A Region-divided Search Algorithms for Game Amazons

Xu Hongxia[1], Xing Yuan[2], Zuo Guoyu[1]

1. College of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124
E-mail: xhxccl@bjut.edu.cn

2. College of Computer Science, Beijing University of Technology, Beijing 100124, China
E-mail: 490441089@qq.com

Abstract: A kind of alpha-beta search algorithm based on region-divided checkerboard for the game of Amazons was used to improve the efficiency. Hierarchy searching is adopted in this approach. Searching is firstly carried in the divided regions, then the searching result of the whole checkerboard is gotten according to those valued of various regions. The evaluation and integration of divided regions are also adjusted and optimized. In addition, the definition and recognition of divided regions are provided. Furthermore, the feasibility and complexity of the search algorithm are analyzed. Finally the efficiency of the algorithm is proved by experiments data. .

Key Words: Game of Amazons, divided-regions, alpha-beta searching algorithm

1 Introduction

The game of Amazons is invented by an Argentinean who is called Walter Zamkauskas in 1988 [1]. It combines some features of games of chess and Go. Many researchers on computer games are focus on it for its simple rules and plentiful strategies. The game of Amazons has large searching space and each step even has more than 2000 moves. So it is very important for the computer game of Amazons to have an effective and practical searching algorithm.

At present, Alpha-beta searching is a popular algorithm that is applied in Amazons computer game [2]. But each step of this game has huge number of available moves, so the searching layers can't be expanded deeply. There would be millions of leaves node for a two layers searching tree. So alpha-beta searching algorithm needs to be improved by using pruning some redundant branches. The checkerboard of Amazons games is obviously regional, especially at the later of on-going game. Many squares in the board are set by barriers, which fragmented the whole board and several 8-connected regions are formed. These regions are independent, that is to say, pieces playing in one region won't influence that in other regions [3]. However, many searching algorithms don't consider these characteristics of independent regions currently, the generations of next move not only consider the moves of opponent in local region, but also take into account those in other regions, so the searching efficiency is decreased drastically.

An alpha-beta searching algorithm based on divided regions is proposed in this paper, according to those features mentioned above. This algorithm firstly searches at each separated regions, then on the basis of these searching results, the outcome of the whole board can be gotten and the next move can be generated. When playing pieces next time, the program only needs to take into account those sub-boards that have changes between last two steps, and won't consider those ones that have no moves. So this hierarchical searching algorithm can cut many redundant branches of the searching tree, moreover, lots of searching time can be saved for only searching those changed sub-boards. In addition, this searching algorithm based on divided-regions needs to improve its evaluation functions on basis of the situation of checkerboard in order to get more ideal searching outcome.

2 Definition and Dipartition of regions for Amazons board

According to the playing rules of Amazons game, the board is easily formed several 8-connected regions illustrated as figure 1 as a result of placing barriers. In this paper, these regions are also called sub-board.

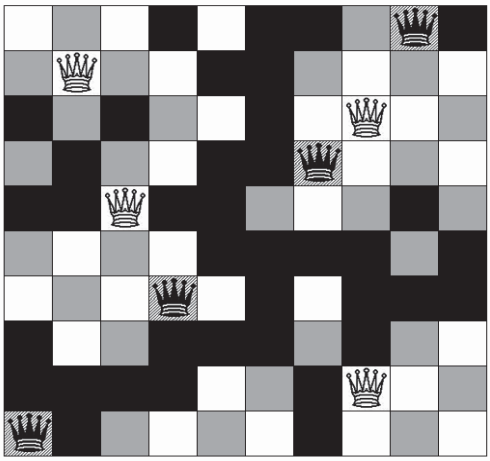

Figure 1：a region-divided checkerboard

2.1 Dipartition of regions

Sub-boards are some 8-connected regions essentially. The method of dividing regions can utilize many classical approaches in computer graphics, such as seed filling, edge filling, i.e. Sub-board has no difference with the whole board, except for the number of pieces.

So those algorithms applied in the whole board can also be applied in sub-boards.

