Grammar based Cooperative Learning for Evolving Collective Behaviours in Multi-agent Systems
Supplementary Material

Dilini Samarasinghe, Michael Barlow, Erandi Lakshika, Kathryn Kasmarik

I. Syntax of Rules

Figure 1 illustrates an example aggregated rule that is derived based on the flowchart in Figure 2 (this is presented described in the paper in pages 3 and 4 but is added again here for clarity) which formulates the syntax that the rules should follow.

First, rule component 1 is initialised, and the **IF** control structure is selected. As the next step, the **CONDITION** component is selected. The process follows the path of the top arrow from **IF** and selects the logical connector 'AND'. From there, it then selects the relational connectors 'GREATER THAN' and 'LESS THAN' as the arguments for the 'AND' connector. Next, it selects the set of parameters according to the flowchart. For the 'GREATER THAN' component, 'distance to flock center' and '150' are chosen as the attribute and value, respectively. For the 'LESS THAN' component, 'separation distance' and '50' are selected. Then the process moves to the **THEN** component and from the two options available (an action or another nested if structure), an action is selected, which is 'move towards flock centre'. Then the process moves to the **ELSE** component, and again from the available two options (an action or another nested if structure), the action 'move towards boid' is selected. This ends the design of the first rule component, but instead of ending the process, it chooses to select another rule component to form an aggregation of two rule components, and moves to the beginning.

Then, the rule component 2 is initialised, and the **IF** control structure is selected. As the **CONDITION**, the process chooses the bottom arrow and directly moves to select a single relational connector without trying to combine more relational connectors. As such, 'GREATER THAN' component is selected. Next, it selects the set of parameters for the component as 'separation distance' and '100'. Then the process moves to the **THEN** component and selects the action 'move towards boid'. As the **ELSE** component, the process chooses to include another nested if control instead of an action. Therefore, it moves back to the **IF** control and selects a **CONDITION** component, which is the relational connector 'EQUAL TO'. In the next step, parameters are selected as 'distance to flock center' and '10'. Moving on to the **THEN** component, an action 'move away from flock centre' is selected. As the **ELSE**, another action 'move towards flock centre' is selected. The design of the rule component 2 is finalise, and the process chooses to end without adding any more rule components to the aggregation.

As such, the designed rule can be read as a combination of two major components. The rules are applied by each boid on

