Mixing Search Strategies for Multi-Player Games

Inon Zuckerman
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel 92500
zukermi@cs.biu.ac.il

Ariel Felner
Information Systems Engineering
Deutsche Telekom Labs
Ben-Gurion University
Be'er-Sheva, Israel 85104
felner@bgu.ac.il

Sarit Kraus
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel 92500
sarit@cs.biu.ac.il

Abstract

There are two basic approaches to generalize the propagation mechanism of the two-player Minimax search algorithm to multi-player (3 or more) games: the *MaxN* algorithm and the *Paranoid* algorithm. The main shortcoming of these approaches is that their strategy is fixed. In this paper we suggest a new approach (called MP-Mix) that dynamically changes the propagation strategy based on the players' relative strengths between MaxN, Paranoid and a newly presented *offensive* strategy. In addition, we introduce the *Opponent Impact* factor for multi-player games, which measures the players' ability to impact their opponents' score, and show its relation to the relative performance of our new MP-Mix strategy. Experimental results show that MP-Mix outperforms all other approaches under most circumstances.

1 Introduction and Background

From the early days of Artificial Intelligence research, game playing has been one of the prominent directions of research, since outplaying a human player has been viewed as a prime example of an intelligent behavior.

The main building block of game playing engines is the adversarial search algorithm, which defines a search strategy for the next action selection. When a player needs to select an action, he spans a search tree where nodes correspond to states of the game, edges correspond to moves and the root of the tree corresponds to the current location. We refer to the player whose turn it is to move as the *root player*.

In two-player, zero-sum sequential turn taking games, values from the leaves are propagated according to the minimax principle. That is, in levels of the root player, it takes the maximum among the children while in levels of the opponent, it takes the minimum among the children. A multi-player game with $n > 2$ players, where the players take turns in a round robin fashion, is more complicated. The assumption is that for each node the evaluation function returns a vector H of n values where h_i estimates the merit of player i. In multi-player games, two search strategies were suggested to propagate values from H: *MaxN* [Luckhart and Irani, 1986] and Paranoid [Sturtevant and Korf, 2000].

The straightforward generalization of the two-player minimax algorithm to the multi-player case is the MaxN algorithm. It assumes that each player will try to maximize its own heuristic value (in the heuristic vector), while disregarding the values of other players. That is, when it is player i's turn the best h_i value of all children is propagated. Minimax can be seen as a specific instance of MaxN, where $n = 2$.

A different approach is the *Paranoid* algorithm where the root player assumes that the opponent players will work in a coalition against it and will try to minimize its heuristic value. The strategy is that when it is player i's turn, it will select the action with the lowest score for the root player (and not the action with the highest score for player i as in MaxN). In two players zero sum games these two approaches converge, because what's best for one player is worst for the other. The Paranoid strategy allows the root player to reduce the game to a two-player game: the root player (me) against a meta player which will include all the other players (them). The reduction gives Paranoid a technical advantage over MaxN since it can apply the *deep pruning* (i.e., full alpha-beta punning) technique. Korf [1991] found that the most significant part of the alpha-beta pruning procedure for two player games (called deep pruning) cannot be generalized to MaxN with 3 or more players. Thus, Paranoid may visit a smaller number of nodes for a given depth of the search.

As seen in [Sturtevant, 2002] there is no definite answer of which strategy is better, as the answer is probably domain dependent. Nevertheless, both algorithms are **fixed** in the way they propagate values throughout the game. However, neither of these assumptions is reasonable for the entire duration of the game. There are situations where it is more appropriate to follow the MaxN strategy, while on other occasions the Paranoid strategy might seem to be the appropriate one.

In this paper we focus on multi-player games where there is a single winner and no reward is given to the losers (they are all equal losers regardless of their relative position). We call these *single-winner* games. Assume that all players play according to the MaxN strategy and consider a situation where one player becomes stronger than the others and advances towards a winning state. The understanding that there is no difference whether a losing player ends up second or last (as only the winner is rewarded) should trigger a losing player to take explicit actions to prevent the leader from winning, even if the actions temporarily worsen its own situation. This form of reasoning should lead to a dynamic change in the search strategy to our newly suggested *offensive* strategy, in which a non leading player selects the actions that worsen the situation of the leading player. At the same time, the leading

player also understands the situation and might switch to a more *defensive* strategy and use the Paranoid strategy, as its underlying assumption does reflect the real game situation.

