Enhancing Search Efficiency by Using Move Categorization Based on Game Progress in Amazons

Yoshinori Higashiuchi[1] and Reijer Grimbergen[2]

[1] Department of Computer Science, Saga University, Saga, Japan.
hi_yoshi@fu.is.saga-u.ac.jp
[2] Department of Informatics, Yamagata University, Yonezawa, Japan.
grim@yz.yamagata-u.ac.jp

Abstract. Amazons is a two-player perfect information game with a high branching factor, particularly in the opening. Therefore, improving the efficiency of the search is important for improving the playing strength of an Amazons program. In this paper we propose a new method for improving search in Amazons by using move categories to order moves. The move order is decided by the likelihood of the move actually being selected as the best move. Furthermore, it will be shown that the likelihood of move selection strongly depends upon the stage of the game. Therefore, our method is further refined by adjusting the likelihood of moves according to the progress of the game. Self-play experiments show that using move categories significantly improves the strength of an Amazons program and that combining move categories with game progress is better than using only move categories.

Keywords: Selective search, move categorization, game progress, Amazons.

1 Introduction

Amazons is a two-player perfect information game with very simple rules [1]. From a computational point of view, its main feature is the large number of legal moves, particularly in the early stages of the game (in the initial position there are 2,176 possible moves). Even though the number of legal moves decreases as the game progresses, the average number of moves in an Amazons' position (479) [2] is considerably larger than chess (35), shogi (80) or Go (250) [3]. Because of the large average number of moves, the well-known search techniques that have been so successful in other games cannot be easily applied to Amazons. Therefore, Amazons has attracted some attention recently as a topic of research. These efforts have focused on building an evaluation function that can evaluate positions accurately without deep search [4, 5] and on how to do selective search in Amazons [6]. In this paper, we will present a new search method for Amazons which is partly a method for selective search but mainly aims at improving search efficiency.

Amazons is a relatively new game, so there are no known strategies on how to play the opening and there is no expert feedback available to decide which moves are good in the middle game. In other game programs (e.g. chess, Go, or shogi), years of experience have led to heuristics for moves that are likely to lead to an advantageous position. These moves can then be given priority during the search. In chess, for example, moves that capture material or moves that cover the center are searched early, while sacrifices are searched last. By using these heuristics, the efficiency of α-β search can be improved, increasing search speed.

The research presented in this paper presents two methods to improve the search efficiency of an Amazons program. First, we will propose a set of move categories and use a calculation method from Realization Probability Search [7] to order moves based on these move categories. Second, we will show that the importance of move categories changes as the game develops. Therefore, when deciding the move ordering it is important to take the progress of the game into account.

In Sect. 2 we will start with an explanation of the properties of Amazons and how these properties lead to heuristics for good moves. These heuristics will be used to group moves into different categories. In Sect. 3 each category is assigned a priority based on statistical data from game records. In Sect. 4 a simple method to measure progress in Amazons is presented and the relation between game progress and the move categories is given. In Sect. 5 the results of a number of experiments comparing the performance of programs without move categories, with move categories and with move categories based on game progress will be presented. Finally, in Sect. 6 we will give conclusions and suggestions for future work.

2 Move Categories in Amazons

Because of the large number of possible moves in Amazons, under tournament conditions it is impossible to do a deep full-width search (the full-width search version of our program can only search to a depth of 2 or 3 ply in the opening to 6 or 7 ply near the end of the game). Therefore, using selective search is the only way to do a reasonable look-ahead. The most common domain-independent methods for selective search, like the null-move heuristic [8], ProbCut [9] and Multi-ProbCut [10] use a shallow search to estimate the result of deep searches. There are two reasons why these methods face problems in Amazons. One reason is that there is no deep search in Amazons until the endgame. Predicting the results of shallow searches by even shallower searches is risky. The second problem is that Amazons programs suffer from the even-odd iteration instability. There are no known methods to do quiescence search in Amazons, so there can be important changes in the evaluation function value after playing a move. A lot of effort into building an evaluation function for Amazons goes into minimizing this effect, but there are still significant differences between the evaluation of even and odd iterations, especially in the opening. This makes it hard to predict the result of a d ply search with a $d-1$ ply search. Also, in the case of shallow searches, the differences between a d ply search and a $d-2$ ply search are usually too large to be useful for a prediction.

