A Defense Model for Games with Incomplete Information

Wojciech Jamroga

Parlevink Group, University of Twente, Netherlands
jamroga@cs.utwente.nl
http://www.cs.utwente.nl/~jamroga

Abstract. Making a decision, an agent must consider how his outcome can be influenced by possible actions of other agents. A 'best defense model' for games involving uncertainty assumes usually that the opponents know everything about the actual situation and the player's plans for certain. In this paper it's argued that the assumption results in algorithms that are too cautious to be good in many game settings. Instead, a 'reasonably good defense' model is proposed: the player should look for a best strategy against all the potential actions of the opponents, still assuming that any opponent plays his best *according to his actual knowledge*. The defense model is formalized for the case of two-player zero-sum (adversary) games. Also, algorithms for decision-making against 'reasonably good defense' are proposed.
The argument and the ideas are supported by the results of experiments with random zero-sum two-player games on binary trees.

1 Introduction

Under uncertainty, an agent must consider all the possible situations, called often the *possible worlds*. Although he might not be able to distinguish between many of them, the actual 'state of affairs' severely influences the future course of action and the final income the agent is going to gain.

A two-player poker game may be a good example. The set of possible worlds Ω simply consists of all the possible card distributions. Suppose that MAX[1] has ♡AKJ8 ♣7 in the actual game. Then MAX can restrict his reasoning to the situations that seem plausible to him – namely, all the distributions where MAX has ♡AKJ8 ♣7 (see figure 1). In most situations MAX can't identify MIN's actual beliefs, because he doesn't know which cards MIN actually possess. Note however, that *if* MAX knew the actual world precisely, he would try to figure out the set of situations that seem plausible to MIN, as well as MIN's future line of action. The set would consist of all the distributions where MIN has a

[1] The agent of concern is often called the *MAX* player (since he should maximize his output) in the theory of zero-sum games. His opponent is labeled *MIN* then – he maximizes *his* output, which means that he tries to minimize the resulting score of MAX.

F. Baader, G. Brewka, and T. Eiter (Eds.): KI 2001, LNAI 2174, pp. 260–274, 2001.

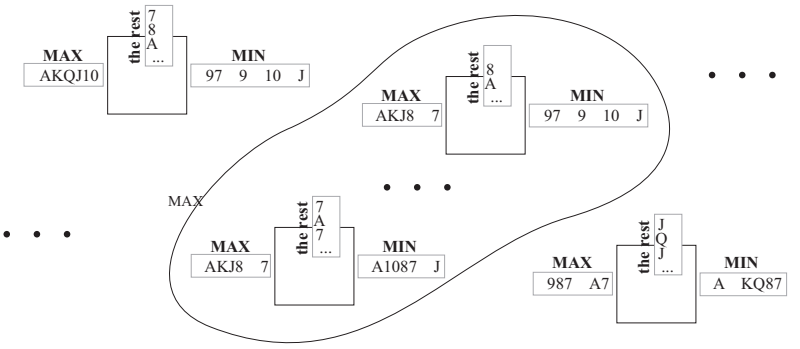

Fig. 1. The set of possible worlds Ω. In the actual game MAX has ♡AKJ8 ♣7, so he can restrict the set to Ω_{MAX}.

particular hand of five cards (in the example on figure 2: ♠97 ♡9 ♢10 ♣J). Note that such a hand must contain no cards possessed by MAX because MAX *knows* that MIN can't have them.

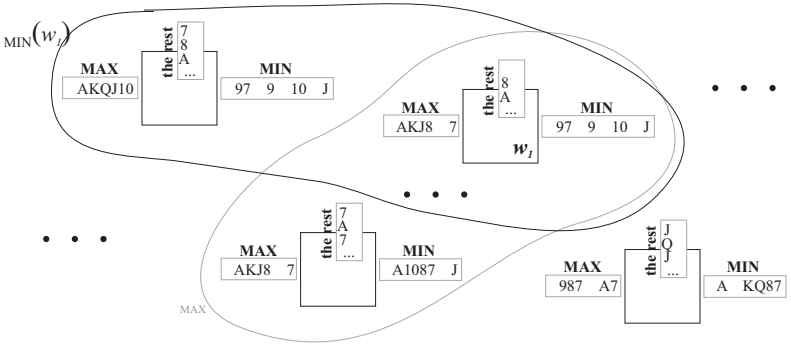

Fig. 2. The set of plausible worlds according to MIN – if w_1 is the actual distribution.

This is the idea underlying the decision-making algorithms *gvm* and *find-optimal*, proposed in this paper. The player can consider every world from Ω separately – identifying the possible distributions of resources as well as possible beliefs of the opponents. Then he can choose the action that gives him the highest expected outcome over all the worlds.

2 Best Defense

In Game Theory, a player is assumed to play against optimal defense, since a 'rational opponent' makes always the best decision. For zero-sum games this implies that the opponent chooses the worst move from the player's perspective. A best defense model for imperfect information games was proposed in (Frank 1996),

(Frank & Basin 1998a) and (Frank & Basin 1998b). It contains the following assumptions:

1. MIN has perfect information about the situation,
2. MIN knows MAX's actual strategy,
3. MAX's knowledge is limited to just knowing the set of all possible situations (worlds) Ω,
4. the strategy adopted by MAX must be a pure strategy.

