Convex Optimization II

Lecture 2: Mixed-Integer Programming

Hamed Shah-Mansouri

Department of Electrical Engineering

Sharif University of Technology

1400-2

OUTLINE

- Preliminaries
- Mixed Integer Programming
- Modeling Techniques and Examples
- Solvers
- Summary

PRELIMINARIES

- Linear programming (LP) allows variables to be real values (i.e., $\mathbf{x} \in \mathbf{R}^n$).
- In mixed integer programming, some variables have to be integer values.
- This apparent small difference makes mixed integer programming much harder to solve than LP.
- LP problems can be solved in polynomial time to the size of input.
- Mixed integer programming can take exponential time to the size of input.
- Various problems can be formulated as mixed integer programming problems. Examples include decision making, scheduling, resource management and allocation problems.
- Frankly speaking, where we need to choose between two or a few alternatives (e.g., whether or not to take some specific actions), we need to deal with integer programming problems.

Mixed Integer Programming Problem

- Given matrices A and B, vectors $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$

$$
\begin{aligned}
\text{minimize} \quad & \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y} \\
\text{subject to} \quad & A\mathbf{x} + B\mathbf{y} = \mathbf{b} \\
& \mathbf{x}, \mathbf{y} \succeq \mathbf{0} \\
& \mathbf{x} \text{ integer}
\end{aligned}
$$

- Integer programming problem if there are no continuous variables $\mathbf{y}$.
- Binary (zero-one) integer programming problem if there are no continuous variables $\mathbf{y}$, and $\mathbf{x}$ are restricted to be either 0 or 1.
- Even if there are inequality constraints, we can represent the problem in the above form by adding slack or surplus variables.

Example: The Zero-one Knapsack Problem

- Given n items. The jth item has weight w_j and its value is v_j.
- Given a bound K on the weight that can be carried in a knapsack.
- Objective: select items to maximize the total value.
- Define a binary variable x_j, which is equal to 1 if item j is chosen, and is equal to 0 otherwise.

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} v_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} w_j x_j \leq K \\ & x_j \in \{0, 1\}, \quad j = 1, \ldots, n. \end{aligned}$$

Forcing Constraints

- In discrete optimization problems, certain decisions are dependent.
- Suppose decision A can be made only if decision B has also been made.
- We can introduce binary variables x (respectively, y) equal to 1 if decision A (respectively, B) is chosen, and 0 otherwise.
- The dependence of the two decisions can be modeled by using the constraint

$$x \leq y$$

- That is, if $y = 0$ (decision B is not made), then $x = 0$ (decision A cannot be made).

Relations Between Variables

- If the constraint is of the form

$$\sum_{j=1}^{n} x_j \leq 1,$$

where all variables x_j are binary, then at most one of the variables x_j can be equal to one.

- If the constraint is of the form

$$\sum_{j=1}^{n} x_j = 1,$$

where all variables x_j are binary, then exactly one of the variables x_j is equal to one.

Example: Facility Location Problem

- Given n potential facility locations and a list of m clients who need to be serviced from these locations.
- There is a fixed cost c_j of opening a facility at location j.
- There is a cost d_{ij} of serving client i from facility j.
- Objective: Select a set of facility locations and assign each client to one facility, while minimizing the total cost.

Example: Facility Location Problem (cont.)

- Define binary decision variable y_j for each location j, which is equal to 1 if facility j is selected, and 0 otherwise.
- Define binary decision variable x_{ij}, which is equal to 1 if client i is served by facility j, and 0 otherwise.
- The facility location problem can be formulated as

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} x_{ij} \\ \text{subject to} & \displaystyle\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots m \\ & x_{ij} \leq y_j, \qquad i = 1, \ldots m,\ j = 1, \ldots n \\ & x_{ij}, y_j \in \{0, 1\}, \quad i = 1, \ldots m,\ j = 1, \ldots n. \end{array}$$

Disjunctive Constraints

- Let decision vector $\mathbf{x} \succeq \mathbf{0}$.
- Given two constraints $\mathbf{a}^T\mathbf{x} \geq b$ and $\mathbf{c}^T\mathbf{x} \geq d$, in which vectors $\mathbf{a}, \mathbf{c} \succeq \mathbf{0}$.
- We can model the requirement that at least one of two constraints is satisfied.
- Define a binary variable y and impose the constraints

$$
\begin{aligned}
&\mathbf{a}^T\mathbf{x} \geq yb \\
&\mathbf{c}^T\mathbf{x} \geq (1-y)d \\
&y \in \{0, 1\}.
\end{aligned}
$$

Disjunctive Constraints (cont.)

- Given m constraints $\mathbf{a}_i^T\mathbf{x} \geq b_i$, $i = 1, \ldots, m$, where vector $\mathbf{a}_i \succeq \mathbf{0}$ for each i and the decision vector $\mathbf{x} \succeq \mathbf{0}$.

- We can model the requirement that at least k of m constraints are satisfied.

- Define m binary variables y_i, $i = 1, \ldots, m$, and impose the constraints

$$
\begin{aligned}
& \mathbf{a}_i^T\mathbf{x} \geq b_i y_i, \qquad i = 1, \ldots, m \\
& \sum_{i=1}^{m} y_i \geq k \\
& y_i \in \{0, 1\}, \qquad i = 1, \ldots, m.
\end{aligned}
$$

Restricted Range of Values

- Suppose we want to restrict variable x to take values in a set $\{a_1, \ldots, a_m\}$.

