Automatic Synthesis of Swarm Behavioural Rules from their Atomic Components

Dilini Samarasinghe
The University of New South Wales
Canberra, Australia
d.samarasinghewidanaarachchige@student.adfa.edu.au

Erandi Lakshika
The University of New South Wales
Canberra, Australia
e.henekankanamge@adfa.edu.au

Michael Barlow
The University of New South Wales
Canberra, Australia
m.barlow@adfa.edu.au

Kathryn Kasmarik
The University of New South Wales
Canberra, Australia
k.kasmarik@adfa.edu.au

ABSTRACT

This paper presents an evolutionary computing based approach to automatically synthesise swarm behavioural rules from their atomic components, thus making a step forward in trying to mitigate human bias from the rule generation process, and leverage the full potential of swarm systems in the real world by modelling more complex behaviours. We identify four components that make-up the structure of a rule: control structures, parameters, logical/relational connectives and preliminary actions, which form the rule space for the proposed approach. A boids simulation system is employed to evaluate the approach with grammatical evolution and genetic programming techniques using the rule space determined. While statistical analysis of the results demonstrates that both methods successfully evolve desired complex behaviours from their atomic components, the grammatical evolution model shows more potential in generating complex behaviours in a modularised approach. Furthermore, an analysis of the structure of the evolved rules implies that the genetic programming approach only derives non-reusable rules composed of a group of actions that is combined to result in emergent behaviour. In contrast, the grammatical evolution approach synthesises sound and stable behavioural rules which can be extracted and reused, hence making it applicable in complex application domains where manual design is infeasible.

CCS CONCEPTS

• **Computing methodologies → Multi-agent systems**; **Genetic programming**; *Control methods*; *Artificial life*;

KEYWORDS

Multi-agent Systems, Grammatical Evolution, Genetic Programming, Swarm Intelligence, Artificial Life

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205546

ACM Reference Format:
Dilini Samarasinghe, Erandi Lakshika, Michael Barlow, and Kathryn Kasmarik. 2018. Automatic Synthesis of Swarm Behavioural Rules from their Atomic Components. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3205455.3205546

1 INTRODUCTION

Multi-agent models for swarms adopt a bottom-up approach in modelling virtual behavioural dynamics. The individual agents are codified with simple rules so that they act according to their own discrete perceptions without any centralized control, yet the local interactions among them give rise to emergent fluid motions at the group level. The self-organizing agents whose micro level interactions help explain the macro structures and emergent behaviours of a system make these models more effective in simulating composite dynamics and in adapting to changes of the environment [21]. However, simulating such collective swarm and team behavioural interactions of the real world in artificial environments is increasingly becoming challenging due to numerous application requirements that desire more complex and life-like behaviours. Manual design of such behaviours would require an understanding of the high level macro behaviours in a way to decompose them into micro-level behaviours which can be adopted by individual agents of the system. Given a pre-specified task description, designing aggregate sets of behavioural rules is difficult with a hand crafting approach as mere intuition is insufficient to get an insight when the complexity increases. Such limitations of human bias have made a significant impact in leveraging the full potential of swarm systems in real world domains.

A better alternative for domains where human reasoning and capabilities become limited in comprehending and solving complex problems is to explore automatic synthesis mechanisms of rules. The existing mechanisms, often involving evolutionary approaches, either focus on evolving parameters related to pre-designed rules or on finding the best subset from a pool of hand generated behaviours to result in required emergent behaviours. The need for rigorous manual tuning of parameters and/or the design of a pool of primary behaviours suggest that human intervention during the synthesis process is still significant. As a result, the levels of complexities that can be reached with the mechanisms are still limited. As opposed to the existing approaches, focusing on the intrinsic logic which defines the behavioural rule structure and generating rules with the atomic components that form the said structure (Section 3 describes the atomic components we identified as control structures, parameters, preliminary actions and logical/relational connectives) would help mitigate human bias to a greater extent.

The work presented in this paper is aimed at exploring whether evolutionary approaches could be successfully employed in automatically generating rules for emergent behaviours from pools of their atomic components. The method tries to limit the human bias in the process and address the requirement of more complex behavioural outputs. We propose a grammatical evolution (GE) based approach where the syntax of the behavioural rules is maintained by a grammar represented in Backus-Naur form [14] designed based on the identified atomic components. The same rule space is then employed in a genetic programming (GP) based environment to define the tree structure of the evolving programs. The two models are evaluated with a boids behaviour simulation system where the collective motion of the group of agents is automatically controlled based on the evolving behavioural rules. The key contributions of the paper are as follows:

(1) Introduction of an evolutionary computing based automatic synthesis mechanism for multi-agent behavioural rules where the entire rule structure can be evolved from their atomic components based on a valid syntax unlike the existing mechanisms where the rule structure is pre-defined.
(2) Evaluation of the proposed mechanism with GE and GP based models for evolving simple and complex behavioural rules.
(3) An initial exploration of adopting a modularised approach to generate more complex behavioural rules starting from evolved groups of simple behaviours using the proposed mechanism.
(4) An analysis of the rule structures generated with GE and GP models discussing the possibility of reverse engineering them.

The rest of the paper is organized as follows. Section 2 summarises the existing literature on modelling swarm behavioural rules and their limitations. Section 3 describes the overview of the proposed grammatical evolution based approach and the genetic programming approach. The experimental setups for the evaluations are presented in Section 4 and, in Section 5 a detailed discussion of results is carried out. Finally, Section 6 concludes the paper with possible future research directions.

