In the AAAI 93 Fall Symposium – Games: Planning and Learning

Designing A Computer Opponent for Wargames: Integrating Planning, Knowledge Acquisition and Learning in WARGLES

Michael R. Hieb **David Hille**[1] **Gheorghe Tecuci**[2]
Center for Artificial Intelligence, Department of Computer Science
George Mason University, 4400 University Drive
Fairfax, Virginia 22030-4444 USA
hieb@aic.gmu.edu hilled@anser.org tecuci@aic.gmu.edu

Abstract

This paper presents a general approach to designing a computer opponent for nondeterministic adversarial games which is able to learn from interaction with a human expert. This approach is based on an integration of planning, knowledge acquisition and learning. We illustrate this approach with WARGLES (WARGame LEarning System), which plays strategic level wargames. WARGLES consists of a game playing system, a computer opponent and a learning component for the computer opponent. The computer opponent utilizes an adversarial planner with an incomplete and partially incorrect knowledge base. The learning component revises this knowledge base to improve the quality of play of the computer opponent based on interactions with an expert.

Introduction

A wargame is a simulation of military operations involving two or more opposing forces using rules, data, and procedures designed to depict real life situations. A wargame does not involve actual military combat and is used to learn about a conflict without actually having to fight it. The objective of playing a wargame is to practice the execution of operations and to gain insight into the nature of a conflict. A wargame allows the players to explore the effects of their decisions, and predicts likely results of policies and procedures.

A wide range of wargames are utilized in military analysis (Davis, 1986; Perla, 1990; Dunnigan, 1992). The domain of wargames is complex because wargames correspond to actual military operations. A special type of wargame involves a computer opponent that plays against a human or computer adversary, in contrast to non-interactive simulations or role-playing games involving only humans. The current state of the art in commercial wargames for personal computers is such that an expert player can learn relatively quickly how to defeat a computer opponent. This is partially due to the fact that, in general, the computer opponent in such games does not learn. Instead, it follows a fixed problem solving strategy or a small set of alternative strategies which could easily be discovered by a human player.

Learning methods have been designed for other classes of games. For instance, Samuel's checkers player (Samuel, 1967) is one of the first successes in machine learning. Recent systems have employed multistrategy learning techniques as in MORPH in the area of chess (Gould and Levinson, 1993) and HOYLE in the area of Morris games (Epstein, 1992).

In this paper we present initial results on developing WARGLES (WARGame LEarning System), a learning and problem solving system that plays a class of games called strategic wargames. WARGLES is composed of a game playing system, a knowledge-based computer opponent and a learning component which learns during contests with a human expert or from transcripts of prior contests. To reason at a strategic level, WARGLES' computer opponent plans its actions and uses plausible reasoning to deal effectively with the nondeterministic aspects of the game.

Current monostrategy machine learning techniques are difficult to apply to general problem solving situations encountered by WARGLES because they are too narrowly focused. Therefore, WARGLES employs a multistrategy task-adaptive learning (MTL) method based on building plausible justification trees (Tecuci, 1993). The system is able to adapt a learning strategy or a combination of strategies to the learning task, defined by the available input knowledge, the learner's background knowledge and the learning goal (Michalski, 1993). While our long term goal is to have unsupervised learning during a contest, the current learning method involves a human expert, in an apprenticeship learning scenario of the kind employed by DISCIPLE (Tecuci & Kodratoff, 1990).

WARGLES is the first step in a research project aimed at designing a general methodology for multistrategy task-adaptive learning and problem solving for the class of nondeterministic adversarial games. Complexity in these games is due to factors such as the uncertainty of the outcome of actions, the difficulty of predicting the adversary's actions, and the vast search space of the game.

The rest of the paper is organized as follows. The next section gives the specific details of the class of games WARGLES plays and compares WARGLES to other games. Section 3 describes the architecture of WARGLES and its integration of planning and learning. Section 4 presents an example of a contest showing the computer opponent planning and learning. Section 5 relates WARGLES to other computer game research. The last section describes the strengths and weaknesses of the game playing and learning approach presented, and discusses future research.

---

[1] Also with ANSER, Arlington, VA 22202

[2] Also with Romanian Academy, Bucharest, Romania

WARGLES Game System

WARGLES - The game WARGLES plays is a simple military simulation between two countries conducted on a N by N grid of squares. Figure 1 shows an initial board setup for a scenario WARGLES might play, with a 20 by 20 grid. Each player has a capital city and commands a number of units representing armored and infantry army divisions. Infantry units are represented by rectangles with crosses and armored units are represented by rectangles with circles. Infantry units can move up to two squares per turn and armored units can move up to three squares per turn. Units can move in all 8 directions, like the king in chess. Only one unit can occupy a square.

In Figure 1 the east player's capital city and all of its 12 units are on the left side of the board, facing the west player, who has a similar configuration. The objective of the game is to occupy the capital of the opponent by moving a unit into it. Play alternates between the two players in the form of turns. During a turn players may move any or all of their divisions and may use them to attack enemy units. Each unit projects a *zone of control* into the 8 squares that surround it. A unit may not move over another unit nor may it move directly from one square in an enemy zone of control to an adjacent square in an enemy zone of control.

A unit may attack an enemy unit if the unit is adjacent to it. There are three possible outcomes of an attack: the defending unit may be forced to retreat; the defending unit may be destroyed if it is forced to retreat but cannot retreat; or if the attack fails, nothing happens. The probability of success when a unit attacks another unit is known to the players. These probabilities can be varied to achieve different scenarios. In the game illustrated throughout the paper the probability that an attack by a unit against another unit of the same type will succeed is 0.4. The probability of success for an attack by an armored unit against an infantry unit is 0.6, while the probability that an attack by an infantry unit against an armored unit will succeed is 0.2.

If a defender is forced to retreat, it must move to one of three squares away from the attacker. The attacker then occupies the square vacated by the defender. If the defender is unable to move away from the attacker, the defender is destroyed.

