More Games of No Chance
MSRI Publications
Volume **42**, 2002

Experiments in Computer Amazons

MARTIN MÜLLER AND THEODORE TEGOS

Abstract. Amazons is a relatively new game with some similarities to the ancient games of chess and Go. The game has become popular recently with combinatorial games researchers as well as in the computer games community. Amazons combines global full-board with local combinatorial game features. In the opening and early middle game, the playing pieces roam freely across the whole board, but later in the game they become confined to one of several small independent areas.

A *line segment graph* is an abstract representation of a local Amazons position. Many equivalent board positions can be mapped to the same graph. We use line segment graphs to efficiently store a table of *defective territories*, which are important for evaluating endgame positions precisely. We describe the state of the art in the young field of computer Amazons, using our own competitive program *Arrow* as an example. We also discuss some unusual types of endgame and *zugzwang* positions that were discovered in the course of writing and testing the program.

1. Introduction

The game of Amazons was invented by Walter Zamkauskas. Two players with four playing pieces each compete on a 10×10 board. Figure 1 shows the initial position of the game. The pieces, called *queens* or *amazons*, move like chess queens. After each move an amazon shoots an arrow, which travels in the same way as a chess queen moves. The point where an arrow lands is *burned off* the playing board, reducing the effective playing area. Neither queens nor arrows can travel across a burned off square or another queen. The first player who cannot move with any queen loses.

Amazons endgames share many characteristics with Go endgames, but avoid the extra complexity of Go such as ko fights or the problem of determining the safety of stones and territories. Just like Go, Amazons endgames are being studied by combinatorial games researchers. Berlekamp and Snatzke have investigated play on sums of long narrow $n \times 2$ strips containing one amazon of each

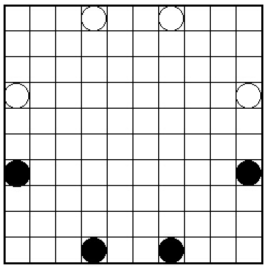


Figure 1. The playing board of Amazons.

player [1; 15]. Even though $n \times 2$ areas have a simple structure, sum game play is surprisingly subtle, and full combinatorial game values become very complex.

Amazons has also caught the attention of programmers who work on games such as shogi, Othello or Go. In 1998 and 1999, Hiroyuki Iida organized the first two computer Amazons championships, which were held at the Computer Games Research Institute of Shizuoka University [10; 13]. The winning program in both tournaments was *Yamazon*, written by the leading shogi programmer Hiroshi Yamashita [16]. Shortly after the second tournament, Michael Buro's *amsbot* won a *dream match* played on the GGS server [6] against the top four contestants. In summer 2000, another Amazons tournament was held at the Computer Games Olympiad in London. *8QP* by top chess programmer Johan de Koning convincingly won all its games in a field of six mostly new programs. *Yamazon* took second place. Yet another strong new program, *Amazong*, was developed by Jens Lieberum, who was the winner of the human Amazons tournament held at the combinatorial games workshop at MSRI in Berkeley in 2000.

This paper contributes to our understanding of the game of Amazons as follows: Section 2 discusses partitioning an Amazons board. In Section 3 we introduce the new concept of a *line segment graph*, which provides an abstract representation of a local Amazons position. Section 4 deals with *territories*, which are areas controlled by one player, and with the problem of *defective* territories. As an empirical result, we computed tables of small defective territories. Section 5 discusses *zugzwang* positions, which frequently occur in the game of Amazons. Section 6 contains a brief discussion of an evaluation function for computer Amazons, and Section 7 wraps up with a summary, pointers to Amazons resources and future work.

2. Board Partitioning in Amazons

The aim of board partitioning is to decompose a full-board Amazons position into a sum of independent subgames which can be analyzed locally. In the opening, creating such a partition is impossible, since the whole board is connected in many ways. In the late middle game and endgame, the board becomes more and more fragmented.

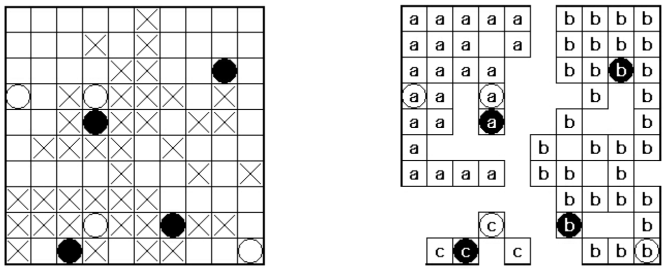


Figure 2. Board position and basic decomposition.

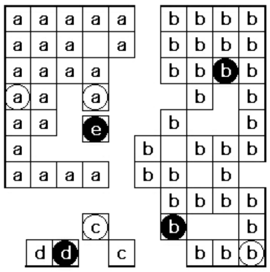


Figure 3. Partition using blocking amazons.

Starting from a given Amazons position, a basic board partition is given by the 8-connected components of all points that have not yet been burned off by arrows. Areas are independent since amazons can never move or shoot across a burned-off square into a different connected component. Figure 2 shows a board position and its basic decomposition. Burned-off squares are either indicated by a cross in the figures or omitted altogether.

