Integer Programming
ISE 418

Lecture 17

Dr. Ted Ralphs

Reading for This Lecture

- Wolsey, Chapters 10 and 11
- Nemhauser and Wolsey Sections II.3.1, II.3.6, II.3.7, II.5.4
- CCZ Chapter 8
- "Decomposition in Integer Programming," Ralphs and Galati.
- "Selected Topics in Column Generation," Lübbecke and Desrosiers

The Decomposition Bound

By exploiting our knowledge of $\mathrm{conv}(\mathcal{S}_R)$, we wish to compute the so-called *decomposition bound*.

$$z_{\mathrm{D}} \quad = \quad \max_{x \in \mathrm{conv}(\mathcal{S}_R)} \left\{ c^\top x \mid A''x \geq b'' \right\}$$

$$z_{\mathrm{IP}} \leq z_{\mathrm{D}} \leq z_{\mathrm{LP}}$$

This can be done using three different basic approaches:

- Lagrangian relaxation (dynamic generation of extreme points of $\mathrm{conv}(\mathcal{S}_R)$)
- Dantzig-Wolfe decomposition (dynamic generation of extreme points of $\mathrm{conv}(\mathcal{S}_R)$)
- Cutting plane method (dynamic generation of facets of $\mathrm{conv}(\mathcal{S}_R)$).

Lagrangian Relaxation

- Suppose as before that our IP is defined by

$$\begin{aligned} \max\ & c^\top x \\ \text{s.t. } A'x &\leq b' \text{ (the ``nice'' constraints)} \\ A''x &\leq b'' \text{ (the ``complicating'' constraints)} \\ x &\in \mathbb{Z}^n \end{aligned}$$

  where optimizing over $\mathcal{S}_R = \{x \in \mathbb{Z}^n \mid A'x \leq b'\}$ is "easy."

- Lagrangian Relaxation (for $u \geq 0$):

$$LR(u) : z_{LR}(u) = ub'' + \max_{x \in \mathcal{S}_R} \{(c^\top - uA'')x\}.$$

The Lagrangian Dual

- The next step is to obtain a dual problem formed by allowing u to vary.
- We are looking for the value of $u \geq 0$ that yield the lowest upper bound.
- The Lagrangian dual problem, LD, is

$$z_{LD} = \min_{u \geq 0} z_{LR}(u)$$

- The Lagrangian dual can be rewritten as the following LP

$$z_{LD} = \min_{\eta, u} \{\eta + ub'' \mid \eta \geq (c^\top - uA'')s, s \in \mathcal{E}, u \geq 0\}$$

- This can be solved using a cutting plane algorithm where the separation problem is an optimization problem over the set $\text{conv}(\mathcal{S}_R)$.

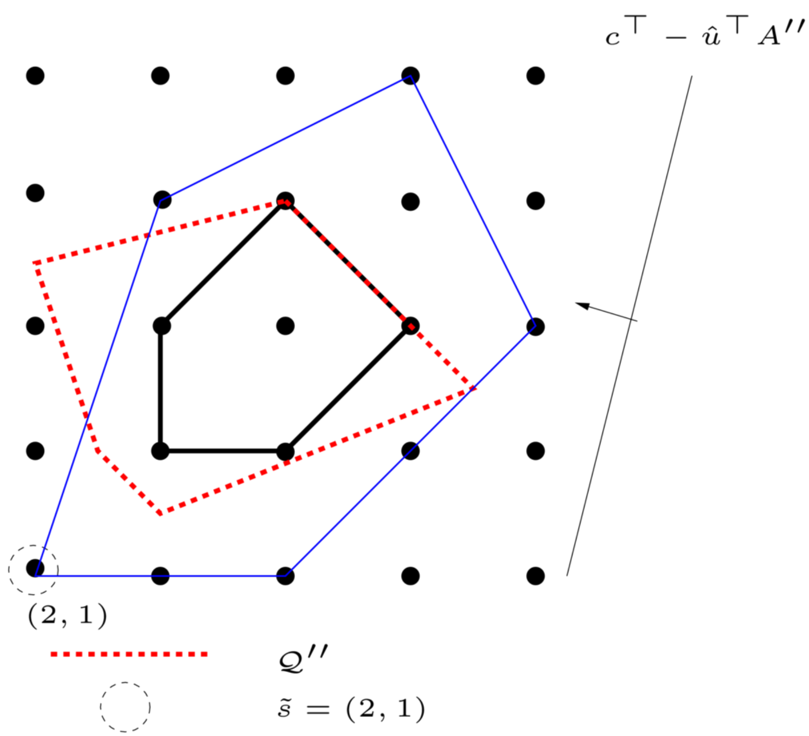
Solving the Lagrangian Dual with Subgradient Optimization

- Note that $(c^\top - uA'')x$ is an affine function of u for a fixed x.
- This tells us that $z_{LR}(u)$, when viewed as a function of u, is the maximum of a finite number of affine functions.
- Hence, it is piecewise linear and convex on the domain over which it is finite.
- We can easily minimize any convex function which we can evaluate and subdifferentiate using a technique called *subgradient optimization*.
- The procedure iteratively adjusts the weights according to the degree of violation of each constraint.
- There are a wide range of implementations of this basic idea.

