UNIVERSITÀ
DEGLI STUDI
FIRENZE

UNIVERSITÀ DEGLI STUDI DI FIRENZE

DIPARTIMENTO DI INGEGNERIA INFORMATICA

Elaborato di Intelligenza Artificiale
Negozi e Magazzini

Autore:
Haka Alban

N° Matricola:
7118416

Corso:
Intelligenza artificiale

Docente corso:
Paolo Frasconi

Introduzione

Problema
Una catena di distribuzione deve decidere sull'acquisto di un certo numero di magazzini, m, da utilizzare per lo stoccaggio di merci. I magazzini dovranno essere operativi per un solo anno. Ci sono inoltre n negozi da rifornire, ciascuno rifornito da un solo magazzino. Il costo complessivo (durante tutto l'anno) per rifornire il negozio i dal magazzino j è c_{ij}, $i = 1, \ldots, n$; $j = 1, \ldots, m$. Ogni magazzino può rifornire al massimo S_j negozi, $j = 1, \ldots, m$, $1 \leq S_j \leq n$. Alla fine dell'anno i magazzini saranno rivenduti. I prezzi di acquisto e di vendita dei magazzini sono rispettivamente A_j e V_j, $j = 1, \ldots, m$. Si determini quali magazzini conviene acquistare e, per ciascun negozio, quale sia il magazzino che lo dovrà rifornire, in modo da minimizzare il costo totale dell'operazione. Si assuma che tutti i costi siano numeri interi compresi tra 10 e 100.

Obiettivo:
L'obiettivo del modello è determinare la soluzione ottima minimizzando il costo totale. L'implementazione del modello è stata sviluppata con MiniZinc.

Formalizzazione del problema

Variabili

- n: numero di negozi
- m: numero di magazzini
- c_{ij}: costo annuale per rifornire il negozio i dal magazzino j
- S_j: numero massimo di negozi rifornibili dal magazzino j
- A_j: costo di acquisto del magazzino j
- V_j: prezzo di rivendita del magazzino j
- $supplier_i$: magazzino che rifornisce il negozio i
- a_j: variabile booleana, vale 1 se il magazzino j è stato acquistato, 0 altrimenti

Domini

- $N = \{1, \ldots, n\}$: insieme degli indici dei negozi
- $M = \{1, \ldots, m\}$: insieme degli indici dei magazzini
- $C = \{10, \ldots, 100\}$: dominio dei costi di fornitura
- $S_j \in \{1, \ldots, n\}$: dominio della capacità di ciascun magazzino
- $a_j \in \{0, 1\}$: dominio booleano per l'acquisto dei magazzini
- $supplier_i \in M$: dominio dei fornitori assegnabili ai negozi

Vincoli

- ogni negozio è rifornito da un solo magazzino:

$$\forall i \in N, \quad supplier_i \in M$$

- un negozio può essere rifornito solo da un magazzino acquistato:

$$\forall i \in N, \quad a_{supplier_i} = 1$$

- un magazzino non può rifornire più negozi rispetto alla propria capacità:

$$\forall j \in M, \quad \sum_{i \in N} [supplier_i = j] \leq S_j$$

Funzione obiettivo

Minimizzare il costo totale:

$$\sum_{i=1}^{n} c_{i,supplier_i} + \sum_{j=1}^{m} a_j \cdot (A_j - V_j)$$

Implementazione in MiniZinc

Il modello CSP è stato sviluppato utilizzando MiniZinc, un linguaggio di modellazione dedicato alla risoluzione di problemi di soddisfacimento di vincoli e ottimizzazione. Per la risoluzione del problema, è stato scelto il solver Gecode 6.3.0, uno dei più veloci e performanti, ottimale per problemi di pianificazione con vincoli complessi. I dati di input sono stati forniti tramite file `.dzn`, un formato di MiniZinc che separa la definizione del modello dai dati specifici. Questo approccio permette di fornire al modello un set di dati diversi, in modo da poterlo testare con istanze differenti e analizzare al meglio il suo corretto funzionamento.
Il problema è stato implementato nel seguente modo:

```
int: n;
int: m;

% Domini
set of int: N = 1..n;
set of int: M = 1..m;
set of int: C = 10..100;

% Variabili
array[N, M] of var C: c; % costo c[i,j] per rifornire il negozio i dal
    magazzino j
array[M] of int: S; % capacita' di ciascun magazzino
array[M] of int: A; % costo di acquisto di ciascun magazzino
array[M] of int: V; % valore di rivendita
array[M] of var bool: a; % true se il magazzino e' acquistato
array[N] of var M: supplier; % supplier[i] e' il magazzino che rifornisce
     il negozio i

% Vincoli
constraint forall(i in N)(
    a[supplier[i]] % vincolo che garantisce che ogni negozio e' rifornito
         solo da magazzini acquistati
);

constraint forall(j in M)(
    sum(i in N where supplier[i] = j)(1) <= S[j] % vincolo sulla capacita
        ' massima di ciascun magazzino
);

% Funzione obiettivo: minimizzare il costo totale
var int: total_cost =
    sum(i in N)(c[i, supplier[i]]) +  % costi di rifornimento
    sum(j in M)(bool2int(a[j]) * (A[j] - V[j])); % costo netto per ogni
        magazzino

solve minimize total_cost;
```

Dati di esempio

I dati di esempio sono i seguenti:

```
n = 4; % numero di negozi
m = 3; % numero di magazzini

c =[| 15, 20, 18,
    | 16, 22, 19,
    | 14, 18, 21,
```

```
    | 12, 17, 20|]; % costi di rifornimento

S = [2, 2, 1]; % capacita' dei magazzini
A = [50, 60, 40]; % costo di acquisto dei magazzini
V = [10, 15, 10]; % valore di rivendita dei magazzini
```

Output ottenuto:

- Magazzini acquistati: 1 e 3
- Assegnazioni:
  - Negozio 1 → Magazzino 1
  - Negozio 2 → Magazzino 1
  - Negozio 3 → Magazzino 3
  - Negozio 4 → Magazzino 3
- Riepilogo:
  - Magazzino 1: 2 negozi
  - Magazzino 2: 0 negozi
  - Magazzino 3: 2 negozi
- Costo totale: 12 + 13 + 20 + 10 + (30 + 20) = 105

Conclusioni

In questa implementazione, l'algoritmo ha individuato una soluzione ottimale che minimizza il costo totale, rispettando tutti i vincoli relativi alla capacità dei magazzini, al rifornimento dei negozi e all'acquisto dei magazzini stessi. Nei file `.dzn` sono state create due istanze diverse con un numero maggiore di parametri, al fine di testare in modo più efficace l'algoritmo. Per maggiori dettagli, consultare il file `README`.

Riferimenti

- `https://www.minizinc.org/`
- Documentazione MiniZinc: `https://www.minizinc.org/doc-2.6.3/en/`