9-1-2014

# Epaminondas: Exploring Combat Tactics

David W. King

Gilbert L. Peterson
*Air Force Institute of Technology*

# EPAMINONDAS: EXPLORING COMBAT TACTICS

*David W. King*[1] *and Gilbert L. Peterson*[2]

Air Force Institute of Technology, Wright-Patterson AFB, Ohio

ABSTRACT

*Epaminondas* is a two-person, zero-sum strategy game that combines long-term strategic play with highly tactical move sequences. The game has two unique features that make it stand out from other games. The first feature is the creation of *phalanxes*, which are groups of pieces that can move as a whole unit. As the number of pieces in a phalanx increases, the mobility and capturing power of the phalanx also increases. The second feature differs from many other strategy games: when a player makes a *crossing*, a winning move in the game, the second player has an opportunity to respond.

This paper presents strategies and heuristics used in a Min-Max Alpha-Beta agent that plays at a novice level. Furthermore, it defines the state-space and game-tree complexities for *Epaminondas*. Finally, a new version of MCTS is implemented that uses the Alpha-Beta heuristic function during node selection to guide MCTS to more promising areas of the search tree. Additionally, in an effort to overcome the MCTS tactical weakness, the MCTS player implements the Alpha-Beta search once the game reaches 15 turns. Results show that the added heuristic value and the switch to Alpha-Beta for endgame play, positively impact the performance of MCTS, surpassing novice Alpha-Beta win ratios at certain time intervals.

## 1. INTRODUCTION

Robert Abbott invented *Epaminondas* in the early 1970s. Abbott first published the game as *Crossings* in Sid Sackson's (1969) book, *A Gamut of Games*, alongside the more well-known game *Lines of Action*. A few years later, Abbott (2006) expanded *Crossings* from an 8 x 8 to a 12 x 14 checkered board and modified the capture rules to increase the game's complexity and to place more emphasis on flanking maneuvers. He dubbed his new variant *Epaminondas*. This paper's purpose is multifold. We mention three of them: (1) it is the first paper to perform a computational analysis in the domain of *Epaminondas*; (2) we develop two agents to play the game based on strategies and heuristics derived through human versus human, and human versus agent play (this allows us to estimate *Epaminondas*' state-space and game-tree complexities); (3) the Min-Max Alpha-Beta and MCTS agents played against one another at multiple time intervals to determine how well a heuristic-guided MCTS fairs in a highly complex tactical game. The results show that heuristic values improve MCTS play; however, its playing strength is not consistent across all time trials and may indicate a weak heuristic function. So, further modification to the MCTS selection equation is required.

It is possible that the high game-tree complexity of *Epaminondas* coupled with longer mating sequences may temper MCTS' effectiveness. Coquelin and Munos (2007) show that MCTS agents can play games with high state-space and game-tree complexities if the board transitions are smooth. *Chess* is a good counterexample of this behavior, since favorable board positions can instantly become untenable. According to Ramanujan, Sabharwal, and Selman (2010) this is due to *trap* states existing in *Chess* where a Min-Max Alpha-Beta agent is able to identify a shallow trap (a mate in three for example) while MCTS agents fail to estimate such a state correctly. Others have noted similar issues in tactical board positions in *Hex* and even in some *Go* board states (Yoshimoto *et al.*, 2006; Browne, 2013). The results of our work are promising in that there may be a way to use

[1]email:davidking.jr@gmail.com
[2]email:gilbert.peterson@afit.edu

MCTS'strategic strength while incorporating the tactical strength of Alpha-Beta to shore up MCTS' weakness in tactical games (cf. Browne, 2013; Ramanujan *et al*., 2010).

## 2. EPAMINONDAS

*Epaminondas* is a zero-sum, two-person strategy game with perfect information. It is played on a 12 x 14 checkered board with 28 black and 28 white pieces. Figure 1 shows the initial starting position. White plays first, followed by Black, and so forth. Players cannot pass. Both players have the goal to move pieces, called phalanxes, onto their opponent's back row. This is called a *crossing*. The opponent has one move to respond with their own crossing, or by capturing a crossed piece. Whoever has more pieces on their opponent's back row after one full turn, wins the game. Below we discuss the main ingredients of the game: Phalanxes and Movement (2.1), Objective (2.2), Capture (2.3), Winning (2.4), and Basic Strategies (2.5).

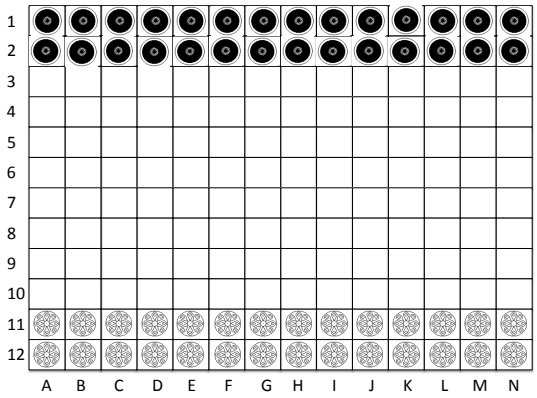



Figure 1: Epaminondas starting position.

### 2.1 Phalanxes and Movement

A phalanx is a connected group of one or more pieces. The pieces must be horizontally, vertically or diagonally inline with one another. According to Abbott (2010), the phalanxes are representative of Ancient Greek battle formations where soldiers lined up side by side, and front to back in squares, to face off against enemy armies. Phalanxes of size one, can move in any direction. They cannot move onto occupied squares. Groups of two or more phalanxes can only move in straight, orthogonal or diagonal lines (forward and backward) depending on the orientation of the phalanx. Pieces can belong to multiple phalanxes at once. The number of spaces a phalanx can move is less than or equal to the number of pieces in the phalanx. A one-piece phalanx can move one space, a two-piece phalanx can move one or two spaces, a three-piece phalanx can move one, two, or three spaces and so on. Phalanxes can split for moves as well. For example, a player can split a larger phalanx into a smaller one and move the appropriate spaces accordingly. Phalanxes cannot move through friendly or opposing pieces. Only in the case of a legal capture, a phalanx can move into an occupied square.

