Automatic Generation of 2-AntWars Players with Genetic Programming

Johannes Inführ and Günther R. Raidl

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
{infuehr,raidl}@ads.tuwien.ac.at
http://www.ads.tuwien.ac.at

Abstract. In this work, we show how Genetic Programming can be used to create game playing strategies for 2-AntWars, a deterministic turn-based two player game with local information. We evaluate the created strategies against fixed, human created strategies as well as in a coevolutionary setting, where both players evolve simultaneously. We show that genetic programming is able to create competent players which can beat the static playing strategies, sometimes even in a creative way. Both mutation and crossover are shown to be essential for creating superior game playing strategies.

Keywords: Automatic Strategy Creation, Strongly Typed Genetic Programming, Game Rule Evaluation

1 Introduction and Applications

Being able to automatically generate competent artificial intelligence has a multitude of advantages. In this work, we use automatically created artificial intelligence to play a game, and in that domain such a method has three main advantages. First of all, during the creation of a game, a lot of different variations of game rules can be tested with this method. If for instance dominating strategies are found, then the game rules are unsuitable [11]. Secondly, the automatically created game players can be used to enhance the game implementation testing. The third advantage is that the created players can also be used as opponents for human players without having to painstakingly create decision rules or scripts. A method to automatically create strategies is also useful in many other domains requiring strategic behaviour, such as in business and economics.

AntWars is a competitive two-player game with local information that was introduced as part of a competition accompanying the Genetic and Evolutionary Computation Conference 2007 [1,6]. Both players control an ant in a toroidal world and have to collect randomly placed pieces of food. The player who collects more food wins. 2-AntWars is an extension of AntWars. The main aim of the extension was to create a game that allows various different strategies without an obvious best strategy. In 2-AntWars, each player controls two ants in a rectangular world four times the size of the AntWars world. Controlling

R. Moreno-Díaz et al. (Eds.): EUROCAST 2011, Part I, LNCS 6927, pp. 248–255, 2012.

two ants increases the complexity of the problem considerably because now each player has to decide which ant to move in addition to selecting the direction of the move. Furthermore, the decision to start a battle (by moving an ant onto a position already occupied by an ant of the enemy) requires more deliberation, because unlike AntWars, in 2-AntWars the attacked player still has a chance to win the battle. The complete description of 2-AntWars can be found in [5]. Figure 1 shows a possible initial configuration of the playing field of 2-AntWars. The marked areas around the ants show the portion of the playing field that can be seen by each player.

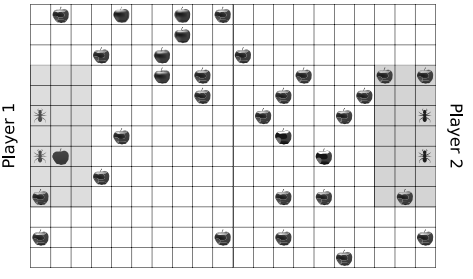


Fig. 1. The 2-AntWars playing field

2 Related Work

Using genetic programming (GP) to develop game players is not a particularly new idea. Even the first book of Koza [7] on GP already contained an example of automatic generation of a movement strategy for an ant that tries to follow a path of food (artificial ant problem) and a lot of research has been done since then. In [11], genetic programming was used to develop players of a turn based strategy game. In [2] space combat strategies were created. Other forms of predator-prey interaction were analyzed in [3] and [9]. Genetic programming has also been used to develop soccer [8] and chess end game players [4]. However, these works were mainly concerned with the end result, in this paper we also want to provide additional insight into genetic programming by analyzing the development of the population during a run.

3 Implementation

Genetic programming is an evolutionary algorithm that can be used for program induction [7]. The particular variant we use in this work is Strongly Typed Genetic Programming [10], as 2-AntWars makes use of different data types (e.g. position of an ant, movement direction, etc.) in a natural way. The population features a ring structure, and individuals, which are created using the ramped half-and-half method, are placed so that each has exactly two neighbors. The applied rank based selection operator uses a small neighborhood around an individual (seven individuals in total) to select the successor of that individual in the

next generation. The crossover operator used is subtree crossover, the mutation operators are point mutation (i.e. change a node in the program tree), replace mutation (i.e. replace a subtree of the program tree with a newly grown tree), grow mutation (i.e. add a node inside the program tree) and shrink mutation (i.e. remove a node from the program tree).

