ISSN:0974-7230

Journal of Computer Science & Systems Biology

The International Open Access Journal of Systems Biology and Applied Computer Science

Available online at: OMICS Publishing Group (www.omicsonline.com)

Digital Object Identifier: http://dx.doi.org/10.4172/jcsb.1000099

Computer Science
Systems Biology

Headleand and Teahan, J Comput Sci Syst Biol 2013, 6:2
http://dx.doi.org/10.4172/jcsb.1000099

Research Article Open Access

Grammatical Herding

Chris Headleand and William J. Teahan*

School of Computer Science, Bangor University, Bangor, Wales, UK

Abstract

Automatic programming algorithms have often looked towards biology to provide inspiration for how agents can learn to solve problems. Evolutionary and swarm based methods in particular have shown great promise in how this objective can be achieved. We present Grammatical Herding, a new fitness-based automatic programming algorithm based on a simple set of rules inspired by the herd movements of horses. In this paper, we establish the design of the new algorithm and test it against a standard benchmark problem, the Santa Fe Trail.

Introduction

Self-programming agents promise to play a significant role in the future of AI with systems designed that can adapt to a changing environment. Various algorithms have been designed to enable systems to achieve this goal of iteratively creating or improving candidate programs. While there are many proposed algorithms, an interesting subset is the systems that take direct inspiration from biology, evolutionary algorithms being an obvious example. These machine-learning techniques apply simple rules but have the capacity to generate potentially complex behaviours.

Another example of simple rules within a natural system is the way birds flock. Without a central controller or directed intelligence, the creatures are able to swarm and navigate towards a common goal. This flexible, decentralised, self-organising behaviour has been applied to various computer algorithms including Craig Reynolds boids [1]. It also serves as the inspiration for Particle Swarm Optimisation [2].

Particle Swarm Optimisation (PSO) is a computational method that optimises solutions to a problem iteratively moving candidates (particles) around the problem search space. The movements of the particles are directly influenced by the position of their current personal best and the best solutions found at positions discovered by other particles. These movements (the position and velocity of the particle) are updated iteratively as new solutions are found by the swarm. The assumption is made that the population of particles (the swarm) will cohere towards an optimum solution.

This paper examines a new approach to automatic program generation through application of the Swarm paradigm. It proposes additional rules that can be applied to improve search based automated programming algorithms. A secondary aim is to show that biology can continue to serve as an inspiration in field of AI and/or improve existing algorithms.

Automatic Programming

Automatic programming is a collection of methods for generating programs without the need for human intervention. In the following section, we will briefly explore a subset of this group that has taken inspiration from biology and evaluate their individual merits.

Grammatical evolution

Grammatical evolution (GE) is an evolutionary algorithm where a program in an arbitrary language is evolved over several generations based on assessing the fitness of each member of the population and selectively breeding the best candidates [3]. The genotype of the agent is mapped using a context free grammar to produce the program (phenotype).

A favoured alternative evolutionary method for constructing programs is Genetic Programming (GP) where usually a tree-like structured expression is directly manipulated during genetic crossover and mutation [4]. In contrast to this, GE applies the genetic operations to a binary string genome, which is then subsequently mapped to the program using a context free grammar. This mapping process has an advantage over GP of reducing bloat and increasing the possibility that each member of the population will be valid.

Further improvements have been made to grammatical evolution in the form of Constituent Grammatical Evolution (CGE) [5]. This extends the algorithm with the additional bio-inspired concepts of constituent genes and conditional behaviour switching.

Grammatical swarm

Grammatical Swarm (GS) combines the Particle Swarm algorithm with the GE-like genotype-to-phenotype mapping to generate programs in an arbitrary language [6]. The standard equations for PSO are used with constraints placed on velocity, trajectory and partial location values.

Grammatical Swarm is particularly interesting as the agents generating programs within the system can be considered as "embodied" in the way that they explore the solution space represented as an N-dimensional environment. O'Neill and Brabazon [6] stated that in experiments conducted against standard benchmarks such as the Santa Fe Trial, GS generated comparative or in some cases better solutions than GE.

Background and Inspiration

When considering the swarm paradigm, the common examples used are birds and insects. But there are many variations of this particular class of group behaviour to be found in significantly more

***Corresponding author:** William J. Teahan, School of Computer Science, Bangor University, Bangor, Wales, UK, E-mail: w.j.teahan@bangor.ac.uk

Received December 26, 2012; **Accepted** January 19, 2013; **Published** January 22, 2013

Citation: Headleand C, Teahan WJ (2013) Grammatical Herding. J Comput Sci Syst Biol 6: 043-047. doi:10.4172/jcsb.1000099

complex animals, including humans; for example, crowd behaviour. This raises the following question: "Could further inspirations from more complex creatures be applied to algorithms such as Grammatical Swarm to extend the approach?"

