On Addressing the Challenges of Complex Stochastic Games Using "*Representative*" Moves

Armando H. Taucer, Spencer Polk, and B. John Oommen

School of Computer Science, Carleton University, Ottawa, Canada**

Abstract. The problem of achieving competitive game play in a board game, against an intelligent opponent, is a well-known and studied field of Artificial Intelligence (AI). This area of research has seen major breakthroughs in recent years, particularly in the game of Go. However, popular hobby board games, and particularly Trading Card Games, have unique qualities that make them very challenging to existing game playing techniques, partly due to enormous branching factors. This remains a largely unexamined domain and is the arena we operate in. To attempt to tackle some of these daunting requirements, we introduce the novel concept of "*Representative*" Moves (RMs). Rather than examine the complete list of available moves at a given node, we rather propose the strategy of considering only a *subset of moves* that are determined to be *representative* of the player's strategic options. We demonstrate that in the context of a simplified Trading Card Game, the use of RMs leads to a greatly improved search speed and an extremely limited branching factor. This permits the AI player to play more intelligently than the same algorithm that does not employ them.

1 Introduction

The problem of playing a competitive board game, intelligently and effectively, against a human player is canonical in AI. Over the years, a broad range of literature has been published addressing the problem of game playing, introducing a wide range of highly effective techniques, for many different types of games [7–9]. Historically, the literature has emphasized classical, deterministic, two-player board games, and in particular, *Chess* and *Go* [7, 13]. AI systems for many such games are powerful enough to overwhelm even the best human players.

In recent years, perhaps influenced by their growing popularity in popular culture, card-based games such as Bridge and Poker have seen an increased emphasis in the literature [4, 5]. However, there are still some categories of popular games (particularly amongst "hobbyist" game players), which have been the subject of proportionally limited investigation within the field. One of these includes large-scale, complex, multi-player strategy board games, such as the popular Settlers of Catan and Carcassone [6, 12]. Another is the category of Trading Card Games (TCGs)[1], which are an interesting hybrid between classical board games and card games.

** The third author holds the positions of *Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR*. The third author is also an *Adjunct Professor* with the Dept. of ICT, University of Agder, Grimstad, Norway. The e-mail addresses of the authors are armando.h.taucer@gmail.com, andrewpolk@cmail.carleton.ca, oommen@scs.carleton.ca respectively.

[1] TCGs compose a market of over $600 million [2].

In TCGs, which include popular games such as *Yu-Gi-Oh!* and *Magic: The Gathering*, players typically construct their own decks, selecting a subset of potentially thousands of published cards, each with differing effects, and place them onto a dynamically constructed field of play, where they act as the pieces of the game. Despite their interesting qualities, they have seen almost no attention from AI researchers, and this can be primarily attributed to the fact that resolving them is far from trivial. Besides, they are not easily modeled using the traditional Mini-Max or Monte-Carlo sets of strategies.

Both large-scale, hobby board games and TCGs are characterized by extremely broad, complex game states, which are manipulated by players who are able to take *multiple* actions on their turn (often as many as their available resources allow), and consist of both deterministic and stochastic elements. These elements cause these games to have extremely complicated game trees. They, thus, pose a particular challenge to established game playing strategies applicable for deterministic/stochastic games. To address these challenges, in this work, we introduce the concept of *generalizing* available actions into *Representative Moves* (RMs), and consider *them* in our search. While this will necessarily lead to unrealistic look-ahead in the game tree, the impact in decision making is far surpassed by the benefits in execution time and available search depth.

The remainder of the paper is laid out as follows. Section 2 describes the established techniques upon which we base our work, and Section 3 describes the novel concept of RMs. Section 4 describes the simplified TCG that we will use in our model, and how RMs can be applied to it. Section 5 describes our experimental design, and Sections 6 and 7 report our results and analysis. Lastly, Section 8 concludes the paper.