Figure 2 provides an example of the number of moves available to a player in one small area. This small-area position contains 30 possible moves and demonstrates the complexity of the *Epaminondas* move generation.

### 2.2 Objective

The objective of the game is to move one's pieces across the board to the opponent's back row. If, at the start of White's turn, White has more pieces on Black's back row than Black has on White's back row, White wins. The same applies at the start of Black's turn.

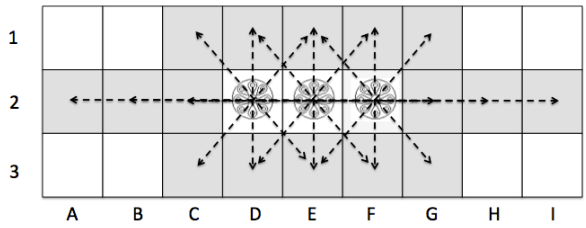



Figure 2: Example phalanx moves in gray.

### 2.3 Capture

In order to move onto an enemy-occupied square, the number of pieces in the attacking phalanx must strictly outnumber the number of pieces in the defending phalanx. A defending phalanx is defined as the number of opposing pieces that fall in the attack direction of the attacking phalanx. For example, in Figure 3, the defending phalanx consists of White's pieces at F2, G2, and H2, yielding a defending phalanx of size 3. If the attacking phalanx is of equal size or smaller, movement halts in front of the defending phalanx. Otherwise, a player may capture the opponent's entire defending phalanx. If they chose to do so, they remove the entire defending phalanx from the board and the attacking phalanx's movement halts at the space previously occupied by the lead piece of the enemy phalanx. Figure 3 shows an example of a legal capture. Note: if White had a piece at I2, then Black's movement would stop at E2.

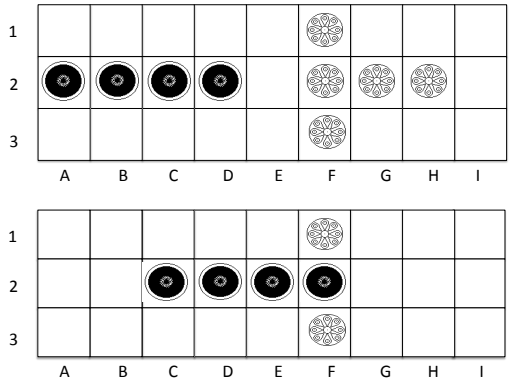



Figure 3: After capture: [E2,D2,C2,B2] x [F2,G2,H2].

### 2.4 Winning

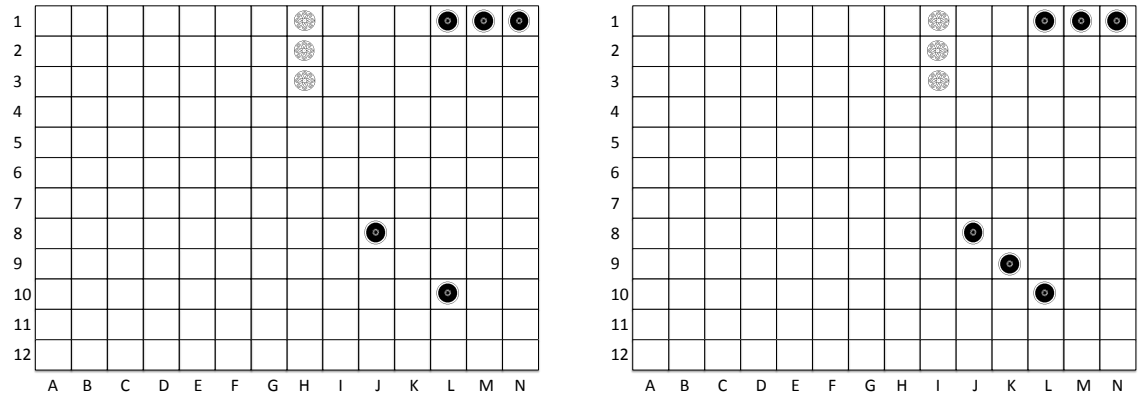
A player wins if he[3] has more pieces on the opposing home row than the opponent has on his opposing home row after one full turn. Figures 4a and Figure 4b clarify winning game conditions.

In Figure 4a, White moves onto Black's back row [H2,H3,H4]-[H1]. In this instance, Black can neither respond by capturing White's crossed piece, nor can Black complete a crossing. Black will make a move, and then, because it is White's turn, and White has more pieces on Black's back row than Black has on White's back row, White wins the game.

In Figure 4b, Black can respond by capturing White's piece [L1,M1,N1]x[I1]. Black can also move onto White's back row [L10,K9,J8]-[N12]. Either move results in an equal number of pieces on each opposing back row; zero in the former, one apiece for the latter. After Black has moved, the game would continue.

Pieces that moved onto a back row are still available for future moves; however, usually once pieces are on an

[3]For brevity, we use 'he' and 'his', whenever we mean 'he or she' and 'his or her'.




(a) White wins.

(b) Black can respond.

Figure 4: Epaminondas endgame positions.

opponent's back row they stay until captured or the game ends. One additional win condition, not covered in the article by Handscomb (2000), is exhaustion of pieces. Exhaustion of opposing pieces is a *de facto* win condition for a player. Finally, to help alleviate draws, Abbott added a rule of symmetry (see Handscomb, 2000). A player cannot move his piece onto the row furthest from the piece if it creates a pattern of left-to-right symmetry.

### 2.5 Basic Strategies

Developing strategies for a new game is both interesting and challenging. The authors relied on basic strategies for multiple two-player board games in which capture and board position are important. Winands (2000), in his work on *Lines of Action*, provides seven basic principles: threats, solid formations, mobility, blocking, centralization, material advantage, position values, and initiative. Although written specifically for *Lines of Action* some of these heuristics are applicable to *Epaminondas*. This is particularly true for threats (winning moves), solid formations (applicable to phalanxes), mobility (how many and how far pieces can move), and material advantage (having more pieces than ones opponent). After watching and playing against the *Epaminondas* agent, the authors developed five further strategies: softening, cutting, channeling, closing gaps, and sweepers.

