A Pattern-Based Bayesian Network for the Game of Nine Men's Morris

D. Vrajitoru
Computer and Information Sciences
Indiana University South Bend
South Bend, IN 46617, USA

ABSTRACT

In this paper we present a game-playing algorithm for the game of Nine Men's Morris based on a small pattern database. The player is using a combination of game-specific heuristics and a pattern-based Bayesian network to make decisions about the next move. The Bayesian network is dynamically constructed and can be learned during the game play. The player we present uses a fuzzy logic approach and the pattern network to make decisions. The goal of the approach is to provide a fast search algorithm, that is adaptable and capable of learning.

1. INTRODUCTION

Building intelligent agents that can play a game and present a reasonable challenge for a human player represents an important part of game development. It is particularly significant for two player games where a second player is not always available. In this context, perfect players that can always win are not desirable unless the human player has high level proficiency. In our paper we aim to create a low resource adaptable player suitable for a mobile system.

Multiple algorithms can traditionally be found for this purpose. The classical algorithms of minmax [6] and alphabeta [16] are still in use. Game-specific heuristics are good alternatives when available. Evolutionary strategies have also been employed to develop such players [2, 16].

Bayesian networks are graphs representing joint or conditional probabilities. They have multiple applications to various problems such as classifier systems [4], knowledge representation [5], data analysis and prediction [20, 19]. They have been used in games as a way to model a network of players in game theory [15, 1]. They also offer a framework for games that can adapt to the proficiency level of the player and provide an appropriate level of challenge [18]. This is also the intended application of the system we present here. Patterns have been used for move prediction in a good number of complex games [10, 17]. They are especially popular for the games of Go [11] and Chess [12, 14].

Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015), June 22-25, 2015, Pacific Grove, CA, USA.
ISBN 978-0-9913982-4-9.

.

In this paper we present a player for the two-player board game called Nine-Men's Morris. The game was solved in [8] using a very large database of states and a traditional alphabeta search but is still considered a challenge for human players. The game was also used by the Hoyle general game learning system in [7] with positive results. More recently, in [3] the authors develop a tree-search player for this game using genetic programming. A retrograde analysis is used in [9] to develop ultra-strong solutions starting from a given game position and to avoid game loops.

The agent we present in this paper is based on a Bayesian network of game patterns obtained by dividing the board into quarters diagonally. The division uses the 4-way symmetry of the board. The patterns are organized in a graph where the edges represent possible moves with weights computed using the observed frequency of the moves. To make a new move, the agent examines the current patterns on the board and makes a choice from the set of adjacent patterns using a fuzzy scoring system.

The move selection method employed is a variation of the best-first algorithm where nodes that promise higher rewards have a higher probability to be chosen. Each node maintains information that facilitates an estimation of the reward associated with the pattern. This information is not simply equal to the implicit minmax evaluation of that node, such as in [13]. Rather, it is based both on information from the neighbors and on other game-specific measures. Our approach is somewhat similar to [8] in the fact that we are also using a database of states. However, our database is significantly smaller, which makes it more feasible for an online or mobile app.

2. NINE-MEN'S MORRIS

The game of Nine-Men's Morris is an old game that can be traced back to the Roman Empire and has been popular in various parts of Europe over time. Similar games and variations can also be found in Africa and Asia. It is a two-player board game with perfect information involving a finite number of positions and 9 tokens or stones for each player.

It consists of a board or table containing three concentric squares connected as shown in Figure 1, where two alternating players can place tokens. Each of them initially has 9 tokens of colors red and black respectively. The first stage of the game, or *placement* stage, consists in the player placing the tokens on the board, one at a time in alternation. After that the game enters a *moving* stage where players move the tokens on the table by one spot on connecting lines on the

board. The goal is to make horizontal or vertical lines of 3 adjacent tokens of the same color, or *mills*. After closing a mill, the player can remove one token from the opponent, except for those that are part of a closed mill. The player removing all the tokens of the opponent wins the game.