Geometry of the Lagrangian Dual

LD iteratively produces single extreme points of $\mathrm{conv}(\mathcal{S}_R)$ and uses the violation of the relaxed constraints to adjust the dual solution.

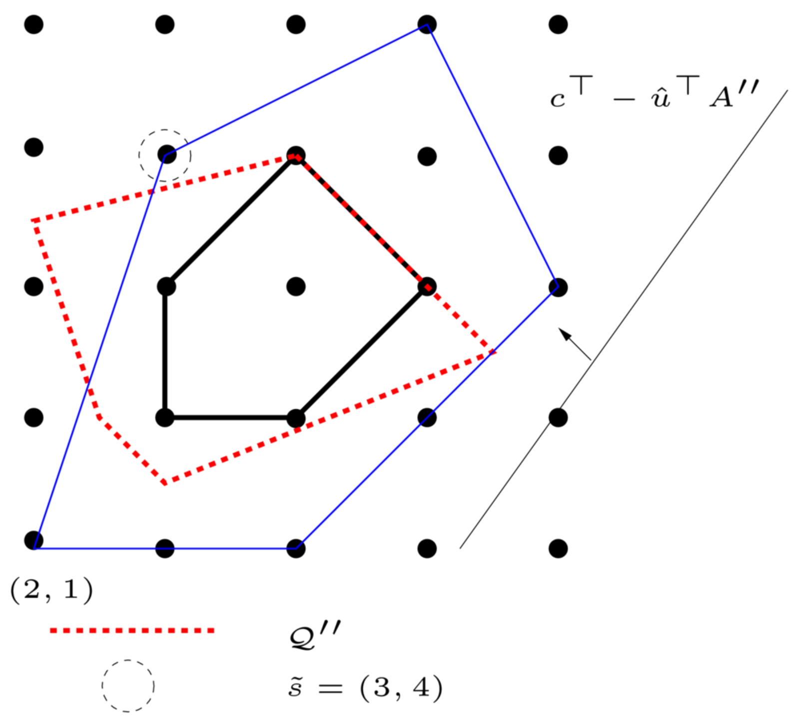
- **Master**: $z_{\mathrm{LD}} = \min_{u \in \mathcal{R}_+^{m''}} \left\{ \max_{s \in \mathcal{E}} \left\{ c^\top s + u^\top (b'' - A'' s) \right\} \right\}$
- **Subproblem**: $\mathrm{LR}(c^\top - u^\top A'')$



Geometry of the Lagrangian Dual

LD iteratively produces single extreme points of $\mathrm{conv}(\mathcal{S}_R)$ and uses the violation of the relaxed constraints to adjust the dual solution.

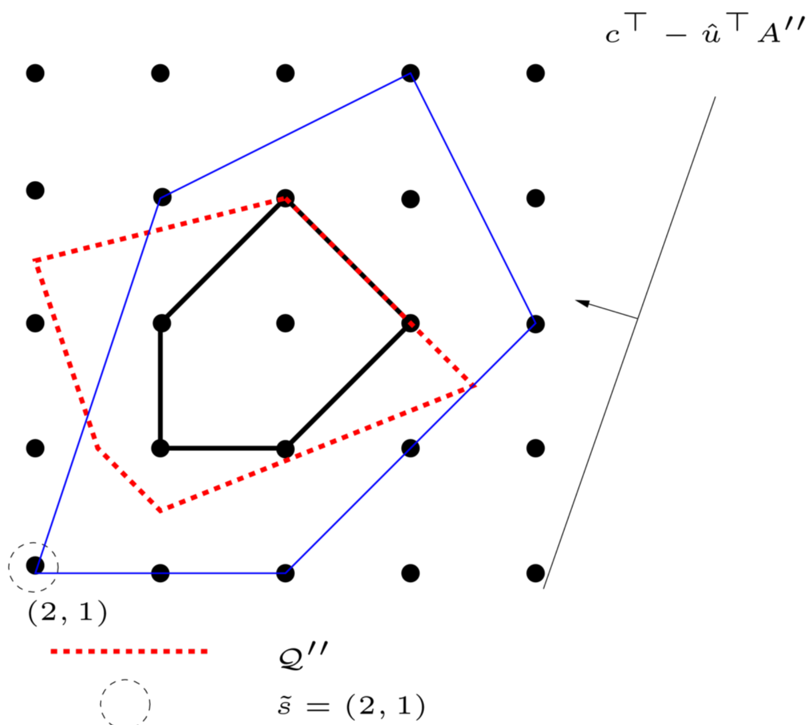
- **Master**: $z_{\mathrm{LD}} = \min_{u \in \mathcal{R}_+^{m''}} \left\{ \max_{s \in \mathcal{E}} \left\{ c^\top s + u^\top (b'' - A''s) \right\} \right\}$
- **Subproblem**: LR$(c^\top - u^\top A'')$



Geometry of the Lagrangian Dual

LD iteratively produces single extreme points of $\operatorname{conv}(\mathcal{S}_R)$ and uses the violation of the relaxed constraints to adjust the dual solution.

