Exercise 2. Global Optimization

Dongho Kang
16-948-598

March 19, 2017

MATLAB R2016b version was used for coding and testing:

MathWorks, MATLAB R2016b (9.1.0.441655)
64-bit (maci64)

The ***code*** directory contains followings:

- ***main.m*** Script .m file for exercise.
- ***functions dir*** Function .m files
  - **SolveWithLP** Function .m file for solving a problem with the LP and simple testing.
  - **SplitProblem** Function .m file for split a parent problem into two children problems.
  - **FindBestCandidate** Function .m file for finding the best candidate between two children node.
  - **PushToStack** Function .m file for stack push operation.
  - **PopFromStack** Function .m file for stack pop operation.
  - **NewProblem** Function .m file for creating new problem (instance of struct).
  - **FindInliers** Function .m file for finding inliers with a input model.
  - and other sub-functions: **VisualizeMatch**, **SaveOptHistory**, **ComputeInlierLb**
- ***data dir*** data and image files provided.

For running exercise, adjust threshold parameter (default of 3 pixel) and run the ***main.m*** on the MATLAB environment. More details are stated in the *Running* section.

1 EXERCISE : BRANCH AND BOUND FOR CONSENSUS SET MAXIMIZATION

1.1 DESCRIPTION

In this exercise, branch and bound algorithm based on depth-first search(DFS) was implemented. The implementation steps are as follows:

- derivation of the problem formulation in the canonical form of the linear programming
- branch and bound algorithm implementation
- logging and plotting the result of branch and bound algorithm

1.1.1 LINEAR PROGRAMMING FORMULATION

Let the set S of the input data be partitioned into an inlier-set $S_I \subseteq S$ and an outlier-set $S_O = S \setminus S_I$. Note that

- the model $\Theta = (T_x, T_y)$ where T_x and T_y represent the translation along the x and y axis.
- the i-th input correspondence is (p_i, p_i') where p_i and p_i' represent the points in the left and right images: $p_i = (x_i, y_i)$ and $p_i = (x_i', y_i')$. There are n input correspondences.

then the consensus set maximization problem can be formulated as follows:

$$\begin{aligned} \max_{\Theta, S_I} \quad & card(S_I) \\ s.t. \quad & \left| x_i + T_x - x_i' \right| \leq \delta, \forall i \in S_I \subseteq S \\ & \left| y_i + T_y - y_i' \right| \leq \delta, \forall i \in S_I \subseteq S \end{aligned} \tag{1.1}$$

To solve the problem by linear programming, alternative formulation using z_i and the relaxation can be applied to (1.1):

$$\max_{\Theta, \mathbf{z}} \quad \sum_{i=1}^{N} z_i \tag{1.2}$$

$$s.t. \quad z_i \left| x_i + T_x - x_i' \right| \leq z_i \delta, \forall i \in S_I \subseteq S \tag{1.3}$$

$$and \quad z_i \left| y_i + T_y - y_i' \right| \leq z_i \delta, \forall i \in S_I \subseteq S \tag{1.4}$$

$$and \quad z_i \in [0, 1], \quad \forall i = 1 \ldots N \tag{1.5}$$

$$and \quad \underline{T_x} \leq T_x \leq \overline{T_x}, \quad \underline{T_y} \leq T_y \leq \overline{T_y}, \quad \forall i = 1 \ldots N \tag{1.6}$$

(1.3) and (1.4) can be expressed into:

$$\begin{aligned} -z_i \delta \leq z_i (x_i + T_x - x_i') \leq z_i \delta \\ -z_i \delta \leq z_i (y_i + T_y - y_i') \leq z_i \delta \end{aligned} \tag{1.7}$$