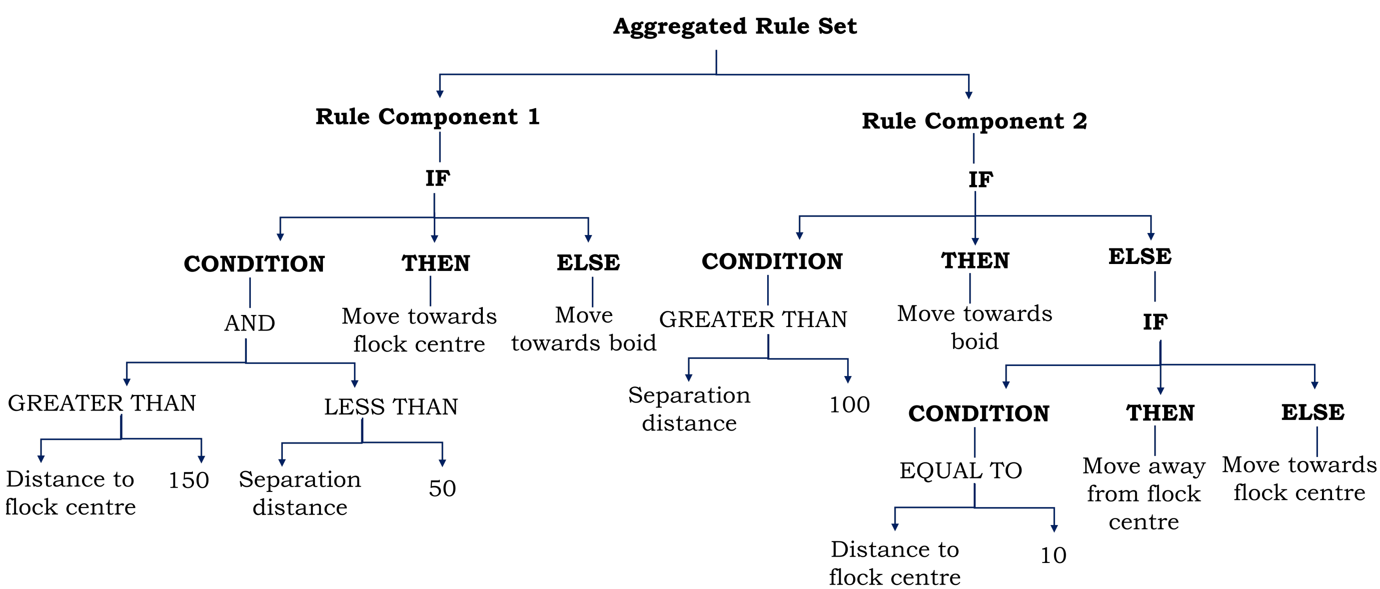


Fig. 1. Example rule generated based on the syntax defined.

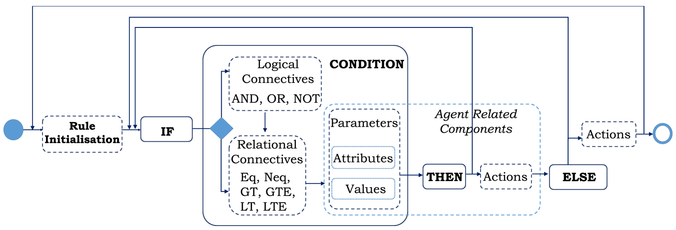


Fig. 2. Syntax followed by the grammar in designing the structure of the behaviour rules. The flow chart describes the steps that should be taken during the formation of a simple or an aggregated rule and allows for the combination of several such rules as required, for achieving complex behaviours.

each of its neighbour boids within the vision range separately. The first components says if the boid is within a distance of 150 units from the flock centre and the separation distance to its current neighbour in consideration is less than 50 units, it should move towards the flock centre, otherwise, should move towards the neighbour boid. The second rule component says, if the boid's separation distance from the neighbour is greater than 100 units, then it should move towards that neighbour, or else should again evaluate whether its distance from the flock centre is 10 units. If it is 10 units, then the boid should move away from the flock centre, if not, it should move towards the flock centre.

The grammar is designed such that the terminal rules that emerge from following the mapping process adhere to the syntax described in this section when being mapped to the solution space with the mapping function.

II. Grammar used in the Experiments

For the proposed experiments with the boids simulation system, the CFG of which the production rules are illustrated in Figure 3 is used. It is the same grammar used with the former homogeneous model proposed in [1], and is represented in Backus-Naur form defined by G = { ν, τ, ρ, ς }. ν represents the non-terminals and τ represents the set of terminals. ρ defines the production rules and ς the start symbol which in this case is $\langle$S$\rangle$. Each production rule is consisted of a non-terminal component on its left side and a combination of non-terminals and terminals on the right side which can replace the left component. The terminals are components which cannot be further replaced.

The syntax as described by the grammar is as follows: The first production rule $\langle$S$\rangle$ ensures each individual rule is consisted of a $\langle$distance$\rangle$ and an $\langle$angle$\rangle$ which determine the range of vision of each boid in the simulation system. The parameters are initially replaced by random values for each boid to maintain heterogeneity and will be evolved through the process. The next production rule of $\langle$S1$\rangle$ is designed such that individual behaviours are not confined to single rules but rather can be the result of an aggregation of a set of single rules combined using a weight $\langle$W$\rangle$, that is assigned to every rule. The weights are also evolved over the generations. Each single rule $\langle$I$\rangle$ follows [*if*] [*condition*] [*then-do*] [*else-do*] syntax. $\langle$O$\rangle$, which is the condition, could be composed of one of the relational connectives *between* or *LTE* (less than or equal to), or a combination of them tied using *and* or *or*

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

Fig. 3. Production rules (ρ) of the grammar (G) represented in Backus-Naur form [1]. The combination of non-terminals and terminals on the right can replace the non-terminal they are associated with. The terminal components cannot be replaced. The range of values for distance and angle are (0 - width of the world], and (0 - 2π] respectively. The weight values are decided based on the number of individual rules derived for a particular aggregated set of rules and by randomly assigning a percentage weight to each rule.

logical connectives. Based on the outcome of the condition, the *then* and *else* components of the rule could lead to another nested *if* rule or to a preliminary action $\langle$A$\rangle$. *LTE* returns a boolean result based on evaluations whether the first argument is equal to or less than the second argument. Similarly, *between* also returns a boolean value with respect to whether the first argument is within its second and third arguments. The first argument relevant to the two connectives can be either *distance to flockcentre* or *separation distance* defined under non-terminal $\langle$P$\rangle$. $\langle$A$\rangle$ defines the preliminary actions and the action $\langle$turn by$\rangle$ accepts the angle to turn as an argument. The random values for angles and distances ($\langle$distance$\rangle$, $\langle$angle$\rangle$) are evolved over the generations along with the rule.