*Softening* moves a piece in the way of an attacking phalanx. The opponent must first capture the singleton piece before trying to capture the original phalanx. This can lead the attacker into situations where the defender can bring other phalanxes into play and recapture the attacking phalanx or use the cutting strategy. Cutting involves trying to attack phalanxes at their midpoint. This reduces their mobility and attack capability. Channeling involves opening long vertical, or diagonal, paths into the opponents back row. One can easily compare this to similar strategies in *Chess* where owning a vertical row by a Rook or a diagonal by a Bishop can hinder the opponent's capabilities. While opening channels is good for attacking, thwarting one's opponent from doing the same is just as important. In closing gaps, one fills spaces created by offensive phalanxes. This builds larger phalanxes on the home row for defense which we call *sweepers*. So, sweepers is the term applied to connected phalanxes on the home row. These form a solid defensive unit for immediately capturing crossed pieces. Finally, piece domination is defined as having more pieces than an opponent, yielding weight to capture moves.

## 3. LEONIDAS – MIN-MAX ALPHA BETA AGENT

We chose to investigate *Epaminondas* over *Crossings* because of its unique sized board and move complexity. There is only one other known agent that plays *Epaminondas* but it disallows crossed pieces from moving which is incorrect (unknown, 2011). LEONIDAS is the only *Epaminondas* agent that incorporates all of Abbott's (2010) changes and plays at a novice level. In particular, LEONIDAS is an aggressive player and once that strategy is known, it becomes easy to lay traps for the agent and defeat it. It is hoped that this work will spur further *Epaminondas* research and leads to the development of more *Epaminondas* agents to play against.

## 4. EVALUATION FUNCTION

LEONIDAS uses the Min-Max Alpha-Beta algorithm with the enhancements of move ordering, the history heuristic, killer moves and transposition tables with Zobrist hashing (Knuth and Moore, 1975; Schaeffer, 1989; Zobrist, 1970). The earliest version of LEONIDAS used a rather simple capture rule to guide its play. This agent was used to refine and debug the overall logic of the program since the rules of *Epaminondas* lead to complex moves. Once the agent could play the game, we developed six distinct heuristics used by LEONIDAS to evaluate the board state. They are: mobility, material dominance, crossings, center of mass, home row defense, and territory.

### 4.1 Mobility

In *Epaminondas*, mobility is critical for both offense and defense. The mobility function averages the number of spaces that can be traversed by all the player's phalanxes. It then adds the greatest distance any of those phalanxes can travel. For example, if a player has only one piece left on the board and it can move freely in all eight directions, the state is scored as $\frac{1}{8} = 0.125$. The greatest distance that any of this player's phalanxes can travel is 1. The total mobility score for this player is $0.125 + 1 = 1.125$.

### 4.2 Material Dominance

In many games, possessing more pieces is indicative of a winning position. This function returns the difference of the sums of opposing pieces. A negative value indicates material advantage for the opponent while a positive value indicates otherwise. A value of zero means neither player has a material advantage.

### 4.3 Crossing

The object of the game is to cross to an opponent's back row. This heuristic sums the number of pieces on the opponent's back row in the current board state. The agent assigns one point for each piece. If a player has two pieces on the opponent's back row then the position is assigned two points, three pieces equals three and so on.

### 4.4 Center of Mass

In many board games, such as *Chess*, the center squares play an enabling role for winning. This is well explored in *Chess* with many openings concentrating on either controlling, or contesting, the middle of the board (cf. Gufeld and Kalinichenko, 1997). The center of mass function calculates the Euclidean distance for all pieces from the center of the board for each side. The agent subtracts these scores. A positive value indicates a higher center of mass for a player.

### 4.5 Home Row Defense

As crossings are important to winning, preventing one's opponent from making crossings is vital. One defensive maneuver is to build a large phalanx on the back row. The agent uses these phalanxes to capture crossed pieces. The function created to encode this idea calculates the largest contiguous phalanx on the home row in *Epaminondas*. For example, if a player has four pieces connected on their home row, then this function returns a value of four.

### 4.6 Territory

Owning territory is important in board games such as *Chess* and *Go*. The idea of territory in *Epaminondas* is a little more abstract since territory can be contested. In a manner similar to *Go*, the territory function looks at each piece individually and looks at the spaces around it (all eight directions). If a space is empty, or occupied by a friendly piece, the square is given a +1. Otherwise, the square is given +0. The total territory score is the

summation of all the squares surrounding the player's pieces not occupied by enemy pieces. For example, if a piece is in the middle of the board, with no enemy pieces around it, its territory score will be an 8. The opposing territory score is also calculated and then subtracted from the original player's score. A positive value indicates a strong territory score. In the future, this function should take into account contested squares, *i.e.,* squares that are contested by both sides, as well as weighing certain board squares more heavily than others.

## 5. COMPLEXITY ANALYSIS

After implementing these heuristics, LEONIDAS played itself for 1,000 games. To enable fair play, and produce tighter results, the agent randomized the first five moves for each player. This is similar to Winands' (2000) and Schadd's (2011) research where they biased the alpha-beta algorithm to produce "real" game play for their respective games. Below we discuss the state-space complexity (5.1) and the game-tree complexity (5.2).