Often, amazons which help to wall off some area as *territory* can be utilized to improve the board partitioning. Figure 3 shows such an improved decomposition for the same position as in Figure 2. The basic area c of Figure 2 has been divided into two areas c and d controlled by White and Black respectively. Area a has been further subdivided into a *dead* black stone e and a remaining area a controlled by two white amazons.

A *blocker* is an amazon that divides the board into two or more regions. Decomposition using blockers is not as strict as the basic decomposition into connected components, since blockers can move away. However, blockers often help to surround a territory. When they leave their position, they always have the option of shooting an arrow back at their origin square, and thereby keeping the partition intact.

3. Line Segment Graphs

For the analysis of Amazons positions, the fact that it is played on a rectangular grid does not matter. Even after taking into account the obvious symmetries of mirroring, rotation and translation, many positions that look different on a grid are in fact equivalent in terms of possible moves and game outcomes. Consider a representation of an Amazons board where all points on the same contiguous horizontal, vertical or diagonal line are joined by a line segment. All the structural information about an Amazons position is contained in two geometric primitives:

1. The points that are contained in each line segment, and
2. the relative order of the points on a line segment.

For brevity, we will often use the term line instead of line segment. The actual embedding of lines on a grid does not matter. For game play, of course, it must be known which points are occupied by amazons.

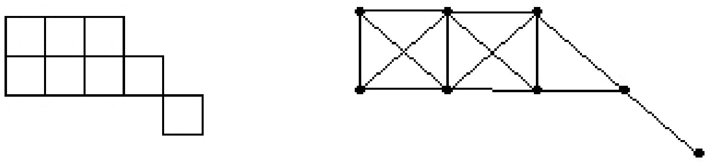
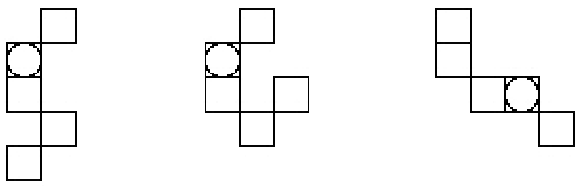


Figure 4. An Amazons region and its line segment graph.



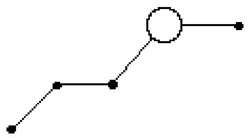


Figure 5. Amazons regions sharing the same line segment graph.

Example 1. Figure 4 shows an Amazons region and its underlying structure of line segments. Figure 5 illustrates that differently shaped regions on an Amazons board can be mapped to the same constellation of line segments.

The following definitions formally specify a *line segment graph (LSG)* and describe its basic properties. Let $V = \{v_0, \dots v_k\}$ be a finite set of points.

Definition 1. A line segment over V is an ordered set of two or more points $[p_0, ..., p_n]$, $n > 0$, with the properties

1. Distinctness: $p_i \in V$ and $p_i = p_j \iff i = j$.
2. Ordering on a line: $[p_0, ..., p_n] = [q_0, ...q_m] \iff n = m$ and either $p_i = q_i$ or $p_i = q_{n-i}$ for all $i = 0 \ldots n$.

In words, only the relative order of points on the line is significant.

Definition 2. The *line distance* between points p_i and p_j on a line $[p_0, ..., p_n]$ is defined by $ld(p_i, p_j) = |i - j|$. $ld(p, q) = \infty$ if p, q are not on a line.

Definition 3. A *line segment graph* G is a pair $(V, L(V))$, where V is a finite set of points and $L(V)$ a set of line segments over V.

We'll call a LSG *well-formed* if it fulfills the following simple geometric conditions:

1. each point is contained in at least one line.
2. the unique intersection property: two distinct lines have either zero or one points in common. We write $l \cap k = p$ if p is the unique point contained in both l and k, and $l \cap k = \varnothing$ otherwise.

Definition 4. A *path* $P(p, q)$ between two points p, q in a line segment graph $(V, L(V))$ is a set of line segments $\{l_1 \ldots l_n\}$ such that $p = p_0, q = p_n, p_{i-1} \in l_i, p_i \in l_i, l_i \cap l_{i+1} = p_i$.

The distance between p and q along path $P(p, q)$ is defined as

$$d(p, q, P) = \sum_{i=1}^{n} ld(p_{i-1}, p_i).$$

The shortest distance between points in a LSG is defined as the minimum distance along all possible paths connecting p and q, and as ∞ in the case that no path exists.

Definition 5. A line segment graph $(V, L(V))$ is *connected* if there is a path between each pair of points.

LSG created from Amazons positions on a standard board obey further restrictions:

1. The shortest path property: if two points p, q are on the same line, $ld(p, q) < \infty$, then there is no shorter path between them: $d(p, q) = ld(p, q)$;
2. the grid restriction: each point is contained in at most four lines; and
3. the line length restriction: no line is longer than the board size.

3.1. Generating Small Line Segment Graphs. We have written programs that generate all Amazons positions up to size 10 that can fit within some rectangular box of size up to 56. We have also converted all these positions into LSG form. Table 1 shows the numbers and Figure 6 the ratio between the number of LSG and Amazons positions on a board. For both positions on the board and LSG the numbers have been minimized by eliminating isomorphic cases. Figure 7 shows all LSG up to size 4 that can be realized on a standard board.

