Eindopdracht - Prescriptive Analytics: Optimization

Student: Justin Koning, Studentnummer: 303204

Project Scheduling

Exercise 1 MIP formulation

1a) Formulate the problem as a mixed integer linear programming (MIP) model.

Parameters:

N = number of projects

P = number of employees

W_j = weight project j

L_{jk}= duration project j assigned to project k

D_j = deadline project j

M = BigM - equal to the maximum of the total durations of projects that are performed by an employee, minus the minimum of all durations

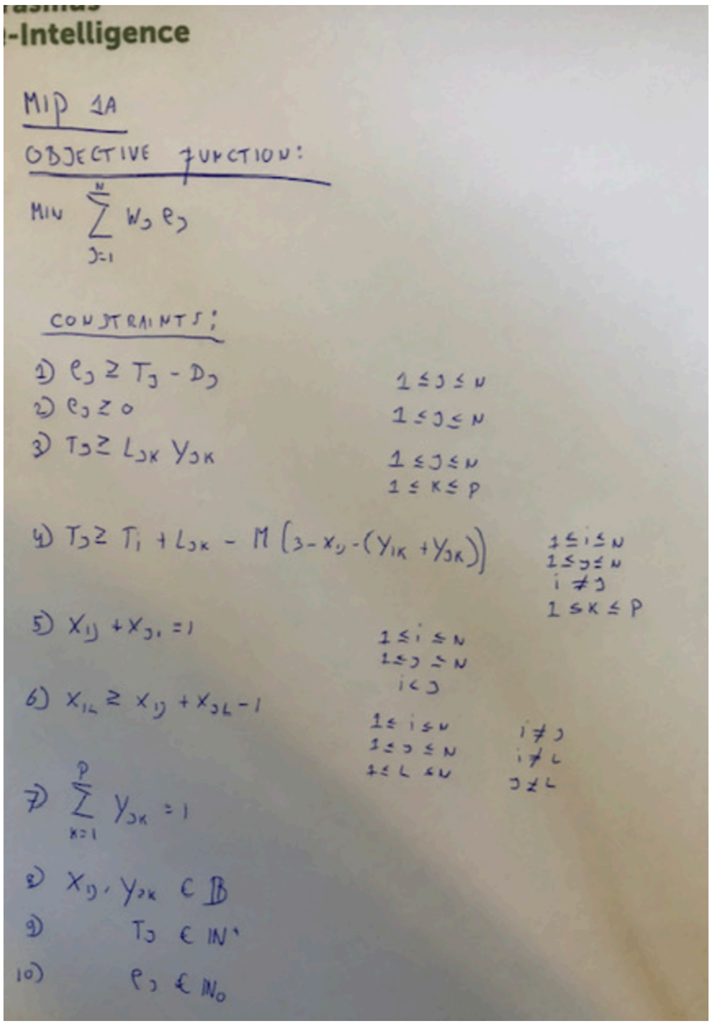
<u>Decision variables:</u>

X_{ij}= 1, when project i is before project j, else 0

Y_{jk}= 1, when project j is assigned to person k, else 0

E_j= amount of tardiness project j

T_j= finishing time of a project j



-Intelligence

MIP 1A

OBJECTIVE FUNCTION:

$$\text{MIN} \sum_{j=1}^{N} W_j e_j$$

CONSTRAINTS:

1) $e_j \geq T_j - D_j$ $\quad 1 \leq j \leq N$

2) $e_j \geq 0$ $\quad 1 \leq j \leq N$

3) $T_j \geq L_{jk} Y_{jk}$ $\quad 1 \leq j \leq N$, $1 \leq k \leq P$

4) $T_j \geq T_i + L_{jk} - M\left(3 - x_{ij} - (Y_{ik} + Y_{jk})\right)$ $\quad 1 \leq i \leq N$, $1 \leq j \leq N$, $i \neq j$, $1 \leq k \leq P$

5) $X_{ij} + X_{ji} = 1$ $\quad 1 \leq i \leq N$, $1 \leq j \leq N$, $i < j$

6) $x_{iL} \geq x_{ij} + x_{jL} - 1$ $\quad 1 \leq i \leq N$, $1 \leq j \leq N$, $1 \leq L \leq N$, $i \neq j$, $i \neq L$, $j \neq L$

7) $\sum_{k=1}^{P} Y_{jk} = 1$

8) $X_{ij}, Y_{jk} \in \mathbb{B}$

9) $T_j \in \mathbb{N}^+$

10) $e_j \in \mathbb{N}_0$

<u>Constraint explanations:</u>

1 and 2) Tardiness is the difference between the finishing time of project j and its deadline, but must be equal to or larger than zero.