### 5.1 State-Space Complexity

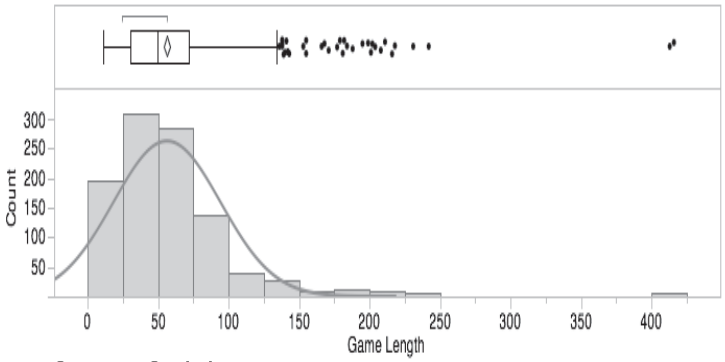
The state-space complexity of a game is defined as the number of legal positions reachable from the initial board position (Allis, 1994). We used Winands' (2000) stricter mathematical estimate of the state-space.

$$\sum_{B=1}^{maxBPieces} \sum_{W=1}^{maxWPieces} \binom{numSquares}{B} \binom{numSquares - B}{W} \tag{1}$$

where *B* equals the number of black pieces and *W* equals the number of white pieces. Winands further refines state-space estimates by eliminating positions that, while theoretically possible, are unachievable through play. The only states removed for *Epaminondas* were those in which each side possesses one piece on the board. These positions are impossible to reach through legal game play. The state-space complexity for *Epaminondas* is $2.41 \text{x} 10^{61}$, placing it above *Checkers*, *Lines of Action*, and *Chess* (Allis, 1994; **Undefined reference**; Schaeffer *et al.*, 2007; Schadd, 2011).

### 5.2 Game-Tree Complexity

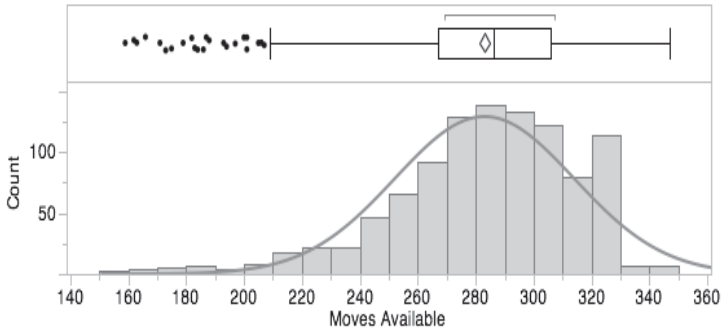
The game-tree complexity of a game is defined as "the number of leaf nodes in the solution tree of the initial position of the game," where the solution tree for a move is of full width and is of sufficient depth to determine the game-theoretic value of that move (Allis, 1994). The estimate is derived by raising the average branching factor of the game by its average game length. The average branching factor for *Epaminondas* is 283 with an average game length of 56. This yields a game-tree complexity of approximately$10^{137}$. This places *Epaminondas* above *Chess* $(10^{123})$ and below *Go* $10^{360}$ (Allis, 1994; Rijswijck, 2000). It also places *Epaminondas* squarely in the category of *unsolvable* by current methods according to the categories defined by Van den Herik, Uiterwijk, and Van Rijswijck (2002). Figures 5 and 6 show the average games length and branching factor for *Epaminondas* (Game length = 1 turn = 1 ply). Box plot diamonds represent the mean with widths showing the 95% confidence interval of the mean.




**Summary Statistics**

| | |
|---|---|
| Mean | 56.23 |
| Std Dev | 37.99 |
| Std Err Mean | 1.20 |
| Upper 95% Mean | 58.59 |
| Lower 95% Mean | 53.88 |

Figure 5: Epaminondas game lengths.




**Summary Statistics**

| | |
|---|---|
| Mean | 283.04 |
| Std Dev | 30.95 |
| Std Err Mean | 0.98 |
| Upper 95% Mean | 284.96 |
| Lower 95% Mean | 281.12 |

Figure 6: Epaminondas branching factor.

## 6. LEONIDAS (MCTS HYBRIDS) – MONTE CARLO TREE SEARCH AGENT

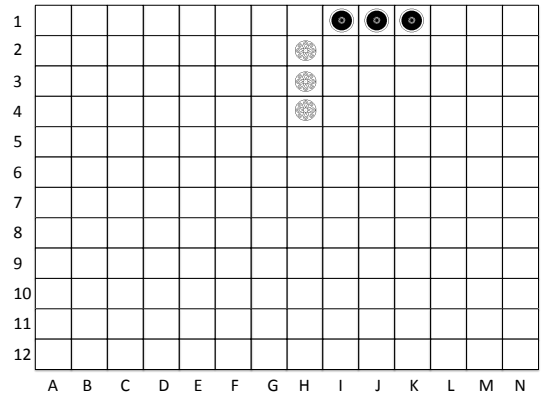
With a new tactical domain in hand, we decided to develop an MCTS-based agent. LEONIDAS (MCTS-Hybrid) is the result of the poor performance by flat MCTS agents in competitively playing the novice Alpha-Beta agent. The MCTS-Hybrid agents add the board's heuristic value to the UCB1 selection equation (Equation 2) to guide the agent to more promising areas of the search tree. After the fifteenth move the agent switches to a minimax player to overcome MCTS' weak tactical play.

$$Value = X_j + \frac{HValue(State)}{n_j} + C*\sqrt{2 * \frac{\ln(n)}{n_j}} \tag{2}$$

