Lince Rumainum
DSA 5113
Advanced Analytics & Metaheuristics
Final Exam

Problem 1 – OK Highway Patrol

Assumptions:

- There might not be any officer in a segment.

The **objective** is to maximize the expected number of speeders caught across the segments

$$\sum_{j}^{J} y_j * s_j$$

Where J = {1, 2, 3, 4, 5, 6}, which is a list of segments

Decision Variables:
y_j : the number of officers covering the segment that day (integer value)
Set:
J = 1, 2, 3, 4, 5, 6
Parameters:
maxAvailOfficer – maximum available officers
s_j – number of expected speeders per officer for segment *j*, $j \in J$
$maxAssignedOfficers_j$ – maximum number of officers that can be assigned per segment *j*, $j \in J$

Constraints:
(maximum number of officers being assigned at each segment) , $j \in J$

$$y_j \leq maxAssignedOfficers_j$$

(maximum number of officers allowed at each segment), $j \in J$

$$\sum_{j}^{J} y_j \leq maxAvailOfficer$$

(non-negatives)
$y_j \geq 0$
$s_j \geq 0$
$maxAvailOfficer \geq 0$
$maxAssignedOfficers_j \geq 0$

Part b

Below is *data file*:

```
#indicate it's a data file---------------------------------------------
data;

#parameters and sets---------------------------------------------------
# set of airports
set J := segment1 segment2 segment3 segment4 segment5 segment6;

# parameter variables:
# u - the maximum number of officers that can be assigned at that segment
# s - the expected number of speeders per officer at that segment
```

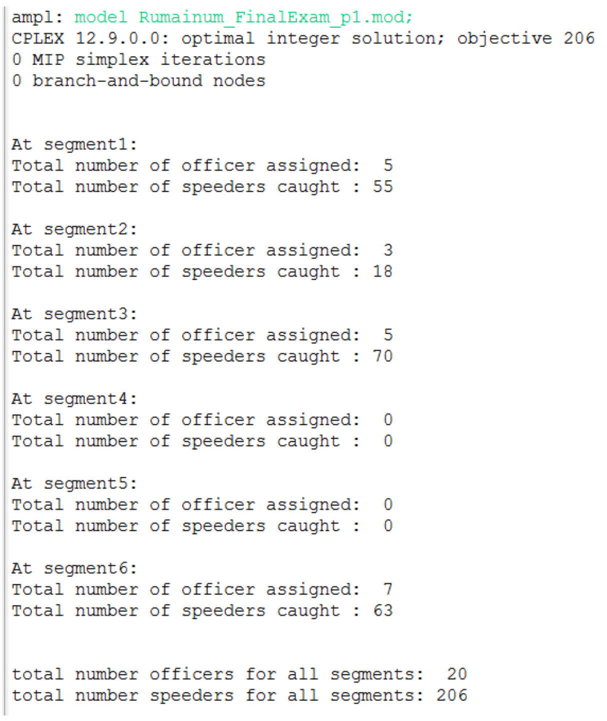
```
param:       u,                s      :=
 segment1      5               11
 segment2      9               6
 segment3      5               14
 segment4      6               4
 segment5      8               2
 segment6      7               9;

#maximum available officers
param maxAvailOfficer = 20;
```

Below is ***model file***:

```
#reset ampl----------------------------------------------------------------------------
reset;

#options-------------------------------------------------------------------------------
option solver cplex;

#parameters and sets-------------------------------------------------
set J; # set of segments

# parameter variables:
# u - the maximum number of officers that can be assigned at that segment
# s - the expected number of speeders per officer at that segment
param s {J} >= 0;
param u {J} >= 0;

#maximum available officers
param maxAvailOfficer >= 0;

#decision variables----------------------------------------------------------------
var y {j in J} >= 0 integer; #total number of officers assigned at the
segment


#objective: maximize total speeder caught------------------------------------
-----
maximize totalSpeeders: sum {j in J} y [j] * s[j];

#constraints:----------------------------------------------------------------------
# maximum number of officer that can be assigned at each segment
subject to maxAssignedOfficer{j in J}:  y [j] <= u[j];

#total number of officers available to be assigned at all segment
subject to maxTotalOfficer: sum {j in J} y [j] <= maxAvailOfficer;

#data------------------------------------
data Rumainum_FinalExam_p1.dat;

#commands----------------------------------
solve; # solve the integer LP problemm

#Print out solution
#spaces
printf "\n\n";
# display how many officers and the total speeders per segment
```

```
for {j in J} {
	printf "At %s: \n", j;
	printf "Total number of officer assigned: %2.0f \n", y[j];
	printf "Total number of speeders caught : %2.0f \n", y[j]*s[j];
	printf "\n";
}
printf "\n";

# display total officers assigned and the total speeders for all segments
printf "total number officers for all segments: %3.0f \n", sum {j in J} y
[j];
printf "total number speeders for all segments: %3.0f \n", totalSpeeders;

#spaces
printf "\n\n";
```