MAX maximizes his expected payoff value (over Ω). The opponent (MIN) is assumed to be omniscient; thus, he can maximize his payoff directly.

Since the problem of finding optimal strategy (even in such a simplified setting) is NP-complete, there is a strong need for suboptimal but less complex algorithms. A number of minimaxing algorithms – including vector minimaxing (*vm*) and payoff-reduction minimaxing (*prm*) – were proposed in (Frank, Basin & Matsubara 1998) and (Frank & Basin 1998b). The algorithms were then compared to the algorithm of Monte Carlo sampling (*MC*),[2] based on classical minimaxing. In the competition an algorithm was claimed better if it was finding strategies close to optimal more frequently than its competitor – within the notion of 'optimality' defined above. In a series of experiments on random tree games, Monte Carlo sampling algorithm was definitely outperformed by *prm* (and it turned out to be slightly worse than *vm*, too). However, it's not clear why an algorithm that plays very well against an omniscient opponent should also win in a more realistic competition.

2.1 Experiments with Random Games

The experiment idea is strongly based on the experiments done by (Frank, Basin & Matsubara 1998). The test was conducted for games on complete binary trees of depth D. For any of N possible worlds from Ω a payoff is assigned to every tree leaf; the payoff may be either 0 or 1. If a game ends in leaf l and world w appears to be the actual world, the player wins the payoff value assigned to (l, w), and the opponent loses the same amount. A possible world is described with a list of payoffs for all the tree leaves in this world, called a *payoff vector*. Thus, to generate a random game, one must generate a random payoff vector for each world. Note that two different worlds can have same payoff vectors.

An example of such a game is shown on figure 3.

To make this a fair competition between *algorithms* – the competitors must be provided with equal chances of winning. Say the algorithms are called A and B. Now, for a particular (randomly generated) game:

1. first A plays as MAX and B as MIN. The strategies are identified and the expected value of payoff computed (over Ω);
2. then *the same* game is played again – A plays as MIN and B as MAX;
3. at the end the difference between payoffs is computed. If the difference is positive, A wins; if it's negative then B is the winner.

[2] (Corlett & Todd 1985), (Ginsberg 1999)

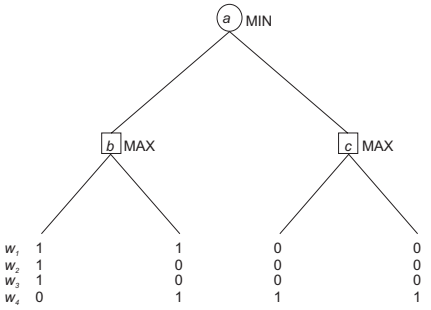

Fig. 3. A game tree of depth $D = 2$ with $N = 4$ possible worlds

1000 games with MAX as a starting player and 1000 games with MIN as a starting player were played during every such competition. The algorithms involved in the competition were: *MC*, *vm*, *prm*, and the simplest algorithm for finding the optimal strategy against 'best defense' – checking all the possible player's strategies one by one (let's name it *opt-bd*, for instance). The output of every competition is described by A's 'triumph supremacy' (number of rounds won by A minus rounds lost by A) and A's payoff supremacy (the average expected payoff value per 1000 rounds). Most experiments were conducted for games with $D = 8, N = 1000$, except the competitions involving *opt-bd* algorithm – $D = 4, N = 1000$ (analyzing any game of more than 4 turns is practically infeasible for *opt-bd*). The results of the actual experiments are shown on figure 4.

	triumph supr.	payoff supr.
MC vs. vm	3.4%	0.2
MC vs. prm	39.1%	9.2
vm vs. prm	33.9%	8.0

	triumph supr.	payoff supr.
MC vs. opt-bd	36.3%	7.1
vm vs. opt-bd	38.4%	8.1
prm vs. opt-bd	19.1%	4.2

Fig. 4. The competition output: triumph supremacy (in [%] of total rounds played) and payoff supremacy (per 1000 rounds). Experiment setting: $N = 1000$ worlds; tree depth $D = 8$ (array on the left), $D = 4$ (array on the right).

However, the setting of the experiment above may be somewhat misleading since we implicitly assumed that both agents have the same knowledge. In real situations most agents can at least eliminate some of the worlds from Ω as being impossible. For instance, in card games every agent holds some cards in his hand. He knows the cards he has, so he can exclude all the card distributions inconsistent with this knowledge. Since different players have access to different pieces of the reality – the worlds actually possible for MAX ($\Omega_{MAX} \subset \Omega$) and for MIN ($\Omega_{MIN} \subset \Omega$) should differ in most cases.