2 RELATED WORK

Multi-agent Behavioural Rules

The seminal work on multi-agent behavioural rule modelling was presented by Reynolds [18] where a simulation model was developed with simple rules leading to aggregate motions of a flock of birds. The model adopts a simple weighted linear combination of three steering behaviours [20]: Alignment, Cohesion and Avoidance which correspond to steering towards average heading of neighbour boids, moving towards neighbour boids and forming a group and, maintaining a distance from neighbours avoiding collisions, respectively. They are applied on each individual boid separately based on their local perception of neighbouring flock mates.

This model unveiled new dimensions in computer animations by replacing the traditional approach of scripting individual paths of agents, with a distributed behavioural model where agents follow their own course of actions based on a pre-defined set of rules. Since the seminal model, similar multi-agent systems have repeatedly been discussed in the literature [11, 12, 22]. Although the said behavioural models are capable of defining the characteristics of expected tasks and behavioural patterns, the rules are essentially hand-crafted based on human perception of natural swarms and teams. Thus, these approaches are subjected to human bias and more critically, are prone to fail in situations where the requirement is to model numerous and complex interactions.

Automatic Synthesis of Rules

Automatic design methods have often been explored as a more desirable substitute for manual design of behavioural rules and can be broadly discussed under two categories; reinforcement learning and evolutionary frameworks. Reinforcement Learning (RL) based approaches are commonly explored in the domains where reinforcement information, expressed as rewards and penalties, can be provided for the actions [7]. The agents in the system learn behaviours through trial-and-error interactions with a dynamic environment by maximizing a cumulative reward [1, 6]. However, automation of swarm behaviours has achieved limited success with this approach [15] due to the limitations in decomposition of global reward at the emergent level into individual rewards for agents. Hence, they fall behind when multi-agent systems pursuing more complex behavioural tasks are at hand.

Evolutionary Computing [3], inspired by natural evolution is more capable than RL to automatically evolve individual behaviours in multi-agent contexts. Genetic algorithms [5], which are one of the most popular types of evolutionary algorithms, have been a common choice in this field in trying to model multi-agent behavioural patterns. However, these algorithms have no control over the structure of the individuals that are being evolved. Hence, despite the fact that these efforts have tried limiting the human involvement in rule generation, they have concentrated primarily on automatic evolution of the parameters necessary for formulation of behavioural rules rather than exploring the capability of automatic generation and evolution of rules themselves. The agent rules are either manually designed [2, 10] or they are represented by an artificial neural network (ANN) [23]. Thus, the rules have to be decided with human involvement or using an ANN that hinders reverse engineering, analysis and combination of individual rules for complex behaviour generation [4].

Genetic programming [8] is adopted as a better alternative, since it evolves solution spaces consisting of computer programs that naturally embody a tree structure which is also the natural representation of behavioural rules. The solution space is confined to a primitive set of functions and terminals and the function set is required to adhere to closure property [8]. These requirements limit the control over the structure of the programmes being evolved. Nevertheless, the nature of the rules composed of functions and terminals have encouraged such approaches to be tested in evolving behaviours in several work [9, 19, 24]. Due to the limited control provided by GP over the rule structure, most of these approaches intrinsically focus on finding a group of simple behaviours that can result in a specified emergent behaviour rather than concentrating on evolving the emergent behavioural rules from their structure.

Being closely related to GP, grammatical evolution [14] also evolves programs abiding a tree structure, but as opposed to GP, a grammar controls the nature that the solution space is being explored and restricts the programs to a particular syntax through a context-free grammar (CFG). Behaviour trees for Mario AI benchmark are evolved using GE in [16], and horse gait optimizations are explored in [13] where motion data from a walking horse is optimized for a horse model. The work presented in [4] employs a grammar based method in a multi-agent system evolving a food foraging task. However, the approach does not incorporate the atomic components of rules, but rather combines a set of preconditions, low-level behaviours and actions, all hand crafted to evolve simple rules. We see our work as a further enhancement to the currently explored GP and GE techniques for swarm behavioural rule generation and a step forward towards eliminating human bias in the synthesis process by decomposing the rule structure to its atomic components to evolve behaviours. The next section presents details of the proposed evolutionary approach.

3 EVOLUTIONARY MODEL FOR AUTOMATIC SYNTHESIS OF BEHAVIOURAL RULES

Building a Rule Space from Atomic Components

The proposed approach is based on deriving emergent behaviours from combining the basic components of the rules. A close examination on the behavioural rule structures shows that they ideally assume a similar pattern in implementation and consist of a logic formulated by 4 components: control structures, parameters, preliminary actions and logical or relational connectives. For example, consider the following rule in a boids system:

[if] [distance to boid] [<] [2.5] [move away from boid]

The rule can be broken down into several components including a control structure: *If*, parameters: *distance to boid* and the value *2.5*, a relational connective: < and a preliminary action: *move away from boid*. A similar structure is observed in aggregate rule sets with a logical connective such as *and*,*or* being used to combine several rules together. With this understanding, a rule space was constructed from a pool of these 4 types of components for a boids system. As all rules are primarily of if-else format, *if* was used as the construct and the *else* component is built into the structure. A commonly used set of logical and relational connectives were chosen from the general pool of mathematical operators. The parameters and preliminary actions were chosen by examining the dynamics of natural flocks and their hand crafted implementations to test the feasibility of the approach, and a richer component set is intended to be used with future experiments.