Results are shown below:



```
ampl: model Rumainum_FinalExam_p1.mod;
CPLEX 12.9.0.0: optimal integer solution; objective 206
0 MIP simplex iterations
0 branch-and-bound nodes


At segment1:
Total number of officer assigned:  5
Total number of speeders caught : 55

At segment2:
Total number of officer assigned:  3
Total number of speeders caught : 18

At segment3:
Total number of officer assigned:  5
Total number of speeders caught : 70

At segment4:
Total number of officer assigned:  0
Total number of speeders caught :  0

At segment5:
Total number of officer assigned:  0
Total number of speeders caught :  0

At segment6:
Total number of officer assigned:  7
Total number of speeders caught : 63


total number officers for all segments:  20
total number speeders for all segments: 206
```

Part c – i

New objective: maximize the speeder and maximizing the captain's visiting time

$$\sum_{j}^{J} y_j * s_j + \sum_{j}^{J} x_j * \text{capTime}_j$$

Where J = {1, 2, 3, 4, 5, 6}, which is a list of segments

New decision Variables:

x_j : whether the segment is covered by any officer or not (binary value – 0 indicates segment is not covered and 1 indicates segment is covered)

z_j : whether the segment is visited by the Captain or not (binary value – 0 indicates segment is not covered and 1 indicates segment is covered)

New parameters:
maxAvailTime – maximum available time that captain has to visit
capTime$_j$ – round trip time that captain use to visit segment *j*, $j \in J$

New constraints:
(segment is covered)

$$\sum_{j}^{J} z_j * capTime_j \leq maxAvailTime$$

(Captain visit segment with at least one officer assigned to it) for each segment *j*, $j \in J$