III. Robustness of the Algorithms across Problems

This section analyses the robustness of our proposed team learning and cooperative coevolution algorithms in addressing different problems based on the four behaviours; alignment, avoidance, cohesion and flocking. It is evaluated in terms of the variability of the best solutions achieved after 1000 generations for each of the four beahaviours across 15 runs each. Table I presents the statistical results (maximum, minimum,

TABLE I
ROBUSTNESS OF THE ALGORITHMS

Algorithm	Behaviour	Minimum	Maximum	Mean	Median	Standard Deviation
TL	Alignment	0.0004	0.0338	0.0093	0.0063	0.0095
	Avoidance	0.0244	0.1218	0.0601	0.0487	0.0354
	Cohesion	0.0890	0.1793	0.1390	0.1342	0.0224
	Flocking	0.1697	0.2648	0.2168	0.2192	0.0302
CCE	Alignment	0.0234	0.0763	0.0461	0.0425	0.0139
	Avoidance	6.6613E-16	0.0731	0.0422	0.0487	0.0208
	Cohesion	0.1591	0.2054	0.1722	0.1720	0.0120
	Flocking	0.2518	0.2902	0.2702	0.2737	0.0115
Extended CCE	Alignment	0.0026	0.0108	0.0054	0.0054	0.0021
	Avoidance	0.0000	4.774E-15	7.1794E-16	2.2204E-16	1.2677E-15
	Cohesion	0.0665	0.0970	0.0829	0.0835	0.0067
	Flocking	0.1934	0.2611	0.2291	0.2245	0.0185

mean, median and standard deviation) of the best solutions achieved for the four behaviours for the team learning, co-operative coevolution and extended cooperative coevolution approaches.

In examining the standard deviation of the best solutions for evolution of each behaviour, it is evident that all three approaches are capable of generating consistent solutions as they maintain a low standard deviation (< 0.05) across all four problems tested. The extended CCE approach maintains the lowest standard deviation of the 3 algorithms. CCE approach comes next, however the other statistics prove that the solutions are not near optimal. Hence, it can only be concluded that CCE generates consistent solutions which are not satisfactory. TL has the lowest robustness in comparison to other two, but still with a low standard deviation of < 0.036 for all problems giving both satisfactory and consistent results. In conclusion, both the TL and the extended CCE approaches exhibit a robust performance within and across problems with minimal variability among solutions and satisfactory fitness values.

IV. Sensitivity Analysis based on Mutation and Crossover Rates on Team Learning Algorithm

The following section analyses the effect of crossover and mutation rates on the performance of the team learning algorithm. The analysis is not conducted with the cooperative coevolution mechanism, as it does not incorporate crossover and the representative is replaced with the best phantom agent applied with single point mutation (necessarily leaving mutation rate at 1.0).

The comparison is conducted with the flocking behaviour. Three different mutation rates and crossover rates are compared; 0.5, 0.05, 0.005 and 0.5, 0.8, 1.0 respectively and the results are averaged across 5 runs. For the variations in mutation rates, crossover rate was maintained at 1.0, and for the variations in crossover rates, mutation rate was maintained at 0.5, to be consistent with other experiments. Figure 4 illustrates the comparison of fitness distribution of the best solutions obtained with each of the variations of mutation and crossover rates. The distributions of the best solutions overlap for both mutation and crossover variations implying that there's not enough evidence to suggest a significant difference in the impact of variations on the performance.