3.1 Grammatical Evolution Model

A grammar is established based on the rule space and the general syntax of the behavioural rules. For the proposed work, a CFG which is represented in Backus-Naur form is used. Figure 1 depicts the production rules of the grammar (G), formulated based on the rule space, and defined by the tuple { ν, τ, ρ, ς } where ν is the set of non-terminals, τ the set of terminals, ρ is the set of production

$$
\begin{array}{rcl}
\langle \text{S} \rangle & \models & \langle \text{angle of vision} \rangle\ \langle \text{distance of vision} \rangle\ \langle \text{S1} \rangle \\
\langle \text{S1} \rangle & \models & \langle \text{S1} \rangle\ \langle \text{S1} \rangle \ \mid\ \langle \text{I} \rangle^{*} \langle \text{W} \rangle \\
\langle \text{I} \rangle & \models & \text{if } \langle \text{O} \rangle\ \langle \text{I} \rangle\ \langle \text{I} \rangle \\
 & & \mid\ \text{if } \langle \text{O} \rangle\ \langle \text{I} \rangle\ \langle \text{A} \rangle \\
 & & \mid\ \text{if } \langle \text{O} \rangle\ \langle \text{A} \rangle\ \langle \text{I} \rangle \\
 & & \mid\ \text{if } \langle \text{O} \rangle\ \langle \text{A} \rangle\ \langle \text{A} \rangle \\
\langle \text{O} \rangle & \models & \text{and } \langle \text{O} \rangle\ \langle \text{O} \rangle \ \mid\ \text{or } \langle \text{O} \rangle\ \langle \text{O} \rangle \\
 & & \mid\ \text{LTE } \langle \text{P} \rangle\ \langle \text{distance} \rangle \\
 & & \mid\ \text{between } \langle \text{P} \rangle\ \langle \text{distance} \rangle\ \langle \text{distance} \rangle \\
\langle \text{P} \rangle & \models & \text{separation distance} \\
 & & \mid\ \text{distance to flockcentre} \\
\langle \text{A} \rangle & \models & \text{move away from boid} \ \mid\ \text{move forward} \\
 & & \mid\ \text{move away from flockcentre} \\
 & & \mid\ \text{move towards flockcentre} \\
 & & \mid\ \text{move towards boid} \ \mid\ \text{turn by } \langle \text{angle} \rangle \\
 & & \mid\ \text{match velocity with boid} \\
\langle \text{distance} \rangle & \models & \text{random distance} \\
\langle \text{distance of vision} \rangle & \models & \text{random distance} \\
\langle \text{angle} \rangle & \models & \text{random angle} \\
\langle \text{angle of vision} \rangle & \models & \text{random angle} \\
\langle \text{W} \rangle & \models & \text{random weight}
\end{array}
$$

Figure 1: Production Rules (ρ) of the Grammar (G).

rules and ς represents the start symbol which in this case is ⟨S⟩. The components on the left of each production are the non-terminals, which can be replaced with a combination of terminals and non-terminals as defined in the production. The rest of the components are the terminals which cannot be further replaced.

The syntax of the rules is defined as follows: The production rule of ⟨S⟩ states that, for each individual rule, a ⟨distance of vision⟩ and an ⟨angle of vision⟩ is specified which will determine the vision range of boids in the flock. The parameters are initially replaced by random values which will be evolved over the generations. The syntax of ⟨S1⟩ is designed such that an individual rule can consist of a single simple rule or an aggregate set of simple rules which are combined based on a weight assigned to each simple rule. The non-terminal ⟨W⟩ represents this weight which is randomly decided for the initial generation. Each simple rule ⟨I⟩ follows the syntax [*if*] [*condition*] [*then-do*] [*else-do*]. The condition ⟨O⟩ can be one of the relational connectives *between* or *LTE* (less than or equal to), or several relational connectives combined with logical connectives *and* or *or*. The *then* and *else* actions could either be another *if* condition generating nested rules within one rule or a preliminary action ⟨A⟩. The relational connective *LTE* evaluates whether its first argument is less than or equal to its second argument. Similarly, *between* evaluates whether its first argument is within its second and third arguments. The first arguments of the two connectives are one of the two parameters ⟨P⟩, *separation distance* or *distance to flockcentre*, and the other arguments are random distance values. The preliminary actions are as defined under the non-terminal ⟨A⟩ and the action ⟨turn by⟩ accepts an argument which is the angle it should turn. All random angle and distance values (⟨distance⟩ , ⟨angle⟩) generated in the initial generation are then evolved to find the best parameter values suitable.

Grammatical evolution initiates with a genotype encoded in binary strings, known as Codons, that can be mapped to a syntactically correct phenotype of valid programs based on the designed grammar. It is then evolved using a genetic algorithm, based on an appropriate fitness criteria with crossover and mutation operations to automatically generate behavioural rules. Algorithm 1 illustrates the process of grammatical evolution. The mapping of the genotype to the phenotype is done by generating the corresponding integer string for the binary values and applying the following mapping function on each integer value:

(Integer Value) MOD (no. of production rules for the current non-terminal)

The mapping process can be initiated by considering the number of production rules for the start symbol ⟨S⟩ and a decision is made as to which rule component will replace it. The process continues until eventually an expression consisting entirely of terminals is reached. As can be seen, this approach provides more control over the structure of the rule that is being evolved and eliminates the limitations caused by closure property as crossover and mutation operations are performed on the binary string rather than on the tree structure unlike genetic programming. Thus, it is adopted in the proposed approach to further mitigate human intervention in generation of behaviour rules for multi-agent systems.