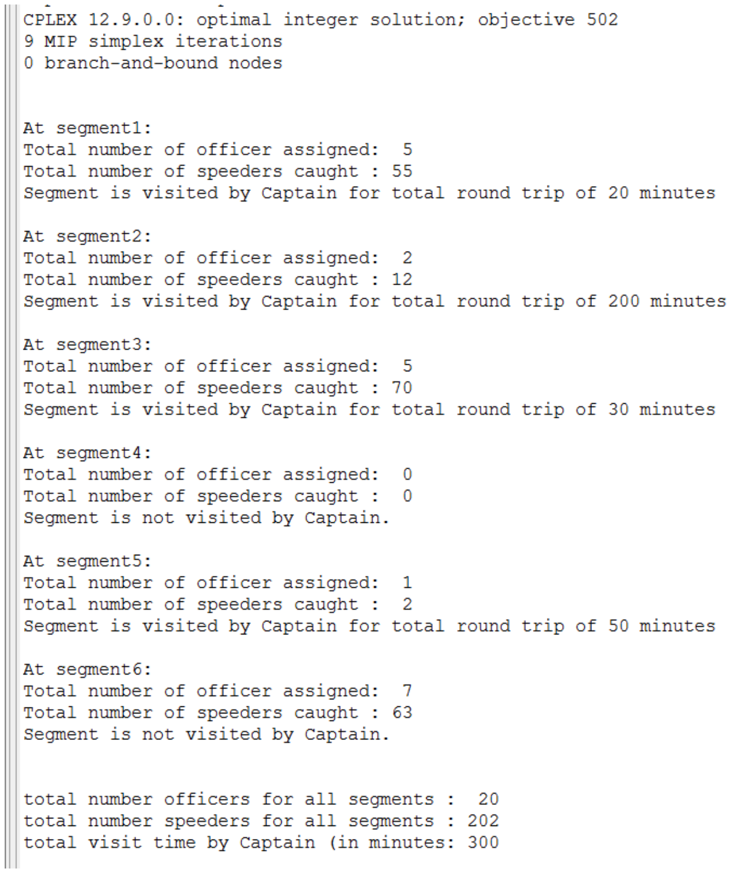
$$z_j = x_j$$

(indicator whether or not officer is assigned for each segment), $j \in J$

$$x_j * s_j \leq y_j * s_j$$

Part c – ii

Modifying the code according to new update objective, decision variables, and constraints gives the results shown below:



```
CPLEX 12.9.0.0: optimal integer solution; objective 502
9 MIP simplex iterations
0 branch-and-bound nodes


At segment1:
Total number of officer assigned:  5
Total number of speeders caught : 55
Segment is visited by Captain for total round trip of 20 minutes

At segment2:
Total number of officer assigned:  2
Total number of speeders caught : 12
Segment is visited by Captain for total round trip of 200 minutes

At segment3:
Total number of officer assigned:  5
Total number of speeders caught : 70
Segment is visited by Captain for total round trip of 30 minutes

At segment4:
Total number of officer assigned:  0
Total number of speeders caught :  0
Segment is not visited by Captain.

At segment5:
Total number of officer assigned:  1
Total number of speeders caught :  2
Segment is visited by Captain for total round trip of 50 minutes

At segment6:
Total number of officer assigned:  7
Total number of speeders caught : 63
Segment is not visited by Captain.


total number officers for all segments :  20
total number speeders for all segments : 202
total visit time by Captain (in minutes: 300
```

Part d – i

New assumptions:

- There should be at least one officer for each segment so no county will be overlooked.
- All officers has to be assigned to a segment.

New objective: maximize the minimum expected number of speeders caught across the segments

$$\text{minimize } S$$

New decision Variables:

S: set of maximum number of speeders caught across the segments

New constraints:

(maximum total time for captain to visit the segment)

$$S \geq \sum_{j}^{J} y_j * s_j$$

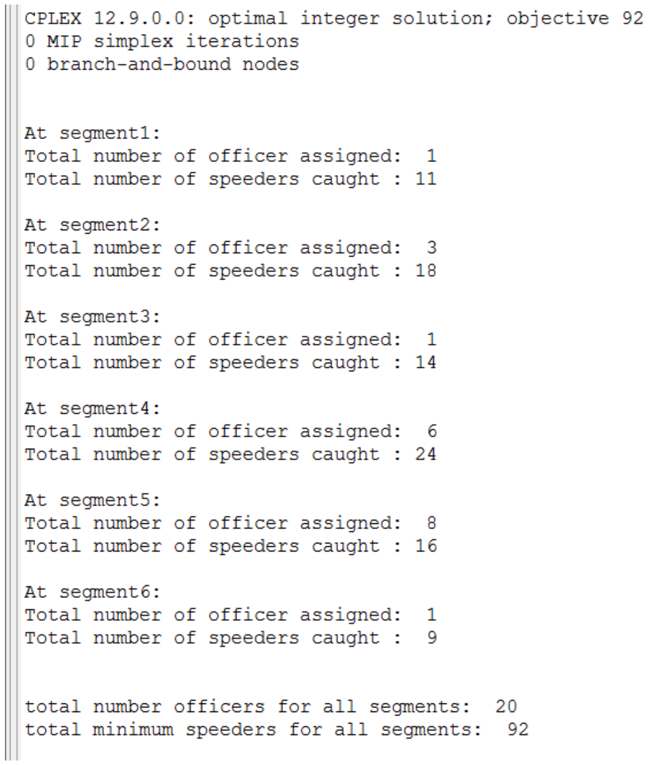
(maximum number of officers allowed at each segment) , $j \in J$

$$\sum_{j}^{J} y_j = maxAvailOfficer$$

(at least 1 officer per segment) $y_j \geq 0$

Part d – ii

Modifying the code according to new update objective, decision variables, and constraints gives the results shown below:

