Combinatorial optimization
Optional Homework: Employee Scheduling in an IT Company problem and List Scheduling Algorithm

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

Abstract

This homework is devoted to the problem of Employee Scheduling in an IT Company, which is a modified version of Resource-Constraint Project Scheduling problem. To solve the problem of employee scheduling in an IT company, you will implement a specialized list scheduling algorithm.

1 Resource-Constrained Project Scheduling Problem

A resource-constrained project scheduling problem (RCPSP) [1] deals with resources of certain capacities and tasks of known processing times and resource demands, linked by precedence relations. The problem is to find a schedule of minimal length such that the precedence relations and the resource capacities are respected.

Formally, the RCPSP is defined by a 6-tuple $\langle A, p, G, R, B, b\rangle$:

- *Tasks* of the project, denoted by set $A = \{A_1, ..., A_n\}$.
- *Processing times* $p_i \in \mathbb{N}$ of task A_i.
- *Precedence relations* are given by a graph $G = (V, E)$ with vertices being the tasks $V = A$ and edges being a pair precedence relation. Therefore, if $(A_i, A_j) \in E$, then A_i must precede A_j in the schedule.
- *Resources* are formalized by set $R = \{R_1, \cdots, R_q\}$.
- Resource *demands* of tasks are denoted by $b \in \mathbb{N}^{n\times q}$, where $b_{i,v}$ is the amount of resource R_v that task A_i uses per time unit.
- *Capacities* B_v of resource R_v. In particular, a resource with capacity $B_v = 1$ is called disjunctive resource and it is considered in Lab 11 that dealt with Bratley algorithm.

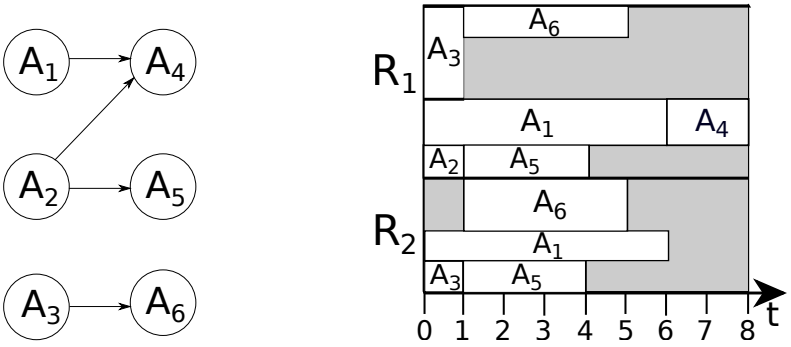
A solution to RCPSP is a schedule $S \in \mathbb{N}^n$ which defines start time S_i of each task A_i. An optimal solution S for RCPSP has minimal length as in Equation (1) and satisfy precedence (Equation (2)) and resource (Equation (3)) constraints, where $A_t = \{A_i \in A \mid S_i \leq t < S_i + p_i\}$ represents the set of tasks being processed at time t.

$$\textit{Minimize:} \quad \max_{A_i \in A}(S_i + p_i) \tag{1}$$

$$S_j - S_i \geq p_i, \qquad \forall (A_i, A_j) \in E \tag{2}$$

$$\sum_{A_i \in A_t} b_{iv} \leq B_v, \qquad \forall R_v \in R, t \geq 0. \tag{3}$$

From the definition of set A_t follows that a task cannot be interrupted once it is started, which means that we schedule tasks *non-preemptively*.



(a) Graph of precedence relations of the considered project

(b) Example of the solution to the considered project

Figure 1: Project specification and solution.

Table 1: A project with 6 tasks and 2 resources

A_i	A_1	A_2	A_3	A_4	A_5	A_6
p_i	6	1	1	2	3	4
b_{i1}	2	1	3	2	1	1
b_{i2}	1	0	1	0	1	2

1.1 An Example of RCPSP

Let us consider an RCPSP instance that is given in Table 1.1 with 6 tasks and 2 resources with capacities $B_1 = 7$ and $B_2 = 4$. The precedence relation graph of this instance is displayed in Figure 1a. A schedule of minimal length $S_4 + 2 = 8$ is displayed in Figure(1b) as a 2-dimensional Gantt chart where the x axis is the time and the y axis is the resource occupation.

2 Employee Scheduling in an IT Company

Let us look at the problem of assignment project tasks to employees (developers) in an IT company. Here, the project typically corresponds to the software development, where tasks are the analysis of the client requirements, development of the application, tests, etc. Tasks are time-dependent on each other, limited in time and each task needs particular number of employees and skills to be performed, depending on the project requirements: analysis method, programming language (Python, C++, Java, etc.), etc. Thus, the problem is to find a solution, i.e. an assignment of the project tasks to the employees in time, such that the project constraints are not violated and the project duration is minimized.

Formally, the problem comprises n tasks that are to be scheduled on q resources that are employees. The tasks are time dependent with a precedence relations graph G and demands on the number of required employees e_i working on this task simultaneously for p_i hours. Additionally to RCPSP formulation, each task is limited in time by its release time r_i. Moreover, one specific skill k_i is required to perform task A_i and for each employee R_v there is a set of skills $l_v = \{k_1, \cdots, k_{m_v}\}$ it is capable of.