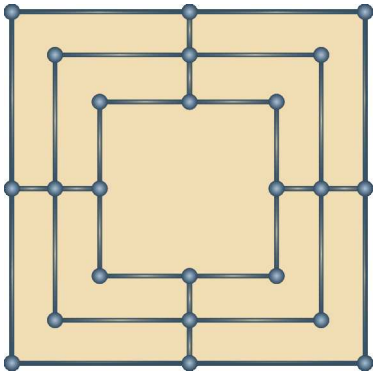

Figure 1: Table configuration for the game

Knowledgeable human players develop strategies that help with winning the game. For example, Figure 2 shows two such strategic token placements. On the left we have a strategy where two tokens not directly in line allow the placement of a third one, in line with them both, making it possible for the player to complete two mills, vertical and horizontal. The opponent can block one of the mills in the next move, but not both. On the right we have a state with a closed mill and two pieces below it that are one move away from a mill. Here the player can move the middle token between the two mills and capture one of the opponent's tokens at every move, while the configuration persists.

Figure 2: Two strategic configurations

3. GAME-SPECIFIC HEURISTICS

We developed 3 heuristics for playing the game based on simple rules considering the current configuration of the board: *Newbie*, *Basic*, and *Focused*. They mimic simple strategies commonly developed by human players.

Each of these players must have an algorithm for the *placement* phase and one for the *moving* phase. In the first one, players take turns placing their 9 tokens on the board. In the second phase, the players must move tokens during their respective turns, or pass their turn if a move is not possible. However, they are not allowed to pass the turn if they are able to move at least one token. Capturing opponent tokens by closing mills is possible in both phases.

The *Newbie* strategy consists of attempting to close a mill in both phases whether by the placement of a new token, or by moving a token. We call this a *closing move*. Otherwise if no such move is available, this player will simply choose a random valid move. This mimics a human player who has just been explained the mechanics of the game.

The *Basic* strategy is identical to the Newbie one in the placement phase. In the moving phase, if a closing move is not available, it will attempt to perform an *opening move* instead. Such a move breaks an existing mill so that it can be closed in the next turn to capture a token. If neither of these moves is available, this player will also fall back to a random choice. This mimics a human player with some experience but without more sophisticated strategies.

The *Focused* player places a new token by trying a closing move first. If no such move is available, it will look for a position that prevents the opponent from closing a mill, or a *blocking move*. If neither of these moves is available, then a random free position on the board is chosen. In the moving phase this player is identical to the Basic one. Even though the heuristic is fairly simple, it can prove enough of a challenge to a human player who isn't very careful with every move.

Starting	Black Won	Red Won	Draw
	Basic	Newbie	
Alternating	53	47	0
Black	**62**	38	0
Red	42	**56**	2
	Focused	Basic	
Alternating	**86**	14	0
Black	**90**	10	0
Red	**84**	15	1

Table 1: Heuristics comparisons over 100 matches played

Table 1 compares these 3 strategies in 100 matches, both in conditions where each player has the first move in the game, and where the first player alternates. From this table we can see that the Basic player is a little more efficient than the Newbie one, while the Focused player is a lot more efficient than the Basic one. This allows us to use the three heuristics as a level-based benchmark for measuring the performance of the Bayesian player. We show in bold the results that are significantly better using a binomial test of significance with p=0.5.

4. THE BAYESIAN NETWORK

Generally, a Bayesian network is a weighted graph where the weights of the edges represent joint or conditional probabilities of the vertices. In our case, the vertices represent partial board states. The directed edges represent possible moves in the game and the weight of each edge reflects the probability of the move occurring. Given the symmetrical nature of our board, we decided to store *patterns* on the board instead of the entire states of the table. For this, we divide the board in four regions diagonally. These quadrants can be rotated to obtain the same basic configuration. Thus, each state of the board can be described by a set of eight patterns, four for each player.

Thus, a pattern is such a quadrant with a number of tokens of a given color on it between 0 and the maximum of 9. A quadrant has 9 positions that can be either occupied or not by a token. To store a pattern, we convert it into a binary representation with one bit per location: 0 for a free location, 1 for a present token. In decimal, this gives us a number between 0 and $2^9 - 1 = 511$. For example, the pattern in Figure 2 right is converted to binary (top-down, left to right) as 000 111 101 = 61.

A set of properties is stored for each pattern, such as the

number of times it has been encountered during the recorded matches (or frequency) and a score indicating how good of a state it represents for the current player. This is computed as a combination of several values, as follows:

- *millScore*: This score value counts the lines of 3 tokens or mills that are present in that pattern. A pattern can contain at most 4 mills.
- *tokenScore*: This score value is the simple count of tokens in the pattern.
- $freqScore$: A third score is the number of times that the pattern has been encountered in the games used for training divided by the maximum frequency of any node in the network. This way the value is between 0 and 1 and does not grow unreasonably over time.
- *mutualScore*: The last score value is computed as a weighted average of the scores of the neighbors using the weights of the edges multiplied by a scaling factor called $mutualWeight$ (such as 0.2). This way, a pattern that has a particularly good score can also improve the scores of the neighbors. The mutual score is updated in a single passage at the end of every match that is used for learning. The $mutualWeight$ parameter, which is kept less than 1, has the effect of diminishing the mutual effect of patterns with the distance.