The task of a 2-AntWars player is to decide which ant to move in which direction. To increase solving efficiency, this task is decomposed into the following functions, developed independently by GP: (a) calculating the belief in food at every position of the 2-AntWars world (introduced by [6], extended in this work to allow development of this function), (b) calculating estimated positions of the opposing player's ants, (c) determining possible moves for both ants and finally (d) deciding which ant to move. The functions at the same position in the population constitute a 2-AntWars player.

For the details concerning the genetic programming implementation, the used function set and the player structure, see [5].

4 Results

In this section, we present the results of developing 2-AntWars players against three different fixed playing strategies and against another evolving population of players. The results were achieved by using a population size of 1000, performing 1000 generations, 60% crossover probability (per function of an individual), 0.1% mutation probability (per statement of a function) and a 30% chance for each of the four mutation types to be applied. In the following, a species is defined as a continuous part of the population that has different properties than the individuals surrounding it.

4.1 Greedy Opponent

The basic playing strategy of the greedy opponent is to concentrate on effectively collecting food and ignoring the other player. Figure 2 shows that the player developed by genetic programming is almost immediately as good as the greedy opponent. The rest of the run is spent on improving beyond the performance of the greedy player, so that in the end of the run the developed player significantly outperforms the greedy opponent. The division of labour of the developed ants was surprising. Instead of using both ants roughly equally often (as the greedy opponent does), the developed players use up the moves of one ant before they use the other ant. This behaviour was typical for all the performed runs.

While developing players against the greedy strategy, we tested the influence of mutation on the performance of genetic programming. The results are summarised in Figure 3. As can be seen, the developed players of the different runs reached two distinct performance levels. The deciding factor between those two levels is whether or not the player is able to use both ants. With mutation, there is a chance of about $^2/_3$ to develop a player belonging to the high performance group. When disabling mutation, the chance drops to about $^1/_8$, which demonstrates that, for the 2-AntWars problem, mutation plays a crucial role for developing new behaviours.

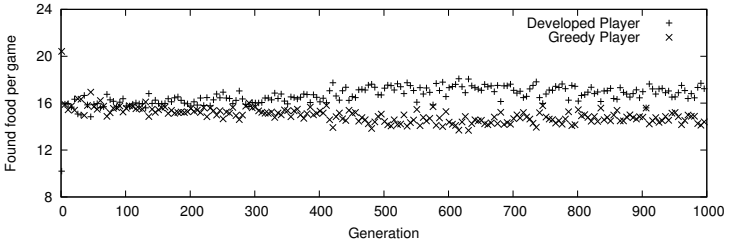
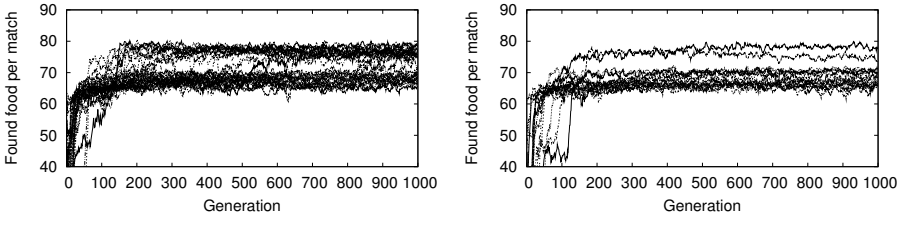


Fig. 2. Average number of found food pieces (after 50 games) of the best individual per generation against the greedy opponent



(a) with mutation (b) without mutation

Fig. 3. Population development of multiple runs with and without mutation against the greedy opponent

4.2 Scorched Earth Opponent

The Scorched Earth player trades the potential of high scores that the greedy strategy provides for increased security of winning the game. Since winning a game only requires to collect one piece of food more than half of the available food, the scorched earth player moves his ants quickly to the center of the playing field (ignoring food on the way), collects some food pieces from the enemies half of the playing field and then moves back to the starting position and collects all the food on its side. Figure 4 shows that while this strategy may be good in theory, it is beaten decisively by the developed player. The developed player adapted to the opponent by collecting food from the opponent's half of the playing field before it is collected by the opponent. The pronounced jump in performance around generation 190 is caused by reducing the greediness of the ant that is moved first; it leaves food on the players half of the playing field to be collected later by the second ant in order to reach the opposite half of the playing field faster.