New experiment: MAX and MIN find their strategies with respect to separate sets of worlds considered to be possible. Ω_{MAX} and Ω_{MIN} are generated on

random for every game. The players' knowledge is assumed to be adequate, i.e. the output is computed as the expected value of payoff over the worlds from $\Omega_{MAX} \cap \Omega_{MIN}$ only. Since there is one more random factor in the setting, 20000 games per round are played instead of 2000. The results are shown on figure 5.

	triumph supr.	payoff supr.
MC vs. vm	2.1%	2.1
MC vs. prm	21.7%	17.7
vm vs. prm	21.7%	14.1

	triumph supr.	payoff supr.
MC vs. opt-bd	18.8%	12.1
vm vs. opt-bd	20.4%	13.0
prm vs. opt-bd	10.6%	7.0

Fig. 5. The competition again: players' belief sets are generated randomly. Experiment setting: $N = 1000$ worlds; tree depth $D = 8$ (array on the left), $D = 4$ (array on the right).

The results of the experiments show that it doesn't have to be beneficial for a player to assume that the opponent reaches the upper bound of his theoretical capabilities (especially in the context of his knowledge about the actual situation). The cautious algorithms: *prm* and *opt-bd* were in fact outperformed by Monte Carlo sampling, which was considered very suboptimal. A possible reason lies in incoherence of the adopted best defense assumptions with the situations being encountered in the actual games.

In perfect information games the opponent (given sufficient resources) plays sub-optimally only by his own fault. He can always use the best defense strategy since he can find it by minimaxing. The agents can always play best defense in perfect information games (just by finding the strategies with minimax). On the other hand, in games with incomplete information the opponent is seldom able to fulfill the 'best defense' assumption because his knowledge is insufficient. Thus, the model makes the player assume a defense which is impossible to be met in most cases.

3 Reasonably Good Defense

As the experiments showed, it is not beneficial for the player to overestimate capacities of the opponent too much. The best defense model by Frank & Basin refers clearly to the worst possible line of events, but this line is quite unlikely to occur.

In a probabilistic framework a model of MIN's beliefs is necessary. The model should include MIN's beliefs about the actual situation as well as beliefs about the player's beliefs. The beliefs may depend on the actual world and the state of the game. MIN maximizes his expected payoff over Ω with respect to his actual state of belief (i.e. he minimizes the payoff for MAX in zero-sum games). MAX should maximize his expected payoff over Ω and the set of possible MIN belief states.

Reasonably good defense model:

> If nothing suggests the contrary, the opponent should be assumed capabilities similar to the player. Thus, MAX's knowledge and skills, and his model of MIN should be symmetrical.

In particular – if (in a given situation) no specific knowledge is available about likelihood of some possible worlds or different opponent beliefs, equal probabilities should be supposed a priori, also when modeling the beliefs of other agents.

3.1 A Simple Case

The problem is often analyzed in a simplified version, when all the worlds are equally probable by rule, but the agents can identify some of them as implausible at some point of a game. MAX's knowledge can be described as $\Omega_{MAX} \subset \Omega$. The player can't know which worlds $w \in \Omega$ are considered to be plausible by the opponent. However, he can restrict the set of possible MIN's belief sets Ω_{MIN} to those beliefs which are consistent with his own knowledge – like in the example presented on figures 1 and 2 back in section 1.

To evaluate a MIN node s, MAX may consider evaluations for all the possible opponent's belief sets Ω_{MIN}. (Gambäck et al. 1993) present an interesting example of such an analysis, concerning Bridge bidding. The player sees his own cards (hand) – so he can generate possible distributions from Ω_{MAX} by assigning the remaining cards to the other players (the authors call each of the alternative assignments an *R-deal*). Every world (R-deal) $w \in \Omega_{MAX}$ determines a MIN's belief set $\Omega_{MIN}(w)$ – namely, $\Omega_{MIN}(w)$ is a set of worlds which cannot be distinguished from w by the opponent (in the actual state of the game). In the case of Bridge bidding, for instance $\Omega_{MIN}(w)$ consists of all the distributions w' in which MIN has exactly the same cards as in w. Whenever MAX needs to consider opponent's decisions he can model the opponent's view by generating $\Omega_{MIN}(w)$ for each $w \in \Omega^0_{MAX}$ (since analogous function $\Omega_{MAX}(w)$ is needed to model the opponent's knowledge about the player's possible beliefs, let's rather call the MAX's actual belief Ω^0_{MAX} to avoid confusion).

Note that the actual shape of $\Omega_{MIN}(w)$ depends on the game rules. For instance, two players can't possess the same card in a poker game. So if MIN has ♢A1087 ♣J in a world w then $\Omega_{MIN}(w)$ includes all the situations of MIN having exactly ♢A1087 ♣J, and MAX having none of the cards (the rest of the deck must contain none of these cards, too). However, this would not work for Canasta, where two complete decks of cards are mixed and dealt – so two different players can even possess hands of the same shape!

