GEESE: Grammatical Evolution Algorithm for Evolution of Swarm Behaviors

Extended Abstract

Aadesh Neupane
Brigham Young University
Provo, Utah
aadeshnpn@byu.edu

Michael A. Goodrich
Brigham Young University
Provo, Utah
mike@cs.byu.edu

Eric G. Mercer
Brigham Young University
Provo, Utah
egm@cs.byu.edu

KEYWORDS

Swarms; Grammatical Evolution

ACM Reference Format:
Aadesh Neupane, Michael A. Goodrich, and Eric G. Mercer. 2018. GEESE: Grammatical Evolution Algorithm for Evolution of Swarm Behaviors. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 MOTIVATION AND PROBLEM STATEMENT

Simple organisms have evolved local interactions among individuals that produce useful collective behaviors: Bacteria interact with each other to move across cell surfaces efficiently by synthesizing a large number of flagella [1]; ant colonies have evolved collective behaviors for foraging, nest defense, path planning and construction [2]; bees are able to select best sites among many good sites at their disposal [8]; and fish are able to avoid predators by organizing themselves in collective shapes that deter predation [4].

Mimicking these collective behaviors in robots swarms would be beneficial for understanding social intelligence, collective cognition, and potential applications in engineering, artificial intelligence, and robotics. The problem of finding a set of individual behaviors to obtain a desired collective behavior is a hard problem. Conventionally, mimicking these behaviors with robots requires researchers to study actual behaviors, derive mathematical models, and implement these models as algorithms.

There are different ways to obtain the desired behaviors in swarms such as artificial neural networks, genetic programming based structures, logic-based symbolic controllers and behavior-based controllers. One approach to identifying individual behaviors that induce desirable collective behavior is to use Grammatical Evolution (GE), a type of Evolutionary Algorithm. GEs [6, 7] work by restricting the search space by "seeding" the solution space using domain-specific knowledge. Thus GEs seek to find solutions to the problems for which Genetic Programming (GP) [5] takes too long.

We propose a distributed algorithm, Grammatical Evolution algorithm for Evolution of Swarm bEhaviors (GEESE), which uses grammatical evolution to evolve (a) a primitive set of human-provided rules into (b) productive individual behaviors that (c) exhibit desirable collective behaviors.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

When multiple agents are distributed in different spatial regions, the search for high-quality solutions can be accelerated if all the agents start in a different spatial location and interact with each other, sharing their knowledge of the search space accumulated so far. Distributed evaluation of fitness and searching different parts of the spatial domain suggest that a multi-agent GE may generate effective collective behaviors in swarms. GEESE is implemented as a distributed algorithm.

2 APPROACH

Grammatical Evolution (GE) is a context-free grammar-based GP paradigm that is capable of evolving programs or rules in many languages [6, 7]. GE adopts a population of genotypes represented as binary strings, which are transformed into functional phenotype programs through a genotype-to-phenotype transformation. The transformation uses a BNF grammar, which specifies the language of the produced solutions. In GE, there is a central population of genomes where each genome is assigned a fitness or quality value. Only the portion of the population having higher fitness values are selected for genetic operations.

GEESE is similar to GE in terms of initialization, genetic operators, and genotype-to-phenotype mapping. GEESE starts with a fixed number of agents initialized with a random string of integers (genotype). In a mild abuse of notation, the genotype of an agent is also referred to as an agent in GEESE; i.e, each agent has its own individual genotype. Each agent is capable of performing three basic functions: *sense*, *act* and *update* in a given environment.

Sense. During the *sense step*, agent A_j uses input from its sensors to get information about the environment. If agent A_j senses other agents A_i nearby, denoted $\textsc{Neighborhood}(A_j)$, agent A_j requests each agent $A_i \in \textsc{Neighborhood}(A_j)$ to share its genotype G_i. The agent temporarily stores the genotypes of nearby agents in $\mathcal{M}_j$.