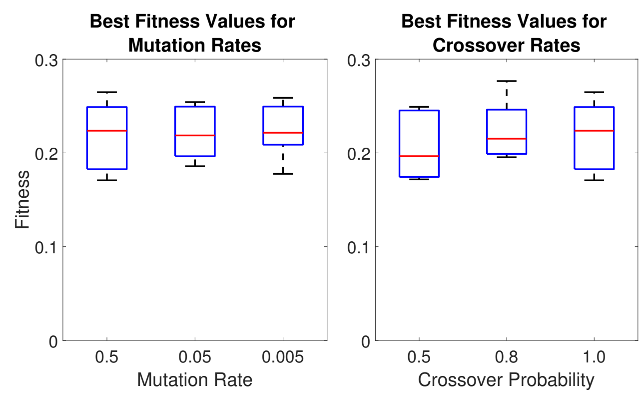


Fig. 4. Comparison of performance of the best solutions averaged across 5 runs of each variatoin for mutation and crossover rates. The central line indicates the median result, bottom and top edges indicate the 25th and 75th percentiles, respectively. The whiskers extend to the extreme results not considered outliers.

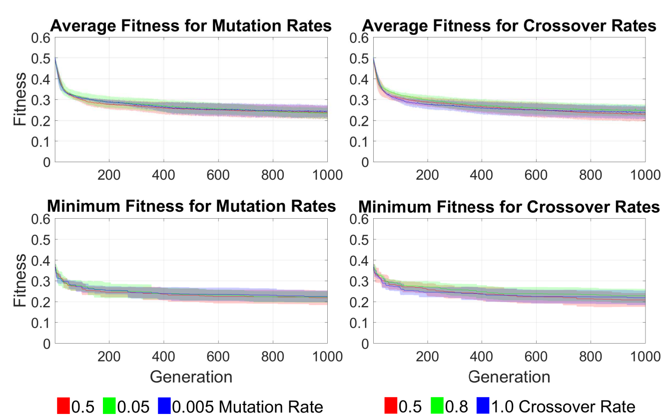


Fig. 5. Evolution results for the flocking behaviour with varying mutation and crossover rates. The left column illustrates the average fitness progression of the solution space through 1000 generations whereas the right column illustrates the evolution of the best solution through 1000 generations. The experiments were repeated for 5 runs and the shaded area depicts the standard deviation.

Furthermore, evolutionary results over 1000 generations for each variation of mutation and crossover rates were analysed. Figure 5 depicts these results which suppport the previous claims. From the results, it is evident that both mutation and crossover rates have statistically insignificant ($p > 0.05$) difference in the impact on the performance of the algorithm.

V. Rule Complexity Analysis

Further evaluations results of the rule complexity analysis presented in the paper are given here. Figure 6 illustrates the variation of the average number of rules in relation to the fitness for all evolutionary runs over 1000 generations. The patterns observed are similar to those observed with the variation of average cyclomatic complexity in relation to the fitness.

A correlation between the number of rules and the fitness cannot be observed based on the results. However, the homogeneous model explores a broader range of rules at the beginning of the evolution process, and as it converges to a solution settles for aggregated rules with around 2 individual rules in general, but go up to 8 rules in some cases. TL, similar to cyclomatic complexity variations, remain restricted in its strategy and from the beginning of the evolutionary process, limits its explorations to a closer neighbourhood. In both the CCE methods, the phantom agents explore varying number of rules from 2 - 10 throughout the evolutionary process even after the representatives have converged to a solution maintaining a steady fitness.