- **Master**: $z_{\text{LD}} = \min_{u \in \mathcal{R}_+^{m''}} \left\{ \max_{s \in \mathcal{E}} \left\{ c^\top s + u^\top (b'' - A''s) \right\} \right\}$
- **Subproblem**: $\text{LR}(c^\top - u^\top A'')$



Textbook Subgradient Algorithm

- The idea of the subgradient algorithm is to first fix u and determine x by optimzing over $\mathcal{S}_R$.
- Then update u according to the observed violations.
- Here is a basic *subgradient algorithm* for solving the Lagrangian dual:
  1. Choose initial Lagrange multipliers $u^0 \geq 0$ and set $t = 0$.
  2. Solve the Lagrangian subproblem $LR(u^t)$ to obtain x^t.
  3. Calculate the current violation of the complicating constraints $\gamma^t = b'' - A''x^t$.
  4. Set $u_j^{t+1} \leftarrow \max\{u_j^t - \theta^t\gamma^t, 0\}$ where θ^t is the chosen *step size*.
  5. Set $t \leftarrow t + 1$ and go to step 2.
- This algorithm is guaranteed to converge to the optimal solution as long as $\{\theta^t\}_{t=0}^{\infty} \rightarrow 0$ and $\sum_{t=0}^{\infty} \theta^t = \infty$
- In practice, one usually uses a geometric progression for the step sizes.
- Sometimes, it's difficult to know when the optimal solution has been reached.

Performing the Updates

- Suppose we have an estimate L^* of the optimal value.
- We can choose u^{t+1} such that the Lagrangian objective of x^t is L^*.
- In other words, we want

$$c^\top x^t + u^{t+1}\gamma^t = L^*$$

- At the same time, we have that $u^{t+1} = u^t - \theta_k \gamma^t$ (in the equality constrained case), so we have

$$c^\top x^t + [u^t - \theta_t \gamma^t]\gamma^t = L^*$$

Performing the Updates (cont.)

- Finally, solving and putting it all together, we obtain

$$\theta_t = \frac{L(u^t) - L^*}{\|\gamma^t\|^2}$$

- Since we do not usually know a good value for the new target, we can instead use the value of the best know solution.
- We also scale by a small factor that we reduce as the algorithm progresses.
- We then finally have

$$\theta_t = \frac{\alpha^t [L(u^t) - LB]}{\|\gamma^t\|^2}$$

- Here α^t is an additional factor used to reduce the step size over time.
- Typically, we start with $\alpha^0 = 2$ and reduce α^t by half when the Lagrangian objective does not improve for a specified number of iterations.

Example: Knapsack Problem

- We consider a binary knapsack problem $\max_{x\in\{0,1\}^n}\{c^\top x \mid a^\top x \le b\}$ for $a, c \in \mathbb{Z}^n_+$ and $b \in \mathbb{Z}_+$.
- If we relax the knapsack constraint, we have only bound constraints left.
- The relaxation can be solved simply by setting any variable with a positive coefficient to its upper bound and variable with negative coefficient to its lower bound.
- Thus,

$$LR(u) = \sum_{i=1}^{n} \max\{0, c_i - ua_i\} + ub \tag{1}$$

- Note that the feasible region in this case has all integral extreme points, so $z_{LD} = z_{LP}$.

Example: Knapsack Problem (cont.)

- Let us assume from here on that the variables are arranged in non-increasing order by the ratio c_i/a_i.
- Under this assumption, we can rewrite (**??**) equivalently as:

$$LR(u) = \sum_{i=j}^{n} c_i + u(b - \sum_{i=j}^{n} a_i) \tag{2}$$

where $j = \mathrm{argmin}\{i \mid c_i - ua_i \geq 0\} = \mathrm{argmin}\{i \mid c_i/a_i \geq u\}$.

- We know $LR(u)$ will be minimized when it has a zero subgradient, which will occur for $u = c_j/a_j$, where $\sum_{i=1}^{j} a_i \leq b \leq \sum_{i=1}^{j+1} a_i$.
- Note that this optimal solution is exactly the same as the optimal dual solution to the LP relaxation, derived from LP duality.

Example: Knapsack Problem (cont.)