WARGLES is similar to a simplified version of the Tactics II board wargame, one of the first widely available board wargames [Avalon Hill, 1958].

Relationship to Other Games - The games that WARGLES is concerned with differ from chess in that they are nondeterministic and the player can move none, some or all pieces in one turn, as opposed to chess, where a player can only move one piece per turn. These differences increase the search space of moves compared to chess. In chess there are 20 different moves possible in the first turn. In contrast to this, the number of different moves a player may make in the first turn of a WARGLES type game is $O(10^{26})$, if the player has 12 units. Because of the vast search space in this class of games, traditional minimax search techniques used in games such as chess may not be applicable.

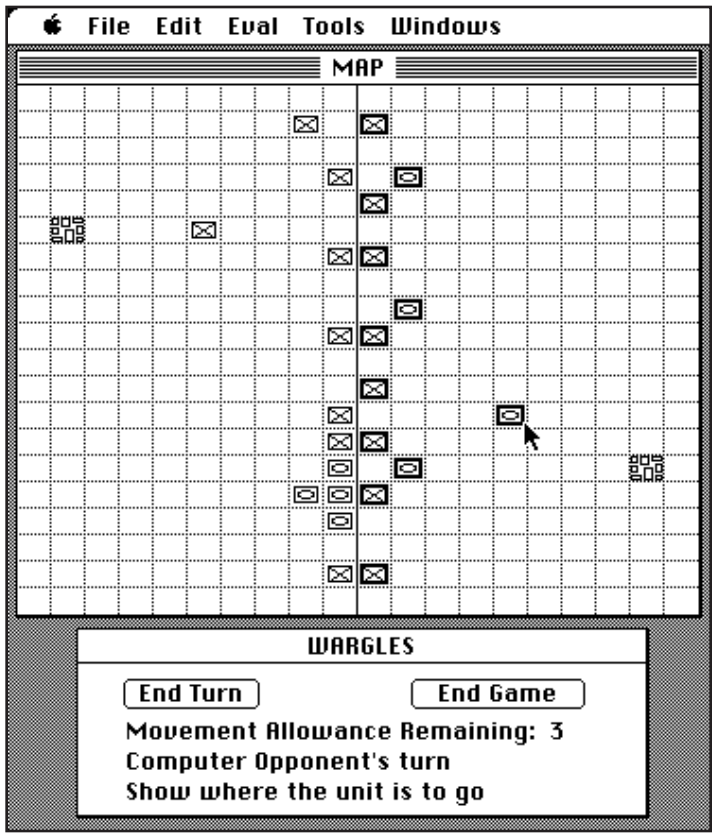

Figure 1: WARGLES Board

Backgammon, like WARGLES, is also nondeterministic and has similar elements of strategy, where pieces are captured and a player moves the pieces towards a goal (home). However, one can move at most 4 pieces during one turn. Due to the 21 different outcomes from the throw of a pair of dice and the 20 different ways to play the pieces (moves), there are about 400 possible moves per turn (Berliner, 1980).

A game with a search space similar to that of WARGLES class of games is Diplomacy. The number of possible opening combinations for six players for the first turn is $O(10^{16})$(Hall & Loeb, 1992). Diplomacy has such a large search space since, as in WARGLES type games, a player may give orders to any of a group of units. Also, a computer opponent in Diplomacy must have a good static evaluation function, since deep searching is difficult due to the computational complexity. Like Diplomacy, the WARGLES class of games simulate real processes, and are less abstract than many other classes of games.

WARGLES Architecture

The overall architecture of the WARGLES system is shown in Figure 2. It consists of 3 modules and an archive of past contests. The modules include a game system which provides a game interface and enforces the rules of the game, a knowledge-based computer opponent, and a

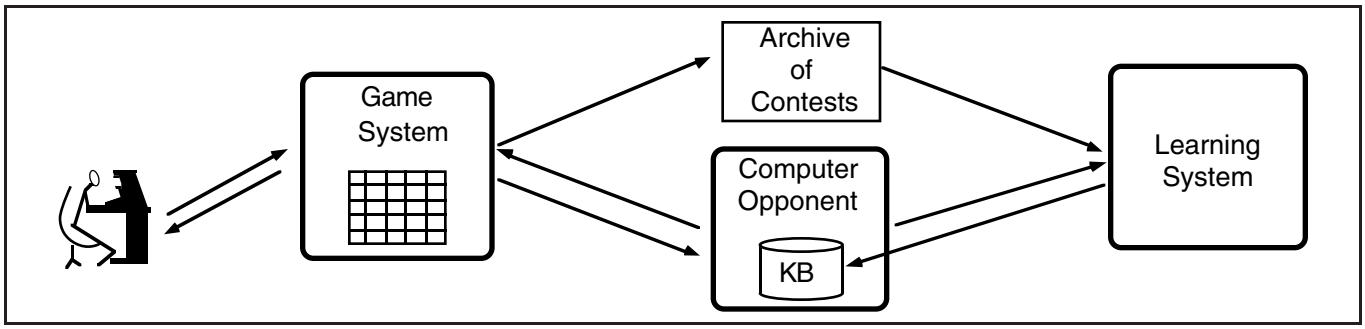

Figure 2: WARGLES Architecture

learning system which revises the knowledge base of the computer opponent. WARGLES is implemented in Common Lisp on the Macintosh.

The WARGLES game system is designed to be capable of accommodating either human or computer opponents. A transcript of each contest is stored in an archive file. This transcript contains all the information necessary to analyze the game played (including the decision process of WARGLES). In particular, a line of the game transcript represents a state change consisting of:

- the player (P_i);
- the action taken (A_i);
- the game state (S_i) prior to the action A_i;
- the game state (S_{i+1}) after the action A_i;
- the set of goals $G=\{G_i...G_k\}$ used by the planner to determine A_i.

