Journal of
Computer Science & Systems Biology

Headleand and Teahan, J Comput Sci Syst Biol 2013, 6:3
DOI: 10.4172/jcsb.1000110

Research Article Open Access

Swarm Based Population Seeding of Grammatical Evolution

Chris Headleand and William J Teahan*

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

Abstract

Evolutionary Algorithms, although powerful, are known to be wasteful and time consuming, requiring the evaluation of a large number of candidates. However the strength of the methodology is their ability to continually optimise the population hopefully ensuring a near optimal final solution. When applied to automatic programming tasks, the same limitations are observed, notably the time taken to develop a solution. An alternate, swarm-based method 'Grammatical Herding' suffers from the opposite concerns. Whilst it generates moderate fitness solutions quickly, these candidates often lack the optimisation of solutions generated via an evolutionary approach. This study details a hybrid technique 'Seeded Grammatical Evolution' where Grammatical Herding (GH) is used to seed the initial population of a Grammatical Evolution (GE) algorithm, with the result that the final solution is produced faster than one produced by GE alone and more effective (fitter) than one produced by GH. In this paper, we explore the background to the study including the initial work that inspired the approach. We also discuss the design of the algorithm and finally the results. We conclude that the hybrid approach is not only capable of producing a fast solution but also achieves state of the art results on a standard benchmark problem, the Santa Fe Trail.

Keywords: Algorithms; Evolution; Genomea

Introduction

Automatic Programming (AP) solves problems iteratively by evaluating and further optimising candidate solutions represented as computer programs. One modern approach to AP are the grammar based techniques, notably: Grammatical Evolution (GE) [1], which follows the evolutionary algorithm paradigm; Grammatical Swarm (GS), a particle swarm optimisation approach [2]; and Grammatical Herding (GH) which takes inspiration from the herding of mammals [3]. The grammar-based techniques are an example of bio-inspired artificial intelligence, with the GE algorithm inspired by the biological process for generating a protein. In this process, the genome consisting of DNA (this corresponds to the binary string in GE) is transcribed to RNA (the grammar in GE) and translated into amino acid sequences and used in the protein resulting in the phenotype (the output program in GE).

This family of techniques has been incorrectly compared to Genetic Programming (GP), but both approaches are fundamentally different in their methodologies. The main fundamental differences are found in both their crossover and mutation functions. GP also manipulates code directly as a tree, whereas the grammar techniques instead manipulate a binary string that is subsequently mapped via a grammar to executable code. This mapping process is what sets the grammar techniques apart as they can be considered language independent, able to evolve code given (usually) a context free grammar expressed in BNF. The fundamental differences between these methodologies are further discussed in detail in [4] and a full exploration of the crossover operation can be found [5].

However, evolution with these algorithms takes a significant number of iterations, each of which requires usually hundreds of unique candidate evaluations to solve a problem. There is a scaling problem as the number of candidates requiring evaluation increases substantially with the complexity of the problem due to the increasing size of the search space. Additionally, there is no guarantee that the final generated solution will be optimal, or meet a given success criteria defined by the fitness function, with the search often becoming trapped in local optima.

This paper discusses the development of a novel grammar-based hybrid algorithm devised to further improve the efficiency of the problem solving process. Our new solution, seeded Grammatical Evolution (sGE), takes the efficiency advantages from a swarm-based technique (GH) and the optimisation advantages of Grammatical Evolution with the aim to further enhance the current state of the art.

Background and Motivation

Evolutionary algorithms, whilst being a powerful technique, take a significant amount of processing time to solve a problem due to the number of individual candidate evaluations required. Natural evolution is a slow process of trial and improvement taking millions of years; its computational equivalent (whilst not taking quite as long) suffers from the same blind watchmaker method [6] of iterative improvement. Simply put, the solutions are not designed, but found through an artificial form of natural selection.

