $z^j_0$. If $z^j_0 = 0$ and $z^j \neq 0$ then $z^j$ is a ray of $Q^j$.

This leads to a general version of lift-and-project.
Finding a disjunctive cut

Disjunctive cuts

In this handout, we focus on extending the cut generation LP to find violated constraints.
Assume we have a point $\bar{x} \not\in \text{conv}(P)$. We want to find a valid cutting plane $\pi^T x \leq \pi_0$, valid for each $x \in \bigcup Q^j$ but violated by $\bar{x}$. This constraint must satisfy

$$\pi \leq (A^j)^T \lambda^j, \quad \pi_0 \geq (b^j)^T \lambda^j, \quad \lambda^j \geq 0, \quad \text{for } j = 1, \ldots, q.$$ 

We also need a normalization constraint, perhaps

$$\sum_{j=1}^{q} (e^j)^T \lambda^j = 1,$$

where $e^j$ is the vector of ones, with dimension equal to the number of rows in $A^j$. 
Cut generation LP

This leads to the general form of the cut generation LP:

\[
\begin{align*}
\max_{\pi, \pi_0, \lambda^j} & \quad \pi^T \bar{x} - \pi_0 \\
\text{subject to} & \quad \pi \leq (A^j)^T \lambda^j \quad \text{for } j = 1, \ldots, q \\
& \quad \pi_0 \geq (b^j)^T \lambda^j \quad \text{for } j = 1, \ldots, q \\
& \quad \lambda^j \geq 0 \quad \text{for } j = 1, \ldots, q \\
& \quad \sum_{j=1}^q (e^j)^T \lambda^j = 1 \quad \text{for } j = 1, \ldots, q
\end{align*}
\]
When we have a simple disjunction

\[ (g^T x \leq h_0) \lor (g^T x \geq h_1) \]

we can set up the cut generation LP as in the following AMPL code.
Setting up the parameters and variables

param n;
param m;
param A{1..m, 1..n}; # the matrix A has m rows and n columns
param b{1..m};

param g{1..n}; # the disjunction is
param h{0..1}; # g^Tx <= h0 V g^Tx >= h1

param xbar{1..n} >= 0;

var pi{1..n};
var pi0; # the cut is pi*x >= pi0

var lambda{0..1,1..m} >= 0; # multipliers for Ax <= b
var rho{0..1,1..n} >= 0; # multipliers for x >= 0
var mu{0..1} >= 0 ; # multipliers for the disjunction
maximize objective: \( \sum_{j \in 1..n} x_{\text{bar}}[j] \times \pi[j] - \pi_0; \)

subject to \( \text{inpolar} \{j \in 0..1, i \in 1..n \}: \)
\( \pi[i] \leq \sum_{k \in 1..m} A[k,i] \times \lambda[j,k] - \rho[j,i] \)
\( \quad + (-1)^{\ast j} \ast g[i] \ast \mu[j]; \)

# need to get pi as a linear combination of the
# the constraints, for each part of the disjunction
 Remaining constraints

subject to find \( \pi_0 \{ j \in 0..1 \} \):
\[
\pi_0 \geq \sum_{k \in 1..m} b[k] \cdot \lambda[j,k] + (-1)^j \cdot h[j] \cdot \mu[j];
\]

# need to ensure \( \pi_0 \) is small enough so that the constraint is valid

subject to normalize:
\[
\sum_{i \in 0..1, \ j \in 1..m} \lambda[i,j] + \sum_{i \in 0..1, \ j \in 1..n} \rho[i,j] + \sum_{i \in 0..1} \mu[i] = 1;
\]

# need to bound the set of feasible solutions,
# to ensure a finite solution
Data set

Taking the following data set gives the problem considered in the previous handout:

```plaintext
# the constraints are
# 2x1 - 2x2 <= 1
# x1    <= 1
# x2    <= 1
#
# the disjunction is
# x1 <= 0 V x1 >= 1
#
param n:=2;
param m:=3;
param A : 1 2 :=
         1 2 -2
         2 1 0
         3 0 1
;
param b := 1 1 2 1 3 1;
param g := 1 1 2 0;
param h := 0 0 1 1;
param xbar := 1 0.5 2 0;
```
Output

ampl: reset; model disjunction.mod; data disjunction.dat; solve;
CPLEX 12.8.0.0: optimal solution; objective 0.1666666667
6 dual simplex iterations (4 in phase I)
ampl: display pi, pi0;
pi [*] :=
1    0.333333
2   -0.666667
;
pi0 = 0
Output

ampl: display lambda, mu, rho;

lambda :=
0 1 0
0 2 0
0 3 0
1 1 0.333333
1 2 0
1 3 0
;

mu [*] :=
0 0.333333
1 0.333333
;

rho :=
0 1 0
0 2 0
1 1 0
1 2 0
;
Modified problem

Adding the extra constraint $19x_1 + 10x_2 \leq 10$ gives the data file

```plaintext
# the constraints are
# 2x1 - 2x2 <= 1
# 19x1 + 10x2 <= 10
# x1     <= 1
# x2     <= 1
# the disjunction is
# x1 <= 0 V x1 >= 1
param n:=2;
param m:=4;
param A : 1 2 :=
    1  2 -2
    2 19 10
    3  1  0
    4  0  1
;
param b := 1 1 2 10 3 1 4 1;
param g := 1 1 2 0;
param h := 0 0 1 1;
param xbar := 1 0.5 2 0;
```
A simple disjunction

Output

CPLEX 12.8.0.0: optimal solution; objective 0.225
6 dual simplex iterations (4 in phase I)
ampl: display pi, pi0;
pi [*] :=
1 0.45
2 0
;

pi0 = 0
Check the validity of the constraint

The only other nonzero variables are:

multipliers for the disjunction (0.45 for \( x_1 \leq 0 \) and 0.5 for \( -x_1 \leq -1 \))

and the multiplier \( \lambda = 0.05 \) for \( 19x_1 + 10x_2 \leq 10 \) in the \( -x_1 \leq -1 \) side of the disjunction.

Check:
\[
0.05(19x_1 + 10x_2 \leq 10) + 0.5(-x_1 \leq -1) = (0.45x_1 + 0.5x_2 \leq 0).
\]