We therefore introduce the *MaxN-Paranoid mixture* (MP-Mix) algorithm which is a multi-player adversarial search algorithm that switches search strategies dynamically according to the game situation. The MP-Mix algorithm examines the current situation and decides, whether the player should propagate values from the leaves of the game tree to the root according to the MaxN, Paranoid, or the newly presented Directed Offensive strategy.

To evaluate the algorithm we implemented MP-Mix in two single-winner multi-player games: the *Hearts* card game, and the *Risk* strategy board game. We conducted extensive experiments and the results show that the MP-Mix's winning rate is higher in most settings. In addition, we introduce the *Opponent Impact* factor (OI) which is a game specific property describing the scope of impact a single player has on the performance and score of other players. We measure the OI values experimentally and discuss its influence and relation on the performance of MP-Mix.

2 The Directed Offensive Search Strategy

Before discussing the MP-Mix algorithm we first introduce a new propagation strategy called the Directed Offensive strategy (denoted *offensive*) which complements the Paranoid strategy in an offensive manner. In this new strategy the root player first chooses a *target* opponent it wishes to attack. It then explicitly selects the path which results in the lowest evaluation score for the target opponent. Therefore, while traversing the search tree the root player assumes that the opponents are trying to maximize their own utility (just as they do in the MaxN algorithm), but on its own tree levels it selects the lowest value for the target opponent. Our new *offensive* strategy actually uses the Paranoid assumption but in an offensive manner and complements the *defensive* Paranoid strategy suggested by [Sturtevant, 2002]. In fact, a defensive Paranoid behavior is reasonable only if there are indeed reasons to believe that others will try to attack the root player. Another attempt to complement the Paranoid behavior was done in [Lorenz and Tscheuschner, 2006] with the coalition-mixer (*comixer*) player that examines coalitions.

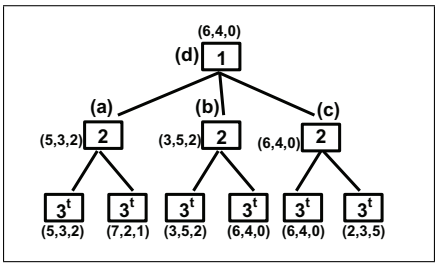

Figure 1: 3-players Offensive game tree (target = player3)

Figure 1 shows an example of a 3-player game tree, when the root player runs a directed *offensive* strategy targeted at player 3, (labeled 3^t). In this case, player 2 will select the best nodes with respect to its own evaluation. Thus, it will choose the left node in all three subtrees, a, b and c (as $3 > 2$, $5 > 4$ and $4 > 3$). Now, the root player will select node c as it contains the lowest value for player 3^t (as $0 < 2$).

3 The MP-Mixed Algorithm

The MP-Mix algorithm is a high-level decision mechanism. When it is the player's turn to move, it examines the situation and decides which propagation strategy to activate: MaxN, Offensive or Paranoid. The chosen strategy is activated and the player takes its selected move.

Algorithm 1: MP-Mix(T_d, T_o)

```
foreach i ∈ Players do
    H[i] = evaluate(i);
end
sort(H);        // decreasing order sorting
leadingEdge = H[1] − H[2];
leader = identity of player with highest score;
if (leader = root player) then
    if (leadingEdge ≥ T_d) then
        Paranoid(...);
    end
else
    if (leadingEdge ≥ T_o) then
        Offensive(...);
    end
end
MaxN(...);
```

The pseudo code for MP-Mix is presented in algorithm 1. It receives two numbers as input, T_d and T_o, which denote defensive and offensive thresholds. First, it evaluates the score value of each player ($H[i]$) via the $evaluate()$ function. Next, it computes the $leadingEdge$, which is the score difference between the two highest valued players and identifies the leading player (leader). If the root player is the leader and $leadingEdge > T_d$, it will activate the Paranoid strategy (i.e., assuming that others will want to hurt it). If someone else is leading and $leadingEdge > T_o$, it will choose to play the offensive strategy and attack the leader. Otherwise, the MaxN search strategy will be selected.

When computing the $leadingEdge$, the algorithm only considers the heuristic difference between the leader and the second player (and not the differences between all opponents). This difference provides the most important information about the game's dynamics - a point where one leading player is too strong. To justify this, consider a situation where the leading edge between the first two players is rather small, but they both lead the other opponents by a large margin. This situation does not yet require explicit offensive moves towards one of the leaders, since they can still weaken each other in their own struggle for victory, while, at the same time, the weaker players can narrow the gap.