Now, introducing the auxiliary variables $w_{ix} = z_i T_x$ and $w_{iy} = z_i T_y$ to avoid the bilinear terms, (1.6) can be relaxed by concave and convex envelopes:

$$
\begin{aligned}
w_{ix} = z_i T_x \geq \max(\underline{z_i} T_x + \underline{T_x} z_i - \underline{z_i}\,\underline{T_x}, \overline{z_i} T_x + \overline{T_x} z_i - \overline{z_i}\,\overline{T_x}) \\
w_{ix} = z_i T_x \leq \min(\overline{z_i} T_x + \underline{T_x} z_i - \overline{z_i}\,\underline{T_x}, \underline{z_i} T_x + \overline{T_x} z_i - \underline{z_i}\,\overline{T_x})
\end{aligned}
\tag{1.8}
$$

max and min is not a linear function, thus changed (1.8) into the following forms:

$$
\begin{aligned}
w_{ix} \geq \underline{z_i} T_x + \underline{T_x} z_i - \underline{z_i}\,\underline{T_x} \\
w_{ix} \geq \overline{z_i} T_x + \overline{T_x} z_i - \overline{z_i}\,\overline{T_x} \\
w_{ix} \leq \overline{z_i} T_x + \underline{T_x} z_i - \overline{z_i}\,\underline{T_x} \\
w_{ix} \leq \underline{z_i} T_x + \overline{T_x} z_i - \underline{z_i}\,\overline{T_x}
\end{aligned}
\tag{1.9}
$$

changed (1.8) into linear programming constraint form:

$$
\begin{aligned}
\underline{z_i} T_x + \underline{T_x} z_i - w_{ix} \leq \underline{z_i}\,\underline{T_x} \\
\overline{z_i} T_x + \overline{T_x} z_i - w_{ix} \leq \overline{z_i}\,\overline{T_x} \\
-\overline{z_i} T_x - \underline{T_x} z_i + w_{ix} \leq -\overline{z_i}\,\underline{T_x} \\
-\underline{z_i} T_x - \overline{T_x} z_i + w_{ix} \leq -\underline{z_i}\,\overline{T_x}
\end{aligned}
\tag{1.10}
$$

likewise, $w_i y$:

$$
\begin{aligned}
\underline{z_i} T_y + \underline{T_y} z_i - w_{iy} \leq \underline{z_i}\,\underline{T_y} \\
\overline{z_i} T_y + \overline{T_y} z_i - w_{iy} \leq \overline{z_i}\,\overline{T_y} \\
-\overline{z_i} T_y - \underline{T_y} z_i + w_{iy} \leq -\overline{z_i}\,\underline{T_y} \\
-\underline{z_i} T_y - \overline{T_y} z_i + w_{iy} \leq -\underline{z_i}\,\overline{T_y}
\end{aligned}
\tag{1.11}
$$

and finally, as $w_{ix} = z_i T_x$ and $w_{iy} = z_i T_y$, (1.7) can be expressed into:

$$
\begin{aligned}
z_i x_i + w_{ix} - z_i x_i' - z_i \delta \leq 0 \\
-z_i x_i - w_{ix} + z_i x_i' - z_i \delta \leq 0 \\
z_i y_i + w_{iy} - z_i y_i' - z_i \delta \leq 0 \\
-z_i y_i - w_{iy} + z_i y_i' - z_i \delta \leq 0
\end{aligned}
\tag{1.12}
$$

Now, as MATLAB function **linprog** solve following problem,

$$
\begin{aligned}
\min_x c^T \mathbf{x} \\
s.t. A\mathbf{x} \leq b \\
l_b \leq \mathbf{x} \leq u_b
\end{aligned}
\tag{1.13}
$$

set the unknown vector, coefficient vector, and l_b, u_b vectors for linear programming to

$$
\begin{aligned}
\mathbf{x} &= (T_x, T_y, z_1, \dots, z_n, w_{1x}, \dots, w_{nx}, w_{1y}, \dots, w_{ny})^T \\
l_b &= (\underline{T_x}, \underline{T_y}, \underline{z_1}, \dots, \underline{z_n}, -\infty, \dots, -\infty)^T \\
u_b &= (\overline{T_x}, \overline{T_y}, \overline{z_1}, \dots, \overline{z_n}, \infty, \dots, \infty)^T \\
c &= (0, 0, -1, \dots, -1, 0, \dots, 0)^T
\end{aligned}
\tag{1.14}
$$