Act. During the *act step*, agent A_j checks its memory $\mathcal{M}_j$ where it stores all the genotypes received from agents in its neighborhood. If its memory $\mathcal{M}_j$ is empty, then it doesn't perform any action; otherwise, it performs a series of operations. First, it adds its own genotype to the memory. Second, a *selection* operator is performed on its memory to get parents. Selection samples from memory a subset of genotypes to be used to form a new population. Third, the *crossover* operator is applied to the parents to add children to the population; parents are discarded from the population after crossover. The set of children is mutated using a *mutation* operator,

fitness is evaluated for the set of children, and the mutated genotypes are added to the population. The highest performing child, G^*, is returned.

Update. During the *update step*, an agent checks whether the best genotype returned by the *act* step is superior to the current genotype. Agent A_j will replace genotype G_j with G^* if the fitness of G^* exceeds the fitness of G_j.

The advantage of GEESE over standard GE is that each agent is capable of applying genetic operators on its own. Using GEESE, each agent is able to compute GE onboard without centralized storage of genome population; i.e. GEESE computation is distributed and performed online. This enables each agent to search the evolutionary fitness landscape starting from a different location in the landscape. Instead of evaluating the whole population at each generation, GEESE evaluates locally, increasing the chances of average individuals to reach the next generation. This enables GEESE to maintain genetic diversity and slow down convergence.

3 EVALUATION

Santa Fe Trail. Santa Fe Trail [3] is toroidally connected grid structure where food is scattered in a predefined way. The objective of the Santa Fe Trail problem is to evolve a program that can navigate this trail, finding all the food. An agent can perform three moves: turn left, turn right, and move ahead.

For this experiment, 50 evolutionary runs were conducted for both conventional GE and for GEESE. Population size was set to 100, maximum generation to 50, mutation probability to 0.01 and crossover probability to 0.9 for the experiment.

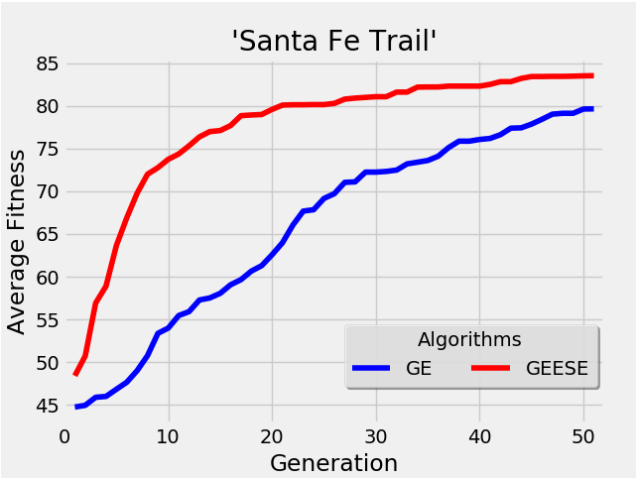

Figure 1: On average, GEESE converges quicker than GE.

Figure 1 demonstrates that GEESE converges to a solution faster than standard GE. The solid line is the mean fitness. GEESE required fewer generations with a smaller effective population size to solve the Santa Fe Trail problem in comparison with standard GE. The *hit rate* (percentage of trials that collect all food) for Standard GE with 50 runs is just 6% whereas the hit rate from GEESE is 57%. Also, one of the programs evolved by GEESE was able to complete the trail in 324 steps which is fewer than any other known solution [10]. This shows that GEESE outperforms standard GE in the Santa Fe Trail problem.

Foraging. GEESE was evaluated on a foraging scenario known as the *center place food foraging* problem [9]. The agent's task is to collect food from a *source* region in the environment and bring the food to a *hub* region in the environment. Initially, all the agents are located inside the hub. Agents carry the food units from source to hub where the food is stored. Agents do not have prior information regarding the source location. Agents carry as many food units back to the hub as possible during a fixed time frame.