Whilst the physical time taken can be significantly reduced through parallelism and an increase of processing power, an alternative approach would be to reduce the total number of generations and the number of individual evaluations required to achieve the objective. In a previous study [3], it was noted that Grammatical Herding was more efficient at generating a solution, but this was at the sacrifice of optimisation. In certain real world applications, a solution needs to be generated quickly, but sufficiently optimised to achieve the desired objective, especially if autonomous agents are to be human competitive.

This understanding from our previous research provided the insight that led us to the new hybrid algorithm discussed here which uses Grammatical Herding to seed Grammatical Evolution. The next two sections identify the principle differences between these two algorithms.

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

Received June 04, 2013; **Accepted** July 17, 2013; **Published** July 20, 2013

Citation: Headleand C, Teahan WJ (2013) Swarm Based Population Seeding of Grammatical Evolution. J Comput Sci Syst Biol 6: 132-135. doi:10.4172/jcsb.1000110

Grammatical evolution

Grammatical Evolution (GE) is a genetic algorithm based automatic programming method where candidate solutions are evolved over repeated generations [1]. The system was designed in part to avoid some of the complications presented in genetic programming (an alternate program synthesis method). By manipulating a bit string before mapping via usually a context free grammar, the candidate solution can be evolved without the need to directly manipulate code. The flow diagram for the method is shown in Figure 1.

This method has been further developed with Constituent Grammatical Evolution [7]. This takes additional bio-inspiration from the concepts of constituent genes and conditional behaviour switching producing state-of-the art results on various benchmark problems such as the Santa Fe Trail problem.

Grammatical herding

We recently described a novel automatic programming algorithm called Grammatical Herding [3]. The aim of the algorithm is to use a swarm-based search heuristic for automatic programming in an arbitrary language. Many swarm systems have demonstrated that simple rules can produce complex results, such as Boids [8] or Particle Swarm Optimisation [9]. It is from the latter that Grammatical Herding takes its initial inspiration. Instead of looking to insects, it bases its rule set on the movement of mammals.

Utilising the Grammatical Evolution genotype to phenotype mapping process and inspiration from herd movements, it treats the possible solution space as an environment and drives candidates towards areas of the search space known to produce high fitness solutions. A flow diagram for our algorithm is provided in Figure 2.

In Grammatical Herding, three classes of agents are maintained. Firstly there is the herd, which contains all agents, equivalent to the population in standard genetic algorithms. Within the herd, there are 'Betas', a subset of the fittest agents, which are used to direct the movements of the herd based on their best location and fitness. Within the Betas, a final subset of agents is maintained known as the 'Alphas'. The Alphas are the Betas with the highest fitness. These are used to steer underperforming agents by setting their current location to the best location of the Alphas. This has the effect of "herding" weaker members of the herd towards higher fitness areas within the search space. The inspiration from this comes directly from observations of herds of horses in the wild where stallions drive weaker/younger members of the herd towards the Alpha mares (Figure 3).

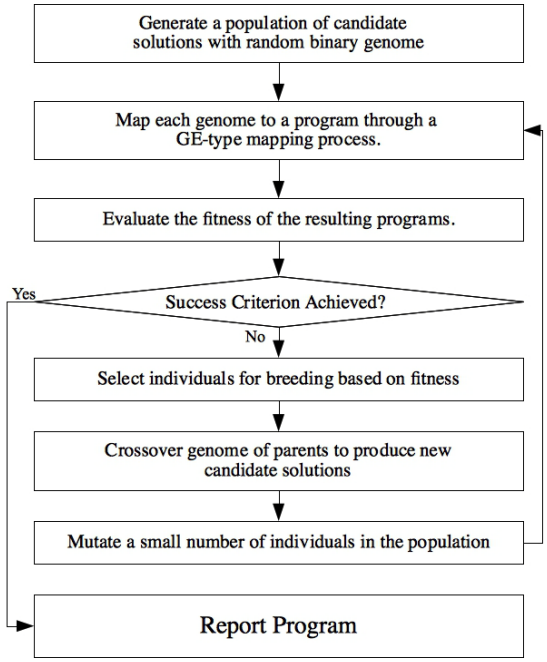

Figure 1: Design of the Grammatical Evolution algorithm.

