**Il puzzle della sequenza**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | Base | | Base+Implicita | | Globale | | Globale+Implicito | |
|  | Non riesce | Tempo | Non riesce | Tempo | Non riesce | Tempo | Non riesce | Tempo |
| 500 | 618 | 21s | 495 | 15 secondi | 989 | 1s82ms | **493** | **907 ms** |
| 1000 | 1743 | 35 minuti e 46 secondi | 995 | 1 minuto e 5 secondi | 1989 | 2s443ms | **993** | **1s635ms** |

La tabella sopra riporta i guasti e il tempo totale di due modelli implementati per il problema Puzzle.   
Ciascuno di essi è stato eseguito con o senza la presenza di due vincoli impliciti.

Mentre la differenza tra la presenza o meno dei vincoli impliciti era già descritta nel precedente problema Puzzle, qui ci concentriamo sulla differenza data dalla globalizzazione dei vincoli del problema.

Mentre nell'esercizio precedente per il problema nQueens abbiamo visto un miglioramento nel modello con vincoli globali, qui il modello base presenta meno fallimenti rispetto a quello globale .   
Probabilmente la ragione potrebbe essere la migliore espressività del vincolo scomposto rispetto a quello globale.

Tuttavia il modello globale , con e senza vincoli impliciti, richiede meno tempo rispetto al modello base a causa di una propagazione più efficiente .

* Sì, il numero di guasti nel modello base è inferiore a quello del modello globale. Inoltre, i modelli base+implicito e globale+implicito presentano quasi lo stesso numero di fallimenti. Eppure il modello globale+implicito è molto più veloce in entrambi i casi. Ok, sappiamo che la propagazione con i GC è molto più efficiente. Ma **cosa c'è di sbagliato nei meta vincoli che rallentano significativamente il risolutore** , anche se in termini di fallimenti sembrano essere abbastanza buoni?

Il Modello Base richiede molto più tempo di quello Globale perché anche se i vincoli scomposti possono rilevare la stessa incoerenza dei vincoli globali, probabilmente devono impiegare molto più tempo nella propagazione per avere lo stesso effetto. Quindi devono propagare molti più vincoli per ottenere la stessa propagazione.

* I meta vincoli non rallentano i tempi di risoluzione. Se osservi cosa sta succedendo, **la maggior parte del tempo viene dedicata alla compilazione** . La domanda è: perché impiegano molto **tempo per essere compilati** ?

Osservando il tempo di compilazione noto che il flatTime rispetto al tempo totale è molto. Questo perché il FlatZinc generato produce milioni di variabili piatte e vincoli piatti. Quindi probabilmente il motivo per cui i meta vincoli rallentano il risolutore è dovuto al tempo necessario per la compilazione di ciascun vincolo .   
Infatti , vedo che ci sono circa 250.000 vincoli reificati nelle statistiche di compilazione, ovvero vincoli creati dalla reificazione per gestire espressioni con connettivi logici.

* Esattamente! **I meta vincoli sono vincoli su vincoli, ma per poterlo fare, il risolutore crea variabili booleane** . Quindi , nel tempo O (n^2) crea booleani per ogni espressione X[j] ==i, in modo **simile al modo in cui abbiamo eseguito la scomposizione between** (dove abbiamo creato noi stessi tali booleani nel modello, mentre con i meta vincoli il il risolutore lo fa automaticamente per nostro conto).
* Nel modello globale+implicito , sono utili entrambi i vincoli impliciti e perché/perché no?

Sì, noto che uno dei vincoli impliciti non migliora il risultato. Infatti il secondo vincolo implicito è sufficiente per esprimere lo stesso vincolo.

* Perché pensi che il primo vincolo implicito sia diventato ridondante con il modello GCC ?

Il primo vincolo implicito è ridondante perché invece di sommare tutti gli elementi nella sequenza per contare il numero di occorrenze di un numero, posso solo guardare ciascun elemento X[i] e moltiplicarlo per i per calcolare la stessa informazione.

* Quanto al vincolo implicito: no, non è questo il motivo, altrimenti sarebbe ridondante anche nel modello base. È diventato ridondante con la presenza di gcc. Perché?

Probabilmente l'utilizzo del primo vincolo implicito migliora il modello base perché la scomposizione del vincolo globalCardinality non ha l'intera visione della sequenza come quella globale .

Infatti il vincolo globalCardinality può considerare tutti i valori contemporaneamente .   
Conta quante volte il valore Vj assegnato alla variabile compare nella sequenza. Il conteggio di un valore potrebbe influenzare il conteggio di altri valori, quindi il vantaggio è che più conteggi sono correlati tra loro .

Il vincolo scomposto invece pone un vincolo per ogni valore, quindi si perde la visione globale del problema. Quindi qui il primo vincolo implicito può aumentare la propagazione perché fornisce le informazioni extra già presenti nel vincolo globalCardinality .

* Sì, il tuo ragionamento è perfetto, ma puoi dire esplicitamente perché gcc ha quelle informazioni?

Il gcc ci fornisce questa informazione grazie al terzo parametro che conta le occorrenze. Infatti ogni Oj conta quante volte vj appare nella sequenza, quindi ci dà la cardinalità di Xi (dove Xi è assegnato a vj ).

* Sì, ogni Oj ci fornisce il numero di occorrenze di vj . Ma non hai ancora creato il collegamento con il vincolo implicito

Il primo vincolo implicito conta le occorrenze e controlla che siano uguali a n. Tuttavia, possiamo usare gcc per valutare la stessa espressione.   
Infatti ciò che fa gcc è vincolare che il numero di volte in cui X assume il valore vj venga conteggiato da Oj .   
Pertanto, può raccogliere i conteggi ma anche vincolare il numero di conteggi al numero di occorrenze di vj .   
Il gcc può sostituire tutte le informazioni del primo vincolo.

In generale, il vincolo gcc ci dice che la somma delle occorrenze deve essere minore o uguale a n.   
Ma allora poiché consideriamo che gli eventi sono gli Xi, stiamo anche dicendo che la sommatoria di tutti gli Xi deve essere uguale a n.

Il meta vincolo scomposto:

vincolo per tutti ( es in 0..n-1)(X[ i ] = ( somma (j in 0..n-1)(X[j]=i)));   
  
Xi è il numero di occorrenze di i.   
Controlliamo nella sequenza quante volte il numero i ricorre nella sequenza.   
Quindi contiamo quante volte Xj è uguale a i e sommiamo il tutto e otterremo il numero totale di occorrenze. E questo verrà assegnato al valore Xi

vincolo cardinalità\_globale ( X,0.. n-1,[X[ i ] | i in 0..n-1]);

% il primo vincolo implicito

vincolo n= somma (i in 0..n-1)(X[i]);