2 Background

The vast number of established search techniques for adversarial two-player and multi-player games can be broadly divided into two paradigms, namely those of stochastic methods, such as the Monte-Carlo Tree Search (MCTS) algorithm and its many successful variants [1, 10, 12], and the deterministic methods, generally based on the well-known Mini-Max algorithm [7, 11]. Both paradigms are based upon an intelligent, efficient, and informed search of the game tree, which refers to the set of game states reachable by each player making legal moves to alter the board. Mini-Max based approaches generally search to a selected *ply* depth, pruning sections of the tree using methods such as the well-known alpha-beta strategy, which will not impact its end state, thus enabling a deeper and more effective search with available computational resources [7]. MCTS, and its many successful variants, such as the UCT algorithm, directs its search through random game playing, potentially weighted by learned or expert-provided strategies, and thus it determines the path to explore at each step *via* a bandit problem [1].

Both of these paradigms must adapt when they encounter cases of imperfect information. For example, if applied to the game of *Backgammon*, the search must account for the fact that the available moves to each player, and therefore the efficacy of specific strategies, are partly based upon a roll of the die. In both these paradigms, this can be handled by the incorporation of “chance” nodes, which represent the possible, weighted outcomes of a random event [7]. In the deterministic context, “chance” nodes are incorporated into the Mini-Max strategy in the Expectiminimax algorithm, where each node

is assigned a score equal to its *expected value*, assuming players perform intelligently with the resources available to them [7]. MCTS schemes can also incorporate "chance" nodes in the manner employed by the Expectiminimax [3]. Intuitively, the incorporation of "chance" nodes can seriously impact the performance of the search, as they lead to a much larger functional branching factor, and effective tree pruning, particularly in the deterministic case, where it becomes a much greater challenge.

While relatively simple random elements, such as rolling a die, can transform a simple search problem into a far more challenging one, these problems are exacerbated in the context of "hobby" board games, such as *Settlers of Catan*, and TCGs, which often have very complex random events and player moves made from of a wide range of individual parts. Current applications of game playing strategies to hobby games generally require a great deal of bias and expert knowledge to perform competitively [12]. This requirement necessarily limits the development of a domain-*independent* approach. In the context of TCGs, while a number of attempts have been made to play subsets, or accomplish specific tasks, in popular TCGs, to the best of our knowledge, no attempt to create a competitive TCG player exists in the literature. This is possibly because the number of available moves and random components involved are too daunting for current strategies, which do not rely heavily on expert knowledge. In this work, we will refer to these types of games as Complex Stochastic Games (CSGs).

As the extremely large number of "chance" nodes and branching factor of TCGs are a major hurdle in achieving competitive play, we suggest that an effective method for tackling them could be to simplify the game tree in some way. While this would have the effect of considering an invalid or incomplete game state, it may not be as catastrophic to do so as one may intuitively believe. This is analogous to what happens in the context of multi-player games. In recent years, the Best-Reply Search (BRS) has been shown to achieve excellent, competitive play, even through it explores invalid game trees [8]. The BRS is based on an observation that in a multi-player board game, the moves of the other players are not as important as the moves of the perspective player, and simplifies the turn order and search by considering all opponents as if they were a single "super-opponent", and searching as if the game had only two players [8] – even though it considers invalid turn orderings. An example of this grouping in a single level of a BRS tree is shown in Figure 1. We propose an analogous strategy for CGSs.

3 Representative Moves

While there are many difficult challenges to overcome in achieving competitive game play in the domains of TCGs and hobby board games, a critical concern that severely hampers state-of-the-art techniques is the extremely large branching factors of these games, due to the wide range of possible decisions available to the player, and the presence of a substantial number of "chance" nodes in the game tree[2]. While traditional pruning methods can impact the branching factor, if there are hundreds of possible moves available to a player at each turn, their applicability will be limited. Therefore, it is worthwhile seeking out novel and radical methods for limiting the search space,

[2] As an example, in a TCG deck, there may be twenty or more possible cards, each with a different functional purpose, which the player could draw on any turn.

