Projecte - PROP

Pau Ramos
Jia Le Chen

HEXercici

Disseny de l’heurística

L'heurística implementada té com a objectiu avaluar l'estat del tauler del joc Hex per proporcionar una puntuació que guiarà les decisions estratègiques del jugador. Aquest sistema es basa en diversos factors que tenen en compte tant la posició actual de les fitxes com el progrés de la partida. L'heurística utilitza diferents pesos per prioritzar aspectes del joc segons la fase:

- **Etapa inicial**: En aquesta etapa, es dona prioritat a l'ocupació de l'àrea central, la creació de ponts i la reducció dels espais lliures de l'oponent.
- **Etapa intermèdia**: Es dóna més pes al bloqueig de moviments de l'oponent i a les amenaces dobles.
- **Etapa final**: Es prioritzen les amenaces directes i les connexions clau per assegurar la victòria.

La funció “evaluateConnections” analitza la capacitat del jugador per crear connexions entre les seves fitxes mitjançant:

    - **Ponts**: Configuracions que connecten dues fitxes pròpies a través d'espais buits.
    - **Plantilles de vora i interior**: Configuracions predeterminades que faciliten les connexions en diferents zones del tauler.
    - Les connexions alineades amb la direcció objectiu del jugador reben una puntuació addicional multiplicada per alignmentMultiplier.

La funció “evaluateMisalignedMoves” penalitza moviments que no avancen cap als costats objectiu del jugador

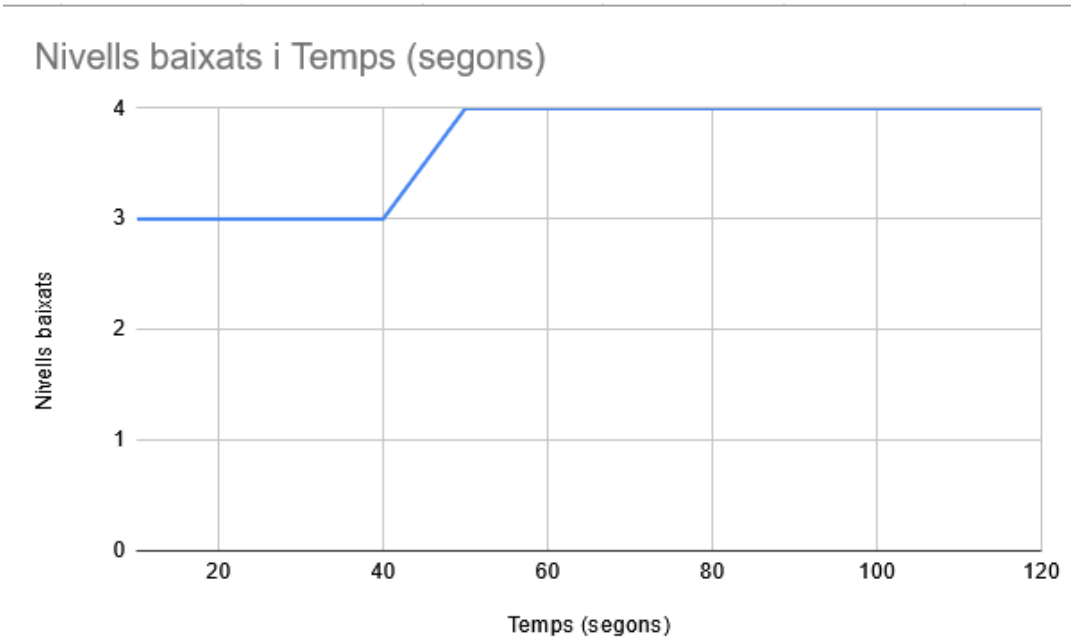
La funció “calculateDistanceToVictory” implementa l'algorisme de **Dijkstra** per calcular la distància mínima entre els costats del tauler.

Tots aquests factors es combinen en la funció “eval”, que retorna una puntuació total per al tauler: una puntuació positiva indica avantatge per al jugador mentres que una puntuació negativa suggereix que l'oponent té una posició més forta.

L'heurística està dissenyada per ser flexible i adaptativa, canviant el focus segons la fase del joc. Prioritza moviments estratègics que augmenten les connexions pròpies, bloquegen l'oponent i maximitzen la influència en zones clau. Aquesta avaluació guia el jugador cap a la victòria amb una estratègia equilibrada i ben informada.

Gràfica de número de nivells baixats segons temps

A la següent gràfica es pot observar els nivells baixats la primera tirada de la partida.



Temps (segons)	Nivells baixats	Nodes explorats
10	3	633
20	3	1349
30	3	1807
40	3	1905
50	4	2864
60	4	2823
70	4	3130
80	4	3866
90	4	4100
100	4	5200
110	4	6162
120	4	8084

Estratègies d'optimització

- ## Ordenació dels taulells segons la seva puntuació

  S'avalua cada moviment disponible al tauler utilitzant l'heurística dissenyada i s'ordenen els taulers de major a menor puntuació.

```
moveList.sort((a, b) -> {
    HexGameStatus tauler1 = new HexGameStatus(hgs:status);
    tauler1.placeStone(point: a.getPoint());
    HexGameStatus tauler2 = new HexGameStatus(hgs:status);
    tauler2.placeStone(point: b.getPoint());
    PlayerType currentplayer = status.getCurrentPlayer();
    int valor1 = heuristica.eval(board: tauler1, player: currentplayer);
    int valor2 = heuristica.eval(board: tauler2, player: currentplayer);
    return Integer.compare(x: valor1, y: valor2);
});
```

- ## Agafar els 30 millors moviments

  Un cop ordenats tots els moviments possibles, només es segueix l'algorisme amb els 30 taulers (o menys, si la llista té una longitud més curta de 30) que tenen la millor puntuació heurística, així es redueix de forma substancial els taulers a avaluar.

```
for (int i = 0; i < 30 && i < moveList.size(); ++i) {
    if (timeout) break;
    MoveNode mn = moveList.get(index: i);
    HexGameStatus newStatus = new HexGameStatus(hgs:status);
    newStatus.placeStone(point: mn.getPoint());
```

- ## Poda alfa-beta

  Per implementar la poda, es passa a la funció de minimax un paràmetre alpha i un paràmetre beta. La primera crida es fa inicialitzant alpha al valor més petit possible (Integer.MIN_VALUE) i beta al valor més elevat possible (Integer.MAX_VALUE). A cada crida de la funció, s'actualitza alpha o beta (depenent de si la fase és maximitzadora o minimitzadora) amb el valor heurístic més gran o més petit respectivament. Aleshores, si la beta és inferior a l'alfa i la poda està habilitada, trenca el bucle que genera nous nodes dins de la funció minimax.

```
if (maximitzant) alfa = Math.max(a: alfa, b: millorRes);
else beta = Math.min(a: beta, b: millorRes);

if (beta <= alfa) break;
```