MNK-GAME

Relazione

Cheikh Ibrahim · Zaid
Matricola: 0000974909

Xia · Tian Cheng
Matricola: 0000975129

Anno accademico
2020 – 2021

Corso di Algoritmi e Strutture Dati
Alma Mater Studiorum · Università di Bologna

1 Introduzione

Il progetto MNK-Game consiste nella realizzazione di un algoritmo in grado di giocare a una versione generalizzata del Tris.
La criticità maggiore risiede nella valutazione delle possibili mosse da eseguire che crescono esponenzialmente nel progredire del gioco.
Contemporaneamente, l'algoritmo deve avere la capacità di effettuare scelte qualitativamente accettabili.

2 Scelte progettuali

2.1 Interfaccia `MNKPlayer`

La funzione `selectCell` dell'interfaccia `MNKPlayer` implementa le :)

2.2 Albero di gioco

Per la valutazione e la scelta della mossa da eseguire viene implementato un albero di gioco gestito nella classe `GameTree`.
Ciascun nodo dell'albero contiene:

- La descrizione del nodo rappresentato
- Il riferimento al nodo padre e una lista concatenata contenente i figli
- Un punteggio euristico

2.3 Euristica sui punteggi

Tramite la classe `BoardStatus` è possibile rappresentare ed effettuare la stima del possibile esito di una configurazione di gioco.
Tale punteggio euristico viene calcolato tramite un algoritmo basato sulla programmazione dinamica.
Siano $M[0..n-1]$ il vettore in input contenente la configurazione di gioco di ciascuna cella rappresentata dalle costanti `PLAYER`, `OPPONENT`, `FREE` e $S[0..n-1]$ il vettore contenente l'output, ove $S[i]$ contiene una coppia di interi rappresentanti il numero di celle allineabili e il numero di mosse necessarie per vincere selezionando la i-esima cella.

$$S[0] = \begin{cases} (0,0) & \text{se } M[0] = \texttt{OPPONENT} \\ (1,0) & \text{se } M[0] = \texttt{PLAYER} \\ (1,1) & \text{se } M[0] = \texttt{FREE} \end{cases}$$

$$S[i] = \begin{cases} (0,0) & \text{se } M[i] = \texttt{OPPONENT} \\ (S[i-1].\text{aligned}+1, S[i-1].\text{moves}) & \text{se } S[i-1].\text{first} < K \ \texttt{AND} \ M[i] = \texttt{PLAYER} \\ (S[i-1].\text{aligned}+1, S[i-1].\text{moves}+1) & \text{se } S[i-1].\text{first} < K \ \texttt{AND} \ M[i] = \texttt{FREE} \\ (S[i-1].\text{aligned}, S[i-1].\text{moves}-1) & \text{se } M[i] = \texttt{PLAYER AND } M[i-K] = \texttt{FREE} \\ (S[i-1].\text{aligned}, S[i-1].\text{moves}) & \text{se } M[i] = \texttt{PLAYER AND } M[i-K] = \texttt{PLAYER} \\ (S[i-1].\text{aligned}, S[i-1].\text{moves}) & \text{se } M[i] = \texttt{FREE AND } M[i-K] = \texttt{FREE} \\ (S[i-1].\text{aligned}, S[i-1].\text{moves}+1) & \text{se } M[i] = \texttt{FREE AND } M[i-K] = \texttt{PLAYER} \end{cases}$$

2.4 Generazione parziale dell'albero

2.5 Valutazione mosse "interessanti"

3 Conclusione