Duality and Discrete Optimization

Lecturer: Pradeep Ravikumar
Co-instructor: Aarti Singh

Convex Optimization 10-725/36-725

Discrete Optimization

minimize $f(x)$

subject to $x \in X, \qquad g_j(x) \leq 0, \qquad j = 1, \ldots, r,$

where X is a *finite* set.

Discrete Optimization

minimize $f(x)$

subject to $x \in X, \qquad g_j(x) \leq 0, \qquad j = 1, \dots, r,$

where X is a *finite* set.

- Example: 0-1 Integer programming:

$$X = \big\{(x_1, \dots, x_n) \mid x_i = 0 \text{ or } 1,\ i = 1, \dots, n\big\}.$$

Example: Network Flow

- Think of:
  - Nodes i with $s_i > 0$ and $s_i < 0$ as production and consumption points, respectively.
  - s_i supply or demand of node i.
  - Arcs (i, j) as transportation links with flow capacity c_{ij} and cost per unit flow a_{ij}
  - Problem is to accomplish a minimum cost transfer from the supply to the demand points.
- Important special cases: Shortest path, max-flow, transportation, assignment problems.

Example: Network Flow

- Given a directed graph with set of nodes $\mathcal{N}$ and set of arcs $(i,j) \in \mathcal{A}$, the (integer constrained) minimum cost network flow problem is

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij} x_{ij}$$

subject to the constraints

$$\sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j \mid (j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\, i \in \mathcal{N},$$

$$b_{ij} \le x_{ij} \le c_{ij}, \quad \forall\ (i,j) \in \mathcal{A}, \qquad x_{ij}: \text{ integer},$$

where a_{ij}, b_{ij}, c_{ij}, and s_i are given scalars.

Example: Network Flow

- The minimum cost flow problem has an interesting property: If the s_i and c_{ij} are integer, the optimal solutions of the integer-constrained problem also solve the $relaxed$ problem, obtained when the integer constraints are neglected.
- Great practical significance, since the relaxed problem can be solved using efficient linear (not integer) programming algorithms.

Unimodularity

- This is special case of $unimodularity$:
  - A square matrix A with integer components is $unimodular$ if its determinant is 0, 1, or -1.

Unimodularity

- This is special case of $unimodularity$:
  - A square matrix A with integer components is $unimodular$ if its determinant is 0, 1, or -1.
  - If A is invertible and unimodular, by Kramer's rule, the inverse matrix A^{-1} has integer components. Hence, the solution x of the system $Ax = b$ is integer for every integer vector b.

Unimodularity

- This is special case of $unimodularity$:
  - A square matrix A with integer components is $unimodular$ if its determinant is 0, 1, or -1.
  - If A is invertible and unimodular, by Kramer's rule, the inverse matrix A^{-1} has integer components. Hence, the solution x of the system $Ax = b$ is integer for every integer vector b.
  - A rectangular matrix with integer components is called $totally\ unimodular$ if each of its square submatrices is unimodular.

Unimodularity

- This is special case of $unimodularity$:
  - A square matrix A with integer components is $unimodular$ if its determinant is 0, 1, or -1.
  - If A is invertible and unimodular, by Kramer's rule, the inverse matrix A^{-1} has integer components. Hence, the solution x of the system $Ax = b$ is integer for every integer vector b.
  - A rectangular matrix with integer components is called $totally\ unimodular$ if each of its square submatrices is unimodular.

- A polyhedron $\{x \mid Ex = d,\, b \leq x \leq c\}$ has integer extreme points if E is totally unimodular and b, c, and d have integer components.

- The matrix E corresponding to the minimum cost flow problem is totally unimodular.

Non-unimodular Problems

- Unimodularity is an exceptional property.
- Nonunimodular example (Traveling salesman problem): A salesman wants to find a minimum cost tour that visits each of N given cities exactly once and returns to the starting city.

Example of Non-Unimodular Problem: Traveling Salesman Problem

- Let a_{ij}: cost of going from city i to city j, and let x_{ij} be a variable that takes the value 1 if the salesman visits city j immediately following city i, and the value 0 otherwise.

