Advanced Optimization Methods

Report

Integer and Binary Integer Linear Programming

Author:
Piotr Pawełko
Krzysztof Miler

Laboratory date:
28.04.2022

1. Integer linear programming

The task is to solve problem using *branch and bound method* in which solution must be integers. The problem is:

Find the maximum of the objective function:

$$max \rightarrow c^T x$$

under constraints:

$$Ax \leq b$$

$$x \geq 0,\ x \epsilon Z$$

Set of parameter:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1.1567 & 1 \\ 1.4289 & 1 \\ 0.4933 & 1 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 0 \\ 2.2886 \\ 6.6232 \\ 2.7572 \end{bmatrix}$$

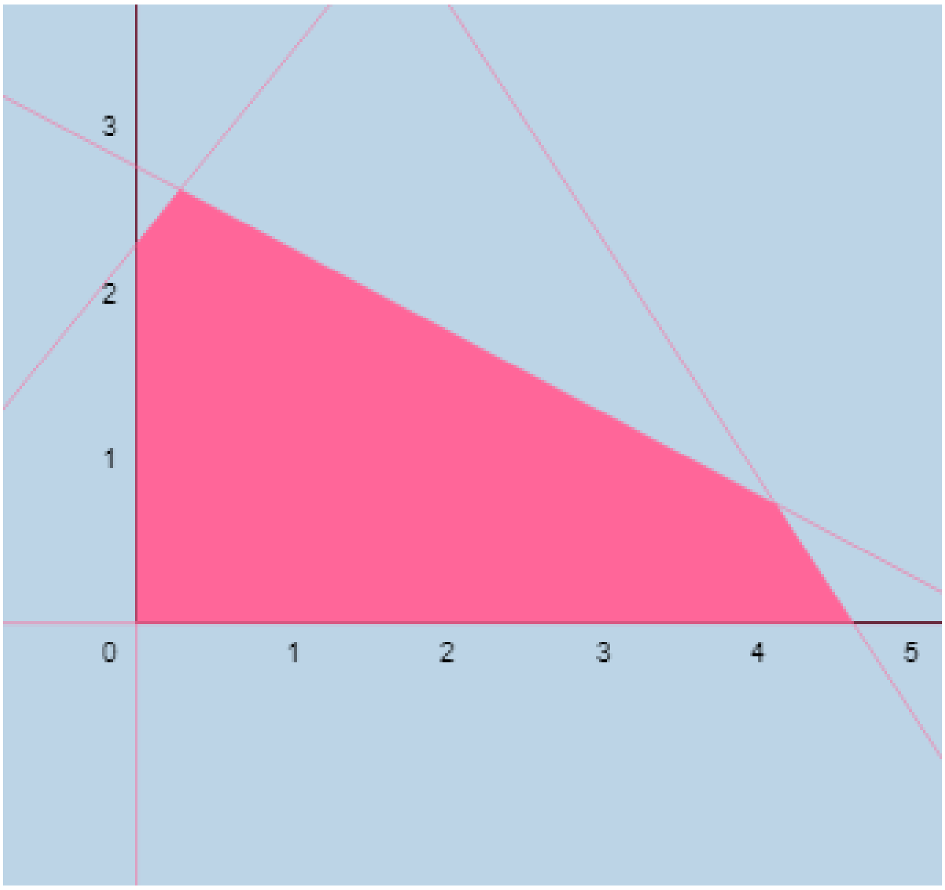

Figure 1 Feasible region

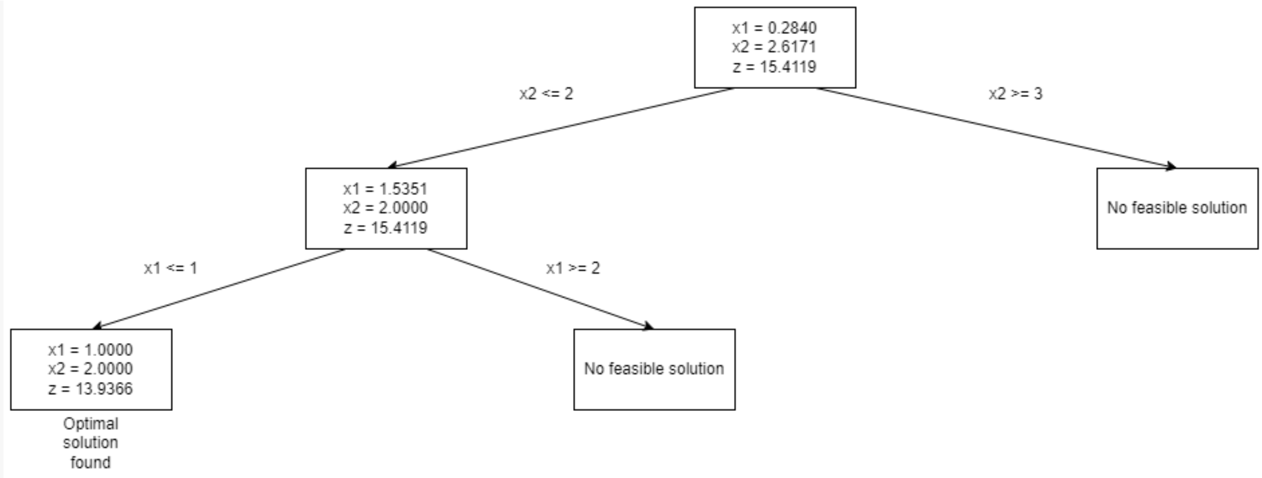

Figure 2 Solution tree

Code for solving:

```
ObjF = [2.7572 5.5897];

Cons = [-1.9785 4.6353 5.5897 2.2886 2.7572 6.6232];
f = -1 * ObjF;
A = [-1 0; 0 -1; Cons(1,4)/Cons(1,1) 1; Cons(1,6)/Cons(1,2) 1;
Cons(1,5)/Cons(1,3) 1];
b = [0; 0; Cons(1,4); Cons(1,6); Cons(1,5)];
options = optimoptions('linprog','Display','none');
[x, fval] = linprog(f, A, b, [], [], [], [], options);
x = 2×1
0.2840
2.6171
fval = -15.4119
% x2 <= 2 and x2 >= 3
A1 = [A; 0 1];
b1 = [b; floor(x(2))];
A2 = [A; 0 -1];
b2 = [b; -ceil(x(2))];
[roz1, fval1] = linprog(f, A1, b1, [], [], [], [], options);
roz1 = 2×1
1.5351
2.0000
fval1 = -15.4119
[roz2, fval2] = linprog(f, A2, b2, [], [], [], [], options); % no feasible
% x1 <= 1 and x1 >= 1
A1 = [A1; 1 0];
b1 = [b1; floor(roz1(1))];
A2 = [A2; -1 0];
b2 = [b2; -ceil(roz1(1))];
[roz1, fval1] = linprog(f, A1, b1, [], [], [], [], options);
roz1 = 2×1
1
2
fval1 = -13.9366
```

```
[roz2, fval2] = linprog(f, A2, b2, [], [], [], [], options); % no feasible
```

Solution found by MATLAB toolbox by using *intlinprog* function:

```
LP:                Optimal objective value is -15.411921.

Heuristics:        Found 2 solutions using ZI round.
                   Upper bound is -13.861300.
                   Relative gap is 0.51%.

Cut Generation:    Applied 2 Gomory cuts, and 1 mir cut.
                   Lower bound is -13.936600.
                   Relative gap is 0.00%.


Optimal solution found.

Intlinprog stopped at the root node because the objective 
value). The intcon variables are integer within tolerance,


x =

    1.0000
    2.0000


fval =

  -13.9366
```

Figure 3 Matlab toolbox result

2. Binary integer linear programming

