INSA Rennes, 4GM - 2021-2022
Operations Research-Project
Ayse N. Arslan : aarslan@insa-rennes.fr
Jeremy Omer : jeremy.omer@insa-rennes.fr

Branch-and-Bound for the one-machine sequencing problem

I. Objectif

The purpose of this project is to develop a branch-and-bound algorithm for the one-machine sequencing problem. The branch-and-bound algorithm described below reposes on generating lower and upper bounds at each node through algorithmic means. It is also possible to describe a branch-and-bound algorithm based on a mixed-integer programming formulation of the problem and its continuous relaxation. A MIP solver will execute such an algorithm in the background when solving a given MIP formulation. Your objective will be to compare the numerical performance of your algorithm to that of the MIP solver.

II. Problem description

We would like to sequence a set of n jobs on a single machine. We will denote this set of jobs as $N = \{1, \ldots, n\}$. For each job $i \in N$, we are given its release date a_i (meaning that job i cannot be started before $t = a_i$), its processing time on the machine d_i, and a tail duration q_i that is spent in the system but not occupying the machine. The objective is to minimize the makespan of producing all jobs in N, *i.e.*, the completion time of the latest job that leaves the system. We note that the completion time of job i, given that it starts at time $t \geq a_i$, is given as $t + d_i + q_i$, whereas the machine is available to process another job starting from $t + d_i$.

An instance of this problem with $n = 7$ jobs is given in Table 1. This instance will be used to illustrate various concepts in the following.

Jobs i	1	2	3	4	5	6	7
Release dates a_i	10	13	11	20	30	0	30
Processing times d_i	5	6	7	4	3	6	2
Tails q_i	7	26	24	21	8	17	0

Table 1 – *Example instance with $n = 7$ jobs.*

We start by defining "the conjunctive graph", $G^{\mathcal{S}} = (V, A^{\mathcal{S}})$ associated to a given sequence $\mathcal{S}$. To this end, we define the set of vertices of this graph by $V = N \cup \{s, t\}$, where s represents the beginning and t represents the end of processing of all jobs. We write $A^{\mathcal{S}} = A_1^{\mathcal{S}} \cup A_2^{\mathcal{S}} \cup A_3^{\mathcal{S}}$, where $A_1^{\mathcal{S}} = \{(s, i) \mid i \in N\}$, $A_2^{\mathcal{S}} = \{(i, j) \mid \text{job i precedes job j in } \mathcal{S}\}$, $A_3^{\mathcal{S}} = \{(j, t) \mid j \in N\}$. The cost of arc $(s, i) \in A_1^{\mathcal{S}}$ is given by a_i, the cost of arc $(i, j) \in A_2^{\mathcal{S}}$ is given as d_i, and the cost of arc $(i, t) \in A_3^{\mathcal{S}}$ is given as $d_i + q_i$.

The conjunctive graph associated to the sequence $\mathcal{S} = \{6, 1, 2, 3, 4, 5, 7\}$ is given in Figure 1. We associate with a sequence $\mathcal{S}$, the schedule $\mathcal{T} = \{t_i \mid i \in N\}$ where the starting time t_i of job i is equal to the value of the longest path from s to i in $G^{\mathcal{S}}$. For sequence $\mathcal{S}$, we have $t_1 = 10, t_2 = 15, t_3 = 21, t_4 = 28, t_5 = 32, t_6 = 0, t_7 = 35$. The makespan of this schedule can be calculated as the longest path from s to t in $G^{\mathcal{S}}$. For the given example, this path is given as $\{s, 1, 2, 3, 4, t\}$ with a makespan of 53.