2.2 Classification of regions

Although some independent regions can be divided based on computer graphical algorithms, the searching algorithm to decide next new move is connect with the layout of pieces on checkerboard. So in this paper, regions are classified into three types: terminating sub-board, competing sub-board and filling sub-board. The last two types are also called executing sub-board.

- Terminating sub-board: that the pieces can't move in any way。
- Competing sub-board: that the pieces involved both players can go on moving。
- Filling sub-board: that the pieces involved only one player can go on moving。

2.3 Influence of moves by regions classification

The whole board can be divided into several types of sub-boards after dipartition. How to choose the sub-board that should move the pieces first, after searching all sub-boards. Firstly, terminating sub-boards can be ignored because of no moving. As to competing and filling sub-boards, most of people prefer to choose competing sub-boards, especially when the pieces of own are scattered in the both types of sub-boards. Because players can make situation is favorable towards their own by moving pieces in competing sub-board, while moving pieces in filling sub-board only can reduce the territories of own, so can deal with it later. However, moves in competing sub-board are not always benefic. Such as the competing su-board illustrated as Figure 2, if the White first moves, then it will lose half territory. But if the Black first moves, the White can keep all territories. If at that moment, the White has another filling sub-board, it will certainly move pieces in filling-board firstly. Consequently, if filling sub-boards and competing sub-boards exist at the same time, it is not always a correct criterion to move first in competing sub-board, the contrast and analysis should be done to these two types of sub-boards.

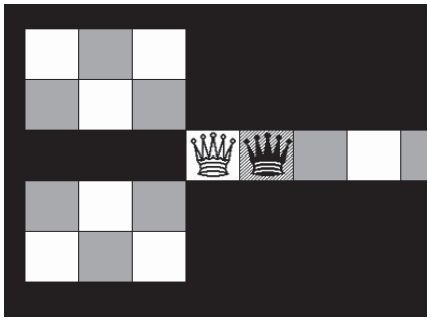

Figure2： Competing sub-board

3 Alpha-beta search based on divided regions

3.1 Executing condition of search

In fact, search based on divided regions is a method breaking complex problem into several sub-problems first and solving then. Some problems that satisfied following three conditions could use this method.

- The problem can be discomposed into several simpler similar sub-problems
- These sub-problems are independent
- The problem can be solved based on solutions to those discomposed sub-problems

Search of Amazon games satisfies these conditions. By the depiction of section 2.1, we can see that the whole board B of Amazon game can be divided into several 8-connect regions b_1, b_2, b_n according to those squares occupied by barriers. These sub-boards have no essential difference with the whole board except for the numbers of pieces. So they are belongs to the same type of problems. The searching processes of various sub-boards are independent. So weather the move generation or the evaluation function definition, they both have no intersection each other. When combining the various searching results to one whole searching result, the definition of evaluation function should be considered. At present, the heuristic values of evaluation function usually include two parts: the area of territories, the mobility of active pieces [4]. As the whole board is divided into many sub-boards, the territory area of one player can be got by sum up all that values of each sub-board. The mobility of pieces is the sum of that of four Amazons. When pieces are scattered into different sub-boards, the mobility of each sub-board can be calculated respectively. Then the mobility of whole board can be got by accumulate these outcomes. From above description, we can draw a conclusion that it is feasible to get the searching result of whole board according to those searching results of each sub-board.

It is a difficulty for machine game to evaluate chess game, and so does the search of region division. When some player was determined to gain victory, the general evaluation function will not return a heuristic value but an infinite value. Therefore, the evaluation function should be modified for the search of region division so as to return the heuristic value after victory or defeat of a sub-board. The victory or defeat of a sub-board does not represent one of the whole chessboard. If the search value of the sub-board is an infinity, it will affect the search value of the whole chessboard. Furthermore, the evaluation value is generally zero for the terminating sub-board because their pieces have been blocked. Both their territory and mobility are zero. Thus, the effect of the search value of competing sub-boards and filling sub-boards on the search value of the whole chessboard will be discussed while the terminating sub-boards ignored.

3.2 Min-max Search of region division

Firstly, it will be proved after the region division that the min-max search of each sub chessboard could deduced that of the whole chessboard.

