Problema del Lot-Sizing: specifica

Il problema che è stato analizzato viene chiamato Lot-Sizing. Si tratta di un problema di *production planning* a una macchina.
I seguenti dati vengono forniti per ogni istanza:

- Un insieme $I = \{0, \ldots, m-1\}$ che rappresenta i tipi di oggetto che possono essere prodotti.
- Un insieme $P = \{0, \ldots, p-1\}$ che rappresenta i periodi temporali (discreti) durante i quali produciamo un oggetto.
- Una matrice $I \times P$ (la chiamo D) tale che $d_t^i = 1$ se un oggetto di tipo i è richiesto al tempo p, $d_t^i = 0$ altrimenti.
- Una matrice $I \times I$ (la chiamo C) tale che c^{ij} rappresenta il costo che va pagato quando la macchina passa dalla produzione di un oggetto di tipo i ad uno di tipo j ($\forall i \; c^{ii} = 0$).
- Un array H di lunghezza m tale che h^i rappresenta il costo dato dal tenere in magazzino un oggetto di tipo i per un periodo (una unità temporale).

Assumendo di dover produrre esattamente un oggetto per ogni periodo, il problema consiste nel determinare quale oggetto vada prodotto dalla macchina in ciascun periodo di tempo. La soluzione quindi può essere rappresentata tramite un array V di lunghezza p, dove ogni V^t rappresenterà il tipo di oggetto da produrre al tempo t.

Si ha un unico constraint di tipo hard che possiamo chiamare **NoBacklog**: ogni oggetto deve essere prodotto non più tardi rispetto a quando è richiesto. In altre parole, per ogni oggetto i e ogni periodo t, il numero di oggetti i prodotti fino al tempo t deve essere maggiore o uguale al numero totale di richieste di i fino al tempo t.

La **funzione obiettivo** da minimizzare è costituita da due elementi:

- **StockingCost**: è la somma dei costi di stoccaggio di tutti gli oggetti. Questi costi vengono generati quando la macchina produce un oggetto al tempo t ma l'oggetto è richiesto solamente "dopo" (ad un tempo $t' > t$).

- **SetupCost**: è la somma dei costi di setup della macchina: vengono generati ogni volta che questa passa dal produrre un tipo di oggetto ad uno diverso. Il primo oggetto prodotto genera un SetupCost nullo.

Riporto qui di seguito una piccola istanza esemplificativa in formato MiniZinc:

```
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|];
```

con la corrispondente soluzione ottima:

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

di costo 569.