3) Finishing time of project j must be at least equal to the duration of project j that is assigned to person k.

4) Finishing time of a project j must be at least equal to the finishing time of project i plus the duration of project j, if project i is directly before j and the project is assigned to the same person.

5) If project i is before j, then j cannot be before i.

6) Transitivity, if i is before j, and j is before l, then i must be before l.

7) A project j can be assigned to at most 1 person.

8) Gives the ranges of the decision variables X_{ij} and $Y_{jk,}$ where B = {0,1}.

9) Gives the ranges of the decision variable $T_{j,}$ where N^+ is the set of all positive natural numbers**.**

10) Gives the ranges of the decision variable $E_{j,}$ where N_0 is the set of all non-negative natural numbers.

1b) Implement the model in R and solve the small instance with the Gurobi solver. Report the assignment and order of the projects, the finishing times, as well as the optimal objective value

The small instance gives the following results:

Objective value:

Minimum total fine: 19

Assignment and order of projects:

Employee 1: 6, 2, 10, 7 and 4

Employee 2: 8, 1, 5, 9 and 3

Finishing times of projects:

Project 1: 6

Project 2: 10

Project 3: 30

Project 4: 20

Project 5: 9

Project 6: 8

Project 7: 19

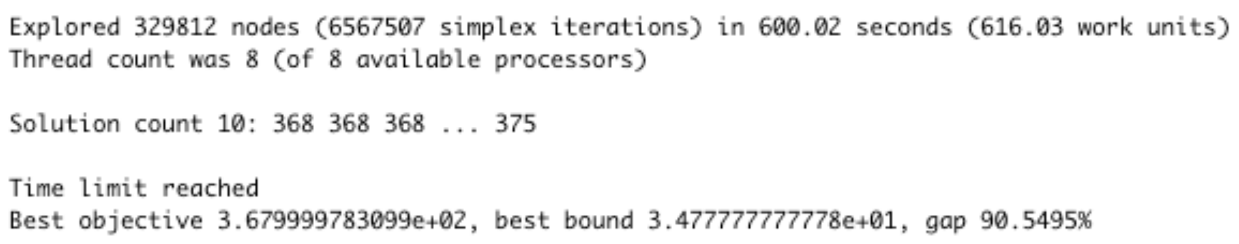
Project 8: 3

Project 9: 19

Project 10: 13

1c) Run the large instance for 10 minutes using the Gurobi solver. What is the best objective value found so far? What do you know about the optimality gap, i.e., how far is it from the optimal objective value?

The best feasible objective solution so far is 368 and has an optimality gap of 90.5% (to the (relaxed) lower bound). See output below:



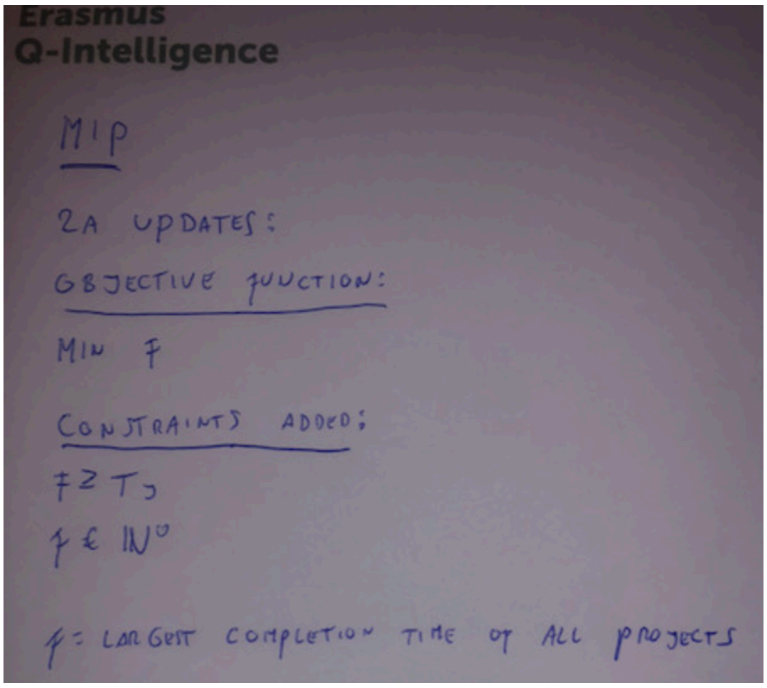
```
Explored 329812 nodes (6567507 simplex iterations) in 600.02 seconds (616.03 work units)
Thread count was 8 (of 8 available processors)

Solution count 10: 368 368 368 ... 375

Time limit reached
Best objective 3.679999783099e+02, best bound 3.477777777778e+01, gap 90.5495%
```

2a) Formulate the problem with the second objective as a MIP and clearly explain any (additional) variables and constraints used.

