Adversary Modeling and Simulation in Cyber Warfare

Samuel N. Hamilton and Wendy L. Hamilton

Abstract Modeling and simulation provide many excellent benefits in preparation for successful cyber operations. Whether used for creating realistic training environments, testing new cyber warfare techniques, or predicting possible adversary actions, it is critical for such simulations to take into account the possibility of an active cyber adversary, able to adapt its plans to network conditions. Without real-time high fidelity modeling and simulation, training fails to address how to cope with intelligent and adaptive opponents, and operations become trial and error exercises rife with high-risk improvisation in situations where the adversary does not follow a well defined script. Unfortunately, current simulation techniques are insufficient to model adversaries capable of dynamic adjustment to changes in the simulation environment. Either adversary actions are completely pre-scripted, or live red teams are required to be on hand to tailor adversary actions to circumstances. In this paper, we present a technique for avoiding the prohibitive cost associated with requiring live red team participation during each use of a simulation environment while still providing the advantages dynamic adversary modeling provides. Our approach utilizes game theoretic techniques, using a new probability based search technique to curtail the search-space explosion issues that previous attempts in this area have encountered. This technique, entitled **Partially-Serialized Probability Cutoff Search**, also includes a new approach to modeling time, allowing modeling of anticipatory strategies and time-dependent attack techniques.

1 Introduction

The success of game theory in creating world-class competitors in domains such as chess, checkers, backgammon, and othello is well known. The strongest effect this

Distributed Infinity Inc., La Mesa CA 91941
e-mail: shamilton@distributedinfinity.com; whamilton@distributedinfinity.com

has had in the gaming community is not the entry of these programs in competition, but in helping human players with preparation and analysis. By considering huge numbers of possibilities, computers have been shown capable of finding exceptions to general rules and exploiting them mercilessly. In some cases, whole new theories on how to play are developed because of this [1]. We believe game theory will not ultimately replace the human analyst in the domain of cyber warfare, but can supply him with a powerful tool for suggesting approaches, techniques, or potential threats that might not occur to him, and provide the ability for better and more detailed analysis by simulating possible futures. In some sense, this can already be done, in that network simulation techniques have been dramatically improving in the last few years, allowing people to provide training in simulated cyber environments such as in [4], or to determine the efficacy of their tools and techniques in simulated environments before deploying them for network defense. What has been missing from these simulated environments, however, is the ability to actively model adversaries with the potential to sense, interact and plan based on the effects produced by the tools or techniques being tested. Even in [4], adversaries are simulated more in lines with Sim-City AI techniques than those applied to model sophisticated plan development such as used by chess players. This is understandable, since researchers have had much less success applying standard game theoretic techniques in this area, due to the number of complexities involved in modeling the cyber warfare game space compared to well defined games such as chess.

To overcome these challenges, we have developed a new search that addresses many of the fundamental problems with applying standard search approaches to the domain of cyber warfare. The search has the following characteristics:

- each player may simultaneously initiate multiple moves.
- moves have explicit time durations and affect state upon initiation, during, and upon completion.
- moves may have multiple possible outcomes from the same starting conditions.
- each player has a separate explicitly modeled view of the game space that may be inaccurate.
- each player has a separately modeled set of goals that may or may not come into conflict.

Each of these features are a distinct departure from previous approaches to game theoretic modeling, and would act to further exacerbate the search space explosion if not accompanied by the partially-serialized probability cutoff search presented here. Figure 1 presents a summary of the partially-serialized probability cutoff search technique utilized to curtail this search explosion. Results for this new technique, described in detail in section 4, have been gathered and published in a final report by the Disruptive Technology Office [16]. Success rate was measured by predicting incident response team actions during a red on blue cyber warfare exercise hosted by the Space and Naval Warfare Center. In over 70% of the scenarios, our search was able to pick the identical action chosen by the incident response team. A more detailed result analysis can be found in section 4.

2 Related Work

In this section we describe the current state of research in game theory, concentrating on areas relevant to tactical analysis in cyber warfare. There are two fundamental pieces to a tactical analysis engine: the search technique, and the evaluation function. The evaluation function reports how good a position looks to the player whose move it is.