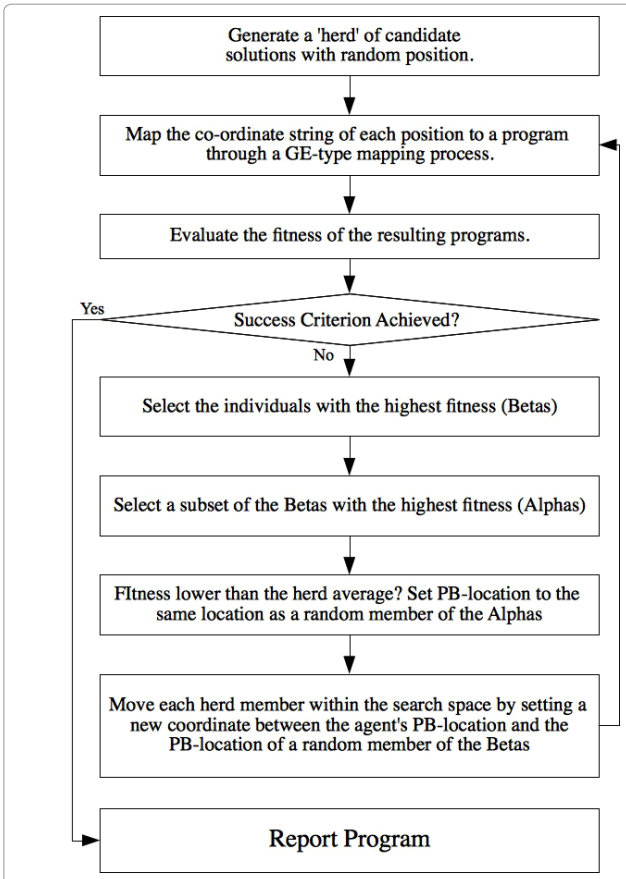

Figure 2: Design of the Grammatical Herding algorithm.

The PB-location (Personal Best location) is the multi-dimensional coordinate within the search space where an agent achieved its highest fitness. For more details on the design of the algorithm [3].

GH was selected as the swarm algorithm to seed GE in this paper as it shares the same mapping process. This also allowed us to use the same system for generating the initial genome string [10-13]. This limited the risk of additional programmatic bias in the implementation.

Seeded Grammatical Evolution

In this section, we discuss the design and implementation of the hybrid algorithm. The algorithm has two main stages, the Seeding Stage where GH is used and the Optimising Stage where GE is used (Figure 4). This section specifically focuses on the seeding process and the criteria for 'handover' before the experimental results.

Design

For this initial study, both algorithms were used unmodified, with

the focus being on using GH to seed the initial population of GE. However, this required that two key decisions be made, notably:

1. At what threshold should the algorithm switch from swarm to evolutionary computation?
2. How should the GE population be seeded, and what ratio of the GH herd should form the GE population?

These considerations are discussed in detail over the following sections. For the purpose of this discussion, 'threshold' refers to a value-based criterion for performing handover. Handover is the process of moving from the seed algorithm (GH) to the optimisation algorithm (GE).

Fitness threshold

The fitness threshold changeover involved moving from GH to GE when a specific fitness had been achieved by one of the agents within the herd. The principle concept behind this method was to take advantage of the speed that GH was able to traverse the possible search space. It had been noted in a previous study [3] that GH was able to achieve a fitness of over 50% within relatively little iteration. The hope was that, by using this method, the high fitness areas within the search environment could be identified quickly and passed to GE for more fine-grained optimisation. For a pilot study, four different fitness thresholds were trialled (20%, 40%, 60% and 80%) to gauge their performance. This preliminary investigation indicated that the 40% to 60% fitness threshold experiments produced the best results. Due to this, a threshold of 50% was set in the main study. It was assumed that this produced useful results, as although a strong population had been generated, the swarm had not yet converged (ensuring high genetic diversity).

Figure 3: Two stallions drive a herd of feral horses forward. Photo taken at a ranch in North America.