Example of Non-Unimodular Problem: Traveling Salesman Problem

- Let a_{ij}: cost of going from city i to city j, and let x_{ij} be a variable that takes the value 1 if the salesman visits city j immediately following city i, and the value 0 otherwise. The problem is

$$\text{minimize} \quad \sum_{i=1}^{N} \sum_{\substack{j=1,\ldots,N \\ j\neq i}} a_{ij}x_{ij}$$

$$\text{subject to} \quad \sum_{\substack{j=1,\ldots,N \\ j\neq i}} x_{ij} = 1, \qquad i = 1,\ldots,N,$$

$$\sum_{\substack{i=1,\ldots,N \\ i\neq j}} x_{ij} = 1, \qquad j = 1,\ldots,N,$$

plus the constraints $x_{ij} = 0$ or 1, and that the set of arcs $\{(i,j) \mid x_{ij} = 1\}$ forms a connected tour

Example of Non-Unimodular Problem: Traveling Salesman Problem

- Let a_{ij}: cost of going from city i to city j, and let x_{ij} be a variable that takes the value 1 if the salesman visits city j immediately following city i, and the value 0 otherwise. The problem is

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} \sum_{\substack{j=1,\ldots,N \\ j\neq i}} a_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{\substack{j=1,\ldots,N \\ j\neq i}} x_{ij} = 1, \qquad i = 1,\ldots,N, \\
& \sum_{\substack{i=1,\ldots,N \\ i\neq j}} x_{ij} = 1, \qquad j = 1,\ldots,N,
\end{aligned}
$$

plus the constraints $x_{ij} = 0$ or 1, and that the set of arcs $\{(i,j) \mid x_{ij} = 1\}$ forms a connected tour, i.e.,

$$\sum_{i\in S,\, j\notin S} (x_{ij}+x_{ji}) \geq 2, \quad \forall \text{ proper subsets } S \text{ of cities.}$$

Example: Graphical Model Inference

- Consider a random vector $X = (X_1, \dots, X_p)$ with distribution:

$$P(X) \propto \left\{ \sum_{s \in V(G)} \theta_s(x_s) + \sum_{(s,t) \in E(G)} \theta_{st}(x_s, x_t) \right\},$$

  i.e. X follows a pairwise graphical model distribution with graph $G = (V, E)$.

- An important "inference" problem in graphical models is to solve:

$$\begin{aligned} &\arg \max_{x_1, \dots, x_p} \; P(x) \\ &\equiv \arg \max_{x_1, \dots, x_p} \left\{ \sum_{s \in V(G)} \theta_s(x_s) + \sum_{(s,t) \in E(G)} \theta_{st}(x_s, x_t) \right\} \end{aligned}$$

- Called the Maximum A Posteriori (MAP) problem, this is an integer program when the values taken by the random variables lies in a discrete set e.g. $\{0, 1\}$.

- A large class of combinatorial optimization problems can be cast graphical model MAP problems, including satisfiability problems, decoding audio signals, among others.

Approaches to Integer Programming

- Enumeration of the finite set of all feasible (integer) solutions, and comparison to obtain an optimal solution (this is rarely practical).

Approaches to Integer Programming

- Enumeration of the finite set of all feasible (integer) solutions, and comparison to obtain an optimal solution (this is rarely practical).
- Constraint relaxation and heuristic rounding.
  - Neglect the integer constraints
  - Solve the problem using linear/nonlinear programming methods
  - If a noninteger solution is obtained, round it to integer using a heuristic
  - Sometimes, with favorable structure, clever problem formulation, and good heuristics, this works remarkably well

Approaches to Integer Programming

- Implicit enumeration (or branch-and-bound):
  - Combines the preceding two approaches
  - It uses constraint relaxation and solution of noninteger problems to obtain certain lower bounds that are used to discard large portions of the feasible set.
  - In principle it can find an optimal (integer) solution, but this may require unacceptable long time.
  - In practice, usually it is terminated with a heuristically obtained integer solution, often derived by rounding a noninteger solution.