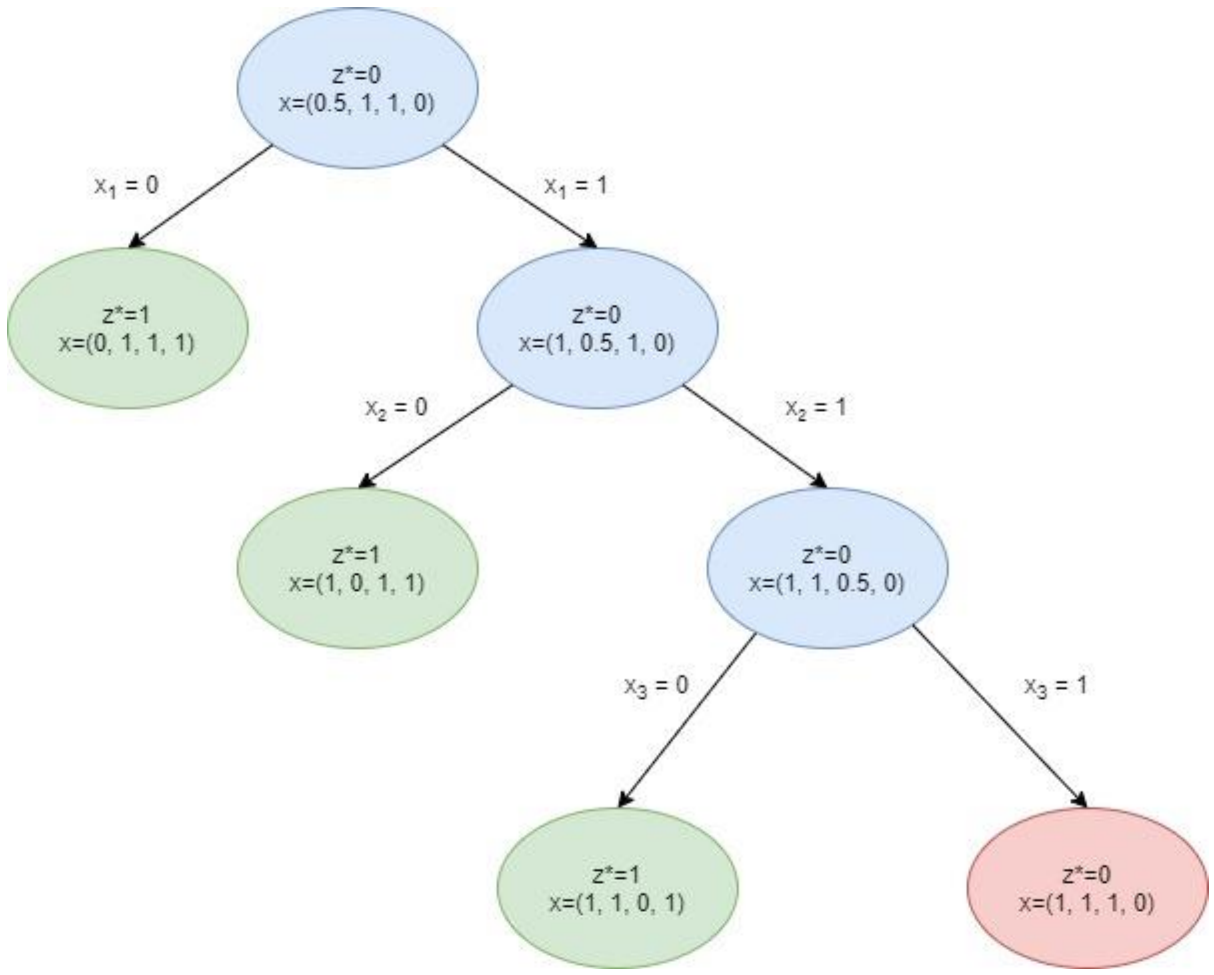


```
CPLEX 12.9.0.0: optimal integer solution; objective 92
0 MIP simplex iterations
0 branch-and-bound nodes


At segment1:
Total number of officer assigned:  1
Total number of speeders caught : 11

At segment2:
Total number of officer assigned:  3
Total number of speeders caught : 18

At segment3:
Total number of officer assigned:  1
Total number of speeders caught : 14

At segment4:
Total number of officer assigned:  6
Total number of speeders caught : 24

At segment5:
Total number of officer assigned:  8
Total number of speeders caught : 16

At segment6:
Total number of officer assigned:  1
Total number of speeders caught :  9


total number officers for all segments:  20
total minimum speeders for all segments:  92
```

Problem 2 – Branch-And-Bound

$$\min x_4$$
$$\text{s.t. } 2x_1 + 2x_2 + 2x_3 + x_4 = 5$$
$$x_\text{i} \in \{0,1\}, i = 1, \ldots, 4$$

(a) Solve this problem by drawing out the complete branch-and-bound tree. Make sure to label the upper and lower bounds, the branches, the LP-relaxation optimal solution, and which nodes are fathomed and why.