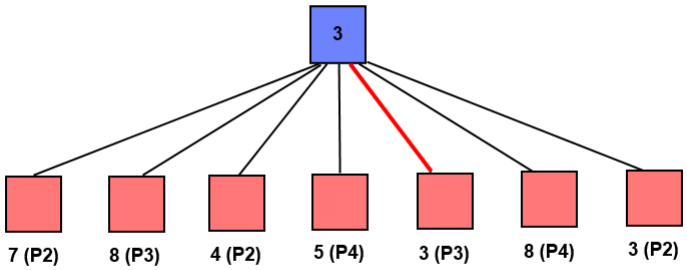

Fig. 1: The operation of a single level of the Best-Reply Search. The scores that are reported have the opponent's player number listed next to them (in parenthesis) to assist in the clarification.

to improve performance in these games. To achieve this, we propose the concept of *Representative Moves* (RMs).

The basic concept of RMs is as follows. In the scenario where a game, possibly, allows hundreds of moves, particularly when some are based on random chance, rather than considering all of these possible moves and their chance nodes, we, instead, consider a game model with only a much smaller set of moves. This smaller set of moves is "representative" of the total number available, standing in for "classes" of moves. They are either chosen representatives of their class, or some sort of average of the moves of that class. The classes of moves can be determined based on expert knowledge of the game's strategy, or potentially through unsupervised learning mechanisms. As a simple example, consider a game, similar to Backgammon, where during his turn, a player rolls two dice, and may advance two of his pieces by individual values shown on the pair of dice, towards a goal space. The player could move two different pieces, the same piece twice, reach the goal with one or two of his pieces, or perhaps, due to occupied spaces ahead, only move one, or even none of his pieces. These six types of moves could be the classes. Rather than consider all possibilities, we would consider these as RMs.

The main reason to do this is to limit the branching factor of the game. Instead of considering hundreds of available moves, we instead perform a search based upon distinct *strategies* available to the players at each level of the tree. Intuitively, by selecting only RMs at each level of the tree, we will be considering an incomplete version of the game tree, or, if averages are taken, potentially illegal board states. However, this is what the BRS does for multi-player games, and, despite this apparent weakness, it performs extremely well in the environments to which it applies [8]. When considering an exorbitantly large space, such as those associated with TCGs, this can potentially allow us to achieve a more substantial look-ahead, and faster search.

A disadvantage of using RMs is that a move, or random outcome, which is ignored, could be critical to formulating a winning strategy. Besides, it may not always be possible to properly select RMs. For example, in *Chess*, this technique would be unlikely to perform well, as it is difficult to strategically classify a player's available moves.

4 Game Model

The RMs paradigm was developed specifically to tackle the domain of TCGs, and therefore, it would be best to test its performance within the context of an actual TCG. However, due to the extreme complexity of these games, it is very difficult to implement an engine capable of fully capturing all their aspects. Therefore, in this exploratory work, we have created a custom TCG, designed to follow the conventions of the genre, while being a much simpler example than any on those available in market. TCG games generally represent a battle of some sorts, typically, in a fantasy or science fiction setting, with the names of cards, game mechanics, and so forth contributing to this story. Players draw cards from their own, customizable deck (with a unique set of rules for deck customization), and typically place these cards onto the "field of play", where they remain, functioning thereafter in a manner similar to game pieces, or have some effect and are then removed. For example, in a game like *Magic: the Gathering*, based on the concept of dueling wizards, a card could represent a dragon that could be summoned to battle, or a spell to destroy one of the opponents' summoned monsters. On each turn, a player can typically take as many actions as his cards would allow, plus some general-purpose acts, such as drawing a card from his deck.

In our custom TCG, each player has at their disposal a deck of twenty-five cards. As is typical of TCGs, each player has his own deck. However, in this case, the cards in each deck are identical. Two of these cards, when played, remove an opponent's card from the field immediately, and are then removed themselves; this is analogous to "Removal" cards in *Magic: the Gathering*. The other twenty-three represent "soldiers" of varying strength and quality, each having a value for "Fight", "Heal". and "Attack". Thus, each individual card can have different strengths and weaknesses. When played, these cards remain within the field of play, and afford the player additional actions on this turn and future turns, as is typical of TCG cards. The specific values of the "soldier" cards are chosen to provide an even spread of each value, with higher values being less common than lower values. The specific soldier cards are recorded in Table 1.

Table 1: Statistical values of cards in our custom TCG.