4.3 Hunting Opponent

The Hunting strategy is the most aggressive strategy studied here. It relies on quickly neutralizing one or even both ants of the opposing player in order to gain a significant food gathering advantage. Figure 5 shows the development of

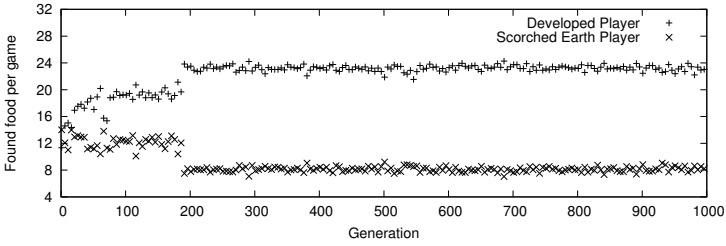


Fig. 4. Average number of found food pieces (after 50 games) of the best individual per generation against the scorched earth opponent

artificial players against this strategy. The hunting opponent proved to be the most difficult of the three opponent types to deal with. It took 2000 generations (instead of the default 1000) to develop a strategy that could barely beat this opponent, but the found counter strategy proved quite interesting: The developed player hides one of his ants at the top side of the playing field, then collects food with his other ant on the lower side of the playing field, while the hunting ant of the opponent searches the developed player's starting position in vain.

4.4 Evolving Opponent

The results of coevolutionary runs are outlined in this subsection. Figure 6 shows the results of a coevolutionary run that took the better part of a week (Core i-7 920, 2 threads) to complete, even though the population size was reduced to 500 (per player). During the first 4000 generations, player 1 is clearly superior, because player 2 was not able to use both of its ants, but after player 2 also developed the effective use of both ants, both player strategies struggled to beat one another, without a clear winner. From the game development point of view this is encouraging, because it shows that genetic programming was not able to find a dominating strategy.

Another effect that was clearly visible during a coevolutionary run was the influence of crossover on the overall performance, which is shown in Figure 7.

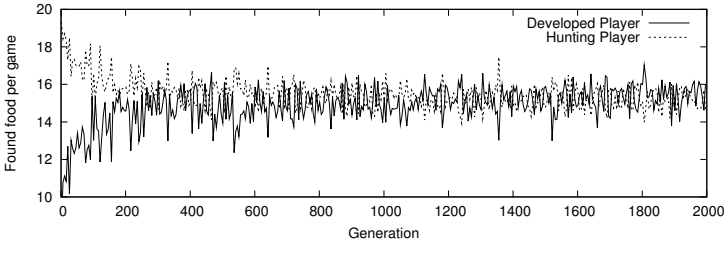


Fig. 5. Average number of found food pieces (after 50 games) of the best individual per generation against the hunting opponent

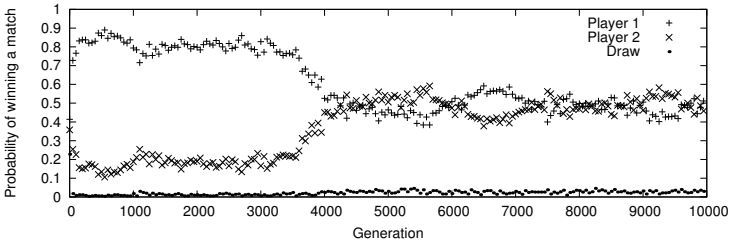
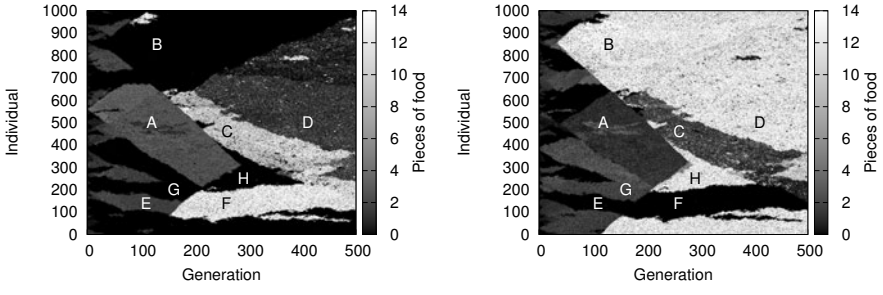


Fig. 6. Development of win probability during a coevolutionary run



(a) Found food ant 1 (b) Found food ant 2

Fig. 7. Influence of the crossover operator on the development of a player

After the first 100 generations, two dominant species existed in the population, which spread rapidly (labeled A and B). They had different means of achieving their performance. Species A used both of its ants, but every ant rather ineffectively, while species B very effectively used only one ant. Where both species met in the population, species C emerged, containing properties of both A and B. C used both ants to collect food, and moved ant 1 with the same proficiency as species B moved ant 2. Later species D emerged, changing the ant that was primarily used to collect food. The same effect can be seen in another part of the population, where B spread throughout the population and met species E. Immediately, species F arose, combining properties of both. The same thing happened when F met G, but this time it took about 20 generations, nearly driving G to extinction. This shows how fast attributes spread throughout the population with the help of the crossover operator and how new and improved individuals emerge when crossover combines useful behavioural traits.

