Elaborato IA

Lorenzo Agnolucci

1 Introduzione

In questo elaborato si studia un modello sviluppato con *Minizinc* per risolvere un problema di soddisfacimento di vincoli, ovvero la disposizione su una scacchiera $n \times n$ di un insieme di k oggetti che hanno la forma dei pezzi del *Tetris*. In particolare, si considerano i tempi di risoluzione ricavati empiricamente in funzione di n e k.

2 Modello

Il modello utilizza 4 diversi tipi di variabili X_i:

1. una matrice $n \times n$ di X_1 con D_1=$\{0,..,k\}$ per rappresentare una casella della scacchiera e che assume un valore intero uguale a 0 se la casella è vuota, o uguale a i se è occupata dall'i-esimo oggetto.
2. un array $1 \times k$ di X_2 con D_2=$\{1,..,n\}$ per indicare la riga di una delle 4 caselle occupate dal il blocco i-esimo.
3. un array $1 \times k$ di X_3 con D_3=$\{1,..,n\}$ per indicare la colonna di una delle 4 caselle occupate dal blocco i-esimo.
4. un array $1 \times k$ di X_4 con D_4=$\{0,1,2,3\}$ per rappresentare la rotazione dell'oggetto i-esimo rispetto alla posizione standard.

Di conseguenza, il numero totale di variabili è $n^2 + 3 * k$. I vincoli usati per risolvere il problema sono 4:

1. Vincola ciascun oggetto ad occupare 4 caselle, cosicché quelle vuote siano $n^2 - 4 * k$.
2. Impone che il numero totale di caselle occupate dai blocchi sia inferiore al numero di caselle della scacchiera.
3. Limita le possibili rotazioni delle forme in cui esse coincidono tra loro (ovvero I, O, S, Z).
4. Impone per ciascun blocco che le 4 caselle siano occupate in accordo alla forma e alla rotazione di esso.

I parametri in input sono la dimensione della scacchiera n e il numero di oggetti k, rappresentato nella forma di un array in cui k_i indica il numero di pezzi di tipo i con i=1,..7, nell'ordine I, J, L, O, S, T, Z.
Per migliorare le prestazioni e ridurre i tempi di risoluzione, il modello utilizza le *search annotations* di *Minizinc* per specificare come svolgere la ricerca finalizzata alla risoluzione del problema. In particolare, si è scelto di ricercare dando precedenza alle posizioni sulle righe e colonne della scacchiera dei vari pezzi, e in seguito considerando le possibili rotazioni. In entrambi i casi la ricerca inizia dalle variabili con il più piccolo dominio rimanente (secondo la strategia *fail first*), che vengono impostate a valori casuali nel caso delle posizioni sulla scacchiera, al valore minore possibile nel caso delle rotazioni. Inoltre viene utilizzata anche una *restart annotation* che fa sì che quando viene raggiunto un certo valore del numero dei nodi dell'albero di ricerca, la ricerca riparta dalla cima. Ciò permette di impedire alla ricerca di rimanere bloccata in un area non produttiva dell'albero. L'output del modello è una semplice rappresentazione della scacchiera, in cui gli 0 rappresentano le caselle vuote e i numeri da 1 a k i vari pezzi. Nella Table 1 viene riportato un esempio di output per n=9 e k=14.

0	0	0	0	6	6	0	14	4
0	0	2	0	0	6	14	14	4
5	0	2	11	0	6	14	4	4
5	0	2	11	11	0	9	9	12
5	5	2	11	13	9	9	12	12
0	0	0	13	13	10	10	0	12
0	0	0	13	10	10	3	3	3
8	8	0	7	7	0	0	0	3
8	8	0	7	7	1	1	1	1

Table 1: Esempio di output per n=9 e k=14

3 Risultati sperimentali

I test sono stati eseguiti su una macchina con le seguenti specifiche:

- Processore: Intel i7-4712HQ 2.30 GHz
- RAM: 16 GB
- Sistema operativo: Windows 10

Per ogni valore, il tempo di esecuzione riportato è stato calcolato facendo la media dei tempi di 5 esperimenti ripetuti, in modo che il risultato finale sia più attendibile possibile. Il solver utilizzato per risolvere il problema è

Gecode 6.1.0, nella versione built-in di *Minizinc*, configurato per lavorare con 10 thread. I tempi sono stati misurati usando le statistiche riportate in output da *Minizinc*, escludendo il tempo necessario alla compilazione.