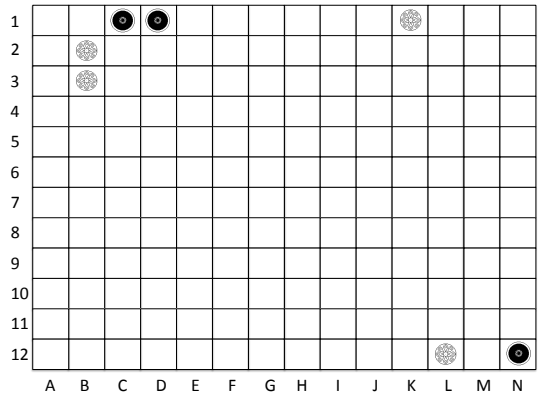
In the modified UCB1 equation, $X_j$ is the win ratio of the current state, n is the number of times the parent state has been visited, $n_j$ is the number of times the current state has been visited, and *C* is a constant between 0 and 1 where higher values and lower values adjust the amount of exploration performed by the agent (Browne *et al.*, 2012). For LEONIDAS, $C = 1$, inline with the original UCB1 formula developed by Auer, Cesa-Bianchi, and Fischer (2002). Finally, HValue(State) is the heuristic value of the current board. At the expansion, the agent makes the move and calculates the child board's heuristic value before adding it to the parent node. The heuristic

value guides MCTS towards more promising moves. As simulations play out, the agent decides between closely related board states. Once the game exceeds fifteen full turns (full turn = 1 move per side), the agent switches to the minimax algorithm. The authors chose to enhance MCTS in this manner to overcome the late tactical strength of the novice Alpha-Beta player which thwarted UCB1 versions of MCTS in the midgame even to the end game – both agents failed to score any victories against the novice Alpha-Beta player without this improvement. Finally, we implemented the All Moves As First (AMAF) (see Browne *et al*., 2012) enhancement to test its ability to improve MCTS' play. We refer to these modifed MCTS agents as UCB1-Hybrid and AMAF-Hybrid, respectively.

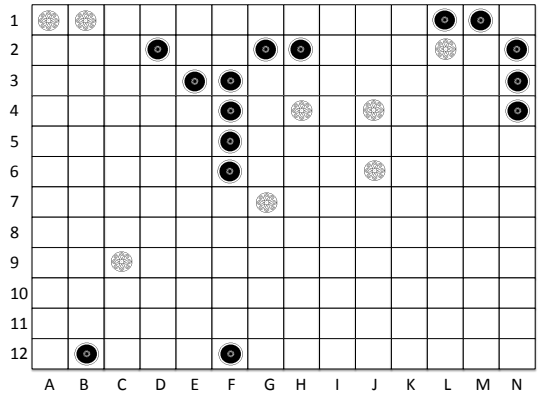
We tested the implementation of the MCTS-Hybrid agents by running them against three *Epaminondas* puzzles authored by Abbott for *Abstract Games* (cf. Handscomb, 2000). These puzzles display the strong tactical sequences contained in *Epaminondas* positions and serve as the only known examples of *Epaminondas* end-game positions. The Min-Max Alpha-Beta agent, set to depths of 7, 5, and 9, respectively, solved puzzle 1 in 9.20 s, puzzle 2 in 0.35 s, and puzzle 3 in 116.20 s. The MCTS-Hybrid agents returned winning moves at 30.00 s for puzzle 1, 5.00 s for puzzle 2, and 180.00 s for puzzle 3. Due to random play outs being used in the simulation portion of the MCTS-Hybrid agents, we set increasing time intervals until the agents returned correct winning moves at least 75% of the time. In shorter time intervals, those established by the Alpha-Beta agent for each puzzle, the MCTS-Hybrid algorithms displayed greedy move selection. For example, in Figure 7a, the MCTS-Hybrid agents moved the entire phalanx onto the back row. Although the agent makes a crossing [H3,H4,H5-H1], the move leads to a loss. Black can respond by capturing the crossed piece [I1,J1,K1xH1]. This loss of material will lead to an eventual Black win. We discuss this observed behavior further in Section 8.




(a) Puzzle 1: White to win in three.




(b) Puzzle 2: White to win in two.




(c) Puzzle 3: White to win in four.

Figure 7: Epaminondas tactical puzzles.

In Figure 7a, it is White's turn and can win in three full turns (White move + Black move = 1 full turn). The main threat is dropping the three-piece phalanx onto Blacks back row. However, doing this move first allows Black to recapture easily. Solution: White: [H3, H4, H5]-H2. Black: [I1, J1, K1]-J1. White: H2-G1. Black: [I1, J1, K1] x G1. White: [H3, H4] x H1. Black: N/A. White wins since Black cannot recapture.

In Figure 7b, White threatens another crossing onto Black's back row and again, Black has sufficient defenders to prevent this. White must force Black to move its two-piece phalanx on Row 1 in order to win. White wins in two full turns. Solution: White: [L12]-M12. This forces Black to move its two-piece phalanx. All possible positions are a loss. For example, [C1, D1]-E1, White responds with B2-A1 and Black is too far away to capture. Moving towards A1 results in a White capture and win.

In Figure 7c, White has fewer pieces than Black, however, White is threatening to make a crossing. The correct move for White is subtle but forces a win. White wins in four turns. Solution: White: J4-I3. Black: N2-N1. White: L2-K2. Black: [N1, M1, L1]-I1. White: K2-L1. Black: [N1, M1, L1] x L1. White: [I3, J4] x L1. Black: N/A. White wins since Black cannot respond by capturing White's piece or by making a crossing.

## 7. EXPERIMENT SET-UP

Monte-Carlo and Alpha-Beta agent experiments were run on three 3.1 GHz Intel Xeon Dells, running Windows 7 Enterprise Edition 6.1 using Java release 7.1 (x86). The native operating systems scheduled the game simulations without any interference or modification by the programs running the agents. The games were not restricted to any preset number of moves, allowing the agents to play full games. Time intervals were used to show the progress of the MCTS agents in case the time to make decisions increased.

LEONIDAS (UCB1-Hybrid) and LEONIDAS (AMAF-Hybrid) used a constant $C$ value set to 1 in-line with the original UCB1 algorithm analysis by Auer *et al*. (2002). Each program played against LEONIDAS (Alpha-Beta) at 1, 5, 10, 15, 30, 45, and 60-second time intervals. After node selection, game simulations were played out through random move selection with win and loss outcomes propagated back up the tree upon game completion. The Alpha-Beta player was set to a search depth of 3, using iterative deepening. If a time out occurred, the Alpha-Beta agent played the best move encountered before the search terminated. It must be noted that the Alpha-Beta agent completed depth of 3 searches in under 5 seconds. So, only under the 1 second time constraint the agent did have to end its search early.

