Computational Intelligence in Strategy Games

V Rao Vemuri

Department of Applied Science, University of California, Davis
rvemuri@ucdavis.edu

Abstract - Presented are issues in designing smart, believable software agents capable of playing strategy games, with particular emphasis on the design of an agent capable of playing Cyberwar XXI, a complex war game. The architecture of a personality-rich, advise-taking game playing agent that learns to play is described. The suite of computational-intelligence tools used by the advisers include evolutionary computation and neural nets.

I. CONFLICT SIMULATIONS

Strategy games, in general, and conflict simulations in particular, offer a fertile ground to study the power of computational intelligence (CI). Board games like Chess or Checkers are widely studied strategy games because the environment in which the user interacts with the game is not a simulation of the problem domain; it *is* the problem domain. As a result, many vexing problems like imperfect effectors and sensors, incomplete or uncertain data, ill-defined goal states can be bypassed. However, games like Chess are only highly stylized abstractions of realistic conflicts. Evolutionary computational methods have been proved to be highly successful in handling these problems [1] [2] [3].

Simulations of warfare, hereafter referred to as conflict simulations, serve as excellent testbeds for studying the learning behavior and decision making capabilities of intelligent, game-playing agents because of the following reasons:

- They represent more realistic situations of practical significance
- Large amounts of crucial background knowledge is already available
- Diversity of the underlying scenarios offer a challenge to the design process
- Utility of intelligent computer opponents for military training and strategic decision-making
- Scalability of the system

In broad strokes, the decisions made by an agent during conflict simulations are not too unlike the actions taken by a human player at a board game like Monopoly and Backgammon or a card game like Bridge. Whoever "controls more points" are essentially "in charge" of the situation. However war games differ from stylized games like Chess and Checkers in many ways:

- The rules are much more complex than in Chess
- The games are typically multiplayer games; each player representing an individual, a nation, a group, or a coalition
- The ultimate goal in playing these new generation war games is not so much to win the game; rather, it is to study a variety of "what if" scenarios in order to develop decision making capabilities.

The gaming environments in war games pose a number of problems at the strategic, interface and the run-time environment levels. Although the later two are also important, treatment of the strategic-level is the primary target of this paper. It can involve the selection of strategies, translation of strategies into move sequences and responding to opponent's actions and so on.

The war game taken up for study is Cyberwar XXI, a board game designed by Joe Miranda of Hexagon Interactive [4]. The game would provide users with some predictive capability to project the effects of high-tech weapons. The game's rules are based on the concept that modern warfare occurs simultaneously on four primary levels of conflict:

- "Battle Level" is where conventional ground and sea forces clash.
- "AirSystem Level" is where air power is pitted against national infrastructure.
- "Infowarfare Level" is where cybernetic, intelligence and special operations forces conduct combat using computer viruses, electronic warfare, and media manipulation.
- "Economic Level" is where information about the financial transactions of the opponents are tracked to gain better insights on the participants or where actions like sanctions are used to coax the opponent to a different point of view.

The design connects all these levels with cascading effects – that is, the propagation of effects caused by actions at one level to other levels.

The game is played in turns. Each turn is divided into segments called phases. The various phases through which the game progresses are depicted in Figure 1. Without delving into the details of the game, suffice it to say that the players of the game in question are required to make several different types of decisions in different phases of the game. These decision points are summarized below:

- The selection of a InfoWar Squares, which in turn determines the number of Strategy Cards that are available to a player in each phase of the game. A "strategy card" gives a player certain combination of military assets and certain number of opportunities to use those assets. (A sample card is shown in Figure 2.)
- The selection of the prescribed number of (say M) strategy cards, from a deck of N, for each phase of the game. Given a limited number of strategy cards, each player has to decide on the optimum mix of assets to accomplish his/her goal.
- The selection of missions in each phase from the set of allowed missions, and deciding how many of the available resources to allocate to each mission.

1. Chaotic Events Phase
No decisions by CI. Random factors control events.

2. Initiative Determination
No decisions by CI. Who plays first is decided here.

3. Mobilization (Selection of Strategy Cards)
Each agent selects some of strategy cards (SC) to use.
3.1 Parameters: (number of cards that may be chosen, list of legal cards to choose from, game state that influences the value of the selected card)
3.2 Considerations: (actions/impulses granted by the card and in what space, reinforcements received, limitations on the placement of reinforcements, cascading effects, Information Warfare cost incurred, history of opponent SC selection, etc.)
3.3 Heuristics: Do not select cards whose requirements cannot be met. Select cards with the intention of using them in a specific way. Consider look-ahead planning at this stage.
3.4 CI Ideas: Assess the value for each potential card combination and action/impulse sequence, develop a set of rules to guide the selection of cards.