The search technique determines which moves to consider, and returns the move or series of moves it considers best at the end of the search. The search forms a tree of moves and counter moves, and the leaves are judged using an evaluation function. While our search populates the tree with moves from both sides, in a majority of cyber warfare research to date the tree is populated by only moves by a single side. In this case, such as when using the planning techniques applied by Adventium Labs in [2], these techniques lack the ability to anticipate opponent actions. The same holds true for most attack graph generation techniques, such as those used in [3]. Such techniques can be quite useful for identifying security holes, since the depth of search enabled by simulating only actions by a single side is greater, but they do not provide the dynamic interactive plan generation necessary for simulating live network attackers or defenders.

When generating a tree populated by moves from two parties, the most popular technique is to use a derivative of the mini-max search [6], which propagates leaves back to the root by having parents alternatively assigned the minimum/maximum value of its children, ending with the root maximizing. This works under the assumption that ε_{p_1}, the evaluation function of player 1, is the opposite of ε_{p_2}, the evaluation function of player 2. Thus, $\varepsilon_{p_1} = -\varepsilon_{p_2}$. In other words, this assumes that what is bad for you is good for your opponent, and vice-versa. This assumption is clearly valid in normal game arenas such as chess and othello, where the resources, goals, and moves allowed by both parties are the same. These conditions do not hold in the domain of command and control planning. There are a variety of reasons

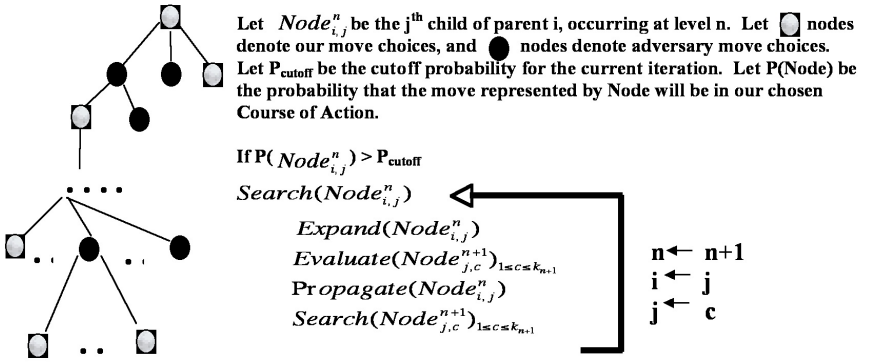

Fig. 1 Partially-Serialized Probability Cutoff Search

for $\varepsilon_{p_1} \neq -\varepsilon_{p_2}$. First, your opponent likely has different goals and priorities than you. Second, both sides are unlikely to have the same view of state. Even so, in the domain of command and control planning, there has been some limited success based on the assumption that , $\varepsilon_{p_1} = -\varepsilon_{p_2}$ [5]. We believe, however, that in the domain of cyber warfare improved results can be obtained by considering alternative evaluation functions.

To address this issue, our previous work section concentrates on three areas of game theory because of their particular relevance to cyber warfare. The first is pruning, as the difference between our evaluation function and an opponent evaluation function invalidates many of the traditional techniques. The second is opponent modeling, since it is necessary to define the opponent evaluation function. The third is tuning our own evaluation function.

2.1 Pruning

While we cannot use a mini-max derived search due to its reliance on the assumption that $\varepsilon_{p_1} = -\varepsilon_{p_2}$, we can use what we will refer to as a max-max' search, where nodes in the tree alternate between passing the maximum child according to the evaluation function of the player whose turn it is. Unfortunately, pruning techniques available in a max-max' search are much more limited than in mini-max searches, where alpha-beta pruning [6] can mathematically eliminate many possibilities. In alpha-beta pruning, moves are eliminated by showing that searching a move cannot change the outcome of the search. Unfortunately, in max-max' searches, it has been shown that no such pruning method is possible [7].

Since in any sufficiently complex game some type of pruning is necessary to limit the explosion of the search space with each increase in search depth, alternative techniques have been explored. One group of techniques is derived from best-first searches. In a best-first search, a tree of explored possibilities is kept open, and each leaf is judged as to how promising it is. The most promising node is then expanded, and its children added to the queue. In this scenario, the most promising move is defined as the most likely to be in the final predicted move group. The problem with best-first searches is that the memory requirements are explosive, so in a game where a large number of positions must be considered this solution is not feasible.