Algorithm 1 Grammatical Evolution

Input: ρ: CFG production rules set
β: Population size
Ω: Maximum generations
Output: I_b: Individual rule with the best fitness

procedure GrammaticalEvolution(ρ, β, Ω)
  $pop \leftarrow$ InitializePopulation(β, ρ)
  EvaluateFitness(pop)
  $I_b \leftarrow$ GetMostFitIndividual(pop)
  $\omega \leftarrow 0$
  while $\omega \neq \Omega$ **do**
    $valid \leftarrow false$
    while $valid == false$ **do**
      $parents \leftarrow$ ParentSelection(pop)
      $child_{M1}, child_{M2} \leftarrow null$
      for $parent1, parent2 \in parents$ **do**
        $children \leftarrow$ Crossover($parent1, parent2, prob_{cross}$)
        for $child1, child2 \in children$ **do**
          $child_{M1} \leftarrow$ Mutate($child1, prob_{mut}$)
          $child_{M2} \leftarrow$ Mutate($child2, prob_{mut}$)
      if MapWithCFG($\rho, child_{M1}$) $==$ $true$ AND MapWithCFG($\rho, child_{M2}$) $== true$ **then**
        $valid \leftarrow true$
        $I_W \leftarrow$ GetTwoWorstFitIndividuals(pop)
        for $I_{W1}, I_{W2} \in I_W$ **do**
          $I_{W1} \leftarrow$ ReplaceIndividual($I_{W1}, child_{M1}$)
          $I_{W2} \leftarrow$ ReplaceIndividual($I_{W2}, child_{M2}$)
    EvaluateFitness(pop)
    $I_b \leftarrow$ GetMostFitIndividual(pop)
    $\omega{+}{+}$
  return I_b

3.2 Genetic Programming Model

GP employs individuals embodying a tree structure with functions and terminals. To cater to the need, the same rule space used for GE was considered in defining the programs evolved in this context for a fair evaluation of the two approaches. All parameters except for *distance of vision* and *angle of vision* and all primitive actions except *turn by* were pooled as terminals in the GP solution, as they represent the inputs to the computer programs being evolved, and actions enabling the boids to make movements resulting in emergent behaviour respectively. They take no arguments and hence are better categorised as terminals. The two parameters *distance of vision* and *angle of vision* were eliminated from the rule space as they are essential components for all individuals and cannot be made part of the evolutionary process where a random combination might or might not select them as part of the program tree. Instead, experimentally determined values, *PI* as the angle and *100* as the distance were hand coded into the programs in order to define the vision range of boids. A preliminary sensitivity analysis conducted with different combinations of parameter values showed variation in performance but did not exceed that of the GE approach. This is one limitation of GP as rigorous manual tuning is required to determine the parameter values in contrast to the GE approach where all parameters and structural details can be conveniently handed over to the algorithm to automatically generate.

All the control structures, connectives and the action *turn by* from the rule space were categorised as functions in this approach since these accept arguments and are better suited for the inner nodes of the program trees. Each of the functions ⟨if⟩, ⟨between⟩, ⟨LTE⟩, ⟨and⟩, ⟨or⟩, ⟨turn by⟩ accept 3, 3, 2, 2, 2, and 1 argument(s) respectively.

Thus, the function (F) and terminal (T) sets for the rule space can be represented as follows:

F = {⟨if⟩, ⟨between⟩, ⟨LTE⟩, ⟨and⟩, ⟨or⟩, ⟨turn by⟩}

T = {separation distance, distance to flockcentre, random angle, random distance, move away from boid, move towards boid, move away from flockcentre, move towards flockcentre, move forwards, match velocity with boid}

As GP performs the genetic operations such as crossover and mutation on the initially determined function and terminal sets in generating program trees, all functions should be well defined for arguments consisting of any combination of functions and terminals from the primitive set. Hence in the above case, every primitive component returns a numerical value in order to preserve this closure property required by the algorithm by allowing combination of subtrees during the evolution process. *Separation distance* and *distance to flockcentre* return those values calculated at the given time for a specific boid. *random angle* and *random distance* generate random values in specific ranges at their first encounter and these values will be preserved over the generations of evolution. The actions *move away from boid, move towards boid, move away from flockcentre, move towards flockcentre, move forwards* and *match velocity with boid* return the numerical values 1, 2, 3, 4, 5, and 6 respectively. Of the primitive function set, ⟨if⟩ evaluates its first argument and if it is not 0 then it returns the numerical value of the second argument. Otherwise it returns the numerical value of argument three. Similarly, ⟨between⟩ evaluates its first argument and if the value returned is between the second and third argument values it returns 1, else returns 0. ⟨LTE⟩, ⟨and⟩, ⟨or⟩ functions return 1 or 0 based on the evaluation of their two arguments. ⟨turn by⟩ simply returns the numerical value of its only argument.

4 EXPERIMENTAL SETUP

4.1 Evolutionary Setup

The evolutionary algorithms for GE and GP approaches were implemented adopting the steady state replacement (SSR) mechanism, replacing only the two worst fit solutions from the previous population with offspring generated from crossover and mutation operations. An initial population of valid 30 individuals generated randomly was evolved for 100 generations. As for the parent selection operator, tournament selection with a tournament size of 5 was used. SSR was conducted with one point crossover for the two worst fit individuals with single point mutation with probability 0.005.

For the GE model, an individual was constructed with 25 codons of size 8 bits. In order to allow for generation of sufficiently complex rules, a maximum wrapping value of 50 was introduced which was determined experimentally. I.e. if the individual runs out of codons before reaching a valid expression during the mapping process, it is wrapped and the codons are reused. If the individual does not map to an expression of all terminals by the end of 50 wraps, it is deemed invalid.