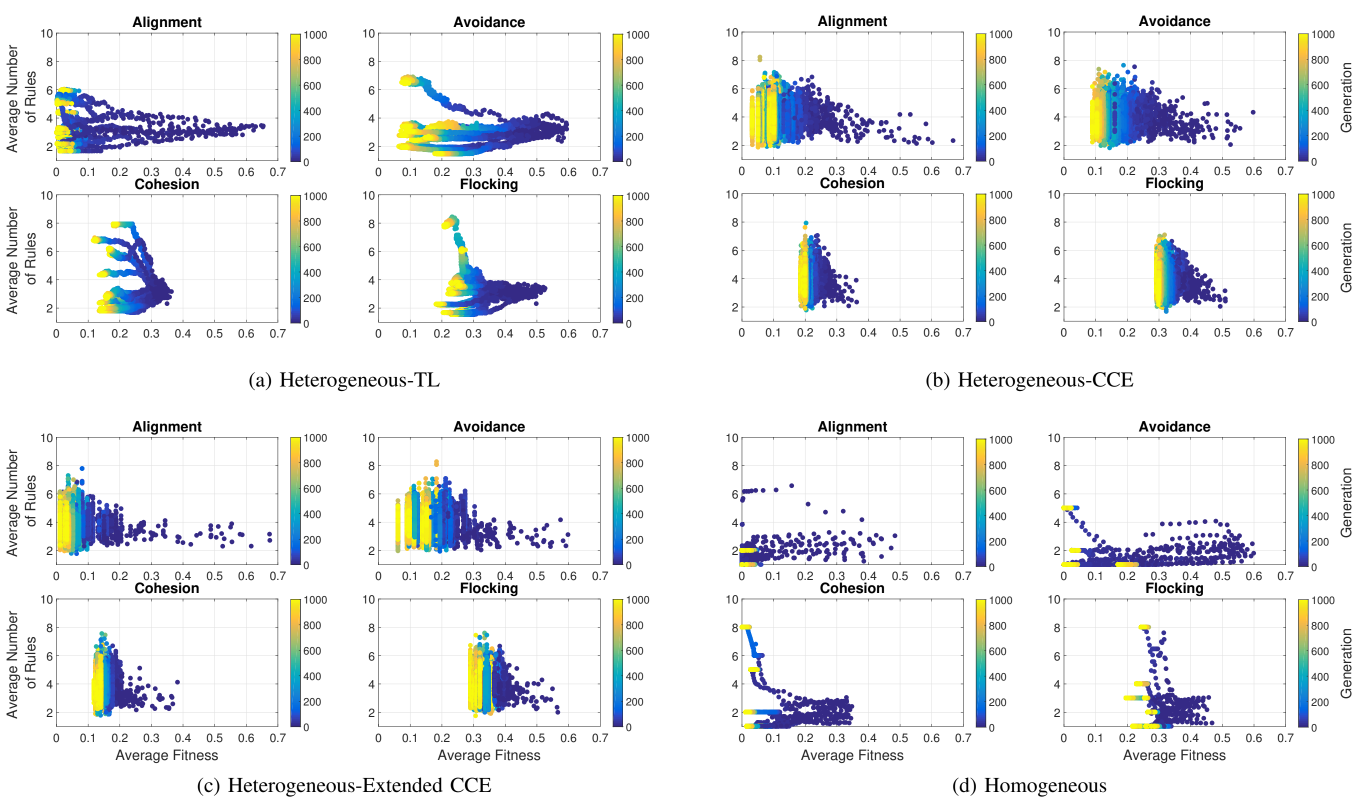
VI. Analysis of Evolved Rule Structures

In this section, we demonstrate the process of mapping a genome to a behaviour rule and analyse the features of the rule, considering an example for the flocking behaviour. Figure 7 illustrates an indicative rule evolved for an agent engaged in flocking behaviour. As can be seen, it has the capacity to decide the number of aggregated rules, their structures, as well as components, that will address simpler actions which can jointly generate the expected emergent behaviour.

```
[ Angle of vision=0.79 Distance of vision=314.35
  [ Rule 1 * 0.75
    [ <IF>  <LTE>  distance_to_flock_centre  96.75
      [ <IF>  <BETWEEN>  distance_to_flock_centre  63.46 35.44
          match_velocity_with_boid
          move_away_from_flockcentre
    move_towards_boid]]]
  [ Rule 2 * 0.25
    [ <IF>  <LTE> seperation_distance  330.70
      turn_by 4.91
      turn_by, 5.36]]]
```

Fig. 7. Indicative rule of an agent engaged in flocking behaviour.

Within the context of our simulation environment, the interactions among agents are based on vision. Hence, the syntax was designed in such a manner that each rule will evolve the components (angle of vision, distance of vision) related to vision range as well. For the example considered here, the respective values 0.79 (within 0-2*PI) and 314.35



(a) Heterogeneous-TL (b) Heterogeneous-CCE (c) Heterogeneous-Extended CCE (d) Homogeneous

Fig. 6. Variation of number of rules in relation to the fitness of the rule structures for 15 evolutionary runs with the TL, CCE, extended CCE methods against the homogeneous model over 1000 generations. The values are averaged across all agents of a group of 30 for each generation. The individual values are colour coded based on the generation.

(within 0-width of the environment) were assigned. The main rule is an aggregation of two simpler rules each carrying a repective weight of 75% and 25%. The first rule decides the movement of the agent by specifying that it should move towards the neighbouring agents if it is too far from the flock centre, stay within the flock and align with the rest of the neighbours if it is within acceptable distance or should move away from the centre of the flock to maintain shape if it is too close to the centre. The second rule decides the angle to turn; if the agent is relatively close to the neighbouring agent (with a lower separation distance) the turn should be narrower. The combination of these two simpler rules is expected to result in an agent contributing towards maintaining a satisfactory flocking behaviour throughout the system. Given that this is a heterogeneous context, each agent will follow a different rule each evolved in the same manner resulting in the emergent flocking behaviour desired.