β-pruning [8] has been explored specifically with respect to max-max' searches. In β-pruning, the opponent evaluation function and search depth are assumed to be known, and the opponent is assumed to search using a mini-max based strategy. These assumptions allow the use of weaker versions of alpha-beta pruning. It is unclear how realistic these assumptions are unless playing against a known computer opponent running on inferior hardware.

Another search technique proposed for max-max' searches is $\alpha\beta*$ [7]. This technique assumes that the opponent uses an evaluation function similar to your own (i.e. it uses the same heuristics to judge a position) but weights them differently. The technique takes into account the fact that the opponent evaluation function is

not completely accurate, but assumes it is correct within a certain error range. Pruning is then done for values outside of that range, using techniques derived from the motivating principals of the alpha-beta search. We found the limitations on the evaluation function in this case to be too restrictive in the chosen domain, thus is $\alpha\beta*$ is not applicable to our search.

2.2 Modeling the opponent evaluation function ε_o

Most literature in this area assumes that the opponent uses some type of mini-max search that goes to a fixed depth [7, 8]. The problem then reduces to determining what that depth is, and what evaluation function is in use.

One approach to identifying depth and evaluation characteristics is to iteratively increase model depth, and use reinforcement techniques to learn which depth best predicts opponent behavior. If the evaluation function is given, this technique works quickly and effectively [8].

For the evaluation function, the usual assumption is that an opponent evaluation function uses a subset of the heuristics in our own evaluation function, with different weights. A hill climbing algorithm can then be used based on a function of the number of correct opponent move predictions taken from a list of previous opponent moves or games. A linear programming technique using pattern recognition methods was proposed in [11], and used with some success to find optimal weights in a follow up pass after hill climbing in the domain of checkers [10].

2.3 Determining self evaluation function ε_s

Over the years some very effective techniques have been developed for tuning evaluation functions. In [10], a method was proposed for tuning the weights in a list of heuristics in the domain of checkers. This turned out to be good enough to build a World Champion checker program. A world-class backgammon program used neural networks to learn using similar techniques [11]. In computer chess, Deep-Blue and its predecessor Deep-Thought rose to the top riding an evaluation function that was also automatically tuned [12].

While there are some differences between the techniques used in each case, there are some significant similarities. First, a human comes up with a list of heuristics expected to be correlated with success or failure. Then, a large number of master level games are analyzed at a low depth, and the best move is selected. This move is checked against the actual move played. If the move matches, the heuristic weighting is reinforced.

There has also been some work on automated methods for heuristic generation in the domain of othello [13]. This work was successful in finding valid heuristics, though they did not actually improve the program results since the slow evaluation

function was more of an impediment than the improved accuracy could compensate for. A method of linearly combining Boolean feature lists into heuristics and then weighting them has also been explored [14].

In the domain of command and control planning, there are no large databases of games at any level, and the heuristics are constantly changing. Thus, within this domain this approach is currently untenable as a viable solution.

3 Partially-Serialized Probability Cutoff Search

We have developed a new search that addresses many of the fundamental problems with applying standard search approaches to the domain of cyber warfare. Fundamentally, the primary challenge in this domain is that the search space is significantly more complex than traditional games, and a number of characteristics of this domain not modeled in traditional game theory approaches (time, simultaneous moves, stochastic move outcomes, non-symmetric player goals) either threaten the validity of the approach if not modeled, or exacerbate the search-space explosion if modeled. A more complete enumeration of these challenges can be found in [15].

The approach presented here addresses these issues by introducing a pruning method that enables significant improvements in tree analysis, sufficient to model human blue and red team[1] reasoning. It does this while simultaneously blending the state and move representation issues listed above that normal game theoretic approaches ignore.

Figure 2 shows the basic architecture of the search engine. The output of the engine is a course of action recommendation, which consists of a series of actions and expected adversary actions, with timing sequence information included. The output will be in tree format, and response suggestions to non-primary adversary actions are included in this tree.