An algorithm for finding the decision against 'reasonably good defense' (2 players, zero-sum game, no information about worlds' likelihood except that some worlds are actually impossible; the player's belief doesn't change when moving to another game state – no information flow) is shown on figure 6. *Generalized vector minimaxing* (*gvm*) is a more universal version of algorithms like Monte Carlo or vector minimaxing from (Frank et al. 1988). The algorithm has been inspired by ideas from (Carmel & Markovitch 1996) and (Gambäck et al. 1993) – the player looks forward for his opponent's decision in every possible situation, and then maximizes his expected output against such defense. *Gvm* allows to model the players' knowledge on any arbitrary level, since the functions Ω_{MIN}, Ω_{MAX} are assumed to mutually encode a player's beliefs about his

gvm (Game, s, player, Worlds);

Generalized vector minimaxing. Returns the evaluation vector $(eval[w_1], ..., eval[w_n])$ for node s, together with the player's chosen move. Parameters:

Game: game definition, including functions $Succ(s)$ – returning the set of successors for node s, $Strat(player)$ – returning the set of all the possible player's (complete) strategies, $payoff(l)$ – returning the MAX's payoff vector $(payoff(l)[w_1], ..., payoff(l)[w_n])$ in leaf l, $\Omega_{MIN}(w)$, $\Omega_{MAX}(w)$ – returning MIN's and MAX's beliefs in a particular world w;

s: game state (node);

player: the agent who makes a decision at this node (MAX or MIN);

Worlds: the agent's actual belief (Ω^0_{MAX} or Ω^0_{MIN} in this case);

if $Succ(s) = \emptyset$ then return $(nil, payoff(s))$;
else:

- for every $s' \in Succ(s)$ compute $e_{s'} = (e_{s'}[w_1], ..., e_{s'}[w_n])$ as:
  $$e_{s'}[w] = \begin{cases} gvm(Game, s', \text{MIN}, \Omega_{MIN}(w))[w] & \text{if } player = \text{MAX} \\ gvm(Game, s', \text{MAX}, \Omega_{MAX}(w))[w] & \text{if } player = \text{MIN} \end{cases}$$
  for every world $w \in \Omega$;
- if *player*=MIN then return $(s', e_{s'})$ such that $\sum_{w \in Worlds} e_{s'}[w]$ is minimal. else return $(s', e_{s'})$ such that $\sum_{w \in Worlds} e_{s'}[w]$ is maximal.

Fig. 6. Generalized vector minimaxing.

opponent's beliefs as well as his beliefs about his opponent's beliefs about his beliefs etc.

Note that if

- $\Omega_{MAX}(w) = \Omega^0_{MAX}$ (the opponent knows the player's state of belief)

then $gvm(Game, s, MAX, \Omega^0_{MAX})$ returns the same strategy as the *vector minimaxing* algorithm proposed by Frank, Basin & Matsubara. If we also assume that

- $\Omega_{MIN}(w) = \{w\}$ (the opponent always knows the actual situation),

we obtain the instance of vector minimaxing that was actually used in (Frank et al 1998).

On the other hand, if the game definition includes the following assumptions:

- $\Omega_{MIN}(w) = \{w\}$,
- $\Omega_{MAX}(w) = \{w\}$;

then *gvm* becomes equivalent to classical Monte Carlo minimaxing.

The main disadvantage of *gvm* is that it's not always able to find the optimal strategy – due to *non-locality*, a phenomenon observed originally in (Frank 1996), and formalized in (Frank & Basin 1998a). The game tree on figure 3 demonstrates the phenomenon well. Consider MAX's decision at node b. If the analysis was to be 'local', MAX would have to prefer the left-hand branch, since his expected

payoff equals 0.75 then (against 0.5 when following the right-hand branch). But making decision at node b means that MIN has made his decision to move to b, not to c, at node a. If we assume that MIN is rational, then his decision must make sense – and it makes sense only when he believes that worlds w_1, w_2, w_3 are irrelevant. In other words, w_4 is obviously the only world considered possible by MIN at this moment. If we assume that his beliefs are adequate (they can be incomplete, but never false), then MAX has to restrict his computation to w_4 alone, and therefore to pick up the right branch.

The implication of non-locality is that any 'compositional' algorithm that looks only forward, not backward, is bound to be suboptimal. Thus, the player should evaluate his decisions against whole strategies of the opponent, not their parts only. On the other hand, it's not possible to simulate the opponent's minimaxing over the whole game tree, because this would lead to an infinite loop.

Figure 7 presents an algorithm for finding the optimal strategy against reasonably good defense. The algorithm is based on the equilibrium definition for zero-sum games. It computes the minmax and the maxmin over the sets of players' strategies, and if they lead to the same result then the optimum has been found. Unfortunately, there is often no such an optimum. In this case *findoptimal* returns the minmax strategy, which describes the lower bound of the outcome the player can expect (since the assumption that the opponent always knows the player's strategy beforehand defines the upper bound of the opponent's knowledge). Another drawback of the algorithm is its computational complexity.

Note also that if the game definition includes the following assumptions:

1. $\Omega_{MIN}(w) = \{w\}$,
2. $\Omega_{MAX}(w) = \Omega^0_{MAX}$;

then $findoptimal(Game, MAX, \Omega^0_{MAX})$ computes the optimal strategy with respect to the 'best defense model' by Frank and Basin. In this sense their 'best defense' is a special case of 'reasonably good defense'. Yet the opponent's omniscience is not assumed (in practice) to be an inherent property of games, but has to be stated explicitly via Ω_{MIN}, Ω_{MAX} functions definition. Moreover, the algorithm indicates whether it's necessary to make any stronger claims about the opponent's knowledge to obtain a solution.