- Let us now consider an instance with $n = 3$ described by the data $a = [3\ 1\ 4]$, $c = [10\ 4\ 14]$, and $b = 4$.
- Since the cost vector c is non-negative, the first solution will be to choose all items, i.e., set all variables to value 1.
- If we don't normalize the residuals, then we have $u_1 = u_0 + \theta_0\gamma_0 = \sum_{i=1}^{n} a_i - b$.
- Here is the sequence of iterates:

t	x^t	γ_t	u_t	θ_t
0	$[1\ 1\ 1]$	4	0	1
1	$[0\ 1\ 0]$	-3	4	$\frac{1}{2}$
2	$[1\ 1\ 1]$	4	$\frac{5}{2}$	$\frac{1}{4}$
3	$[0\ 1\ 1]$	1	$\frac{7}{2}$	$\frac{1}{8}$
4	$[0\ 1\ 0]$	-3	$\frac{29}{8}$	$\frac{1}{16}$
5	$[0\ 1\ 1]$	1	$\frac{55}{16}$	$\frac{1}{32}$
6	$[0\ 1\ 1]$	1	$\frac{111}{32}$	$\frac{1}{64}$

- The same solution is now repeated and the sequence will converge to the optimal value of $7/2$.

Example: Knapsack Problem (cont.)

- Note that the optimal solution was reached in the fourth iteration on the previous slide, but this was prior to convergence.
- The sequence above is not actually unique because of the fact that there is an alternative optimal solution to the Lagrangian subproblem in iteration 3.
- Here is an alternative sequence:

t	x^t	γ_t	u_t	θ_t
0	[1 1 1]	4	0	1
1	[0 1 0]	-3	4	$\frac{1}{2}$
2	[1 1 1]	4	$\frac{5}{2}$	$\frac{1}{4}$
3	[0 1 0]	-3	$\frac{7}{2}$	$\frac{1}{8}$
4	[0 1 0]	1	$\frac{25}{8}$	$\frac{1}{16}$
5	[0 1 1]	1	$\frac{51}{16}$	$\frac{1}{32}$
6	[0 1 1]	1	$\frac{103}{32}$	$\frac{1}{64}$

- We can see that this sequence will converge to $104/32 = 3.25$ rather than to the optimum.

Dantzig-Wolfe Decomposition

- In this technique, we utilize the fact that every point in $\mathrm{conv}(\mathcal{S}_R)$ can be written as the convex combination of extreme points of $\mathrm{conv}(\mathcal{S}_R)$.
- Here is the Dantzig-Wolfe LP:

$$\begin{aligned} \max \quad & c^\top x \\ s.t. \quad & \sum_{s \in \mathcal{E}} \lambda_s s = x \\ & A'' x \leq b'' \\ & \sum_{s \in \mathcal{E}} \lambda_s = 1 \\ & \lambda \in \mathbb{R}_+^{\mathcal{E}} \end{aligned}$$

  where $\mathcal{E}$ is the set of extreme points of $\mathrm{conv}(\mathcal{S}_R)$.

- As we observed previously, if we enforce integrality of x, this is a reformulation of the IP.
- This is a relaxation of IP; solving yields an upper bound on z_{DW}.
- Typically, x is not explicitly present in the formulation.

Dantzig-Wolfe LP

We can rewrite the Dantzig-Wolfe LP in the following two forms

$$
\begin{aligned}
\max\ & c^\top \left(\sum_{s \in \mathcal{E}} s \lambda_s \right) \\
s.t.\ \ & A'' \left(\sum_{s \in \mathcal{E}} s \lambda_s \right) \leq b'' \\
& \sum_{s \in \mathcal{E}} \lambda_s = 1 \\
& \lambda \in \mathbb{R}^{\mathcal{E}}_{+}
\end{aligned}
$$

$$
\begin{aligned}
\max\ & \sum_{s \in \mathcal{E}} (c^\top s) \lambda_s \\
s.t.\ \ & \sum_{s \in \mathcal{E}} (A'' s) \lambda_s \leq b'' \\
& \sum_{s \in \mathcal{E}} \lambda_s = 1 \\
& \lambda \in \mathbb{R}^{\mathcal{E}}_{+}
\end{aligned}
$$

Solving the Dantzig-Wolfe LP

- We solve this Dantzig-Wolfe LP (often called the *master problem*) using column generation.
- We begin with a restricted set of columns generated heuristically.
  - Start with a subset of "promising" columns.
  - Solve the *restricted master problem* (RMP) with just these columns.
  - *Price* the remaining columns and add those with positive reduced costs.
  - Iterate.

Column Generation

- In each iteration, we need to find a column with the *most positive reduced cost* or prove that there is no such column.
- *This is an optimization problem!*
- If we can solve this optimization problem, then we can solve the LP without explicitly listing the columns.
- This is nothing more than the cutting plane method applied to the dual.
- There are many variants of this basic algorithm, which we will discuss in more detail later.
- All are based in the ability to *generate a column* with positive reduced cost, given the current dual prices.
- In Dantzig-Wolfe, the column generation subproblem is an optimization problem over $\mathcal{S}_R$, which we know how to solve efficiently.
- In fact, it is precisely the Lagrangian subproblem!