WARGLES Computer Opponent - The computer opponent in WARGLES is shown in Figure 3. An adversarial planner is utilized with a knowledge base containing a goal tree consisting of goal nodes (see Figure 7). Inference rules in a knowledge base determine the values of various parameters of the goals. A plausible reasoner controls the inference during plan generation. A goal node has parameters denoting its feasibility, satisfaction, and method of execution. An example of a goal node is shown in Figure 4.

WARGLES uses a general-purpose planning system,

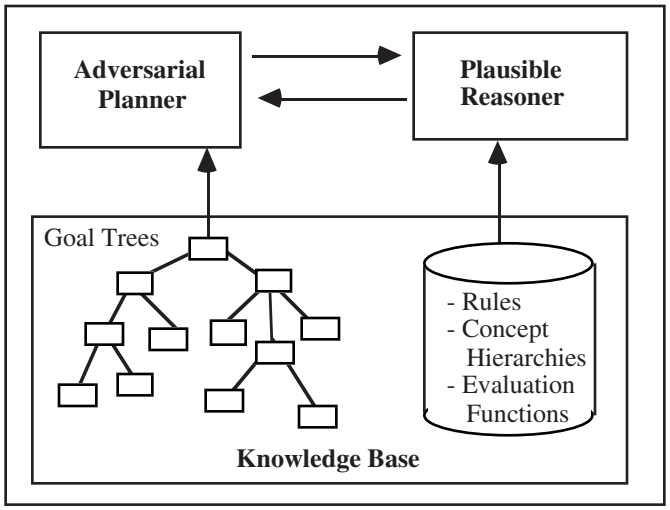

Figure 3: WARGLES Computer Opponent

Contingency Planner 2 (CP2), adapted to the knowledge base of WARGLES. CP2 (Young & Lehner, 1986; Lehner, 1990) is a knowledge-based system that consults goals and related information contained in the knowledge base to create a plan of action.

GOAL NODE	
Name:	Win-game
Subgoals:	Occupy-enemy-capital, Protect-own-capital
Preconditions:	None
Alternatives:	None
Feasible:	Not Capital-occupied(Own-capital)
Satisfied:	Capital-occupied(Enemy-capital)

Figure 4: Goal Node Parameters

The CP2 planner generates a plan for a turn by evaluating the actions of a proponent and the counter-actions of its adversary. To build a plan, CP2 matches each goal of the proponent (the computer opponent) to a counter-goal of its adversary. For each combination of goals, CP2 determines actions of the proponent and adversary to accomplish their respective goals, projects possible states resulting from the actions, together with the probabilities of reaching the different states; evaluates the possible states, based on a static evaluation function; and finally returns the actions considered best, in the order in which they should be taken. By considering only actions associated with accomplishing specific goals, CP2 limits its search to a tiny fraction of the search space of the domain.

The planner interacts with the knowledge base, which consists of a goal hierarchy and relevant set of rules that determine the goal parameters. To determine whether a goal is feasible, CP2 consults the plausible reasoner. The reasoner refers to the rules associated with the specified goal pertaining to the feasibility of the goal. Similarly, the reasoner consults rules pertaining to the satisfaction of the goals, and so on.

Changes in the knowledge base resulting from the learning process, influence the quality of plans produced by CP2. As the system improves its knowledge base, the ability of the planner to generate sequences of actions to accomplish the goals of the computer opponent improves as well.

The components of the WARGLES computer opponent have different degrees of generality. The adversarial

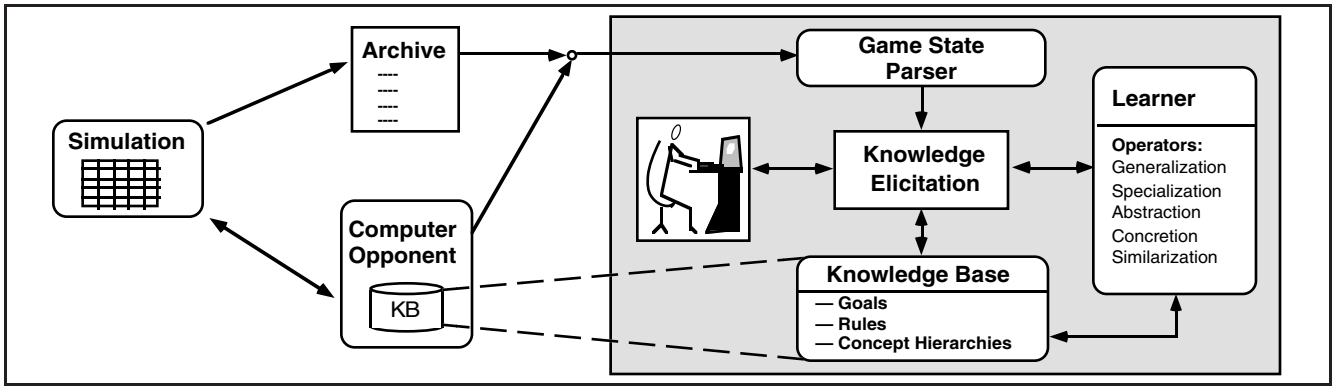

Figure 5: WARGLES Learner

planner and plausible reasoner are general problem-solving engines. The knowledge in the goal tree is applicable to a general class of games (in this paper, the class of strategic wargames). Thus, portions of this goal tree might be reused or modified for other games. The rules in the knowledge base consist of knowledge specific to the particular type of game being played, as well as general concepts related to that class of games.

WARGLES Learner - A conceptual diagram of the learning system of WARGLES is shown in Figure 5. The main learning goal is to improve the incomplete and partially incorrect knowledge base of the computer opponent. There are several learning scenarios for achieving this goal. One is a post-analysis of a completely played game. In this scenario, the input to the learner is a transcript of a contest. This transcript is shown to a human expert who identifies wrong actions made by WARGLES and guides it in improving the KB such that it no longer makes mistakes in similar situations.