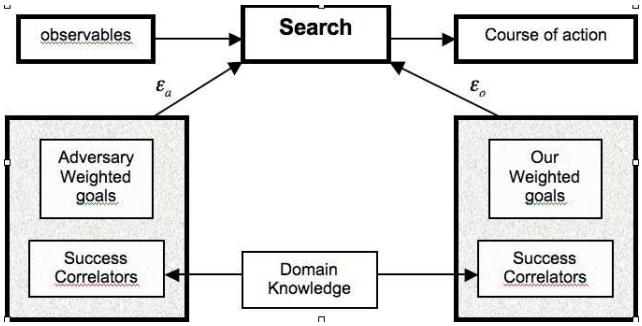

Fig. 2 Search Architecture

[1] For those not familiar with this terminology, a blue team is assigned a defensive role, and a red team is assigned a mission in conflict with the blue team goals.

The input to the search comes from three sources. The first is a list of observables, which is information about the perceived state of the network. This information may include inaccuracies, and is in fact expected to contain some in certain cases as moves by both parties may be designed to mislead adversaries by producing inaccurate observables.

The second and third input sources are in the form of evaluation functions, one representing our mission and associated goals, and one representing a hypothesized adversary's mission and associated goals. Note that there is no linkage between our evaluation function and that of the adversary. This means that while the search uses both functions, neither evaluation function has direct knowledge of the goals of the opponent. Each evaluation function is generated by weighting each goal, and applying these weights to state characteristics correlated with achieving these goals (taken from a database derived from domain knowledge expertise).

In the work presented herein, the concentration our research has been on the development of the search itself, not on the design of the evaluation function, which was done only as a pre-requisite for gathering meaningful results when comparing our search effectiveness to human analyst teams. Evaluation function design is a very interesting area in itself, and it is our intent to pursue this area further in subsequent work. For example, multiple hypothesized adversaries could be generated and modeled simultaneously using a modified version of this algorithm, with a worst-case linear increase in the search time with respect with the number of adversaries modeled. In cases where adversary goals overlapped, the increase would be much less. How to best generate adversary hypotheses could also be a fruitful area of future research.

In the next two sections, we will describe how a game tree is formed and the search algorithm applied.

3.1 Move Definition and Game Tree Construction

A cyber warfare move is defined to have the following characteristics:

- A list of preconditions.
- Effect on state upon initiation.
- Effect on state upon completion.
- A list of conditional effects during execution.
- Timing information for the entire move and each effect.
- A list of possible outcomes and probabilities for each effect.

Note that one of the possible effects is to generate an observable for one or both players, which is what changes their perception of state.

In addition to explicitly modeling timing effects and stochastic move outcomes, our move set differs from traditional game move sets in another fundamental way. In a traditional game, two opposing players alternate moves. In most real-world domains such as cyber-warfare, this is simply not the case. Each player has the

option of choosing multiple moves that are executed simultaneously. In fact, both players will frequently be executing multiple actions at the same time.

To accommodate this, our search utilizes an untraditional approach to tree construction. All simultaneously chosen moves are serialized, such that they are listed in order in the tree despite the fact that they will be executed at the same time.

This is accomplished by introducing a new move type: Pass. A pass indicates that the player will not be choosing to begin any additional actions until the next time slice. All moves chosen before a pass are interpreted as beginning at the same time. Figure 3 shows a simple example of this.

When there are no players left to choose a move at a particular time slice, each action chosen is entered into a queue of actions that have been initiated but not completed, and time is advanced. Time is advanced to the next "interesting time" which is defined as the first of the following events.

- a move in the action queue completes
- a move in the action queue produces an observable
- a predefined objective/goal relevant time is reached
- a maximum Pass time is reached

While each move has a duration and set of possible outcomes associated with it, both players may or may not be aware of these outcomes. Awareness of the state of the network is based on available resources, and may be contingent on making moves to gain information. Even when a move produces an observable, such as a message logged by a deployed Intrusion Detection System, the players may not be in a position to see the observable without further action. In the event that a player can see the observable the system will give that player a chance to see and respond to the observed event.

Note that both players are not necessarily given the option of moving during a particular slice of time. Players only move if one of the defined events occurs such that they are aware of it. Thus, if the defender completes a move, the defender will have the option to choose more moves but the attacker will only have that option if the event has produced an observable.