Consequently, domain-dependent methods for selective search are needed in Amazons. One method, proposed by Avetisyan and Lorentz [6], is to use the evaluation function to evaluate each move after it has been played and discard moves for which the evaluation is below a certain threshold. This method was refined by making a difference between evaluation after the Amazon move and shooting the arrow.

Rather than eliminating a certain number of possible moves from the search, we will propose a method to use knowledge about good Amazons moves to improve the efficiency of α-β search. A common method for improving search efficiency is to use information that has become available during search. The best move of the previous iteration, killer moves and the history heuristic are examples of such methods. By trying these moves first, the probability of a cut is increased, and search speed is improved. However, when a new position is encountered, this information is unavailable but searching good moves first will still improve search speed. In this case heuristic, game-specific information is needed to suggest which moves to search first.

In Amazons, with its short history, heuristics for selecting good moves are unknown. However, as pointed out by Lieberum [5] and confirmed by our own experience in playing and programming Amazons, confining one or more opponent Amazons to a small space is an important strategic theme. From this, a number of straightforward ways to limit the opponent Amazons' moving ability and improving the moving ability of one's own Amazons come to mind. We have categorized these heuristics into 10 basic categories that are given in Table 1.

Table 1. Basic move categories in Amazons.

No.	Category	Type	#
1	Move blocks opponent Amazons	0, 1, 2, 3 or more	4
2	Arrow blocks opponent Amazons	0, 1, 2, 3 or more	4
3	Move adjacent to opponent Amazon	true, false	2
4	Arrow adjacent to opponent Amazon	true, false	2
5	Blocking a single Amazon in multiple ways	true, false	2
6	Move previously blocked Amazon	0, 1, 2	3
7	Move Amazon to which Amazon moved adjacently	true, false	2
8	Move Amazon to which arrow was shot adjacently	true, false	2
9	Block Amazon that moved on previous move	true, false	2
10	Move Amazon not blocking any opponent Amazon	true, false	2

As pointed out, in Amazons it is important to confine the opponent Amazons to a small space, especially in the opening. Blocking the movement of the opponent Amazons, either by the move or by the arrow, is therefore a candidate for a good move. These are categories 1 and 2 in Table 1. An example of a move that blocks three opponent Amazons with both the Amazon and the arrow is given in Fig. 1. The white move blocked three black Amazons with the move (A7, D10 and G10) and also three black Amazons with the arrow (D10, G10 and J7). This is a move that is often played as the first move in a game of Amazons. We make a difference between blocking 0, 1, 2 and 3 or more opponent Amazons, so the total number of categories is 4. Note that we are also making a difference between Amazons and arrows blocking the opponent. While an arrow is a simple block, the Amazon that blocked is not only blocking the opponent Amazon, but at the same time also blocked by the opponent Amazon. We feel that this difference between Amazon and arrow should be reflected in the move categories. This is an important difference with work by Soeda [11], a proposal for move categories for Amazons that made no difference between Amazon and arrow.

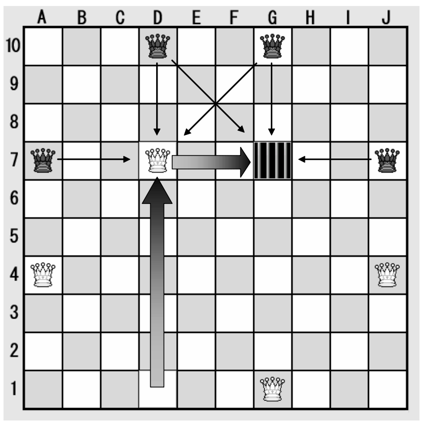

Fig. 1. D1-D7(G7): Amazon and arrow block 3 opponent Amazons.

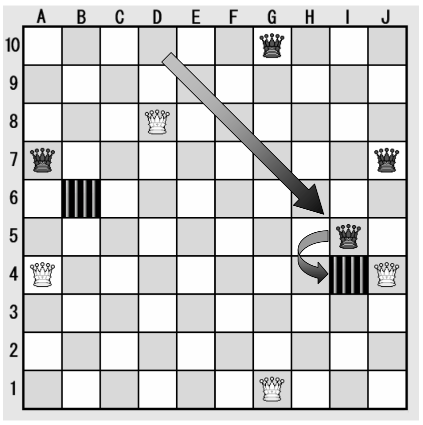

Fig. 2. D10-I5(I4): Amazon and arrow adjacent to opponent (J4), also blocking J4 twice.