Note that this problem is a modification of RCPSP, since instead of providing strict demands of tasks on each resource as in RCPSP (given by matrix b), a constraint given by skills is introduced that only limits which resources can process a particular task. Moreover, disjunctive resources (i.e. with capacities $B_v = 1$) are considered. Therefore, the employee scheduling in an IT company problem solution is defined by both vector of start times $S = (S_1, S_2, \cdots, S_n)$ (i.e. each task is done on all e_i resources at the same time) and a vector of resource assignments $z_i = [z_{i,1}, z_{i,2}, \cdots, z_{i,e_i}]$ for each task T_i.

Table 2: Task parameters of the considered project

A_i	p_i	r_i	k_i	e_i
A_1	3	1	2	1
A_2	1	0	1	2
A_3	1	0	3	1
A_4	2	3	2	1
A_5	3	3	1	1
A_6	4	4	3	1

R_v	l_j
R_1	{1}
R_2	{2,3}
R_3	{1,2,3}

The parameters for the problem instance with $n = 6$ project tasks and $q = 3$ employees that we consider are listed in Table 2, while G is the same as in the previous section (Figure 1a). The example solution is shown in Figure 2.

Figure 2: Example of the solution to the Employee Scheduling in an IT Company problem.

3 List Scheduling Algorithm

List scheduling algorithm is based on the idea of gradually constructing the schedule task-by-task in some predefined order of tasks that respects precedence and timing constraints, putting the tasks to the earliest possible place on free resources.

The complete pseudocode of the modification of List Scheduling algorithm for the Employee Scheduling in an IT company problem is given in Algorithm 1. The two main sets of variables that list scheduling algorithm operates on are the time availability t_v of each resource R_v and start times s_i of each task A_i. First of all, the list scheduling algorithm sorts the list of all tasks L according to some priority rule. Afterwards, it continues in iterations and at each iteration it chooses the first task in L that is free (i.e. all of its predecessors are already scheduled). Then, it schedules the task at the first possible time moment, respecting the skills constraints, precedence relations, release dates and demand on the number of required resources. Lastly, task from list L is removed and the algorithm continues in the next iteration.

Algorithm 1 List Scheduling Algorithm for Employee Scheduling Problem

Require: number of resources q; number of non-preemptive tasks n; resource skills $l_j = [k_1, \cdots, k_{m_j}]$ for each resource j; task processing times p; digraph of precedence relations G; required skills for tasks k and release dates r.

Ensure: start times $s = [s_1, s_2, \cdots, s_n]$ and resource assignments $z_i = [z_{i,1}, z_{i,2}, \cdots, z_{i,e_i}]$ for each task T_i.

function LS-EMPL(q, n, l, p, G, k, r)
  $t_v := 0$ for all $v \in \{1, 2, \cdots, q\}$;
  $s_i := 0$ for all $i \in \{1, 2, \cdots, n\}$;
  Sort tasks in list L;
  for $w := 1$ `to` n **do**
    In the set of free tasks, choose T_i which is the first in list L;
    $R^w := \{R_j | k_i \in l_j\}$; ▷ resources with necessary skill
    Choose e_i resources from R^w with minimal t_v: $R^{chosen} := \{R^w_{j_1}, \cdots, R^w_{j_{e_i}}\}$;
    $s_i := max\{max_{R_v \in R^{chosen}}\{t_v\}, max_{T_j \in Predecessors(T_i)}\{s_j + p_j\}, r_i\}$;
    Remove T_i from L;
    $z_i := R^{chosen}$;
    $t_{R^{chosen}_1} := t_{R^{chosen}_2} = \cdots = t_{R^{chosen}_{e_i}} = s_i + p_i$;
  end for
  return s, z;
end function

3.1 A homework assignment

A homework assignment: Implement LS-Empl algorithm with an appropriately chosen priority assignment strategy. Upload your source codes to the Upload System where it will be automatically evaluated.

3.2 Input and 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 (the output file has to be created by your program).

The input file has the following format. The first line of the input file consists of a natural number n specifying the number of tasks and a natural number q specifying the number of resources. The following lines consist of vector of required skills for each task k, vector of tasks resource demands e, vector of processing times p and vector of release dates r. Afterwards, the next q lines consist of vectors $l_j = [k_1, \cdots, k_{m_j}]$ for $j \in \{1, \cdots, q\}$ each one on a new line ending with 0. The last n lines are numbers of tasks that must precede task T_i in graph G, each line also ending with 0.

The output file consist of $n + 2$ lines, each with integer numbers. The first line represents the makespan value, i.e. the latest completion time over all tasks. The second line contains the vector of start times S and the n following lines display the resource assignment vectors z_i for $i = 1, \cdots, n$ (order of employees in z_i can be arbitrary).

Example 1

Input:

```
6   3
2   1   3   2   1   3
1   2   1   1   1   1
3   1   1   2   3   4
1   0   0   3   3   4
1   0
2   3   0
1   2   3   0
0
0
0
1   2   0
2   0
3   0
```

Output:

```
8
1   0   0   4   3   4
2
1   3
2
2
1
3
```

References

[1] C. Artigues, S. Demassey, and E. Neron, *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons, 2013.