Updated (MinMax) MIP model:



Erasmus
Q-Intelligence

MIP

2A UPDATES:

OBJECTIVE FUNCTION:

MIN F

CONSTRAINTS ADDED:

$F \geq T_J$

$F \in \mathbb{N}^0$

F: LARGEST COMPLETION TIME OF ALL PROJECTS

Added constraint explanation:

F = The makespan f is the largest completion time of all projects T_j

2b) We like to find some non-dominated points by the weighted sum method for the small instance. In particular, find the two non-dominated points corresponding to the weights λ = (0.95, 0.05) and λ = (0.05, 0.95).

See R script - the MIP model has been ran for both lambdas without any epsilon constraints.

The results of the weighted sum method for the small instance are the following:

Non-dominated point corresponding to:

λ (0.95,0.05) **= (19, 30)**

Non-dominated point corresponding to:

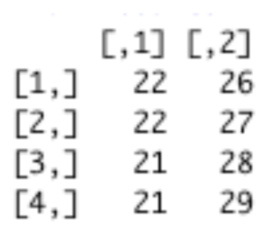
λ (0.05,0.95) **= (31, 25)**

2c) Use an appropriate method to find the set of non-dominated points 'between the two points' found in part b. To be precise, find the non-dominated points for which the second objective values are between the second objective values of the two mentioned points. Plot these non-dominated points in a graph.

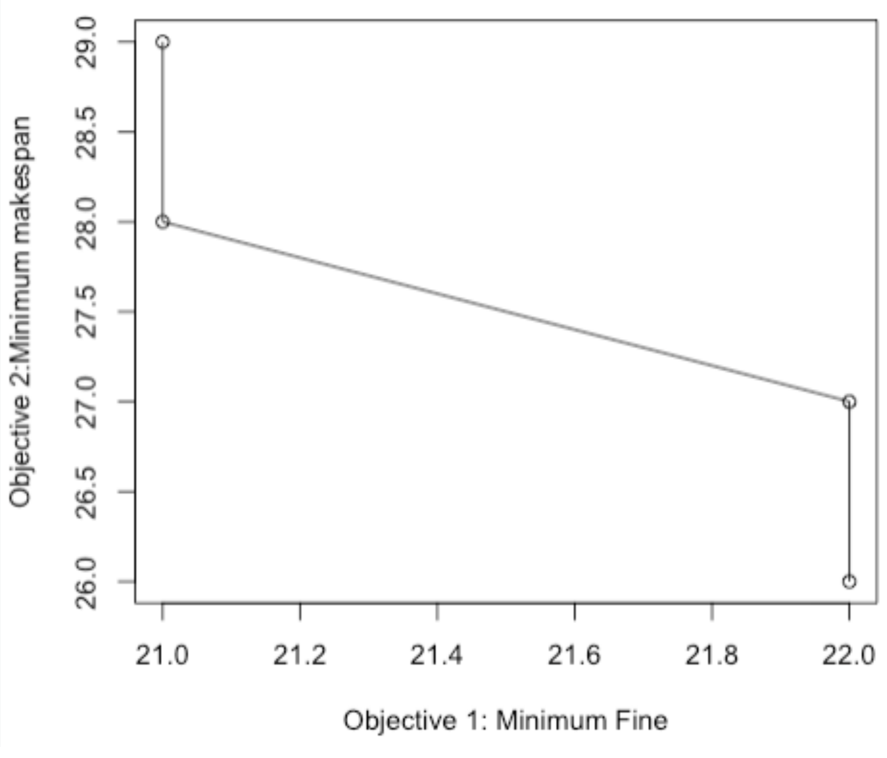
The model including epsilon constraint is in de script ('BiObjectiveModel')

The epsilon constraint method is used to find the set of non-dominated points between the points identified in 2b (y1=(19, 30) and y2=(31,25), where objective 1 is optimized while bounding the second objective to epsilon. At each iteration epsilon is increased with 1 (as the objective values are integer values) in the interval 30-25 = 5.

