Scheduling with release dates on a single machine to minimize total completion time

Academic project for the course of
Algorithms & Models for Discrete Optimization

UNIVERSITY OF ROME, TOR VERGATA, 2017-2018

Alessio Moretti (0239045) - **alessio.moretti@live.it**

Introduction

The problem

the goal is to find an optimal schedule for a serie of jobs with ***release dates*** (r_j), a given ***processing time*** (p_j), and a ***weight*** (w_j) - all the attributes with positive values - without applying any kind of preemption.

The schedule must be optimal regarding the execution of the jobs on a single machine with the **minimisation of the completion time**, assuming unitary weights. Therefore, we can formulate the problem instance as follows:

$$1|r_j|\sum C_j$$

When all the release dates are equal, it is possible to find an optimal solution sequencing the jobs in non-increasing order of w_j/p_j. However, when release dates assume arbitrary values, the problem is **NP-hard.**

The approach

A **branch-and-bound algorithm** will be proposed, that can be generally used both for weighted and unweighted jobs scheduling, with a relaxation procedure based upon a *job splitting* routine and a set of dominance rules to let the computation tree grow in a limited way.

Job splitting as relaxation technique

Observation

given a job j sequenced immediately after job i then the effect of replacing these two jobs by a single composite job with processing time $p_i + p_j$ and total weight $w_i + w_j$ and increasing the total completion time by $p_j w_i$.

However the **jobs composition leaves the problem unchanged.**

Using the following result of Posner (see *References* section), we have a tool to compute the **lower bound**:

Theorem 2.2. *Suppose σ^* is an optimal schedule for problem P and that σ_1^* is the corresponding optimal schedule for P_1. If σ_2^* is an optimal schedule for problem P_2, then*

$$\sum_{i \in N_1} w_i C_i(\sigma_2^*) + \mathrm{CBRK} \leq \sum_{i \in N_1} w_i C_i(\sigma_1^*) + \mathrm{CBRK} = \sum_{i \in N} w_i C_i(\sigma^*).$$

where

$$\mathrm{CBRK} = \sum_{h=1}^{k-1} w_{i_h} \sum_{j=h+1}^{k} p_{i_j}.$$

CBRK is the cost of breaking the job in k pieces with general splitting (no fixed weight for splitting is applied)

- **Optimal scheduling**
- **Job splitting with constraint** on the contiguity of pieces schedule
- **Job splitting with relaxation** with no constraint on the contiguity of pieces schedule

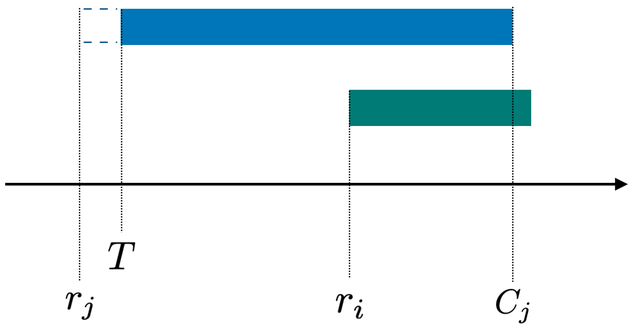
Job splitting used to model preemption

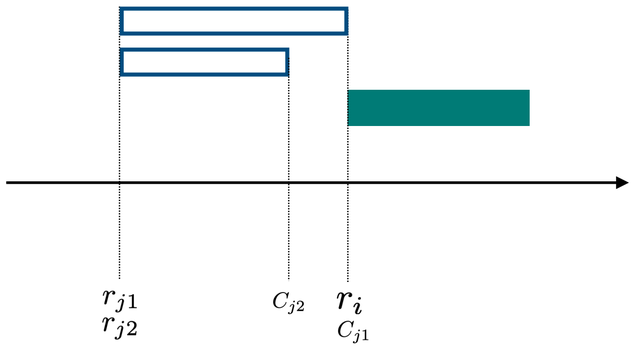
In the literature it is common to find **jobs preemption is a way to obtain lower bounds.**

We can use the job splitting to model the preemption of a job if, during its execution, is found at least another job that:

- has a **shorter weighted processing time** p_j/w_i (in case of unitary weight, a shorter processing time)
- has a **release date before** $T + p_j$ where T is the current execution time and the sum the **completion time**

If the conditions above are satisfied, we can split the selected job as illustrated below





They have the same release date
but a new weighted processing time computed
according to the lighter job release date

Therefore we can obtain the following schedule:

Heuristic algorithm

In this section we will provide a **greedy approach** in order to computer an upper bound to the problem at the root of the branch-and-bound tree and to derive sufficient conditions to have a simple lower bound computation procedure based upon job splitting.