4.2 Simulation Environment

A boids behaviour simulation environment was adapted for conducting the experiments. The autonomous virtual agents (boids) were modelled with a hybrid architecture consisting of both reactive and deliberative agent properties. I.e. the boids are capable of interacting with the environment and reacting based on the environmental changes and at the same time are driven to achieve a common goal defined by the fitness measure. The interactions were implemented focusing on a single perception which is vision. Each boid has a sense of their neighbourhood based on a specified vision range and adjusts their behaviour through evolution based on the neighbourhood and the evolved rules.

The simulations were conducted with 150 boids in a wrap-around environment for 15 evolutionary runs each as presented in Section 5, with known seed values for the random number generator, ensuring that the experiments can be repeated.

4.3 Fitness Measure

For the initial experiments on evolving micro behaviours, it was investigated whether the model is capable of evolving Reynolds' three rules for flocking: alignment, cohesion and avoidance which were hand crafted in his approach to generate realistic emergent behaviour. Quantitative measures were used in evaluating the fitness of each of the 3 micro behaviours. The order measure, introduced by Vicsek [22] was used in evaluating the alignment behaviour. Equation 1 depicts the order measure (V_{avg}) which is the absolute value of the average of normalized velocities of the boids. Average separation distance between boids was used as the fitness measure in quantifying cohesion behaviour. The separation distance function s_i to calculate average separation distance for a single boid from other boids is given in Equation 2 . Average of s_i (S_{avg}) was considered as the cohesion measure. The quantitative measure for avoidance was adopted from the work of [17] applying average separation distance among boids in the function. Equation 3 depicts the function D_i with experimentally determined parameter values used (δ = 100, γ = 0.99, μ =1000). For penalizing flocks with collisions, if s_i was less than or equal to 500 units it was made equal to μ. Average of D_i (D_{avg}) was taken as the avoidance measure of the flock.

$$V_{avg} = \frac{-1}{\eta} | \sum_{i=1}^{\eta} v_i | \tag{1}$$

$$s_i = \frac{1}{\eta - 1} \sum_{j=1}^{\eta} distance(d_i - d_j) \qquad where j \neq i \tag{2}$$

$$D_i = -1 + \frac{1}{1 + \exp^{-\delta(s_i - \gamma\mu)/\mu}} \tag{3}$$

η - total number of boids
v_i - normalized velocity of boid i
s_i - average separation distance for boid i from other boids

For the experiments of the second phase involving complex behaviours, flocking was chosen to be tested as it is an emergent behaviour that can only be generated by an appropriate combination of several simple behaviours. As for the fitness evaluation measure, a simple equi-weighted combination of the previous 3 fitness measures for micro behaviours was utilized as given in equation 4. All four fitness measures are minimising functions and range from values 0-1.

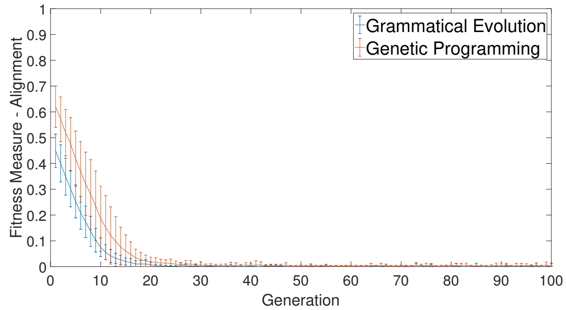
$$FlockingMeasure = \frac{1}{3}V_{avg} + \frac{1}{3}S_{avg} + \frac{1}{3}D_{avg} \tag{4}$$

5 RESULTS

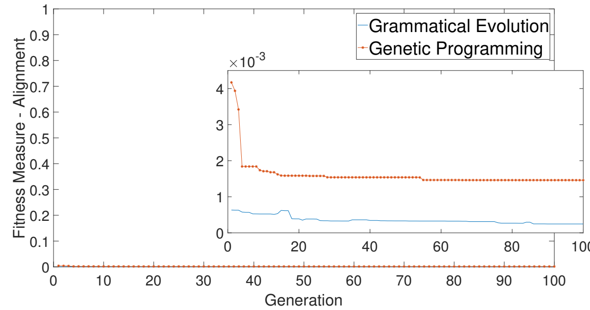
5.1 Evolution of Behaviours from Scratch

Figure 2 illustrates the results of the experiments for evolving the 3 micro behaviours alignment, cohesion and avoidance. Both GE and GP experiments were repeated for 15 evolutionary runs each and the average fitness progression of the population over generations and the progression of the most fit individual are presented.

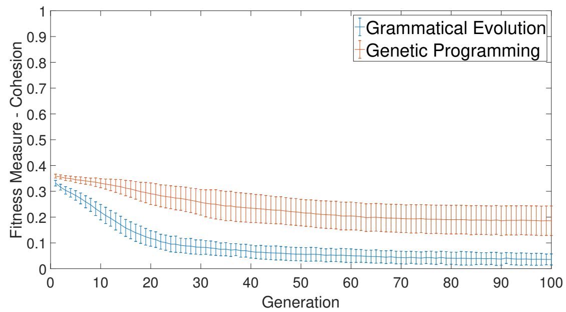
The results demonstrate that both approaches are successful in evolving better behaviours through fitness minimisation. At the start of the evolution, the average fitness values of both approaches remain closer to each other and as the evolution progresses, GE demonstrates a slightly better drop in the fitness compared to GP. In order to statistically determine the significance of the difference between the two models, we adopted Mann-Whitney U test as the sample sizes are small and are not normally distributed. At 99% confidence level the p-values obtained for the most fit values of all 3 samples, alignment, cohesion and avoidance were less than 0.001 proving that GE approach is better than the GP approach in general. Nevertheless, both approaches start with significantly better individuals in the population for the alignment behaviour and experience only a slight drop in fitness for the best individual throughout the generations. The existence of the action *match velocity with boid* could be the main cause as adjusting velocities to match other flock members can easily generate aligned



