Modeling Social Preferences in Multi-player Games

Brandon Wilson, Inon Zuckerman, and Dana Nau
Department of Computer Science,
Institute for Advanced Computer Studies, and
Institute of Systems Research
University of Maryland
College Park, MD 20742
{bswilson,inon,nau}@cs.umd.edu

ABSTRACT

Game-tree search algorithms have contributed greatly to the success of computerized players in two-player extensive-form games. In multi-player games there has been less success, partly because of the difficulty of recognizing and reasoning about the inter-player relationships that often develop and change during human game-play. Simplifying assumptions (e.g., assuming each player selfishly aims to maximize its own payoff) have not worked very well in practice.

We describe a new algorithm for multi-player games, Socially-oriented Search (SOS), that incorporates ideas from *Social Value Orientation* theory from social psychology. We provide a theoretical study of the algorithm, and a method for recognizing and reasoning about relationships as they develop and change during a game. Our empirical evaluations of SOS in the strategic board game Quoridor show it to be significantly more effective against players with dynamic interrelationships than the current state-of-the-art algorithms.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Graph and tree search strategies*

General Terms

Economics, Algorithms

Keywords

game-tree search, multi-player games

1. INTRODUCTION

Search algorithms such as the classical Minimax search algorithm [13] are perhaps the most important component of computer programs for games of strategy. In two-person games, these algorithms have been highly successful, and there are many games in which the best computerized players perform as well or better than the best human players (e.g., [14, 15]). However, such algorithms have generally been much less successful in multi-player games—partly because they lack ways of recognizing and reasoning about inter-player relationships, which are a very influential component of the strategic behavior of human players [18]. For example, consider situations where a player has lost any "practical" chance of winning the game, but its actions might determine which of the other players will eventually win—or where a local grievance in the early stages of the game gives birth to vindictive actions in later stages. In both cases, interpersonal relationships will have considerable weight when the players reason about future courses of action.

The standard approach to dealing with this problem has been to make simplifying assumptions. For example, the Max-n algorithm [8], a generalization of Minimax to n-player games, assumes that the players will each try selfishly to maximize their own utility values, while being indifferent to the other players; and the Paranoid algorithm [16] assumes that a player's opponents will always select the actions that are worst for that player, without considering their own chances of winning.

Such simplifying assumptions rarely hold in real-world games. For example, in games such as Risk or Diplomacy, it is very common for players to decide that one player (or a small group of players) is too strong, and join forces temporarily to fight that player (a fact that was successfully exploited by the MP-Mix algorithm presented in [20]). Such interpersonal relationships can also develop when games are played repeatedly [2].

The fundamental question is ***how to describe and reason about these relationships during game play.*** Our approach is to model the players' interpersonal relationships by incorporating ideas from *Social Value Orientation* (SVO) theory into game-tree search.

Social Value Orientation (SVO) theory was first described by Messick and McClintock in [4], and [3] gives a recent overview. In this theory, the space of interpersonal behaviors is viewed as a two-dimensional spectrum in which one axis represents altruism versus aggression, and the other represents varying degrees of individualism. SVO theory states that these differences in interpersonal orientations might arise due to reasons including subjective preferences and beliefs, and ascriptive characteristics such as nationality and ethnicity (for example see Grid-Group theory [9]). These orientations often change dynamically during interactions, based on the players' actions in the previous rounds [1].

Our new algorithm, Socially Oriented Search (SOS), uses a social-range matrix structure, based on SVO theory, to keep track of inter-player relationships and use them to guide the search. Since the interpersonal values are often unknown and change dynamically during play, we also present an on-

Cite as: Modeling Social Preferences in Multi-player Games, Brandon Wilson, Inon Zuckerman, and Dana Nau, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 337-344.

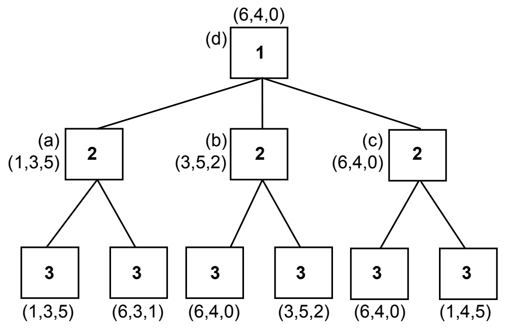

Figure 1: Propagating with the Max-n assumption

line learning technique that adapts the stored relationship values as the game progresses. In our evaluations of the algorithm in the *Quoridor* game, it played significantly better than the Max-n and Paranoid algorithms when playing against adversaries with varying interpersonal orientations.

Our main contributions can be summarized as follows:

- SOS, a novel multi-player search algorithm that considers interpersonal orientations.
- A theoretical analysis of the solution computed by SOS and conditions under which SOS converges to well-known multi-player algorithms.
- A learning version of the algorithm that learns and adapts social orientations when they are unknown.
- An empirical evaluation comparing the performance of SOS with both Max-n and Paranoid algorithms.

2. BACKGROUND AND RELATED WORK

In search algorithms for extensive-form games, when a player needs to select an action, it 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 state. We refer to this player as the *root player*. The leaves of the tree are evaluated according to a heuristic evaluation function and the values are propagated up to the root.