Blue indicates that it was an active node,
Green indicates that it reaches the optimal solution, and
Red indicates fathomed node due to infeasibility.

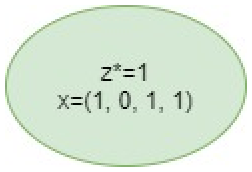
Start with x=(0.5, 1, 1, 0) and incumbent objective solution of 0 and then once branch out to see different x_1-value, it found its first integer solution at $x_1 = 0$ with objective value of 1 and x=(0, 1, 1, 1), then again when it branch out at $x_1 = 0 \rightarrow x_2 = 0$, with objective value of 1 and x=(1, 0, 1, 1), then again when it branch out at $x_1 = 0 \rightarrow x_2 = 1 \rightarrow x_3 = 1$, , with objective value of 1 and x=(1, 1, 0, 1).

(b) Define the best valid inequality you can for this problem.

$$x_1 + x_2 + x_3 + x_4 \geq 3$$

(c) Draw and label the branch-and-bound tree again, but this time, incorporate your valid inequality.

Note: This is should reduce the size of your B&B tree!



Problem 3 – Neighborhood-Based Search Questions

Consider the following 2D problem

$$\min\ (x_1 - 1)^2 + (3x_2 - 1)^2 - x_1x_2$$
$$\text{s.t.}\ -4 \leq\ x_i\ \leq 4\ , i\ =\ 1,2$$
$$x_{\text{i}}\ integer, i\ =\ 1,2$$

(a) What is the size of the solution space. Explain.

The size of the solution space is 81 because x1 and x2 are integers, which makes each having 9 different possible solution between -4 and 4, which are {-4, -3, -2, -1, 0, 1, 2, 3, 4}.

(b) Define and explain a valid neighborhood definition that could be used in hill-climber algorithm for this problem. What is the size of the neighborhood you have defined?

The neighborhood definition that could be used in hill-climber algorithm is shifting x_i by 1 (with the exception of x_i = 4 then shifting it by -1 instead) while the others stay the same. For example, if the point is at (-4, 4), its neighbors would be (-3, 4) and (-4, 3). The size of the neighborhood defined is 2.

(c) Apply the iterated hill-climber algorithm from Chapter 2 (page 43-44) of the How to Solve It textbook to this problem assuming an initial location of (-2,3) with an evaluation of 79. Complete 2 iterations of the algorithm by hand and demonstrate the steps.

FIRST ITERATION:

```
begin
    t ← 0
    initialize best = 79
    repeat
        local ← FALSE
        v_c ← (1,2)
        evaluate v_c ← 23
        repeat
            select all new points in the neighborhood of v_c ← (2,2), (1,3)
            select the point v_n from the set of new points
```

```
                with the best value of evaluation function eval ← (1,3)
              if eval($v_n$) is better than eval ($v_c$) → if 61 is better than 23
                    then $v_c$ ← $v_n$ : $v_c$ ← (1,3)
              else local ← TRUE
          until local
          t ← t + 1
          if $v_c$ is better than the best
              then best ← $v_c$
      until t = MAX
end
```

REPEAT INSIDE THE LOCAL LOOP

```
          repeat
              select all new points in the neighborhood of $v_c$ ← (2,3), (1,4)
              select the point $v_n$ from the set of new points
                with the best value of evaluation function eval ← (1,4)
              if eval($v_n$) is better than eval ($v_c$) → if 117 is better than 61
                    then $v_c$ ← (1,4)
              else local ← TRUE
          until local
```

REPEAT THE LOCAL LOOP

```
          repeat
              select all new points in the neighborhood of $v_c$ ← (2,4), (1,3)
              select the point $v_n$ from the set of new points
                with the best value of evaluation function eval ← (2,4)
              if eval($v_n$) is better than eval ($v_c$) → if 114 is better than 117
                    then $v_c$ ← $v_n$
              else local ← TRUE
          until local
          t ← 0 + 1
          if $v_c$ is better than the best → if 117 is better than 79
              then best ← $v_c$ : best ← 117
```

SECOND ITERATION:

```
begin
      t ← 0
      initialize best = 79
      repeat
          local ← FALSE
          $v_c$ ← (-2,4)
          evaluate $v_c$ ← 138
          repeat
              select all new points in the neighborhood of $v_c$ ← (-1,4), (-2,3)
              select the point $v_n$ from the set of new points
                with the best value of evaluation function eval ← (-1,4)
```

```
            if eval($v_n$) is better than eval ($v_c$) → if 129 better than 138
                    then $v_c$ ← $v_n$
            else local ← TRUE
        until local
        t ← 1 + 1
        if $v_c$ is better than the best → if 138 is better than 117
            then best ← $v_c$: best ← 138
    until t = MAX
end
```

*Since **MAX** is two then the algorithm would come to an end with best value of 138.*

(d) Assume that throughout a search algorithm you come across two particular solutions, A = (-1,-3) and B = (1,-1). Demonstrate the evaluations and moves from path-relinking starting at solution A and ending at solution B.

