# 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

## 1 Introduction

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 [1,3]. 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. In 2-AntWars, each player controls two ants in a rectangular world four times the size of the AntWars world. Controlling 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, and he has to keep the location of the ants in mind because moving an ant into the boundary of the world would make it immovable. Furthermore, the decision to battle with an ant of the opponent (by moving an ant to a location that is occupied by an ant of the enemy) requires more finesse than in AntWars. In AntWars, the aggressor wins instantly as the player who is attacked can not counteract. In 2-AntWars, the defending player has the possibility to move his second ant to the position of the battle to win. The complete description of 2-AntWars can be found in [2].

In this work we studied how Genetic Programming can be used to create competent 2-AntWars players.

## 2 Genetic Programming

Genetic Programming (GP) is an evolutionary algorithm that can be used for program induction [4]. The particular variant we use in this work is Strongly Typed Genetic Programming [6], 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 to select individuals for the next generation. The used crossover operator 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).

## 3 Structure of the 2-AntWars player

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 subtasks: calculating the belief in food at every position of the 2-AntWars world, calculating estimated positions of the opposing player's ants, determining possible moves for both ants and finally deciding which ant to move. Thus, GP has to evolve six different functions in total, one for each of these subtasks. Food belief is a concept developed for AntWars to judge if an unseen position of the world contains food or if food that was seen (but cannot be seen currently by the ants) is still there [3]. In its original form, food belief was calculated by a static function, in this work the calculation function for food belief is allowed to change. Every function of the 2-AntWars player is assigned a separate score, except for the functions determining the moves and the one deciding which ant to move. Those three functions form a function group that is assigned a single score, because what the decision function should do depends strongly on the functions calculating ant moves. Enforcing a specific behaviour with a fitness function would be very restricting. Each function (or function group) is evolved separately, the functions sharing the same position in the population constitute a 2-AntWars player during evaluation.

## 4 Results

The developed 2-AntWars players are evaluated in an evolutionary setting, competing against handcrafted players, and in a coevolutionary setting, against a second population of evolving 2-AntWars players. We show that local rank selection is sufficient to give rise to different species inside the population which battle for space during the development of 2-AntWars players. Mutation and crossover are useful for the development of players, albeit in different roles. Mutation helps with the creation of new behaviours, crossover combines behaviours to form superior players. The created behaviours are surprisingly diverse. Ants are hidden, used as bait, and generally exploit the weaknesses of the opposing players. The movement of the ants shows strong greedy tendencies, but non-greedy behaviour can also be observed when it is advantageous. The players created by coevolutionary methods generally perform better than the players created by playing against fixed playing strategies. Code bloat [7] is observed during player development (and reined in by an upper limit of statements per function and lexicographic parsimony pressure [5]), but it was not uniform across the population. Parts of the population increase their size drastically while other parts show only moderate bloat.

## References

1. Antwars competition at the Genetic and Evolutionary Computation Conference 2007 (Oct 2010), http://www.sigevo.org/gecco-2007/competitions.html#c3
2. Inführ, J.: Automatic Generation of 2-AntWars Players with Genetic Programming. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms (July 2010), supervised by G. Raidl
3. Jaskowski, W., Krawiec, K., Wieloch, B.: Winning ant wars: Evolving a human-competitive game strategy using fitnessless selection. In: O'Neill, M., et al. (eds.) EuroGP. LNCS, vol. 4971, pp. 13–24. Springer (2008)
4. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems). The MIT Press (1992)
5. Luke, S., Panait, L.: A comparison of bloat control methods for genetic programming. Evolutionary Computation 14(3), 309–344 (2006)
6. Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation 3(2), 199–230 (1995)
7. Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines 10(2), 141–179 (2009)