The values T_d and T_o have a significant effect on the behavior of an MP-Mix player. These values can be estimated using machine learning algorithms, expert knowledge or simple trial-and-error procedures. Decreasing these thresholds will yield a player that is more sensitive to the game's dynamics and reacts by changing its search strategy more often. In addition, when setting $T_o = 0$ and $T_d > 0$, the player will always act offensively when it is not leading. When setting the

value in the opposite way, $T_o > 0$, $T_d = 0$ the player will always play defensive strategy when leading. When setting the thresholds to values that are higher than the maximal value of the heuristic function, we will get a pure MaxN player.

4 Experimental Results

In order to evaluate the performance of MP-Mix, we implemented players that use MaxN, Paranoid and MP-Mix algorithms in two popular games: the *Hearts* card game and the *Risk* strategic board game.[1] The offensive strategy is not reasonable as a stand alone and was only used by MP-Mix. We ran a series of experiments with different settings and environment variables in order to test the MP-Max algorithm.

We used two methods to bound the search tree. The first method was to perform a full width search up to a given depth. This provided a fair comparison to the logical behavior of the different strategies. However, since the Paranoid strategy can perform *deep pruning* we also performed experiments which limited the number of nodes visited. This provided a fair comparison to the actual performance as Paranoid can search deeper for a given number of nodes. To do this, we used *iterative deepening* to search for game trees as described by [Korf, 1991]. The player builds the search tree to increasingly larger depths, where at the end of each iteration it saves the current best move. During the iterations it keeps track of the number of nodes it visited, and if this number exceeds the node limit, it immediately stops the search and runs the current best move (which was found in the previous iteration).

4.1 Experiments Using *Hearts*

Game description

Hearts is a multi-player, partial-information, trick-taking card game designed to be played by exactly 4 players. A standard 52 card deck is used, with the cards in each suit ranking in decreasing order from Ace (highest) down to Two (lowest). At the beginning of a game the cards are distributed evenly between the players, face down. The game begins when the player holding the Two of clubs card starts the first trick. The next trick is started by the winner of the previous trick. The other players, in clockwise order, must play a card of the same suit that started the trick, if they have any. If they do not have a card of that suit, they may play any card. The player who played the highest card of the suit which started the trick, wins the trick (and starts the next trick).

Normally, each player scores penalty points for cards in the tricks they won (therefore players usually want to avoid taking tricks). Each heart card scores one point, and the queen of spades card scores 13 points (tricks which contain points are called "painted" tricks).[2] Each single game has 13 tricks and distributes 26 points among the players. Usually, the game does not end after the deck has been fully played. *Hearts* is usually played as a tournament, where the game continues until one of the players has reached or exceeded 100 points (a predefined limit) at the conclusion of a trick. The player with the lowest score is declared the winner.

[1] Rules for Risk can be found at http://www.hasbro.com/risk/ Hearts rules can be found at http://games.yahoo.com/help/he

[2] In our variation of the game we did not use the "shoot the moon" rule in order to simplify the heuristic construction process.

While there are no formal partnerships in *Hearts* it is a very interesting domain due to the specific point-taking rules. When playing *Hearts* in a tournament, players might find that their best interest is to help each other and oppose the leader. For example, when one of the players is leading by a large margin, it will be in the best interest of its opponents to give it points, as it will decrease its advantage. Similarly, when there is a weak player whose point status is close to the tournament limit, its opponents might sacrifice by taking painted tricks themselves, as a way to assure that the tournament will not end (which keeps their hopes of winning alive). This internal structure of the game calls for use of the MP-Mix algorithm.

Experiments' design

We implemented a *Hearts* playing environment and experimented with the following players:

(1) Random (RND) - This player selects the next move randomly from the set of allowable moves.

(2) Weak rational (WRT) - This player picks the lowest possible card if it is starting or following a trick, and picks the highest card if it does not need to follow suit.

(3) MaxN (MAXN) - Runs the MaxN algorithm.

(4) Paranoid (PAR) - Runs the Paranoid algorithm.

(5) MP-Mix (MIX) - Runs the MP-Mix algorithm (thresholds are given as input).

The heuristic function was manually tuned and contained the following features: the number of cards which will duck or take tricks, the number of points taken by the players, the current score in the tournament, the number of empty suits in the hand (the higher the better) and the numeric sum of the playing hand (where lower is better).