Even more aggressive is moving or shooting an arrow to the square adjacent to an Amazon (category 3 and 4). This is often a threat to trap the Amazon within the next few moves. In Fig. 2, the move D10-I5(I4) puts an Amazon and an arrow adjacent to the Amazon on J4. This Amazon on J4 now has very little space and to avoid being trapped white might have to move it next.

If a single Amazon can be blocked in more than one way, this is also a threat to trap this Amazon (category 5). If an opponent Amazon can be trapped early, the game becomes a fight of four Amazons against three, which is often a winning advantage. In Fig. 1, the black Amazons on D10 and G10 are blocked twice and so is the white Amazon on J4 in Fig. 2.

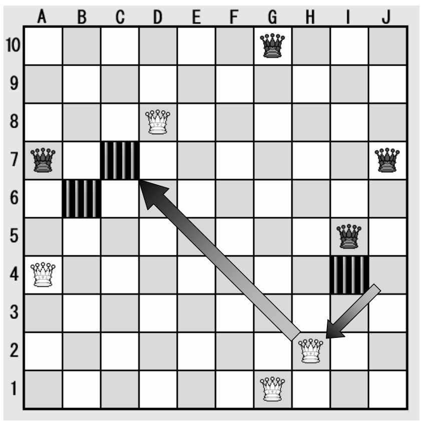

Fig. 3. J4-H2(C7): moving the threatened Amazon (J4), also blocking twice.

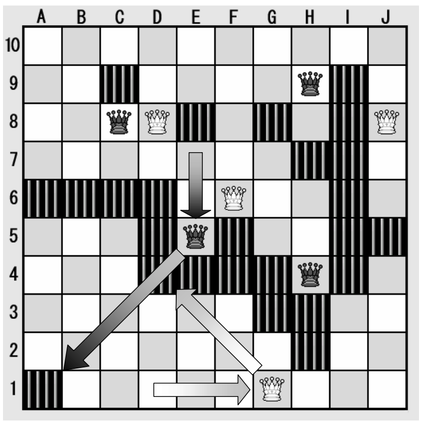

Fig. 4. D1-G1(D4): blocking the Amazon that just moved (E5) with a free Amazon.

Moving the Amazon that is in danger of being trapped is the idea behind categories 6, 7 and 8. In Fig. 2, moving the Amazon on J4 (which was blocked twice) leads to the position of Fig. 3. Moving an Amazon that was blocked on the previous move helps avoiding a trap (category 6). Note that we make a difference between the number of times the Amazon was blocked on the previous move (0, 1 or 2 times), giving a total of 3 different categories. Moving the Amazon to which an Amazon moved adjacently or an arrow was shot adjacently are categories 7 and 8, which mirror categories 3 and 4.

Blocking the Amazon that moved on the previous move (category 9) is based on the assumption that the previous move had meaning. An opponent Amazon tried to avoid being trapped or tried to claim or attack some territory. In Fig. 4, the previous move E7-E5(A1) is an attempt to enter the white territory at the bottom left. To keep the black Amazon from entering, white blocks this Amazon with D1-G1(D4).

Finally, category 10 is a move where an Amazon is not blocking any opponent Amazon. Often, this means that the Amazon is idle and should be moved or can be moved freely. In Fig. 4, the Amazons on G1 and J8 are Amazons that are not blocked by any opponent Amazon. J8 is an Amazon that is almost trapped and moving it to try and escape might be good. The Amazon on G1 is defending the white territory at the bottom left, so this Amazon will often move inside this territory to make sure that the opponent cannot enter.

The 10 basic categories are reflecting different aspects of moves, and we already gave examples of moves belonging to a combination of categories. The total number of combinations is $4^2 \times 3^1 \times 2^7 = 6,144$. Of these 6,144 combinations there are 4,724 combinations that are theoretically impossible. For example, when shooting an arrow adjacent to an opponent Amazon, this also automatically blocks an opponent Amazon at least once, so it never happens that category 4 is true and category 2 is 0. In our experiments, we have used the remaining 1,420 categories.

Using combinations of categories is important, because moves with multiple meanings are expected to be better than moves that belong to only a single category. Examples of move categorization for the moves in the figures are given in Table 2.

Table 2. Examples of move categories.

