Mixed Integer Programming

Edited by Neng-Fa Zhou

Brooklyn College & Graduate Center
The City University of New York

MIP and IP

- MIP
  - Mixed Integer Programming involves optimization problems where some variables are constrained to be integers.
- IP
  - All the variables are integer
- BIP
  - All the variables are binary (0/1)

LP Relaxation

- **Definition**
  - Relax integer variables to allow them to be continuous.
- **Purpose**
  - To find a bound for the objective function.
  - If the solution happens to satisfy the integer constraints, then it is an optimal solution of the original MIP.

Branch-and-Bound Method

- **Overview:** A systematic method for solving MIPs by exploring branches of possible solutions.
- **Steps:**
  - Solve the LP relaxation.
  - If the solution is integer, it's optimal.
  - If not, branch on a variable (create subproblems).
  - Solve subproblems and prune branches that cannot yield better solutions.
  - Fathom (disregard) a subproblem if
    - Its bound is worse than the current best
    - It is infeasible
    - Its solution is already integer

Summary of the BIP Branch-and-Bound Algorithm

Initialization: Set $Z^* = -\infty$. Apply the bounding step, fathoming step, and optimality test described below to the whole problem. If not fathomed, classify this problem as the one remaining "subproblem" for performing the first full iteration below.

Steps for each iteration:

1. *Branching:* Among the *remaining* (unfathomed) subproblems, select the one that was created *most recently.* (Break ties according to which has the *larger bound.*) Branch from the node for this subproblem to create two new subproblems by fixing the next variable (the branching variable) at either 0 or 1.
2. *Bounding:* For each new subproblem, apply the simplex method to its LP relaxation to obtain an optimal solution, including the value of Z, for this LP relaxation. If this value of Z is not an integer, round it down to an integer. (If it was already an integer, no change is needed.) This integer value of Z is the *bound* for the subproblem.
3. *Fathoming:* For each new subproblem, apply the three fathoming tests summarized above, and discard those subproblems that are fathomed by any of the tests.

Optimality test: Stop when there are *no remaining* subproblems; the current *incumbent* is optimal.[4] Otherwise, return to perform another iteration.

An Example

Maximize $Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$,

subject to

$$
\begin{aligned}
(1) \quad & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 \\
(2) \quad & x_3 + x_4 \leq 1 \\
(3) \quad & -x_1 + x_3 \leq 0 \\
(4) \quad & -x_2 + x_4 \leq 0
\end{aligned}
$$

and

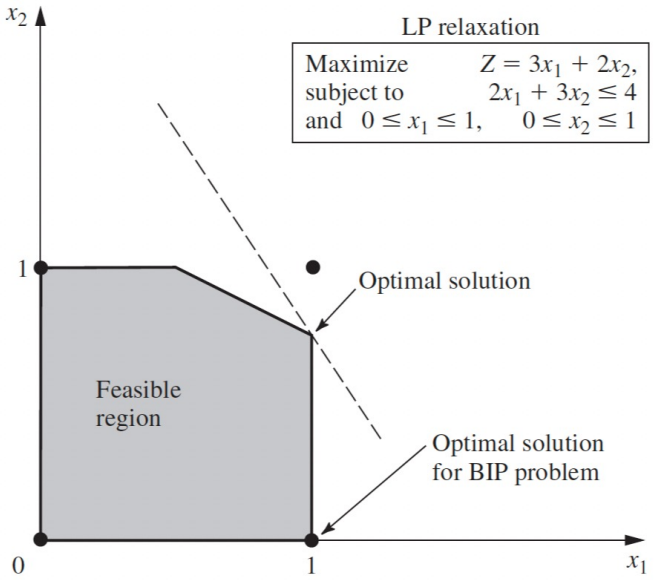
(5) x_j is binary, for $j = 1, 2, 3, 4$.