Finally, it is worth noting that – while *gvm*, as a minimaxing algorithm, has to be suboptimal for games with incomplete information – it can probably be improved in terms of accuracy. It demands for some reduction of the impact of non-locality on the decision-making process. Frank, Basin and Matsubara has already done it for traditional minimaxing within their 'best defense model' – *prm* algorithm is one of the results.

3.2 Computational Complexity

Not surprisingly, *opt-bd* is highly inefficient since it checks every possible player's strategy – its complexity is doubly exponential on the tree depth and linear on the number of worlds (namely, $o(N * b^{b^D})$ – where b stands for the branching

findoptimal (Game, player, Worlds);

The function returns the player's chosen strategy. It indicates also whether the strategy is optimal, or if it refers to the lower bound of the player's actual output (when the optimum doesn't exist). The algorithm presented below defines the function for $player = MAX$. If $player = MIN$, the algorithm is quite analogous.

[Function *output* computes the payoff value given a complete strategy pair and a particular world from Ω.]

For every $str \in Strat(MAX)$:

- for every $w \in \Omega$: let $strvect[w] = str$;
- for every $w \in Worlds$: $min[w] = maximize(MIN, strvect, \Omega_{MIN}(w))$;
- let $eval[str] = \sum_{w \in Worlds} output(str, min[w], w)$;

Let str_1 be that str for which $eval[str]$ is maximal;

For every $strvect \in \{\Omega \rightarrow Strat(MIN)\}$ such that $\forall_{w,v}\ (v \in \Omega_{MIN}(w) \Rightarrow strvect(v) = strvect(w))$:

- for every $v \in Worlds$: $max[v] = maximize(MAX, strvect, \Omega_{MAX}(v))$;
- let $eval[strvect] = \sum_{w \in Worlds} \sum_{v \in \Omega_{MIN}(w)} output(max[v], strvect[v], v)$;

Let $strvect_2$ be that $strvect$ for which $eval[strvect]$ is minimal;

if $eval[str_1] = max_{str}\{\sum_{w \in Worlds} output(str, strvect_2[w], w)\}$

- then return (str_1, OPTIMUM);
- else return (str_1, BOUND);

maximize (Game, player, strvect, Worlds);

The function searches the set of all *player*'s strategies, trying to maximize the expected payoff value over the given set of possible worlds against given opponent's strategy vector. Of course, MIN player wants to maximize *his* own payoff, i.e. to minimize the score defined by the *payoff* function.

Argument *strvect* is a function of type $\Omega \rightarrow Strat(MIN)$ for $player = MAX$, and $\Omega \rightarrow Strat(MAX)$ for $player = MIN$

if $player = MAX$ then:

- return the strategy $str \in Strat(MAX)$ for which $\sum_{w \in Worlds} output(str, strvect[w], w)$ is maximal;
- else: return the strategy $str \in Strat(MIN)$ for which $\sum_{w \in Worlds} output(strvect[w], str, w)$ is minimal;

Fig. 7. Algorithm for finding the optimal play against *reasonably good defense*.

factor of the game tree).[3] That's why suboptimal algorithms are useful: *MC* and *vm* as well as *prm* have the time complexity of $o(N * b^D)$ (exponential on the tree depth, linear on the number of worlds).

When the opponent's beliefs are taken into account, the complexity increases. The complexity of *findoptimal* algorithm is $o(N^2(b^{\frac{b^D}{2}})^{N+1})$, so in practical applications a suboptimal (but faster) algorithm is necessary. The complexity of *gvm* is exponential on the tree depth and polynomial on the number of worlds: $o(N^D * b^D)$. The practical complexity of the algorithm may be slightly reduced if we assume that players' beliefs must be adequate: $w \in \Omega_{MIN}(w)$, $w \in \Omega_{MAX}(w)$ for every w. Then the construction of evaluation vectors can be restricted to $w \in Worlds$ only (instead of all $w \in \Omega$).

There is a number of methods that can be used to reduce the search time at the expense of its accuracy. Sampling is often used to make the search feasible when the number of possible situations is huge; Gambäck, Rayner and Pell (Gambäck et al. 1993) propose such an approach for the case of bridge bidding, and Ginsberg's successful GIB program (Ginsberg 1999) uses Monte Carlo sampling in the complex domain of bridge card play. Evaluation approximator may help to keep the search depth at a reasonable level – Gambäck et al. used a trained neural network to implement such an approximation function, and they reported good results. Also, pruning techniques and heuristic search can be used for most domains of application.