In *Hearts*, players can not view their opponent's hands. In order to deal with the imperfect nature of the game the algorithm uses a Monte-Carlo sampling based technique (adopted from [Ginsberg, 2001]) with a uniform distribution function on the cards. It randomly simulates the opponent's cards a large number of times, runs the search on each of the simulated hands and selects a card to play. The card finally played is the one that was selected the most among all simulations. The sampling technique is crucial in order to avoid naive and erroneous plays, due to improbable card distribution.

Experiment 1: Fixed setting, $T_o = \infty$, $T_d \in [0, 50]$

Our intention was to compare the performance of MIX with that of MAXN and PAR. In our first set of experiments we arbitrarily set three of the players to always be (PAR, PAR, MAXN). The fourth player was varied as follows. We first used MIX as the fourth player and varied its defensive threshold, T_d, from 0 to 50. To evaluate the advantages of a defensive play when leading, the offensive threshold, T_o, was set to ∞. We then used MAXN and PAR players as the fourth player, in order to compare their performance in the same setting. The depth of the search was set to 6 and the technical advantage of Paranoid (deep pruning) was thus neglected.

For each variation of the fourth player we ran 800 tournaments, where the limit of the tournament points was set to 100 (each tournament usually includes 7-13 games). The results in figure 2 show the difference in the tournaments' winning percentages of the fourth player and the best player among the other three fixed players. A positive value means that the fourth player was the best player as it achieved the highest

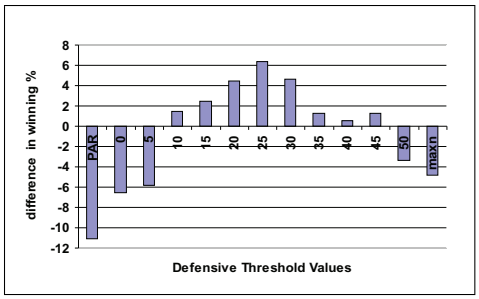

Figure 2: Experiment 1 - Difference in winning percentage

winning percentage, whereas a negative value means that it was not the player with the highest winning percentage.

The results show that PAR was the worst player (in this case a total of 3 PAR players participated in the experiment) resulting in around -11% winning less than the leader (which in this case was the MAXN player). The other extreme case is presented in the rightmost bar, where the fourth player was a MAXN player.[3] In this case it lost by a margin of only 5% less than the winner. When setting the fourth player to a MIX player and the defensive threshold at 0 and 5, it still came in second. However, when the threshold values increased to 10 or higher, the MIX player managed to attain the highest winning percentage, which increased almost linearly with the threshold. The best performance was measured when T_d was set to 25, as the MIX player performed significantly better than both MAXN and PAR players, as it attained a positive winning difference of 11% $(6-(-5))$ or 17% $(6-(-11))$, respectively ($\mathcal{P} < 0.05$).

Experiment 2: Random setting, $T_o = \infty$, $T_d = 25$

In this experiment we did not have a fixed environment but independently randomized four players for each tournament from the following set of players {RND, WRT, PAR, MAXN, MIX}. The MIX player had $T_o = \infty$, $T_d = 25$. This would result in games which are random in their players' composition and in their relative position (i.e., 5^4 for 625 possible combinations). We set the search bound to 20K nodes. Here, Paranoid can apply deep pruning and thus search deeper. We kept a record of the winner and losers of each tournament and counted the number of wins and losses of each player. We performed 1200 random tournaments.

The winning percentage of each player was computed as $\frac{wins}{wins+losses}$. [4] The MIX player led the other players with 43% winning games, the MAXN had 38%, PAR 34% , WRT 12% and RND 1% . Here again the MP-Mix strategy attained the best average performance. It is important to note that while PAR had the deep pruning advantage, it still came in last among the minimax based players.

Experiment 3: Adding the offensive strategy

In our third experiment we used the following players: {MAXN, PAR, OMIX, DMIX, MIX}. Where OMIX is an offensive oriented MP-Mix player with $T_o = 20, T_d = \infty$, DMIX is a defensive oriented MP-Mix player with $T_o = \infty, T_d = 20$ and MIX is an MP-Mix player with $T_o = 20, T_d = 20$. The environment was fixed with 3 players of the MAXN type and for the fourth player we plugged in each of the MP-Mix players described above. In addition, we changed the fixed depth limitation to a 50K node limit. Here too, the Paranoid search would be able to perform deep pruning and search deeper.