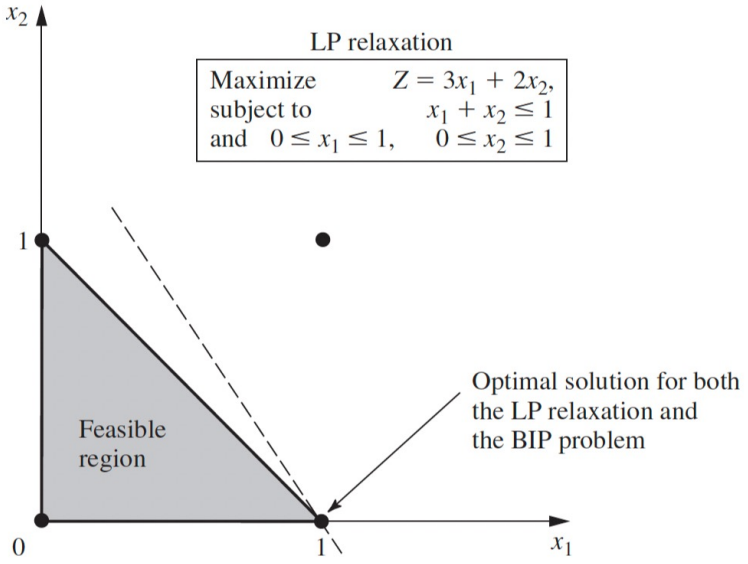
Cutting Planes Method

- **Definition:** A technique to tighten the feasible region of the LP relaxation by adding additional constraints (cutting planes).

- **Purpose:** To eliminate non-integer solutions without excluding any feasible integer solutions.

Example

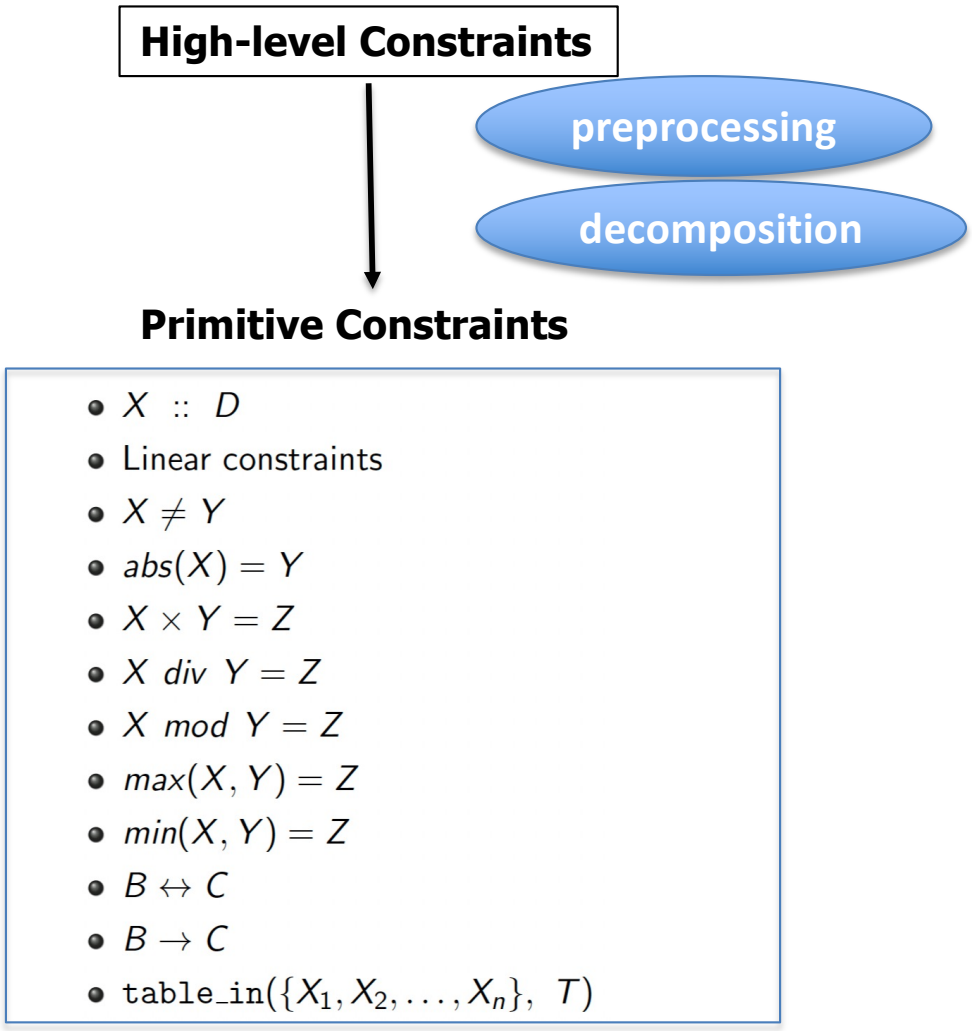

LP relaxation
Maximize $Z = 3x_1 + 2x_2$,
subject to $2x_1 + 3x_2 \le 4$
and $0 \le x_1 \le 1$, $0 \le x_2 \le 1$
x_2
1
0
1
x_1
Feasible region
Optimal solution
Optimal solution for BIP problem


LP relaxation
Maximize $Z = 3x_1 + 2x_2$,
subject to $x_1 + x_2 \le 1$
and $0 \le x_1 \le 1$, $0 \le x_2 \le 1$
x_2
1
0
1
x_1
Feasible region
Optimal solution for both the LP relaxation and the BIP problem