Size	Board	LSG
1	1	1
2	2	1
3	5	3
4	22	11
5	94	42
6	524	199
7	3031	960
8	18769	4945
9	118069	25786
10	755047	137988

Table 1. Small regions and LSG.

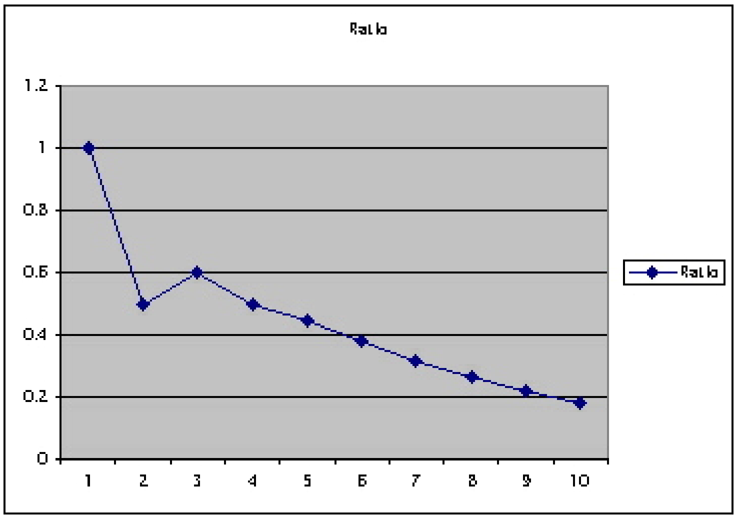


Figure 6. Ratio LSG/board positions.

3.2. Isomorphism Testing for Line Segment Graphs. In order to build and use a database of LSG, isomorphism of line segment graphs must be tested efficiently. The following mapping, proposed by Brendan McKay in a private communication, transforms the LSG isomorphism test problem into isomorphism testing for colored undirected "ordinary" graphs. The result is in a format suitable for input to *nauty*, McKay's efficient program for computing graph automorphisms and isomorphisms [11; 12]. Given an LSG $G = (V, L(V))$, create a colored graph $G' = (V', E, col)$ as follows:

1. Both points and lines of G become vertices of G': $V' = V \cup L(V)$. It is also possible and more efficient to omit all the lines of length 2 here.

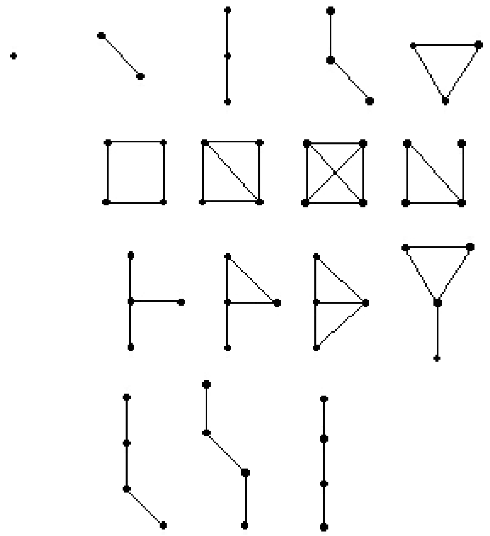


Figure 7. All LSG up to size 4.

2. In G', color all vertices from V with one color (green), and all vertices from $L(V)$ with a different color (red).
3. Construct the set of edges E: Connect green points if and only if they are adjacent points on some line. Furthermore, connect a green point representing a point v with a red point representing a line l if and only if $v \in l$.
4. Extend the mapping to graphs containing amazons by changing the coloring of the points which contain amazons from green to the color of the amazon.

It is easy to see that a given G' completely determines the underlying LSG G, even with the optimization that does not create red points for lines of length 2.

4. Filling Territory and Defective Areas

An area that contains only amazons of one color is a *territory* and represents a number of free moves for one player. The question is exactly how many moves? In most cases, a territory can be completely filled, and n empty squares yield n moves.

Definition 6. A *k-defective territory* provides k less moves than the number of empty squares in the territory. A k-defective territory is said to have k *defects*.

Example 2. Figure 8 shows the three smallest defective territories, namely two 1-defective areas with 2 empty squares and one 2-defective territory with 3 empty squares. In each case, the amazon can only move once because she has to shoot back to her current position, disconnecting herself from the remaining empty square(s).

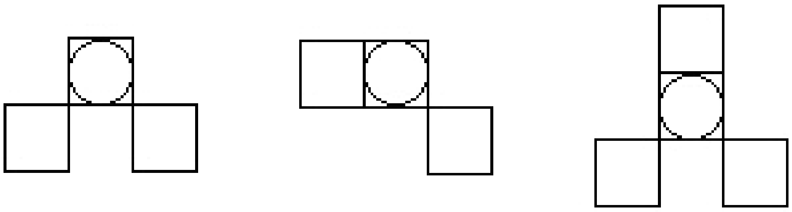


Figure 8. Small defective territories.

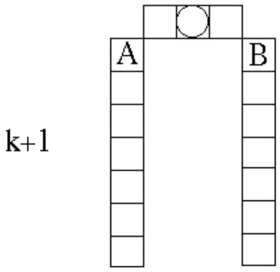


Figure 9. k-defective territory.