Fight	Heal	Attack	Fight	Heal	Attack	Fight	Heal	Attack
1	1	1	1	1	2	1	0	3
2	2	2	3	2	2	0	4	4
3	3	3	1	1	5	4	4	0
4	4	4	6	0	0	3	3	0
4	0	3	3	3	2	1	3	3
2	4	2	4	0	4	4	1	1
0	0	7	2	4	1	6	6	6
4	1	0	5	1	0	-	-	-

Each player begins with fifty "Health Points" (HP), conceptually representing his remaining vitality. To begin play, each player draws five cards. Unlike a normal TCG, for the sake of simplicity, each player's hand is visible. During each turn, the player must first draw a card, unless this would cause him to have more than five cards in his

hand. The player may then place a card from his hand onto the field of play[3]. Then, for each of the player's cards on the field, he may take one of the following actions:

1. Do nothing.
2. Increase his HP by the card's "Heal" value.
3. Decrease his opponent's HP by the card's "Attack" value.
4. Have this card do battle with another on the field. The card with the lower "Fight" value is removed. In the event of a tie, both are removed.

The first player whose HP is reduced to zero loses the game, and his opponent wins.

5 Experimental Design

While the concept of defining and utilizing RMs has intuitive value, its performance must be gauged in a quantitative manner so to ensure that the balance between its benefits and drawbacks supports its use in real game playing engines. The task of performing a formal analysis of game playing algorithms [11] is well-known to possess extreme difficulties. Consequently, as is the accepted practice in the literature, we have chosen to verify its performance via experimentation. In this exploratory work, we will perform such an experimental verification using the custom TCG[4] described above.

Our goals in verifying the performance of RMs are twofold. Firstly, we seek to confirm that the use of RMs does, indeed, save substantial amounts of search time. Secondly, we seek to determine whether this benefit in performance is obtained at the cost of effective game play, or whether this even improves the play. To do this, we first measure the execution speed of both a basic Expectiminimax algorithm, and also that of an Expectiminimax algorithm that employs RMs. More specifically, in this regard, we measure both the execution time, and the total number of leaf nodes expanded, where the latter serves as a "platform agnostic" metric. We then measure their performance against each other, over a number of games, to examine the impact of RMs on the win rate. We conduct this experiment both to an equivalent move depth, and thereafter to a greater depth whenever RMs are employed, if the latter is found to work faster.

To apply RMs to our custom TCG, we must first have some concept of how the classes of moves, and their representatives, are selected. Observe that in our custom TCG, each card (except the "Removal" cards) has a total of three values ("Fight", "Heal", and "Attack"), each of which represents a distinct strategy. Thus, when determining which card to play next, we consider three possible RMs, one for each of the three values. Each of these three moves has a value equal to the average of "Fight", "Heal", or "Attack", of all remaining cards in the player's hand and deck, with the value doubled for cards in the hand. This produces three "representatives", roughly corresponding to the player's current capacity in each of the three aspects of the game.

[3] In a typical TCG, the ability to put a card onto the field of play is limited by some form of resource, often obtained from other cards. Thus, careful deck construction is an important consideration, as powerful cards are of no strategic use if they cannot be played. As our game does not factor in deck construction, we have omitted resource cards from the game.

[4] We apologize for the detailed description of the game's moves, but it is necessary to clearly describe our work.

This leaves us with the task of considering the “Removal” cards which function differently and which do not have three values. To factor them into our “representative” set, we observe that a “Removal” card has a purpose similar to a card with a high “Fight” value, that is, that of removing the opponent’s cards from their field of play. We, therefore, consider each of the two “Removal” cards to be equivalent to a card with the values 5 “Fight”, 0 “Heal”, and 0 “Attack”. Observe that this can “remove” almost anything from the field, but can do so only a single time. This is thus is not quite as good as the few cards which have 6 “Fight” values. This allows us to consider the “Removal” cards to be part of our three representatives.

For the sake of fairness, each of the players makes use of the same evaluation function. The evaluation function takes the player’s “Health Points”, and adds to it the total sum of all values, on all cards, on the player’s field, divided by three. We subtract from this value a small factor which favours more “balanced” hands, thus encouraging the player to retain more strategic options. This factor is calculated by summing the three values on each card, and subtracting the lowest total value, from the highest (i.e., a card with “Fight” 3, “Heal” 1, and “Attack” 1 would have a sum of 5). The final value of the evaluation function is the difference between the perspective player’s result, and his opponent’s.