Let B be the whole chessboard and bi the sub chessboard. S(B) represents the min-max search result of chessboard B, E(B) is the value of evaluation function on B. When the search layer number is 0 or the victory or defeat is determined, S(B)=E(B).

If the search of region division will be applied, on the one hand, the search value of the original chessboard needs to be determined according to the search value of each sub chessboard; on another hand, the optimum moving method of the original chessboard will be selected from all that of each sub chessboard. If there are many moving methods in chessboard B, let Bi be the chessboard after moving method i. Also Bi can be understood as the sub-node of searching tree B. When the number of moving method of B is not 0, $S(B)=\max S(B_i)$, otherwise $S(B)=E(B)$. Let the optimum moving method of B be $B_{\max}$, then $B_{\max}=\arg\max(S(B_i))$, $S(B)=S(B_{\max})$.

Let the whole chessboard B have executing sub-boards of $b_1, b_2, ..., b_n$. How to select $B_{\max}$ from optimum moving methods of $b_{1\max}, b_{2\max}, ..., b_{n\max}$ will be considered. The maximum in $b_{i\max}$ can not be directly used as $B_{\max}$, because the evaluation values of the predominated sub chessboards are often higher than those of the inferior sub chessboards. Therefore, the change of evaluation value by chess move or non-move of each sub chessboard will be investigated. The evaluation value of chess non-move should be also obtained. The so called chess non-move means the selection of abandonment during its own side move and the whole chessboard put to the opposite side without any move. The chess non-move also can be understood as an empty moving by its own side[5]. Let the chess non-move of the sub chessboard b_i be b_{i_N}. Here, $S(b_i)\neq S(b_{i_N})$. Although the chessboards of both sides are same, their sequences of moving pieces are different. Therefore, their search results are different.

Now the search value of the optimum move $S(b_{i\max})$ and the search value of non-move $S(b_{i_N})$ of the sub chessboard have been obtained. When selecting the optimum chess move, the maximum of $S(b_{i\max})-S(b_{i_N})$ in each sub chessboard can be used.

$S(b_{i\max})-S(b_{i_N})$ means the change for the status by selecting sub-board i for chess move. Certainly the sub-board making status better would be selected.

It should be considered how to use the search value of the sub chessboard for representing that of the whole chessboard $S(B)$. When B has been decided by victory or defeat or the search layer number is 0, B is not able to produce any move method. It means $S(B)=E(B)$. Similarly, the sub chessboard is not able to generate any move method and $S(b_i)=E(b_i)$. The evaluation value of the whole chessboard $E(B)$ can be the summation of all evaluation values of sub chessboards. So, $E(B)=\sum E(b_i)$. As mentioned above,

$$S(B)=\sum S(b_i) \tag{1}$$

When the search layer number is not 0 and B has not decided by victory or defeat, the search value of the whole chessboard equals to the maximum one of all move methods, that is, $S(B)=\max S(B_i)$. As to one moving method B_i of the whole chessboard, it actually only occurred in one of sub-boards while the others didn't move. If B_i appeared in the sub chessboard j and its move method was b_{j_k}, it could be taken that the move method of b_{j_k} was selected in the sub chessboard j and the non-move method of b_{i_N} in the others. Therefore, $S(B_i)$ can be the summation of the search values of move methods $S(b_{j_k})$ and non-move methods $S(b_{i_N})$, that is,

$$S(B_i)=S(b_{j_k})+\sum_{i=1}^{n}S(b_{i_N})-S(b_{j_N}) \tag{2}$$

Therefore,

$$S(B)=\max_{j,k}\{S(b_{j_k})+\sum_{i=1}^{n}S(b_{i_N})-S(b_{j_N})\} \tag{3}$$

Because $\sum_{i=1}^{n}S(b_{i_N})$ does not change with j、k，equation (3) can be transformed into

$$S(B)=\max_{j,k}\{S(b_{j_k})-S(b_{j_N})\}+\sum_{i=1}^{n}S(b_{i_N}) \tag{4}$$

$\max_{j,k}\{S(b_{j_k})-S(b_{j_N})\}$ can be represented by

$$\begin{aligned}\max\{&S(b_{1_1})-S(b_{1_N}),S(b_{1_2})-S(b_{1_N}),...,S(b_{1_{k_1}})-S(b_{1_N})\\ &S(b_{2_1})-S(b_{2_N}),S(b_{2_2})-S(b_{2_N}),...,S(b_{2_{k_2}})-S(b_{2_N})\\ &\cdots\cdots\\ &S(b_{n_1})-S(b_{n_N}),S(b_{n_2})-S(b_{n_N}),...,S(b_{n_{kn}})-S(b_{n_N})\}\end{aligned}$$