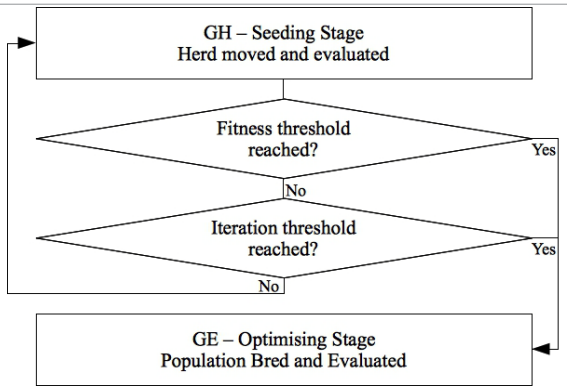

Figure 4: Design of the seeded Grammatical Evolution algorithm.

Iteration threshold

In addition to the fitness threshold, an iteration threshold was set for the GH stage. This defined when the changeover occurred after a fixed number of iterations had been completed. The reasoning behind this second threshold was as a fallback position if GH was unable to find a high fitness solution. By placing a limit on the number of iterations, GE could automatically take over if the GH algorithm was unsuccessful.

For the purpose of this algorithm, the iteration threshold was set at 100 iterations.

Seed ratio

A final consideration was how many members of the GH herd should be selected for breeding. Initially, the thought was to transition the whole herd into the GE population (a 100% seed ratio). However, this transition process could also be an opportunity to add some additional genetic diversity into the initial GE population. With that thought considered, the main study was comprised of four experiments (each containing 100 individual runs) each utilising a different seed ratio. The seed ratios explored in this study were 25%, 50%, 75% and 100% (i.e. full population handover).

Experimental Results

In this section, we discuss the results from our initial study on the Santa Fe Trail problem. The Santa Fe Trail problem was selected due to the quality of the background research in this area. As a benchmark test, the Santa Fe Trail problem has a very clear objective with several well-documented solutions, allowing us to compare final results with other AP algorithms such as genetic programming.

Solving the trail involves finding a set of rules that allow an agent to find food along a predefined path with gaps. The agent is able to sense one step ahead of itself, turn left and right, and move forward. The quality of the final solution is ascertained by counting the number of 'steps' taken to complete the trail, the lower the number of steps the stronger the solution. Currently the best-published solution (according to our review of the literature) is the one discussed by Georgiou and Teahan [7] which takes 337 steps. The fitness of each candidate solution is quantified by the amount of food the agent is able to eat along the trail with a maximum number of 89 (100% fitness).

Each experiment was conducted using the settings for GH and GE listed in Table 1. These settings were based on research conducted by Headleand and Teahan [3] and through some preliminary benchmarking between GH and GE. Our initial benchmarking led us to believe that these setting may be optimal for this problem.

The noticeable differences in settings are the maximum and minimum number of codons as GE utilises a varied length string; GH by comparison uses a fixed length string. Also, note that GE mutates a small number of the population; GH, however, sets all Alphas to wander randomly within the environment. The crossover in GE is fixed at 0.7 in these experiments; in GH, however, the 'crossover' depends on the relative position of the agent's current personal best (PB) location and the PB location of the Beta it is moving towards, utilising a weighted average function described in [3].

Each experimental set consisted of 100 individual runs. Each run was given a maximum of 500 to find a solution.

Settings	GH	GE
Codon length	8 bits	8 bits
Minimum number of codons	20	15
Maximum number of codons	20	25
Size of herd / population	700	700
Mutation probability	N/A	0.01
Crossover probability	N/A	0.70
Number of Alphas	10	N/A
Number of Betas	40	N/A

Table 1: Settings used in the experiments.

	Seeding ratio			
	25%	50%	75%	100%
Avg. generations to find a solution	342	237	143	156
Avg. generations be-fore solution did not improve further	388	312	429	277
Avg. steps used by solutions found	675.0	641.5	405.7	526.5
Steps in best solution found	609	545	303	405

Table 2: Results of the seeding experiments, with GH being used to seed GE on the Santa Fe Trail problem.