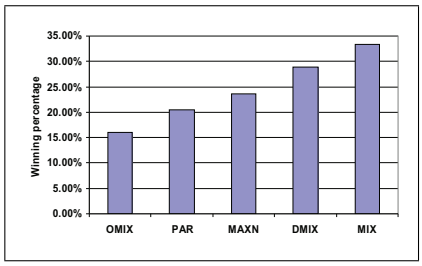

Figure 3: Experiment 3 - Winning percentage per player

The results from running 500 tournaments for each MIX player are presented in figure 3. The best player was the MIX player that won over 32% of the tournaments. The DMIX came in second with 28%, while the MAXN player in the same environment managed to win only around 23% of the tournaments. The PAR player won slightly over 20% of the tournaments. Surprisingly, the OMIX player was the worst one, winning only 16% of the tournaments. The reason for this was that the OMIX player took only offensive moves against 3 MAXN players. This was not the best option due to the fact that when it attacks the leading player it weakens its own score but at the same time the other players advance faster towards the winning state. Thus, in this situation the OMIX player sacrifices himself for the benefit of the others.

4.2 Experiments Using *Risk*

Our next experimental domain is a multilateral interaction in the form of the *Risk* board game.

Game description

The game is a full-information, strategy board game that incorporates probabilistic elements and strategic reasoning in various forms. The game is a sequential turn-based game for two to six players, which is played on a world map where each player controls an army, and the goal is to conquer the world (i.e., occupying all 42 territories is equivalent to eliminating all other players). Each turn consists of three phases: Reinforcement, Attack and Fortification.

Risk is too complicated to formalize and solve using classical search methods. First, each turn has a different number of possible actions which changes during the turn, as the player can decide at any time to cease its attack or to continue if it has territory with at least 2 troops. Second, the number of *opening* moves for all 6 players is huge ($\approx 3.3 * 10^{24}$) compared to two-player games (400 in *Chess* and $144,780$ in *Go*).

In order be able to work in this complex domain, we reduced the branching factor of the search tree to 3 by only expanding the 3 most promising moves (called the *highest bids* in [Johansson and Olsson, 2006]). Each of these moves were not a single attacking action, but a list of countries to

[3] When T_d is very large it converges to the MAX player as it will never switch the strategy. In contrast, low T_d values are closer to PAR as the switch happens more often.

[4] Note that the winning percentages did not add up to 100%, as each player played a different number of games.

conquer from the source (which the player held at the time), to a specific destination (which it wanted to conquer).

Before continuing with the technical details we would like to exemplify the intuition of the need to use MP-Mix in the *Risk* game domain. In the early stages of the *Risk* game, rational players tend to expand their borders locally, usually trying to capture a continent and increase the bonus troops they receive at each round. In more advanced stages, perhaps one player will become considerably stronger than the rest of the players (e.g. it might control 3 continents which will give it a large bonus every round). The other players, having the knowledge that there is only a single winner, might understand that unless they put some effort into attacking the leader (which might not be their best actions heuristically), it will soon be impossible for them to change the tide, and the leading player will win. In such situations, the leading player might understand that it is reasonable to assume that everybody is against him, and switch to a Paranoid play (which might yield defensive moves to guard its borders). In case the situation changes and this player is no longer a threat (as it was weakened by its opponents), it should switch its strategy again to its regular self maximization strategy, namely MaxN.

Experiments' design

We worked with the *Lux Delux* [5] environment and implemented three types of players: MAXN, PAR and MIX. Our evaluation function was based on the one described in [Johansson and Olsson, 2006].

Experiment 1: Fixed setting, $T_o = \infty$**,** $T_d \in [0, 40]$

In our first experiment we ran environments containing 6 players, 2 of each of the following types: MIX, MAXN and PAR and we used the "lux classic" map without bonus cards. In addition, the starting territories were selected at random and the initial placement of the troops was uniform.