Principle of Branch & Bound

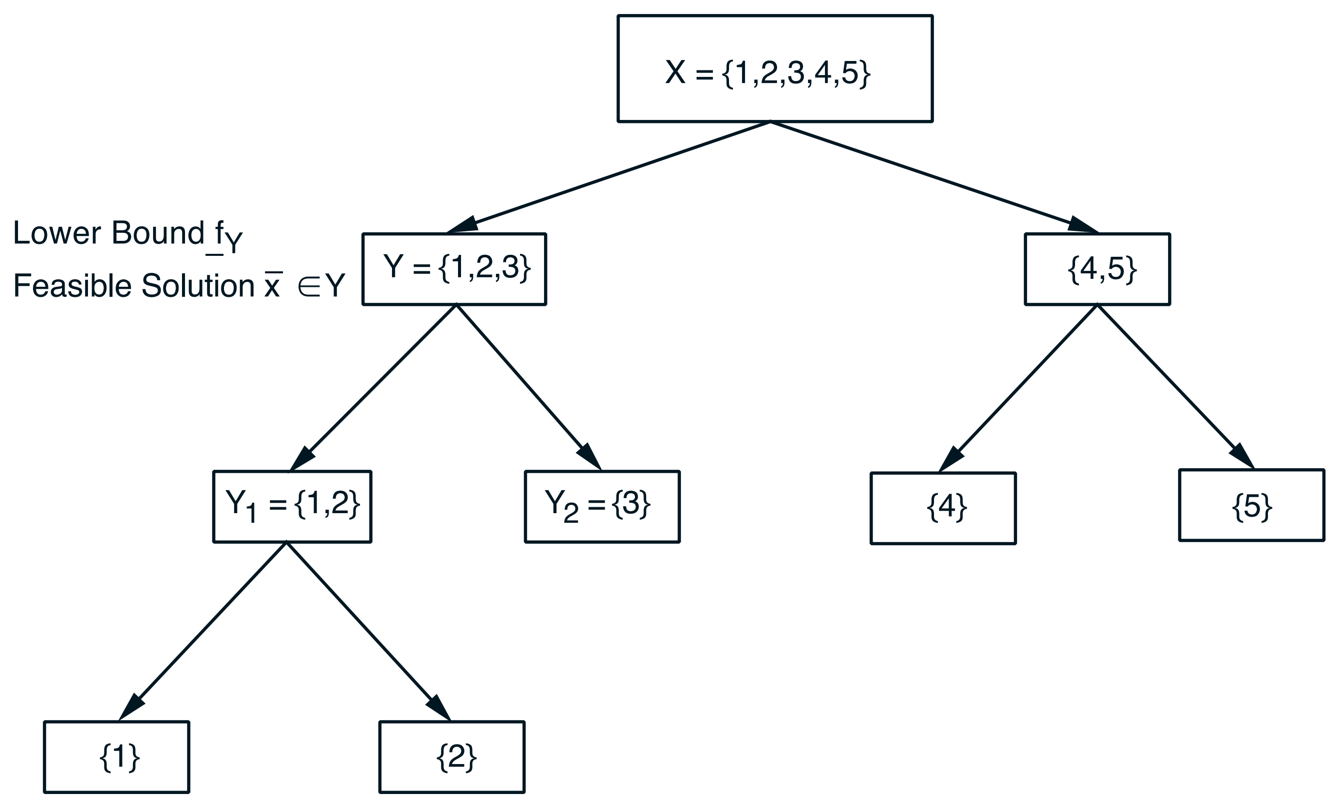
- **Bounding Principle:** Consider minimizing $f(x)$ over a finite set $x \in X$. Let Y_1 and Y_2 be two subsets of X, and suppose that we have bounds

$$\underline{f}_1 \leq \min_{x \in Y_1} f(x), \qquad \overline{f}_2 \geq \min_{x \in Y_2} f(x).$$

Then, if $\overline{f}_2 \leq \underline{f}_1$, the solutions in Y_1 may be disregarded since their cost cannot be smaller than the cost of the best solution in Y_2.