Horses, for example, have some very interesting behavioural traits that are additional to the standard flocking that makes their herd movements quite unique.

Herd behaviour of horses

Within the hierarchy of the herd, there are several specific roles, which have evolved to optimise the group's movements when grazing and being chased by predators. The following three areas of their social behaviour were identified as possible sources for inspiration.

Lead Mare: The lead or "boss" mare directs the group's movements based on knowledge of the terrain and the resources available. She is generally the fittest mature female in the herd with the greatest experience of the environment.

Herd Stallion: The role of the lead stallion is to patrol the fringes of the herd. This is partly to protect the mares from predators but also to drive straggling members back towards the group. By pushing the weaker members forward, they are less likely to be lost to unfamiliar territory or attacked by predators [7,8].

Grazing behaviour: Herds of domesticated horses were observed by the primary author of this paper at various locations around North Wales and West England to gain observational data regarding how they searched the land for resources during grazing.

Their behaviour indicated that each horse would remember particularly lush sections of the grazing space. But occasionally they would look up and wander towards other horses within the herd; at a seemingly random point during their movement, they would drop their head and check the fauna.

Application

In order to make practical use of these observations for an algorithm, we first considered how they could be simplified into basic rules. The following rules were established, to be applied to the design of our new algorithm:

1. Agents will traverse the search-space by moving between their remembered personal best position (pB) and the best positions found by the fittest members of the population. This corresponds to the type of flocking behaviour observed by the horses during grazing.
2. A point will be selected between their personal-best location and the personal best location of a high fitness agent as opposed to searching the entire space. This corresponds directly to the search type behaviour observed during the horses grazing where junior members of the herd would move towards the lead mares.
3. The weakest agents will be driven towards the locations of the fittest agents. This corresponds to the herding behaviour enforced by the herd stallions.

The New Algorithm

Nomenclature

Several phrases are used repeatedly within the following sections in the context of the system design (Table 1).

Fitness	A value representing the successfulness of the program after evaluation.
Herd	The population of agents within the search.
Coordinates	The binary string used to identify the positions of each agent within the search.
Betas	An arbitrary number of agents within the population with the highest personal-best fitness. The pB of these agents is used to steer the search of the other agents within the system.
Alphas	A subset of agents within the betas with the highest personal-best fitness. These agents are used as targets for the weakest members to be driven towards.
pB	The memory of the personal best position achieved by an individual agent. This memory includes the fitness score and the location that fitness was found in.

Table 1: Phrases used repeatedly in the context of the system design.

Design and bio-inspiration

Our new algorithm generates executable programs by traversing the possible program search space with a swarm of embodied agents. These agents treat the solution space as an N-dimensional environment and traverse it based on both their memory of high fitness locations within that environment and the locations of the fittest members of the herd.

When the population is first spawned, they are created with a binary string (the coordinates). This binary string is systematically split into a subset of decimal values that represent their current location and these values specify the dimensions within the search environment. Whenever an agent moves to a new location by traversing towards a beta (high fitness) agent, the new location is stored and through a GE type mapping process is converted to executable code.

This process of movement, and evaluation, is continued and at each iteration, if the fitness has been improved from the agent's current pB, the "memory" of the agent is updated to represent this.

Additional to this process, we established the following rules inspired by PSO, Grammatical Swarm and observations taken from herds of horses. These six basic rules govern the movements of the agents within the environment and the subsequent programs that are generated.

Rules:

1. When an agent moves to a new position within the search space, a program is compiled based on their current coordinates. This program is subsequently evaluated and provided a score based on its fitness.
2. Each agent maintains a memory of the score and position where they achieved their personal best fitness.
3. A list is maintained containing a subset of the agents that have achieved the highest personal best fitness and the coordinates at which it was achieved. This list is referred to as the betas list.
4. Each agent will move between the coordinates at which they found their personal best fitness and the coordinates of the personal best of a random member of the "betas".

5. The fittest agents within the betas list are known as the alphas.
6. The members of the herd with the weakest personal-best fitness are driven towards a random member of the alphas. Their personal-best coordinates are set to the same as the personal-best fitness of the selected alpha.