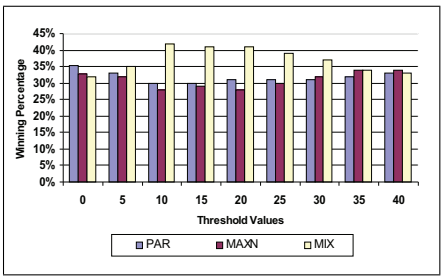

Figure 4: Risk experiment 1 - results

Figure 4 presents the results for this environment where we varied the defensive threshold value (T_d) of the MIX players from 0 to 40, while $T_o = \infty$ in order to study the impact of defensive behavior and the best value for T_d. The numbers in the figure are the average wining percentage per player type for 750 games. The peak performance of the MIX algorithm occurred with $T_d = 10$ where it won 43% of the games. In contrast PAR won 30% and MAXN won 27%. The MIX player continued to be the leading player as the threshold increased to around 30. Nonetheless, above this threshold the performances converged to that of MAXN since the high thresholds almost never resulted in Paranoid searches.

[5] http://sillysoft.net/lux/

Experiment 2: Random setting, $T_o = 10$**,** $T_d = 10$

In the second experiment we used 3 specialized expert knowledge players (not search oriented) with different difficulty levels to create a varied environment. All three players were part of the basic *Lux Delux* game package: the *Angry* player was a player under the "easy" difficulty level, the *Yakool* was considered "medium" and *EvilPixie* was a "hard" player in terms of difficulty levels. These new players, together with the search based players: PAR, MAXN, and MIX (where $T_d = 10, T_o = 10$) played a total of 750 games with the same environment setting as the first experiment.

The results show that in this setting again, the MIX player achieved the best performance, winning 27% of the games, *EvilPixie* was runner-up winning 20% of the games, followed by the MAXN and PAR players winning 19% and 17%, respectively. *Yakool* achieved 15% and *Angry* won 2%.

5 Opponent Impact

The experimental results clearly show that MP-Mix improved the players' performances. However, we can see that the improvement in the *Risk* domain is much more impressive than in the *Hearts* domain. An important question that emerged is under what conditions and game properties would the MP-Mix algorithm be more effective and advantageous? For this purpose we defined the *Opponent Impact* factor (OI), which measures the impact that a single player has on the outcome of the other players.

Definition 5.1 (Influential State) *A game state for player A with respect to player B is called an* influential state*, if action α exists such that the heuristic evaluation of B is reduced after activating α by A.*

We can now define $InfluentialStates(G, H)$ for a game G and a heuristic function H, to be a function that returns the set of influential states with respect to any two players. Similarly, $TotalStates(G, H)$ will return the set of all game states.

Definition 5.2 (Opponent Impact)
Let G be a game, H be a heuristic function, then
$OI(G, H) = |InfluentialStates(G, H)|/|TotalStates(G, H)|$

The *OI factor* of the game is defined as the percentage of *influential states* in the game with respect to all players. The intuition behind the OI is as follows. Consider the popular game, *Bingo*. In this game each player has a board filled with different numbers, and numbers are randomly selected one at a time. The first player to fill its playing board is the winner. It is easy to see that in *Bingo*, there is no way for one player to impact the heuristic score of another player. Thus, the OI of that game would be zero (as $|InfluentialStates(G, H)| = 0$).

In another game, called *GoFish*, the objective of the game is to collect "books", which are sets of four cards of the same rank, by asking other players for cards the player thinks they might have. The winner is the player who has collected the highest number of books when no cards are left in the players' hands or in the deck. Here, theoretically, at any given state the player can decide to impact a player's well being by asking him for a card. The opponent's impact value of *GoFish* is equal to 1 (as $|InfluentialStates(G, H)| = |TotalStates|$).

In addition, there are games that can be divided into two parts with respect to their OI value. For example, in

Backgammon, both players can usually hit the opponent's pawns if they are open ("blot"), yielding a game with a positive OI value. However, the final stage of a game (called "the race"), when the opponent's pawns have passed each other and have no further contact, is a zero OI game.

When trying to understand and estimate the OI values for both games we encounter the following phenomenon. When playing *Risk*, one has a direct impact on the merit of other players when they share borders, as they can directly attack one another. Sharing a border is common since, when viewing the world map as an undirected graph there are 42 nodes (territories), each with at least two edges. In contrast, in *Hearts*, a player's ability to directly hurt a specific player is considerably limited and occurs only on rare occasions.