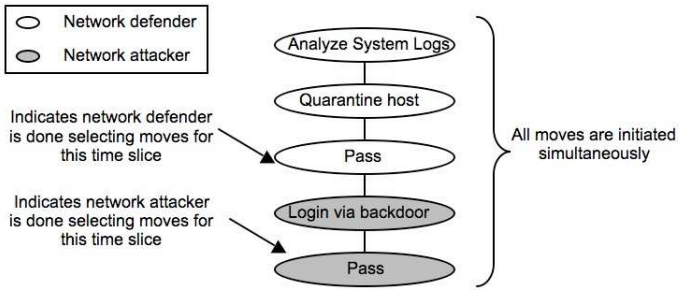

Fig. 3 Snippet of a Partially-Serialized Game Tree

3.2 Search

The Partially-Serialized Probability Cutoff Search is designed to deal with a much more complicated search environment than in most traditional games. Instead of having a single game space represented, we have five. Attacker Opinion, Defender Opinion, Attacker Opinion of Defender, Defender Opinion of Attacker, and Ground Truth. Each move updates these states independently depending on the observables produced over the duration of the move. In addition, each player may have asymmetric goals and objectives. To further complicate things, moves can be defined to have multiple stochastic outcomes. The benefit of these five opinions of state over traditional methods is increased realism, since in the real world the attacker and defender rarely know for sure the true state of things.

The interaction between the two players and the Ground Truth value is worth expounding upon, because of the significant impact it has on the search results. While it is true that neither evaluation function has explicit knowledge of the adversary's evaluation function, there is an interaction between these two functions through observables generated in the Ground Truth state representation and propagated to both parties within the search. That means that the course of action generated for player 1 does take into account actions that player 2 (given its evaluation function) might want to take for those actions that produce observables. The result of this is that while the search does effectively generate moves that can foil multiple courses of action an opponent might take to reach its objectives, it does not currently consider methods for foiling multiple possible adversaries. While we consider this an extremely interesting direction to explore in the future, we view the current approach sufficient for current training and simulation needs, for which even modeling a single group of sophisticated adversaries united in a single set of goals would be a major leap forward. Until the issue of efficient multiple adversary support is addressed, we have found it possible to roughly simulate this scenario by designing adversary evaluation functions that weight numerous separate goals and objectives such that the goal set of a single adversary becomes a super-set of the goals of a hypothesized group of adversaries.

To address the multitude of challenges presented by maintaining an ongoing interaction between a complex move representation and five separate states , we designed a search capable of taking both players separate perspectives into account. The search does not rely on depth cutoffs, which would interact harshly with the partially serialized game tree. Instead it utilizes a probability cutoff. The search will continue to expand all nodes that have a probability equal to or higher than the probability cutoff. The probability cut-off is then iteratively decreased, improving the quality of the search continuously and updating the user with an improved set of analyses after each iteration.

Probabilities are not expert defined (except in the case of stochastic move outcomes) but are instead generated automatically within the context of the algorithm. The algorithmic search is executed as follows:

1. generate children of node n for player whose move it is in the partially serialized tree.
2. generate position value for each child from each player's perspective
3. propagate values up the tree
4. propagate probabilities down the tree
5. select a child whose probability is above the cutoff and repeat

The memory requirements of the search are linear with respect to the depth of the tree, as is the amount of time to analyze a single node. Thus, for roughly balanced trees, these characteristics are approximately logarithmic with respect to the number of nodes. Values are calculated for leaf nodes n (such as when a child is first generated) as follows:

$$V_{n,1} = \varepsilon_{p_1}(S_{n,p_1,p_2})$$

and

$$V_{n,2} = \varepsilon_{p_1}(S_{n,p_1,p_2})$$

Where $V_{n,1}$ is the value for player 1 from player 1's perspective, $V_{n,2}$ is the value for player 1 from player 2's perspective, ε_{p_1} is the evaluation function of player 1, and S_{n,p_1,p_2} is a state estimate at node n for player 2 from player 1's perspective. Note that the evaluation function results are a product of state estimates, which may include values related to anticipated adversary goals. In this fashion, it is not necessary to model all characteristics of an anticipated adversary's goals within the evaluation function as it can be dynamically hypothesized and weighed within the state representation given previously encountered evidence.