Example 3. Figure 9 illustrates that there are k-defective areas for any k. The amazon in the middle cannot avoid losing one of the long strips on the left or on the right. The furthest points that the queen can reach with its first shot are A and B, and all k or more points beyond the first shot are lost.

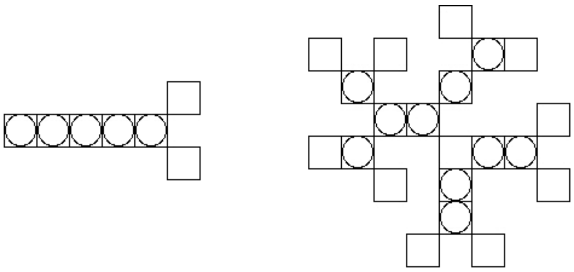


Figure 10. Defective territories containing many amazons.

Usually, two amazons in the same territory are much more powerful than a single one. However, there are small defective areas containing two or more amazons. Figure 10 shows some examples.

In theory, it is difficult to decide whether a given territory provides a certain number of moves. Michael Buro has recently proven that this is an NP-complete problem [5]. In practice, a table of all small defective areas combined with a simple filling heuristic for larger areas seems adequate.

4.1. Dead Ends.

Definition 7. A *dead end* in a LSG is a subgraph with the following property: if an amazon moves into the dead end (or already is in the dead end), she either

1. cannot fill the dead end region completely, or
2. has to disconnect the region from the rest of the board.

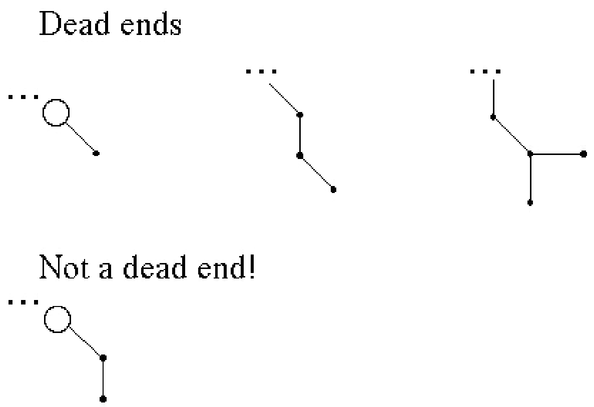


Figure 11. Dead ends.

Figure 11 shows some examples of dead ends, and one example which is not a dead end.

Definition 8. The *defect count* of a dead end is the minimum number of defects caused by a dead end if an amazon has to move out of the dead end. In the first two examples in Figure 11 the defect count is one while in the rightmost example the defect count is two.

LSG are certain to be defective if they contain more disjoint dead ends than amazons. To avoid a local defect in a dead end, some amazon's path has to end there. Therefore, each amazon can eliminate at most one dead end. Given a LSG containing n amazons and a number of disjoint dead ends, the sum of all but the n largest defect counts of dead ends is a lower bound on the number of defects of the LSG.

4.2. Small Defective Regions. We have built a database of all defective regions up to size 11 that can fit into a box of maximum size 56 on a 10×10 board, and again converted them to LSG format. Table 2 shows the number of defective board positions and LSG for areas up to size 11. Empirically, the number of LSG of a given size and defectiveness seems to grow exponentially slower than the number of such positions on the Amazons board. Figure 12 shows the ratios on a logarithmic scale. As a concrete example, Figure 13 shows all 12 3-defective Amazons areas of size 8 in our database and the single LSG that is equivalent to each of the 12 areas. Figure 14 shows all small highly defective

Size	1-defective	2-defective	3-defective	4-defective
3	2/1	-	-	-
4	2/1	1/1	-	-
5	37/6	-	-	-
6	236/12	23/3	-	-
7	2238/125	101/8	5/1	-
8	16442/666	1111/71	12/1	-
9	125797/4610	12974/370	274/14	-
10	929277/28500	137976/2665	5464/70	42/3
11	6747413/186564	1208467/17974	193410/1310	1188/17

Table 2. Small defective regions and LSG.

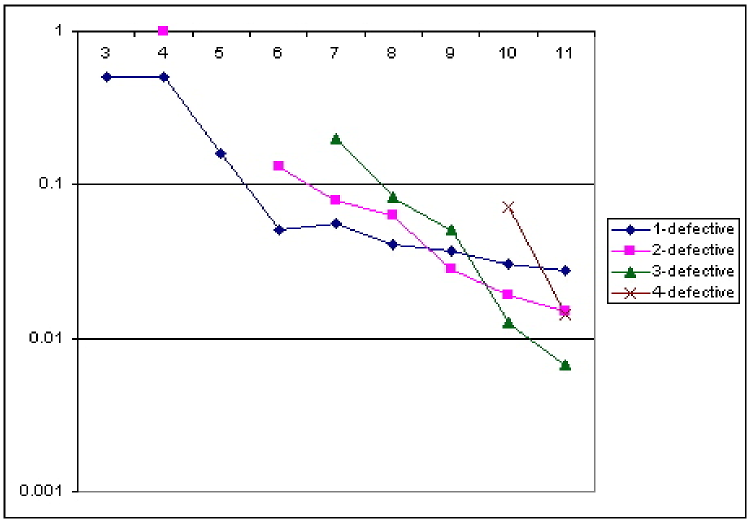


Figure 12. Ratio of number of LSG and number of regions on board.