Our results are presented in the following section.

6 Results

Table 2 shows our execution time results for Expectiminimax when RMs were employed. In all cases, to ensure equivalent execution, each player had the starting hand 6/6/6, 2/1/1, 4/1/1, 2/3/2, and 1/3/3 (“Fight”/“Heal”/“Attack”). As expected from any game tree search algorithm, the runtime, and the number of leaf nodes examined, grow explosively as the *ply* depth is increased.

Table 2: Execution time when RMs are employed.

Depth	Runtime (ms)	Leaf Nodes
3	108	3,100
4	246	62,400
5	2,200	238,100

Table 3 shows our execution time results for Expectiminimax when RMs were not employed. The same starting position was used, as with the previous experiments, to ensure consistency. As is immediately obvious, the use of RMs *vastly* cuts down the search space, with an Expectiminimax search to a *ply* depth of 3, taking much longer than one to a depth of 5 when RMs were not employed. At a greater depth, execution of the search showed no signs of halting after well over 30 minutes of execution time.

Table 4 shows our results when RM-enabled and Non-RM players faced each other, with equal search depths of 3. We found that, even when they were allowed only equiv-

Table 3: Execution time when RMs are not employed.

Depth	Runtime (ms)	Leaf Nodes
3	132,100	263,836,900
4	Inf.	Inf.

alent search depths, RMs achieved an 80% win rate over a non-RM search, winning with an average of 22.9 HP, and taking an average of 29.6 turns to finish the game.

Table 4: Games between RM and non-RM players at equivalent search depth.

Winner	RM final HP	Non-RM final HP	Number of Moves
RM	25	-2	26
Non-RM	-3	28	24
RM	37	0	22
RM	59	-2	45
RM	6	-1	24
RM	21	-3	16
RM	20	-1	21
RM	40	-2	20
Non-RM	-5	11	42
RM	29	0	56

Table 5 shows our results when RM and Non-RM players faced each other, where the RM player was allowed to search to a depth of 5, given its proven efficiency. We found that, even allowed only equivalent search depth, RMs achieved an 72.7% win rate over a non-RM search, winning with an average of 19.9 HP, and taking an average of 26.2 turns to finish the game. The power of using RMs in the strategy is obvious!

7 Discussion

Our results very clearly demonstrate two things. First, the use of RMs significantly improves the speed and efficiency of the search. Secondly, despite considering only a small, representative portion of the complete game tree, not only does the use of RMs not hamper the strategic ability of the player in our custom TCG, but in fact, it achieves a noticeably higher win rate, compared to a player not using the RM technique.

Our first observation, about the speed of the search, is somewhat predictable. Indeed, the use of RMs converts a total of up to 25 possible cards that the player could place onto the field and which he must consider, into a total of three. However, it is still striking to observe how extreme the change is. Even comparing a search to a *ply* depth of five, using RMs, to a search to a depth of three, not using RMs, the RM-enabled search is two orders of magnitude faster. It also considers three orders of magnitude fewer leaf nodes. This result clearly demonstrates that the use of RMs can, indeed, enable a search to a much greater depth, in the same available processing time.

Table 5: Games between RM and non-RM players at differing search depths.

Winner	RM final HP	Non-RM final HP	Number of Moves
RM	8	-1	16
Non-RM	0	16	32
RM	54	0	33
RM	18	-1	21
RM	29	-2	17
RM	24	0	24
RM	8	-5	25
Non-RM	-2	42	26
Non-RM	-5	15	26
RM	41	-1	36
RM	44	-1	32

The second observation is far more interesting. Our original, stated goal was to demonstrate that the use of RMs does not *reduce* the strategic capabilities of the Expectiminimax algorithm. However, we observed that it, in fact, regularly defeated a player who does not use it. More importantly, this effect was observed even when the primary predicted benefit of RMs, i.e., the faster search, was not factored in, to allow it to search deeper in the tree. This is a very interesting and unexpected, result. A possible explanation for this is that considering only the primary strategic factors of the game, rather than individually considering every possible card, RMs enabled a more focused search on the game's components. The RM technique did not perform better at a larger search depth, although given the restricted number of games we could play to completion (due to how slow the non-RM player was), this may easily be due to random chance.