The pattern's score is computed as

$$score = (w_L \cdot millScore + w_T \cdot tokenScore + w_F \cdot freqScore + w_M \cdot mutualScore)^k$$

where the values w_L, w_T, w_F, and w_M are configurable weights used to calibrate the specialized scores. We have started with the values $w_L = 10$, $w_T = 1/3$, $w_F = 3$, and $w_M = 1$ in this research, chosen to bring the difference scores to similar ranges. We raise the total score to the power k to increase the difference between good moves and poor ones in a controllable way. We experimented with values between 1 and 4 for the parameter k.

There are situations where a move can take a token from a quadrant to another. In this case, only the old and new pattern in each quadrant separately are connected by an edge. The recorded old and new positions of the token are, in each situation, relative to the quadrant that the move is recorded for.

5. FUZZY PLAYER

In this section we present a player algorithm that uses the Bayesian network described in the previous section, called the *Fuzzy* player. It is using ideas taken from fuzzy logic to make move decisions.

As a general rule, the player starts by identifying all the available moves and assigns them an initial score of 1. Then the player identifies the 4 patterns on the table and retrieves their move information from the Bayesian network. Then for each possible move recorded in the network, if the computed score is higher than the currently assigned one, then the move's score is updated with this new value.

After this, a probabilistic choice is made, giving each possible move a chance to be chosen proportionate to the score. This way, the player is more likely to make a move that has a higher score, but the choice is not deterministic and predictable. The system also has a chance to discover a good move by accident.

Thus, in the *placement* phase, the Fuzzy player starts by assigning a score of 1 to each free position on the table, which it then updates based on the Bayesian score, if available. After this, it makes a probabilistic score-based choice. The *movement* phase is similar, except that it based on positions where the existing tokens can be moved.

To improve the player's performance, we also consider that some moves change the pattern in more than one quadrant. Figure 3 shows an example of a *corner move* where two quadrants are affected. The move is registered in the left quadrant, but the change affects the top quadrant more significantly, closing a mill. Thus, the move would get a better score were we to assign it the score of the top pattern.

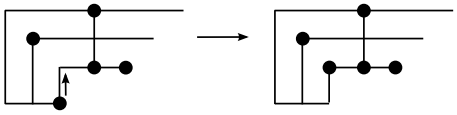

Figure 3: A move affecting the patterns in two quadrants

To implement this feature, for each quadrant, we look at all the moves that add a corner token to the quadrant. Then we check if any related move is available in the neighboring quadrant. If that is the case and if the score of the new pattern is better than the current score for that move, we update the score with the larger one.

Alpha-beta or minmax algorithms take opponent's moves into account naturally. To implement a similar feature in our Fuzzy player, we introduced a *defense mechanism*. After scoring its own moves, the player also scans for all the opponent's moves. If any of them could be blocked by one of its moves, and the opponent's move has a higher score, we update the player's blocking move with this score. To avoid over-focus on opponent's moves, we introduced a threshold $rateThr$ for the rate between the scores of the new and the old patterns. Only moves where the rate is higher than $rateThr$ are considered.

6. EXPERIMENTS

We trained the network initially in 430 games. The first 400 were composed of 100 games of each heuristic against itself, and 100 of Focused versus Basic. We intentionally trained the player with the better heuristics more times than with the simplest one. The next 30 games were played by an experienced human against each of the 3 heuristics. The human player won in all the games but one played against the Focused player.

Initial results suggest that values of 3 and 4 for the parameter k give better results, combined with $rateThr = 2$. Table 2 shows results of 100 games of the Fuzzy player against all 3 heuristics where the players alternate for the starting move. The Fuzzy player can win against the Newbie more than half of the time, and more than 40% of the time against the others. Significant differences are in bold.