Heuristic H

Step 1 *take the collection of unscheduled jobs*
initialize execution time and weighted completion time

Step 2 *set execution time as the min of all the release dates*
find a subset of feasible jobs with release date lesser or equal to the execution time
select all the feasible jobs with the minimum release date

Step 3 *schedule the job with the minimum weighted processing time* p_j/w_i
remove the job from the unscheduled set
if all the jobs are scheduled exit, else go to Step 2

Assumptions

We are assuming that if a job i is processed when another job j is available, then it must be a better job then the other ant its successors (precedence constraints take the form of *parallel chains*).

Simple Split procedure

The procedure we will use to compute a lower bound, in the branch-and-bound algorithm, for the remaining unscheduled jobs at each level of the tree, **is derived from the Heuristic H.**

We will add a new step to the Heuristic H with the **introduction of the job splitting routine** already described.

Heuristic H

Step 1 *take the collection of unscheduled jobs*
initialize execution time and weighted completion time

Step 2 *set execution time as the min of all the release dates*
find a subset of feasible jobs with release date lesser or equal to the execution time
select all the feasible jobs with the minimum release date

Step 3 *select a feasible job (minimum weighted processing time p_j/w_i)*
if there is any job with release date before the feasible completion time
if shorter weighted processing time
apply job splitting routine

Step 4 *schedule the job with the minimum weighted processing time (splitted jobs included)*
remove the job from the unscheduled set
if all the jobs are scheduled exit, else go to Step 2

They lower bound can be retrieved by the sum of the weighted completion time and the CBRK:

$$LB_{SP} = WC + CBRK$$

Dominance rules

Before proceeding with the description of the branch-and-bound algorithm, we need to establish the dominance rules that are used to limit the number of children for each non-leaf node of the generated tree.
We will refer to ***dominated job*** **if the resulting generated scheduling is not optimal.**

We will use the following rules, also described in Hariri-Potts paper (see *References* section).

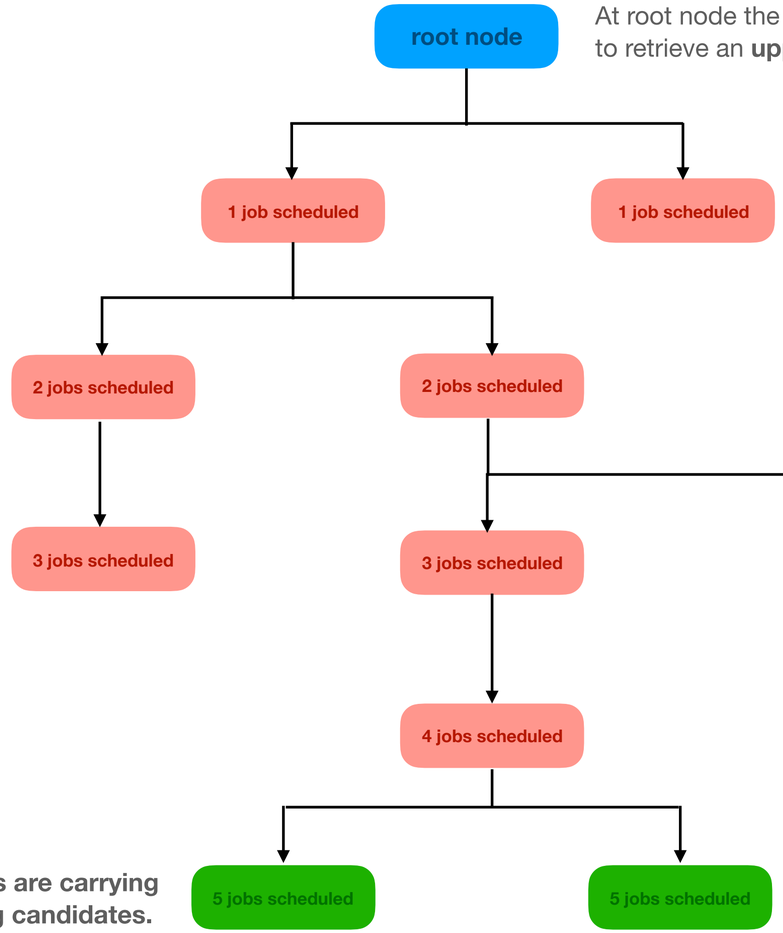
Thm 2 (Rinaldi & Sassano) *any job j is dominated by the feasible job i, if this one has the minimum weighted processing time and has the earliest due date.*

Thm 3 (Dessouki & Deogun) *selected job i can dominate any other job j if its completion time is the minimum amongst the unscheduled set of jobs and its release date is before the minimum completion time.*

Thm 4 (Hariri & Potts)
dynamic programming
weighted completion time selecting job i is less or equal then the one computed choosing job j in other words, if the final two jobs of a partial sequence can be interchanged without increasing the sum of weighted completion times - and leaving machine available for further processing.