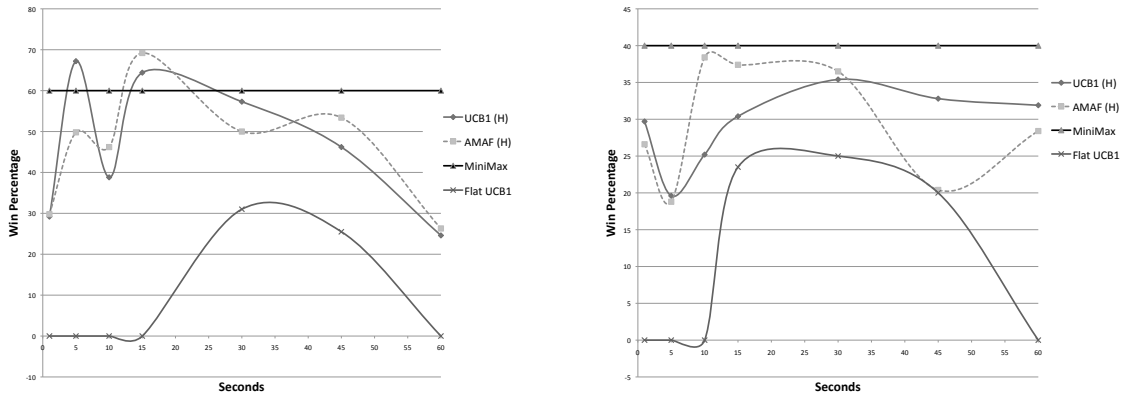
## 8. COMPARISON OF LEONIDAS AGENTS

Table 1 shows the results of the LEONIDAS Flat-UCB1 and two MCTS-Hybrid variants' performance against the novice, depth of three, LEONIDAS Alpha-Beta player. For reference, a baseline White Alpha-Beta player, set to a search depth of three, wins about 60% of its games against an equivalent Alpha-Beta player. (So, playing White is a clear advantage.) At a depth of five, the base line player plays closer to 50%. While Flat-UCB1 fails to gain any traction in the domain, each hybrid algorithm (as White) breaks the 50% win threshold at the 5-second mark, slightly dipping below it at 10, then maintaining a 50% or better win rate up to 45 seconds. The data suggests that using the heuristic value of a board state to guide move selection proves beneficial for MCTS-based agents. Additionally, the AMAF-Hybrid enhancement shows positive results when comparing the number of simulations and nodes visited. We tallied the average number of simulations and nodes visited by each agent as they played against the Alpha-Beta player for the 15, 30, 45, and 60-second time intervals (see Table 2). These totals also include the simulation rates and node counts of a Flat-UCB1 agent which failed to win any games against the Alpha-Beta player. When compared to the UCB1-Hybrid, AMAF-Hybrid produces equivalent to better results with fewer simulations and node exploration. For example, at 45 seconds, on average, UCB1-Hybrid runs over 1,200 more simulations and visits over 3,000 more nodes than AMAF-Hybrid; yet, UCB1-Hybrid wins 7.2% fewer games. A similar difference in the two algorithms appears when the agents play as Black.

As Black, Flat-UCB1 again falls behind the hybrid algorithms while AMAF-Hybrid exceeds UCB1-Hybrid's performance over the 10, 15, and 30-second time intervals. Flat-UCB1 ties AMAF-Hybrid at the 45 second mark but, overall, Flat-UCB1's performance is poor in the domain. As stated earlier, an Alpha-Beta player (as Black) set to a search depth of three attains a 40% win rate, a rate nearly achieved by AMAF-Hybrid at 10, 15, and 30-seconds. When comparing the average simulations achieved and the nodes visited, AMAF-Hybrid's performance

Table 1: LEONIDAS UCB1, UCB1-Hybrid and AMAF-Hybrid Win/Loss Percentages.

| | 1 sec | | 5 sec | | 10 sec | | 15 sec | | 30 sec | | 45 sec | | 60 sec | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Flat-UCB1 | White | Black | White | Black | White | Black | White | Black | White | Black | White | Black | White | Black |
| Win | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 23.5 | 31.0 | 25.0 | 25.5 | 20.0 | 0.0 | 0.0 |
| Loss | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 76.5 | 66.0 | 73.0 | 72.5 | 77.0 | 100.0 | 100.0 |
| Draw | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.0 | 3.0 | 2.0 | 2.0 | 3.0 | 0.0 | 0.00 |
| | 1 sec | | 5 sec | | 10 sec | | 15 sec | | 30 sec | | 45 sec | | 60 sec | |
| UCB1-Hybrid | White | Black | White | Black | White | Black | White | Black | White | Black | White | Black | White | Black |
| Win | 29.2 | 29.7 | 67.2 | 19.6 | 38.8 | 25.2 | 64.4 | 30.4 | 57.3 | 35.4 | 46.2 | 32.8 | 24.6 | 31.9 |
| Loss | 66.8 | 66.9 | 27.8 | 78.2 | 56.8 | 72.4 | 34.8 | 68.2 | 39.9 | 62.5 | 49.5 | 64.3 | 71.9 | 66.1 |
| Draw | 4.0 | 3.4 | 5.0 | 2.2 | 4.4 | 2.4 | 0.8 | 1.4 | 2.8 | 2.1 | 4.3 | 2.9 | 3.5 | 2.0 |
| | 1 sec | | 5 sec | | 10 sec | | 15 sec | | 30 sec | | 45 sec | | 60 sec | |
| AMAF-Hybrid | White | Black | White | Black | White | Black | White | Black | White | Black | White | Black | White | Black |
| Win | 29.8 | 26.6 | 49.8 | 18.8 | 46.2 | 38.4 | 69.2 | 37.4 | 50.0 | 36.5 | 53.4 | 20.4 | 26.3 | 28.4 |
| Loss | 68.4 | 71.8 | 47.2 | 78.4 | 50.2 | 59.6 | 26.1 | 60.4 | 45.4 | 61.4 | 41.8 | 77.8 | 71.0 | 65.2 |
| Draw | 1.8 | 1.6 | 3.0 | 2.8 | 3.6 | 2.0 | 4.7 | 2.2 | 4.6 | 2.1 | 4.8 | 1.8 | 2.7 | 6.4 |




(a) Playing as White.

