Lot-Sizing Problem

Problem specification
for the project of the course
Advanced Scheduling Systems (2016-17)

version 1.1 (Feb 5, 2016)

We consider a simplified version of the Discrete Lot-Sizing Problem defined by Pochet and Wolsey [2, Chapter 14] and listed in [1, Prob. 58]. Instances are artificial and created by a generator.

The problem consists of the following entities.

- A set $I = \{0, \ldots, m-1\}$ of items that represent the goods that must be produced.
- A set $P = \{0, \ldots, p-1\}$ of periods in which it is possible to produce.
- A single machine that, in any period can produce exactly one piece of an item.
- An binary $I \times P$ matrix D such that $d_t^i = 1$ if a piece of item i is requested at time t, $d_t^i = 0$ otherwise.
- An integer $I \times I$ matrix C such that c^{ij} represents the cost in setting up the machine from producing item i to producing item j. We assume $c^{ii} = 0$ for all i.
- An integer-valued vector H of size m such that h^i represents the cost for stocking one piece of item i for one period.

The problem consists in determining the item produced by the machine in any period of P. We assume that $\sum_{i=0}^{m-1} \sum_{t=0}^{p-1} d_t^i = p$, so that the total number of pieces to be produced is equal to the number of periods.

A solution is thus a vector V of size p, whose elements are values in the set I. The value $v_t \in I$, with $t \in P$, represents the item produced at time t.

For each $i \in I$, the number of t such that $v_t = i$ must be equal to $\sum_{t \in P} d_t^i$, so that the total number of pieces produced of an item must equal its total demand.

The problem includes also the following hard constraint:

- NoBacklog: Each piece must be produced not later than when it is requested. This means that for any item i, at any time $t \in P$ the number of pieces produced until t must be greater or equal to the requests of i up to t.

The problem asks to find a solution that is feasible with respect to the NoBacklog constraint, and that minimizes the setup and stocking costs. The objective function is thus composed by the following two components (soft constraints):

- StockingCost: Sum of the stocking periods of all pieces of all items multiplied by the stocking cost h^i of each item i.
- SetupCost: Sum of setup costs for all periods $t \in P$. It is assumed that no setup cost is added for the first produced piece.

Figure 1 shows a file that contains a small instance.

```
Periods = 8;
Items = 3;
Demands = [|0, 0, 0, 0, 0, 1, 0, 1
           |0, 0, 1, 1, 0, 0, 1, 1
           |0, 0, 0, 0, 0, 1, 1, 0|];
StockingCosts = [10, 15, 12];
SetupCosts = [|0, 131, 109
              |193, 0, 175
              |101, 136, 0|];
```

Figure 1: A file containing a toy instance in MiniZinc format

A solution is written as a MiniZinc array. For example the array

```
[0, 0, 1, 1, 2, 2, 1, 1]
```

is a feasible solution of the instance of Figure 1. In fact, we see that no backlog occurs, its setup cost is $c^{0,1} + c^{1,2} + c^{2,1} = 131 + 175 + 101 = 442$, and its stocking cost is $5h^0 + 6h^0 + h^2 + h^2 = 50 + 60 + 12 + 12 = 134$. The total cost is then $442 + 134 = 576$.

References

[1] Ian P Gent and Toby Walsh. CSPLib: a benchmark library for constraints. In *Principles and Practice of Constraint Programming (CP'99)*, pages 480–481. Springer, 1999. Available from `http://www.csplib.org`.

[2] Yves Pochet and Laurence A Wolsey. *Production planning by mixed integer programming.* Springer Science & Business Media, 2006.