To examine the influence of the $mutualWeight$ factor used in updating the mutual scores of the nodes, we performed an experiment with different values for this parameter. To level the starting point, we have reset the network scores

k	Black Won	Red Won	Draw
	Fuzzy	Newbie	
3	55	45	0
4	54	46	0
	Fuzzy	Basic	
3	39	**60**	1
4	44	55	1
	Fuzzy	Focused	
3	35	**65**	0
4	42	**58**	0

Table 2: Fuzzy Player with $rateThr = 2$, 100 games where the starting player alternates

k	$mutualWeight$	Black Won	Red Won	Draw
		Fuzzy	Newbie	
3	0.1	48	52	0
3	0.2	42	**58**	0
3	0.3	53	47	0
4	0.1	**57**	42	1
4	0.2	**56**	44	0
4	0.3	45	55	0
		Fuzzy	Basic	
3	0.1	48	52	0
3	0.2	49	50	1
3	0.3	45	54	1
4	0.1	49	50	1
4	0.2	**63**	37	0
4	0.3	44	**56**	0
		Fuzzy	Focused	
3	0.1	32	**68**	0
3	0.2	31	**69**	0
3	0.3	28	**72**	0
4	0.1	33	**67**	0
4	0.2	46	54	0
4	0.3	35	**64**	1

Table 3: Influence of the $mutualWeight$ parameter in 100 games where the starting player alternates

and updated them in two passes at the beginning of each experiment, to consider both immediate neighbors and nodes further away. Table 3 shows the result of these experiments with $k = 3$ or 4, $rateThr = 2$, and $moveThr = 5$.

Table 3 seems to indicate that for $k = 4$ the value of 0.2 that we have chosen is consistently better than the values 0.1 or 0.3, while for $k = 3$ the value 0.1 is a little bit better. The best performance in this table against the Fuzzy player was achieved with $k = 4$ and $mutualWeight = 0.2$ where the Fuzzy player won 46% of the games.

The next experiment, shown in Table 4, examines the best value for the parameter w_T, or the weight of the number of tokens in the pattern over the total score. Here we used $mutualWeight = 2$, $rateThr = 2$, $k = 4$, and $moveThr = 5$. This table suggests that values of 0.33 or 0.4 can be expected to perform better.

The last experiment, shown in Table 5, examines the best value for the parameter w_L, or the weight of the number of mills in the score. Here we used $mutualWeight = 2$, $rateThr = 2$, $k = 4$, $w_T = 0.33$, and $moveThr = 5$. This table suggests that values between 10 and 15 can be expected to perform better, the latter providing the largest number of wins against the Focused player (48%).

w_T	Black Won	Red Won	Draw
	Fuzzy	Newbie	
0.25	52	48	0
0.33	56	40	0
0.4	62	21	0
0.5	56	43	1
	Fuzzy	Basic	
0.25	47	52	1
0.33	63	37	0
0.4	39	60	1
0.5	46	53	1
	Fuzzy	Focused	
0.25	42	58	0
0.33	46	54	0
0.4	44	55	1
0.5	32	68	0

Table 4: Influence of the w_T parameter in 100 games where the starting player alternates

w_L	Black Won	Red Won	Draw
	Fuzzy	Newbie	
5	46	54	
10	56	40	0
15	54	46	0
20	64	36	0
	Fuzzy	Basic	
5	47	53	0
10	63	37	0
15	58	42	0
20	54	46	0
	Fuzzy	Focused	
5	39	61	0
10	46	54	0
15	48	52	0
20	42	58	0

Table 5: Influence of the w_T parameter in 100 games where the starting player alternates

7. CONCLUSION

In this paper we presented an adaptable intelligent agent for the two-player board game Nine Men's Morris.

We compared this agent with three game-specific heuristics of various difficulty level. Experiments showed that by varying the values of the parameters involved in the Fuzzy player, its performance against the heuristics can be dramatically improved. With the optimal values of these parameters in our tests, the player defeats the lower level and intermediate level heuristics the majority of the time, and can defeat the high level heuristic about half of the time. This last heuristic is challenging even for an experienced human player.

Our experiments show that game-specific heuristics are difficult to beat, but they are not easy to generalize to other board games. The Fuzzy player is more general purpose, but one has to find the right combination of parameters for it to be efficient.

This feature of the player is interesting for practical applications. Thus, when a human player is only starting to learn the game, the Fuzzy player can start with settings that are easy to beat. As the level of the user progresses, the settings can be adapted to present more of a challenge.

References

[1] Itai Ashlagi, Dov Monderer, and Moshe Tennenholtz. Routing games with an unknown set of active players. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 195:1–195:3, New York, NY, USA, 2007. ACM.