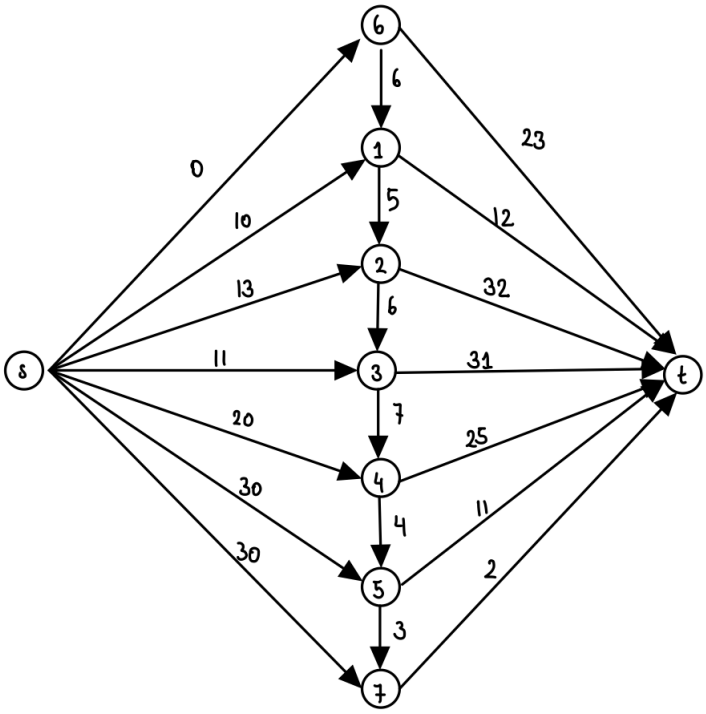

Figure 1 – *Conjunctive graph $G^{\mathcal{S}}$ for the sequence $\mathcal{S} = \{6, 1, 2, 3, 4, 5, 7\}$.*

In the following, we will describe a branch-and-bound algorithm by describing how lower and upper bounds can be obtained, and how branching can be performed.

III. Branch-and-Bound algorithm

1. Lower bounds

A lower bound can be obtained using the following proposition.

Proposition 1. *For all $I \subseteq N$*

$$h(I) = \min_{i \in I} a_i + \sum_{i \in I} d_i + \min_{i \in I} q_i \tag{1}$$

is a lower bound on the optimal makespan.

2. Upper bounds

Upper bounds, obtained as the makespan of a given schedule, can be produced using a heuristic known as the Schrage schedule. Below, we describe the algorithm permitting to construct the Schrage schedule.

Let U be the set of jobs already scheduled, $\overline{U}$ be the jobs to be scheduled, and t be the time.

- Step 1 : Initialize : $U = \emptyset$, $\overline{U} = N$, and $t = \min_{i \in \overline{U}} a_i$.
- Step 2 : At time t, schedule $i \in \overline{U}$ such that $a_i \leq t$ and $i \in \text{argmax}_{i \in \overline{U}} q_i$. Break ties arbitrarily.
- Step 3 : Set $U = U \cup \{i\}$, $\overline{U} = \overline{U} \setminus \{i\}$, $t_i = t$, $t = \max\{t_i + d_i, \min_{i \in \overline{U}} a_i\}$.
  If $U = N$ terminate. Otherwise, go to Step 2.

The makespan of the sequence $\mathcal{S}$ produced by the above algorithm can be calculated as the length of the longest path on the associated conjunction graph $G^{\mathcal{S}}$.

For the instance presented in Table 1, the Schrage algorithm results in the sequence $\mathcal{S} = \{6, 1, 2, 3, 4, 5, 7\}$, and the schedule $t_1 = 10, t_2 = 15, t_3 = 21, t_4 = 28, t_5 = 32, t_6 = 0, t_7 = 35$, with a makespan of 53.

3. Branching

The branching rule is based on the following theorem that can be applied to any Schrage schedule.

Theorem 1. *Let L be the makespan of the Schrage schedule.*

(i) If this schedule is not optimal then there exists a critical job c and a critical set J such that :

$$h(J) > L - d_c \tag{2}$$

where $h(\cdot)$ is as defined previously. Consequently, the distance to the optimum from the Schrage schedule is less than d_c ; moreover, in an optimal schedule, either c will be processed before all the jobs of J, or c will be processed after all the jobs of J.

(ii) If this schedule is optimal, then there exists J such that $h(J) = L$.

Finding the critical job and the critical set Let $\mathcal{C} = \{s, j_1, j_2, \ldots, j_p, t\}$ be the longest path in the conjunctive graph corresponding to sequence $\mathcal{S}$. Either $q_{j_p} = \min_{r \in \{j_1, \ldots, j_p\}} q_r$ in which case the current Schrage schedule is optimal or there exists $k < p$ such that $q_{j_k} < q_{j_p}$. Let $c < p$ be such that c is the largest index with $q_{j_c} < q_{j_p}$. We let $J = \{j_{c+1}, j_{c+2}, \ldots, j_p\}$.

Using Theorem 1, starting from a Schrage schedule, two new nodes can be created based on the critical job j_c and the critical set J found by applying the above rule.

For the sequence $\mathcal{S} = \{6, 1, 2, 3, 4, 5, 7\}$ for which the graph $G^{\mathcal{S}}$ is presented in Figure 1, the longest path is given by $\{s, 1, 2, 3, 4, t\}$. On this path, we have $j_c = 1$ and $J = \{2, 3, 4\}$ as job 1 is the only job on the path with $q_1 < q_4$.