CatNo.	1	2	3	4	5	6	7	8	9	10	Example move
1	3	3	F	F	T	0	F	F	F	F	Figure 1: D1-D7(G7)
2	1	2	T	T	T	1	F	F	F	F	Figure 2: D10-I5(I4)
3	0	2	F	F	F	2	T	T	F	F	Figure 3: J4-H2(C7)
4	0	1	F	T	F	1	F	F	T	T	Figure 4: D1-G1(D4)

3 Priority Ordering of Moves

To assess the importance of the categories proposed in Sect. 2, we investigated how often a move from a certain category was actually played. For this we used 11,000 games from our Amazons program The Amazons Saga (TAS). In recent Computer Olympiads, TAS has shown that it can play on par with the strongest Amazons programs.

The method we used is the same as for determining the realization probabilities of move categories in Realization Probability Search (RPS), which has been a very successful approach in shogi [7]. The realization probability of a category is calculated with the following formula:

$$P_i = \frac{A_i}{B_i}$$

P_i: The realization probability of category i
A_i: Number of times a move from category i was played
B_i: Number of positions where a move from category i was possible

The realization probability of a category is the ratio of positions where a move from a certain category was possible and the number of times that this move was actually played. The calculation has only been done until the stage where all territory is fixed. By fixed territory is meant that all Amazons have their own (i.e. non-overlapping) territory and the rest of the game is only about filling the territory with the maximum number of moves. Because this is a simple counting problem, usually a game of Amazons is stopped in a position with fixed territory. Both players count the number of moves needed and the winner is agreed upon.

Although in RPS the game records of expert players are used, we think it is important to use the game records of the same program to calculate the realization probability values. These values strongly depend upon the evaluation function, so improvement of search speed can only be expected if the moves are ordered in the way the program 'likes them', i.e. that have a high probability of leading to a good evaluation. Ideally, realization probabilities should be recalculated with every change in the evaluation function. However, we feel that the evaluation function of TAS is stable enough to give reliable results.

A similar method for calculating realization probabilities in the absence of expert moves was proposed by Hashimoto et al. [12]. Their method calculates the probabilities by having a Lines of Action program play itself, recording the categories of the moves that were possible in each position and the category of the move that was selected after searching the position. If these positions and the selected moves are stored for future recalculation of the realization probabilities, the method is identical to ours. If not, our method has the advantage that if categories are changed, the recalculation of the realization probabilities can be done without search, using the standard set of games. If the evaluation function is changed (i.e. the program might select different moves), a new set of games is necessary for both methods.

The realization probabilities of the 10 basic categories of Table 1 are given in Table 3.

Table 3. The realization probabilities of the basic categories.

Category	RP (%)
Move blocked 0 Amazons	33.1
Move blocked 1 Amazon	56.9
Move blocked 2 Amazons	13.1
Move blocked 3 or more Amazons	6.6
Arrow blocked 0 Amazons	14.9
Arrow blocked 1 Amazon	71.4
Arrow blocked 2 Amazons	17.2
Arrow blocked 3 or more Amazons	5.3
Move adjacent to opponent Amazon	52.3
Arrow adjacent to opponent Amazon	72.3
Multiple opponent block	36.2
Blocking the previously moved Amazon	50.6
Moving the Amazon that was blocked once by the previous move	49.0
Moving the Amazon that was blocked twice by the previous move	33.9
Moving Amazon to which Amazon moved adjacently	32.8
Moving Amazon to which arrow was shot adjacently	34.5
Moving a non-blocking Amazon	45.6

The realization probabilities of Table 3 show that our basic categories in general have a high probability of being played and are therefore valid candidates for good moves. There are two notable exceptions: the categories "Move blocked 3 or more Amazons" and "Arrow blocked 3 or more Amazons" have a low realization probability which seems counter-intuitive. However, our experience with Amazons shows that the high mobility of Amazons makes it very rare to trap more than one Amazon. Rather than trying to confine multiple Amazons, it is better to try and trap a single Amazon and then use the advantage of having four Amazons fighting three.

In general RPS, the next step is to use the realization probabilities of move categories to decide the depth of the search. Moves with a high realization probability are searched deeper than moves with a low realization probability. Unfortunately, this method can currently not be used in Amazons programs because of the aforementioned even-odd iteration effect. Searching moves with a high probability one ply deeper makes the search unstable. General RPS is therefore not feasible until there is a solution to the even-odd iteration effect in Amazons.