If dominance rules are satisfied and there is still unscheduled jobs that are not dominated, the respective children states are computed with their Lower Bound using Procedure SS already described.

Branch-and-Bound



At root node the **Heuristic H** is applied to retrieve an **upper bound** to the solution

Lower bound is computed using Procedure SS. If it not smaller or equal to any other already computed LB, the **active search is stopped.**

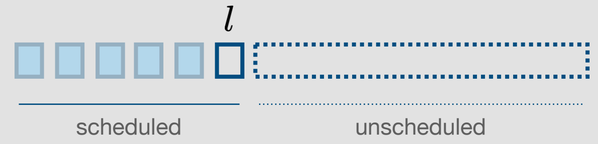
If **dominance theorem 2 is satisfied**, then a single child is created with the same LB of the parent node.

If **dominance theorem 2 is not satisfied**, then the other dominance rules are applied to reduce the number of candidate children states in the tree.

FORWARD SEQUENCING BRANCHING RULE

Leaf nodes are carrying scheduling candidates.

At each level l of the tree, the first l jobs are considered as scheduled and not into consideration for next computations.

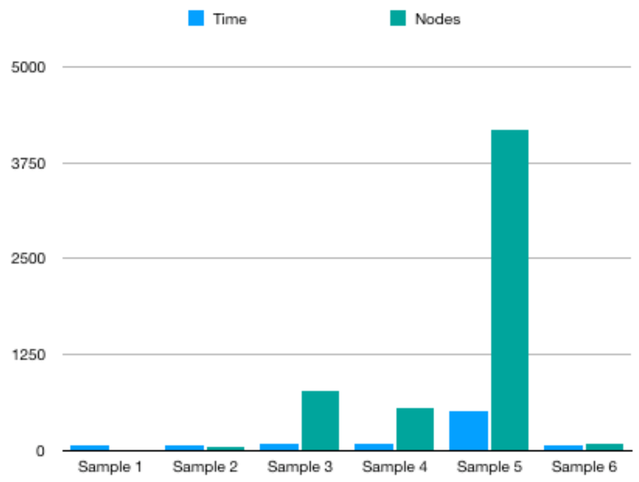


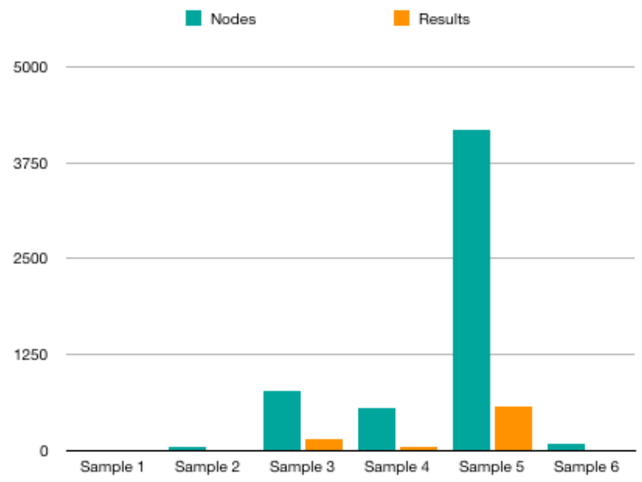
Computational effort

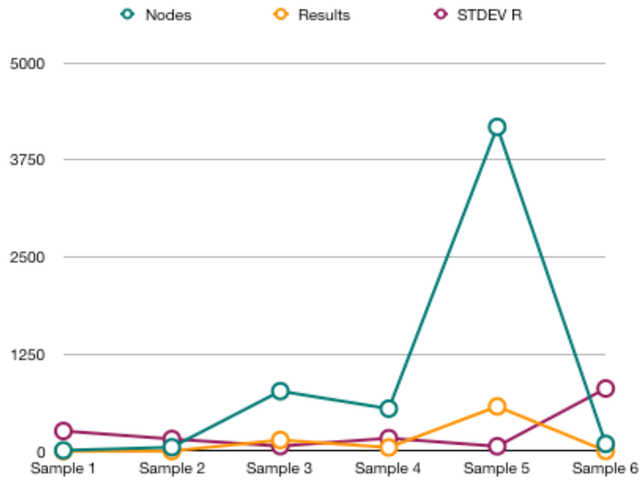
Heuristic H computation	$O(l \log l)$
Lower Bound computation	$O(l \log l)$
Dominance rule application	$O(l)$
Branching rule	$O(l^2 \log l)$