The Santa Fe Trail

The Santa Fe Trail is an example of an artificial ant problem used to benchmark automatic programming algorithms such as GE and GP. It consists of finding a set of rules that allow an agent to find food along a predefined trail. The fitness of the ant is measured by the amount of food the ant is able to find along the trail. To complete the problem, the ant has a limited amount of actions it can perform to solve the problem:

MOVE	: Move forward one step.
TURN-LEFT	: Turn left 90 degrees.
TURN-RIGHT	: Turn right 90 degrees.
FOOD-AHEAD	: Is there any food directly ahead?

The problem is interesting from a programming perspective as there are several possible solutions each with their own qualities. Which solution is "best" depends on the subjective conditions used for evaluation. One option could be an optimum solution that takes the least total steps by simply following the path with movement operators and no sensing steps. This method would certainly be the "fastest" for the ant taking the least total operation. Alternatively a solution that takes more steps but is more general, allowing the ant to sense its way around the trail could be rated higher.

An Initial investigation

A model was built in NetLogo to test the application of the rules against a standard benchmark for automated programming, notably the Santa Fe Trail. The jGE extension was used to generate the coordinates of the initial population and map the resulting bit strings of each generation using a BNF grammar to create the programs [9].

As previously described, the system treats each binary string as coordinates for the current location within the search space. The binary string is broken down into sets of dimensional coordinates by splitting the binary string into several equal length sub-strings. Each of these smaller strings is treated as one of the coordinate dimensions to position the agent within the search space environment. These individual coordinates are treated as codons by the jGE extension during the grammar mapping in the standard GE genotype to phenotype process.

With this information available, it is relatively simple to plot the position of the agents within the search space. Each agent within the "herd" is also provided with the facility to store a memory of the position they visited that achieved the highest fitness and is set as their personal-best (pB) location.

After each iteration, the agents select a random member of the betas list and move towards it, adjusting their coordinates to equal a position between their own pB location and the pB location of the beta they had selected. This position was determined by weighting the attractions of the two points in gravity like fashion. The equilibrium point between the pull of the pB and the pull of the target agent's pB was set as the new position. Once all the agents have selected new coordinates, the binary string is mapped to code using the GE genotype to phenotype method. This code is subsequently compiled and the fitness evaluated. If the fitness of the new location is higher than their previous pB, the pB is updated for the next iteration.

Additional investigations

Three further experiments were conducted to explore possible optimisation of the method.

Movements: As mentioned previously, a weighted attraction method was used to provide the results found in this paper. However, two other methods were evaluated that showed promise for further evaluation.

1. The position between the agent's pB and the pB of the target agent was set to random. This method generally took significantly longer to find a solution; however, there were fewer instances of the agents getting stuck at local optima.
2. The distance between the two agents was evaluated and the resulting value was used to position the agent either between the two pB scores or ahead of the target pB (along the same trajectory). This took inspiration from the velocity/momentum approach used in PSO. This was expressed as follows:

L : The new location.

T : The target pB location.

R : The range between the agent's pB location and the pB location of the target (T).

$L = T + (\text{random } R - \text{random } R)$.

Dimensions of the search space: As we manipulate a binary string as our coordinates, this can be utilised in several different ways. This experiment chose to treat each codon (to use the GE paradigm) as a separate dimension. However, a further experiment was taken which simplified the coordinates by splitting the string into 8 sub-strings (as opposed to 15). Interestingly, this reduction in dimensions did not create a significant drop in the performance of the search. Similarly, increasing the dimensional search by splitting the string into 30 dimensional coordinates had little effect on the algorithm.

Results

The interface of our model allowed us to test various attributes within the simulation. These specific variables were defined as:

1. The amount of agents within the betas group.
2. The amount of agents within the alphas group.
3. The maximum number of iterations.
4. The total size of the herd (population).

The results showed that when the herd contained a sufficient number of agents and the betas were focused (our most successful experiments have the betas at a 2:100 ratio), the algorithm was almost always capable of producing a high fitness generalised solution where the agents sensed their way along the trail.

The following table (Table 2) displays our results for the experiments we ran with our model.

H = The amount of agents within the herd.

H	B	A	I	S	F
1000	30	5	250	635	100
1000	20	5	172	397	100
1000	20	2	356	900	69.6
700	30	5	375.5	785	100
700	20	5	221.5	737	100
700	20	2	400	900	97
400	30	5	400	900	38.2
400	20	5	102	900	62.9
400	20	2	43	900	47.1

Table 2: Selected results produced via applying our algorithm to the Santa Fe Trail.

B = The amount of agents within the betas.

A = The amount of agents within the alphas.

S = The average movement steps to complete the trail (i.e. MOVE, TURN-LEFT or TURN-RIGHT).

I = The average number of iterations before no further improvements were made to the fitness; 400 was set as a maximum cut-off.

F = The best fitness (percentage of trail solved).