- Define m binary variables y_j, $j = 1, \ldots, m$, and the constraints

$$
\begin{aligned}
& x = \sum_{j=1}^{m} a_j y_j \\
& \sum_{j=1}^{m} y_j = 1 \\
& y_j \in \{0, 1\}, \qquad j = 1, \ldots, m.
\end{aligned}
$$

Set Covering, Set Packing, and Set Partitioning

- Let $M = \{1, \dots, m\}$ be a finite set and $N = \{1, \dots, n\}$.
- Let $M_1, M_2, \dots, M_n$ be a given collection of subsets of M.
  - e.g., the collection may consist of all subsets of size at least k, for $k \leq m$.
- A subset F of N (i.e., $F \subseteq N$) is a cover of M if $\cup_{j \in F} M_j = M$.
- $F \subseteq N$ is a packing of M if $M_j \cap M_k = \emptyset$ for all $j, k \in F, j \neq k$.
- $F \subseteq N$ is a partition of M if it is both a cover and a packing of M.

Set Covering, Packing, and Partitioning (cont.)

- In set covering problem, c_j is the cost of set M_j. We seek a minimum-cost cover. The weight of a subset F of N is $\sum_{j \in F} c_j$.
- In set packing problem, however, c_j is the weight or value of set M_j and we seek a maximum-weight packing.
- In set partitioning problem, both minimization and maximization versions are possible.
- To formulate them as integer programming problems, let A be the $m \times n$ incidence matrix of the family $\{M_j \mid j \in N\}$ whose entries are given by

$$a_{ij} = \begin{cases} 1, & \text{if } i \in M_j \\ 0, & \text{otherwise.} \end{cases}$$

- Let decision variable $x_j, j \in N$, which is equal to 1 if $j \in F$, and 0 otherwise. Let $\mathbf{x} = (x_1, \ldots, x_n)$. Then, F is a cover, packing, partition if and only if

$$A\mathbf{x} \succeq \mathbf{1}, \qquad A\mathbf{x} \preceq \mathbf{1}, \qquad A\mathbf{x} = \mathbf{1},$$

  where $\mathbf{1}$ is an m-dimensional vector with all components equal to 1.
- e.g., Try $m = 4$ and $|M_j| \geq 2$, where $|\,.\,|$ denotes the cardinality of the set.

Set Covering Problem Example: Facility Location

- Given a set of potential sites $N = \{1, \dots, n\}$ for the location of installing base stations (or wireless access points).
- A base station placed at site j costs c_j.
- Given a set of wireless users $M = \{1, \dots, m\}$ that have to be served.
- The subset of users that can be served by base station located at site j is M_j. For example, M_j might be the set of users that are within 500 meters from the base station located at site j.
- The problem of choosing a minimum-cost set of locations for the base stations such that each user is within 500 meters from some base stations is a set covering problem.
- There are many other applications of this type, including the assignment and scheduling problems, routing problems.

Algorithms

- Exact algorithms: Guaranteed to find an optimal solution, but may take an exponential number of iterations.
  - e.g., cutting plane, branch-and-bound, branch-and-cut, and dynamic programming methods.
- Approximation algorithms can provide in polynomial time a sub-optimal solution together with a bound on the degree of sub-optimality.
  - e.g., ϵ-approximation algorithm for zero-one knapsack problem.
- Heuristic algorithms can provide a sub-optimal solution, but without a guarantee on its quality.
  - e.g., local search methods, simulated annealing.

SOLVERS FOR MILP

- CPLEX is a commercial optimization software package. It can solve various problem types including LP problems, mixed integer programming problems, and quadratic programming problems. It is available with several modeling systems including AMPL, MATLAB, MPL, and TOMLAB.

- MOSEK is another commercial optimization software package. It can also solve LP, mixed integer problems, convex problems, etc. Interfaces with MOSEK include C/C++, Java, MATLAB.

- Version 2.0 of CVX supports mixed integer disciplined convex programming (MIDCP). Website: http://cvxr.com/cvx/

- Matlab: With specific function contributed by users. Example: http://www.mathworks.com/matlabcentral/fileexchange/6990-mixed-integer-lp.

Solving Binary Integer Programming Problem Using MATLAB

Example:

$$\begin{aligned} \text{minimize} \quad & -9x_1 - 5x_2 - 6x_3 - 4x_4 \\ \text{subject to} \quad & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 9, \\ & x_3 + x_4 \leq 1, \\ & -x_1 + x_3 \leq 0, \\ & -x_2 + x_4 \leq 0, \\ & x_1, x_2, x_3, x_4 \in \{0, 1\} \end{aligned}$$

In Matlab, we have

$$\begin{aligned} & f = [-9; -5; -6; -4]; \\ & A = [6\ 3\ 5\ 2;\ 0\ 0\ 1\ 1;\ -1\ 0\ 1\ 0;\ 0\ \ -1\ 0\ 1]; \\ & b = [9; 1; 0; 0]; \\ & x = intlinprog(f, intcon, A, b) \end{aligned}$$

- https://www.mathworks.com/help/optim/ug/intlinprog.html

SUMMARY

- Integer programming (IP) problems arise frequently when some (or all) decision variables must be restricted to integer values.
- They are many applications involving yes-or-no decisions that can be represented by binary (0-1) variables.
- When solving IP problems, the most important determinants of computation time are the number of integer variables and whether the problem has some special structure that can be exploited.
- For a fixed number of integer variables, BIP problems generally are much easier to solve than problems with general integer variables, but adding continuous variables (MIP) may not increase computation time substantially.