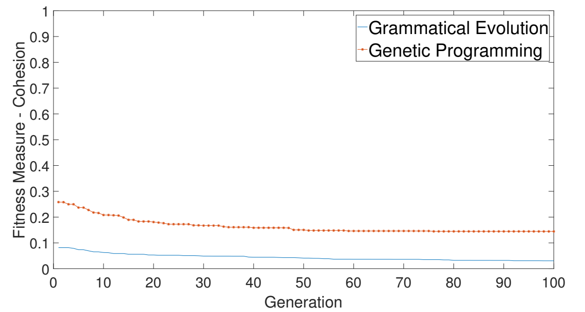
(a) Evolution of the average fitness of the population : Alignment.



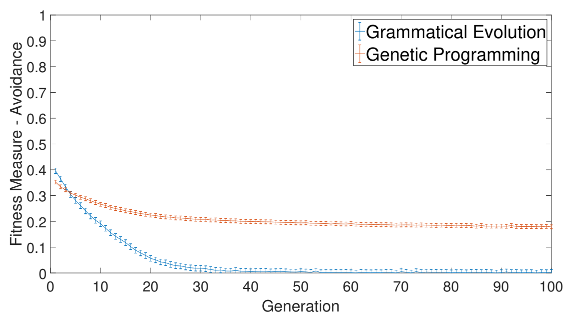
(b) Evolution of the most fit solution : Alignment.



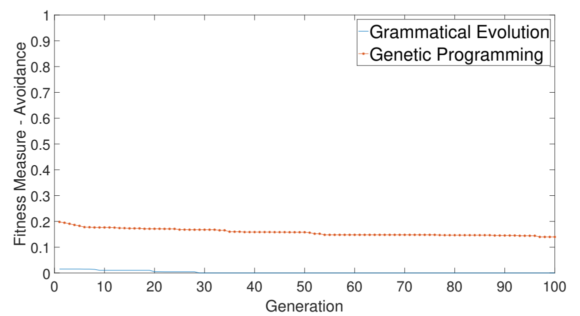
(c) Evolution of the average fitness of the population : Cohesion.



(d) Evolution of the most fit solution : Cohesion.



(e) Evolution of the average fitness of the population : Avoidance.



(f) Evolution of the most fit solution : Avoidance.

Figure 2: Evolution results for micro behaivours. Figures 2a, 2c, 2e represent the average fitness progression of the population with error bars representing standard deviation from 15 different experiments for alignment, cohesion and avoidance behaviours. Figures 2b, 2d, 2f represent the fitness progression of the most fit solution for the 3 behaviours.

flocks. Other two behaviours are not as straightforward as alignment and hence take more generations to evolve better rules. The next set of experiments evolve much complex behaviours and a similar observation is not evident due to the complexity of the task.

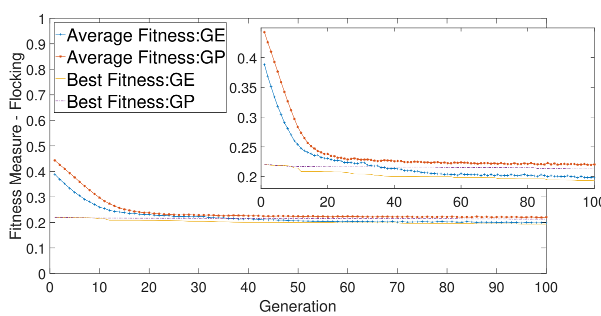
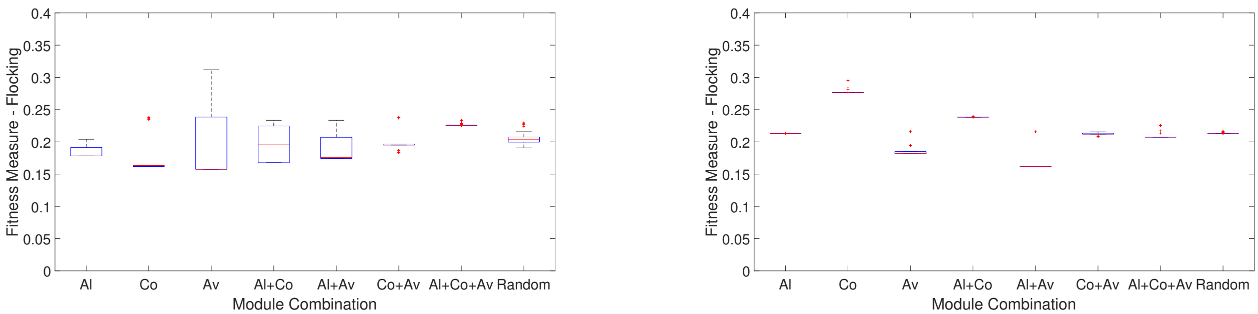


Figure 3: Evolution results for flocking behaviour. The average fitness progression and the progression of the most fit solution for both GE and GP models are shown in the graph.

Figure 3 illustrates the evolution results averaged over 15 runs for evolving flocking behaviour which is more complicated to derive. Both models were still capable of successfully generating the complex flocking behaviour, however the p-value for the results with quantitative measure was 0.014 (> 0.05) which suggests that there is not enough evidence to state a significant higher performance for GE approach unlike the previous case of simple behaviours.