LSG from the database: 1-defective LSG of size 3, 4 and 5, 2-defective LSG of size 4 and 6, 3-defective LSG of size 7 and 8 and 4-defective LSG of size 10.

5. Zugzwang in Amazons

Zugzwang positions in Amazons are interesting because in contrast to most other games, they have to be played out, and it is nontrivial to play them well.

5.1. Definition and Examples of Zugzwang Positions.

Example 4. The left side of Figure 15 shows an example of *zugzwang* in Amazons. A white blocker sits on an articulation point of its territory. If White is to move, one side or the other of the territory below is lost, since the white

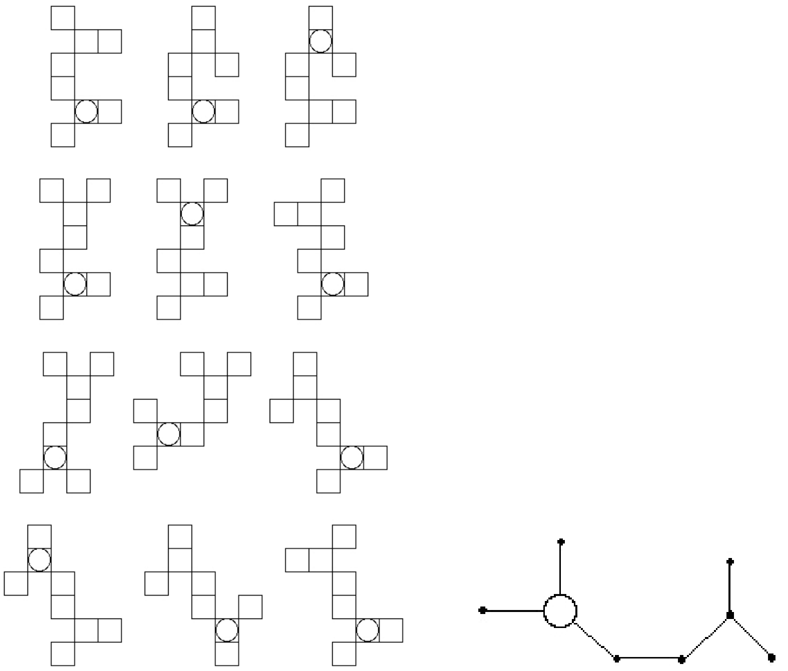


Figure 13. 3-defective Amazons areas and LSG of size 8.

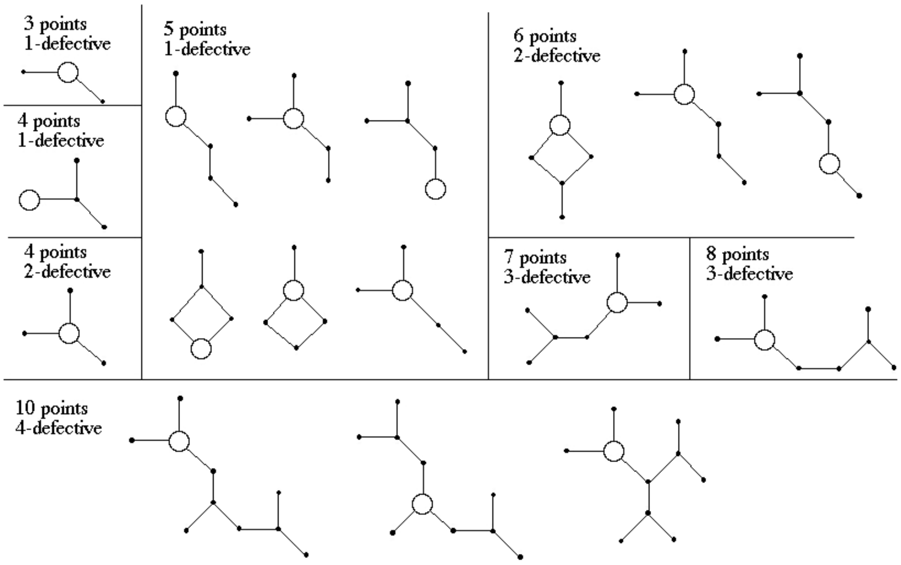


Figure 14. Table of all small highly defective LSG.

queen must shoot back to its origin square to prevent black from moving in there. White would much prefer if Black moved first since Black can only retreat to the territory above and block the entrance, thereby giving White control over

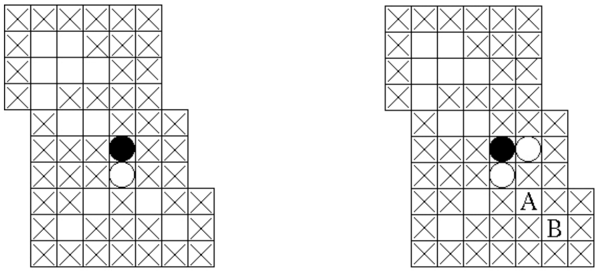


Figure 15. Zugzwang (left), and a blocker that need not block (right).

the complete area on both sides below. The combinatorial game value of this zugzwang position is $3|6 = 4$. Black is guaranteed four more moves than White locally, but can get more if White is forced to move first.