The Dantzig-Wolfe Subproblem

- In Dantzig-Wolfe, we have a column for each member of $\mathcal{E}$.
- For $s \in \mathcal{E}$, if we take

$$\begin{aligned} c_s &= c^\top s \\ A_s &= A'' s, \end{aligned}$$

then the reduced cost of the column associated with s with respect to a given dual solution u is

$$c_s - uA_s - \alpha = c^\top s - u(A''s) - \alpha = (c^\top - uA'')s - \alpha,$$

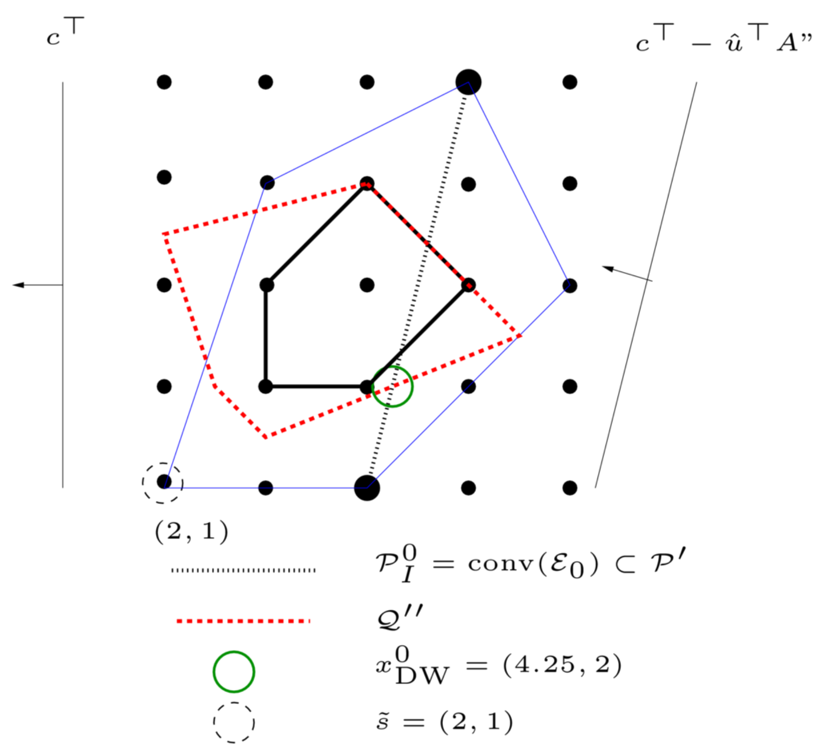
where α is the dual multiplier on the convexity constraint.

- Since α is a constant with respect to this subproblem, the column generation subproblem is equivaent to $LR(u)$.

Geometry of Dantzig-Wolfe Decomposition

DW utilizes an *inner* approximation of $\mathrm{conv}(\mathcal{S}_R)$

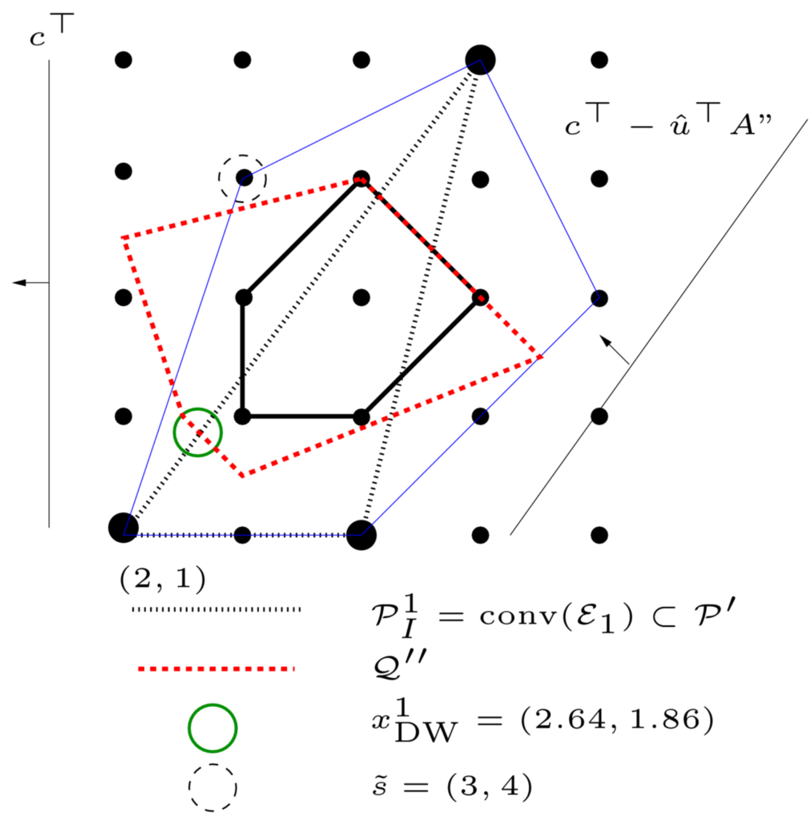
- **Master**:
  $z_{\mathrm{DW}} = \max_{\lambda \in \mathcal{R}_+^{\mathcal{E}}} \left\{ c^\top \left(\sum_{s \in \mathcal{E}} s\lambda_s \right) \mid A'' \left(\sum_{s \in \mathcal{E}} s\lambda_s \right) \leq b'', \sum_{s \in \mathcal{E}} \lambda_s = 1 \right\}$
- **Subproblem**: $LR(c^\top - u^\top A'')$