5.2 Modularised Approach of Evolution

The next phase of the experimental analysis was to evolve flocking behaviour from individually evolved micro-behavioural rules. The experiment was conducted as a two-step process, initially evolving random populations for alignment, cohesion and avoidance, and then combining the evolved solution sets to form the initial population for the experiment on flocking. 7 experiments were conducted



(a) Evolution of flocking from modules with GE.

(b) Evolution of flocking from modules with GP.

Figure 4: Fitness variability of the most fit solution for GE and GP models over 100 generations with different module combinations. Results of 7 experiments each with initial populations of individually evolved micro-behavioural rules for each of alignment (Al), cohesion (Co), avoidance (Av) and all possible combinations, are compared with that of the average of 15 experiments with random initial populations.

with different initial populations; the first 3 experiments used populations of 30 which consisted entirely of individual sets evolved for one of the 3 micro behaviours. For the next 3 experiments, two sets from previously evolved populations were combined equally to take 15 individuals from each. The final experiment combined 10 from each of the 3 sets to form the initial population of 30.

Figure 4 compares the variability of the fitness of the best solution over 100 generations for the above 7 experiments with that of evolution results from a random population (results averaged over 15 runs) discussed before, for both GE and GP models. From the results in figure 4a of the GE approach it is observed that except for the evolution with initial rules from all 3 modules, all other experiments outperformed the evolution results with the random module. Also, from the results of the modules avoidance, alignment+cohesion, and alignment+avoidance, it is evident that GE approach is capable of exploring a large solution space reaching better fitness values even starting from weaker solutions at the initial generation. Such an observation cannot be made with the GP approach in figure 4b, and the variability is very narrow for all experiments. Also, 3 out of 7 experiments performed weaker than the random module while two more performed better only marginally. The p-value of the most fit solution for the average of 7 experiments of each GE and GP models was less than 0.001 indicating that GE approach performs significantly better than GP approach with the modularised approach unlike in the case of directly evolving from a random population. On the other hand, GP has less consideration on modules during evolution. The insights gained through this experiment could be useful in future experiments with GE for evolving much complex behaviours that cannot be easily generated from a random population.

5.3 Analysis of Evolved Rule Structures

Figure 5 illustrates two indicative rules evolved for flocking from random initial populations by the GE and GP models. A majority of the evolved rules are complex and consist of a larger number of nodes in the tree than the presented two rules. We selected the presented rules based on their convenience of presentation as the analysis is not affected by the rule length. The GE rule is essentially an aggregate of two single rule vectors combined on weights 0.75 and 0.25 for each respectively. The angle and distance of vision were decided as 3.38 radians and 253 units. First rule component evaluates whether the separation distance from a neighbour boid is less than or equal to a value of 249 and if it is greater it moves towards that boid trying to form a group and if not it evaluates another *if* condition nested into the second argument of the first condition. The second *if* condition determines whether the distance to the flock centre is less than or equal to a value of 25 and whether the separation distance is between 34 and 91. If both are true the boid moves away from the flockcentre, else it keeps moving forward. The second rule component which was given a less weight, evaluates whether the distance to the flock centre is less than a value of 451 and matches the velocity of the boid to the neighbour’s velocity. Otherwise it moves towards the flockcentre. Simply put, the rule tries to form a group by moving towards the neighbours while avoiding collisions by moving away from the flockcentre if they are too close. At the same time, it tries to align with the neighbours if it is in a group and tries to be involved in the flock by moving towards the flockcentre if it is at a larger distance away from the centre. A manual design approach of a rule for flocking may not have foreseen such details and certainly would consume more time and resources in tuning the parameters to the appropriate values due to the rigorousness of the task. The rule can be extracted from the evolutionary environment and used in non-evolutionary contexts with similar world designs as the evolved result is sound and stable.

On the other hand, the GP rule cannot be interpreted as the GE rule and the structure unlike the GE rule, does not reveal any valuable understandings on the behaviour rule in a way that can be analysed and reused in another context. Although the atomic components of the structure is utilized in the evolution process, the results evolved are essentially a group of actions that chose to behave appropriately based on the current parameter values. The combination of the actions, *turn by*, *match velocity with boid*, *move towards boid* and *move forward* have been able to generate acceptable flocking behaviour, but the rule structure does not provide any information that can be reverse engineered to understand or enhance the rules at a later stage. The numerical values returned by each of the terminals play a key role in maintaining the structure and facilitating the closure property, while in reality the evolved rule is not useful in a non-evolutionary environment. We identify this to be the major drawback of GP approach in contrast to GE where rules can be reused, analysed and modified based on the requirements. Particularly, with complex requirements where human

```
[Angle_of_vision=3.38 Distance_of_vision=253
 [Rule1*0.75
   [<If> [<LTE> Separation_distance 249]
     [<If> [<and> [<LTE> Distance_to_flockcentre 25]
                  [<Between> Separation_distance 34 91]]
             Move_away_from_flockcentre
             Move_forward]
     Move_towards_boid]]
 [Rule2*0.25
   [<If> [<LTE> Distance_to_flockcentre 451]
         Match_velocity_with_boid
         Move_towards_flockcentre]]]
```

(a) An indicative rule evolved with GE.

```
[<LTE> Distance_to_flockcentre
    [<Between> Separation_distance
          [<And> [<Turn_by> Match_velocity_with_boid]
                 Move forward]
          [<If> Move_towards_boid
                 [<And> Separation_distance 2.8]
                 [<Turn_by> Move_forward]]]]
```

(b) An indicative rule evolved with GP.

Figure 5: Analysis of rule structures generated by the GE and GP models.

comprehension of the task is limited, the proposed GE approach can be quite useful in gaining insight into the problem domain.