Example 5. The position shown on the right side in Figure 15 looks similar to the zugzwang on the left side but it is different. White can obtain all points by refusing to defend the blocking point. In this position, White can move to A and shoot at B, then *plod* along and fill the whole area in right-to-left order. The black amazon can never move down, since it is tied to the defense of its own, bigger, territory by the other white queen. The combinatorial game value of this position is 4 as well, but White can never get into zugzwang.

Definition 9. We define a *simple zugzwang* position in Amazons as a game $a|b$ with a, b integers such that $a < b - 1$.

By the *Simplicity Rule* [2] of combinatorial game theory, the value of a zugzwang position $a|b$ is the simplest number x such that $a < x < b$. For example, $-2|1 = 0$, $-10|-2 = -3$ and $3|6 = 4$. In Amazons, the first player to get into zugzwang can still win the overall game. Zugzwang positions do *not* affect the main question of which player can win a sum game. In a full board situation involving one or more zugzwangs, it is sufficient to treat the zugzwangs just like an ordinary number.

Zugzwangs do matter if we want to determine the exact score with optimal play. Optimizing the score is important since it is used as the tiebreaking method in tournaments.

Example 6. Figure 16 illustrates a different type of zugzwang. A white amazon defends an area W of size w, a black amazon defends an area B of size b, and both have the option of either retreating to their area or trading it in for a common region C of size c. An amazon who retreats gives the area C to the opponent, while an amazon who moves into C loses control over her own area. In the example in Figure 16, $w = 8, b = 9$ and $c = 3$. Black's options are $B_1 = b - 1 - w - c$ (retreat) and $B_2 = c - 1 - w$ (trade), while White's options are $W_1 = b + c - (w - 1)$ for retreating and $W_2 = b - (c - 1)$ for trading. In the example, $G = \{B_1, B_2 | W_1, W_2\} = \{-3, -6 | 5, 7\} = 0$. In this case both players

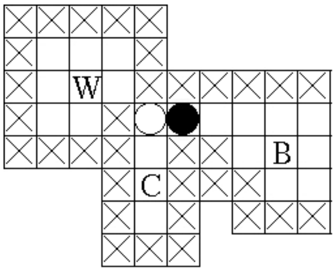


Figure 16. Another example of zugzwang.

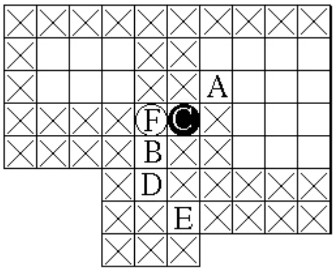


Figure 17. No zugzwang, hot position.

prefer to retreat if they get into zugzwang. If one player's territory is smaller, or if the bottom area is bigger, then it would become better to trade. An example is $w = 8, b = 5$ and $c = 3$, where Black's trading option -6 dominates the retreating option -7.

Example 7. Figure 17 shows a situation that looks very similar to Figure 16, but is a hot game with value $8|7$. Black has a good move to A, shooting to B. This neutralizes the points D and E and leaves C as Black's privilege. On the other hand, White has a skilful move to B, shooting at D. Because of the path $B - C - A$ leading into the black area, Black cannot afford to move forward to F. White can later move back to F and gains.

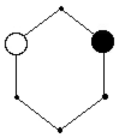


Figure 18. A zugzwang situation that leads to hot intermediate positions.

5.2. Finding Small Zugzwang Positions. We have searched our database of small Amazons positions for zugzwang situations. Candidates were identified by performing two minimax searches, one with each player going first. We analyzed the cases where moving first leads to a worse score than moving second. Some of the zugzwangs we found involve intermediate hot positions. For example, the

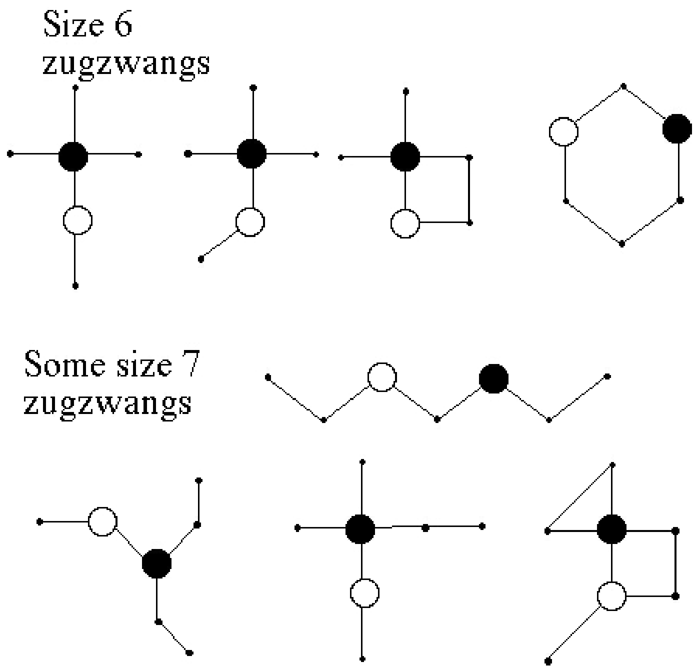


Figure 19. Some small zugzwang positions.

game value of the hexagon in Figure 18 is $0|-2||2|0 = 0$, with intermediate hot positions $0|-2$ and $2|0$. Figure 19 shows the four smallest zugzwang positions, which have size 6, and a few representative examples of the many zugzwang positions of size 7.

