Posudek bakalářské práce

Matematicko-fyzikální fakulta Univerzity Karlovy

Autor práce	Avazagha Ahmadov
Název práce	Tournament environment for the board game Hive with an implementation of sample bots
Rok odevzdání	2024
Studijní program	Informatika
Specializace	Artificial Inteligence

Autor posudku	Vladan Majerech	Vedoucí
Pracoviště	Katedra teoretické informatiky a matematické logiky	

K celé práci

	lepší	OK	horší	nevyhovuje
Obtížnost zadání	X			
Splnění zadání	X	X		
Rozsah práce *... textová i implementační část, zohlednění náročnosti*	X	X		

The game environment was implemented, sample bots were created, test turnament played, but the MCTS bot has too slow implementation to be usable.

Textová část práce

	lepší	OK	horší	nevyhovuje
Formální úprava *... jazyková úroveň, typografická úroveň, citace*	X			
Struktura textu *... kontext, cíle, analýza, návrh, vyhodnocení, úroveň detailu*	X			
Analýza		X		
Vývojová dokumentace		X		
Uživatelská dokumentace	X			

I was a bit confused by the one hive rule. Definition of the connectivity is incorrect, and the Figure 1.5 would be more illustrative if the west white ant would be one step south. It looks like only moving articulations is forbidden, but start tile and end tile of a slide should have common neighboring nonempty tale (where tales in one stack are neighboring as well).

It would be nice to have 3 times repetition rule similarly as in chess, repeating the position 40 times to get draw is boring. Similarly it would be nice to have some time control (either limits per move or per entire game or a combination ... time controll formats on arimaa.com are user friendly, I would probably allowed independent time controlls for the two players).

Standard move encoding is not compatible with the search purposes (positions when two ants/beatles/...of one player exchange their position are considered different in the encoding while they are equally good considering the winning chances). This should be addressed for the case efficient transposition tables storing principal moves are used. It would be nice to mention how the engine solves the problem with failing "pointy-top direction"assumption.

The alpha-beta bot plays rather well, but I am surprised by the bad results of the expert player. May be I am too lucky beginner. Depth 1 search (evaluation testing) reveals incentive to introduce beatles in early stages in vicinity of its own queen. It seems to me they are much more effective to introduce them latter in vicinity of the opponents queen. Considering aplha-beta pruning efficiency, starting search with the best candidate move is very helpful. I think it is worth to make full one level shorter alpha-beta search to get the approximate principal variants, so the iterative deepening storing principal variants in the transposition table speeds up the alpha-beta search. It is challenging for alpha-beta bots to cope with time control problems.

MCTS with UCT works well with huge number of playouts. It cannot play well when the number of playouts is small constant factor higher than the branching factor. The incentive is to play the mostly played/winning child, but UCT forces at least one playout for children of an expanded node, so the bigger branching factor the expanded node have, the bigger starting number of playouts is. The UCT favors less played nodes to be selected for later playouts (considering winning rate as well), but in the scenario of very limited number of playouts, there is no time to equalize the starting discrepancy so the winning rate affects the result only marginally. There is PUCT search variant which does not force playout for each child. It would work with equal probabilities of children, but it allows starting preferences of the moves according an evaluation function. PUCT does not suffer from different branching factor discrepancies in the initial number of playouts and therefore the wining rate affects the number of playous even when small number of playouts was played. For MCTS bots the time controll should not cause too much problems, it just determines the time to spent on current move (could vary depending on the game progress) and sets an interruption time acordingly, then the infinite MCTS loop is started and when interrupted the best root child is returned.

The main problem anyways is, why MCTS is unable to play thousands of playouts during few seconds. The random move selection and execution must be much faster.

Implementační část práce	lepší	OK	horší	nevyhovuje
Kvalita návrhu *... architektura, struktury a algoritmy, použité technologie*		X	X	
Kvalita zpracování *... jmenné konvence, formátování, komentáře, testování*		X	X	
Stabilita implementace	X	X		

I did not get the StopWatch use in Simulate. Student chosen a game independent implementation where moves to be considered are obtained from the game engine rather to be calculated internally. The interface requires the engine to generate all the moves and the agent selects a move index and chooses the move. The interface could divide the process to two calls, in first the engine returns the number of possible moves, in second the engine returns the move in the given index (in range). Or the engine could return interval of moves at the time denoting the interval start, such that the move index belongs to the interval. Such interface allows engine to optimize the move generation as counting number of possible moves of a movable piece is much easier then generating all the connected data.

Used Zobrist hashing considers the positions with switched pieces of the same type and player to be different except improbable cases, it is prepared for the case even non beatle pieces would be on levels upto 5. If added piece position has several neighbors, the engine includes the move among valid moves for each neighbor it has (and it standardizes by using first unused piece among the piecetype). It is different according the notation, but the game logic does not change so the branching factor is higher than it must be. Hashing positions and processing them just once would not affect the game logic. For each such possible move engine checks if the Queen rule does not disqualify it (even when the check is not coordinate dependent). Why are not the loops nested the other way to do the test just once? When a property like who is the active player can have just two values, switching the property means always xorring the Zobrist hash by the xor of both values. There is no reason to do it step by step in property dependent order.

Fortunately when considering moves of already introduced pieces, only one notation of the final coordinate is used so the branching factor is not affected there. IsNotIsolated and CanSlideIn check occupation of two positions, one fails if none is occupied, other when both are occupied. Both functions are called in the move validity check. It would be faster to reduce the check to "exactly one of the positions is occupied". But let us not stop there. During the move generations the same pair is checked repeatedly. Caching the results (making the graph) would reduce the computation costs. Similarly for ant movements we can compute the connected sliding components first and just check to which components the ant can step (it will help in calculating the number of moves, when we should list them all, we must traverse it anyways). IsBreakingHive does not test the piece is the only piece at an articulation point, but that after removing the piece the graph is disconnected. So when the graph is disconnected moving pieces is not allowed and pieces could be only introduced. In either case running dfs for each piece is more expensive then finding articulations once and just checking if the position is articulation (and the piece to move is the only piece there).

The engine allows adding piece to coordinate 0,0 even when it is not the first turn (notation without the piece describing the destination position) and the Human interface allows it. This allows creating disconnected graph in the situation the pieces moved away. It definitely is not ok. Opponent then can only place pieces and cannot connect the graph to restore the mobility. Only one player can extend the component containing 0,0 coordinate and eventually connect the graph by added pieces to restore the mobility. I do not know how to pass using human interface when there are no moves available.

I have managed to run out of the zobrist coordinates causing application to close, the move had to be done on the edge on human interface region (queen with grasshopper of both colors can march rather quickly, bottom left corner has no interfering buttons).

In the thesis is written that MCTS returns root child with maximal total winScore, but in the implementation the move which will be UCT selected for the next iteration is returned (so actually less branching child is prefered in the case of small number of playouts, but generally we do not want the small number of playouts to be bonus in the answer).

I do not like that the game engine determines the color or type of a piece by a "nameswitch". I bet bit arithmetics must be much faster for such purposes. Similarly switching is used on places where I would use an array indexing. I am not sure in which cases the compiler could optimize it.

Celkové hodnocení	Very Good
Práci navrhuji na zvláštní ocenění	No

Datum Podpis