Integer Programming

Lecture 18

Decomposition Methods

- Many complex models are built up from simpler structures.
  - Subsystems linked by system-wide constraints or variables.
  - Complex combinatorial structures obtained by combining simpler ones.
  - Simple models with additional "complicating constraints."
- Decomposition is the process of taking a model and breaking it into smaller parts.
- The goal is either to
  - reformulate the model for easier solution;
  - reformulate the model to obtain an improved relaxation (bound); or
  - separate the model into stages or levels (possibly with separate objectives).

Block Structure

- "Classical" decomposition arises from *block structure* in the constraint matrix.
- By relaxing/fixing the linking variables/constraints, we then get a model that is separable.
- A separable model consists of multiple smaller submodels that are easier to solve.
- The separability lends itself nicely to parallel implementation.

$$\begin{pmatrix} A_{01} & A_{02} & \cdots & A_{0\kappa} \\ A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_{\kappa\kappa} \end{pmatrix} \quad \begin{pmatrix} A_{10} & A_{11} & & & \\ A_{20} & & A_{22} & & \\ \vdots & & & \ddots & \\ A_{\gamma 0} & & & & A_{\kappa\kappa} \end{pmatrix}$$

The Decomposition Principle

- Decomposition methods leverage our ability to solve either a relaxation or a restriction.
- Methodology is based on the ability to solve a given *subproblem* repeatedly with varying inputs.
- The goal of solving the subproblem repeatedly is to obtain information about its structure that can be incorporated into a *master problem*.
- At a high level, *most solution methods* for discrete optimization problems are based on the decomposition principle.
- Constraint decomposition
  - Relax a set of *complicating constraints* to obtain a more tractable problem.
  - Leverages ability to solve either the optimization or separation problem for the *relaxation* (with varying objectives and/or points to be separated).
- Variable decomposition
  - Fix the values of *complicating variables* to expose the structure.
  - Leverages ability to solve a *restriction* (with varying right-hand sides).

Example: Block Structure (Linking Constraints)

Generalized Assignment Problem (GAP)

$$
\begin{array}{llll}
\max & \sum_{i \in M} \sum_{j \in N} p_{ij} x_{ij} & & \\
& \sum_{j \in N} w_{ij} x_{ij} & \leq & b_i & \forall i \in M \\
& \sum_{i \in M} x_{ij} & = & 1 & \forall j \in N \\
& x_{ij} & \in & \{0,1\} & \forall i,j \in M \times N
\end{array}
$$

- The problem is to assign m tasks to n machines subject to capacity constraints.
- The variable x_{ij} is one if task i is assigned to machine j.
- The profit associated with assigning task i to machine j is p_{ij}.
- If we relax the requirement that each task be assigned to only one machine, the problem decomposes into n knapsack problems.

Example: Block Structure (Linking Variables)

Facility Location Problem

$$
\begin{array}{lll}
\min & \displaystyle\sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} x_{ij} & \\
\text{s.t.} & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & \forall i \\
& x_{ij} \leq y_j & \forall i, j \\
& x_{ij}, y_j \in \{0, 1\} & \forall i, j
\end{array}
$$

- We are given n facility locations and m customers to be serviced from those locations.
- There is a fixed cost c_j associated with facility j.
- There is a profit d_{ij} associated with serving customer i from facility j.
- We have two sets of binary variables.
  - y_j is 1 if facility j is opened, 0 otherwise.
  - x_{ij} is 1 if customer i is served by facility j, 0 otherwise.
- If we fix the set of open facilities, then the problem becomes easy.

Constraint Decomposition

- We focus for now on constraint decomposition.
- For simplicity, we consider a pure integer optimization problem (ILP) defined as usual by

$$z_{IP} = \max\{c^\top x \mid x \in S\}, \qquad \text{(ILP)}$$
$$\mathcal{S} = \{x \in \mathbb{Z}^n_+ \mid Ax \leq b\}.$$

- We will exploit the ability to solve a relaxation of this problem to generate an improved relaxation.