The hybrid algorithm was highly successful at finding a solution, with only 5 out of 400 unable to find a solution. This is a failure rate of 1.25% for the hybrid algorithm, as opposed to 11% for GH and 13% for GE, a significant improvement.

Table 2 displays further results from our initial study. The columns represent the results after four experimental sets of 100 runs using the various seed ratios as discussed in the previous section. The aim of this research was to see if a hybrid technique could improve the time taken by GE to find a solution, and to ascertain if GE could serve as a suitable optimisation step for a GH herd. The best solution found took 303 steps (Listing 1). By comparison, as discussed earlier, the best solution found by CGE (a GE variant) requires 337 steps that is currently the most effective solution reported in literature. This indicates that the new algorithm not only is able to improve the overall speed of grammatical evolution, but also provides more effective solutions with a lower failure rate.

Conclusions

Seeded Grammatical Evolution (sGE) is a novel automatic programming algorithm that builds on the GE paradigm, with the adoption of an initial non-random population generated through a swarm based method (GH).

The experimental results show that sGE produces results using less total generations than GE (but with the added overhead of the initial GH search), and produces more effective code than that produced purely through GH with a lower failure rate. Additionally, sGE produced one solution that was capable of solving the Santa Fe Trail in 303 steps, 34 steps fewer than what we believe to be the current state of the art result for this benchmark problem.

This study demonstrates that biological processes still provide a rich still potentially untapped source from which to provide inspiration for new techniques that can either lead to improvements on existing algorithms, or provide inspiration for new algorithms that are more effective than existing approaches.

Listing 1: Listing of the best solution generated by the hybrid algorithm that is capable of solving the Santa Fe Trail in 303 steps.

```
ifelse food-ahead
[ move ]
[ turn-left
ifelse food-ahead
  [ ifelse food-ahead
[ move move
ifelse food-ahead
[ move movemove ]
[ move ]
]
[ turn-left ]
]
[ ifelse food-ahead
[ move ]
[ turn-right ]
turn-right
ifelse food-ahead
[ move movemove ]
[ turn-left ]
    move
  ]
]
```

References

1. Ryan C, Collins JJ, O'Neill M (1998) Grammatical evolution: Evolving programs for an arbitrary language. Lecture Notes in Computer Science 1391: 83-96.
2. O'Neill M, Brabazon A (2006) Grammatical Swarm: The generation of programs by social programming. Natural Computing 443-462.
3. Headleand C, Teahan WJ (2013) Grammatical Herding. J Comput Sci Syst Biol 6:043-047.
4. Oltean M, Grosan C (2003) A comparison of several linear genetic programming techniques. Complex Systems 14: 285-314.
5. O'neill M, Ryan C, Keijzer M, Cattolico M (2003) Crossover in grammatical evolution. Genetic Programming and Evolvable Machines 4: 67-93.
6. Dawkins R (1986) The blind watchmaker. WW Norton & Company, USA.
7. Georgiou L, Teahan WJ (2011) Constituent grammatical evolution. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence: 1261 - 1268.
8. Reynolds C (1987) Flocks, herds and schools: A distributed behavioral model. Proceedings of the 14th annual conference on Computer graphics and interactive techniques. SIGGRAPH: 25-34.
9. Kennedy J, Eberhart R (1995) Particle swarm optimization. Neural Networks, 1995. Proceedings, IEEE International Conference on. IEEE 4: 1942-1948.
10. Georgiou L, Teahan WJ (2006) jGE - A Java implementation of Grammatical Evolution. Proceedings of the 10th WSEAS International Conference on SYSTEMS, Vouliagmeni. Athens: WSEAS: 406-411.
11. Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambrige Massachusetts, UK.
12. Mistral K (2005) The Secret Life of Stallions. Retrieved 2012, from Horse Connect.
13. Utah State University (2009) Wild Horse Behaviour. Retrieved 2012, from Animal, Dairy and Veterinary Sciences.