GEESE requires a BNF grammar to evolve a solution. For the foraging problem, we design the grammar based on the primitive behaviors the agent can perform in the environment. The phenotype obtained from the grammar defines a rule. A rule is made up of three elements: *states*, *preconditions* and *transitions*. *States* are the low-level behavior, which is an activity that an agent can execute in the environment. *Preconditions* are the boolean values which are evaluated before transitioning to other states. *Transitions* refer to the probability of transitions between states. Thus, the phenotype encodes a state-machine.

We created a hand-coded benchmark for comparison. The benchmark consisted a set of 13 rules from the grammar. The hand-coded program was able to collect 77 units of food in an average of 284 timesteps. For both standard GE and GEESE, fifty evolutionary runs were executed with 100 agents, 0.9 crossover probability, 0.01 mutation probability, tournament selection, and generational-type replacement. On average, 56 units of food were collected by the agents using standard GE, which is fewer than the hand-coded benchmark. Moreover, the evolved programs lacked communication behaviors, even though the grammar was capable of expressing communication.

The GEESE evolved program was more efficient than the hand-coded program; one evolved program had only 8 rules in contrast to the 13 rules in the hand-coded program. The evolved program on average collected 83 units of food in 284 timesteps which is higher than the benchmark value. The evolved program contained communication behaviors. One of the benefits of using GE for the evolution of swarm behaviors is that evolved behaviors are expressed as a human-readable program.

4 SUMMARY

This paper presented the GEESE algorithm, a grammatical evolution algorithm for a multi-agent system. Results demonstrated the effectiveness of GEESE on the Santa Fe Trail problem, outperforming the state of the art in terms of minimum steps to solve the problem. Additionally, GEESE was used to evolve individual behaviors that lead to successful colony-level foraging, outperforming behaviors evolved by conventional grammatical evolution as well as hand-coded individual behaviors.

ACKNOWLEDGMENTS

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. The work is supported by the ONR grant number N000141613025.

REFERENCES

[1] Matthew F Copeland and Douglas B Weibel. 2009. Bacterial swarming: a model system for studying dynamic self-assembly. *Soft matter* 5, 6 (2009), 1174–1187.

[2] Deborah M Gordon. 1999. *Ants at work: how an insect society is organized*. Simon and Schuster.

[3] David Jefferson, R Collins, C Cooper, M Dyer, M Flowers, R Korf, C Taylor, and A Wang. 1992. *The genesys/tracker system*. Reading, MA: Addison-Wesley.

[4] Yael Katz, Kolbjørn Tunstrøm, Christos C Ioannou, Cristián Huepe, and Iain D Couzin. 2011. Inferring the structure and dynamics of interactions in schooling fish. *Proc. of the National Academy of Sciences* 108, 46 (2011), 18720–18725.

[5] Spyros A Kazarlis, AG Bakirtzis, and Vassilios Petridis. 1996. A genetic algorithm solution to the unit commitment problem. *IEEE trans on power systems* 11, 1 (1996), 83–92.

[6] Michael OâĂŹNeil and Conor Ryan. 2003. Grammatical evolution. In *Grammatical Evolution*. Springer, 33–47.

[7] Conor Ryan, JJ Collins, and Michael O Neill. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *European Conf. on Genetic Programming*. Springer, 83–96.

[8] Thomas D Seeley and Susannah C Buhrman. 2001. Nest-site selection in honey bees: how well do swarms implement the" best-of-N" decision rule? *Behavioral Ecology and Sociobiology* 49, 5 (2001), 416–427.

[9] John H Franks Sudd, Nigel R John H Sudd, and Nigel R Franks. 1987. *The behavioural ecology of ants*. Technical Report.

[10] Paulo Urbano and Loukas Georgiou. 2013. Improving Grammatical Evolution in Santa Fe Trail using Novelty Search.. In *ECAL*. 917–924.