First, evaluate both solution A and Solution B, we have:

	x_1	**x_2**	**$f(x_1,x_2)$**
A	-1	-3	101
B	1	-1	17

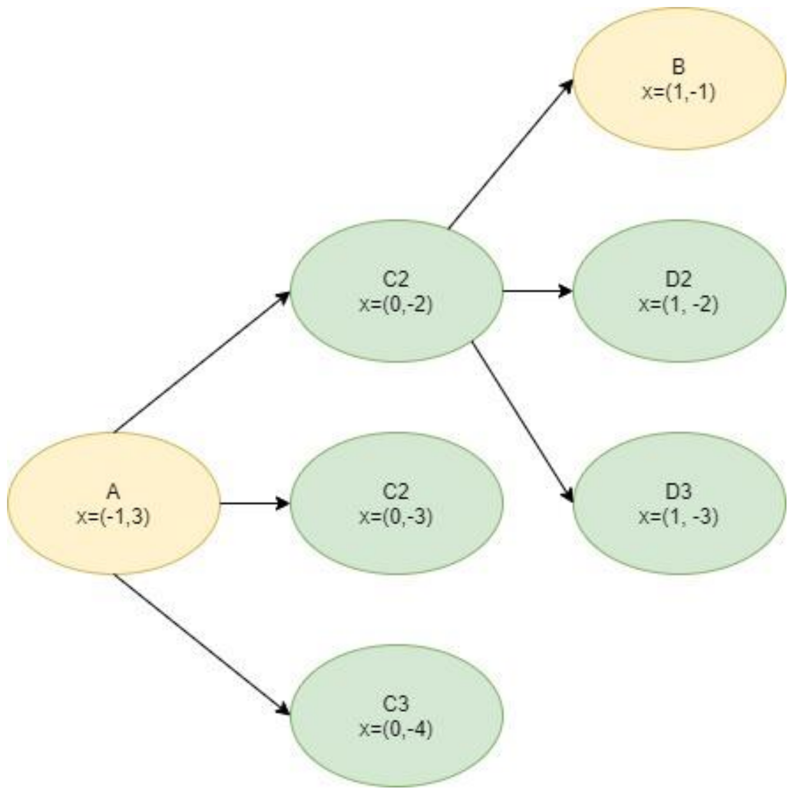
So, we know that we want to go from solution of 101 to 17. To do so, we are going to shift x_1 by +1 until it reach value of 1 and look at current x_2 value and its +1 and -1 values. So, from (-1, -3), we evaluate three different solutions, which are:

	x_1	**x_2**	**$f(x_1,x_2)$**
C1	0	-2	50
C2	0	-3	101
C3	0	-4	170

Since the goal is to find solution value of 17, we pick (0, -2) as the next point. Doing the same approach as we did before, we evaluate the next three solutions, which are:

	x_1	**x_2**	**$f(x_1,x_2)$**
D1	1	-1	17
D2	1	-2	51
D3	1	-3	103

Since the goal is to find solution value of 17 where (1, -1), we successfully guided the solution from starting solution A and ending it at solution B by linking them through solution C1 = (0, -2).



(e) Assume you are using Simulated Annealing to solve this problem. The current temperature is 15.0 and the current solution is (3, 2).