Note that coefficient vector c is $c^T\mathbf{x} = -z_1 - z_2 - \cdots - z_n$. This is for making the maximizing problem into the minimizing problem. Besides, $-\infty \le w_{ix} \le \infty$ and $-\infty \le w_{iy} \le \infty$, because there is no either lower nor upper bound for w_{ix} and w_{iy}.
Lastly, (1.9), (1.10) and (1.11) was changed into the form of (1.12). Thus, matrix A and vector b are as follows:

$$
A_i = \left[\begin{array}{cc|ccccc|ccccc|ccccc}
0 & 0 & 0 & \dots & (x_i - x_i' - \delta) & \dots & 0 & 0 & \dots & 1 & \dots & 0 & 0 & \dots & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & (-x_i + x_i' - \delta) & \dots & 0 & 0 & \dots & -1 & \dots & 0 & 0 & \dots & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & (y_i - y_i' - \delta) & \dots & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 1 & \dots & 0 \\
0 & 0 & 0 & \dots & (-y_i + y_i' - \delta) & \dots & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & -1 & \dots & 0 \\
\underline{z_i} & 0 & 0 & \dots & \underline{T_x} & \dots & 0 & 0 & \dots & -1 & \dots & 0 & 0 & \dots & 0 & \dots & 0 \\
\overline{z_i} & 0 & 0 & \dots & \overline{T_x} & \dots & 0 & 0 & \dots & -1 & \dots & 0 & 0 & \dots & 0 & \dots & 0 \\
-\overline{z_i} & 0 & 0 & \dots & -\underline{T_x} & \dots & 0 & 0 & \dots & 1 & \dots & 0 & 0 & \dots & 0 & \dots & 0 \\
-\underline{z_i} & 0 & 0 & \dots & -\overline{T_x} & \dots & 0 & 0 & \dots & 1 & \dots & 0 & 0 & \dots & 0 & \dots & 0 \\
0 & \underline{z_i} & 0 & \dots & \underline{T_y} & \dots & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & -1 & \dots & 0 \\
0 & \overline{z_i} & 0 & \dots & \overline{T_y} & \dots & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & -1 & \dots & 0 \\
0 & -\overline{z_i} & 0 & \dots & -\underline{T_y} & \dots & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 1 & \dots & 0 \\
0 & -\underline{z_i} & 0 & \dots & -\overline{T_y} & \dots & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 1 & \dots & 0
\end{array}\right]
$$

$$
b_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \underline{z_i}\,\underline{T_x} \\ \overline{z_i}\,\overline{T_x} \\ -\overline{z_i}\,\underline{T_x} \\ -\underline{z_i}\,\overline{T_x} \\ \underline{z_i}\,\underline{T_y} \\ \overline{z_i}\,\overline{T_y} \\ -\overline{z_i}\,\underline{T_y} \\ -\underline{z_i}\,\overline{T_y} \end{bmatrix}
\qquad
A = \left[\begin{array}{c} A_1 \\ \hline A_2 \\ \hline \vdots \\ \hline A_n \end{array}\right]
\qquad
b = \left[\begin{array}{c} b_1 \\ \hline b_2 \\ \hline \vdots \\ \hline b_n \end{array}\right]
$$

Note that $\underline{z_i} = 0$ and $\overline{z_i} = 1$. For avoiding a loop, A and b was reformed in the MATLAB implementation.

1.1.2 Branch and bound

Implemented branch and bound algorithm for finding optimal solution of the consensus set maximization problem. The depth-first search based on stack data structure was used. The stack has two operation **PopFromStack** and **PushToStack**.

The pseudo-code of the implementation is as follows:

Algorithm 1 Branch and bound

```
P0 ← whole problem space
init. opt ← variable for optimal solution (bound of inliers)
init. stack
push(stack, P0)
while stack is not empty do
    Parent = pop(stack)
    if Parent.ObjUpperBound < opt.LowerBound then
        continue; ← bad bound. does not contain optimum for sure
    end if
    if Parent.ObjLowerBound ≥ opt.LowerBound then
        opt = (Parent.ObjUpperBound, Parent.ObjLowerBound)
    end if
    if Parent.ObjUpperBound − Parent.ObjLowerBound < 1 then
        continue; ← Parent is a leaf node thus do not split
    end if
    (LeftChild, RightChild) = split(Parent)
    solveLP(LeftChild, RightChild) ← solve both problems by LP and test
    (BetterChild, WorseChild) = findbetter(LeftChild, RightChild)
    push(stack, WorseChild)
    push(stack, BetterChild) ← BetterChild should be on the top of stack
end while
opt ← now optimal solution
```

The several criteria/strategies of implementation were defined as follows

1. push the original problem $P0$ to the problem stack.