(b) Playing as Black.

Figure 8: LEONIDAS Win Rate Comparison.

Table 2: LEONIDAS UCB1, UCB1-Hybrid and AMAF-Hybrid Win/Loss Percentages.

| Algorithm | Color | Time | Avg Simulations | Avg Nodes Visited |
|---|---|---|---|---|
| Flat-UCB1 | White | 15 | 11,519.19 | 23,224.61 |
| | | 30 | 12,064.93 | 25,596.61 |
| | | 45 | 18,234.52 | 39,881.55 |
| | | 60 | 42,936.88 | 92,357.89 |
| | Black | 15 | 5,992.32 | 11,661.93 |
| | | 30 | 11,609.99 | 23,903.67 |
| | | 45 | 17,861.88 | 38,141.51 |
| | | 60 | 44,400.29 | 93,494.07 |
| UCB1-Hybrid | White | 15 | 785.39 | 1,753.56 |
| | | 30 | 1,614.11 | 3,824.93 |
| | | 45 | 2,139.65 | 5,584.54 |
| | | 60 | 3,753.76 | 9,409.47 |
| | Black | 15 | 763.70 | 1,599.06 |
| | | 30 | 1,512.13 | 3,549.99 |
| | | 45 | 1,532.41 | 3,673.01 |
| | | 60 | 3,044.54 | 7,223.04 |
| AMAF-Hybrid | White | 15 | 616.15 | 1,344.93 |
| | | 30 | 724.05 | 1,707.14 |
| | | 45 | 939.36 | 2,338.58 |
| | | 60 | 1,169.86 | 3,281.28 |
| | Black | 15 | 637.82 | 1,312.78 |
| | | 30 | 810.34 | 1,687.33 |
| | | 45 | 1,116.83 | 2,529.24 |
| | | 60 | 1,161.14 | 2,415.65 |

is impressive. The concern for AMAF-Hybrid comes at the lower time intervals. Regardless of side, AMAF-Hybrid struggles in the 1 and 5 second intervals to overcome the performance hit of maintaining and updating encountered board states. UCB1-Hybrid outperforms AMAF-Hybrid at these intervals. Only at the 10-second point do AMAF-Hybrid results justify the performance hit. The question then becomes, why does UCB1-Hybrid lag behind AMAF-Hybrid in light of accomplishing up to twice as many simulations?

One possible explanation is that UCB1-Hybrid spends more time in bad parts of the tree. Although it runs more simulations and encounters more nodes, UCB-1 Hybrid spends time on board states it has already visited. AMAF-Hybrid, since it updates the scores of already visited nodes, starts the task of concentrating on areas that are more promising quicker than UCB1-Hybrid. UCB1-Hybrid requires more simulations to achieve the same scores. This is evident from the results, since AMAF-Hybrid does not lag too far behind the UCB1-Hybrid performance in the shorter (1 and 5-second) time intervals. However, the heuristic functions, as well as the random payouts, are possible reasons for the decline in performance encountered by both hybrid agents crossing from 30 to 60 seconds.

Figure 9 provides an example of MCTS' tactical weakness. Often, the Alpha-Beta agent will launch two pronged attacks against the opponent's back row early in the game (under the hybrids' 15-turn threshold). Most of the time, MCTS fails to find the correct move to thwart this type of attack. After White moves to I1, the MCTS agent incorrectly splits its home-row phalanx to capture White at I1. This mistake results in an eventual loss when White crosses at C1. In this situation, the correct line of play is to keep the large phalanx on the home row intact; making it capable of capturing an enemy piece that crosses onto the back row at both points. As Figure 9 shows, MCTS splits its home-row phalanx into one phalanx that is just large enough to capture the crossed piece. This tactical error leads to an Alpha-Beta win as it crosses with its second phalanx, capitalizing on its material advantage. Earlier we noted that MCTS agents can solve such tactical situations; however, they require longer periods of time to do so, placing them at a disadvantage when playing Alpha-Beta agents in short time intervals.

Finally, a noted decline in playing strength for all agents appears as they play Black. A behavior replicated throughout all of our experiments, at all time intervals, and every type of agent, including Alpha-Beta. Extending Alpha-Beta from a search depth of 3 to 7 results in closer to 50-50 play but still slightly favors White. We theorize

that this behavior could be due to a poorly refined heuristic function, which, through further modification, could result in more even play at lower search depths. The unrefined evaluation function affects the MCTS-Hybrid agent behavior similarly. As an example, the heuristic function evaluates multiple board states near equivalently to one another, and the states are often non-quiescent. The descendants frequently vacillate in evaluation as the agent plays each one out. The high tactical nature of many board states, mixed with random play outs, places the agent at a disadvantage in situations requiring Alpha-Beta's strong tactical play capability.