To find the probability of accepting the candidate solution, we are using:

$$p = e^{-(f(s_1) - f(s_2))/T}$$

i. **What is the probability of accepting the candidate solution (3; 1)?**

$$p = e^{-(23-5)/15} = 0.3012$$

ii. **What is the probability of accepting the candidate solution (3; 3)?**

$$p = e^{-(23-59)/15} = 11.0232$$

Problem 4 – Population-Based Algorithm

Solution		Encoding	
A	12.9	27.1	33.4
B	13.5	76.2	19.1

The elements of the solutions above are denoted as:

Solution		Encoding	
A	p_{A1}	p_{A2}	p_{A3}
B	p_{B1}	p_{B2}	p_{B3}

Conducting crossover between solutions A and B using the approach in the continuous-valued crossover, assuming that the crossover point is between the first and second element of the chromosome and the random value, β, is 0.25, would results in:

Solution	Encoding		
Offspring$_1$	p_{new1}	p_{B2}	p_{B3}
Offspring$_2$	p_{new2}	p_{A2}	p_{A3}

From calculating α = **roundup**{**β***N_{var}} → α = **roundup**{**0.25***3}→ α =1, we can use:

For Offspring$_1$, the first element will be a new crossover element that can be found using the equation below:

$$p_{new1} = p_{A1} - \beta(p_{A1} - p_{B1})$$

While the rest of the elements will be swapped with elements from solutions B. Calculation for p_{new1}:

$$p_{new1} = 12.9 - 0.25(12.9 - 13.5) \rightarrow p_{new1} = 13.05$$

For Offspring$_2$, the first element will be a new crossover element that can be found using the equation below:

$$p_{new2} = p_{B1} + \beta(p_{A1} - p_{B1})$$

While the rest of the element will be swapped with elements from solutions A. Calculation for p_{new1}:

$$p_{new2} = 13.5 + 0.25(12.9 - 13.5) \rightarrow p_{new2} = 13.35$$

So far, we have crossover all the elements on the Left Hand Side:

Offspring$_1$	13.05	p_{B2}	p_{B3}
Offspring$_2$	13.35	p_{A2}	p_{A3}

Now, we can swamped the rest of the elements to complete the crossover. The rest of the elements are swapped accordingly (solution A's second element with solution B's second element and solution A's third element with solution B's third element). Doing so will resulted in:

Offspring$_1$'s elements:

13.05 | 76.2 19.1

Offspring$_2$'s elements:

13.35 | 27.1 33.4

Therefore, the crossover between Solutions A and B, created:

Solution	Encoding		
Offspring$_1$	13.05	76.2	19.1
Offspring$_2$	13.35	27.1	33.4

Problem 5 – Multiple Objective Optimization

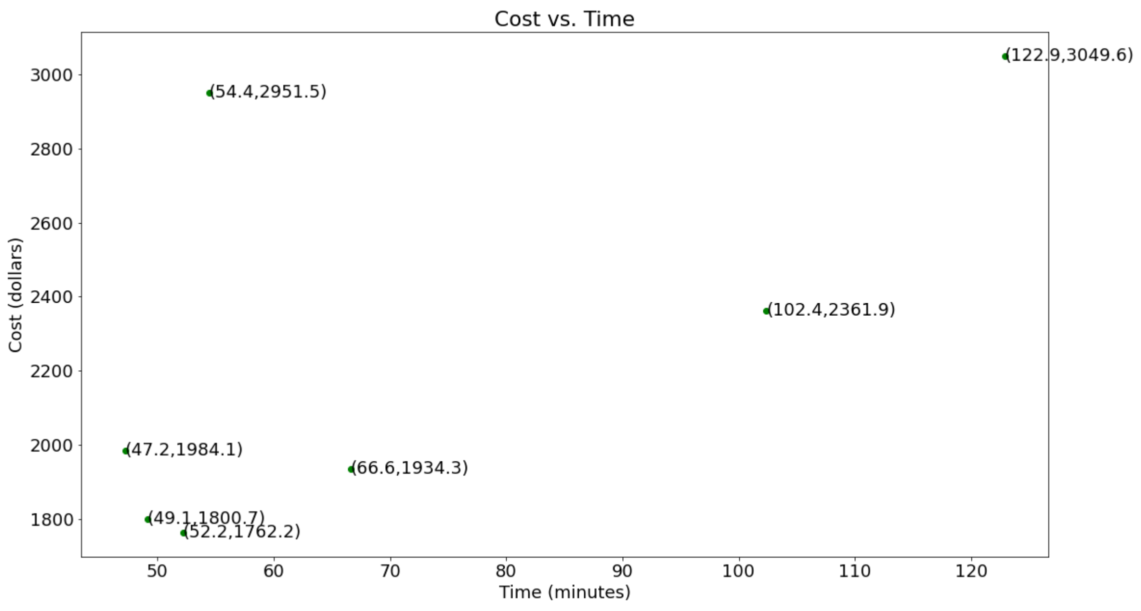
From looking at the data on Table 4, with process of eliminations:

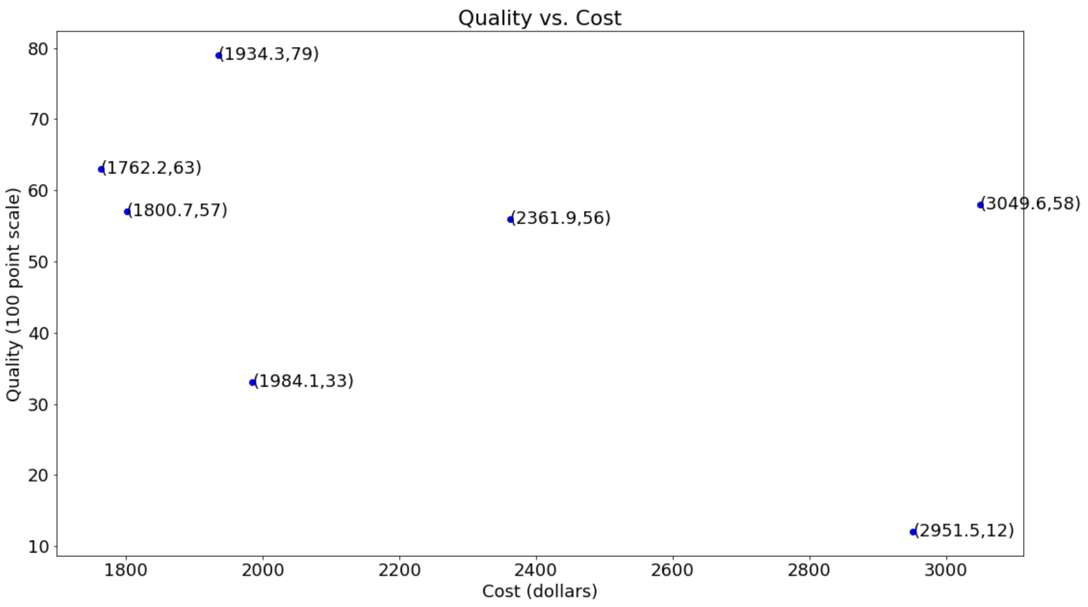
- longer time and higher cost for Solution ID 2 and Solution ID 4 can be eliminated by Solution ID
- Solution ID 1 can be eliminated by Solution ID 3 since it has shorter time, lower cost and higher quality
- Solution ID 6 can be eliminated by Solution ID 7 since it has lower cost and much higher quality

Therefore, Solution ID 3, Solution ID 5, and Solution ID 7 are the non-dominated solutions.
Solution ID 3 cannot eliminate Solution ID 7 since it has lower time and even though it does have a higher cost, the quality is only 7-point difference. As for Solution ID 5, Since it does create the highest quality (over 15-point) from the next best quality, the difference between its time and cost would make sense why it has higher quality.

Another way to look at it is from plotting the data on Table 4. Below two different plots, Cost vs. Time and Quality vs. Cost, we can see that data with high cost and low quality can be eliminated (Solution ID 1, Solution ID 2 and Solution ID 4) and although solution ID 6 is considered to be low cost and short time, the quality of the solution is also low therefore it can be eliminated as well. The solutions that are left are Solution ID 3, Solution ID 5, and Solution ID 7. Those are all the non-dominated solutions. Solution ID 3 has the lowest time of 49.1 minutes, Solution ID 7 has the lowest cost of $1,762.20, while Solution ID 6 has the highest quality of 79. As shown in the plots, those three Solution IDs are closer to each other than the other Solution IDs.

(Plots can be seen on the next page)


Cost vs. Time
Cost (dollars)
Time (minutes)
(122.9,3049.6)
(54.4,2951.5)
(102.4,2361.9)
(47.2,1984.1)
(66.6,1934.3)
(49.1,1800.7)
(52.2,1762.2)
3000
2800
2600
2400
2200
2000
1800
50
60
70
80
90
100
110
120


Quality vs. Cost
Quality (100 point scale)
Cost (dollars)
(1934.3,79)
(1762.2,63)
(1800.7,57)
(2361.9,56)
(3049.6,58)
(1984.1,33)
(2951.5,12)
80
70
60
50
40
30
20
10
1800
2000
2200
2400
2600
2800
3000