While ki means the number of moves of the sub

chessboard i. Merging each transverse, it can be obtained:

$$\begin{aligned}&\max\{\max_k\{S(b1_k)-S(b1_N)\}\\&\quad\max_k\{S(b2_k)-S(b2_N)\}\\&\quad\cdots\cdots\\&\quad\max_k\{S(bn_k)-S(bn_N)\}\}\\&=\max_j\{\max_k\{S(bj_k)-S(bj_N)\}\}\\&=\max_j\{\max_k S(bj_k)-S(bj_N)\}\end{aligned}$$

Because $\max S(bj_k)=S(bj)$, then

$$\max_{j,k}\{S(bj_k)-S(bj_N)\}=\max_j\{S(bj)-S(bj_N)\}$$

The meaning of the above transformation is that the optimum move method can be only selected from that of each sub chessboard without consideration of other move methods of the sub chessboard. Integrating above all equations, thus

$$S(B)=\max_j\{S(bj)-S(bj_N)\}+\sum_{i=1}^{n}S(bi_N) \quad (5)$$

Till then, the expression method was obtained for the search value of the whole chessboard. When B has been decided by victory or defeat or the search layer number is 0, the search value of the whole chessboard is the summation of those of all sub chessboards on basis of equation (1). When not in the above case, according to equation (5), the search value of the whole chessboard is the addition of the maximum difference between the search value of each sub chessboard and that of non-move and the summation of all search values of non-move of sub chessboards. This conclusion is consistent with the selection of the optimum method of the original chessboard. When selecting the maximum of $S(bj)$ - $S(bj_N)$, it actually selected the best move method from all sub chessboards.

3.3 Seeking method for search value of sub-board

The seeking method for search values of sub chessboard $S(bi)$ and non-move $S(bi_N)$ will be discussed.

The only one difference between the competing sub -board and the original chessboard is that the chess number is not always 8. Generally, the search function of the original chessboard can be directly used for searching $S(bi)$ with some modification for the chess number. It is also easy for the seeking method of the search value of non-move $S(bi_N)$. After exchanging the successive moving order of chessboard b, the result could be gotten by solving the search function.

It would be complicated for the filling sub-board. Because the filling sub-board only has one player's pieces, its previous evaluation function would return an infinite value. It must be modified so as to return a heuristic value. However, there still exist some problems although local evaluation is not infinite. When the generation method is proceeding for the opposite side, the number of the generation method is 0 and therefore the search tree only search one layer so as to get bad filling methods. Before using divide regions, the search function may search that after the first step because there are active chess for the opposite site in other sub chessboards. In order to search more deeply for the filling sub-board, the search function should be modified and then transformed into the filling algorithm. An easy filling algorithm based on min-max search is to replace the search function at minimum layer with that at maximum layer. Thus the min-max search is turned into a search tree to search the maximum leaf node. If the structure of the search tree is not going to be changed, the generation of moving method could be changed only to produce empty move during the generation of the opposite site. Namely, the opposite side actually didn't move and gave the chess game without any change to its own side. The modified search function can easily solve $S(bi)$ and $S(bi_N)$, and search to any layer. It can be proved that the search result by divide regions is completely the same as the original one after applying this filling algorithm on the filling sub-board.