Creating two new nodes :

(i) In the first node, we require job j_c to be processed before all the jobs in J by setting

$$q_{j_c} = \sum_{r \in J} d_r + q_{j_p} \tag{3}$$

where j_p is the last job in J.

(ii) In the second node, we require job j_c to be processed after all the jobs in J by setting :

$$a_{j_c} = \min_{r \in J} a_r + \sum_{r \in J} d_r. \tag{4}$$

Based on $j_c = 1$ and $J = \{2, 3, 4\}$, two new nodes can be created by setting either $q_1 = 38$ (job 1 comes before $\{2, 3, 4\}$), or $a_1 = 28$ (job 1 comes after {2,3,4}).

4. Overall solution scheme

Let τ be a node to be processed.

Let $\mathcal{U}$ be the cost of the best known solution (the upper bound). At the root node this bound can be chosen as $+\infty$.

Let $\mathcal{L}(\tau)$ be the lower bound for τ. At the root node this bound can be chosen as $-\infty$ or can be calculated as $h(I)$ for any $I \subseteq N$.

To process node τ :

(i) Construct the corresponding Schrage schedule $\mathcal{S}(\tau)$. Note that the branching decisions will have changed the instance data.

(ii) Let $\mathcal{C}(\tau)$ be the longest path in the associated conjunctive graph $G^{\mathcal{S}(\tau)}$, and $\mathcal{U}(\tau)$ be its cost.

(iii) If $\mathcal{U}(\tau) < U$, then update $\mathcal{U}$.

(iv) Based on $\mathcal{C}(\tau)$ calculate the critical job $j_c(\tau)$ and the critical set $J(\tau)$. If no such job exists then the current Schrage schedule is optimal, STOP.

(v) Create two new nodes based on the branching rule defined by (3)-(4). A lower bound for the new nodes can be calculated as $\max\{\mathcal{L}(\tau), h(J), h(J \cup \{j_c(\tau)\})\}$ with the updated data for job $j_c(\tau)$.

The branch-and-bound tree for the instance of Table 1 is given in Figure 2. Based on the Schrage sequence $\{6,1,2,3,4,5,7\}$, we identify $j_c = 1$ and $\mathcal{J} = \{2,3,4\}$ and we branch by setting either $q_1 = 38$ or $a_1 = 28$. The lower bound for the left-child ($q_1 = 38$) is $\max\{\mathcal{L}(0), h(\{2,3,4\}), h(\{1,2,3,4\})\} = \max\{-\infty, 49, 53\} = 53$, and therefore this node can be pruned. The lower bound for the right child ($a_1 = 28$) is $\max\{\mathcal{L}(0), h(\{2,3,4\}), h(\{1,2,3,4\})\} = \max\{-\infty, 49, 40\} = 49$, therefore this node needs to be processed.

To process node 2, a new Schrage sequence is calculated as $\{6,3,2,4,1,5,7\}$, and has a makespan of 50 (see Figure 3). The upper bound $\mathcal{U}$ is updated to 50. The longest path is given by $\{s,3,2,t\}$, from which we identify $j_c = 3$ and $\mathcal{J} = \{2\}$ as $q_3 < q_2$. We therefore branch by setting either $q_3 = 32$ or $a_3 = 19$. However, the lower bound for both children is 50, which means that the sequence $\{6,3,2,4,1,5,7\}$ is optimal with value 50.