2. start iteration. The iteration of branch and bound terminates when the problem stack is empty

3. bad bound check criteria:
   - m^* is the highest lower bound of the number of inliers obtained so far.
   - if the upper cardinality bound of a problem $< m^*$ then there's no optimum for sure.
   - pop the problem from the stack without splitting.

4. optimal solution update criteria :
   - if the lower cardinality bound of a problem lb is $lb \geq$ the lower cardinality bound of the optimal solution found so far, then update the optimal solution as lb.

5. convergence criteria: if the lower and upper cardinality bound are nearer than 1, stop split.
   - iteration can be terminated because the cardinality bound converged to optimum.
   - but here, to check whether obtained optimum is indeed global optimum, do not terminate iteration but check remaining problems by continuing depth-first search.

6. split the problem space and branching: splitting in half along the longest dimension.

7. solve children problems and obtain the cardinality bound by the LP and simple test
   - compute the upper bound of the number of inliers: the LP with the formulation of (1.1.1) was exploited.
   - compute the lower bound of the number of inliers: simple test with the model Θ obtained by LP. Details are stated below.

8. after solving children problems, push *worse* child problem to the stack first, and then push *better* child:
   - *better* child is a problem with a larger lower cardinality bound. If two children have the same lower cardinality bound, then choose one with a larger upper cardinality bound.
   - as pushing two children, top of the stack is the *better* child now. Thus the *better* child will be popped in the next iteration step.
   - this strategy is for exploring *better* problem first for reducing running time.

9. iteration terminates with the global optimal cardinality bound.

SOLVING A PROBLEM The problem solving step is getting upper bound and lower bound of the number of inliers. Given a certain problem, solves the problem with the LP defined in (1.1.1). Then, the optimal solution of the relaxed problem (T_x^*, T_y^*) and the objective cost $c^T\mathbf{x}^*$ are obtained from:

$$\mathbf{x}^* = (T_x^*, T_y^*, z_1^*, \ldots, z_n^*, w_{1x}^*, \ldots, w_{nx}^*, w_{1y}^*, \ldots, w_{ny}^*)^T \tag{1.15}$$

$$c^T\mathbf{x}^* = -z_1^* - z_2^* - \cdots - z_n^* \tag{1.16}$$

The (-)objective cost, $-c^T\mathbf{x}^*$ can be set as the upper bound of inliers since the the number of inliers of that problem space never exceeds $-c^T\mathbf{x}^*$.

Besides, the lower bound of inliers can be obtained by simple test with (T_x^*, T_y^*) and is set as the $card(S_I)$ as follows:

$$\begin{aligned} & card(S_I) \\ s.t.\quad & \left| x_i + T_x^* - x_i' \right| \le \delta, \forall i \in S_I \subseteq S \\ & \left| y_i + T_y^* - y_i' \right| \le \delta, \forall i \in S_I \subseteq S \end{aligned} \tag{1.17}$$

This was implemented as the function **SolveWithLP**.

1.2 RUNNING

Run the script ***main.m*** after setting the parameters.

- *threshold* : threshold for inliers. Default is 3 (px).
- *padding* : the width of black padding between left and right image for figure(1). Default is 10 (px).

1.3 RESULT

The optimal solution (T_x^*, T_y^*) found by branch and bound for the given problem is as follows:

Table 1.1: The global optimal solution obtained by branch and bound

T_x^*	T_y^*	cardinality bound
-232.00	-156.81	(15, 15.449)

Inlier indices are $(3, 8, 9, 15, 16, 20, 26, 31, 32, 34, 35, 40, 42, 45, 51)$. The identified inliers and plot of cardinality bounds are as follows. The red lines indicate outliers and the green lines indicate inliers:

Figure 1.1: The identified inlier and outlier correspondences

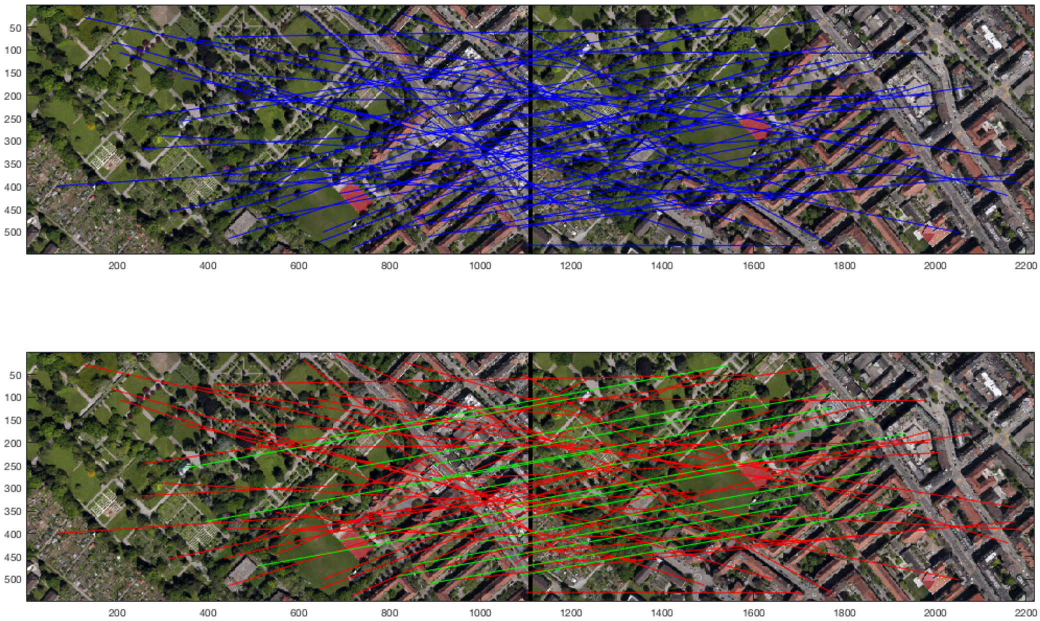
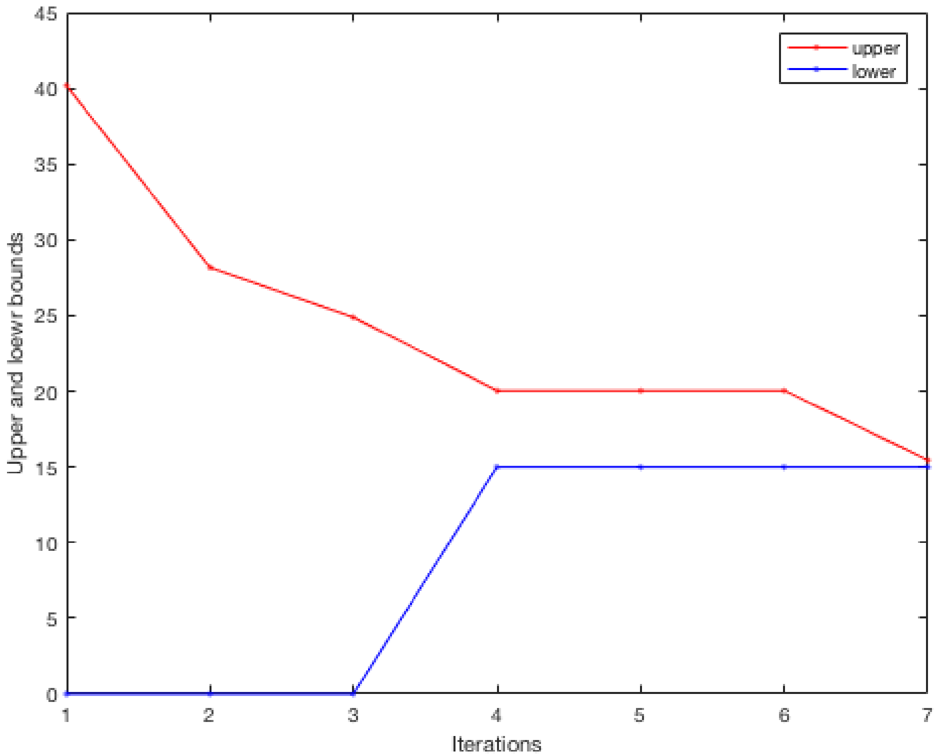