4.5 Additional Results

Further results which we will not discuss here in detail due to space constraints include: Code bloat could be observed during the runs, but was not uniform. Some parts of the population stayed relatively small while others increased in size until the maximum allowed size was reached.

The belief function was instrumental in guiding the exploration of the playing field and a lot of different approaches to calculate the belief were found, but all except one switched directly between 0 and 1 without any intermediate values. The most common behaviour was row- or column-wise switching of food belief from 1 to 0 as the game progresses.

The benefit of the predict function was questionable. Mostly it predicted the opposing player's ants at some average position. Disabling the prediction function (by always returning the last known position) in some cases improved the performance of the players.

5 Conclusion and Future Work

In this work, we have shown how genetic programming can be used to develop playing strategies for 2-AntWars. Local rank selection was sufficient to give rise to different species inside the population which battle for space during the development of players. Both mutation and crossover are useful for developing players, albeit in different roles. Mutation can introduce new behaviour, as we have shown that removing mutation significantly decreases the chances of developing effective strategies for the use of both ants. Crossover is useful to combine traits of different playing strategies to create a superior strategy.

The created playing strategies were surprisingly diverse. The players followed greedy strategies (against the greedy opponent), exploited weaknesses in the enemy's strategies (against the scorched earth opponent) or hid their ants (against the hunting opponent) to win their games. Even though the movements of the ants showed strong greedy tendencies (due to the supplied terminals), non-greedy behaviour could also be observed.

One direction for future work ist a better tuning of parameters, because for example in most of the presented runs 1000 generations were too much. It was also shown that the prediction functions were of questionable benefit, so they could either be improved or replaced by something else, possibly functions that just calculate waypoints, to be used by the movement functions in any way. The game rules themselves are another interesting topic for future research. The discussion of the results against the greedy opponent showed that one ant was nearly sufficient for two competent players to collect all the food on the playing field, so the number of moves possible for an ant might be reduced. Evaluation of the developed strategies against human players could further provide additional insight and determine whether or not the developed strategies are human competitive.

References

1. Antwars competition at the Genetic and Evolutionary Computation Conference 2007 (October 2010),
`http://www.sigevo.org/gecco-2007/competitions.html#c3`

2. Francisco, T., Jorge dos Reis, G.M.: Evolving combat algorithms to control space ships in a 2d space simulation game with co-evolution using genetic programming and decision trees. In: GECCO 2008: Proceedings of the 2008 GECCO Conference on Genetic and Evolutionary Computation, pp. 1887–1892. ACM, New York (2008)
3. Francisco, T., Jorge dos Reis, G.M.: Evolving predator and prey behaviours with co-evolution using genetic programming and decision trees. In: GECCO 2008: Proceedings of the 2008 GECCO Conference on Genetic and Evolutionary Computation, pp. 1893–1900. ACM, New York (2008)
4. Hauptman, A.: GP-endchess: Using genetic programming to evolve chess endgame players. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 120–131. Springer, Heidelberg (2005)
5. Inführ, J.: Automatic Generation of 2-AntWars Players with Genetic Programming. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms (2010)
6. Jaskowski, W., Krawiec, K., Wieloch, B.: Winning ant wars: Evolving a human-competitive game strategy using fitnessless selection. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 13–24. Springer, Heidelberg (2008)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems). The MIT Press, Cambridge (1992)
8. Luke, S., Hohn, C., Farris, J., Jackson, G., Hendler, J.: Co-evolving soccer softbot team coordination with genetic programming. In: Kitano, H. (ed.) RoboCup 1997. LNCS, vol. 1395, pp. 398–411. Springer, Heidelberg (1998)
9. Luke, S., Spector, L.: Evolving teamwork and coordination with genetic programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, pp. 150–156. MIT Press, Stanford University (1996)
10. Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation 3(2), 199–230 (1995)
11. Salge, C., Lipski, C., Mahlmann, T., Mathiak, B.: Using genetically optimized artificial intelligence to improve gameplaying fun for strategical games. In: Sandbox 2008: Proceedings of the 2008 ACM SIGGRAPH Symposium on Video Games, pp. 7–14. ACM, New York (2008)