The given above filling algorithm modified by min-max search actually belongs to a weaker filling algorithm. Currently stronger Amazon games apply a special filling algorithm to give a better filling strategy. If the filling sub-board uses other algorithms, the evaluation method generally differ too much from other sub chessboards to use search values of all sub chessboards for representing the search value of the whole chessboard. If the search based on divide regions is going to be applied, the evaluation value of the sub chessboard by filling algorithm must be unified with other sub chessboards. The transformation function from the evaluation value by filling algorithm to that by the min-max method can be fitted by mathematical statistics method. It could also be established heuristically by the algorithm types of two evaluation functions.

3.4 Alpha-beta search of region division

The search of region division will face new problems after the application of alpha-beta pruning algorithm. The alpha-beta search different from the min-max search transfers two values, that is, alpha and beta. When a root original chessboard is divided into several sub chessboards, the alpha and beta of the original chessboard should be decomposed to each sub chessboard and transferred as alpha-beta search parameters if the alpha-beta algorithm by region division is used to search each sub chessboard. However, this

decomposition method doesn't exist. The alpha, beta value of the original chessboard cannot be used to determine the alpha, beta value of the sub chessboard. Here only several new made sub chessboards are viewed as root nodes of which alpha and beta value are set as negative infinity and positive infinity. That is to say the alpha, beta value by the search before the previous node does not affect the pruning algorithm of the node. The transfer of the alpha, beta does not terminate. The search value of the original chessboard is obtained after the end of the search of each sub chessboard. Now whether the alpha or beta needs to be updated or not is selected by the search value. Then alpha and beta are continuously transferred to the following search.

The pseudo code of the alpha-beta search algorithm for the region division is given below.

```
double DivideConquerAlphaBeta(Board B,int depth, double alpha, double beta)
{
/* Chessboard B is decomposed into n executing sub-boards b[n]. The search value of B is value, the search value of b[i] is value[i], the search value of non-move is nulvalue[i]; */
if(n==1){     //B could not be decomposed and should be alpha-beta searched normally
if(depth==0){
return Evaluate(B);
}
GenerateLegalMoves();
while (MovesLeft()){
MakeNextMove();
val=-DivideConquerAlphaBeta(B,depth-1,-beta,-alpha);
UnmakeMove();
if(val>=beta){
return beta;
}
if(val>alpha){
alpha=val;
}
}
return alpha;
}else{     //B could be decomposed and should be alpha-beta searched by divide regions.
 if(depth==0){
value= Σ Evaluation(b[i]);
}else{
for(each sub chessboard b[i]){
value[i]=DivideConquerAlphaBeta(b[i],depth,negative infinity, positive infinity);
b[i] exchange of successive moving order
nullvalue[i]=-DivideConquerAlphaBeta(b[i],depth-1,negative infinity, positive infinity);
b[i] exchange of successive hands;
}
value=max(movevalue[i]-nullmovevalue[i]+ Σ nullmovevalue[j]);
}
if(value>=beta){
return value;
}
if(value>alpha){
alpha=value
}
return alpha;
}
}
```

3.5 Scale of alpha-beta search of region division

Let each step of the amazon original chessboard has N moves and totally search l layers. The problem scale under the condition of the best alpha-beta search is $2N^{l/2}$. Only with the consideration of the executing sub-board, let each step of the sub chessboard i has n_i moves, $N=\sum n_i$. Because the amazon game appears one chessboard decomposition every about 10 steps, the continuous decomposition of the sub chessboard will not be considered during the search. The problem scale of the search of the sub chessboard i is $2n_i^{l/2}$, that of non-move is $2n_i^{(l-1)/2}$. The summation of the problem scales of all sub chessboards is $\sum(2n_i^{l/2}+2n_i^{(l-1)/2})$. Each step of alpha-beta search based on region division only needs to search the sub chessboard that change between this step and before. If there are totally k sub chessboards, the probability of the move simultaneously by its own site and the opposite site is 1/k and that of the move by the selection of different sub chessboards is (k-1)/k, then the average amount of the changed sub chessboards is (2k-1)/k.

Diving the summation of the problem scales of the sub chessboards by the amount of sub chessboards and then multiplying by the amount of the changed sub chessboards, the problem scale of alpha-beta search based on divide region is obtained.

$$\sum(2n_i^{l/2}+2n_i^{(l-1)/2})\times\frac{2k-1}{k^2} \tag{6}$$

It can be testified that the search scale of alpha-beat based on divide region is smaller than that of the original alpha-beta search with $(2k-1)/k^2$ when k>=2. Furthermore, more k, n_i tends to be more averaged and the search scale is going to continuously decrease.