The results are shown down below:

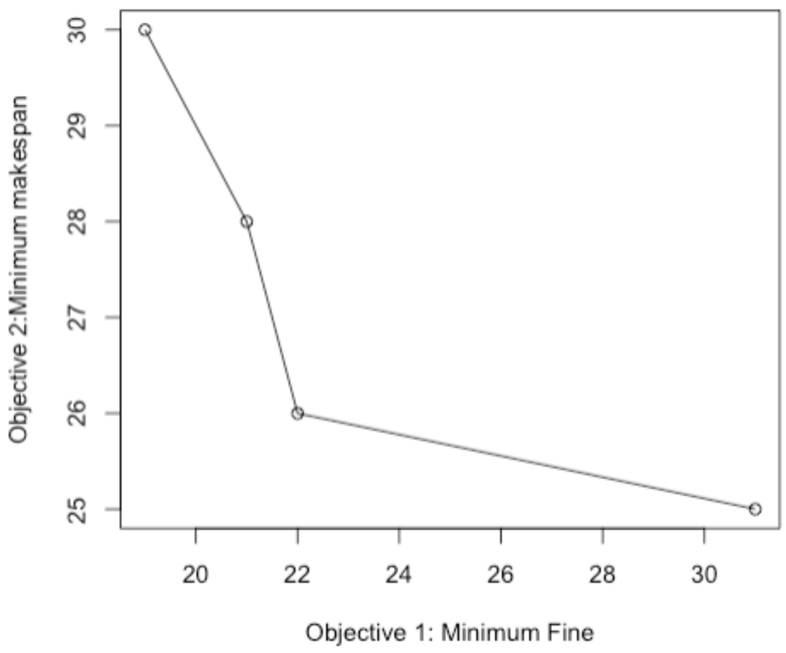


```
     [,1] [,2]
[1,]   22   26
[2,]   22   27
[3,]   21   28
[4,]   21   29
```



The plot clearly shows that point (21,28) dominates point (21,29) and that point (22,26) dominates point (22,27) as these points have the same objective value for the first objective, but is strictly better in terms of the second objective.

The set of non-dominated points between the 2 points from 2b (and including these points) are plotted below:



3a) Describe a greedy heuristic and explain why you expect this may lead to a good solution.

First, create a table with per project j, the average completion time (where this is calculated based on two employees working simultaneously on this project); its weight; and the ratio weight over the average completion time. Then sort the projects based on this ratio in a decreasing order.

The rationale behind this ratio is that higher weights contribute more to the total fine whereas early completion of projects is beneficial for subsequent projects (and meeting the corresponding deadlines) as it contributes many times. Therefore, for a purely greedy heuristic, projects that have a higher ratio should have priority in determining the order.

However, it could be the case that always choosing the best ratio at each iteration does not yield the overall best objective value as each project has different deadlines (ie when two ratios are fairly close to each other, but the deadline of the slightly smaller ratio is way earlier and could in the end give a total fine that is larger). Therefore add a random component (ie GRASP).

So start with project j which has the highest ratio. Then iteratively choose the next project based on a random component (GRASP). To be more precise, choose one of the <u>two</u> next best options (ie projects with highest ratios that has not been chosen yet) randomly from the restricted candidate list and proceed until all projects are finished. Finally, repeat this process many times such that we may get lucky in one repetition where a different order than the purely greedy order yields a better objective.

3b) Implement the greedy heuristic of part a in R and give the obtained objective value of both the small and large instance.

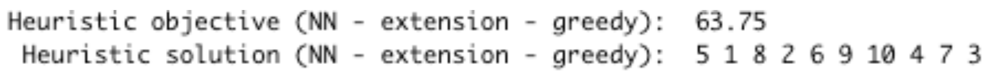
See R script for implementation

<u>Small instance:</u>

A comparison has been made between the pure greedy heuristic and the grasp heuristic. The results are the following:

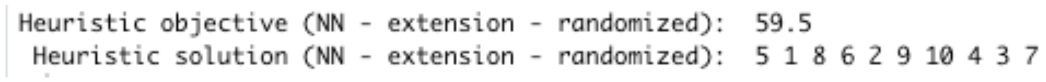
set.seed(29993)

<u>Greedy:</u>



```
Heuristic objective (NN - extension - greedy):  63.75
 Heuristic solution (NN - extension - greedy):  5 1 8 2 6 9 10 4 7 3
```

<u>GRASP:</u>



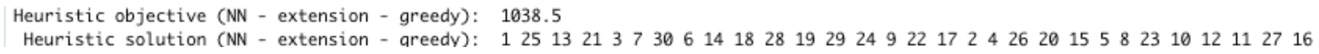
```
Heuristic objective (NN - extension - randomized):  59.5
 Heuristic solution (NN - extension - randomized):  5 1 8 6 2 9 10 4 3 7
```