Geometry of Dantzig-Wolfe Decomposition

DW utilizes an *inner* approximation of $\text{conv}(\mathcal{S}_R)$

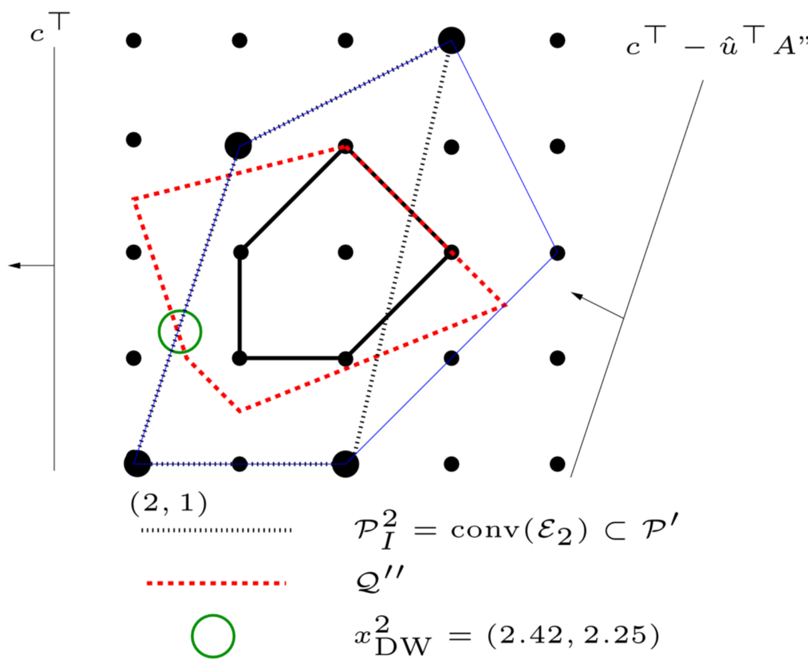
- **Master**:
  $z_{\text{DW}} = \max_{\lambda \in \mathcal{R}_+^{\mathcal{E}}} \left\{ c^\top \left(\sum_{s \in \mathcal{E}} s\lambda_s \right) \mid A'' \left(\sum_{s \in \mathcal{E}} s\lambda_s \right) \le b'', \sum_{s \in \mathcal{E}} \lambda_s = 1 \right\}$
- **Subproblem**: $LR(c^\top - u^\top A'')$



Geometry of Dantzig-Wolfe Decomposition

DW utilizes an *inner* approximation of $\mathrm{conv}(\mathcal{S}_R)$

- **Master**:
  $z_{\mathrm{DW}} = \max_{\lambda \in \mathcal{R}_+^{\mathcal{E}}} \left\{ c^\top \left(\sum_{s \in \mathcal{E}} s \lambda_s \right) \mid A'' \left(\sum_{s \in \mathcal{E}} s \lambda_s \right) \le b'', \sum_{s \in \mathcal{E}} \lambda_s = 1 \right\}$
- **Subproblem**: $LR(c^\top - u^\top A'')$



Block Structure and Dantzig-Wolfe

- When the problem has block structure, the single subproblem may decompose into independent blocks.
- In this case, we can use a separate convexity constraint for each block.
- In some cases, these blocks are *identical*.
- In this case, we use a convexity constraints, but with right-hand side K, where K is the number of blocks.

Example: The Generalized Assignment Problem

- The problem is to assign m tasks to n machines subject to capacity constraints.
- An IP formulation of this problem is

$$\begin{aligned}
\max \; & \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} z_{ij} \\
s.t. \; & \sum_{j=1}^{n} z_{ij} = 1, \qquad i = 1, \ldots, m, \\
& \sum_{i=1}^{m} w_{ij} z_{ij} \leq d_j, \qquad j = 1, \ldots, n, \\
& z_{ij} \in \{0, 1\}, i = 1, \ldots, m, j = 1, \ldots, n,
\end{aligned}$$

- The variable z_{ij} is one if task i is assigned to machine j.
- The "profit" associated with assigning task i to machine j is p_{ij}.

Applying Dantzig-Wolfe to the GAP

- Let's apply Dantzig-Wolfe to obtain a stronger bound for the GAP.
- Note that if we relax the constraint that each item be assigned to a different machine, the problem decomposes by machine.
- This allows us to use a separate convexity constraint for each machine.
- Then we have

$$\max \sum_{j=1}^{n} \sum_{i=1}^{m} p_{ij} \left(\sum_{k=1}^{K_j} \lambda_k^j a_{ik}^j \right)$$

$$\begin{aligned} s.t. \quad & \sum_{i=1}^{n} \sum_{k=1}^{K_j} \lambda_k^j a_{ik}^j = 1, && i = 1, \ldots, m, \\ & \sum_{k=1}^{K_j} \lambda_k^j = 1, && j = 1, \ldots, n, \\ & \lambda_k^j \in \{0, 1\}, j = 1, \ldots, n, k = 1, \ldots, K_j, \end{aligned}$$

Applying Dantzig-Wolfe to the GAP (cont.)

