Resource-constrained project scheduling

Resource-constrained project scheduling

How can a manufacturing company complete the processing of tasks in minimal time when each task requires a certain amount of constrained resource?

The Resource Constrained Project Scheduling Problem (RCPSP) is a generalization of the production specific Job-shop, Flow-shop, and Open-shop Scheduling Problems. Given:

- a set of tasks to be performed,
- a set of `q` resources with given capacities,
- a network of precedence constraints between the tasks, and
- for each task and each resource, the amount of the resource that is required by the task over its execution,

the objective of the RCPSP is to determine a schedule that meets all the constraints and whose makespan (that is, the time at which all tasks are finished) is minimal.

This problem is an example of a scheduling problem in which tasks are represented by special variables that are called interval variables. An interval has a start time, an end time, and a duration. There are special constraints and expressions for intervals, including precedence constraints and resource capacity constraints, which are used to model this problem.

The data

The instance of RCPSP under consideration is the instance `j120_1_2.rcp`[1], that was proposed in 1996. This instance was solved to optimality for the first time in 2009.

The problem consists of 120 tasks and 4 resources.

Each task is described by its duration, the set of successor tasks and the required quantity of resources. For instance, task 1 has a duration of 1, must be performed before tasks 4, 5 and 28, and requires three units of resource 3.

Resources are characterized by their maximal capacity. For instance, resource 3 has a maximal capacity of 13. By convention, there is a task 0 (with duration 0) that is before all the other tasks. There is also a task 121 (with duration 0) that is after all other tasks. The objective of the problem is to find a schedule that minimizes the end time of task 121.

The model

To model this problem, the unknowns, constraints, and objective must be determined. The unknowns are the times that the tasks will start/end. In this model, there are constraints on the order of some pairs of tasks. There is also a set of constraints to represent the limit to the amount of resources that are required during the processing of the tasks. The objective is to minimize the end (completion) time of the final task.

1. R. Kolisch and A. Sprecher. "PSPLIB - A project scheduling problem library". European Journal of Operational Research, 96:205–216, 1996.

Modeling the variables

The model uses a set of interval variables to represent the tasks. The size of each interval variable represents the duration of each task. For instance, task 1 is represented as an interval variable `Task_1` with size 1:

```
Task_1 = intervalVar(size=1);
```

Modeling the constraints

The RCPSP model uses two types of constraints: precedence constraints between tasks and resource capacity constraints.

Precedence constraints between tasks are expressed with `endBeforeStart`. For instance:

```
endBeforeStart(Task_1, Task_4);
```

states that `Task_1` must end before `Task_4` starts.

Resource capacity constraints are expressed by using the cumulative function. The function `pulse(itv,q)` is an elementary cumulative function with value 0 outside of the interval variable `itv`. Between the start and the end of the interval variable, the function has the specified value, `q`. The cumulative function is the sum of all the contributions of the tasks that are performed on the resource. A constraint `cumul function <= capacity` is posted to limit the maximal value of the cumulative function. For instance, on resource 3 with maximal capacity 13, the constraint is:

```
pulse(Task_1, 3) + pulse(Task_2, 1) + ...  <= 13;
```

Modeling the objective function

The objective of this problem is to minimize the end time (makespan) of the schedule. Because of the precedence constraints, `Task_121` is the last task of the schedule. Thus the objective function is:

```
minimize(endOf(Task_121));
```

The solution

The optimal solution to the problem has a makespan value of 109.