Notation

We divide the constraints into two set and use the following notation to refer to various relaxations of the original feasible region.

$$
\begin{aligned}
&\max\ c^\top x \\
&\quad \text{s.t. } A'x \le b' \text{ (the ``nice'' constraints)} \\
&\qquad A''x \le b'' \text{ (the ``complicating'' constraints)} \\
&\qquad\quad x \in \mathbb{Z}^n
\end{aligned}
\tag{MILP-D}
$$

$$
\begin{aligned}
\mathcal{Q}' &= \{x \in \mathbb{R}^n \mid A'x \le b'\}, \\
\mathcal{Q}'' &= \{x \in \mathbb{R}^n \mid A''x \le b''\}, \\
\mathcal{Q} &= \mathcal{Q}' \cap \mathcal{Q}'', \\
\mathcal{S} &= \mathcal{Q} \cap \mathbb{Z}^n, \text{ and} \\
\mathcal{S}_R &= \mathcal{Q}' \cap \mathbb{Z}^n.
\end{aligned}
$$

The Decomposition Bound

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

$$\begin{aligned} z_{\mathrm{IP}} &= \max_{x \in \mathbb{Z}^n} \left\{ c^\top x \mid A'x \le b', A''x \le b'' \right\} \\ z_{\mathrm{LP}} &= \max_{x \in \mathbb{R}^n} \left\{ c^\top x \mid A'x \le b', A''x \le b'' \right\} \\ z_{\mathrm{D}} &= \max_{x \in \mathrm{conv}(\mathcal{S}_R)} \left\{ c^\top x \mid A''x \le b'' \right\} \end{aligned}$$

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

This bound can be computed 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 $\mathrm{conv}(\mathcal{S}_R)$).

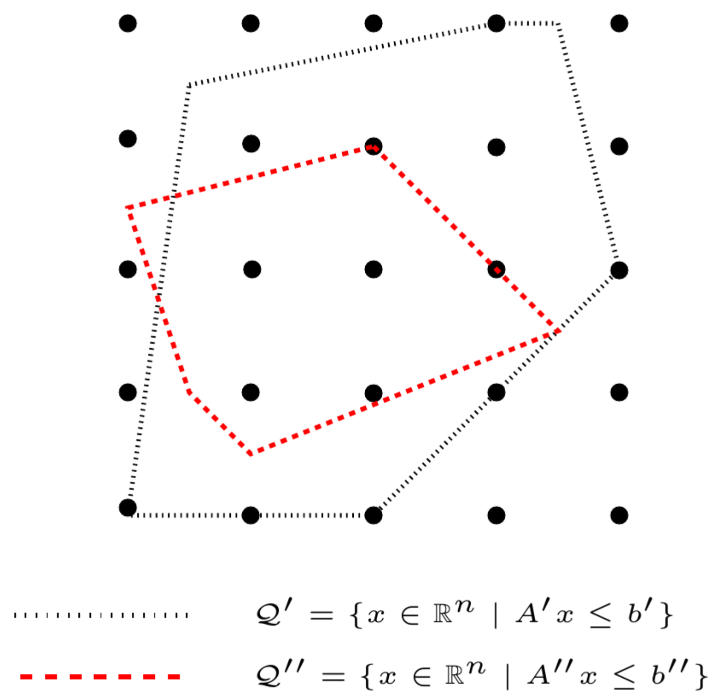
Example

$$
\begin{aligned}
\min \quad & x_1 \\
-x_1 - x_2 &\geq -8, & (1)\\
-0.4x_1 + x_2 &\geq 0.3, & (2)\\
x_1 + x_2 &\geq 4.5, & (3)\\
3x_1 + x_2 &\geq 9.5, & (4)\\
0.25x_1 - x_2 &\geq -3, & (5)\\
7x_1 - x_2 &\geq 13, & (6)\\
x_2 &\geq 1, & (7)\\
-x_1 + x_2 &\geq -3, & (8)\\
-4x_1 - x_2 &\geq -27, & (9)\\
-x_2 &\geq -5, & (10)\\
0.2x_1 - x_2 &\geq -4, & (11)\\
x &\in \mathbb{Z}''. & (12)
\end{aligned}
$$

Example (cont)