[2] Amit Benbassat and Moshe Sipper. Evolving board-game players with genetic programming. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '11, pages 739–742, New York, NY, USA, 2011. ACM.

[3] Amit Benbassat and Moshe Sipper. Evolving players that use selective game-tree search with genetic programming. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 631–632, New York, NY, USA, 2012. ACM.

[4] Concha Bielza and Pedro Larrañaga. Discrete Bayesian network classifiers: A survey. *ACM Comput. Surv.*, 47(1):5:1–5:43, July 2014.

[5] Norazwin Buang, Nianjun Liu, Terry Caelli, Rob Lesslie, and Michael J. Hill. Discover knowledge from distribution maps using Bayesian networks. In *Proceedings of the Fifth Australasian Conference on Data Mining and Analystics - Volume 61*, AusDM '06, pages 69–74, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[6] Yang Cai and Constantinos Daskalakis. On Minmax theorems for multiplayer games. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 217–234. SIAM, 2011.

[7] Susan Epstein. Identifying the right reasons: Learning to filter decision makers. In *Proceedings of the AAAI 1994 Fall Symposium on Relevance*.

[8] R. Gasser. Solving Nine Men's Morris. *Computational Intelligence*, 12(1):24–41, 1996.

[9] Gábor E. Gévay and Gábor Danner. Calculating ultra-strong and extended solutions for Nine Men's Morris, Morabaraba, and Lasker. *CoRR*, abs/1408.0032, 2014.

[10] T. Graepel, M. Goutrié, M. Krüger, and R. Herbrich. Learning on graphs in the game of Go. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN '01)*, pages 347–352, London, UK, 2001. Springer-Verlag.

[11] Tobias Graph and Marco Platzner. Common fate graph patterns in Monte Carlo tree search for computer Go. In *Proceeding of the IEEE Conference on Computation Intelligence in Games*, page 25, Dortmund, Germany, August 26 - 29 2014.

[12] Miroslav Kubat and Jan Žižka. Learning middle-game patterns in chess: A case study. In *Proceedings of the 13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Intelligent Problem Solving: Methodologies and Approaches*, IEA/AIE '00, pages 426–433, Secaucus, NJ, USA, 2000. Springer-Verlag New York, Inc.

[13] Marc Lanctot, Mark Winands, Tom Pepels, and Nathan Sturtevant. Monte Carlo tree search with heuristic evaluations using implicit Minimax backups. In *Proceedings of th3 IEEE Conference on Computational Intelligence in Games*, page 79, Dortmund, Germany, August 26 - 29 2014.

[14] Nicolas Lassabe, Stéphane Sanchez, Hervée Luga, and Yves Duthen. Genetically programmed strategies for chess endgame. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 831–838, New York, NY, USA, 2006. ACM.

[15] Yu Liu, Cristina Comaniciu, and Hong Man. A Bayesian game approach for intrusion detection in wireless ad hoc networks. In *Proceeding from the 2006 Workshop on Game Theory for Communications and Networks*, GameNets '06, New York, NY, USA, 2006. ACM.

[16] Jochen Mohrmann, Michael Neumann, and David Suendermann. An artificial intelligence for the board game 'Quarto!' in Java. In *Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools*, PPPJ '13, pages 141–146, New York, NY, USA, 2013. ACM.

[17] L. Ralaivola, L. Wu, and P. Baldi. SVM and pattern-enriched common fate graphs for the game of Go. *ESANN*, 2005:27–29, 2005.

[18] Nathaniel Rossol, Irene Cheng, Walter F. Bischof, and Anup Basu. A framework for adaptive training and games in virtual reality rehabilitation environments. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, pages 343–346, New York, NY, USA, 2011. ACM.

[19] Mårten Simonsson, Robert Lagerström, and Pontus Johnson. A Bayesian network for it governance performance prediction. In *Proceedings of the 10th International Conference on Electronic Commerce*, ICEC '08, pages 1:1–1:8, New York, NY, USA, 2008. ACM.

[20] Yu Zhang, Zhidong Deng, Hongshan Jiang, and Peifa Jia. Inferring gene regulatory networks from multiple data sources via a dynamic Bayesian network with structural EM. In *Proceedings of the 4th International Conference on Data Integration in the Life Sciences*, DILS'07, pages 204–214, Berlin, Heidelberg, 2007. Springer-Verlag.