The values are propagated up the tree using one of two methods. A node with a stochastic outcome has a child with each outcome, and the value is propagated for each player p as follows:

$$V_{n,p} = \sum_{i=1}^{c} V_{i+j,p} \cdot \rho_{i+j,p}$$

Where c is the number of children, j is the starting node number of the first child, and ρ_{i+j} is the probability of the outcome of child $i+j$. Note that the probability can be different for both players as their opinion of the stochastic nature of the result may differ. The linear weighting by estimated probability implicit here is likely non-optimal, but was chosen for the purposes of speed of calculation.

If the children of a node are choices of player p_1, then values are propagated by:

$$V_{n,1} = \max_{i=1}^{c}(V_{i+j,1})$$

and

$$V_{n,2} = \max_{i=1}^{c}(V_{i+j,2})$$

This is based on an assumption by both players that the player whose move it is will be optimizing their own result. Note that both players may have very different perspectives on what state is, so these results may be very different.

Propagation of the score goes up the tree, and upon reaching the root node triggers the calculation of probabilities. Probabilities of node n with parent σ and sibling children i through j are calculated as follows:

$$\rho_{n,1} = normalize(V_n, \{V_i \ldots V_j\}) \cdot \rho_{\sigma,1}$$

and

$$\rho_{n,2} = normalize(V_n, \{V_i \ldots V_j\}) \cdot \rho_{\sigma,2}$$

By normalizing based on the value, we ensure that moves with higher scores from the perspective of the player who is choosing the move have higher probabilities than lower valued moves. Multiplying by the parent probability ensures that the probability of children nodes always add up to the probability of the parent, from both players perspective. Of course, the probability of the root node for both players is 1.

Note that the normalization technique chosen can have a significant effect on the results of the search. If the result of normalization generally produces values with a small range, the result is a broad shallow search tree, where all possibilities are explored to relatively the same extent. If normalization generally results in a large spread between the highest values and the smallest values, then the search tree is longer and skinnier, such that high probability moves are deeply considered while other possibilities are explored more shallowly.

Figure 4 shows an example tree generated using this technique. Each node is annotated with two scores, one from the perspective of player 1, the second from the perspective of player 2.

The propagation of probabilities down the tree is what enables our aggressive pruning technique. By using these probabilities instead of arbitrary depth values to direct our search, the search trees generated become deeper and less broad, more

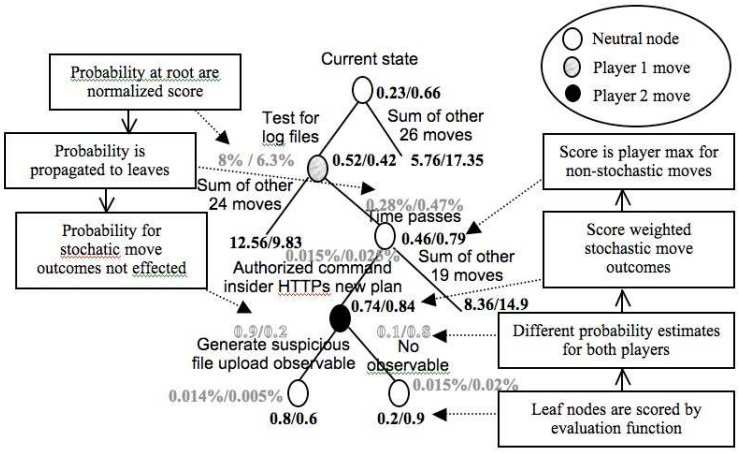

Fig. 4 Snippet of a Partially-Serialized Game Tree

similar to human generated attack trees than most computer generated trees. Of course, since the search is extremely fast, unlike human generated trees, they can contain hundreds of thousands or even million of nodes.