3.3 More Experiments...

To test the new ideas against existing minimaxing algorithms, random binary tree games can be used again. To provide a natural interpretation to the belief functions Ω_{MIN}, Ω_{MAX} every game is treated as an 'imaginary card game'. Every world from Ω is defined by two hands of c 'cards' – one hand for each player. The deck consists of n 'cards'. A player can play only either the lowest or the highest card he possesses at the moment (when it's his turn to move, of course), so at any node (except the leaves) there are exactly two alternative decisions that can be made, regardless of the actual hand the player possesses. The payoffs for every leaf and each possible world are generated at random.

	triumph supr.	payoff supr.
gvm vs. MC	99.8%	94.9
gvm vs. prm	98.1%	76.0
gvm vs. vm	99.8%	90.5
gvm vs. opt-bd	98.9%	83.8

	triumph supr.	payoff supr.
MC vs. vm	2.7%	1.1
MC vs. prm	13.6%	5.2
vm vs. prm	34.8%	12.9
MC vs. opt-bd	77.2%	38.3
vm vs. opt-bd	85.9%	46.6
prm vs. opt-bd	80.8%	30.6

Fig. 8. Example results for $n = 7$, $c = 3$ cards (tree depth $D = 4$, $N = 140$ possible worlds).

[3] in the case of the experiments here: $b = 2$.

Now, $\Omega_{MIN}(w)$ is the set of worlds that cannot be excluded by MIN player in world w. Namely, it consists of these worlds that assume MIN having the hand he actually has. MIN's beliefs about MAX's beliefs, $\Omega_{MAX}(w)$ are defined in the same way. It is assumed that the players lead their cards secretly, i.e. the opponent doesn't know what card was played exactly – he knows only whether it was the highest or the lowest one (the actual world is recognized by both players no sooner than at the end of the game). Game parameters are: the tree depth $D = 2(c-1)$, and the number of possible worlds $N = \binom{n}{c} \cdot \binom{n-c}{c}$.

The *gvm* algorithm is played against other algorithms in a way similar to the experiments before. For a particular game, the game is played for every world (card distribution), and then the average value of payoff is computed. 1000 games are played for every competition: 500 with MAX as the leading player, and 500 with MIN starting the game (only for $n = 8, c = 3$, 200 games has been played due to complexity reasons). The results (triumph supremacy in [%] of total rounds played; payoff supremacy – average/estimated payoff per 1000 rounds) are shown on figures 8 and 9. Figure 8 shows also some example results of a competition between the traditional algorithms to make the comparison more thorough.

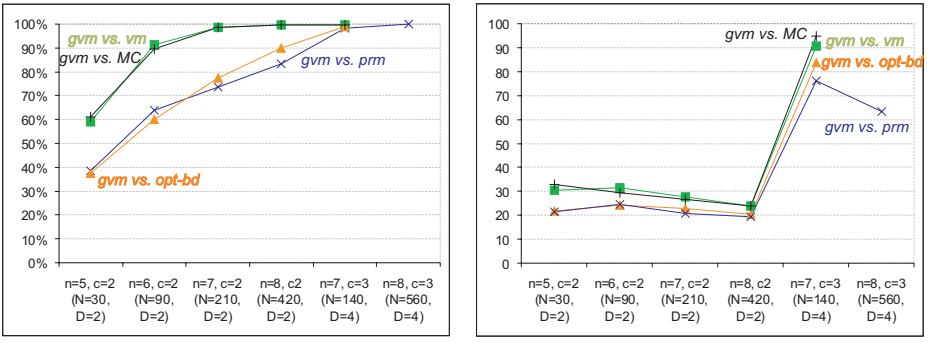

Fig. 9. Triumph supremacy in [%] of total rounds played (left), and Payoff supremacy per 1000 rounds (right).

In every competition *gvm* appeared to be at least 37.5% better than any of the other algorithms (in terms of the triumph supremacy). Moreover, as the game complexity increases, *gvm* starts to win practically 100% rounds.

The results reveal that algorithms like *prm* or *opt-bd* loose less than *MC* or *vm* when played against *gvm*. However, when played against each other, the previous pattern still holds: *MC* wins with *vm*, *prm* and *opt-bd*, *vm* wins with *prm* and *opt-bd* etc. The reason lies probably in the fact that *prm* and *opt-bd* were designed to play against a considerably more potent opponent. Thus, playing against *gvm* they can benefit from their cautiousness. On the other hand, *MC* and *vm* are apparently better off in games against an enemy of the same or similar level of skill and knowledge.

Another observation may be more surprising. When the tree depth increases, the gap between the results of *gvm* grows rapidly – in terms of the payoff supremacy as well as the frequency of winning. However, for a fixed D, *gvm* seems to earn less average payoff if the number of possible worlds increases, while still winning more and more rounds. This means that *gvm* succeeds to find a superior strategy for more initial 'hands of cards', but its expected payoffs for every particular hand decrease. The reason is perhaps that when N grows, more worlds are possible for any particular hand. Even a good algorithm can't play for all the worlds at the same time, so it is bound to fail in quite a number of them. Thus, an average difference in payoffs decrease, although *gvm* is still able to find a strategy better than the others. Moreover, when the tree depth is small, there is a very limited amount of different payoff vectors available. Now, when the number of worlds being considered at every node increase, it becomes more likely that the actual $Worlds$ set may be similar to some of $\Omega_{MIN}(w)$ and/or $\Omega_{MAX}(w)$ sets. Which means that *gvm* minimaxes over similar payoff vectors as its competitors in many games.