In sequential two-player games (where players alternate turns) values from the leaves are propagated according to the Minimax principle [13]. That is, in levels where it is the root player's turn, we take the maximum among the children while in levels where it is the opponent's turn, we take the minimum of the children.

The sequential multi-player game with n players ($n > 2$), 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. The straightforward generalization of the two-player Minimax algorithm to the multi-player case is the Max-n algorithm [8]. It assumes that each player will try to maximize its own component of the heuristic vector, while disregarding the values of other players. Minimax can be seen as a specific instance of Max-n, where $n = 2$. An example of the Max-n propagation procedure is depicted in figure 1, where we see that each player i (player numbers are inside the nodes) selects the action which maximizes element i of the vector without considering the values of the other elements.

A different approach, called the *Paranoid* approach, was first introduced in [11] as part of a proof for search pathology in multi-player games trees, and later was presented in [16]. In this algorithm the root player takes a paranoid assumption that the opponent players will work in a coalition against him and try to minimize its heuristic value. This paranoid assumption allows the root player to reduce the game to a two-player game: the root player against a meta player that includes all the other players. Figure 2 depicts the same game tree as the previous Max-n example, but now values were propagated according to the Paranoid assumption. Here, the root player assumes that players 2 and 3 will select the action that minimizes his value.

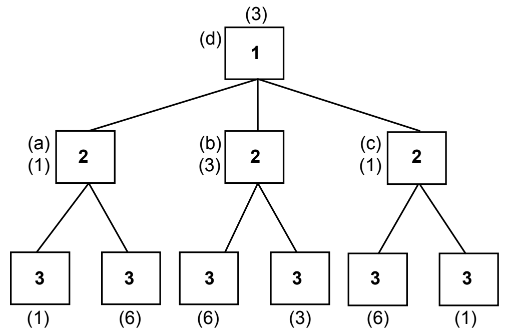

Figure 2: Propagating with Paranoid assumption

Since there is no definitive answer to which approach is better, and the answer is probably both domain and evaluation function dependent [17], a recent algorithm, *MP-Mix*, proposed switching between search strategies dynamically according to the game situation [20]. This algorithm examines the current situation and decides, according to the players' relative strength values, whether the root player should propagate values according to the Max-n principle, the Paranoid principle, or the newly presented Directed Offensive principle. In this 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.

The above algorithms share two common assumptions: the first is that when propagating values, the adversaries use the same heuristic evaluation function as the searching player, while the second states that the adversaries are using a fixed, pre-determined preference ordering on the heuristic values. The first assumption has been dealt with through specific opponent modeling techniques. For example, the *Prob-MaxN* algorithm [19] is a extension of the Max-n algorithm where, given a set of possible evaluation functions as an input, the algorithm dynamically adapts the probabilities of each individual adversary's membership to these prior models. In [10], when playing repeatedly against the same opponent, the authors' algorithm learned its opponent's weaknesses and exploited them in future games. Regarding the second assumption, these models are limited and unable to describe and reason about the more complex relationships that players might encounter in multi-player games.

3. SOCIALLY ORIENTED SEARCH

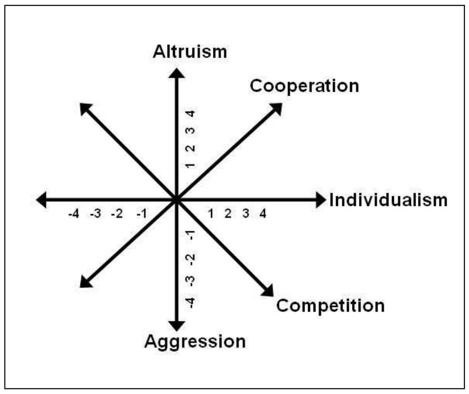

Figure 3: Social Behavior Spectrum

In this paper, we use "Social Value Orientation theory" to refer to a class of theories from the behavioral science literature [4, 3, 6] stating that people differ in their interpersonal orientations, and that they are consistent over time. Figure 3 describes a two-person preference model of the major personal and interpersonal orientations that can occur between players. In this model [6], the player's utility is defined on the horizontal axis, and the outcome of the "other" player is on the vertical axis, and the values reflect a linear combination of payoffs to both players.

Our Socially Oriented Search algorithm utilizes a recently suggested social-range matrix model [7] that supports the description of interpersonal orientations as captured in the two-dimensional social behavior spectrum. The social matrix construct makes it possible to model "socially heterogeneous" systems where players may have different social orientations to each of the other players. The fundamental building block of our search procedure, the social-range-matrix, is defined as follows: for a game consisting of n players, the social-range matrix is an $n \times n$ matrix where element $c_{ij} \in [-1, 1]$ represents how much player i cares about player j. A value of 1 indicates a completely cooperative relationship whereas -1 indicates an aggressive one. Values in between represent varying degrees of social orientation, with 0 indicating complete apathy.[1] Given a utility vector, U, with a utility for for each player, the weighted sum of the utility vector and the ith row of the social range matrix is referred to as the *perceived utility* for player i.