Instead of using realization probabilities to decide the depth of the search, we propose to use the realization probabilities of Table 3 to order moves of new positions. Moves with a high realization probability have a higher probability of being played, so a correlation between realization probability and good moves can be expected.

As pointed out earlier, in our program the 10 basic categories are not used directly. Instead, the 1,420 theoretically possible combinations of basic categories are used. The calculation method for these categories is the same as those for the basic categories. Examples of the categories used in TAS and their realization probability are given in Table 4.

Table 4. Realization probabilities of move categories.

CatNo.	Example move	RP (%)	Order
1	Figure 1: D1-D7(G7)	32.615	2
2	Figure 2: D10-I5(I4)	0.902	371
3	Figure 3: J4-H2(C7)	0.158	876
4	Figure 4: D1-G1(D4)	7.250	25

In Table 4, the realization probability of the moves in Table 2 are given. The table shows that D1-D7(G7) is a move belonging to a category with an extremely high probability (second highest among the 1,420 categories). The reason for this is that such moves are only possible in the opening. Actually, the category with the largest realization probability was only possible in 10 positions. In contrast, the probability of J4-H2(C7) is very low, even though this can be considered a good move. One way of improving the move ordering is to adjust the category priorities according to game progress, which we will explain next.

4 Adjusting Category Priorities Using Game Progress

The strategic features of Amazons shift as the game progresses. In the opening, the mobility and balance of the Amazons are most important. As the game progresses, it becomes more important to secure territory. Large territories in which Amazons can move freely will lead to more available moves at the end of the game, so the opponent will run out of moves first.

Because of this, the realization probabilities of Table 3 are likely to change as the game progresses. To reflect this shift in realization probabilities, the progress of the game has to be used to adjust the realization probabilities of the proposed move categories. There are a number of ways to measure progress in the game of Amazons, for example move number or using territory measurements. Here we will restrict ourselves to the most basic progress measurement: move number. In Amazons, the number of arrows on the board grows with each move. Therefore, the game is over after a maximum of 92 moves. Using the number of arrows (i.e. the number of moves) is therefore a natural choice for measuring progress in Amazons.

For this analysis the same 11,000 games as for calculating the realization probability were used. Despite this large number of games, near the end of the game the moves in certain categories are almost never played. For example, positions where it is possible to block three Amazons or more become very rare near the end of the game. Although this might be a problem of our method, the number of possible moves near the end of the game is relatively small, so deep search can be conducted even without using move categories. Therefore, we do not expect that this problem will influence playing strength much.

Because a sudden change in realization probability is undesirable, we grouped data for the change in realization probability into groups of 8 moves: 0-7 (note that there is no data for move 0), 8-15, 16-23, etc. The results of this analysis for blocking by Amazon move and blocking by arrow (categories 1 and 2) are given in Fig. 5 and Fig. 6 respectively.

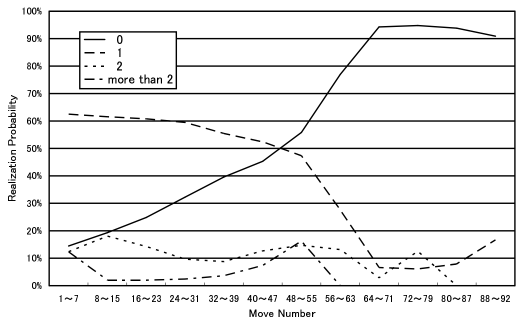

Fig. 5. Changes in realization probability for blocking with the Amazon.

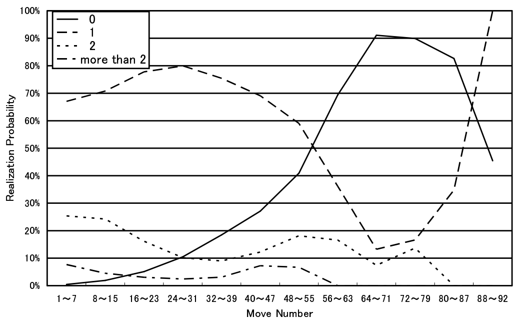

Fig. 6. Changes in realization probability for blocking with the arrow.

From these graphs it is clear that the realization probability of categories can change dramatically as the game progresses, so rather than taking the average realization probability over the whole game, it seems more promising to use realization probabilities based on game progress. Examples of the actual realization probabilities used in TAS for the four moves in Table 4 are given in Table 5.

