Combinatorial Optimization
Lab No. 8
Scheduling and Bratley's algorithm

Industrial Informatics Research Center
http://industrialinformatics.cz/

April 7, 2018

Abstract

The topic of this seminar is scheduling of tasks on resources. We focus on $1|r_j, \tilde{d}_j|C_{max}$ problem, which will be solved by Bratley's algorithm.

1 Scheduling

Scheduling, generally speaking, deals with assigning tasks to resources in time. Let's have a set of tasks T, a set of resources (processors) on which these tasks can be executed, a set of constraints and an optimality criterion. Each task, typically denoted as T_i, is characterized by *processing time* p_i which is the time needed by a resource to process it. Moreover, task T_i can be specified by the following parameters:

- *release time* r_i, the time at which task T_i is ready for processing
- *due date* d_i, a time limit by which task T_i should be completed
- *deadline* $\tilde{d}_i$, a time limit by which task T_i must be completed

$\alpha|\beta|\gamma$ notation, so called Graham-Blazewicz, is commonly used to describe the scheduling problems [1, 2]:

- α contains the characteristics of resources: the number (1 to ∞) and type (P for parallel identical, Q for parallel uniform, etc.)
- β specifies the characteristics of tasks and additional resources (r_j means that each task has release time, *pmtn* means that preemption is allowed, etc.)
- γ denotes an optimality criterion (C_{max} for minimizing the schedule length, L_{max} for minimizing the max. lateness, $\sum U_j$ for minimizing the number of tasks exceeding their due date, etc.)

For example, $P2||C_{max}$ is a formal notation of the scheduling problem where two parallel identical processors are available and objective is to minimize the schedule length (even this "simple" problem is $\mathcal{NP}$-hard). The output of the scheduling is an assignment of tasks to resources in time and it is mostly depicted as a Gantt chart [1] where an independent variable is a discrete time (x-axis) and dependent variable is utilization of resources/processors (y-axis).

2 Branch and Bound algorithm

The Branch and Bound (B&B) algorithm is used to solve discrete optimization problems [1]. This algorithm gradually constructs a tree of partial solutions which are expanded further (branch). If B&B finds an infeasible partial solution or a partial solution such that it is worse than the best found solution, the node with this partial solution is not expanded (bound). Each node of the search tree corresponds to one partial solution and the leaves represent a complete solution. The subtree can be eliminated if:

1. It does not contain any feasible solution.
2. It does not contain an optimal solution.

Basically, the first case implies the second case. However, it can be appropriate to consider them separately for the algorithm construction.

Generally there are two main methods of the solution space searching: breadth-first search and depth-first search. In our case, i.e., the application of the Branch and Bound algorithm in the scheduling, it is advantageous to use the depth-first search since each obtained feasible solution increases the probability of eliminating other parts of the tree without their complete search.

3 The Bratley's algorithm

An example of the use of the Branch and Bound algorithm in the scheduling is a problem formally denoted as $1|r_j, \tilde{d}_j|C_{max}$. It is scheduling on a single processor where each task has a specified release time and deadline, i.e., a time window in which the task has to be processed. The objective is to minimize the schedule length. This problem also belongs to the $\mathcal{NP}$-hard problems, so unless $\mathcal{P} = \mathcal{NP}$ there is no hope of finding deterministic exact algorithm that runs in polynomial time. The problem can be solved, for example, by *Bratley's algorithm* [1] which is based on the branch and bound algorithm. The goal is to search the solution space in the form of a tree where each end node (leaf) represents one solution of the problem. Other nodes are partial solutions consisting of some tasks which are ordered. A node can be pruned on conditions that:

1. **Some tasks exceed their deadline**

   Let's have node N. If there is no option to add arbitrary task at node N without exceeding its deadline, then the partial schedule will never result in the feasible solution. Therefore, all descendant nodes of node N can be eliminated. This follows from the fact that if any of these tasks exceeds its deadline at node N, it will certainly exceed its deadline if scheduled later.