3.1 The SOS Algorithm

Our Socially Oriented Search (SOS) algorithm (presented as algorithm 1) models inter-player relationships with a social-range matrix and incorporates them into the search by weighting the evaluation values in the leaf nodes to obtain the *perceived utility.* Since most games are too complex to search completely, the typical approach is to pre-select a level to cutoff the search and estimate the strength of the states at this level by a heuristic evaluation function. This evaluation is a vector where element i represents the relative strength of the game state for player i. Our algorithm transforms

Algorithm 1 $SOS(s, eval, d, c, p)$: Given an evaluation function, $eval$, and a social range matrix, c, compute and return the perceived utility of state s by searching to depth d. p is the player whose turn it is to move at s and N is the number of players in the game.

```
// Transform evaluation to incorporate social preferences.
// Notice the matrix-vector multiplication.
**If** $d$ is 0, **return** c * $eval(s)$

// Determine maximum of values for children nodes.
Let $mv_1, \ldots, mv_n$ be the children of $s$ and
$bestEvaluation = -\infty$
**for** $i = 1, \ldots, n$ **do**
    $v = \mathsf{SOS}(mv_i, eval, d-1, c, p \bmod N)$
    **if** $v > bestEvaluation$ **then**
        $bestEvaluation = v$
        $bestMove = v$
    **end if**
**end for**

// Return the move with the best perceived utility
**return** $bestMove$
```

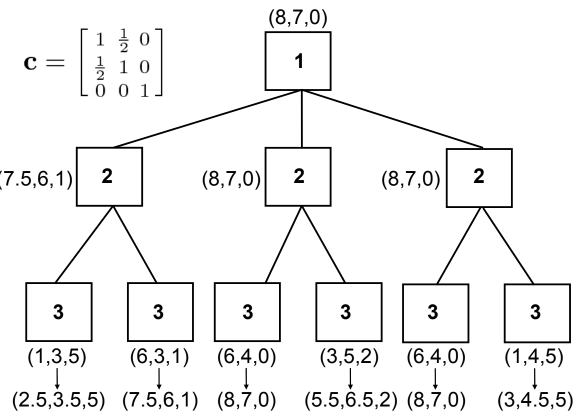

Figure 4: SOS propagation using the social-range matrix ***c*****.**

this vector into the *perceived evaluation*, where element i is the dot product of the heuristic evaluation and row i of the social range matrix. The values are then propagated so that each player maximizes their element of the perceived evaluation.

Figure 4 illustrates an example of a depth-2 search that is guided by the social-range matrix *c*. In this particular matrix, players 1 and 2 are not completely cooperative, as they do not value each other's utility as much as their own but they do have a positive orientation towards one another and they are ignorant of the utility for player 3. Player 3 is selfish, valuing its own utility and ignoring all others. Based on this matrix, the SOS algorithm selects the middle branch as the best decision, leading to a perceived utility of 8 for player 1.

The social range matrix guides an SOS search, selecting the move leading to the maximum perceived utility instead of selecting the move that maximizes or minimizes a single element in the evaluation vector (i.e., the Max-n and Paranoid algorithms). The ability to modify the social range

[1]Note that the social behaviors on the left side of the graph are deliberately excluded from our work, since these behaviors (e.g. *Masochism*, *Sado-Masochism*), are considered mental disorders.

matrix makes our algorithm more flexible than either Paranoid or Max-n. In fact, one can achieve a Max-n or Paranoid search with the appropriate social range matrix. Max-n assumes that all players are selfish and ignorant of other players utility. Property 1 characterizes the features of a social-range matrix that lead to Max-n behavior.

PROPERTY 1. *Using the identity matrix with the SOS algorithm is equivalent to Max-n.*

Paranoid assumes that all players are aggressive toward the principle player with the exception of the principle player itself which is selfish. Property 2 formally characterizes the features of a social-range matrix that lead to Max-n or Paranoid behavior.

PROPERTY 2. *A matrix in which element $c_{ii} = 1$, $c_{ji} = -1, \forall j \neq i$, and all other entries are 0 represents the Paranoid assumption from the point of view of player i. Using it with the SOS algorithm is equivalent to player i using the Paranoid algorithm.*

Coalitions, where a group of players try to maximize or minimize the combined utility of another set of players, can also be represented by social range matrices, as well as an infinite number of other social preferences.

3.2 Online learning SOS

Since the social preferences of each player are not usually known ahead of time, learning algorithms can be used to estimate the social-range matrix based on previous moves. Our learning rule estimates a player's social preferences as the average effect of the player's last k moves. We can estimate the effect of a move from state s_1 to state s_2 as the difference in the normalized heuristic evaluations:

$$\Delta(s_1, s_2) = \frac{eval(s_2)}{max\_eval} - \frac{eval(s_1)}{max\_eval},$$

where max_eval is the maximum evaluation for any player's component of the evaluation. So, given a history of k moves for player i, we estimate the ith row of the social-range matrix as the average effect of the k last moves. Higher values of k provide a richer information set to infer the social-range matrix while lower values allow the matrix to be adapted more quickly to a player whose social preferences change often over the course of the game.