4. InfoSpace Warfare Phase
…
5. AirSystems Space Warfare Phase
…
6. BattleSystems Space Warfare Phase
…
7. Economic Conflict Phase
…
8. Reconstitution Phase

FIGURE 1. PHASES IN A TURN

These selections are to be made with the intention of "maximizing" one's own perceived "value" or utility. Although this perceived utility may differ from individual to individual, experience suggests that a "safe" way of playing the game is to work toward the goal of maximizing the overall InfoWar points one can control, in terms of gaining information dominance at the InfoWar level. So a hypothesis one might posit is that maximal InfoWar point gain with a least loss in units is a desirable outcome. Whether or not this strategy leads to a desired political goal is an issue that can be studied with this simulation.

STRATEGY CARD (BattleSpace)
Name: AirLand Battle
Actions: The player can initiate three Impulses on the BattleSpace level.
Reinforcements: None.
Modifiers: Gain a +1 die roll modifier for all the player's Maneuver attacks.
Cascading Effects: Normal.
Infrastructure Requirement. C4I Infrastructure required.
IW Cost: Lose 20 IW Points
Chaos Level: Raise Chaos Level by 1 die roll upon play.
Other: Only the United States can play this card. May not be combined with any other BattleSpace or Aerial

FIGURE 2. A SAMPLE STRATEGY CARD

II. COMPUTATIONAL INTELLIGENCE AND COGNITIVE MODELS

There is no well-developed theory to solve the problem outlined in Section I. The purpose of this paper is to explore the role of interactive simulations of specially designed "war games" to study decision-making aspects of conflicts. In these interactive games human players will be pitted against believable software agents that come close to mimicking the capabilities of humans.

The goal in developing software agents is not so much in creating lifelike animations using physical laws and bio-mechanical modeling techniques. Rather, the goal is to achieve realism in cognitive modeling, a step beyond behavior modeling. The agents should react appropriately to perceived environmental stimuli and exhibit goal directed behavior. The cognitive models govern what an agent knows, how that knowledge is acquired, and how it can be used to plan actions. These agents are vulnerable to common human foibles like emotion and stress [5]. The objective is in achieving increased realism in the cognitive and emotional behavior of the game-playing agents and in capturing social situations. Finally the agents interact with each other to facilitate the simulation of group behavior. Such cognitive models are capable of directing the new breed of highly autonomous, intelligent agents that are beginning to find use in interactive computer games.

The design emphasis is on human-like behavior in a decision-making environment, not just on speed of the computer or the application of sheer computational power.

The essence of conventional implementations of game playing on computers is search. The most straightforward way of selecting the best move is to explore all possible consequences (exhaustive search) of any action that can be taken in a given state. On a 3 x 3 board of tic-tac-toe, for example, with two players, this results in the need to explore 9! = 362,880 variations - not a formidable number for a computer. If one can think of the operations in Cyberwar XXI's Battle Space as a board game resembling tic-tac-toe on a 100 x 100 grid, then 10,000! variations would result - surely a challenge even to the fastest of the computers.

It is true that classical AI search methods do not do an exhaustive search; they are lot smarter than that. For example, inherent symmetries in the problem can be exploited to reduce the search burden. In complicated and realistic games this may not be possible. Other ingenious tricks and compromises are possible. In any event, the strength of classical search techniques hinges on one's ability to perform a depth analysis and on the quality of static evaluation function chosen.

In minimax search, for example, player A associates a "value" to each possible state of the game and then seeks to minimize this value while player B seeks to maximize the same evaluation function. This approach suffers from two drawbacks:

- Assigning values to states is not a trivial exercise; needless to state that the search result depends on how these values are assigned.

- The assumption that B is a rational player whose value system is the same as that of A, and therefore always chooses the "best" defense as A interprets it.

In games simulating asymmetric conflicts (terrorism is an example of asymmetric conflict), this may not be a valid assumption [4]. One way to overcome this difficulty is to make the evaluation function of B different from that of A. Indeed modeling the opponent's evaluation function is in itself a research topic. A natural way to do this is to observe a player's behavior during the course of a game and use it in conjunction with any prior knowledge about the player.