We observed that in both experiments, the RM player would, on average, end the game with a similar (20 - 23) HP score, and that the game would take a similar (26 - 30) number of turns to complete. Examining individual cases, however, we observe some scenarios where the game took much longer, up to 56 turns, to finish, and some scenarios where the winning player would have a much higher, or lower, HP score. The highest winning HP score was 59, and the lowest was 6. Given that our custom TCG provides the ability for cards to "Heal" the player, it is likely that the cards drawn in games that either took a long time, or ended with high HP scores, had a much higher "Heal" value than other available cards. Conversely, whenever the winner won with a low HP score, it is likely that both opponents entered into a scenario where they had cards with high "Attack" values, and attempted to race to defeat each other. This balance of aggressive and defensive play is typical of real TCGs, such as *Magic: the Gathering*, and suggests that our custom TCG does, indeed, capture some of their strategic qualities.

8 Conclusions and Future Work

Our results in this paper very clearly demonstrate that, although it considers only a portion of the game tree, the use of RMs is capable of achieving competitive game play,

with equivalent techniques that do not employ it. RMs are also capable of vastly improving the execution time of the search. Although we have done an examination of RMs in the context of Expectiminimax, it is also applicable to MCTS, and an examination of its capabilities in such settings is a possible avenue of future work.

In this work, we calculated the RMs based upon domain-specific knowledge of the game tree. However, as we briefly touched upon in Section 3, it would be possible to find classes of moves using an unsupervised learning technique, and elect specific representatives from the classes. Further, while our custom TCG replicates some of the strategic concerns of a real TCG, it is *much* simpler than the ones accessible in the market. It would be very interesting to examine the capacity of RMs in a real-world TCG, although significant work would need to be done in formalizing the game engine, before this would be possible. Finally, we believe that other non-TCG games, like *Settlers of Catan*, or *Backgammon*, could potentially benefit from the use of RMs.

References

1. S. Gelly and Y. Wang, "Exploration Exploitation in Go: UCT for Monte-Carlo Go," in *Proceedings of NIPS'06, the 2006 Annual Conference on Neural Information Processing Systems*, 2006.
2. M. Griepp, "Hobby games market nearly $1.2 billion." News article on hobby gaming's success, with notes on TCG market share, 2016.
3. N. Jouandeau and T. Cazenave, "Monte-carlo tree reductions for stochastic games," in *n Proceedings of TAAI'2014, the 2014 Conference on Technologies and Applications of Artificial Intelligence*, pp. 228–238, 2014.
4. M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, pp. 508–513, 2017.
5. T. Onisawa and T. Yono, "Construction of poker playing system considering strategies," in *Proceedings of CyberGames'06, the 2006 International Conference on Game Research and Development*, pp. 121–128, 2006.
6. M. Pfeiffer, "Reinforcement learning of strategies for settlers of catan," 2004.
7. S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, pp. 161–201. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 3rd ed., 2009.
8. M. P. D. Schadd and M. H. M. Winands, "Best Reply Search for multiplayer games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, pp. 57–66, 2011.
9. C. E. Shannon, "Programming a computer for playing Chess," *Philosophical Magazine*, vol. 41, pp. 256–275, 1950.
10. D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, pp. 484–489, 2016.
11. N. Sturtevant, *Multi-Player Games: Algorithms and Approaches*. PhD thesis, University of California, 2003.
12. I. Szita, C. Guillame, and P. Spronck, "Monte-carlo tree search in settlers of catan," in *Proceedings of ACG'09, the 2009 Conference on Advances in Computer Games*, pp. 21–32, 2009.
13. F. Xiao and Z. Liu, "Modification of uct algorithm with quiescent search in computer go," in *Proceedings of TAAI'10, the 2010 International Conference on Technologies and Applications of Artificial Intelligence*, pp. 481–484, 2010.