3.4 Generalizations

The games analyzed so far were constrained by several important simplifications. More realistic setting should include the following issues:

- for a game node (state) s: not every move (arc) can be taken in a particular situation $w \in \Omega$. Example: a player can lead A♠ only when he has A♠;
- most moves introduce new information. Example: the opponent led A♠. Now, all the worlds in which he hadn't A♠ can be regarded as impossible;
- payoff values for a leaf l are defined only in these worlds in which we can access the leaf. In fact, the players know the situation (more or less) after the last move in many games. Thus – for a particular leaf – payoffs in most worlds make no sense.

To incorporate this perspective, the following assumptions can be made:

- payoff vector is a partial function – $\mathit{payoff}(l) : \Omega \rightharpoonup R$ (values for some l, w can be undefined: $\mathit{payoff}(l)[w] = \mathit{undef}$). It's good to assume that: $a + \mathit{undef} = a \cdot \mathit{undef} = \mathit{max}(a, \mathit{undef}) = \mathit{min}(a, \mathit{undef}) = a$;
- legal moves are determined by a function Acc. $Acc(s)$ denotes the set of worlds in which node s is accessible. Acc can be implemented as follows:
  1. if s is a leaf then $Acc(s) = \{w \in \Omega : \mathit{payoff}(s)[w] \neq \mathit{undef}\}$,
  2. otherwise $Acc(s) = \bigcup_{s' \in Succ(s)} Acc(s')$.
- every move can reveal some new information, so the beliefs may change as the state changes – $\Omega_{MAX}, \Omega_{MIN} : State \times \Omega \to \mathcal{P}(\Omega)$.

The resulting structure resembles in a way the semantics underpinning $\mathcal{LORA}$, a complex modal logic for BDI agents proposed in (Wooldridge 2000) – with the game tree defining the branching of time, and $\Omega_{MAX}, \Omega_{MIN}$ standing for the belief accessibility relations – although $\mathcal{LORA}$ proceeds with qualitative, not probabilistic approach to beliefs.

Next generalization:

gvm (Game, s, player, belief);

Generalized vector minimaxing. Returns the evaluation vector $(eval[w_1], ..., eval[w_n])$ for node s, together with the player's chosen move. Parameters:

Game: game definition, including functions $Succ : State \to \mathcal{P}(State)$ – returning the set of successors for each node, $payoff : State \times \Omega \to \mathbb{R}$ – returning MAX's payoffs, Bel_{MIN},$Bel_{MAX} : State \times \Omega \to ((\Omega \to [0,1]) \to [0,1])$ – beliefs about the likelihood of particular opponent's beliefs in a particular situation;

$s : State$ – the game state being considered;

$player : \{MAX, MIN\}$ – the agent who makes the decision at the state;

$belief : \Omega \to [0,1]$ – the agent's actual belief (a probability function);

if $Succ(s) = \emptyset$ then return $(nil, payoff(s))$;
else:

- for every $s' \in Succ(s)$, $w \in \Omega$, and for every possible opponent's belief *oblf* simulate the opponent's minimaxing:
$$opp_{s'}[oblf, w] = \begin{cases} gvm(Game, s', \text{MIN}, oblf)[w] & \text{if } player = \text{MAX} \\ gvm(Game, s', \text{MAX}, oblf)[w] & \text{if } player = \text{MIN} \end{cases}$$
- compute the expected payoff for every $s' \in Succ(s)$, $w \in \Omega$:
$$e_{s'}[w] = \begin{cases} \sum_{oblf} Bel_{MIN}(s, w, oblf) \cdot opp_{s'}[oblf, w] & \text{if } player = \text{MAX} \\ \sum_{oblf} Bel_{MAX}(s, w, oblf) \cdot opp_{s'}[oblf, w] & \text{if } player = \text{MIN} \end{cases}$$
- if *player*=MAX then return $(s', e_{s'})$ such that $\sum_{w \in \Omega} belief(w) \cdot e_{s'}[w]$ is maximal.
else return $(s', e_{s'})$ such that $\sum_{w \in \Omega} belief(w) \cdot e_{s'}[w]$ is minimal.

Fig. 10. Generalized vector minimaxing revisited.

– players may be able to determine some probabilities for the possible worlds – not only to tell which worlds are plausible and which implausible now. Thus, an actual belief may be a probability function $[0,1] \to \Omega$ instead of being just a subset of Ω;
– in a given state of a game (and a world), more than 1 belief may be possible within the opponent's model. This would mean that the player doesn't know the opponent's reasoning scheme precisely and is bound to guess which belief states can result from the opponent's information analysis. He may also consider some of the possible opponent's beliefs more likely than the others.

A new version of *gvm* that takes the new possibilities into account is shown on figure 10; *findoptimal* algorithm can be generalized in a similar way.

The set of all the possible beliefs is in general infinite, so it demands for some reduction of the problem. A clever sampling of the set may be a good solution.