There are other issues that need further attention. An action by one player may lead to alternative states - each with a different probability of occurrence. That is, the evaluation function will attain its value only with a certain probability. This forces one to consider the issue of using probability distributions to describe the consequences of a move. Classical game theory techniques can be invoked to some extent to address this problem.

IV. STRUCTURE OF THE AGENT(S)

An examination of the rules of the Cyberwar XXI revealed that the decision problem is fairly complex. As decision making by humans is not always rational, believable decision making behavior is not always rational behavior. This characteristic makes it difficult to depend on a rational agent or an agent that depends on systematic search methods to locate a goal state. Furthermore, given the potentially large number of players, the large number of options available to each and the fact that the "opponents" actions are not only hidden from general view but also they may include random actions makes the alpha-beta approach less attractive.

In addition to these considerations, there is a need to operationally decompose agent architecture in terms of some primitive capabilities. These constituent parts, when composed together, should give a variety of agent behaviors.

These considerations called for a design that is flexible, modular and scalable. Instead of having a centralized agent that does some sort of search to find the correct response, we decided to make the central agent very simple (mostly just a multiplexer) and delegate the processing to a bank of Advisors. The advisors would be comprised of relatively simple programs that compute a narrow aspect of the games, and each advisor would pass back to the agent an advice on what it thinks the agent should do. It would then be up to the agent to decide which advice to take (see Figure 3). This is not too unlike a couple of schemes published in the literature [7] [8].

The game-playing agent described here is comprised of a "head" agent assisted by a bank of advisors. The Head Agent is the main interface between the game simulation and the rest of the CI component (although the game's Database/Data structures may also be accessed by other components of the CI engine). The Head Agent receives requests from the main simulation loop whenever there is a need for decision-making assistance from the CI side of the game. This request should include the context (the stage of the simulation where a decision is to be made) of the simulation. Upon receiving this information the head agent will ask the bank of advisors for suggestions on what to do. For instance, if the head agent receives a signal requesting assistance in picking the strategy cards for the game, the head agent will pass this signal to all the advisors. The strategy cards will then be picked considering the suggestions of all advisors.

Each game-playing agent will have a panel of advisors for each task involved in the decision-making process. Therefore one can visualize the possible subset of advisors by looking at the tasks the agent has to perform in order to make the overall decision.

For example, the panel of advisors supporting the Infowarfare Level will have to make the following decisions:

- Strategy card selection
- Play space selection
- Mission selection
  - Decide on targets
  - Decide on missions
  - Decide on Units to carry out missions

High-level advisors (Staff Advisors), at least three within each of the Levels, perform oversight operations. The Staff Advisors typically perform the following tasks:

- Looks at the list of advice, and deletes items that would be overly detrimental to their space (example: if one of the Battle-Level advisors suggests a card that would have a large negative impact on the Information-Level, the IW high level advisor to the Battle-Level sub-agent would delete that action from the proposal list)
- Looks at the list of possible actions before the low level advisors remove any objectionable ones.
- Looks at the Game Information (Database and/or Data Structures).

Within this design there is great deal of flexibility, both in terms of the scope of problems it can handle, and in terms of development. By forcing the advisors to focus on small enough areas, they should be efficient enough to run within the lifetime of the universe. The combination of their advice (by using the trust values) will generate a fairly realistic (but probably not optimal) agent.

V. PERSONALITY AND EMOTION

The CI agent is expected to simulate the effects of stressful inputs on emotional states of the players and the potential impact of these emotional states on the quality of decision-making [9]. Critically, the simulation can capture not merely the actions of the real world players, but also can provide mechanisms for understanding their underlying maneuvers and objectives. It does so by quantifying factors such as political support and the "chaos" of transnational target audiences.

In order to capture the personality aspects of the players, a Personality Engine (PE) is being designed (see Figure 4). Modeling behavior and personality are admittedly very complex and this is one ares evolutionary computing ideas can play a useful role [5]. Until the design is complete, one of the Staff Advisors plays the role of a PE. The PE works in two phases: (a) pruning options available to the agent before they are considered by the agent's static evaluation function. This is tantamount to an agent not even considering an option due to its emotional state. (b) modifying the weights assigned by the agent's evaluation function.