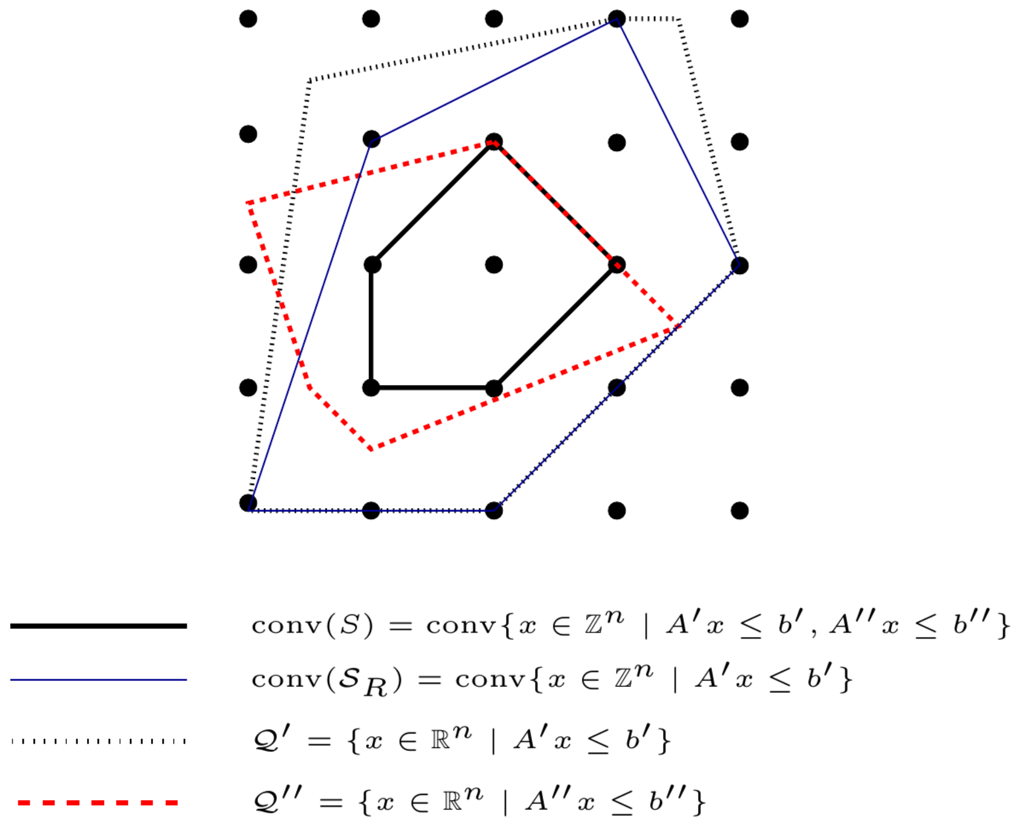
$$
\begin{aligned}
\mathcal{Q}' &= \{x \in \mathbb{R}^2 \mid x \text{ satisfies } (1) - (5)\}, \\
\mathcal{Q}'' &= \{x \in \mathbb{R}^2 \mid x \text{ satisfies } (6) - (11)\}, \\
\mathcal{Q} &= \mathcal{Q}' \cap \mathcal{Q}'', \\
\mathcal{S} &= \mathcal{Q} \cap \mathbb{Z}^n, \text{ and} \\
\mathcal{S}_R &= \mathcal{Q}' \cap \mathbb{Z}^n.
\end{aligned}
$$

Constraint Decomposition in Integer Programming



- Optimization over $\mathcal{S}$ is *"hard"*
- Optimization over $\mathcal{S}_R$ is *"easy"*
- We can generate extreme points and/or facet-defining inequalities of $\mathrm{conv}(\mathcal{S}_R)$ "effectively."

Constraint Decomposition in Integer Programming



- Optimization over $\mathcal{S}$ is *"hard"*
- Optimization over $\mathcal{S}_R$ is *"easy"*
- We can generate extreme points and/or facet-defining inequalities of $\mathrm{conv}(\mathcal{S}_R)$ "effectively."

The Strength of the Decomposition Bound

- We have

$$z_D = \max\{c^\top x \mid A''x \leq b'', x \in \operatorname{conv}(\mathcal{S}_R)\}$$

- From this, we can characterize exactly when the decomposition bound is strong and when it is weak.