The task is to solve binary integer linear programming problem, which means the solution must binary values. The problem is called knapsack problem. We are given *N* items, each if weight w_i and associated value v_i. We should select items to be put into a knapsack so that their combined value is the largest and at the same time their total weight does not exceed a bound w_{max} imposed on the weight that we can carry.

$$max \leftarrow v = \sum_{i=1}^{N} x_i v_i \qquad \sum_{i=1}^{N} x_i w_i \le w_{max}$$

$$x_i \epsilon \{0,1\}, i = 1, \dots, N.$$

Under constraints:

v_i = [12 23 8 6 11], w_i = [12 23 8 6 11], w_{max} = 16

Code with results:

```
values = [12, 23, 8, 6, 11, 10, 27, 2];
weights = [2.2, 4, 2.5, 2.1, 1.8, 3, 5, 0.6];
maxWeight = 16;
f = -values;        % max
A = weights;
b = maxWeight;
intcon = 1:8;
lb = zeros(8,1);
ub = ones(8,1);

x = intlinprog(f,intcon,A,b,[],[],lb,ub)
```

```
LP:                Optimal objective value is -83.000000.

Heuristics:        Found 1 solution using ZI round.
                   Upper bound is -75.000000.
                   Relative gap is 10.53%.

Cut Generation:    Applied 1 mir cut.
                   Lower bound is -83.000000.
                   Relative gap is 0.00%.


Optimal solution found.

Intlinprog stopped at the root node because the objective value is within a gap tolerance of the optimal val
value). The intcon variables are integer within tolerance, options.IntegerTolerance = 1e-05 (the default val

x = 8×1
      1.0000
      1.0000
           0
           0
      1.0000
      1.0000
      1.0000
           0
```

Figure 4 Result for knapsack problem

3. Comments

Branch and bound method is used for finding integer solution for maximization problem given in first task. At first we relaxed problem from integer constraint and used *linprog* function for find overall optimal solution. For next step we choose to add next constraints based on bigger decimal value of variables which split into 2 problems. For one of this problems there was no feasible solution and for another one we found optimal value with one integer value, this solution we also split into 2 problems and one of them we achieved 2 integers values of variables and for the second one there was no feasible solution. The final result was x1 = 1, x2 = 2, with function value of 13.9366. This same exact solution was obtained by *intlinprog* function, which solves integer linear problems. For the second task we had to find binary solution for a knapsack problem. We had all values and weights for each item and maximum weight we can carry in a knapsack. All these values we used in a *intlinprog* function which shows *figure* 4. The result was that we can put first, second, fifth, sixth and seventh item out of eight into a knapsack.