Playing to Train: Case Injected Genetic Algorithms for Strategic Computer Gaming

Sushil J. Louis[1], Chris Miles[1], Nicholas Cole[1], and John McDonnell[2]

[1] Evolutionary Computing Systems LAB
University of Nevada
Reno - 89557
{miles,sushil,ncole}@cs.unr.edu
[2] SPAWAR Systems Center
San Diego, CA
john.mcdonnell@navy.mil

1 Introduction

We use case injected genetic algorithms to learn how to competently play computer strategy games that involve long range planning and complex dynamics [11]. Such games inspire, and are inspired by, military training simulations on which trainees spend considerable time honing their skills. By instrumenting the game interface, we unobtrusively acquire knowledge in the form of cases from human experts playing the game and use case-injected genetic algorithms to incorporate this knowledge in evolving competent game players. The games we have focused on have two sides, Blue and Red, and an evolved player can thus serve two purposes. A competent player for Blue serves as a decision aid for a Blue trainee. At the same time, a competent Blue player serves as a training opponent for a Red trainee. Results in the context of a strike force planning game show that with an appropriate representation, case injection is effective at biasing the genetic algorithm towards producing competent plans that contain important strategic elements used by human players.

Our research focuses on a strike force asset allocation game which maps to a broad category of resource allocation problems in industry and the military. Genetic algorithms can be used in our game to robustly search for effective allocation strategies, but these strategies may may not necessarily approach real world optima as the game (the evaluation function) is an imperfect reflection of reality. Humans with past experience playing the real world game tend to include external knowledge (gained through their experience) when producing strategies for the real-world problem underlying the game. Acquiring and incorporating knowledge from the way these humans play should allow us to carry over some of this external knowledge into our own play. Results show that case injection combined with a flexible representation biases the genetic algorithm toward producing strategies similar to those learned from human players. Beyond playing similarly on the same mission, the genetic algorithm can use strategic knowledge across a range of similar missions to continue to play as the human would. The left side of Figure 1 shows a screen shot of the game we are designing while the right side shows the scenario considered in this abstract.

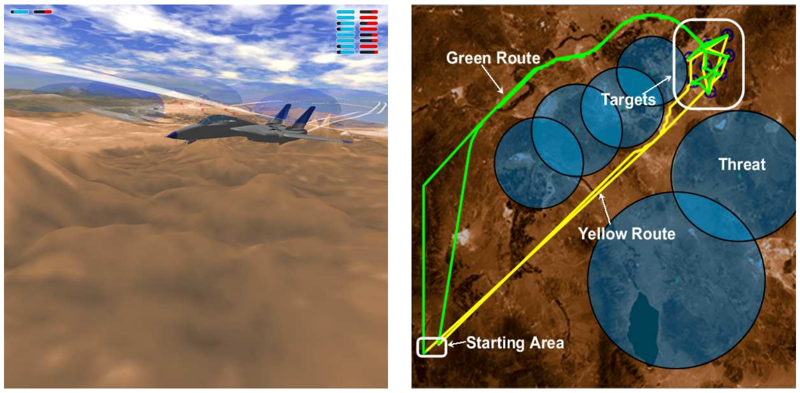

Fig. 1. Left: Game Screen-shot. Right: Mission scenario

2 The Game

Strike force asset allocation consists primarily of allocating a collection of strike assets to a set of targets. We have implemented this game on top of a professional open source game engine making it more interesting than a pure optimization problem, therefore enticing more people to play the "game" so that we can acquire player knowledge from them. The game involves two sides: Blue and Red, Blue allocates a set of platforms (aircraft) to attack Red's targets (buildings) while Red defends targets with defensive installations (threats). These threats complicate Blue's planning, as does the varying effectiveness of Blue's weapons against each target. Potential new threats and targets can also "pop-up" in the middle of a mission requiring Blue to be able to respond to changing game dynamics. Both players seek to minimize the damage they receive while maximizing the damage dealt to their opponent. Red plays by organizing defenses in order to best protect its targets and maximize damage to Blue's platforms. For example, feigning vulnerability can lure Blue into a pop-up trap, or keep Blue from exploiting a weakness out of fear of such a trap. Blue plays by allocating platforms and their assets (weapons) as efficiently as possible in order to destroy Red's targets and threats while minimizing risk. Risk to a platform depends on many factors including the platform's route, the effect of accompanying wingmen, changes in weather, and the presence of threats around chosen targets.