3.1 n crescente

In questo test sono stati misurati i tempi di risoluzione in funzione di n, con k fissato a 28 (ovvero 4 pezzi per tipo). Come si può notare dalla Figure 1, il tempo di risoluzione diminuisce esponenzialmente all'aumentare della dimensione della scacchiera. Ciò è coerente col fatto che le variabili inerenti alla posizione sulla scacchiera sono assegnate scegliendo valori casuali all'interno del loro dominio, poiché all'aumentare del numero di caselle diminuisce la probabilità che due pezzi disposti in modo casuale si sovrappongano.

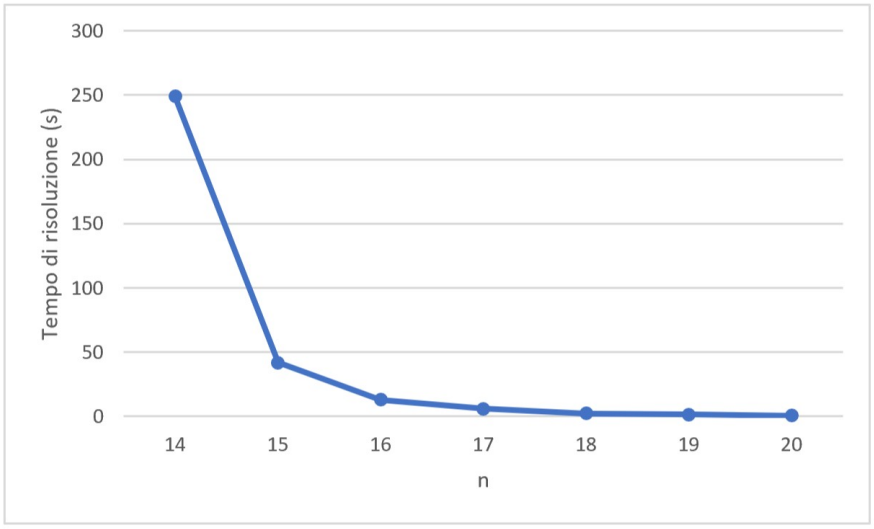

Figure 1: Tempo di risoluzione per k=28 e n crescente

3.2 k crescente

In questo esperimento sono stati calcolati i tempi di risoluzione in funzione di k, con n fissato a 15. k è stato incrementato di 1 per volta, aggiungendo un oggetto in un ordine arbitrario corrispondente a I, J, L, O, S, T, Z. Come si può osservare dalla Figure 2, il tempo di esecuzione è pressoché costante fino a k=29, e in seguito aumenta in modo esponenziale. Analogamente alle considerazioni fatte nella sezione 3.1, questo è dovuto al fatto che all'aumentare del numero di pezzi diminuisce il numero di caselle vuote all'interno della scacchiera, e quindi è più probabile che due oggetti posizionati in modo casuale siano sovrapposti.

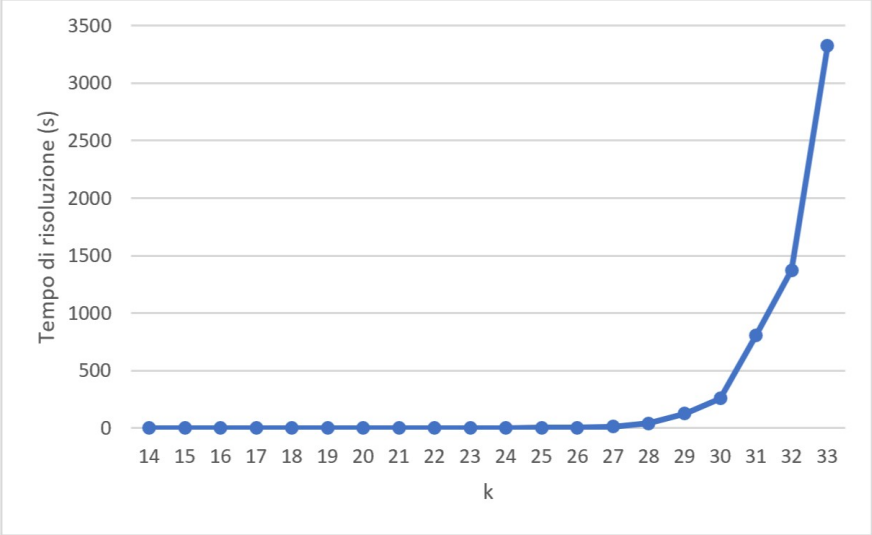

Figure 2: Tempo di risoluzione per n=15 e k crescente