4 Experiment design and result

In order to test the efficiency rise of the original search by alpha-beta search for region division, the following experiment was designed. Firstly, let two AI of the original alpha-beta search algorithm play 200 games. The chessboard and AI thinking time are recorded in each step. Each recorded chessboard is searched by AI of alpha-beta search algorithm based on divide region and the search time is recorded. The time for searching chessboards by original search and alpha-beta search based on divide region are compared. It will be tested whether the searching efficiency is enhanced by divide region method. The search layers of the original search and search by divide region method are set to be the same. In order to make the searching time of each chessboard a little even, the search layers are set to be layer 3 of steps before 24, layer 4 from step 25 to step 36, layer 6 from step 37 to step 56, layer 7 of steps after step 56. During AI game, the parameters of the evaluation function should be adjusted fine after each game so as to decrease the occurrences of the same chess game.

The experimental data is shown as follows:

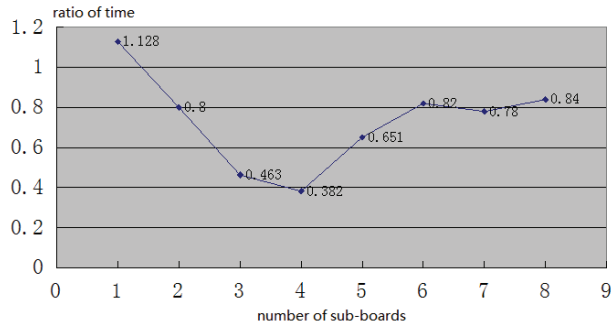

Figure 3: Experiments

The abscissae axis means the number of the sub chessboard, the ordinate axis means the ratio of the average time for alpha-beta search of region division to that of the original alpha-beta search.

It can be seen from Figure 3 that the time for divide region search is longer than the original search when the number of sub chessboards is 1. This is because the region division method doesn't take effect and such operation as chessboard decomposition detains some time. However, the time ratio is in the range acceptable. When the number of sub chessboards is larger than 1, the time for the alpha-beta search of the region division is shorter than that of the original search. The efficiency is risen the most when the number of sub chessboards is 4. The region searching efficiency decrease with increase of the number of sub chessboards because the game is normally in the late situation with many more chessboards. At this moment, the ratio of the terminating sub-board is bigger and the region search improves alpha-beta search little.

5 Conclusion

The alpha-beta search algorithm of the region division by application of characters of amazon chessboard was used to search each sub chessboard and avoided the search for the move of the cross chessboard. With application of additivity of the evaluation function, the search result of the sub chessboard was used to represent that of the whole chessboard so as to decrease searching time effectively with proper results. The experiments results showed that the region division method improved considerably the search efficiency of the alpha-beta algorithm and was beneficial to the improvement of amazon game skill.

References

[1] E.Berlekamp, J.Conway, R.Guy. Winning Ways for Your Mathematical Plays. Academic Press, London, 1982.

[2] Jonathan Schaeffer, Aske Plaat. New Advances in Alpha-Beta Searching. ACM Computer Science Conference, 124-130, 1996.

[3] Anany V. Levitin. Introduction to the Design and Analysis of Algorithms. Addison Wesley, 2002.

[4] Jens Lieberum. An Evaluation Function for the Game of Amazons. Theoretical Computer Science 349, 230-244, Elsevier, 2005.

[5] Omid David Tabibi, Nathan S. Netanyahu. Verified Null-Move Pruning. ICCA J. Vol.22, No.3, 123-132, 1999.

[6] Raymond Georg Snatzke. Exhaustive Search in the Game Amazons. More Games of No Chance, 261-278, 1996.

[7] Xu Xinhe, Deng Zhili, Wang jiao. Challenging issues facing computer game research. CAAI Transactions on Intelligent Systems, Vol(3), No.4, 288-293, 2008