Encoding CSP into MIP

- MIP for combinatorial search
  - Symplex method
  - LP relaxation
  - Branch-and-bound
  - Cutting planes method
- MIP solvers support nonlinear constraints
- Encoding nonlinear constraints into linear ones remains to be important

CSP to MIP



- Constraints are made to be arc-consistent or interval consistent
- No primitive constraints are duplicated

$X :: D$

- Let $D = L_1..U_1 \cup L_2..U_2 \cup \ldots \cup L_m..U_m$
- Introduce a binary variable B_i for each interval $L_i..U_i$
  - $B_i \rightarrow X \geq L_i$
  - $B_i \rightarrow X \leq U_i$
- Translate $X :: D$ to $B_1 + B_2 + \ldots + B_m = 1$

X≠Y and abs(X) = Y

- **X≠Y**
  - B → X > Y
  - ~B → X < Y

≠

- abs(X) = Y
  - T = -X
  - Y = max(X, T)

$X \times Y = Z$, $X \; div \; Y = Z$, $X \; mod \; Y = Z$

- Let X's binary representation be $\langle X_{n-1}, \ldots, X_1, X_0 \rangle$
- Translate $X \times Y = Z$ to:
  - $Z = 2^{n-1} T_{n-1} + \ldots + 2 T_1 + T_0$ ← Binary expansion
    - $X_i \rightarrow T_i = Y$
    - $\neg X_i \rightarrow T_i = 0$
- Convert $X \; div \; Y = Z \; (X \geq 0, Y > 0)$ to
  - $X = Y \times Z + R$
    - $0 \leq R < Y$
- Convert $X \; mod \; Y = Z \; (X \geq 0, Y > 0)$ to
  - $X = Y \times Q + Z$
    - $0 \leq Z < Y$

min(X,Y) = Z

- **min(X,Y) = Z**
  - Z $\leq X$
  - Z $\leq Y$
  - B1 $\rightarrow$ X $\leq$ Z
  - B2 $\rightarrow$ Y $\leq$ Z
  - B1+B2 $\geq Z$

$\neq$

$B \rightarrow X \leq Y,\ B \leftrightarrow X \leq Y$

- $B \rightarrow X \leq Y$
  - $X - (1-B)M \leq Y$

The big-M method

- $B \leftrightarrow X \leq Y$
  - $B \rightarrow X \leq Y$
  - $\sim B \rightarrow X > Y$

$\neq$

Exercise -1

Given a set of foods, each of which has given nutrient values, a cost per serving, and a minimum limit for each nutrient, the objective of the diet problem is to select the number of servings of each food to consume to minimize the cost of the food while meeting the nutritional constraints. The following gives an example. In this example, a diet is required to contain at least 500 calories, 6 ounces of chocolate, 10 ounces of sugar, and 8 ounces of fat.

Type of Food	Calories	Chocolate (oz.)	Sugar (oz.)	Fat (oz.)	Price (cents)
Chocolate Cake (1 slice)	400	3	2	2	50
Chocolate ice cream (1 scoop)	200	2	2	4	20
Cola (1 bottle)	150	0	4	1	30
Pineapple cheesecake (1 piece)	500	0	4	5	80
Limits	500	6	10	8	–

- Model the problem.
- Use a MIP solver to solve the problem.

Exercise-2

Use the branch-and-bound algorithm to iteratively solve the following BIP problem:

$$\text{Maximize} \qquad Z = 2x_1 - x_2 + 5x_3 - 3x_4 + 4x_5,$$

subject to

$$\begin{aligned} 3x_1 - 2x_2 + 7x_3 - 5x_4 + 4x_5 &\leq 6 \\ x_1 - x_2 + 2x_3 - 4x_4 + 2x_5 &\leq 0 \end{aligned}$$

and

$$x_j \text{ is binary,} \qquad \text{for } j = 1, 2, \ldots, 5.$$

Exercise-3

Consider the following program in Picat:

```
import sat.

main =>
    G = graph(),
    N = len(G),
    Vars = new_list(N),
    Vars :: 0..1,
    foreach (U in 1..N-1, V in U+1..N)
       Vars[U] #/\ Vars[V] #=> G[U,V]
    end,
    solve($[max(sum(Vars))],Vars),
    print(Vars).

graph =
    {{0,1,0,0,1,0},
     {1,0,1,0,1,0},
     {0,1,0,1,0,0},
     {0,0,1,0,1,1},
     {1,1,0,1,0,0},
     {0,0,0,1,0,0}}.
```

Translate the program into a linear integer program, and use a MIP solver to solve it.