A Logical Transformation Method of the Motion Rules for Swarms

1st Jingjing Tao
College of Systems Engineering
National University of Defense
Technology Changsha, China
taojingjing19@nudt.edu.cn

2nd Xiaomin Zhu*
College of Systems Engineering
National University of Defense
Technology Changsha, China
xmzhu@nudt.edu.cn

3rd Li Ma
College of Systems Engineering
National University of Defense
Technology Changsha, China
mali10@nudt.edu.cn

4th Meng Wu
College of Systems Engineering
National University of Defense
Technology Changsha, China
wumeng15@nudt.edu.cn

5th Xiaoqing Li
College of Systems Engineering
National University of Defense
Technology Changsha, China
lixiaoqing21@nudt.edu.cn

6th Liyuan Niu
College of Systems Engineering
National University of Defense
Technology Changsha, China
niuliyuan@nudt.edu.cn

2021 7th International Conference on Big Data and Information Analytics (BigDIA) | 978-1-6654-2466-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/BIGDIA53151.2021.9619739

***Abstract*—Swarm intelligence involves designing appropriate rules which make swarms with the same limited ability organize themselves to accomplish desired tasks. Numerous studies have designed many different motion rules. According to a designed rule, a swarm can self-organize into a desired pattern, such as a gathering pattern or a flocking pattern. However, for swarms that need to complete multiple tasks, it tends to be not enough to devise motion rules of forming a single pattern. Therefore, this paper presents a logical transformation method of motion rules for swarms, which essentially matches different motion rules to corresponding conditional states. The method consists of two steps: the construction of the set of conditional states and the set of motion rules, and the matching optimization calculation using genetic programming. The feasibility of this method is verified by simulations.**

***Index Terms*—swarm intelligence, motion rules, gene programming, matching optimization**

I. Introduction

The design of the motion rules is one of the important research directions of swarm intelligence [1]. Currently, typical motion rules for swarms mainly include Reynold's three rules [2], Gene Regulation Network (GRN) [3], and artificial potential field [4]. Reynold's three rules are the general rules for swarms designed according to the flight of a flock of birds. Individuals in a swarm can maintain the normal movement of the swarm by avoiding collision, keeping the consistent direction, and keeping close [5]. Inspired by the generation mechanism from DNA to protein in the organism, GRN can be used to generate a model of motion rules controlled by protein [6]. The formation of the protein concentration field is determined by the location of obstacles, targets, and neighbors. All individuals in a swarm will continue to move in the direction of the greatest reduction in the protein concentration around them, until the concentrations of all individuals in the swarm reach the lowest in the surrounding concentration. In this way, the swarm can move in an efficient manner and eventually form a desired encircling pattern [7]. Artificial potential field is usually used to control the movement direction of individuals in a swarm by constructing concentration field artificially, which is usually applied to the study of swarm gathering and path planning [8]. In the above three motion rules, Reynold's three rules are mainly applied to the movement of swarms, artificial potential field are all built based on human subjectivity, and GRN not only take advantage of the fact that proteins are produced in different concentrations according to the expression of different genes in organisms, but also can be adapted to different environments and to swarms with basic computational and sensory functions.

With the development of swarm intelligence, the design of single motion rule in specific scenarios can no longer meet the needs of swarms to perform multiple tasks [9]. Based on the existing motion rules, if a conversion mechanism between different motion rules can be designed, this problem can be solved. Although the above motion rules are specific to swarms, what these rules actually limit is the movement of each individual in a swarm. Different motion rules correspond to the execution of different tasks in different conditions, which can be easily understood as an if-then relationship [10]–[12]. Therefore, as long as all the matching relations between all different conditional states and tasks to be executed are found, corresponding motion rules can be selected according to different conditional states, so as to design the logical transformation method of the motion rules for swarms. It is obvious that this method consists of two steps. The first step is to list all the possible conditional states and all the motion rules that can be performed. The second step is to find an effective matching scheme between the conditional states and motion rules.