Personality is being modeled using two of the major psychological theories that describe human personality: (a) Trait theory and (b) Needs-motivation approach. The structure of the PE consists of four main modules: Traits module, Needs-Motivation module, Physical module, and Learning module. The traits module emulates personality by assigning the agent a value within the range defined for each of a set of opposed traits and having these traits influence the agent's decisions. The needs-motivation module works by assigning the agent certain values of need for a number of defined factors (i. e., economic, religious, political, etc.). These values influence the agent's decisions by motivating it to satisfy its needs within the World State of the given game. The physical module models the physical state of the agent viewed as a human being. This feature will allow the agent's physical state (tired, angry, stressed, etc.) to influence its decisions. The learning module analyzes past game situations and predicts the opponent's personalities and strategies and uses feedback in the decision-making process.

III. LEARNING AND EVOLUTIONARY GAME PLAYING

In view of the discussion in Section II, unless the evaluation function predicts the state values reliably, the search has to be carried deep into the search tree with the attendant cost of computation. Ways to reduce this cost is through instruction, advice taking, pattern recognition and generalization; in short, via *learning*. What cannot be captured through precise rules can possibly be learned from examples.

One design that was actually implemented, on a trial basis, is a Battle Space advisor that evolves a neural net along the lines suggested by Fogel [1]. This paper compares the effectiveness of the mutation, crossover, and combination operators in evolving specific checkers strategies. The focus is on short-tem evolution, that is, the initial generations of evolution.

ACKNOWLEDGEMENT.

Thanks to Tony Zalewsky, Joe Miranda of Hexagon Interactive for introducing Cyberwar XXI and supporting the effort with a SBIR passthrough AFOSR contact. My students Daniel Castaneda, Chris Marsh, and Leland So contributed to various facets of the design. Dustin Pium of Contra Costa College wrote an experimental code to evolve a neural network capable of learning strategies on a simplified version of the Battle Space by using the method described in reference [1]. Discussions with Sunil Vemuri of Media Labs, MIT helped us to develop a focus. Daniel Castaneda is designing the Personality Engine. Peter von Kleinsmid, an independent consultant, is designing the Game Engine. At the time of this writing the game is not

completely debugged as such no simulation results are shown. Finally thanks to the anonymous reviewers for making many valuable suggestions to improve the paper.

References

[1] Fogel, D. B., (2000) "Evolving a Checkers Player Without Relying on Human Expertise," *Intelligence*, pp 21-27. http: www.natural-selection.com

[2] Chellapilla, K. and D. B. Fogel, (1999a) "Evolving Neural Networks to Play Checkers without Expert Knowledge," *IEEE Trans. Neural Networks*, Vol. 10:6, pp. 1382-1391.

[3] Chellapilla, K. and D. B. Fogel, (1999b) "Evolution, Neural Networks, Games, and Intelligence," *Proc. IEEE*, Vol. 87:9, Sept., pp. 1471-1496.

[4] J. Miranda and S. Marsella, CYBERWAR XXI: Advanced War ighting Concepts, Final Report to AFOSR, Hexagon Interactive, 6750 Wedgewood Place, Los Angeles, CA 90068, 1999.

[5] V. W. Porto and L. J. Fogel, "Evolution of Intelligently Interactive behaviors for Simulated Forces," Sixth International Conference, EP97, Indianapolis, IN pp 419-429, Lecture Notes in Computer Science, No. 1213, Springer, New York, April 1997.

[6] G. Tesauro, "Temporal difference learning in TD-Gammon," Comm. ACM, Vol. 38, No. 3, pp 58-68, 1995.

[7] S. L. Epstein, "For the Right Reasons: The FORR architecture for learning in a skill domain," Cognitive Science, 18(3): 479-511, 1994.

[8] A. F. R. Rahman and M. C. Fairhurst, "Multiple expert classification: a new methodology for parallel decision fusion," Intl. Journal of Document Analysis and Recognition, Vol. 3, pp 40-55, 2000.