4. THEORETICAL ANALYSIS

In this section we document some of the theoretical properties of the SOS algorithm. We demonstrate that SOS computes an equilibrium point, that SOS always permits immediate and shallow pruning but may also leverage deep-pruning under special circumstances, and that the behavior of SOS may converge to that of Max-n or Paranoid based on the social orientations and evaluation function.

4.1 Perceived Equilibria

For an n-player game, a player's strategy is a plan that completely determines the player's behavior for any potential situation that may arise during the course of the game. A strategy profile, $S = \{s_1, s_2, \ldots, s_n\}$, is a set of strategies, one for each player, that completely defines all actions for the game. A strategy profile is a Nash Equilibrium if there is no player that can unilaterally change their strategy and obtain a higher utility. The utility vector corresponding to this strategy profile is an equilibrium point. This concept assumes each player is completely selfish. For a game where players' social orientations may vary, we have a similar concept: the *Perceived Nash Equilibrium* [7]. A strategy profile is a perceived Nash equilibrium if there is no player that can unilaterally change their strategy to obtain a higher perceived utility. Similarly, the perceived utility associated with this strategy profile is a *perceived Nash equilibrium point.*

Assuming that the social range matrix correctly represents all players' social preferences then we can prove that our generalized search procedure identifies a *perceived equilibrium point* in a similar fashion as the proof Luckhart et al. use in showing that Max-n identifies an equilibrium [8].

THEOREM 1. *Given a perfect-information, n-player, non-cooperative game and the players' social orientations in the form of a social-range matrix, SOS computes a perceived Nash equilibrium point.*

PROOF. Let $S = \{s_1, s_2, ..., s_n\}$ denote the strategy profile computed by SOS that leads to the payoff vector produced by our algorithm. The payoff vector, where element i of the vector is the perceived utility for player i, propagated to the root of the search tree based on this set of strategies is $U(S)$. $U(S)$ is a perceived equilibrium point if there is no alternate strategy set $S' = s_1, s_2, ..., s'_j, ..., s_n$ where player j utilizes a different strategy and $U(S')[j] > U(S)[j]$. Now, assume a strategy set exists such that $U(S')[j] > U(S)[j]$, where S is the strategy set identified by our SOS. This means that there is a different set of moves that would lead to a greater perceived utility for player j. More specifically, there must be at least one node in the game tree where the move selected by j would be better if a different move were selected. This contradicts the definition of our algorithm where the move selected at each node maximizes the moving player's perceived utility. $\square$

4.2 Pruning

In general, SOS is only capable of immediate and shallow pruning, however, deep-pruning is also possible under certain circumstances. Our algorithm follows the same propagation rule as Max-n with the exception of the linear transformation that is applied to the node evaluations on the search frontier. Therefore, as long as the heuristic evaluation meets the assumptions that make such pruning possible for Max-n (i.e., the minimum evaluation for each player and the maximum sum of player evaluations is known) then we can perform immediate and shallow pruning. We will not reproduce the proofs here because, with the exception of the evaluation transformation, they are identical to those for Max-n [16].

As discussed by Sturtevant and Korf [16], the Paranoid algorithm allows for deep pruning by reducing the game to two players, the principle player and a coalition of attackers cooperating against the principle player. Therefore, in addition to immediate and shallow pruning, we are also able to take advantage of alpha-beta style pruning when the social-range matrix meets the criteria of the Paranoid matrix as presented in Property 2.

Examining the Paranoid matrix, we see that reducing the game to a two-player game really means that *only* a single column of the matrix is non-zero. In other words, if column i is the non-zero column then every player's perceived utility is oriented around either maximizing or minimizing player i's utility (referred to as cooperating and attacking players, respectively). These players would be max and min respectively on their turns in the corresponding two-player game tree, which in turn makes alpha-beta pruning possible [16]. This means that any situation where the social-range matrix has only one non-zero column can be deep-pruned using alpha-beta, not just when there is one cooperating player and $n-1$ attacking players as is the case of Paranoid.

4.3 Convergence to Max-n and Paranoid

SOS does not always produce different results than Max-n or Paranoid. For example, properties 1 and 2 mention that when the social orientations match the underlying assumptions of either algorithm then the behavior will coincide with the corresponding algorithm. This section expands this concept, using features of the evaluation function as well as information about players' social orientations to determine when the social orientations can have an effect on the algorithm performance and when they can be ignored since the performance converges to that of Max-n or Paranoid.

The identity matrix is not the only matrix that will result in identical behavior to Max-n. Theorem 2 shows that scaling the diagonal of the identity matrix by any positive, real number will also produce the same behavior as Max-n.