There are usually a great many of conditional states for an individual. The selection of conditions often depends on the

This work was supported in part by the National Natural Science Foundation of China under Grant (61872378, 62102445), Xiaomin Zhu is the corresponding author.

task to be completed [13]. For example, the conditional states for the condition of whether find a target can be set as "found target" or "not found target" for the task of searching for a target. The conditional states of "target survival" and "target dead" are not necessary for this task, because no matter which state the target is in, it indicates that the target has been found and the task of searching has been completed. In addition to the surrounding environment conditions, the individuals' own conditions can also affect the implementation of tasks, such as the battery power. Therefore, we can list all the possible conditional states according to the tasks and the performance of the individuals in the swarm.

Based on Reynold's three rules, GRN, artificial potential field and so on, one individual can perform behaviors like searching and attacking, and multiple individuals can emerge behaviors, such as clustering and entrapping. Refer to [9], we name the motion rules performed by an individual according to the behaviors emerged through executing the same motion rules by each one in the swarm. For example, even though an individual is unable to entrap a target, we define the motion rules of entrapping for an individual as: if multiple individuals in a swarm act according to the same motion rules of entrapping, they can emerge the behavior of entrapping. All the motion rules that can be performed by an individual in a swarm can be named with the method described above. So that we can list all the motion rules that can be conducted by an individual in the swarm.

With all the possible conditional states and all the behaviors that can be executed, the next step is to determine the matching relationship between them. This can be viewed as a matching optimization problem, and it is only necessary to select the appropriate behavior for each possible conditional state [10]–[12]. In fact, such a matching relationship is explainable. We can clearly know which behavior an individual will perform under certain conditional states and which behavior it will switch to perform when the conditional states change [14]. For different optimization method, reinforcement learning breaks the explainability of the learning process. Other intelligent optimization methods retain this characteristic better [14]. Evolutionary algorithm is a discrete optimization algorithm inspired by species evolution, which has preferable global optimization ability and convergence [9]. It has obvious evolutionary explainability characteristics. Evolutionary algorithm mainly includes GA, Gene Programming, and other methods. GA is generally used to evolve object whose characteristics can be encoded with binary string. Different encoding segment represent different features. GP is mainly used to evolve expressions with logical tree structure [15]. For the purpose of this study, we chose to use a modified GP algorithm to find an effective matching scheme between conditional states and motion rules, that is, a logical transformation method of the motion rules.

The main contributions of this paper are as follows:

- A logical transformation method of the motion rules is proposed.
- A set containing all conditional states and a set containing all executable motion rules is constructed.
- The modified GP is used for matching optimization of conditional states and executable motion rules.

The rest of this paper is organized as follows: Section II introduces the framework of the proposed method. The construction of sets is illustrated in Section III. Section IV presents the modified GP algorithm in detail. In Section V, the proposed method is verified by simulation.

II. The Method Architecture

The method architecture is presented in Figure 1. Its input contains the task scenario and the swarm which needs to accomplish tasks. According to the swarm and task scenario, we can generalize the information of conditional states that can be perceived and the motion rules that can be realized by the individuals in the swarm. Then the set of conditional states and the set of motion rules can be built. With the two constructed sets, we can then use the modified GP algorithm to optimize the matching relationships between conditional states and motion rules. After that, we will obtain the logical transformation scheme of motion rules for the swarm. The concrete procedures are introduced in the following sections.

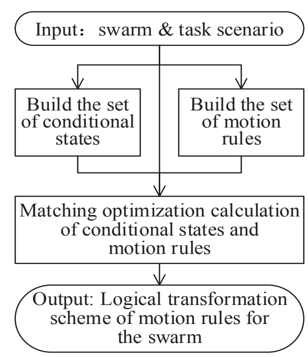


Fig. 1. The framework of the proposed method architecture.

