Progetto di Automated Reasoning
2021-2022

Christian Londero

15 Febbraio 2022

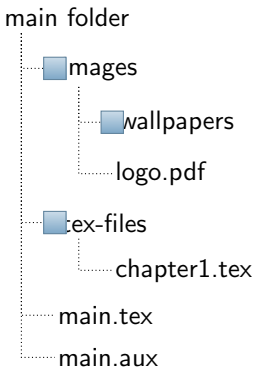
1 Definizione del problema

Si consideri una scacchiera n x n (n è dato in input). Si hanno a disposizione l pezzi a forma di L, s pezzi a forma di quadrato e r pezzi a forma di rettangolo (si veda la figura, la dimensione del rettangolo è 3 x 1, il quadrato è 2 x 2 e il lato lungo della L è 2). l, s, r sono dati in input. L'obbiettivo è di riempire la scacchiera con i pezzi a disposizione in maniera tale da minimizzare le celle vuote/free. Il requisito aggiuntivo è che f celle (date in input) siano già occupate (quindi vietate).
Si veda l'esempio (le celle grigie sono già occupate/vietate).

1.1 Considerazioni

Si presenterà un algoritmo che tenta di minimizzare le celle empty scegliendo da un sottoinsieme dei pezzi. Quindi si può dare in input un numero molto alto di pezzi al fine di "semplificare" la minimizzazione, sebbene si rischia in questo caso di fare utilizzare gli stessi tipi di pezzi quasi ovunque.
Nella sezione dei risultati vedremo infatti che se abbiamo a disposizione un elevato numero di pezzi ($3*l+4*s+3*r \gg n*n$) allora il solving è "facile", rispetto a un numero quasi giusto di pezzi ($3*l+4*s+3*r \simeq n*n$).

2 Struttura della folder



3 Soluzione

L'idea utilizzata per la realizzazione dei modelli è la stessa sia per il modello in minizinc che per quello in asp. Parlando a livello non-implementativo, è la seguente: il programma prende in input i seguenti parametri:

- `n` (intero): dimensione della board;
- `l` (intero): numero di L (2x2) disponibili;
- `s` (intero): numero di quadrati (2x2) disponibili;

- `r` (intero): numero di rettangoli (1x3) disponibili;
- `f` (intero): numero di celle vietate/forbidden/già occupate;
- `forbidden` (array): array lungo f di coordinate X, Y (sono dei fatti/facts in asp ovviamente).

La board è una matrice (array bi-dimensionale) n x n il cui dominio è definito dalla seguente enumerazione: `XXX, EEE, S11, S12, S13, S14, R11, R12, R13, R21, R22, R23, L11, L12, L13, L21, L22, L23, L31, L32, L33, L41, L42, L43`.
Al fine di rendere comprensibile l'enumerazione si osservi la figura qui di seguito.
Quindi ogni cella può assumere uno di quei valori dell'enumerazione, si è quindi proceduto a definire dei

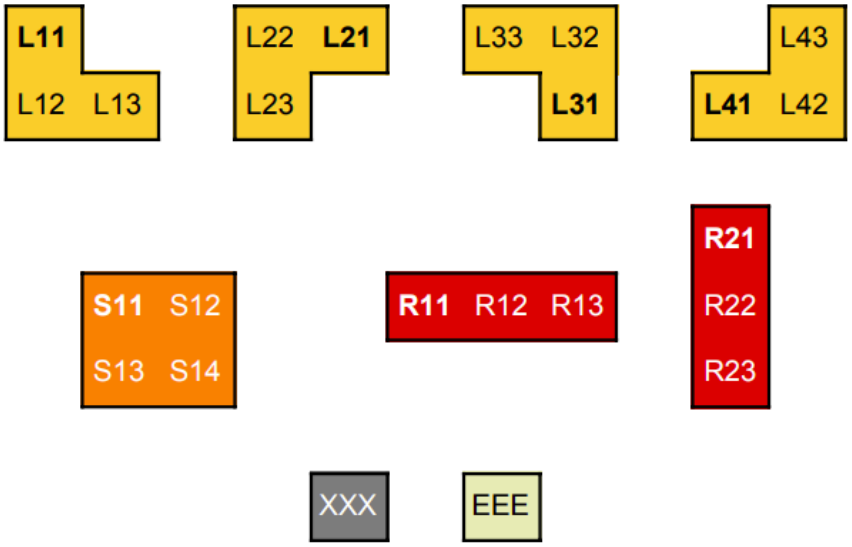


Figure 1: Tutte le possibili shapes con rotazioni

vincoli che mantengano le forme corrette.
Ho utilizzato tale "metodo" al fine di evitare sovrapposizioni indesiderate dei pezzi.

3.1 Ulteriori idee

Avevo pensato ad altre due modellazioni che però nell'implementazione poi si sono rivelate errate/inefficienti:

- indicare con 1 le L, con 2 i quadrati e con 3 i rettangoli, con 0 le empty cells e con -1 i forbidden; tuttavia, sebbene il dominio in questo casi sia notevolmente ridotto, con tale metodo non si riescono a gestire efficientemente le sovrapposizioni di forme uguali con diverse rotazioni
- enumerare ogni pezzo mantenendo il tipo salvato in un "array" lungo k con $k = l + s + r$ (per esempio, ignorando le celle empty e forbidden, se abbiamo $k = 2 + 1 + 3$ avremmo che il dominio delle celle della board è $1..k = 1..6$ e manteniamo un ulteriore array lungo k per indicare il tipo: [1,1,2,3,3,3]), in modo tale che il tipo indichi quali constraint il pezzo i-esimo debba avere e non ci sono sicuramente sovrapposizioni in quanto un pezzo di forma uguale avrebbe un $j! = i$ come "valore" e ogni "valore" può comparire al più 3 o 4 volte nella board a seconda della forma (rispettivamente se L/rettangolo o quadrato); tuttavia questo metodo si è rivelato altamente inefficiente in quanto i domini erano molto più grandi rispetto alla soluzione scelta e portavano quindi a tempi molto più elevati anche con strategie di `int_search` (per minizinc) personalizzate;

4 Risultati minizinc