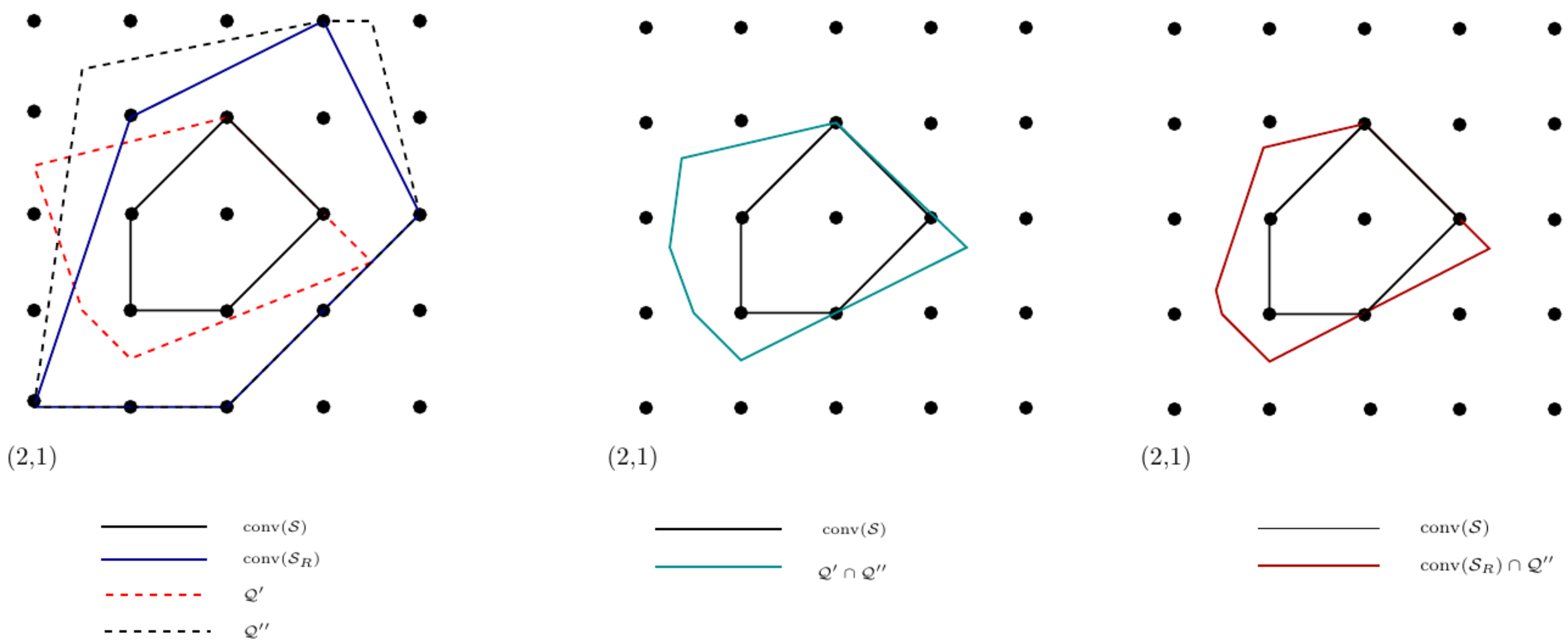
Proposition 1. $z_D = z_{IP}$ *for all objective functions if and only if*

$$\operatorname{conv}\{\mathcal{S}_R \cap \{x \in \mathbb{R}^n_+ \mid A''x \leq b''\}\} = \operatorname{conv}(\mathcal{S}_R) \cap \{x \in \mathbb{R}^n_+ \mid A''x \leq b''\}$$

Proposition 2. $z_D = z_{LP}$ *for all objective functions if and only if*

$$\mathcal{Q}' = \operatorname{conv}(\mathcal{S}_R)$$

Illustrating the Strength of the Decomposition Bound



Comparing the Decomposition Bound to the LP Bound

- The following proposition follows again from the characterization of z_D.

  Proposition 3. *The LP and the decomposition bound are exactly the same for all objective functions if* $\{x \in \mathbb{R}^n_+ \mid A'x \leq b'\}$ *is an integral polyhedron.*

- This follows from the fact that $\mathrm{conv}(\mathcal{S}_R) = \{x \in \mathbb{R}^n_+ \mid A'x \leq b'\}$ in this case.
- Because of the equivalence of optimization and separation, we can in theory always attain this bound using a cutting plane algorithm.
- Incorporating cutting plane methods in with the bounding methods we have discussed so far is a topic for later in the course.