Table 5. Game progress related realization probabilities (%) for the moves in Table 4.

Mvno	Category number			
	1	2	3	4
1 - 7	32.65	0.51	0	0.52
8 - 15	0.01	0.99	0.18	1.20
16 - 23	0.01	0.88	0.21	4.20
24 - 31	0.01	2.22	0.20	8.32
32 - 39	0.01	2.41	0.80	18.39
40 - 47	0.01	1.76	2.53	17.75
48 - 55	0.01	17.65	0.01	30.86
56 - 63	0.01	0.01	0.01	50.44
64 - 71	0.01	0.01	0.01	37.50
72 - 79	0.01	0.01	0.01	75.00
80 - 87	0.01	0.01	0.01	0.01
88 - 92	0.01	0.01	0.01	0.01

5 Experimental Results

To investigate the significance of using game progress combined with move ordering based on our move category proposal, we have used our program TAS. TAS has the following properties:

- General features
  1. Iterative deepening α-β search.
  2. Best move of previous iteration is searched first.
  3. The evaluation function is a linear function of Queen Move Distance, King Move Distance (both explained by Lieberum [5]) and mobility of Amazons.
- Move generation
  1. Each Amazon is assigned a number in TAS. Moves for each Amazon are generated in the order of these numbers, i.e. first all moves for Amazon 1, then all moves for Amazon 2, etc.
  2. The moves of each Amazon and arrow are generated in clockwise order: moving up, moving right up, moving right, etc.
- Priority ordering
  1. Moves are ordered based on realization probability adjusted by game progress.
  2. Moves that are not in any category (i.e. their realization probability is 0%) are pruned.

Our experiments have been conducted with three versions of the same program. The first version is searching without the use of move categories (called *NMC*), generating moves in the order explained above. The second version uses the realization probabilities, but no game progress (called *NGP*). The third version uses realization probabilities adjusted by game progress (called *GP*).

There are 263 move categories that have a realization probability of 0% (i.e. possible but not played) when game progress is not taken into account. When using realization probabilities based on progress, the number of categories with a probability of 0% changes from 590 categories in move 1-7 to 2 or less after move 48. These numbers should be seen relative to the number of possible categories. In move 1-7 there are 1324 categories that are possible, while in move 88-92 only 6 categories are possible. Details are given in Table 6.

Table 6. Number of categories with a realization probability of 0%.

Mvno	NP	(Pos)	MoveNo	NP	(Pos)	MoveNo	NP	(Pos)
1 - 7	590	(1324)	32 - 39	45	(1218)	64 - 71	2	(127)
8 - 15	214	(1404)	40 - 47	9	(944)	72 - 79	1	(90)
16 - 23	140	(1395)	48 - 55	1	(615)	80 - 87	0	(45)
24 - 31	100	(1335)	56 - 63	1	(276)	88 - 92	0	(6)

We have conducted two experiments. First, we selected a number of positions and let each of the test programs search to depth 3, comparing the search time. The results of this experiment give an indication of the speed-ups that can be expected. The second experiment is a self-play experiment between the different versions. By using categories, certain moves will be discarded without any search (the moves that do not fit into a category), so there is a risk of missing good moves that are not in our categories. The self-play experiments will show how playing strength is influenced by using move categories.

5.1 Comparing Search Times to Depth 3

To investigate the savings in search effort, we compared the search times of searching to depth 3 for the three program versions in a number of example positions. Because the number of legal moves in an average Amazons position is high, searching to depth 3 for a large number of positions is infeasible. We limited ourselves to a set of 1,521 positions from 30 Amazons games. The data for 30 games was very similar to the data for 25 games, so we do not think that using more positions will lead to many new insights.