Figure 1.2: The convergence of the cardinality bounds



The branch and bound iteration can be described as the following binary tree illustration:

Figure 1.3: An illustration of DFS tree: one step before exploring $P11$

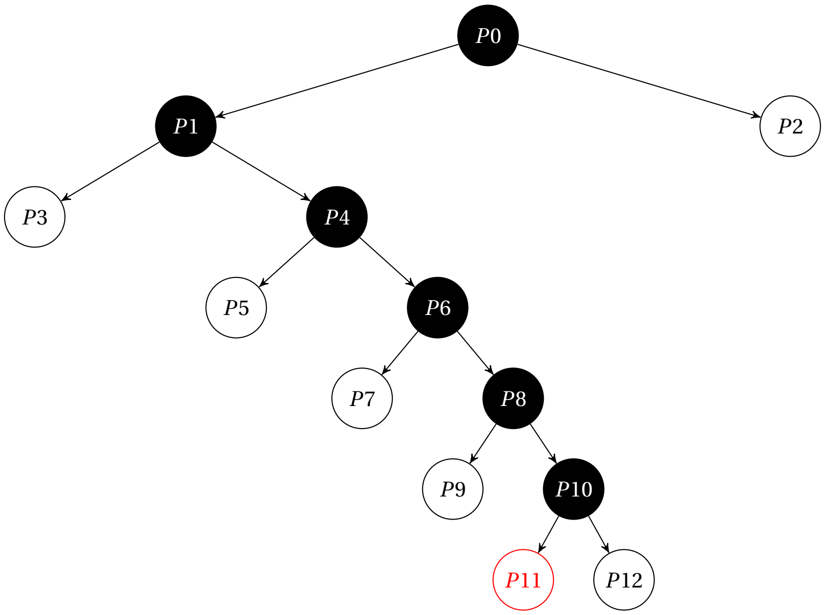


Figure 1.4: An illustration of DFS stack: one step before popping $P11$

P2	P3	P5	P7	P9	P12	P11

Figure 1.5: Problems

	Θ_{Lb}	Θ_{Ub}	Θ_{Opt}	ObjLb	ObjUb
P0	(-1104, -549)	(1104, 549)	(-140.94, -106.08)	0	40.17
P1	(-1104, -549)	(0, 549)	(-336.28, -74.21)	0	28.15
P2	(0, -549)	(1104, 549)	(432.47, 25.20)	0	11.52
P3	(-1104, -549)	(-552, 549)	(-665.11, -66.06)	0	5.05
P4	(-552, -549)	(0, 549)	(-238.00, -154.00)	0	24.88
P5	(-552, 0)	(0,549)	(-239.45, 198.5954)	0	5.56
P6	(-552, -549)	(-552, -549)	(0, 0)	15	20.04
P7	(-552, -549)	(-276, 0)	(-436.00, -259.00)	0	2.88
P8	(-276, -549)	(0, 0)	(-232.00, -158.18)	0	16.46
P9	(-276, -549)	(0, -275)	(-154.00, -401.97)	0	1.53
P10	(-276, -274)	(0, 0)	(-232.00, -156.94)	0	15.89
P11	(-276, -274)	(-138, 0)	(-232.00, -156.81)	15	15.44
P12	(-138, -274)	(0, 0)	(-38.00, -99.03)	0	1.37

As popping $P11$ from the stack, optimal solution is obtained but, since $P2, P3, P5, P7, P8, P12$ are still in the stack, algorithm does not terminate. However, These problems have *bad bound* i.e. have smaller ObjUb (upper bound of the card.) than $P11$'s ObjLb (lower bound of the cardinality), and cannot be a optimal for sure, thus are popped without splitting and the algorithm terminates.

1.4 Discussion

Some comments for the bound of cardinality:

- The lower bound of cardinality for optimal solution monotonically increases.
- The upper bound of cardinality for optimal solution not necessarily monotonically decreases.
  - if the first convergence (local optimal solution) is not the global optimal solution, then the upper bound of cardinality can be increased as continuing depth-first search.
- The global optimal solution is obtained by branch and bound algorithm.