5.3. Playing Sums of Zugzwang Positions. By the *Number Avoidance Theorem* of combinatorial game theory [2], zugzwangs will never be played while there are any noninteger games left. Moreover, the game loser will eventually be forced to use up all the zugzwangs that provide free moves (positive integers for Black, negative integers for White). In the very end the loser will also be forced to play all the zugzwangs that evaluate to zero and collect all the penalties there.

The interesting case are the zugzwangs that are nonzero integers for the winner. From now on, let's assume without loss of generality that Black is the winner. If Black can win a sum game containing a zugzwang such as $3|6 = 4$ in Figure 15 without making the move from 4 to 3 in this subgame, then White will be forced to move to 6, giving Black extra profit. Black wants to use up as few zugzwangs as possible, and maximize the gain when White is forced to play them in the end.

Define the gain for Black in a simple zugzwang position $z = a|b = a+1, 0 \leq a < b-1$, by $gain(z) = z^R - z - 1 = b - a - 2$ (and the gain for White in a position $z = a|b = b-1, 0 \geq b > a+1$, by $gain(z) = z - z^L - 1 = b - a - 2$). Let G be a sum game won by Black, and let k be the largest integer such that

$G - k$ is still a win for Black. Let $\{z_i\}$ be the set of zugzwang positions in G that evaluate to positive integers not greater than k. Black wants to win the game while leaving the largest overall gain, so the problem is to find the subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} z_i \leq k$ and $\sum_{i \in I} gain(z_i)$ is maximized. This is exactly the well-known *knapsack problem.* This problem is NP-complete if the input is given in the form of numbers represented by their usual encoding which has logarithmic size. However, the problem can be solved in pseudopolynomial time [7] by dynamic programming if the input size is proportional to the size of the numbers. This second representation seems more appropriate here since each number n is derived from an Amazons area of size about n.

6. Programs that Play Amazons

6.1. Search. The large branching factor of Amazons in the opening and middle game strongly limits the search depth of full-width search programs. There are over 2000 possible starting moves on the 10×10 board, and although this number decreases quickly in the opening, there are several hundred legal moves during most of the game. Experiments have shown that as in most other games, search depth is strongly correlated to playing strength. Amazons seems to be a very suitable game for experimenting with selective search methods. Can deep search of a few promising move candidates be superior to shallow full-width search? Currently, the verdict is not clear. While some of the top programs are selective, others use a brute force approach, or vary their selectiveness depending on the game stage. Our programs *Arrow* and *Antiope* currently offer both options, selective and full-width search. In *Arrow*, in each position all moves are generated, executed and evaluated statically by the function explained in Section 6.2. In selective search mode only the top 10 to 15 moves in the static evaluation are further expanded in the search. A more systematic approach to selective search, the *MultiProbCut* algorithm [4], is used in Buro's *amsbot* program [3].

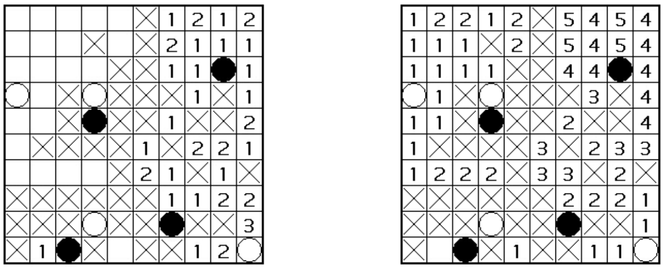


Figure 20. Min-distance function for Black and White.

6.2. Evaluation. Like most Amazons programs, our program *Arrow* uses a simple *min-distance* heuristic evaluation. Unlike most programs, *Arrow* does not use any other features in its static evaluation function. In *min-distance* evaluation, the player who can reach a square more quickly with a queen, ignoring arrow shots, owns that square. Figure 20 shows the min-distance function for both colors in a late endgame position.

The evaluation computes the following function:

1. for each point p, compare db = dist(p,Black) and dw = dist(p,White);
2. if $db < dw$: Black point;
3. else if $db > dw$: White point;
4. else point is neutral.

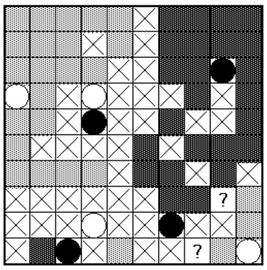


Figure 21. Evaluation using min-distance.

Figure 21 shows the evaluation of the example position. Squares marked with '?' are neutral. This algorithm is simple-minded but fast. It can be implemented efficiently by using bitmap operations. It correctly evaluates enclosed territories and regions that are somewhat open but well-controlled by one player. It also gives reasonable results in well-balanced situations such as fights with 1 vs 1 and 2 vs 2 amazons. The method fails in open and in unbalanced situations such as 1 vs 2 and 1 vs 3 amazons, since it optimistically propagates pieces in all directions at once. A single amazon that tries to defend a large territory can be badly outplayed by two or more attackers approaching from different sides.