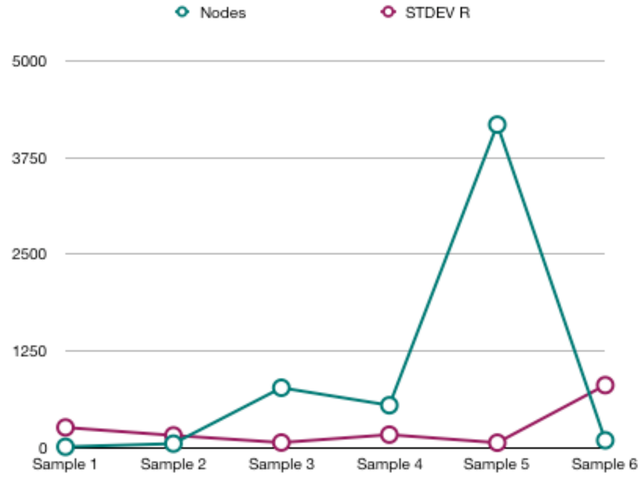
Computational experience

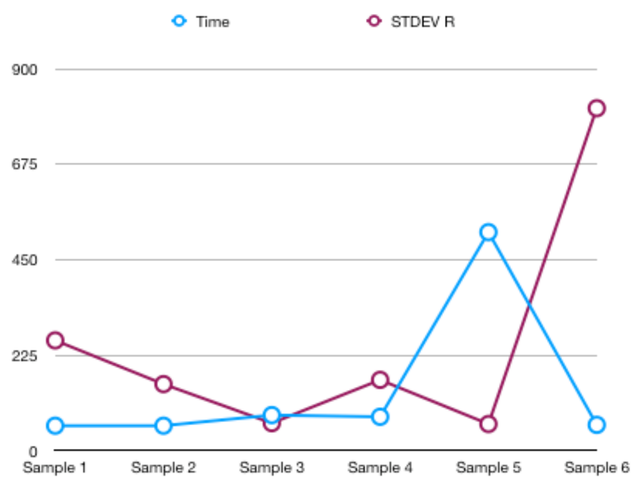
Finally we will dig into the computational experience, taking account of the **number of inputs**, the number of **generated nodes**, the number of **generated feasible results**, the **computational time**.

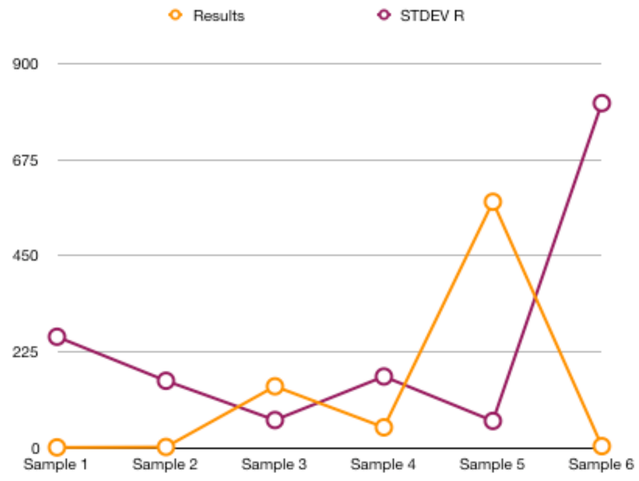












References

An algorithm for single machine sequencing with release dates to minimize total weighted completion time - *A.M.A. Hariri, C.N. Potts (Discrete Applied Mathematics,1983)*

Scheduling with release dates on a single machine to minimize total weighted completion time - *H. Belouadah, M.E. Posner, C.N. Potts (Discrete Applied Mathematics,1992)*

Introduction to scheduling problems - *A. Agnetis (2000)*

The project

Full reference of the project, including this deck and the source code,
is completely open source and available on GitHub.

https://github.com/alessiomoretti/project-razor

Thanks for reviewing this project.

Alessio Moretti
alessio.moretti@live.it

March 5, 2019