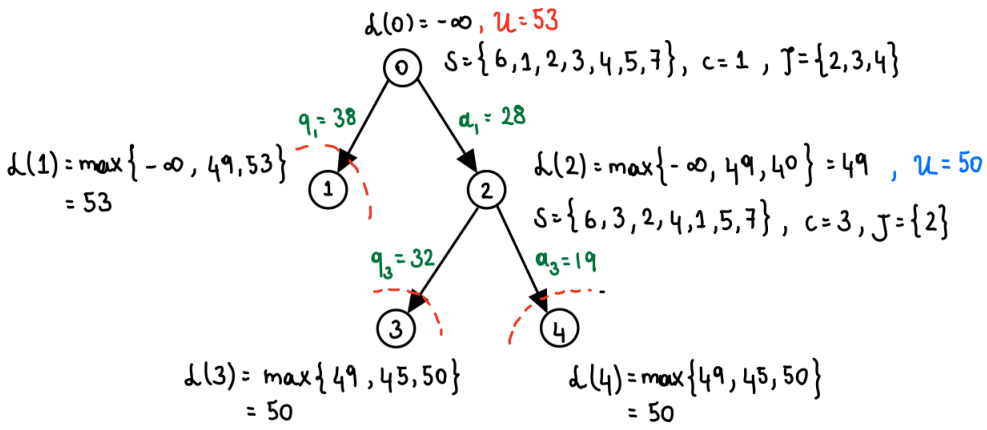

Figure 2 – *The branch-and-bound tree for the instance of Table 1.*

IV. Instances

To test the above algorithm, you may create random instances as follows. For each instance with n jobs, generate $3n$ integers with a uniform distribution between 1 and $a_{\max}$, $d_{\max}$ and $q_{\max}$. You may, for instance, fix $d_{\max}$, and let $a_{\max} = q_{\max} = \frac{1}{50} n d_{\max} K$ with $K \in \{1, \ldots, 25\}$. Note that for testing and debugging purposes it is important to work with the same set of instances. You should therefore fix your random seed throughout the project.

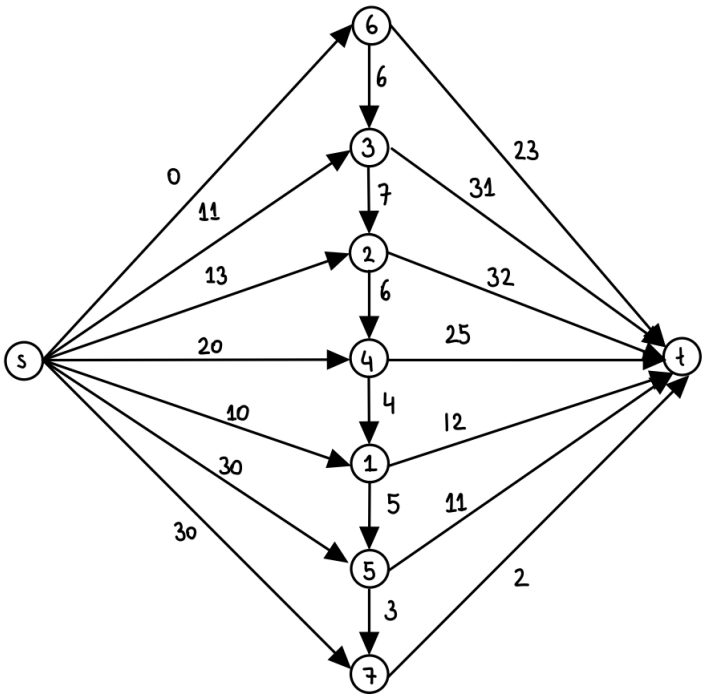

Figure 3 – *Conjunctive graph $G^{\mathcal{S}}$ for the sequence $\mathcal{S} = \{6, 3, 2, 4, 1, 5, 7\}$.*

V. Expected Work

At the end of the project, you should deliver a correct implementation of the branch-and-bound algorithm described above. This implementation should include :

- A data structure representing any instance of the problem
- A data structure for a generic branch-and-bound tree
- A main function branch-and-bound(instance) implementing the branch-and-bound algorithm in a generic manner. The specifics of your implementation can be regrouped in sub-functions.

This branch-and-bound implementation will also be compared to the solution of a MIP formulation of the problem by an optimization solver. To this end :

- Propose a MIP formulation for the problem
- Create a function solve-MIP(instance) which solves your MIP formulation
- Perform a numerical comparison of the two approaches (B&B based on Schrage heuristic) and the solution of the MIP by an optimization solver.

To go further, you may :

- Try to improve the numerical efficiency of your algorithmic implementations
- Test different strategies for node selection within your branch-and-bound algorithm
- Add visual components to your project, such as tracking the B&B tree or presenting the optimal solution for smaller instances

VI. Implementation

Your code will be written in Python, using the package PuLP for solving MIP and LP models coupled with an optimization solver of your choice. Even though Python is not exactly an object-oriented programming language, it disposes of all the necessary functionalities. For a programming project where the data structures are not simple, it is strongly recommended that you use data structures based on classes corresponding to objects that you will manipulate. Therefore, it may for instance be useful to define a class EnumerationTree having as attributes : a set of nodes, the values of bounds etc., and a class Node containing attributes defining each node, its position in the tree, its bounds etc.

It is essential that your code is well commented and your variables/functions/classes have interpretable and intuitive names and formats. A guide to best programming practices in Python is available here and its summary in French can be found here.

VII. Organization :

The project will be done in groups of 2 students, and is **to be delivered before January 5, 2022**. Each group is expected to deliver their code as well as any explanations describing their work as a single Jupyter notebook file. It is strongly advised that each group find a work schedule and a partition of related tasks in order to advance in a timely and consistent manner.