- The columns are subsets of the tasks that can be assigned to one of the machines (called *assignments*).
- The coefficient a^j_{ik} is 1 if task i is assigned to machine j in the k^{th} assignment.

Examining the Dantzig-Wolfe Master for the GAP

- The columns represent feasible assignments of tasks to machines.
- Note that one feasible assignment is to assign no tasks, which would correspond to a column of all zeros.
- Therefore, we could also write the convexity constraints as inequalities.
- Finding an initial feasible set of columns is trivial.
- The relaxation decomposes into a set of knapsack problems.
- Note that the master problem is a relaxation of a set partitioning problem.

The Cutting Plane Method as a Decomposition Method

- Finally, it is possible to exploit our ability to optimize over $\mathcal{S}_R$ in a more traditional cutting plane method.
- Recall the algorithm for separating using an optimization oracle from Lecture 12.
- We can use this algorithm as a means of separating (possibly infeasible) solutions from $\mathcal{S}_R$ in the context of a cutting plane method.

Lagrange Cuts

- Boyd observed that for $u \in \mathbb{R}^m_+$, a *Lagrange cut* of the form

$$(c - uA'')^\top x \geq LR(u) - ub'' \qquad \text{(LC)}$$

is valid for $\mathcal{P}$.

- If we take u^* to be the optimal solution to the Lagrangian dual, then this inequality reduces to

$$(c - u^* A'')^\top x \geq z_D - ub'' \qquad \text{(OLC)}$$

- If we now take

$$x^D \in \operatorname{argmin}\left\{c^\top x \mid A''x \leq b'', (c - u^* A'')^\top x \geq z_D - ub''\right\},$$

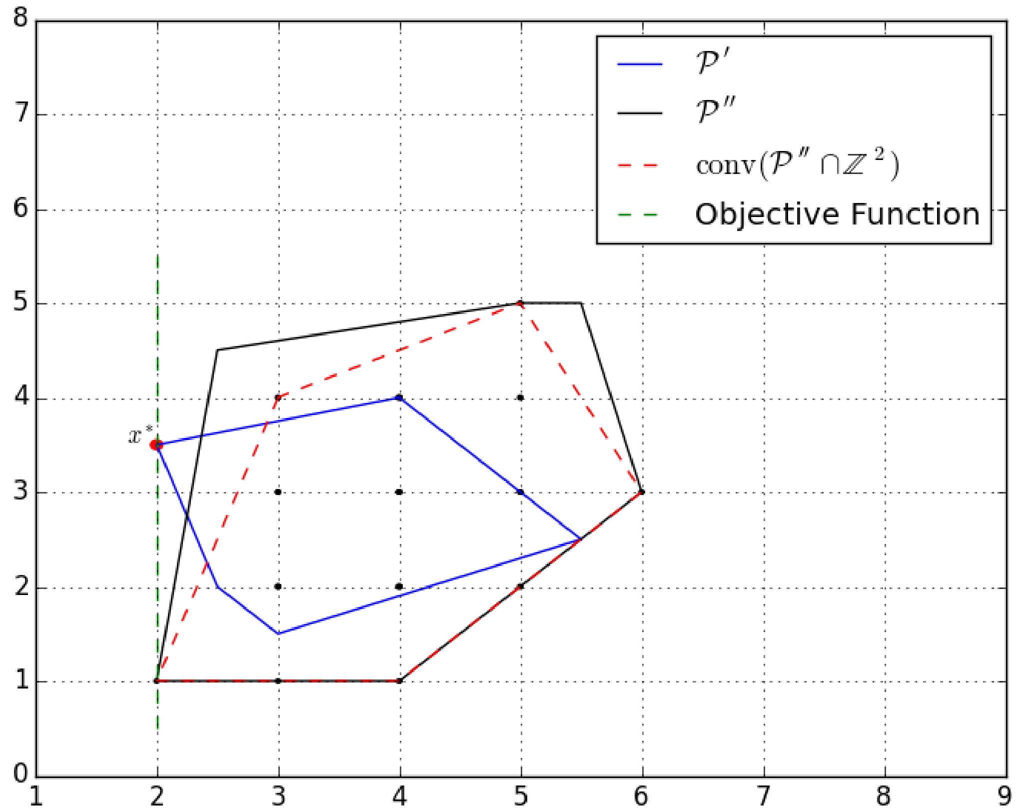
then we have $c^\top x^D = z_D$.

- Such cuts can be generated using an optimization-based oracle.