Theorem 2. *Given a perfect-information, n-player game and the players' social orientations in the form of a social-range matrix, c, the equilibrium strategies computed by SOS and Max-n are identical given that c is the zero matrix except for a positive, non-zero diagonal.*

Proof. Assume that SOS does select a different strategy than Max-n. The strategies discovered by SOS and Max-n will differ if there is at least one decision in the game tree where Max-n selects a state s_1 and SOS selects state s_2 on player i's turn. For Max-n to select s_1 over s_2 means that the i'th component of the evaluation for s_1 must be greater than that of the evaluation for s_2. In other words, $eval(s_1) = \{e_1, ..., e_i, ..., e_N\}$ and $eval(s_2) = \{e'_1, ..., e_i - \epsilon, ..., e'_N\}$, where ϵ is a non-zero, positive real-number such that $e_i - \epsilon$ is still a valid evaluation. Letting c_{ii} be player i's entry in the i'th row of the social-range matrix, SOS will choose s_2 and produce a different strategy set if:

$$(e_i - \epsilon)c_{ii} \geq (e_i)c_{ii}$$

which reduces to

$$c_{ii} \leq 0.$$

This contradicts the non-zero and positive diagonal. $\square$

In addition to the inter-player relationships, having knowledge of the evaluation function can also provide insights into the performance of a search algorithm. In general, an evaluation function with finer granularity (number of possible values) is better as more values means that the strength of states can be estimated more accurately. Also, Nau et al. [12] recently showed that evaluation functions with a finer granularity are less likely to exhibit pathological behavior during Minimax search. We use a granularity-based model of the evaluation function as well to analyze how the relationship between social orientations and evaluation function granularity can impact the behavior of SOS. The function we consider is of the form:

$eval\colon s \to \langle e_1, e_2, \ldots, e_n \rangle \mid \forall i, e_i \in \{0, \delta, 2\delta, \ldots, max\_eval\}$,

where max_eval is the maximum possible value for any single element of the evaluation vector, s is a state of the game, and δ represents the distance between consecutive values of the evaluations. With max_eval fixed, finer grained evaluations can be achieved by reducing δ and coarser-grained evaluations can be achieved by increasing δ.

Theorem 3. *Given a perfect-information, n-player game, the players' social orientations in the form of a social-range matrix, c, and an evaluation function,*
$eval\colon s \to \langle e_1, e_2, \ldots, e_n \rangle \mid \forall i, e_i \in \{0, \delta, 2\delta, \ldots, max\_eval\}$, *the equilibrium strategies computed by SOS are guaranteed to be identical to those of Max-n if, for each player i:*

$$\delta > \frac{\max_{e'_j} \sum_{j \neq i}^{N} c_{ij} * e'_j - \min_{e_j} \sum_{j \neq i}^{N} c_{ij} * e_j}{c_{ii}}.$$

Proof. Consider the turns in the game tree for an individual player. By definition, on player i's turn, Max-n always selects the state that maximizes the i'th component of the utility vector. This means that given that Max-n chooses s_1 over s_2 then we know their evaluations must be of the form
$eval(s_1) = \{e_1, \ldots, e_i, \ldots, e_N\}$ and $eval(s_2) = \{e'_1, \ldots, e_i - x\delta, \ldots, e'_N\}$ where x is a positive integer such that $e_i - x\delta$ is still a valid evaluation. SOS will incorporate the true social orientation of player i and make the same decision if:

$$(e_i - x\delta)c_{ii} + \sum_{j \neq i}^{N} c_{ij} * e'_j < c_{ii}e_i + \sum_{j \neq i}^{N} c_{ij} * e_j.$$

We can guarantee that this condition is true for all possible evaluation vector pairs e and e' if e' is chosen so as to maximize the summation on the left-hand side and e is chosen so as to minimize the summation on the right-hand side. This gives us:

$$(e_i - x\delta)c_{ii} + \max_{e'_j} \sum_{j \neq i}^{N} c_{ij} * e'_j < c_{ii}e_i + \min_{e_j} \sum_{j \neq i}^{N} c_{ij} * e_j.$$

Simplifying this equation we get the result displayed in Theorem 3 which must hold for all players for SOS to select the same move at every level of the search tree as Max-n. $\square$

Intuitively, the relationship presented in Theorem 3 states that with coarser evaluation functions, the social orientations can deviate further from the assumed relationship model of Max-n and yet the behavior of Max-n and SOS will converge. This is significant because the result that finer-grained evaluation functions make social orientations have greater effect further increases the appeal of modeling and using them with SOS.

Recognizing situations where the behavior of SOS will converge to that of Paranoid is also important. Given the social-range matrix, if we know that SOS will behave exactly as a Paranoid then the social-range matrix can be approximated by the Paranoid matrix and SOS gains the advantage of deep-pruning without sacrificing anything by ignoring the social relationships. Theorem 4 formally describes the relationship between the granularity of the evaluation function and paranoid matrix.

Theorem 4. *Given a perfect-information, n-player game, the players' social orientations in the form of a social-range matrix, c, and an evaluation function,* $eval\colon s \to \langle e_1, e_2, \ldots, e_n\rangle \mid \forall i, e_i \in \{0, \delta, 2\delta, \ldots, max\_eval\}$, *the equilibrium strategies computed by SOS are guaranteed to be the same as those of Paranoid if, for the principle player player i:*

$$\delta > \frac{\max_{e'_j} \sum_{j\neq i}^{N} c_{ij} * e'_j - \min_{e_j} \sum_{j\neq i}^{N} c_{ij} * e_j}{c_{ii}},$$

and for every other player k:

$$\delta > \frac{\max_{e_j} \sum_{j\neq i}^{N} c_{kj} * e_j - \min_{e'_j} \sum_{j\neq i}^{N} c_{kj} * e'_j}{c_{ki}}.$$