2. **The estimate of the result of the best possible solution in the subtree is worse than the best achieved result so far**

   Let UB be the makespan of the best achieved complete solution so far. At each node, the estimate of the best possible final solution (lower bound) based on the current partial solution can be determined as

   $$LB = \max\left\{c, \min_{T_j \in V} r_j\right\} + \sum_{T_j \in V} p_j \tag{1}$$

   where c is the current schedule length (i.e., the completion time of the last scheduled task) and V is the set of tasks being not scheduled yet. If $LB \geq UB$ than the particular part of the tree can be eliminated because the further branching of the current node cannot find the optimal solution.

3. **A partial solution is optimal**

   If the following condition

   $$c \leq \min_{T_j \in V} r_j \tag{2}$$

is satisfied in node N, then the partial schedule is optimal and only the current node is expanded further. Node N becomes a root node of the whole search tree since going back in the original tree of partial solutions cannot return a better schedule.

An input example of the problem $1|r_j, \tilde{d}_j|C_{max}$ is on the left side in Figure 1. It is a problem with four tasks having their own processing time, release time and deadline. The problem is solved by Bratley's algorithm based on the branch and bound algorithm. The final solution in the form of a Gantt chart is on the right side in Figure 1.

Input:
$p = (2, 1, 2, 2)$
$r = (4, 1, 1, 0)$
$\tilde{d} = (7, 5, 6, 4)$

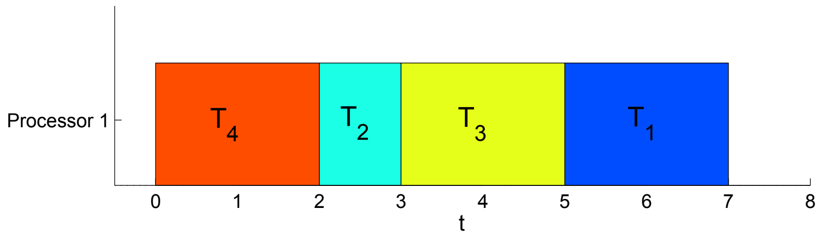

Figure 1: An example of the problem $1|r_j, \tilde{d}_j|C_{max}$ and its optimal solution

A homework assignment: Implement Bratley's algorithm. Upload your source codes to the Upload System.

Hint: To pass the evaluation successfully, you need to implement all three elimination rules, as given above. Otherwise, the running time of your algorithm will be damn too high!

3.1 Input/Output format

Your program will be called with two arguments: the first one is absolute path to input file and the second one is the absolute path to output file which has to be created by your program.

Let n be the number of tasks. Then the input file has $n + 1$ lines and has the following form

$$\begin{array}{lll} n & & \\ p_1 & r_1 & \tilde{d}_1 \\ p_2 & r_2 & \tilde{d}_2 \\ \vdots & & \\ p_n & r_n & \tilde{d}_n \end{array}$$

One space is used as a separator between values on one line. All the values in the input file are non-negative integers.

If the input instance is infeasible, then the output file consists of the single line containing -1. On the other hand, if the input instance is feasible, then the output file consists of n lines and has the following form

$$\begin{array}{l} s_1 \\ s_2 \\ \vdots \\ s_n \end{array}$$

where s_i is the optimal start time of task T_i. All the values in the output file are integers.

Example 1

Input:

```
4
2 4 7
1 1 5
2 1 6
2 0 4
```

Output:

```
5
2
3
0
```

Example 2

Input:

```
3
2 4 7
1 1 2
2 1 2
```

Output:

```
-1
```

References

[1] J. Blazevicz, *Scheduling Computer and Manufacturing Processes*. Springer, second ed., 2001.

[2] R. Graham, E. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, “Optimization and approximation in deterministic sequencing and scheduling theory: a survey,” *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.