III. The Construction of Sets

A. Construction of the Set of Conditional States

According to the target tasks and the performance of the individuals in the swarm, we can list all the different types of perceptual conditions. Then the conditional states of each perceptual conditions can be listed. Finally, these conditional states that belong to different categories of conditions can be arranged and combined to construct the elements of the set of conditional states.

For example, there is a swarm that need to search for an object and move it to a specified position. The perception conditions of the individuals in this swarm include: whether the object is detected, whether the object has been lifted, and whether the individual has reached the specified position. The corresponding conditional states of these perceptual conditions are: the object is detected or the object is not detected, the object is lifted or the object is not lifted, and the individual reaches the specified position or the individual does not reach the specified position. If we use different capital letters to specify these perceptual conditions as A, B, and C and use different numbers to represent different states of the same condition. These conditional states can be expressed as A1, A2, B1, B2, C1, and C2. By arranging and combining these states, we will get all the possible conditional states represented by the set S = {A1B1C1, A1B1C2, A1B2C1, A1B2C2, A2B1C1, A2B1C2, A2B2C1, A2B2C2}.

B. Construction of the Set of Motion Rules

Since an individual in a swarm can only perform one corresponding task based on the motion rules, permutation and combination of executable motion rules is not needed. As long as we list all the executable motion rules of an individual in the swarm, the set of all the executable motion rules can be constructed.

Similarly, take the individual in the previous subsection as an example, the executable motion rules include searching, grabbing, and moving to the specified position, which can be represented by a, b, and c with different lowercase letters. Then these executable motion rules can be represented by the set A = {a, b, c}.

IV. Design of the Modified GP Algorithm

A. GP Algorithm

The main processes of GP are as follows: First, an initial population consisting of individuals with tree structures is randomly generated. These individuals represent expressions with different operations. Second, the individuals in the initial population should be dealt with genetic operations, such as crossover, reproduction, and mutation. After that, the calculated value of all individuals represented by the tree structures should be assessed by comparing it with the real value. A certain number of individuals with small error will be selected as the initial population of the next round of iterative evolution. If the error of an individual in the population or the number of evolutionary iterations meets the requirements of evolution, the optimal individual with the smallest error represented by a tree structure is the output of this algorithm.

The representation of the tree structure is shown in Figure 2. Nodes in the tree consist of branch nodes and leave nodes. A branch node represents an operation on its child nodes. The branches in the figure include "+", "-", and "*", which correspond to adding, subtracting and multiplying the two child nodes from left to right, respectively. A leaf node represents a parameter or constant value and needs to be added under a branch node. Leaf nodes have no child node. So the tree structure in the figure represents the computation of the expression "$x * 3 + 8 - x$".

Genetic operations of crossover, mutation, reproduction, and selection in GP are as follows. Crossover means that one subtree of each of two individuals is randomly selected and swapped. The two selected subtrees generally have different structures and the number of nodes. Mutation usually means that a random leaf node is selected and this node is replaced by a new node or a subtree. Reproduction means simply copying a random individual from a population as a new individual. Selection means selecting a certain number of individuals according to probability as the initial population of the next round of iterative evolution. And the number is equal to the size of the initial population. Individuals with higher fitness values are more likely to be selected.

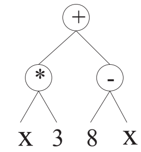


Fig. 2. The representation of a tree structure.