Proof. Similar to Theorem 3, we must show the conditions under which SOS is guaranteed to make the same choices as Paranoid during propagation. The justification for the principle player is identical to the one already shown in the proof of Theorem 3. The difference is that all other players are attacking and trying to minimize the principle player's score whereas in Max-n they are assumed to be maximizing their own utility. Therefore, given that player i is the principle player and that Paranoid chooses s_1 over s_2 on player k's turn then we know their evaluations must be of the form
$eval(s_1) = \{e_1, \ldots, e_i, \ldots, e_N\}$ and $eval(s_2) = \{e'_1, \ldots, e_i + x\delta, \ldots, e'_N\}$ where x is a positive integer such that $e_i + x\delta$ is still a valid evaluation. SOS will incorporate the true social orientation of player i and make the same decision if:

$$(e_i + x\delta)c_{ki} + \sum_{j\neq i}^{N} c_{kj} * e'_j > c_{ki}e_i + \sum_{j\neq i}^{N} c_{kj} * e_j.$$

Now, we can guarantee that this condition is true for all possible evaluation vector pairs e and e' if e' is chosen so as to minimize the summation on the left-hand side and e is chosen so as to maximize the summation on the right-hand side. This leaves us with:

$$(e_i + x\delta)c_{ki} + \min_{e'_j} \sum_{j\neq i}^{N} c_{kj} * e'_j > c_{ki}e_i + \max_{e_j} \sum_{j\neq i}^{N} c_{kj} * e_j.$$

This reduces to the equation in Theorem 4. □

5. EXPERIMENTAL EVALUATION

In this section we provide a discussion of the experimental analysis we performed on our algorithm. All experiments are performed on a four-player version of the game Quoridor. We show that by explicitly modeling social preferences, SOS gains a siginificant advantage over algorithms that make simplifying assumptions. We also show that our approach to learning the social-range matrix, although simple, is quite effective.

5.1 Game description

Quoridor[2] is a full-information board game for 2 or 4 players, that is played on a 9x9 grid (see figure 5). In the 4 player version, each player starts with five walls and a single pawn that is located at the middle grid location on one of the four sides of the square board. The objective is to be the first player to reach any of the grid locations on the opposite side of the board. The players move in clock-wise, sequential order, and at each turn, the player chooses to either:

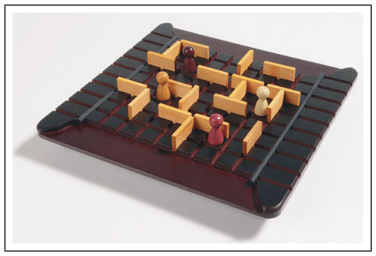
Figure 5: Quoridor board game

1. move his pawn horizontally or vertically to one of the neighboring squares.
2. place a wall piece on the board to facilitate his progress or to impede that of his opponent.

The walls occupy the width of two grid spaces and can be used to block pathways around the board as players cannot jump over them and must navigate around them. When placing a wall, an additional rule dictates that each player has to have at least one free path to a destination on the opposing side of the board. That prevents situations in which players team-up to enclose a pawn inside 4 walls. Walls are limited and useful resource and they cannot be moved or picked up after they are placed on the board.

Quoridor is an abstract strategic game which bears some resemblance of chess and checkers. The state-space complexity of Quoridor is composed by the number of ways to place the pawns multiplied by the number of ways to place the walls, minus the number of illegal positions. Such estimation was computed in [5] for the two-player version of the game and as such places the game in the middle between Backgammon and Chess in terms of the size of the search space. Obviously that search space increases dramatically when playing the 4-player version of the game.

5.2 Experimental Design and Results

We implemented a game environment in C++. The game board was represented as a graph and Dijkstra's algorithm was used to check the legality of wall positions (i.e., to check that there exist a path to the goal). We used a simple and straightforward heuristic evaluation function that sum the total distance of each of the players to the goal. Each player seeks to minimize his own distance while maximizing the opponents' distances. Moreover, to cope with the large branching factor of the game, we limited the possible locations that a wall can be placed to a fixed radius around the pawns.

In the first set of experiments, we played the Max-n, Paranoid, and SOS algorithms against a set of random SOS players that looked only one move ahead. In the initialized stage of each game we first randomized the social orientation of the three random players, as well as the position of the search-based player (as playing order might effect the result). After the initialization stage, in order to provide a robust comparison, each player was played against the same random setting from the same position.

[2]More information on that game can be found of the creator's website: http://www.gigamic.com/

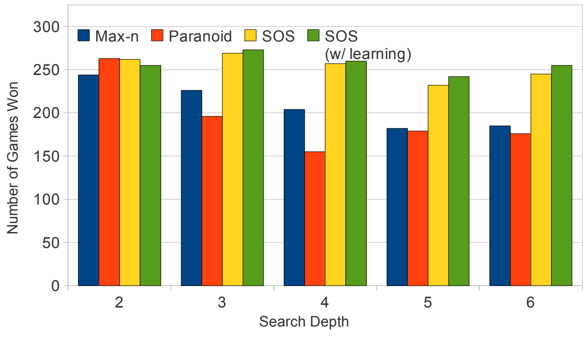

Figure 6: Results of playing Max-n, Paranoid, and SOS players in 500 Quoridor games against 3 random social-preference playing opponents. Two different SOS players were examined, one with absolute knowledge of the social-range matrix and another using naive learning method to approximate it.