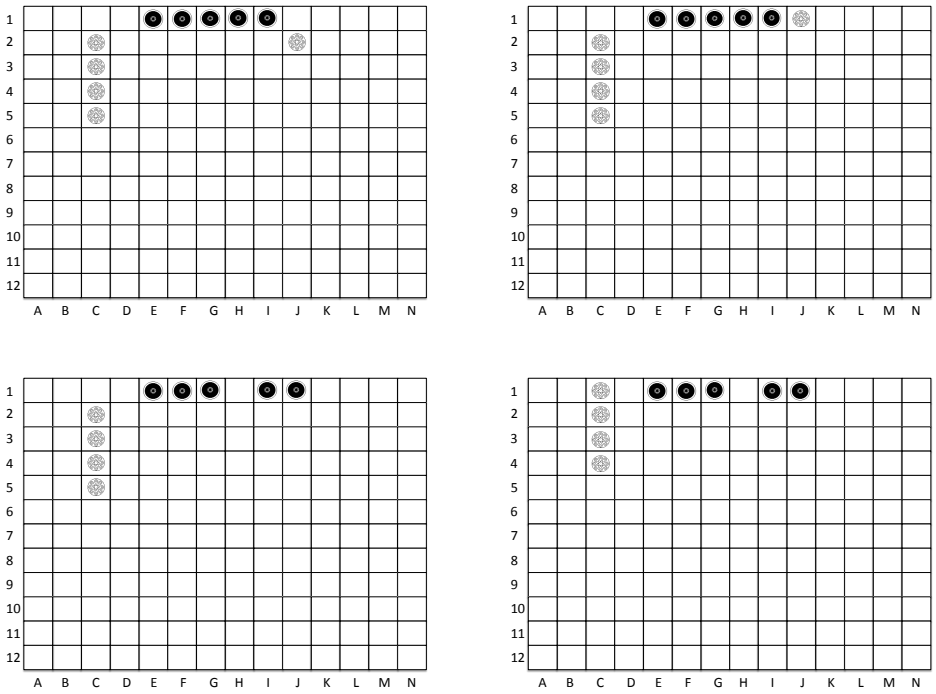


Figure 9: Two Pronged Attack by White.

## 9. CONCLUSION AND FUTURE WORK

This paper introduced the game of *Epaminondas* and four versions of the *Epaminondas* playing agent, LEONIDAS: Alpha-Beta, Flat-UCB1, UCB1-Hybrid, and AMAF-Hybrid. Basic strategies and an evaluation function were developed and implemented to play the game and derive its state-space and game-tree complexities. Results show that *Epaminondas* is more complex than *Chess* but less than *Go*. We also showed that heuristic functions positively affect the MCTS performance. Results indicate that a heuristic-driven MCTS can build strong board positions that enable higher win ratios over the Alpha-Beta player that takes over after 15 moves. Overall, our results show that MCTS agents guided by a heuristic function can achieve Alpha-Beta search performance. In the future, further refinement of the heuristic function may enable MCTS to outperform the Alpha-Beta player regardless of which side it is playing. The small gains shown by a heuristic-driven MCTS agent in a new, large tactical domain warrants further study.

*Epaminondas*' relative obscurity means that there is little expert level knowledge of the game. Future work can easily begin with refining the heuristics presented here. Additionally, *Epaminondas* may be a candidate for a heuristic derived by a TD-Learning strategy along the lines of Tesauro's (2002) work in *Backgammon*. Along these lines, the work by Baxter *et al*. (2000) with the TD-Leaf algorithm may be a good place to start by editing the weights of the evaluation functions versus trying to derive an entire heuristic from a learning agent. Other modifications to the MCTS agent, as those performed by Lorentz and Horey (2013) for *Breakthrough* may increase the MCTS' agents playing strength overcoming Alpha-Beta's tactical advantage and provides another avenue of future research. Finally, if a better heuristic function is derived, one could implement pseudo random play outs by Winands, Björnsson, and Saito (2009) to help accurately evaluate board states.

## 10. REFERENCES


Abbott, R. (2010). Epaminondas. http://www.logicmazes.com/games/epam.html.

Allis, L. V. (1994). *Searching for Solutions in Games and Artificial Intelligence.* Ph.D. thesis, University of Maastricht.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, Vol. 47, No. 2, pp. 235–256.

Browne, C. (2013). Problem Case for UCT. *IEEE Transactions on Computational Intelligence and AI Games*, Vol. 5, No. 1, pp. 69–74.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI Games*, Vol. 4, No. 1, pp. 1–43.

Coquelin, P. and Munos, R. (2007). Bandit Algorithms for Tree Search. *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pp. 67–74, Vancouver, Canada.

Gufeld, E. and Kalinichenko, N. (1997). *An Opening Repertoire For the Positional Player.* Cadogan Chess Books, Berwick Street, London.

Handscomb, K. (2000). Epaminondas . . . a game of classical elegance. http://www.logicmazes.com/games/epam/index.html.

Herik, H. J. van den, Uiterwijk, J. W., and Rijswijck, J. van (2002). Games solved: Now and in the future. *Artificial Intelligence*, Vol. 134, pp. 277–311.

Knuth, D. E. and Moore, R. W. (1975). An Analysis of Alpha-Beta Pruning. *Artificial Intelligence*, Vol. 6, pp. 293–326.

Ramanujan, R., Sabharwal, A., and Selman, B. (2010). On Adversarial Search Spaces and Sampling-Based Planning. *International Conference on Automated Planning and Scheduling.*

Rijswijck, J. v. (2000). Computer Hex: Are bees better than fruitflies? Master's thesis, University of Alberta, Edmonton, Alberta.

Schadd, M. (2011). *Selective Search in Games of Different Complexity.* Ph.D. thesis, University of Maastricht.

Schaeffer, J. (1989). History Heuristic and Alpha-Beta Search Enhancements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. II, No. II, pp. 1203–1212.

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Muller, M., Lake, R., Lu, P., and Sutphen, S. (2007). Checkers is Solved. *Science*, Vol. 317, No. 5844, pp. 1518–1522.

Tesauro, G. (2002). Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, Vol. 134 (2002), pp. 181–199.

unknown (2011). lgfip: The strategic alternative for online games. http://www.igfip.com.

Winands, M. H. (2000). *Analysis and Implementation of Lines of Action.* Ph.D. thesis, Maastricht University.

Winands, M. H., Björnsson, Y., and Saito, J. T. (2009). Evalution Function Based Monte-Carlo LOA. *12th International Conference, ACG 2009*, pp. 33–34.

Yoshimoto, H., Yoshizoe, K., Kanoeko, T., Kishimoto, A., and Taura, K. (2006). Monte Carlo has a way to go. *Proceedings of the 21st National Conference on Artificial Intelligence*, pp. 1070–1075.

Zobrist, A. L. (1970). A New Hashing Method with Application for Game Playing. Technical Report 88, The University of Wisconsin. Republished (1990) in ICGA Journal, Vol 13, No 2, pp. 69-73.