We then discuss how the mapping process is conducted to generate the above rule from the respective genome. Figure 8 illustrates an indicative genome of 100 codons as used in the experiments with this paper, the mapping function used in grammatical evolution and the grammar designed along with the number of production rules for each non-terminal.

As depicted in figure 9 the mapping of the genome to a valid phenome starts with the first non-terminal $\langle$S$\rangle$. As it has only one production rule, the codons are spared and the mapping function is not used. It is simply replaced by the corresponding production rule. Next, the first non-terminal in the current rule is considered. Similar to the previous step, the non-terminal is replaced with the terminal value, which in this case is a randomly generated angle value within a valid range. The same applies to specifying the distance value. The next non-terminal $\langle$S1$\rangle$ has 2 production rules and hence the mapping function is used here for the first time. The first codon to be considered is 164. The mapping function (164 % 2) returns 0 resulting in the selection of 0^{th} production rule to replace the non-terminal $\langle$S1$\rangle$ which is $\langle$S1$\rangle$ $\langle$S1$\rangle$.

Similarly, the process continues until a rule consisting entirely of terminals is produced. The process may terminate before reaching the end of the genome, and also might run out of codons. In the second case, the codons from the beginning can be re-used (wrapped) up to 50 times. In case a terminal rule is not generated within the specified number of wraps the genome is deemed invalid and removed from the population.

Since the rule generation process is based on the syntax defined by the grammar, this approach restricts the solution space only to the set of valid rules unlike other evolutionary mechanisms such as genetic programming which have no means of confining the evolved solutions. Close examination of this rule proves that our proposed grammar based evolutionary mechanism has more flexibility than any approach which involves manual crafting of the rule structures for multi-agent systems. The evolutionary algorithm has the freedom to manipulate all the atomic components of the rule, thus increasing the potential of the algorithm and eliminating the limitations associated with human bias in designing the rules.

References

[1] D. Samarasinghe, E. Lakshika, M. Barlow, and K. Kasmarik, “Automatic synthesis of swarm behavioural rules from their atomic components,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO ’18. New York, NY, USA: ACM, 2018, pp. 133–140.

Genome of 100 codons

164	093	011	041	023	062	023	116	170	019	111	063	237	086	090	082	076	006	039	102
013	054	200	137	022	012	162	201	001	143	097	033	222	252	251	244	252	003	189	146
001	135	028	111	114	241	140	235	065	244	182	066	023	137	195	070	206	079	064	160
007	158	133	130	250	137	152	026	134	224	206	058	099	205	153	182	161	070	045	153
030	107	181	065	226	019	255	095	096	011	210	251	110	111	171	249	080	138	091	056

Mapping Function

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

Grammar

⟨S⟩	$\models$	⟨angle⟩ ⟨distance⟩ ⟨S1⟩	1
⟨S1⟩	$\models$	⟨S1⟩ ⟨S1⟩ | ⟨I⟩*⟨W⟩	2
⟨I⟩	$\models$	if ⟨O⟩ ⟨I⟩ ⟨I⟩	4
		| if ⟨O⟩ ⟨I⟩ ⟨A⟩	
		| if ⟨O⟩ ⟨A⟩ ⟨I⟩	
		| if ⟨O⟩ ⟨A⟩ ⟨A⟩	
⟨O⟩	$\models$	and ⟨O⟩ ⟨O⟩	4
		| or ⟨O⟩ ⟨O⟩	
		| LTE ⟨P⟩ ⟨distance⟩	
		| between ⟨P⟩ ⟨distance⟩ ⟨distance⟩	
⟨P⟩	$\models$	separation distance	2
		| distance to flockcentre	
⟨A⟩	$\models$	move away from boid	7
		| move forward	
		| move away from flockcentre	
		| move towards flockcentre	
		| move towards boid	
		| turn by ⟨angle⟩	
		| match velocity with boid	
⟨distance⟩	$\models$	random distance	1
⟨distance of vision⟩	$\models$	random distance	1
⟨angle⟩	$\models$	random angle	1
⟨angle of vision⟩	$\models$	random angle	1
⟨W⟩	$\models$	random weight	1

Fig. 8. Indicative genome, mapping function and the number of production rules. The genome contains 100 codons each of 8 bits. With the proposed genome structure to represent multiple individuals in one genome, the mapping process will begin at different codons. The mapping function takes the remainder after division of the integer value of the current codon by the number of production rules for the current non-terminal considered. If the non-terminal has only one production rule, the codon is spared to be used at the next step and it is directly mapped to the said production rule. The grammar has 11 non-terminals each with production rules ranging from 1-7.