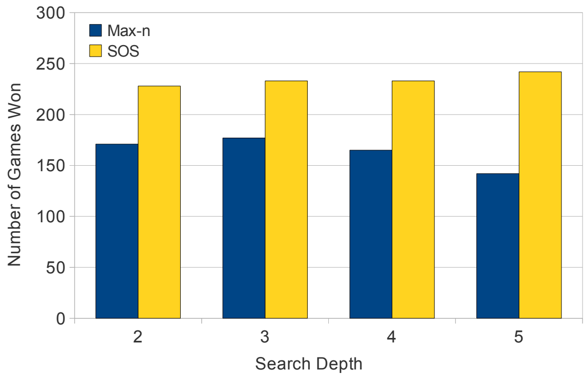

Figure 7: Results of playing Max-n and our SOS player in 500 Quoridor games against two random-preference players. The SOS player had absolute knowledge of the social-range matrix when playing.

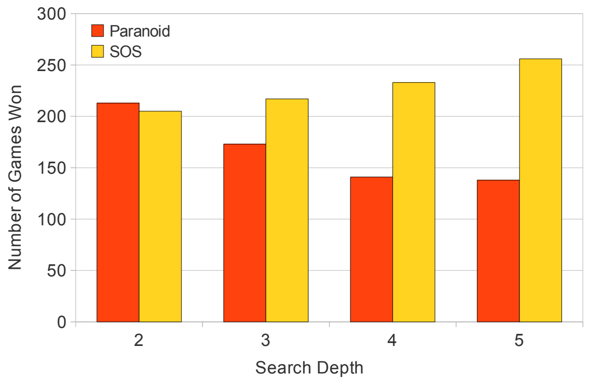

Figure 8: Results of playing Paranoid and our SOS player in 500 Quoridor games against two random-preference players. The SOS player had absolute knowledge of the social-range matrix when playing.

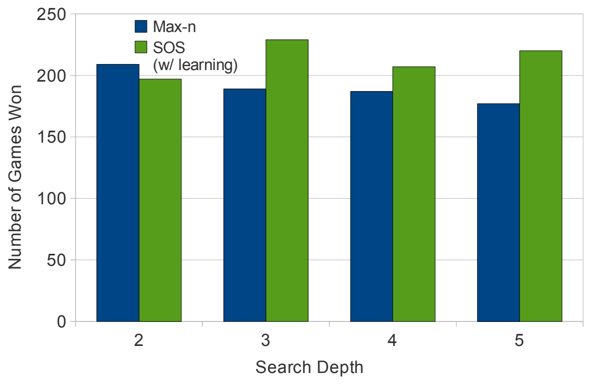

Figure 9: Results of playing Max-n and our SOS player in 500 Quoridor games against two random-preference players. The SOS player learned the social-range matrix with our naive learning algorithm and a history of size 5.

We played both versions of the SOS algorithm, in the first we assume our Socially-oriented player had absolute knowledge of the social-range matrix in this setting. The second version was the SOS with learning ($k = 5$), where the initial social matrix values were initialized to the Max-n matrix, presented in property 1, and the algorithm adapted these values during play. Figure 6 shows that both the versions of the SOS player significantly outperforms both Max-n and Paranoid after depth 2 ($\mathcal{P} < 0.01$ in a 2-tail Z-test). Interestingly, our simple learning rule performed in a manner that is completely comparable to the SOS algorithm with the full and accurate social information.

In the second set of experiments, we replaced one of the random-preference players with a Max-n or paranoid player in order to evaluate their head-to-head performance. Thus prior to each game we randomized two random players and then plugged in an SOS player as third player. The fourth player was Max-n in the first set of experiments and Paranoid in the secondset. We ran 500 games for each depth. Figures 7 and 8 show that the non-learning algorithm, still significantly outperforms Max-n and Paranoid when competing directly against them ($\mathcal{P} < 0.01$). Although both Max-n and Paranoid performance decreases, the paranoid assumption seems to have a greater effect on performance than the rationality assumption of Max-n.

In the last set of experiments we reproduced the same setting as in the second set, but this time used the SOS algorithm with learning, where k was set to 5. That is, the algorithm started each game with a random social matrix and used the suggested learning rule to adapt the values during game play. In figures 9 and 10 we can see that the online learning version of the SOS algorithm also do significantly better than Max-n and Paranoid in most cases ($\mathcal{P} < 0.01$), excluding depth 4 against Max-n, and depth 3 against Paranoid that are not statistically significant.

6. CONCLUSION

In this paper we addressed a fundamental question for search algorithms for extensive-form, multi-player games: *how to describe and reason about inter-player relationships during game play?* Our approach was to model the players' interpersonal relationships by incorporating ideas from *Social Value Orientation* theory into game-tree search.