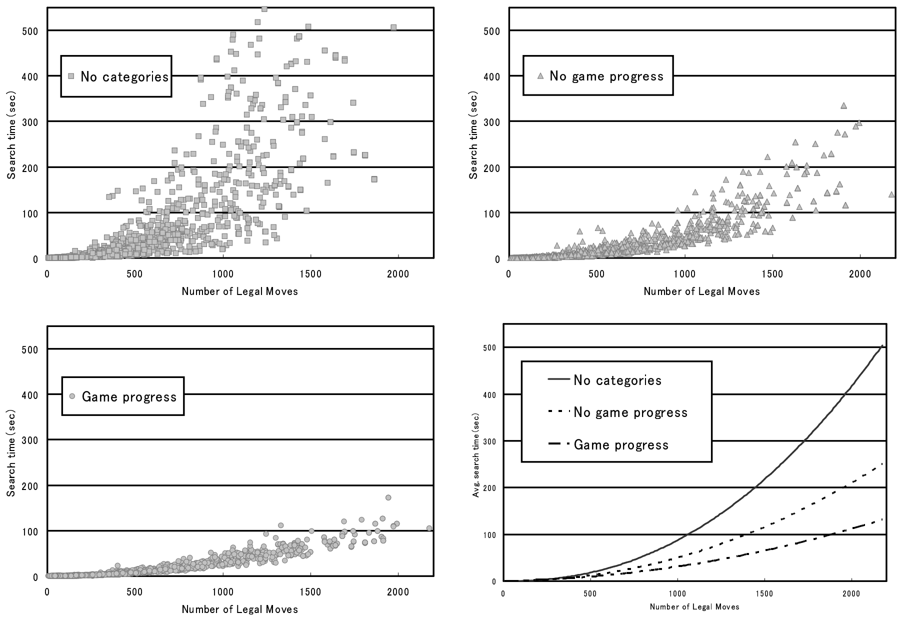

Fig. 7. Search times to depth 3 against number of legal moves for *NMC* (upper left), *NGP* (upper right), *GP* (lower left) and the average search times for each program version (lower right).

In Fig. 7 the search times of searching to depth 3 in the positions of 30 games are given for *NMC*, *NGP* and *GP*. From this figure it is clear that without using move categories, there are many positions that require a long time to finish the search to depth 3. Also, without move categories there seems to be no strong relation between the number of legal moves and the search time. Using move categories improves the search times considerably and there is a much stronger relation between search times and the number of legal moves. Finally, using move categories and game progress further improves the search speed. Also, there is a very strong relation between search times and the number of legal moves when game progress is taken into account.

From the graph showing the average search times against the number of legal moves, it is clear that in case of a high number of legal moves searching without move categories becomes quickly infeasible. Using move categories is more promising and using move categories based on game progress is more promising than using only move categories.

There were a number of positions where *NMC* returned a different move than *NGP* or *GP*. *NGP* played a different move in 95 positions, while *GP* played a different move in 98 positions. The reason for this difference was that a move in the principle variation of *NMC* was deleted because it had a realization probability of 0%. This seems to indicate that there is a risk that good moves are being discarded, but investigating the positions in question showed that none of the discarded moves were particularly good or necessary. From these results, we concluded that the current categories are good enough to lead to an improvement in search speed without losing playing strength. To confirm this, we conducted a number of self-play experiments that we will explain next.

5.2 Results of Self-play Experiments

In Amazons, there is the problem of generating positions where the chances of winning can be considered equal, because there are no experts and there is almost no opening theory. We generated the initial positions of our self-play experiments by using 50 different positions that were randomly selected from the opening book (after the fourth move from the starting position of the game). These positions were then played twice by each program version, once with white and once with black. It is possible that the initial positions are better for one side, so we also collected data of the number of squares that the winning side could freely move to at the end of the game. Even if both versions win a game from the same starting position, the difference in free squares at the end of the game gives an indication of difference in playing strength.

We played a total of 9 matches with 100 games each, giving each program 10 seconds, 30 seconds and 60 seconds per move. The results of these matches are given in Table 7. The results of the matches show that our concerns about ending up in uneven positions were unfounded. Even without considering the differences in free squares at the end of the game, each match result except one gives a statistical probability of more than 95% that the winning version is stronger than the losing version (the 58-42 result between *NGP* and *NMC* gives a probability of 94.5% that *NGP* is stronger than *NMC*). Furthermore, the results are not influenced by the amount of time given per move. The results of the matches for 10 seconds and 60 seconds are very close and it seems that the only difference is that more time can reduce the margin of defeat, as can be seen in the drop of the total number of free squares.

Table 7. Self-play results for 100 games with 10, 30 and 60 seconds per move.

Match	10 seconds		30 seconds		60 seconds	
	Result	SqDif	Result	SqDif	Result	SqDif
NGP - NMC	58 - 42	+146	60 - 40	+92	59 - 41	+87
GP - NMC	79 - 21	+428	72 - 28	+439	75 - 25	+388
GP - NGP	62 - 38	+214	70 - 30	+259	63 - 37	+165

In Table 8 the results for each program version are summarized. The self-play experiments show that using move categories for move ordering significantly improves the playing strength of an Amazons program and that playing strength can be further improved significantly by using game progress to adjust the realization probabilities of the move categories.