[9] I. Wilson, The Artificial Emotion Engine: Driving Emotional Behavior, http://artificial-emotion.com

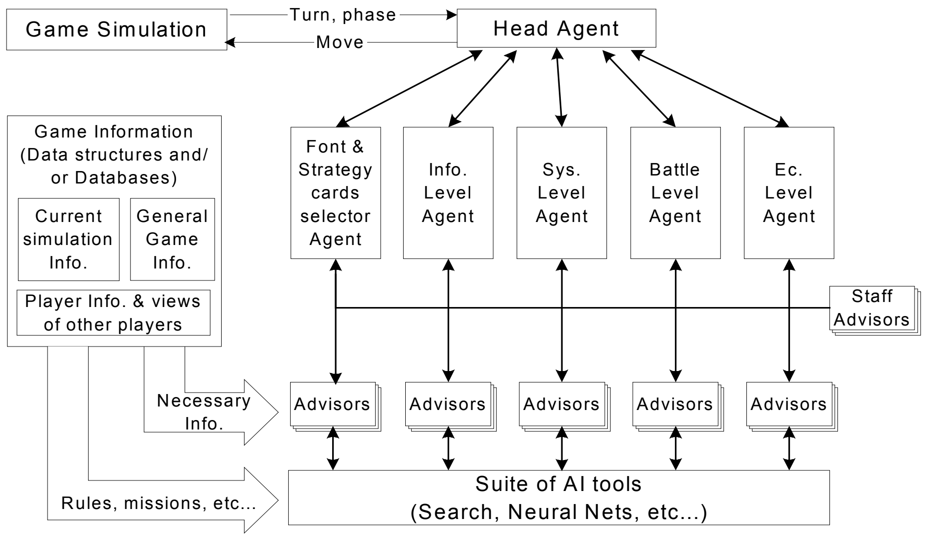

FIGURE. 3. AGENT ARCHITECTURE DIAGRAM.

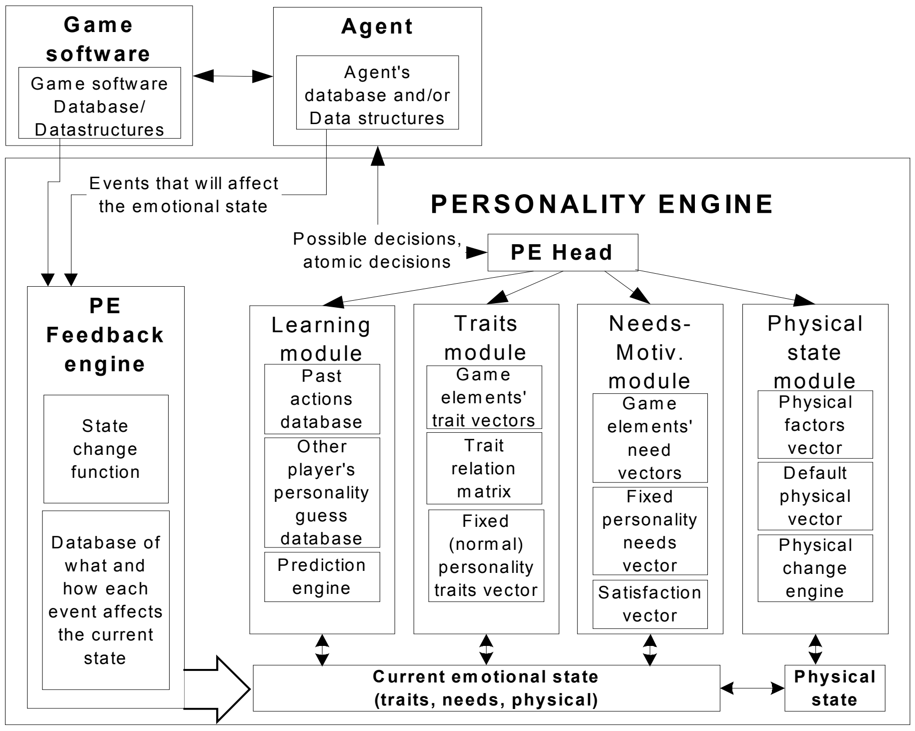
Game software
Game software Database/ Datastructures
Agent
Agent's database and/or Data structures
Events that will affect the emotional state
PERSONALITY ENGINE
Possible decisions, atomic decisions
PE Head
PE Feedback engine
State change function
Database of what and how each event affects the current state
Learning module
Past actions database
Other player's personality guess database
Prediction engine
Traits module
Game elements' trait vectors
Trait relation matrix
Fixed (normal) personality traits vector
Needs-Motiv. module
Game elements' need vectors
Fixed personality needs vector
Satisfaction vector
Physical state module
Physical factors vector
Default physical vector
Physical change engine
Current emotional state (traits, needs, physical)
Physical state

FIGURE. 4. PERSONALITY ENGINE ARCHITECTURE DIAGRAM.