Another scenario, which will be presented in this paper, is an apprenticeship learning one (Tecuci & Kodratoff, 1990; Wilkins, 1990). In this scenario, WARGLES is playing against a human expert player. A learning session starts when WARGLES attempts to take an action considered unsatisfactory by the expert. In this situation, the expert will respond with a counter-action, and will also indicate a better action for the system which the system should have made. The goal of WARGLES is to improve the KB so that in future similar situations, the better action indicated by the expert, is chosen, rather than the incorrect action. The learning problem to be solved is given in Table 1. The learning method was developed in the multistrategy task-adaptive learning framework presented in (Tecuci, 1993).

Given
- — a game state S_i;
- — an incomplete and partially incorrect KB;
- — an incorrect action A_i attempted by the system;
- — a response action A_r of the human expert to A_i;
- — a better action A_b which the system should have taken instead of A_i;

Determine
- — an improved KB that justifies the better action A_b for the game state S_i.

Table 1: The learning problem

First, WARGLES builds a plausible justification tree which shows how the actions of the computer opponent attempted to achieve the goals of the computer opponent. A plausible justification tree is like a proof tree, except that the inferences which compose it may be the result of different types of reasoning (not only deductive, but also analogical, abductive, predictive, etc.) (Tecuci, 1993). Because the action is wrong, the plausible justification tree is also wrong. WARGLES will use the counter-action of the expert and the corresponding game state in order to detect the errors in the justification tree. Then it will use the better action indicated by the expert and will attempt to make those modifications to the KB to entail this better action and no longer entail the wrong one. In this process, the system may modify some inference rules which were used in evaluating the goals and in building the plausible justification trees. This credit/blame assignment problem is intrinsically difficult for an autonomous learner. We are investigating several automatic solutions to this problem, based on the methods proposed by (Wilkins, 1990) and (Tecuci, 1993). However, the method we are currently employing in WARGLES is an interactive one, based on the knowledge elicitation techniques described in (Tecuci and Hieb, 1993; Tecuci and Duff, 1993). This method will be illustrated in the next section.

Illustration of WARGLES

The following section illustrates how WARGLES' computer opponent plans and then learns in a simple scenario. Relevant portions of the knowledge structures are presented along with their use in planning and revision during learning. The example was simplified to concentrate on the relevant aspects of the learning process.

Initial Scenario - Figure 6 shows a game state (S_1) in the middle of a contest between a computer opponent and a human adversary. Locations are given in a standard x-y coordinate system. Both sides have 4 units, 2 infantry and 2 armor. It is the computer opponent's turn to act. Only a partial map of the WARGLES board is shown for this example.

Planning - The computer opponent uses the goal tree in Figure 7 to plan its action for the given scenario. Only a partial goal tree is shown. The adversary's goal tree is similar. In order to satisfy the top level goal of "Win-game," its two subgoals must be satisfied. Here we consider only the subgoal "Protect-own-capital." The planner determines which of the goals are feasible by evaluating the rules for evaluating predicates such as "Goal-feasible(Protect-flanks)." Then a pair-wise comparison of the proponent and adversary goals is conducted to determine which goals succeed and a plan is constructed. In this scenario, the computer opponent knows that the human has the goal "Occupy-enemy-capital" and attempts to prevent this goal by satisfying the feasible subgoals under "Protect-own-capital." This consists of the leaf nodes "Block-enemy-from-entering-own-capital" and Protect-flanks", which include the rules shown in Figure 8 to determine their parameter values. These rules are in the form of mutual implication rules, which are bi-directional and have rule strengths in both the forward (α) and backward directions (β) as described in (Collins & Michalski, 1989).

No action is taken for the goal "Protect-flanks" since the goal is satisfied given the initial state of the game. However other rules determine that the armored unit at location (18,14) may be moved to location (18,16) to protect the capital from the enemy unit at location (15,17) to satisfy the goal "Block-enemy-from-entering-own-capital" (see Figure 6). The set of goals stored in the transcript for the action is G_1= {Win-game, Protect-own-capital, Block-enemy-from-entering-own-capital, Prevent-destruction-of-own-forces, Protect-flanks}.

Playing - Figure 9 shows the action generated by the planning algorithm of the computer opponent. Since all goals seem to be satisfied, no other actions are necessary, so the computer opponent terminates its turn. The next action is by the human player who takes two actions in succession. The infantry unit at location (15,17) moves to location (17,16) and the armored unit at location (15,13) moves to location (16,14) and attacks the infantry unit at location (17,15). This attack succeeds (it has a 0.6 probability of success) and the armored unit moves to location (17,15). The infantry unit attacked cannot retreat due to the zone of control of the infantry unit at location (17,16) and is destroyed. Figure 10 shows the final position of the units after the human player's actions.

As a result of the incorrect action, the computer opponent suffers a loss of one unit and ends up with a very vulnerable position, since its capital is directly threatened.

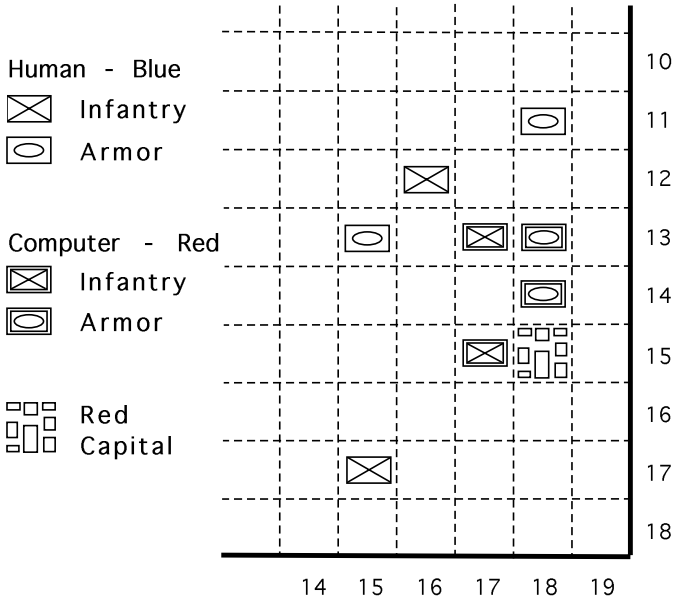

Figure 6: Initial board position for learning (State S_1)