The agents were allowed to take a maximum number of 900 steps; 10 runs were preformed per experiment.

Sample code produced

The following generated code sample was able to complete the trail in the least number of movement steps (397) the system was able to find.

ifelse food-ahead

[move]

[turn-right

ifelse food-ahead

[move]

[turn-right

turn-right

ifelse food-ahead

[move]

[turn-right]

move]]

By comparison, the current state of the art algorithm (CGE) was capable of generating code that completed the trail in 337 steps, although most solutions mentioned in the literature (for example, for GP) take around 400 steps or more.

False positives

Whilst the majority of experiments produced programs where the ant would follow the trail, on occasion the agents would home in on a "false positive" solution that produced high fitness early in the iterations but was inefficient compared to the desired trail following behaviour. Figure 1 shows one program generated by the system that produced high fitness but used all the available steps.

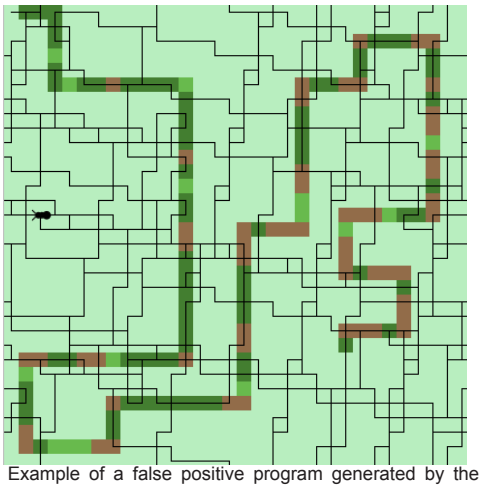

Figure 1: Example of a false positive program generated by the proof of concept model. Interestingly, when allowed to run to 2000 steps, the search pattern of the ant exhibits fractal-like properties.

This problem of getting trapped in local optima solutions within a search is a known issue within the field of self-programming agents. Evolutionary algorithms cope with this concern through mutation. By comparison, the algorithm discussed here tries to overcome this problem through the swarming effect of the herd. However, neither method provides a full proof solution to this problem.

Discussion

The purpose of this new algorithm was to see if we could devise a method where executable code was generated by embodied agents traversing a solution environment. While methods have been developed to do this previously, the aim was to see if further improvement could be achieved by taking inspiration from the observation of biological systems (i.e. horses) that have not been previously used in bio-inspired AI.

The rules established for exploration of the solution space require additional investigation. While the majority of experiments created a successful program (one that was able to achieve a full fitness of 100%), several failed to reach this goal. However, the new method was generally able to home in on moderate to high fitness solutions quicker than GE, often achieving a fitness of over 50% within 10 iterations. A strength of the algorithm was that the majority of the code generated provided a general solution to the problem of trail following and would have preformed equally efficiently on other trail configurations.

To further develop the method, future studies could take additional inspiration from human nature and consider a problem-solving pipeline. When we ourselves are presented with a problem, a common response is to quickly generate a hypothesis by searching the knowledge we have available. We then implement our possible solution, before evolving it through a process of practice and evaluation. An algorithmic equivalent could be to use GH to quickly search a solution space to seed the population of a GE search. Creating a hybrid algorithm could have the overall effect of increasing the speed of GE and increasing the accuracy of GH.

References

1. Reynolds C (1987) Flocks, herds and schools: A distributed behavioral

model. Proceedings of the 14th annual conference on Computer graphics and interactive techniques, SIGGRAPH, New York.

2. Kennedy J, Eberhar R (1995) Particle swarm optimization. Neural Networks. IEEE International Conference on Neural Network, Perth.
3. Ryan C, Collins JJ, O'Neill Michael (1998) Grammatical evolution: Evolving programs for an arbitrary language. Proceedings of the First European Workshop on Genetic Programming 1391: 83-95.
4. Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Massachusetts.
5. Georgiou L, Teahan WJ (2011) Constituent Grammatical Evolution. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence.
6. O'Neill M, Brabazon A (2006) Grammatical Swarm: The generation of programs by social programming. Natural Computing 5: 443-462.
7. Mistral K (2005) The Secret Life of Stallions.
8. Utah State University (2009) Wild Horse Behaviour. Animal, Dairy and Veterinary Sciences.
9. Georgiou L, Teahan WJ (2006) jGE - A Java implementation of Grammatical Evolution. ICS'06 Proceedings of the 10th WSEAS international conference on Systems, Athens.

Citation: Headleand C, Teahan WJ (2013) Grammatical Herding. J Comput Sci Syst Biol 6: 043-047. doi:10.4172/jcsb.1000099