Geometry of the Cutting Plane Method

CPM utilizes an optimization-based oracle to separate from $\mathrm{conv}(\mathcal{S}_R)$

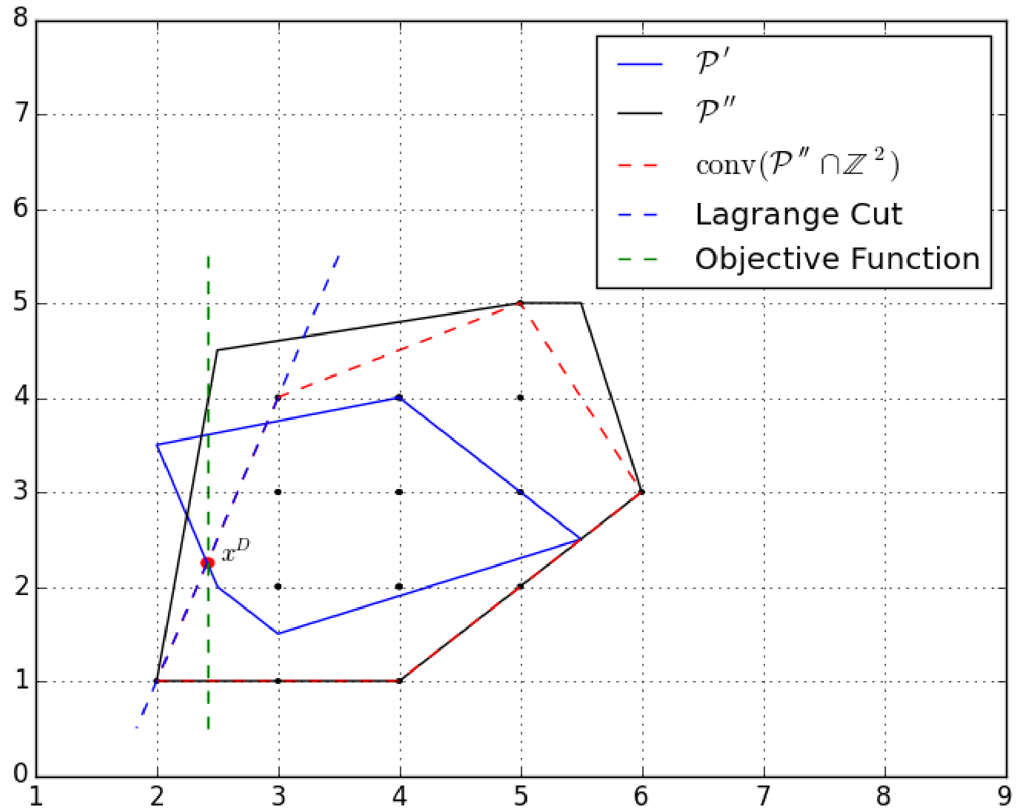
- **Master**:
  $z_{\mathrm{CP}} = \max_{x \in \mathcal{R}^n_+} \{c^\top x \mid A''x \leq b'', (\alpha^k)^\top x \leq \beta^k, 1 \leq k \leq L\}$
- **Subproblem**: $OPT(S_R)$



Geometry of the Cutting Plane Method

CPM utilizes an optimization-based oracle to separate from $\mathrm{conv}(\mathcal{S}_R)$

- **Master**:
  $z_{\mathrm{CP}} = \max_{x \in \mathcal{R}^n_+} \{c^\top x \mid A''x \leq b'', (\alpha^k)^\top x \leq \beta^k, 1 \leq k \leq L\}\}$
- **Subproblem**: $OPT(S_R)$



Comparing the Methods

- Recall that the Lagrangian dual can be rewritten as the following LP

$$z_{LD} = \min_{\eta,u}\{\eta + ub'' \mid \eta \geq (c^\top - uA'')s, s \in \mathcal{E}, u \geq 0\}$$

- It is easy to show that this LP is the dual of the Dantzig-Wolfe LP.
- Thus, both these method produce the same bound (in principle).

$$z_D = z_{LD} = z_{DW}$$

- The cutting plane method just described is yet another method for computing the same bound.
- In practice, there are great differences between these three methods, both algorithmically and numerically.
  - Conceptually, the Lagrangian dual produces only a dual solution and does not include any explicit primal solution information.
  - The Dantzig-Wolfe LP produces a primal solution, which can be used to perform generate valid inequalities and tighten the relaxation.
- Naive implementations are slow to converge and numerical difficulties may prevent the calculation of an exact bound.

Choosing a Decomposition

- Typically, there are multiple choices for decomposing a give IP.
- The definition of the set $\mathcal{S}_R$ determines the strength of the bound.
- However, it is important to choose a relaxation that can be solved relatively easily (but not too easily).
- The relaxation must be solved iteratively in order to solve the Lagrangian dual.
- Recall the TSP example.
- Other Examples
  - Flow Problem with Budget Constraints
  - Facility Location Problem
  - Generalized Assignment Problem

Comparing Decomposition-based Bounding to LP-based Bounding

- The class of methods we have just discussed are called *decomposition-based methods* because they decompose the problem into two parts.
- Up until the mid-1970's, these methods were very popular for solving integer programming problems.
- They can effectively strengthen the bound obtained by LP relaxation alone.
- However, after methods based on strengthening the LP relaxation using valid inequalities were introduced, they fell out of favor.
- It is possible to combine these two approaches.
- This is one of the current frontiers of research in integer programming.