4 Results

The search described above was implemented using a move set developed by the Disruptive Technology Office (formerly ARDA) during the Cyber Strategy and Tactics Workshop. This workshop, held at the Space and Naval Warfare Center (SPAWAR) in San Diego, conducted attacker - defender cyber warfare exercises where the blue teams (defenders) were pitted against red teams (attackers) in a series of three cyber warfare scenarios. The missions and scenarios tested included timing based attacks, which exploited the need for certain cyber resources to be available during particular times. Any automated search approach that did not explicitly model time would be severely challenged to perform in these situations. The attacker and defender evaluation functions were hand-crafted before the experiment was begun, using input from members of both the blue and the red team regarding possible goals, and network state characteristics correlated with achieving success. Software was developed that allowed the blue team and red team to sit in separate rooms, enter their chosen courses of action, and simulate the effect of those actions which would then effect what both teams would see in terms of future observables. Noise events were also simulated, to help disguise which observables were the result of actions by the other team, and which were non-malicious events caused through normal network use (or misuse) by valid operators.

The original experimental design was that the blue team would compete head to head defending the network against our algorithm, with a pre-defined algorithm for success disclosed apriori to both parties. The blue-team experts objected to this experimental design, however, since they felt that defining success in terms of an explicit algorithm would give the computer program an unfair and unrealistic advantage, as it would be able to ”game the system.” Instead, blue team results were gathered against a live red team using several different attack scenarios, and the blue team decisions were labeled the ”gold standard” against which our algorithm was tested. Thus, success for our algorithm was to duplicate the decisions made by the blue team at each critical juncture. There were three blue teams composed of two members each, a cyber security researcher and an experienced cyber defender. These teams were given an unlimited amount of time to discuss their decisions at each junction to maximize the quality of their decisions. Each situation they faced was fed to our algorithm as well, under a strict time limitation of ten seconds per decision. Figure 5 shows our results. In the figure, first indicates that the blue team move choice was ranked first by our algorithm. Second indicates that our algorithm ranked the blue team move as the second best option. On average, at any given point there were over fifty possible choices.

In general, the search proved highly successful, choosing the same move as the blue team over 70% of the time. The remaining cases were shown to blue team members for discussion. A large majority of these cases fell into one of two categories. The first category (roughly 14% of the cases) consisted of situations where the search listed the move chosen by the blue team as one of the top six moves, with a top move whose effect was largely similar to the move chosen by blue. For example, the blue team chose to ”monitor logs in real-time” while the search selected the ”monitor host real-time” move. The second category of moves was the set of moves where the machine selected a more aggressive course of action than the human blue-team. The general conclusion of the blue teams was that during the exercise, the machine-selected moves would have been more effective at catching the attacker, but would have been over-reaction in real-life.

While the conclusion of this experiment with regards to the effectiveness of the search was extremely positive, a number of suggestions were captured in post-experiment interviews with the blue team members. The general conclusions regarding the experimental conditions were:

1. **Larger move set needed.** The blue teams in particular felt that larger numbers of moves were needed. We have subsequently tested our approach with over 6 times as many move types defined and believe the probability search scales well in this area, but have not repeated a full-scale study with the larger move set.
2. **Effective training environment.** The time model, particularly the effect of the ’Pass’ move was difficult for some participants to understand when initially explained, but was intuitive and worked well when they actually saw it in action after play began. The general consensus for both blue and red teams was that the experimental setup would be very effective for training.
3. **More realistic noise needed.** At first, the noise moves were effective, but after a while the human teams were able to reliably identify what was an adversary driven observable, and what was computer generated.

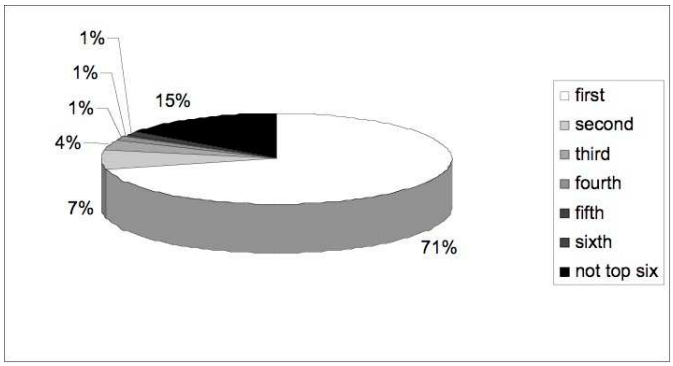

Fig. 5 Search Results Compared to Human Blue Teams

Overall, we considered the experimental results highly gratifying. While it was clear that at the time of the experiment we did not have a mature, ready-to-deploy piece of training software (nor was that the expectation of any of the parties) the capabilities of the search did receive a thorough comparison with highly trained experts in the cyber warfare domain, and proved extremely effective. This is the first example of an automated course-of-action generation algorithm being so tested within this domain that we are aware of.