Our Genetic Algorithm Player (GAP) develops strategies for the attacking strike force, including routing and weapon targeting for all available aircraft. When confronted with popups, GAP responds by replanning with the case injected genetic algorithm to produce a new plan of action. Beyond producing near optimal strategies we would like to bias it toward producing solutions similar to those it has seen used by humans playing Blue in the past. Generating competent opponents is hard and using case-injected genetic algorithms to acquire and use knowledge from human experts should allow us to develop better training opponents and decision aids. Specifically we want to show that when using case injection the genetic algorithm more frequently produces plans similar to those a human has generated in the past on the same mission. We would also like to show that GAP can continue to use this information even when confronted with a different mission.

Previous work in strike force asset allocation has been done in optimizing the allocation of assets to targets, the majority of it focusing on static pre-mission planning [4, 10, 21, 13]. A large body of work exists in which evolutionary methods have been applied to games [3, 16, 15, 6, 17]. However the majority of this work has been applied to board, card, and other well defined games which have many differences from popular real time strategy (RTS) computer games such as Starcraft, Total Annihilation, and Homeworld[1, 2, 5]. Entities in our game exist and interact over time in continuous three dimensional space. Sets of algorithms control these entities, seeking to fulfill goals set by the player leading them. This adds a level of abstraction not found in traditional games. In most RTS games, players not only have incomplete knowledge of the game state, but even identifying the domains of this incomplete knowledge is difficult. John Laird [7, 9, 8] surveys the state of research in using Artificial Intelligence (AI) techniques in interactive computers games. He describes the importance of such research and provides a taxonomy of games. Note that several military simulations share some of our game's properties [20, 18, 14], but the purpose of our game is not to provide a perfect reflection of reality. Rather we want our strike planning game to both provide a platform for research in strategic planning and training, and to have fun.

The simple mission being played out is shown in on the right side of Figure 1. This mission was chosen to be simple enough to have easily analyzable results, and to allow the GA to learn external knowledge from the human. As many games show similar dynamics, this mission is a good arena for examining the general effectiveness of using case injection for learning from humans. The mission takes place in Northern Nevada and California - Lake Tahoe is visible near the bottom of the map. Blue possesses one platform armed with eight (8) assets (weapons) and the platform takes off from and returns to the lower left hand corner of the map. Red possesses eight (8) targets distributed in the top right region of the map, and six threats to defend these targets. The first stage in Blue's planning is determining the allocation of the eight assets to the eight targets. Each asset can be allocated to any of the eight targets, giving $8^8 = 2^{24}$ allocations. The second stage in Blue's planning involves finding routes for each of the platforms to follow during their mission. These routes should be short and simple but still minimize exposure to risk. We categorize Blue's possible routes into two categories. Yellow routes fly through the corridor between the threats, while green routes fly around (see figure 1). The evaluator has no direct knowledge of potential danger presented to platforms inside the corridor area - it has no conception of traps. Because of this, the evaluator optimal solution is the yellow route, as it is the shortest. The human expert however, understands the potential for danger in the corridor. Knowing this, the green route is the human preferred solution. Teaching GAP to learn from the human and produce green strategies even though yellow strategies have higher fitness is one of the goals of our current work.

3 Results

Parameterizing our A* router allows us to produce routes of different types and incorporating this parameter into the chromosome then allows individuals to contain information about not just what weapons to use and which targets to attack, but routing information about how to reach those targets [19]. Specifically, referring to Figure 1 (right), we want GAP to learn the parameter for green routes and overcome the genetic algorithm's preference for short routes in order to avoid traps. By injecting cases that incorporate green routes we can change the proportion of green routes produced by our system, despite such solutions having lower than evaluator-optimal fitness. Furthermore by artificially inflating the fitness of injected solutions and their descendants in the system's population, we can further control the proportion of routes that avoid potential traps. Figure 2 shows the value of the routing parameter, RC, on the x-axis and the number of runs of the genetic algorithm that converged to that value on the y-axis. Values of RC above 1.5 avoid the trap. The left side of the figure

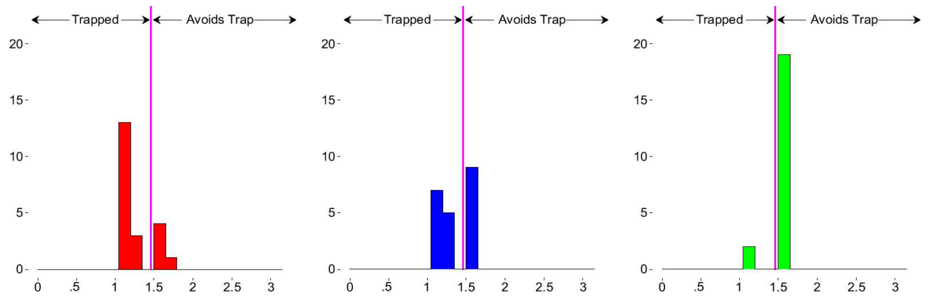

Fig. 2. Distribution of RC values. Left: GA alone. Middle: With case injection. Right: Case injection and fitness biasing.