Learning - An incorrect action made by the computer opponent presents a good opportunity for learning. There are three inputs to this learning process:

- the wrong action made by the computer opponent in the state S_1 (see Figure 6) and the resulting state S_2 (see Figure 9);
- the response of the expert and the resulting state S_3 (see Figure 10);
- the better action which WARGLES should have taken in state S_1. This action is indicated by the human expert and in the scenario described would have resulted in a new state $S_{2'}$ shown in Figure 11. The state $S_{2'}$ is much better then the state S_2 because the computer opponent has all the units protected and the capital city is not threatened.

The learning goal of WARGLES is to improve the KB such that in situations similar to S_1, the computer opponent chooses to take the better action indicated by the human expert, and not the wrong one taken in the current game.

The learning process takes place in several stages, as indicated in the following: First, WARGLES builds and analyzes the plausible justification tree, which shows how the action it chose in state S_1 achieves the goals of the

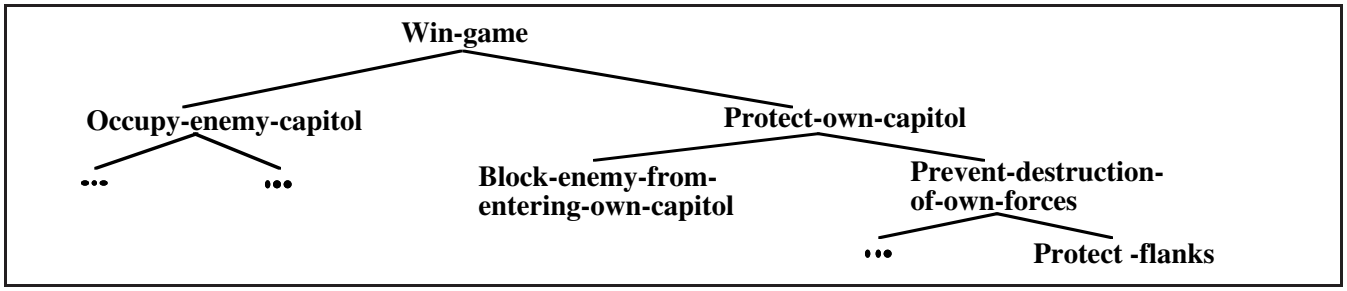

Figure 7: Goal Tree for Computer Opponent

Pertaining to goal, "Block-enemy-from-entering-own-capital":	
R1: Units-available(Block-enemy-from-entering-own-capital) <--> Goal-feasible(Block-enemy-from-entering-own-capital) ($\alpha = 1.0$, $\beta = 0.2$)	The goal of blocking the enemy from the capital is feasible if units are available to accomplish it.
R2: { $\forall$x Direction(x), Enemy-blocked-from(x, Capital) --> No-enemy-from(x, Capital) } <--> Goal-satisfied(Block-enemy-from-entering-own-capital) ($\alpha = 0.9$, $\beta = 0.4$)	This goal is satisfied if the enemy is blocked in all directions (North, South, East, West) or if there is no enemy from these directions.
Pertaining to goal, "Protect Flanks":	
R3: Units-available(Protect-flanks) <--> Goal-feasible(Protect-flanks) ($\alpha = 1.0$, $\beta = 0.05$)	The goal of protecting flanks is feasible if units are available to accomplish it.
R4: { $\forall$x, Location(x), Unit-at(x) --> Flank-protected-at(x) } <--> Goal-satisfied(Protect-flanks) ($\alpha = 0.9$, $\beta = 0.6$)	This goal is satisfied if the flanks of all units are protected.
General Rules	
R5: { $\forall$x, Location(x) [$\forall$y, Direction(y), Unit-at(x) & Enemy-units-near(x, y) --> ($\exists$z, Orientation(z), Friendly-unit-next-to(x, y, z))] <--> Flanks-protected-at(x) } ($\alpha = 1.0$, $\beta = 0.9$)	The flanks are protected for a unit at a location if enemy units are near (can reach the location within one turn) and there is a friendly unit to the right or the left of the unit.

Figure 8: Rules for Goal Tree

computer opponent. This tree is shown in Figure 12. (The predicate GS in Figures 12 and 13 is an abbreviation for Goal-satisfied.) Because the action is wrong, the plausible justification tree is also wrong.

Second, WARGLES has to identify the wrong inferences of the justification tree (i.e., to solve the blame assignment problem). To do this, it analyzes the board situation after the response action of the expert, that is, the state S_3. This situation shows that the goal "Prevent-destruction-of-own-forces" was not, in fact, satisfied, by the action made by WARGLES in situation S_1. Therefore, the part of the justification tree in Figure 12 that shows how this goal is satisfied is wrong. This part of the justification tree is shown to the expert who is asked to identify the wrong implication. The expert indicates that the wrong implication is the one which derives "Flanks-protected-at(17,15)".