6 CONCLUSION AND FUTURE WORK

An automatic synthesis approach for swarm behavioural rules from their atomic components is introduced in this paper. The proposed approach adopts a rule space designed from a pool of control structures, parameters, logical and relational connectives and preliminary actions to derive the behaviours. The model is evaluated with GE and GP based models where the results prove that both the models perform successfully in evolving desired behaviours with atomic components of the rules, while the GE approach is more successful in generating reusable behavioural rules, and it has the potential to evolve more complex behaviours in a modularised approach.

Immediate extensions to the proposed work include enhancing the rule space to include more components, and conducting a rigorous sensitivity analysis to determine the effect of different manual tuned parameters for the GP approach where they cannot be included in the evolutionary model. Future research directions for the above work involve employing heterogeneous agent systems where different boids follow different rules and a set of rules is evolved over generations to obtain complex behaviours. The modularised approach with GE shows strong potential in generating emergent behaviour which can be further experimented and analysed with different modularity and hierarchical techniques to combine modules in different agent communities following different grammars, to evolve more complex and high fidelity behavioural rules which cannot be foreseen by a hand crafting approach.

REFERENCES

[1] Adrian Agogino and Kagan Tumer. 2005. Reinforcement learning in large multi-agent systems. In *AAMAS-05 Workshop on Coordination of Large Scale Multi-Agent Systems. Utrecht, Netherlands.*

[2] Yen-Wei Chen, Kanami Kobayashi, Hitoshi Kawabayashi, and Xinyin Huang. 2008. Application of Interactive Genetic Algorithms to Boid Model Based Artificial Fish Schools. In *Knowledge-Based Intelligent Information and Engineering Systems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 141–148.

[3] A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing.* Springer Berlin Heidelberg, Berlin, Heidelberg.

[4] Eliseo Ferrante, Edgar Duéñez-Guzmán, Ali Emre Turgut, and Tom Wenseleers. 2013. GESwarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation.* ACM, 17–24.

[5] John H. Holland. 1975. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press. 183 pages.

[6] Yaqing Hou, Yew-Soon Ong, Liang Feng, and Jacek M Zurada. 2017. An Evolutionary Transfer Reinforcement Learning Framework for Multiagent Systems. *IEEE Transactions on Evolutionary Computation* 21, 4 (2017), 601–615.

[7] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[8] John R. Koza. 1992. *Genetic programming : on the programming of computers by means of natural selection.* MIT Press. 819 pages.

[9] John R Koza. 1994. Evolution of Emergent Cooperative Behavior using Genetic Programming. *Computing with Biological Metaphors* (1994), 280–297.

[10] Erandi Lakshika, Michael Barlow, and Adam Easton. 2013. Co-evolving semi-competitive interactions of sheepdog herding behaviors utilizing a simple rule-based multi agent framework. In *2013 IEEE Symposium on Artificial Life (ALife).* IEEE, 82–89.

[11] M J Mataric. 1993. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. In *Proceedings Simulation of Adaptive Behavior.* MIT Press, 432–441.

[12] Nicholas A. Mecholsky, Edward Ott, Thomas M. Antonsen, and Parvez Guzdar. 2012. Continuum modeling of the equilibrium and stability of animal flocks. *Physica D: Nonlinear Phenomena* 241, 5 (mar 2012), 472–480.

[13] James E. Murphy, Michael O’Neill, and Hamish Carr. 2009. Exploring Grammatical Evolution for Horse Gait Optimisation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5481 LNCS (2009), 183–194.

[14] M. O’Neill and C. Ryan. 2001. Grammatical evolution. *IEEE Transactions on Evolutionary Computation* 5, 4 (2001), 349–358.

[15] Liviu Panait and Sean Luke. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems* 11, 3 (2005), 387–434.

[16] Diego Perez, Miguel Nicolau, Michael O’Neill, and Anthony Brabazon. 2011. Evolving Behaviour Trees for the Mario AI Competition Using Grammatical Evolution. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6624 LNCS, PART 1 (2011), 123–132.

[17] Vicenc Quera, Francesc S Beltran, and Ruth Dolado. 2010. Flocking behaviour: agent-based simulation and hierarchical leadership. *Journal of Artificial Societies and Social Simulation* 13, 2 (2010), 8.

[18] Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics* 21, 4 (aug 1987), 25–34.

[19] Craig W Reynolds. 1992. An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion. In *Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92).* MIT Press, 384–392.

[20] Craig W Reynolds. 1999. Steering Behaviors For Autonomous Characters. In *Game Developers Conference.* 763–782.

[21] Terry Lima Ruas, Maria das Graças Bruno Marietto, AndreÌĄ Filipe de Moraes Batista, Robson dos Santos FrancÌğa, Alexandre Heideker, Emerson Aguiar Noronha, and FaÌĄbio AragaÌČo da Silva. 2011. Modeling artificial life through multi-agent based simulation. In *Multi-Agent Systems-Modeling, Control, Programming, Simulations and Applications.* InTech.

[22] Tamas Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. 1995. Novel Type of Phase Transition in a System of Self-Driven Particles. *Physical Review Letters* 75, 6 (aug 1995), 1226–1229. arXiv:cond-mat/0611743

[23] Christopher R Ward, Fernand Gobet, and Graham Kendall. 2001. Evolving Collective Behavior in an Artificial Ecology. *Artificial Life* (2001).

[24] Byoung-Tak Zhang and Dong-Yeon Cho. 1998. Evolving Complex Group Behaviors Using Genetic Programming with Fitness Switching. *Artificial Life and Robotics* 4, 2 (1998), 431–439.