5 Conclusion

In this paper, we have presented a new type of search designed for application to the cyber warfare domain. The Partially-Serialized Probability Cutoff Search we have developed has proven extremely effective on the benchmarks tested, successfully modeling the dynamic interaction of network attackers and defenders well enough to suggest the same course of action as domain experts over 70% of the time. It does so without the limitations of most search based approaches by allowing multiple simultaneous moves, asymmetric goals, inaccurate state perception by both attacker and defender, explicitly modeling time, and allowing for moves with multiple stochastic outcomes. This strongly differentiates our approach from all other published approaches to date.

The current version of the search is well adapted to dynamically simulate either network attacker or defender activities, and could be easily adapted to provide an automated red-team capability. Currently, the reasoning model has only been tested in two sided warfare situations, where one side is defending a set of resources, and another side has a mission that includes interfering with one or more of the protected assets. What have not been actively explored are situations where there are more than two active parties. Note, each active party could have thousands of resources, including large networks of computers and cyber warriors of varying degrees of skill, and still fit within the models presented here. As long as the ultimate goal of each member of the party remains roughly the same, simulation of an adversary can remain at the two party level without increasing complexity. When it becomes necessary to model more than two sets of goals, however, the modeling complexity can significantly increase the search space. For the purposes of most simulation needs, this is currently unnecessary, as the complexity of two parties in cyberspace is sufficient for most training or testing purposes. When modeling coalition environments with multiple interested parties, such as may be of interest at the national level, it may be necessary to model significant numbers of clashing goals and interests. We believe this to be possible, but anticipate future work in this area will be necessary to overcome these challenges.

References

1. Tesauro, G.: Temporal Difference Learning and TD-Gammon. Communications of the ACM. **38(3)**, 58–68 (1995)
2. Body, M., Gohde, J., Haigh, T., Harp, S.: Course of Action Generation for Cyber Security Using Classical Planning. ICAPS. (2005)
3. Wang, L., Noel S., Jajodia, S.: Minimmum-Cost Network Hardening using Attach Graphs. Computer Communications. **29(18)**, 3812–3824 (2006)
4. Cone, B., Irvine, C., Thompson, M., Nguyen, T.: A Video Game for Cyber Security Training and Awareness. Computers and Security. **26(1)**, 63–72 (2007)
5. Katz, A., Butler, B.: "Game Commander" - Applying an Architecture of Game Theory and Tree Lookahead to the Command and Control Process. Conference on AI, Simulation and Planning. (1994)
6. Winstron, P.: Artificial Intelligence. Addison-Wesley (1992)
7. Carmel, D., Markovitch, S.: Learning and using Opponent Models in Adversary Search. Technical Report CIS9606. (1996)
8. Donkers, H. et al.: Implementing β-pruning Opponent-Model Search, Technical Report CS 00-05 IKAT, Universiteit Maastricht, Maastricht, The Netherlands. (2000)
9. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley and Sons (1973)
10. Samuel, A.: Some studies in Machine Learning using the Game of Checkers.: IGM Journal of Research and Development **3(3)**, 211–229 (1959)
11. Tesauro, G.: TD-Gammon, a Self-Teaching Backgammon Program, reaches master-level play.: Neural Computation **6(2)**, 215–219 (1994)
12. Hsu, F. et. al.: Deep Thought. In T. A. Marsland and J. Schaeffer (eds.) Computer Chess and Cognition pp.55-78. Spinger Verlag (1990)
13. Buro, M.: Statistical Feature Combination for Evaluation of Game Positions.: JAIR **3**, 373–382 (1995)
14. Utgoff, P.: Constructive Function Approximation.: Technical Report 97-4, University of Mass. (1997)
15. Hamilton, S., Miller, W., Ott, A., Saydjari, O.: The Role of Game Theory in Information Warfare.: The Information Survivability Workshop (2001)
16. Meyers, K., Saydjari, O.: ARDA Cyber Strategy and Tactics Workshop Final Report. (2002).