<u>Large instance:</u>

A comparison has been made between the pure greedy heuristic and the grasp heuristic. The results are the following:

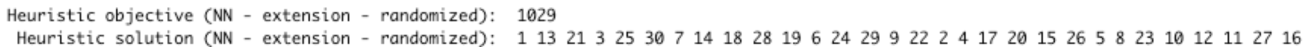
set.seed(112)

<u>Greedy:</u>



```
Heuristic objective (NN - extension - greedy):  1038.5
 Heuristic solution (NN - extension - greedy):  1 25 13 21 3 7 30 6 14 18 28 19 29 24 9 22 17 2 4 26 20 15 5 8 23 10 12 11 27 16
```

<u>GRASP:</u>



```
Heuristic objective (NN - extension - randomized):  1029
 Heuristic solution (NN - extension - randomized):  1 13 21 3 25 30 7 14 18 28 19 6 24 29 9 22 2 4 17 20 15 26 5 8 23 10 12 11 27 16
```

Note that the results are better for the GRASP heuristic than the greedy heuristic with an objective value of 59.5 for the small instance and 1029 for the large instance. Also note that the objective values can be non-integer as a result of the introduction/replacement of the average waiting time as a variable. This variable is real valued.

3c) Describe a local search algorithm. Clearly explain how your neighborhood is defined.

The starting point of the local search is the solution obtained from the GRASP heuristic. Then, a neighborhood is defined as all solutions that can be obtained where two projects i and j are swapped in the initial solution (in terms of their order). The local search will explore the entire neighborhood to see if any swap operation will yield a better solution. If all options in the neighborhood have been explored, then the local search will stop.

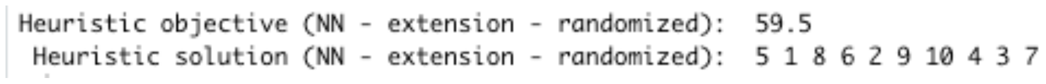
3d) Implement the local search algorithm in R, apply it to the solutions of part b, and report the obtained objective values.

See R script for implementation

<u>Small instance:</u>

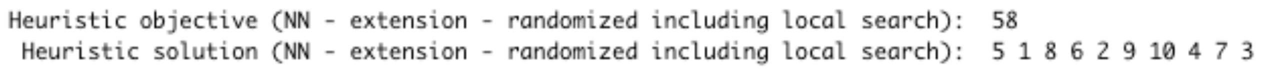
set.seed(29993)

<u>Initial solution from GRASP:</u>



```
Heuristic objective (NN - extension - randomized):  59.5
 Heuristic solution (NN - extension - randomized):  5 1 8 6 2 9 10 4 3 7
```

<u>GRASP with local search:</u>

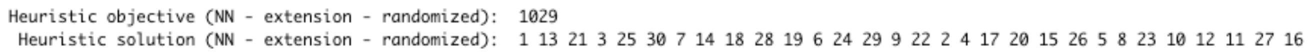


```
Heuristic objective (NN - extension - randomized including local search):  58
 Heuristic solution (NN - extension - randomized including local search):  5 1 8 6 2 9 10 4 7 3
```

<u>Large instance:</u>

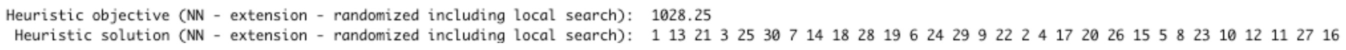
set.seed(112)

<u>Initial solution from GRASP:</u>



```
Heuristic objective (NN - extension - randomized):  1029
 Heuristic solution (NN - extension - randomized):  1 13 21 3 25 30 7 14 18 28 19 6 24 29 9 22 2 4 17 20 15 26 5 8 23 10 12 11 27 16
```

<u>GRASP with local search:</u>



```
Heuristic objective (NN - extension - randomized including local search):  1028.25
 Heuristic solution (NN - extension - randomized including local search):  1 13 21 3 25 30 7 14 18 28 19 6 24 29 9 22 2 4 17 20 26 15 5 8 23 10 12 11 27 16
```

Note that the local search algorithm has improved the initial solutions for both instances. The new objective value in the small instance is equal to 58 where project 3 and 7 have been swapped. The objective value in the large instance has improved to 1028.5 where project 15 and 26 have been swapped.