3.5 Dealing with More Capable Opponents

In the actual experiments, functions like Ω_{MIN} were designed to describe what a rational opponent *must* know in a given situation. On the other hand, the

opponent *may* know (or believe he knows) much more. He can even happen to know the whole situation – he may have guessed it from his own card, the MAX player's first move, or even kibitzers' facial expressions. Frank's best defense assumptions refer clearly to what the opponent can know in the worst possible case.

To deal with such 'possibly more capable' opponent, the player should consider every possible opponent's belief from between what he must and what he can know. For the experiment setting (no difference in probability of plausible worlds) this would mean maximizing the expected value over all the belief sets B such that $\{w\} \subseteq B(s,w) \subseteq \Omega_{MIN}(s,w)$. Thus, to model this situation accurately, it is sufficient to assume

$$Bel_{MIN} = \begin{cases} \frac{1}{k} & \textbf{if} \quad \exists_B \ \{w\} \subseteq B(s,w) \subseteq \Omega_{MIN}(s,w) \ \wedge \ p(w) = \begin{cases} \frac{1}{|B|} & \text{if } w \in B \\ 0 & \text{else} \end{cases} \\ 0 & \textbf{otherwise} \end{cases}$$

within the input for the generalized version of *gvm* or *findoptimal*. [4]

In the general case (analyzed in the previous section) we can have beliefs about opponent's beliefs defined explicitly with probability functions Bel_{MIN}, Bel_{MAX}. Simply, when we suspect the opponent of being more capable than just looking at his card and/or the board, it's good to design the functions so that if any opponent's belief is assumed possible then all the more precise beliefs are also assumed possible.

4 Conclusions

This paper advocates a thesis that assuming a complete omniscience of the opponent may be not quite reasonable in games with incomplete information. Instead, the player should optimize his strategy against the expected performance of the other agent (in the mathematical sense). If the player can identify the opponent's belief for various possible situations, he can do some reasoning in the way Gambäck, Rayner and Pell showed for the specific case of Bridge bidding. Algorithms: *gvm* and *findoptimal* implement the idea, and the results of the experiments suggest that the 'reasonably good defense model', proposed in this paper, may make sense after all. Of course, the algorithms – especially *findoptimal* – are too inefficient to be used in practice, but they can provide a good benchmark for evaluation of suboptimal, faster ones.

The defense model proposed here – in contrast to the model by Frank and Basin – emphasizes the importance of a good information-processing subsystem, necessary to acquire and maintain an adequate knowledge about the opponent. The actual opponent model may be derived from the game rules or learned by the playing agent during the play. The point is that if the player has any (even uncertain) information about the agent he plays against available, he should use it instead of ignoring it. And if the player has *really* no information about the other agent, he may be better off assuming average capabilities of the opponent, rather than capabilities the opponent is unlikely to possess.

[4] k must represent the number of possible B sets in the equation to keep the probability function normalized.

References

1. Carmel D., Markovitch S. (1996), Learning and Using Opponent Models in Adversary Search, Technical Report CIS9609, Technion, Haifa.
2. Corlett R., Todd S. (1985), A Monte-Carlo Approach to Uncertain Inference, In Ross P., ed., *Proceedings of the Conference on Artificial Intelligence and Simulation of Behaviour*, pp. 28-38.
3. Frank I. (1996), *Search and Planning under Incomplete Information. A Study using Bridge Card Play*, Ph.D. diss., University of Edinburgh. Published in 1998 by Springer-Verlag.
4. Frank I., Basin D. (1998a), Search in Games with Incomplete Information. A Case Study using Bridge Card Play, *Artificial Intelligence*, 100(1-2), pp. 87-123.
5. Frank I., Basin D. (1998b), Optimal Play Against Best Defense: Complexity and Heuristics, In Proceedings of the First International Conference on Computers and Games, *Lecture Notes in Computer Science*, Vol. 1558, Springer–Verlag.
6. Frank I., Basin D., Matsubara H. (1998), Finding Optimal Strategies for Imperfect Information Games, In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-98*.
7. Gambäck B., Rayner M., Pell B. (1993), Pragmatic Reasoning in Bridge, Technical Report No. 299, University of Cambridge.
8. Ginsberg M. (1999), GIB: Steps Toward an Expert-Level Bridge-Playing Program. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pp. 584-589.
9. Ginsberg M. (1996), How Computers Will Play Bridge. *The Bridge World*, June.
10. Klir G.J., Folger T.A. (1988), *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, Englewood Cliffs.
11. Kofler E. (1963) *Wstęp do teorii gier. Zarys popularny* [*An Introduction to Game Theory. Popular Approach*]. PZWS, Warszawa.
12. Sen S., Weiss G. (1999), Learning in Multiagent Systems. In: Weiss G. (ed.), *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, pp. 259-298, MIT Press, Cambridge, Mass.
13. Weiss G., ed. (1999), *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge, Mass.
14. Wooldridge M. (2000), *Reasoning about Rational Agents*, MIT Press, Cambridge, Mass.