We have tried a number of more sophisticated evaluation functions including features such as mobility and blockability. However, in test matches with equal time limits the simple but fast evaluation above always beat its more refined but slower competitors. A similar pattern became apparent in the Second Computer-Amazons championship in which *Arrow* participated and took third place. Most of the other programs used a better evaluation function including mobility terms, but searched less deeply. *Arrow* would typically get a bad position in the opening and early middle game, but use its greater search depth and specialized endgame knowledge about blockers to turn the games around later. However, against aggressive human players and programs, such as Jens Lieberum's *Amazong*, *Arrow* gets into trouble quickly. Because of the missing mobility term, it lets its amazons be blocked too easily in the opening.

7. Summary and Outlook

This paper made a number of conceptual and empirical contributions to the study of the game of Amazons. We have defined the concept of *line segment graphs*, an abstract representation that maps many equivalent Amazons positions on a real board into one graph. This representation is shown to achieve excellent compression for storing tables of exceptional *defective* territories. We have also analyzed zugzwang positions in Amazons, which tend to be a source of confusion not only for beginners. Optimal play in a sum of zugzwang positions is shown to be equivalent to solving the well-known *knapsack problem*.

7.1. Amazons Resources. Amazons is still a young game, and we are not aware of any books dealing with the game, or of any player's organizations. Some recent articles about the game are listed in the references [1; 5; 8; 10; 13]. A few opportunities exist for playing on Internet game servers or by email [6; 14]. For exchanging game records, the SGF file format has been extended to include Amazons [9]. We maintain an Amazons web site with links to all known other sites at http://www.cs.ualberta.ca/~tegos/amazons/index.html.

7.2. Future Work.

- We considered only simple zugzwangs in the form of games $a|b$ with integers a and b. However, playing into a zugzwang can lead to a hot game as in Figure 18. We found our small zugzwang positions in Figure 19 by a simple pair of minimax searches. It is conceivable that such minimax values are distorted if the follow-up positions contain further zugzwangs. It would be interesting to find such cases or prove that they cannot exist, and develop a complete theory of zugzwang in Amazons.
- Determine the asymptotic growth ratio of LSG and defective LSG.
- Determine the complexity of isomorphism testing for LSG. This could be easier than general graph isomorphism testing because of the extra structure and constraints.
- 4×4 Amazons is a second player win, and can be solved easily by brute-force search. Using a specialized endgame solver, we have recently shown that 5×5 Amazons is a first player win. The 6×6 case is much harder, and it is unclear whether the first or second player wins. Solve Amazons on 6×6 boards.

References

[1] E. Berlekamp. Sums of $N \times 2$ Amazons. In *Institute of Mathematics Statistics Lecture Notes*, number 35 in Monograph Series, pages 1–34, 2000.

[2] E. Berlekamp, J. Conway, and R. Guy. *Winning Ways*. Academic Press, London, 1982.

[3] M. Buro. Personal communication, 2000.

[4] M. Buro. Experiments with Multi-ProbCut and a new high-quality evaluation function for Othello. In J. van den Herik and H. Iida, editors, *Games in AI Research*, pages 77–96, Maastricht, 2000. Universiteit Maastricht.

[5] M. Buro. Simple Amazons endgames and their connection to Hamilton circuits in cubic subgrid graphs. NECI Technical Report #71. To appear in proceedings of Second International Conference on Computers and Games, 2000.

[6] I. Durdanovic. Generic Game Server (GGS). http://external.nj.nec.com/homepages/igord/gsa-ggs.htm, 1999.

[7] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the theory of NP-Completeness*. Freeman, San Francisco, 1979.

[8] T. Hashimoto, Y. Kajihara, H. Iida, and J. Yoshimura. An evaluation function for Amazon. In *Advances in Computer Games 9*. 2000. To appear.

[9] A. Hollosi. Smart Game Format for Amazons, 2000. Available at http://www.red-bean.com/sgf/amazons.html.

[10] H. Iida and M. Müller. Report on the second open computer-Amazons championship. *ICGA Journal*, 23(1):51–54, 2000.

[11] B. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[12] B. McKay. The nauty page, 2000. Available at http://cs.anu.edu.au:80/people/bdm/nauty/.

[13] N.Sasaki and H.Iida. Report on the first open computer-amazon championship. *ICCA Journal*, 22(1):41–44, 1999.

[14] R. Rognlie. Play by e-mail server for Amazons, 1999. Available at http://www.gamerz.net/pbmserv/amazons.html.

[15] R. G. Snatzke. Exhaustive search in the game Amazons. In this volume, 2001.

[16] H. Yamashita. Hiroshi's computer shogi and Go, 1999. http://plaza15.mbn.or.jp/~yss/index.html. Yamashita's homepage contains some Amazons information and a link to download his *Yamazon* program.

Martin Müller
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2E8
Canada
mmueller@cs.ualberta.ca

Theodore Tegos
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2E8
Canada
tegos@cs.ualberta.ca