Computing the exact value of OI is impractical in games with a large (exponential) number of states. However, we can estimate it by calculating it for a large sample of random states. In order to estimate the Opponent Impact of *Hearts* and *Risk* we did the following. Before initiating a search to select the action to play, the player iterated over all the other players as target opponents. For each move of the root player, it computed the evaluation function for the selected target opponent. We then counted the number of game states in which the root player's action could result in more than a single heuristic value for one of the target opponents. For example, consider a game state in which the root player has 5 possible actions. If the root player's actions would result in at least **two** different heuristic values for one of the target players, we would count this state as an influential one, otherwise, (all 5 actions result in the same target player's heuristic value), we would count it as a non-influential state. In both domains, we ran 100 tournaments for each search depth, and computed the OI factor by counting the percentage of influential states.

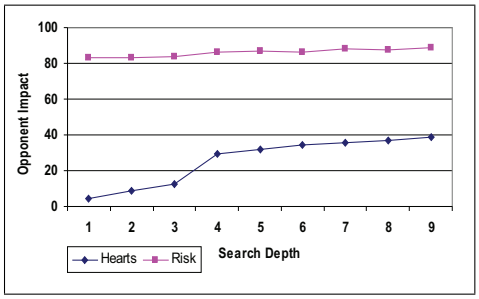

Figure 5: Opponent Impact Measure

The results in figure 5 show that the OI for *Hearts* is valued very low when the depth is lower than 4 (4% in depth 1, and 8% in depth 2). For larger depth limits the OI values monotonically increase but do not exceed 40%. The OI for the *Risk* board game starts higher than 80% (83.12% in depth 1) and it manages to climb to around 88.53% in depth 9. From these results we can conclude the following OI ordering:

$$\text{OI(Bingo)}=0 \prec \text{OI(Hearts)} \approx 0.35 \prec \text{OI(Risk)} \approx 0.85 \prec \text{OI(GoFish)}=1$$

The fact that *Risk* has a higher opponent impact factor is highly reflected in the experiment results, as the relative performance of MIX is much higher than in the *Hearts* domain. In *Risk* players have a larger number of opportunities to act against the leading player than in *Hearts*. In *Hearts* even after reasoning that there is a leading player that should be the main target for painted tricks, the number of states which one could choose as an action against the leader, is limited.

6 Conclusions

We presented the *MP-Mix* algorithm that dynamically changes its search strategy according to the game situation. MP-Mix decides before the turn begins whether to use Paranoid, MaxN or the newly presented *Directed Offensive* search strategy. We experimented in the *Hearts* and *Risk* games and demonstrated the advantages that players gain by using the *MP-Mix* algorithm. Moreover, our results suggest that the benefit of using the *MP-Mix* algorithm in *Risk* is much higher than in *Hearts*. The reason for this difference is related to a game property, which we defined as the *Opponent Impact* (OI) factor. We hypothesize that MP-Mix will perform better in games with a high OI.

In terms of future research it would be interesting to apply machine learning techniques in order to learn the optimal threshold values for different functions. In addition, more research should be performed in order to thoroughly understand the influence of the OI value on the algorithm's performance in different games. It would also be interesting to provide a comprehensive classification of various multi-player games according to their OI value.

Acknowledgments

This work was supported in part by the Israeli Science Foundation under Grants #1357/07 and #728/06 and in part by the National Science Foundation under Grant 0705587. Sarit Kraus is also affiliated with UMIACS.

References

[Ginsberg, 2001] Matthew L. Ginsberg. Gib: Imperfect information in a computationally challenging game. *JAIR*, 14:303–358, 2001.

[Johansson and Olsson, 2006] Stefan J. Johansson and Fredrik Olsson. Using multi-agent system technology in risk bots. In *AIIDE*, pages 42–47, 2006.

[Korf, 1991] Richard E. Korf. Multi-player alpha-beta pruning. *Artificial Intelligence*, 49(1):99–111, 1991.

[Lorenz and Tscheuschner, 2006] Ulf Lorenz and Tobias Tscheuschner. Player modeling, search algorithms and strategies in multi-player games. In *ACG*, pages 210–224, 2006.

[Luckhart and Irani, 1986] Carol A. Luckhart and Keki B. Irani. An algorithmic solution of n-person games. In *Proc. of AAAI-86*, pages 158–162, 1986.

[Sturtevant and Korf, 2000] Nathan R. Sturtevant and Richard E. Korf. On pruning techniques for multi-player games. In *AAAI*, pages 201–207, 2000.

[Sturtevant, 2002] Nathan R. Sturtevant. A comparison of algorithms for multi-player games. In *Computers and Games*, pages 108–122, 2002.