Branch & Bound

- The branch-and-bound method uses suitable upper and lower bounds, and the bounding principle to eliminate substantial portions of X.
- It uses a tree, with nodes that correspond to subsets of X, usually obtained by binary partition.

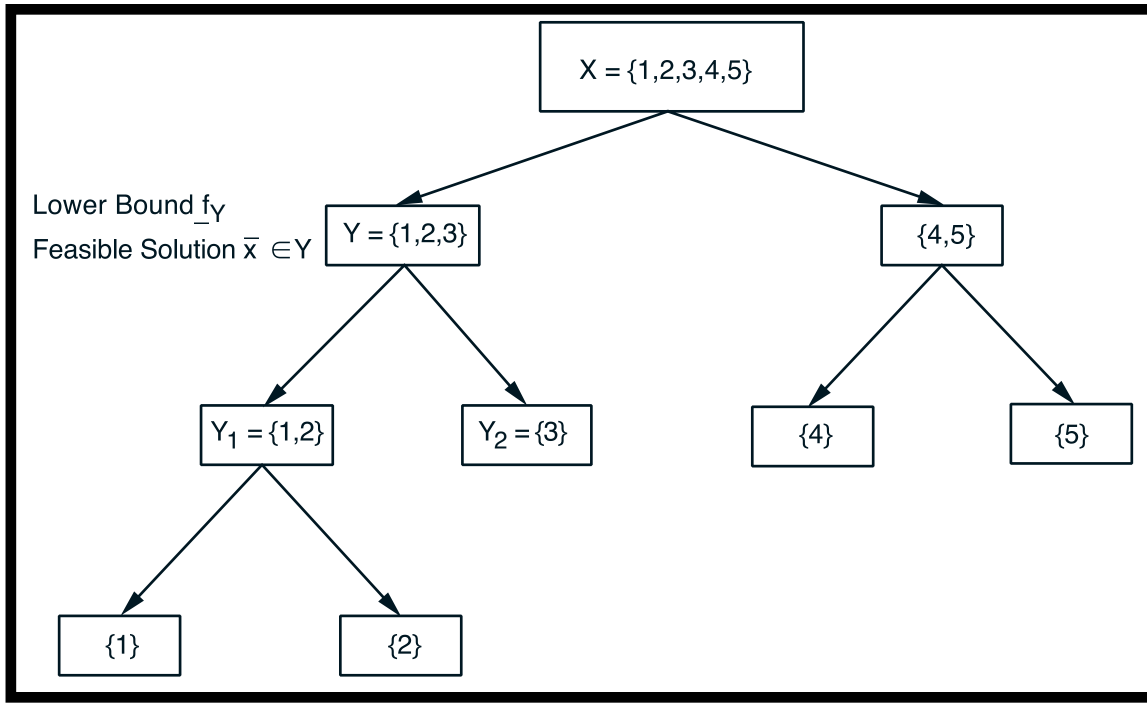


Branch and Bound Tree

- Nodes of graph correspond to a collection $\mathcal{X}$ of subsets of feasible set X
  - The nodes of tree from root to leaves specify a progressively finer partition of X
- The set of all solutions is the root node: $X \in \mathcal{X}$.
- All feasible solutions $x \in X$ are leaf nodes: $\{x\} \in \mathcal{X}$.
- If a set $Y \in \mathcal{X}$ has more than one solution, then there exist disjoint sets $Y_1, \ldots, Y_n \in \mathcal{X}$, such that:

$$\cup_{i=1}^{n} Y_i = Y.$$

  - Set Y is called the *parent* of $Y_1, \ldots, Y_n$.
  - $Y_1, \ldots, Y_n$ are called *children* of Y.
- Each set in $\mathcal{X}$ other than X has a parent



Branch & Bound: Key Ingredient

- For every non-terminal node Y, there is an algorithm that calculates:

(a) A lower bound $\underline{f}_Y$ to the minimum cost over Y

$$\underline{f}_Y \leq \min_{x \in Y} f(x).$$

(b) A feasible solution $\bar{x} \in Y$, whose cost $f(\bar{x})$ can serve as an upper bound to the minimum cost over Y (as well as over X).