Third, the system has to correct the KB such that it no

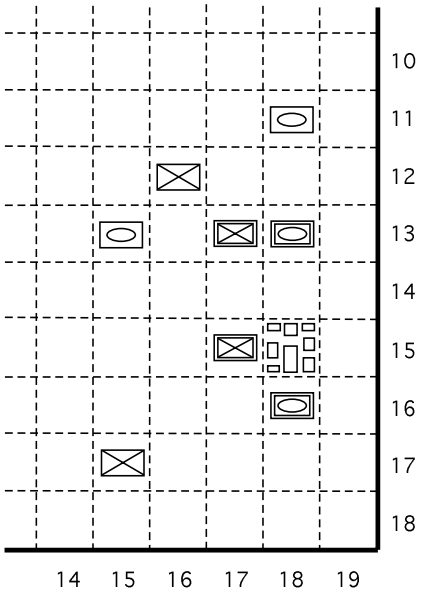

Figure 9: Board position after computer opponent's action (State S_2)

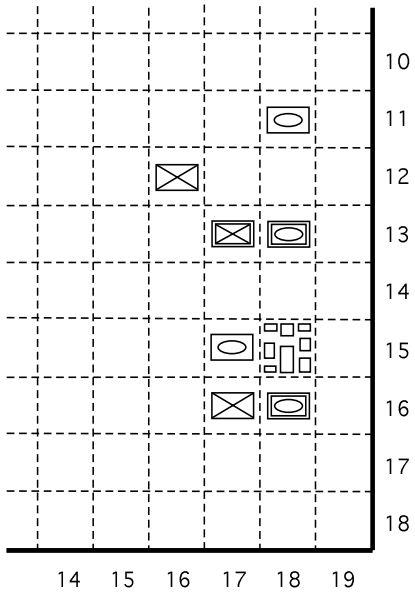

Figure 10: Board position after human player actions (State S_3)

longer entails the wrong plausible justification tree in Figure 12 and, in the same time, entails a new plausible justification tree corresponding to the better action indicated by the expert. The rule to be corrected is the rule R5 in Figure 8 which derived the false fact "Flanks-protected-at(17,15)". Therefore, the instance of this rule from Figure 12 is a false positive example. On the other hand, "Flanks-protected-at(17,15)" may be assumed to be satisfied in situation S_2' and one may make the hypothesis that the updated rule R5 will have a true positive instance in the situation S_2'. These observations guide the process of updating the rule R5.

Indeed, WARGLES will compare the situation S_2' with the situation S_2 and will look for a predicate which is present in situation S_2' without being present in situation S_2. Such a predicate is, for instance, "Friendly-unit-next-to((17,15), West, Left)". This predicate is proposed to the expert as an additional condition that might ensure the satisfaction of "Flanks-protected-at(17,15)", and therefore the satisfaction of the goals "Protect-flanks" and "Prevent-destruction-of-own-forces", as indicated in Figure 13. Because the expert accepted this correction, the only thing which remains to be done is to update the rule R5 to uncover its instance from Figure 12 and to cover the instance from Figure 13. This updated rule is shown in Figure 14. With this new rule, the system would no longer take the wrong action in situations similar to S_1 but, instead, will take the better action indicated by the expert.

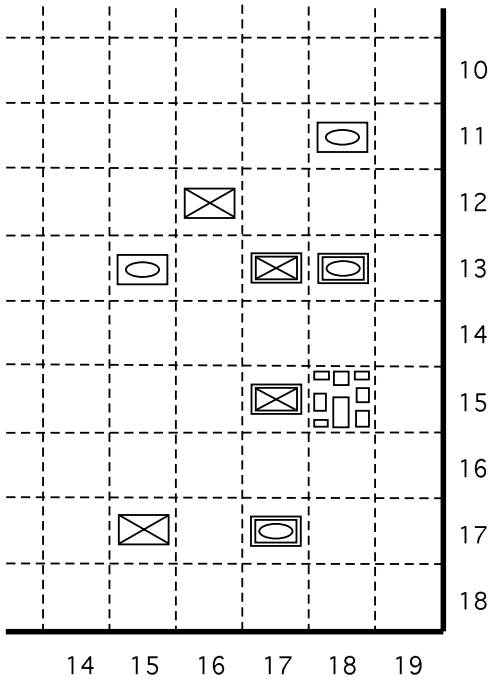

Figure 11: Board position after action elicited from human expert (State S_2')

Related Work

The literature on computer game playing is dominated by various search techniques. Our approach is knowledge-based, is concerned with learning and uses a game (and domain) novel to computer game playing. There are several computer opponents which are concerned with learning, such as SCANCHUNK (Walczak & Dankel, 1993), MORPH (Gould and Levinson, 1993), HOYLE (Epstein, 1992) and those described in (Pell, 1991; 1992).

EURISKO (Lenat, 1983) is a system which

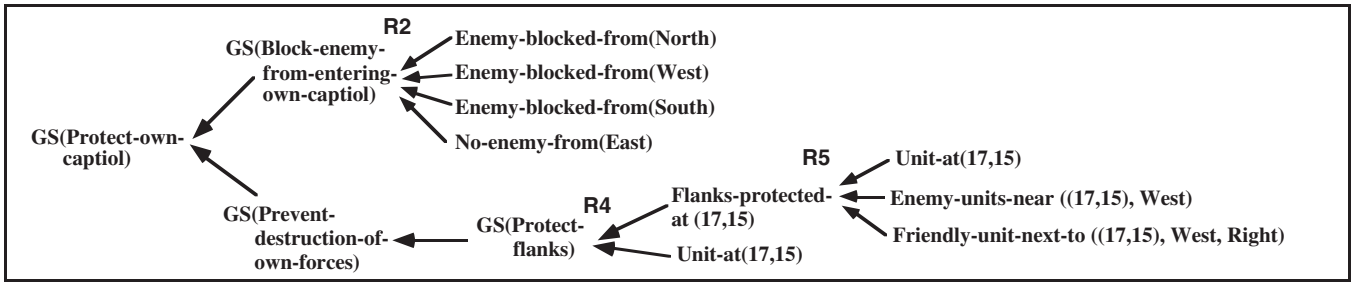

Figure 12: Part of the Plausible Justification Tree for S_2

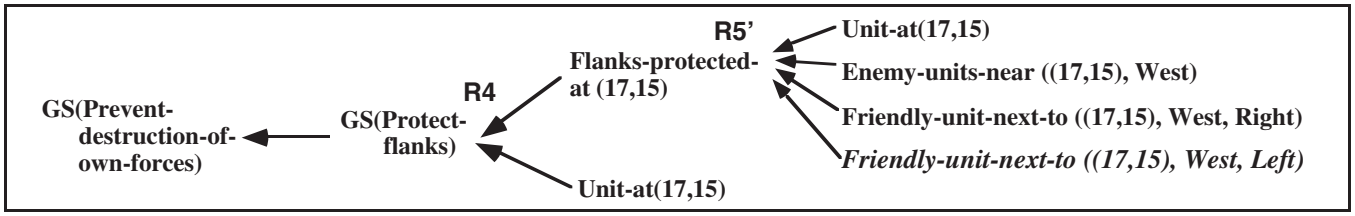

Figure 13: Part of the Plausible Justification Tree for S_2'