Our Socially Oriented Search (SOS) algorithm models re-

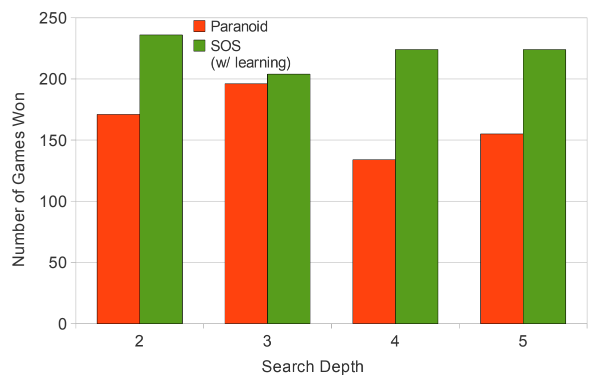

Figure 10: Results of playing Paranoid and our SOS player in 500 Quoridor games against two random-preference players. The SOS player learned the social-range matrix with our naive learning algorithm and a history of size 5.

lationships with a social-range matrix and uses it to compute the *perceived utilities* for each player and guide the search procedure. Our analytical results show that the algorithm can mimic the paranoid and Max-n behaviors by setting the social-matrix elements to specific values, and that the strategy computed by the SOS algorithm is a *perceived equilibrium*. Moreover, our analytical results relate the granularity of the evaluation function to the expected difference in strategies that will be selected by the SOS algorithm and Max-n and Paranoid. This relationship shows that using coarser evaluation functions reduces the ability to recognize social orientations.

We incorporated a simple learning algorithm into SOS to learn the social orientations of the players as the game progresses. The dynamic learning rule uses that the last k actions of each player to adapt its social-matrix during game play. In our evaluations of both the learning and non-learning versions of the algorithm using the Quoridor game, we found that in most cases they produced significantly better play than the Max-n and Paranoid algorithms.

For future work, we plan to evaluate the SOS algorithm in multi-player games that have both probabilistic elements (e.g., Risk) and incomplete information (e.g., Hearts). We also plan to evaluate the learning rule against players that change their orientations during game, as well as experimenting with techniques for learning the value of k.

7. ACKNOWLEDGMENTS

This work was supported in part by DARPA and U.S. Army Research Laboratory contract W911NF-11-C-0037, and AFOSR grant FA95500610405. The information in this paper does not necessarily reflect the position or policy of the U.S. Government, and no official endorsement should be inferred.

8. REFERENCES

[1] W. T. Au and J. Y. yee Kwong. Measurements and effects of social-value orientation in social dilemmas. *Contemporary psychological research on social dilemmas*, pages 71–98, 2004.

[2] R. M. Axelrod. *The evolution of cooperation.* Basic Books, New York, 1984.

[3] Bogaert, Sandy, Boone, Christophe, Declerck, and Carolyn. Social value orientation and cooperation in social dilemmas: A review and conceptual model. *British Journal of Social Psychology*, 47(3):453–480, 2008.

[4] C. G. M. David M. Messick. Motivational bases of choice in experimental games. *Experimental Social Psychology*, 1(4):1–25, 1968.

[5] L. Glendenning. Mastering quoridor - master thesis. Technical report, The University of New Mexico, 2005.

[6] D. Griesinger and J. Livingston. Towards a model of interpersonal motivation in experimental games. *Behavioral Science*, 18:173–188, 1973.

[7] P. Kuznetsov and S. Schmid. Towards Network Games with Social Preferences. *Structural Information and Communication Complexity*, pages 14–28, 2010.

[8] C. Luckhardt and K. Irani. An algorithmic solution of n-person games. In *AAAI*, pages 158–162, 1986.

[9] V. Mamadouh. Grid-group cultural theory: an introduction. *GeoJournal*, 47:395–409, 1999.

[10] S. Markovitch and R. Reger. Learning and exploiting relative weaknesses of opponent agents. *Autonomous Agents and Mutli-Agent Systems*, 10:2005, 2004.

[11] D. Mutchler. The multi-player version of minimax displays game-tree pathology. *Artificial Intelligence*, 64(2):323–336, 1993.

[12] D. S. Nau, M. Luštrek, A. Parker, I. Bratko, and M. Gams. When is it better not to look ahead? *Artificial Intelligence*, 174(16-17):1323–1338, 2010.

[13] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[14] J. Schaeffer, Y. Björnsson, N. Burch, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Solving checkers. In *IJCAI-05*, pages 292–297, 2005.

[15] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53:53–2, 1992.

[16] N. Sturtevant and R. Korf. On pruning techniques for multi-player games. In *AAAI*, pages 201–208, 2000.

[17] N. R. Sturtevant. A comparison of algorithms for multi-player games. In *CG*, pages 108–122, 2002.

[18] N. R. Sturtevant. Current challenges in multi-player game search. In *Computers and Games*, pages 285–300, 2004.

[19] N. R. Sturtevant, M. Zinkevich, and M. H. Bowling. Prob-maxn: Playing n-player games with opponent models. In *AAAI*, pages 1057–1063, 2006.

[20] I. Zuckerman, A. Felner, and S. Kraus. Mixing search strategies for multi-player games. In *IJCAI*, pages 646–651, 2009.