Table 8. Total self-play results.

No	Version	10 seconds	30 seconds	60 seconds
1	GP	141 - 59	142 - 58	138 - 62
2	NGP	96 - 104	90 - 110	96 - 104
3	NMC	63 - 137	68 - 132	66 - 134

6 Conclusions and Future Work

In this paper, we have presented a method for improving the search efficiency of an Amazons program by ordering moves based on move categories. We have also investigated the influence of game progress on the move ordering. Experiments showed that a program using move categories is stronger than a program without move categories and that a program using move categories based on game progress is stronger than a program not taking game progress into account.

One area of future work concerns the realization probabilities of the categories. As explained, there are a number of categories that are so special that they occur in only a small fraction of the positions. As a result, these categories often get a very high or very low probability. To address this problem, different ways to decide realization probabilities need to be investigated.

A different problem is that the different program versions all have the same evaluation function, leading to mutual oversights. Improvement of general playing strength needs to be further assessed by playing other strong Amazons programs like Amazong or Invader.

Another important future work concerns the representation of game progress. The game progress we have used in our experiments is based upon the number of moves played. This is not a perfect solution, because the moment when territory gets fixed differs from game to game and is only loosely related to the number of moves. The number of moves seems to be a good way of measuring progress when games develop in a normal way, but in case of abnormal development (very early or very late fixed territory) the proposed category probabilities can lead the program astray. As a future work, we intend to investigate different methods for measuring progress and compare these with our current findings.

Finally, using game progress to change realization probabilities is an idea that can be applied to other games than Amazons. RPS in shogi might also benefit from dynamically updating realization probabilities using game progress. We are planning to investigate the feasibility of our method in other games than Amazons.

References

1. Wikipedia: http://en.wikipedia.org/wiki/The_Game_of_the_Amazons (2005)
2. Sasaki, N., Iida, H.: Report on the First Open Computer-Amazon Championship. ICCA Journal **22** (1999) 41–44
3. Matsubara, H., Iida, H., Grimbergen, R.: Natural developments in game research: From Chess to Shogi to Go. ICCA Journal **19** (1996) 103–112
4. Hashimoto, T., Kajihara, Y., Sasaki, N., Iida, H., Yoshimura, J.: An Evaluation Function for Amazons. In Van den Herik, H., Monien, B., eds.: Advances in Computer Games 9. Van Spijk, Venlo, The Netherlands (2001) 191–201
5. Lieberum, J.: An Evaluation Function for the Game of Amazons. In Van den Herik, H., Iida, H., Heinz, E., eds.: Advances in Computer Games 10. Kluwer Academic Publishers, Boston, USA (2003) 299–308
6. Avetisyan, H., Lorentz, R.: Selective Search in an Amazons Program. In Schaeffer, J., Müller, M., Björnsson, Y., eds.: Computers and Games: Proceedings CG2002. LNCS 2883. Springer-Verlag, Berlin (2002) 123–141
7. Tsuruoka, Y., Yokoyama, D., Chikayama, T.: Game-tree Search Algorithm Based on Realization Probability. ICGA Journal **25** (2002) 145–152
8. Beal, D.: A Generalised Quiescence Search Algorithm. Artificial Intelligence **43** (1990) 85–98
9. Buro, M.: ProbCut: An Effective Selective Extension of the alpha-beta Algorithm. ICCA Journal **18** (1995) 71–76
10. Buro, M.: Experiments with Multi-probcut and a New High-quality Evaluation Function for Othello. In den Herik, H., H.Iida, eds.: Games in AI Research. Van Spijk, Venlo, The Netherlands (2000) 77–96
11. Soeda, S., Tanaka, T.: Categories for Amazons Moves. In: Game Programming Workshop in Japan '03, Kanagawa, Japan (2003) 118–121 (In Japanese).
12. Hashimoto, T., Nagashima, J., Sakuta, M., Uiterwijk, J., Iida, H.: Application of Realization Probability Search for Any Games - a case study using Lines of Action -. In: Game Programming Workshop in Japan '02, Kanagawa, Japan (2002) 81–86 (In Japanese).