R5': { ∀x, Location(x) [∀y, Direction(y),
Unit-at(x) & Enemy-units-near(x, y)
--> (Friendly-unit-next-to(x, y, right)
& Friendly-unit-next-to(x, y, left))]
<--> Flanks-protected-at(x) } ($\alpha = 1.0$, $\beta = 0.9$)

The flanks are protected for a unit at a location if enemy units are near (can reach the location within one turn) and there are friendly units to the right *and* the left of the unit.

Figure 14: Revision of rule for "Flanks-protected-at"

discovers new concepts by modifying heuristics and evaluating the application of these heuristics. Lenat applied EURISKO in an innovative fashion to the task of designing a naval fleet to successfully compete in a naval wargame. EURISKO worked in a similar domain to WARGLES, and is also knowledge-based. However, WARGLES is quite different, in that we are designing an opponent, rather than optimizing the particular force that the opponent has.

While EURISKO could design an initial fleet by running numerous simulations, it never reacted to the actions of an adversary, or learned from the experience of its fleet. This was due to the nature of the contest, where fleet designs were evaluated in a preset scenario. With WARGLES, the composition of the force is predetermined (using one of various preset scenarios), and the learning problem is how to improve the performance of this force.

The approach taken by Walczak and Dankel (1993) is similar to WARGLES in that it is concerned with learning strategic and tactical knowledge. However, its learning method works by extracting knowledge about the geometric relations between pieces through induction over a series of games. It attempts to deal with the strategy of its adversary and modify its play to react to a given strategy. The system specifically uses information about a particular adversary taken from past contests to tailor it's strategy. The particular method is one of chunking a board position into particular patterns. It then uses these patterns to predict how it's adversary will move, assuming that the adversary will attempt to move into previously determined patterns, and plots it's moves accordingly. Specific heuristics are used to assist the determination of moves, once the adversaries likely move is calculated. The approach is based on a minimax algorithm.

While this approach does acquire useful patterns, it is doubtful that predicting the adversaries movement in WARGLES in this way would work. This method does not take into account new strategies which may be attempted, and requires an extensive history of play of an adversary. The approach stays at the pattern stage and does not select an appropriate overall strategy at the knowledge level.

PARADISE (Wilkins, 1980) is a chess-playing system which uses knowledge-based plans to efficiently limit its search space. It only plays in the middle game. The system uses a large number of rules to construct its plans. However, it does not learn from or adapt to its opponent's moves, and additional knowledge (in the form of rules) must be hand-crafted. It may be that the difficulty of knowledge acquisition prevented this approach from being used in subsequent chess opponents

Conclusions and Future Research

In this paper we have presented an approach to the problem of designing a computer opponent for wargames that can integrate planning, knowledge acquisition and learning. We illustrate this approach by describing our WARGLES system.

Both the planner and the learner have the same knowledge base, in the same representation. The planner uses a goal tree which indexes rules and evaluation functions. The rules are used to prune the search space when constructing a plan considering the adversary's possible responses. This approach is general to adversarial games, although a general goal tree must be constructed for each class of games. The methodology is also applicable to situations covered by game theory (Hamburger, 1980), such as adversarial economic competition between businesses.

However, a limitation of this approach is that for each particular game the specific domain knowledge (both rules and the goal tree) would have to be elicited or learned. Also, currently there are no techniques for modification of the goal tree if it is in error. Another potential problem is blame assignment, in that improbable actions may occasionally succeed (such as a weak force defeating a strong force). Currently, the planner uses an expected value during static evaluation to identify those low probability actions which may have a significant impact (such as cause a loss for the player in the next ply).

The success of WARGLES can be measured by several methods of evaluation. The first is to have successive versions of WARGLES play a computer opponent that does not learn. The measure of improvement would be the ratio of wins to loses as well as the margin of victory. The second method would measure the ratio of wins to loses of successive versions of WARGLES as it learns. A third measure would establish a ranking of players, similar to a chess ranking. WARGLES would then be given a ranking and an improvement in its performance would be measured by the increase in its ranking.

The WARGLES learner is also a general multistrategy task-adaptive learner that deeply and dynamically combines several inferential learning strategies in a plausible justification tree. However, a limitation of the current approach is the important part played by knowledge acquisition, where the system needs an expert's help to solve the blame assignment problem and to correct the KB.

Future research includes further automating the learning process, as well as incorporating other forms of learning as, for instance, experience-based reinforcement learning of moves and conceptual clustering of units into defensive and offensive task forces. We also plan on using a novel knowledge representation called DIH (Hieb & Michalski, 1993), designed for MTL systems, to facilitate both inference and learning.

Because of the extensive research already performed on wargames, there is a wealth of parametric models encoding a large body of military knowledge (Perla, 1990). This knowledge is in a form quite suitable for incorporation into WARGLES. Scaling up the WARGLES military simulator will involve adding in additional features such as terrain, weather, incomplete player knowledge of opponent's forces, etc., to develop more realistic simulations.

Acknowledgments

The authors wish to thank Eric Bloedorn and Michael Tanner for their review of this paper, as well as the anonymous reviewers for their helpful comments and suggestions.