Branch & Bound: Key Ingredient

- These bounds are used to save computation by discarding nodes Y of tree (and all its descendants) that have no chance of containing a solution better than current best solution
- For any node Y in the tree, check if the lower bound $\underline{f}_Y$ is larger than best available upper bound (minimal $f(\bar{x})$ over feasible solutions $\bar{x}$ considered so far)
- If so, we know Y cannot contain optimal solution, so Y and descendants can be safely discarded.

Branch and Bound Algorithm

- The algorithm maintains a node list called OPEN, and a scalar called UPPER, which is equal to the minimal cost over feasible solutions found so far. Initially, OPEN$= \{X\}$, and UPPER$= \infty$ or to the cost $f(\overline{x})$ of some feasible solution $\overline{x} \in X$.

Branch and Bound Algorithm

- The algorithm maintains a node list called OPEN, and a scalar called UPPER, which is equal to the minimal cost over feasible solutions found so far. Initially, OPEN$= \{X\}$, and UPPER$= \infty$ or to the cost $f(\overline{x})$ of some feasible solution $\overline{x} \in X$.

- $\textbf{Step 1:}$ Remove a node Y from OPEN. For each child Y_j of Y, do the following: Find the lower bound $\underline{f}_{Y_j}$ and a feasible solution $\overline{x} \in Y_j$. If

$$\underline{f}_{Y_j} < \text{ UPPER},$$

place Y_j in OPEN. If in addition

$$f(\overline{x}) < \text{ UPPER},$$

set UPPER $= f(\overline{x})$ and mark $\overline{x}$ as the best solution found so far.

Branch and Bound Algorithm

- The algorithm maintains a node list called OPEN, and a scalar called UPPER, which is equal to the minimal cost over feasible solutions found so far. Initially, OPEN$= \{X\}$, and UPPER$= \infty$ or to the cost $f(\overline{x})$ of some feasible solution $\overline{x} \in X$.

- **Step 1:** Remove a node Y from OPEN. For each child Y_j of Y, do the following: Find the lower bound $\underline{f}_{Y_j}$ and a feasible solution $\overline{x} \in Y_j$. If

$$\underline{f}_{Y_j} < \text{ UPPER},$$

place Y_j in OPEN. If in addition

$$f(\overline{x}) < \text{ UPPER},$$

set UPPER $= f(\overline{x})$ and mark $\overline{x}$ as the best solution found so far.

Step 2: (Termination Test) If OPEN is nonempty, go to step 1. Otherwise, terminate; the best solution found so far is optimal.

Termination

- Termination with a global minimum is guaranteed, but the number of nodes to be examined may be huge. In practice, the algorithm is terminated when an ϵ-optimal solution is obtained.
- Tight lower bounds $\underline{f}_{Y_j}$ are important for quick termination.

Lower Bounds

- One method to obtain lower bounds in the branch-and-bound method is by constraint relaxation (e.g., replace $x_i \in \{0, 1\}$ by $0 \leq x_i \leq 1$)

Lower Bounds

- Another method, called *Lagrangian relaxation*, is based on weak duality. If the subproblem of a node of the branch-and-bound tree has the form

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad g_j(x) \le 0, \qquad j = 1, \ldots, r, \\ &\qquad\qquad\quad\; x \in X, \end{aligned}$$

use as lower bound the optimal dual value

$$q^* = \max_{\mu \ge 0} q(\mu),$$

where

$$q(\mu) = \min_{x \in X} \left\{ f(x) + \sum_{j=1}^{r} \mu_j g_j(x) \right\}.$$

- Essential for applying Lagrangian relaxation is that the dual problem is easy to solve