shows the distribution of the routing parameter values without case-injection. More routes lead through the suspicious (to humans) corridor that is ripe for Red's trap. With case-injection (middle), there are more routes that avoid the potential trap although the difference is not significant. The rightmost graph shows the distribution of RC values with both case-injection and increasing the fitness of injected individuals and their descendants (proportional to the amount of injected material). In this case, there is clearly a strong preference for routes avoiding traps. More details on experimental setup and other results are given in our GECCO-04 paper.

This paper considered two broad issues in developing competent opponents for computer gaming - knowledge acquisition from expert players and biasing genetic search using information outside the evaluation function (acquired from humans). We showed that GAP learns to prefer solutions similar to injected cases acquired from humans and to deal with the external to the evaluator concept of a trap. Our preliminary results thus indicate that case injection seems to be a promising approach towards acquiring knowledge and biasing genetic search toward human preferred solutions. We are currently using these techniques in developing a training simulation (computer game) for strike planning.

Acknowledgment

This material is based upon work supported by the Office of Naval Research under contract number N00014-03-1-0104.

References

1. Blizzard. Starcraft, 1998, www.blizzard.com/starcraft.
2. Cavedog. Total annihilation, 1997, www.cavedog.com/totala.
3. David B. Fogel. *Blondie24: Playing at the Edge of AI.* Morgan Kauffman, 2001.
4. B. J. Griggs, G. S. Parnell, and L. J. Lemkuhl. An air mission planning algorithm using decision analysis and mixed integer programming. *Operations Research*, 45(5):662–676, Sep-Oct 1997.
5. Relic Entertainment Inc. Homeworld, 1999, homeworld.sierra.com/hw.
6. Graham Kendall and Mark Willdig. An investigation of an adaptive poker player. In *Australian Joint Conference on Artificial Intelligence*, pages 189–200, 2001.
7. John E. Laird. Research in human-level ai using computer games. *Communications of the ACM*, 45(1):32–35, 2002.
8. John E. Laird and Michael van Lent. Human-level ai's killer application: Interactive computer games, 2000.
9. John E. Laird and Michael van Lent. The role of ai in computer game genres, 2000.
10. V. C-W. Li, G. L. Curry, and E. A. Boyd. Strike force allocation with defender suppression. Technical report, Industrial Engineering Department, Texas A&M University, 1997.
11. Sushil J. Louis. Evolutionary learning from experience. *Journal of Engineering Optimization*, To Appear in 2004.
12. Sushil J. Louis and John McDonnell. Learning with case injected genetic algorithms. *IEEE Transactions on Evolutionary Computation*, To Appear in 2004.
13. Sushil J. Louis, John McDonnell, and N. Gizzi. Dynamic strike force asset allocation using genetic algorithms and case-based reasoning. In *Proceedings of the Sixth Conference on Systemics, Cybernetics, and Informatics. Orlando*, pages 855–861, 2002.

14. D. McIlroy and C. Heinze. Air combat tactics implementation in the smart whole air mission model. In *Proceedings of the First Internation SimTecT Conference, Melbourne, Australia, 1996.*, 1996.
15. Jordan B. Pollack, Alan D. Blair, and Mark Land. Coevolution of a backgammon player. In Christopher G. Langton and Katsunori Shimohara, editors, *Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems*, pages 92–98, Cambridge, MA, 1997. The MIT Press.
16. Christopher D. Rosin and Richard K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Francisco, CA, 1995. Morgan Kaufmann.
17. A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:210–229, 1959.
18. Graeme Murray Serena. The challenge of whole air mission modeling, 1995.
19. Bryan Stout. The basics of a* for path planning. In *Game Programming Gems*, pages 254–262. Charles River media, 2000.
20. Gil Tidhar, Clinton Heinze, and Mario C. Selvestrel. Flying together: Modelling air mission teams. *Applied Intelligence*, 8(3):195–218, 1998.
21. K. A. Yost. A survey and description of usaf conventional munitions allocation models. Technical report, Office of Aerospace Studies, Kirtland AFB, Feb 1995.