This research was conducted in the Center for Artificial Intelligence at George Mason University. The Center's

research is supported in part by the National Science Foundation under grant No. IRI-9020266, in part by the Advanced Research Projects Agency under the grant No. N00014-91-J-1854, administered by the Office of Naval Research and grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under grant No. N00014-91-J-1351.

References

Avalon Hill, (1958). *Tactics II.* Baltimore, MD.

Berliner H.J., (1980). Backgammon Computer Program Beats World Champion, *Artificial Intelligence*. (pp. 205-220) Vol. 14.

Collins, A. & Michalski, R.S., (1989). The Logic of Plausible Reasoning: A Core Theory, *Cognitive Science*. (pp. 1-49) Vol. 13.

Davis, P.K., (1986). Applying Artificial Intelligence Techniques to Strategic-Level Gaming and Simulation, In M.S. Elzas, T.I. Oren, and B.P. Zeigler, Eds., *Modeling and Simulation Methodology in the Artificial Intelligence Era*. (pp. 315-338) Elsevier Science Publishers.

Dunnigan, J.F., (1992). *The Complete Wargames Handbook: How to Play, Design and Find Them.* Revised Edition, William Morrow and Company: New York, NY.

Epstein, S.L., (1992). Prior Knowledge Strengthens Learning to Control Search in Weak Domains, *International Journal of Intelligent Systems*. (pp. 547-586) Vol. 7.

Gould, J. and Levinson, R., (1993). Experienced-Based Adaptive Search, In R.S. Michalski & G. Tecuci Eds., *Machine Learning: A Multistrategy Approach, Volume 4*. Morgan Kaufmann Publishers: San Mateo, CA.

Hall & Loeb (1993). Thoughts on Programming a Diplomat, In H.J. van den Herik & L.V. Allis, Eds., *Heuristic Programming in Artificial Intelligence 3 - The Third Computer Olympiad.* (pp. 123-145) Ellis Horwood: London.

Hamburger, H., (1979). *Games as Models of Social Phenomena*. W.H. Freeman & Co.: San Francisco.

Hieb, M.R. & Michalski, R.S., (1993). Multitype Inference in Multistrategy Task-adaptive Learning: Dynamic Interlaced Hierarchies, In R.S. Michalski & G. Tecuci, Eds., *Proceedings of the Second International Workshop on Multistrategy Learning*. (pp. 3-18) Harpers Ferry, West Virginia.

Lehner, P.E., (1990). Automated Adversarial Planning Search Procedures With Provable Properties, In S. Andirol, Ed., *Advanced Technology for Command and Control Systems Engineering.* AFCEA International Press: Fairfax, VA.

Lenat, D.B., (1983). EURISKO: A Program That Learns New Heuristics and Domain Concepts, *Artificial Intelligence*. (pp. 61-98) Vol. 21.

Michalski, R.S., (1993). Inferential Theory of Learning: Developing Foundations for Multistrategy Learning, In R.S. Michalski & G. Tecuci Eds. *Machine Learning: A Multistrategy Approach, Volume 4*. Morgan Kaufmann Publishers: San Mateo, CA.

Pell, B., (1991). Exploratory Learning in the Game of GO: Initial Results, In D.N.L Levy & D.F. Beal, Eds., *Heuristic Programming in Artificial Intelligence 2 - The Second Computer Olympiad*. (pp. 137-152) Ellis Horwood: London.

Pell, B., (1992). METAGAME: A New Challenge for Games and Learning, In H.J. van den Herik & L.V. Allis, Eds., *Heuristic Programming in Artificial Intelligence 3 - The Third Computer Olympiad*. (pp. 237-251) Ellis Horwood: London.

Perla, P.P., (1990). *The Art of Wargaming.* Naval Institute Press: Annapolis, MD.

Samuel, A.L., (1967). Some Studies in Machine Learning using the game of Checkers - II, *IBM Journal*. (pp. 601-617), Vol. 11.

Tecuci, G. & Kodratoff Y., (1990). Apprenticeship learning in imperfect theory domains, In Y. Kodratoff & R. S. Michalski, Eds., *Machine Learning: An Artificial Intelligence Approach, Volume 3.* Morgan Kaufmann: San Mateo, CA.

Tecuci, G., (1993). A Framework for Multistrategy Learning, In R. S. Michalski & G. Tecuci, Eds., *Machine Learning: An Multistrategy Approach Volume 4.* Morgan Kaufmann: San Mateo, CA.

Tecuci, G. & Duff, D., (1993). Knowledge Base Refinement Through a Supervised Validation of Plausible Reasoning, In R.S. Michalski & G. Tecuci, Eds., *Second International Workshop on Multistrategy Learning* (pp. 301-308) Harpers Ferry, West Virginia.

Tecuci, G. & Hieb, M.R., (1993). Consistency-driven Knowledge Elicitation: Using a Machine Learning-oriented Knowledge Representation to Integrate Learning and Knowledge Elicitation in NeoDISCIPLE, *Knowledge Acquisition Journal.* to appear.

Walczak, C. & Dankel, D. II, (1993). Acquiring Tactical and Strategic Knowledge with a Generalized Method for Chunking of Game Pieces, *International Journal of Intelligent Systems.* (pp. 249-270) Vol. 8.

Wilkins, D.E., (1980). Using Patterns and Plans in Chess, *Artificial Intelligence.* (pp. 165-203) Vol. 14.

Wilkins, D.C., (1990). Knowledge Base Refinement as Improving an Incorrect and Incomplete Domain Theory, In Y. Kodratoff & R. S. Michalski, Eds., *Machine Learning: An Artificial Intelligence Approach, Volume 3.* Morgan Kaufmann: San Mateo, CA.

Young, P.R. & Lehner, P.E., (1986). Applications of a Theory of Automated Adversarial Planning to Command and Control, *IEEE Transactions on Systems, Man, and Cybernetics*. (pp. 806-812), Vol. 6.