< S >

(< angle > < distance > < S1 >)

(0.79 < distance > < S1 >)

(0.79 314.35 < S1 >)

164 % 2 = 0

(0.79 314.35 < S1 > < S1 >)

93 % 2 = 1

(0.79 314.35 < I > * W < S1 >)

11 % 2 = 1

(0.79 314.35 < I > * W < I >* W)

41 % 4 = 1

(0.79 314.35 if < O > < I > < A > * W < I >* W)

(0.79 314.35 if < O > < I > < A > * 0.75 < I >* W)

23 % 4 = 3

(0.79 314.35 if < O > < I > < A > * 0.75
if < O > < A > < A > * W)

(0.79 314.35 if < O > < I > < A > * 0.75
if < O > < A > < A > * 0.25)

62 % 4 = 2

(0.79 314.35 if LTE < P > < distance > < I > < A > * 0.75
if < O > < A > < A > * 0.25)

23 % 4 = 3

(0.79 314.35 if LTE < P > < distance > if < O > < A > < A > < A > * 0.75
if < O > < A > < A > * 0.25)

116 % 7 = 4

(0.79 314.35 if LTE < P > < distance > if < O > < A > < A > move_towards_boid * 0.75
if < O > < A > < A > * 0.25)

170%4 = 2

(0.79 314.35 if LTE < P > < distance > if < O > < A > < A > move_towards_boid * 0.75
if LTE < P > < distance > < A > < A > * 0.25)

19%7 = 5

(0.79 314.35 if LTE < P > < distance > if < O > < A > < A > move_towards_boid * 0.75
if LTE < P > < distance > < A > turn by < angle > * 0.25)

111%2 = 1

(0.79 314.35 if LTE distance_to_flockcentre < distance > if < O > < A > < A > move_towards_boid * 0.75
if LTE < P > < distance > < A > turn by < angle > * 0.25)

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if < O > < A > < A > move_towards_boid * 0.75
if LTE < P > < distance > < A > turn by < angle > * 0.25)

63%4 = 3

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN < P > < distance > < distance > < A > < A > move_towards_boid * 0.75
if LTE < P > < distance > < A > turn by < angle > * 0.25)

237%7 = 6

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN < P > < distance > < distance > match_velocity < A > move_towards_boid * 0.75
if LTE < P > < distance > < A > turn by < angle > * 0.25)

86%7 = 2

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN < P > < distance > < distance > match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE < P > < distance > < A > turn by < angle > * 0.25)

90%2 = 0

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN < P > < distance > < distance > match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance < distance > < A > turn by < angle > * 0.25)

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN < P > < distance > < distance > match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance 330.70 < A > turn by < angle > * 0.25)

82%7 = 5

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN < P > < distance > < distance > match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance 330.70 turn_by < angle > turn by < angle > * 0.25)

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN < P > < distance > < distance > match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance 330.70 turn_by < angle > turn by 5.36 * 0.25)

76%2 = 0

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN separation_distance < distance > < distance > match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance 330.70 turn_by < angle > turn by 5.36 * 0.25)

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN separation_distance 63.46 < distance > match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance 330.70 turn_by < angle > turn by 5.36 * 0.25)

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN separation_distance 63.46 35.44 match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance 330.70 turn_by < angle > turn by 5.36 * 0.25)

(0.79 314.35 if LTE distance_to_flockcentre 96.75 if BETWEEN separation_distance 63.46 35.44 match_velocity move_away_from_flockcentre move_towards_boid * 0.75
if LTE separation_distance 330.70 turn_by 4.91 turn by 5.36 * 0.25)

Fig. 9. Mapping process. It is started with the first non-terminal ⟨S⟩. Whenever there is only one production rule involved, the corresponding production rule is selected without applying the mapping function as it is unnecessary. Next the first non-terminal of the current production rule is considered and the mapping function is applied if there are more than one production rules to select the next candidate. The process is continued until the rule consists entirely of terminals.