B. Modified GP Algorithm

Since this paper needs to use evolutionary algorithm to get the matching relations between conditional states and motion rules for specific task scenario, some adjustments should be made to GP algorithm to make it suitable for the optimization goal of this study. First, we fix the structure of the tree to be optimized. As shown in Figure 3, the tree is divided into four layers and each individual in the swarm should follow the logic of this tree. In this figure, the leaf node "x" represents the current conditional state. The leaf node "y" represents the motion rule to be executed. "s_i" and "s_j" represent two different conditional states in the set of conditional states. In addition, "a_i" and "a_j" are both motion rules in the set of motion rules. However, "a_i" and 'a_j" might be the same motion rule. The branch node "if" means if the first child node returns "true", the second child is executed and return the execution result "success" or "failure", otherwise return "failure" directly. The branch node "=" indicates: If both child nodes represent conditional states and the conditional states of its two child nodes are the same, return "failure". Otherwise return "failure". Similarly, if both child nodes represent motion rules and the first child node represents the motion rule to be executed, execute the motion rule represented by the second child node and return the execution result "success" or "failure". Otherwise return "failure". The root node "tick" means iterating through this tree, and if the previous(left) "if" node returns "failure", the next(right) "if" node will be performed until one node returns "success". Then skip all subsequent nodes for the next iteration. Each time the individual will judge which conditional state the individual belongs to, and then the individual performs the corresponding motion rule "y". It is important to note that each conditional state can exists and an individual in the swarm can only conduct one motion rule at a time, so all the conditional states to be judged in the tree must contain all of the conditional states in the constructed set and cannot be repeated.

In fact, the set of these judgements is the set of conditional states. For each individual with tree structure in the initial evolutionary population, the order of these judgements and corresponding motion rules are generated randomly. Generic

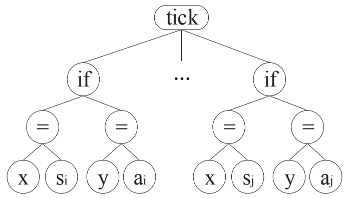
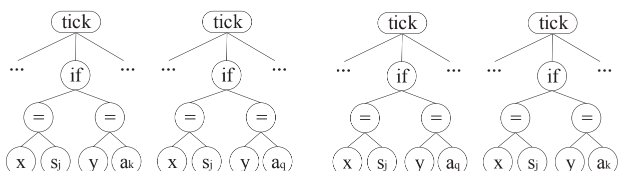


Fig. 3. The tree with fixed structure.

operations also involve crossover, mutation, reproduction, and selection. But due to the design of the fixed tree structure, we adjust these operations appropriately.

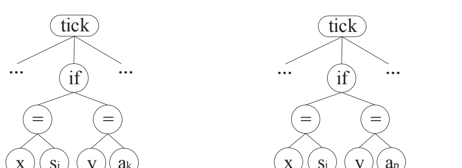
Selection and reproduction operations remain the same as the original GP algorithm. Because the tree structure is fixed, the new crossover operation means swapping the nodes representing the motion rules corresponding to the same conditional state, which is shown in Figure 4. This ensures that the structure of the trees and the set of conditional states remain unchanged during evolution.



(a) Trees before crossover. (b) Trees after crossover.

Fig. 4. Two random trees before and after crossover.

Similarly, in the original algorithm, mutation means replace a leaf node with a node or a subtree. In order to preserve the tree structure, we adjust it to replace a leaf node with another. In addition, the node being replaced can only be the leaf node that represents the motion rule to be executed. An example of mutation is shown in Figure 5. In this figure, the leaf node "a_k" mutates to "a_p" and other nodes remain the same.



(a) The tree before mutation. (b) The tree after mutation.

Fig. 5. A random tree before and after mutation.

V. Simulations and Analysis

A. Experimental Evaluation

To verify the feasibility of our proposed method, we set up a simulation experiment scenario as shown in Figure 6. A small red circle represents an individual robot. The rectangle in the lower right corner represents the base station. A swarm consisting of nine identical robots is at the base station. Large circles with crosses in the scenario represent targets. Each target contains a threshold of the intensity of entrapping. The target can only be transported to the base station if the intensity of entrapping of surrounding robots is greater than or equal to this threshold. Tasks the swarm needs to complete include entrapping these targets and transporting the entrapped targets to the base station. In this scenario, the motion rules that the swarm can follow include the establishment of artificial potential field with the base station as concentration center to search or transport targets and the use of GRN to entrap targets, which can be summarized as searching, entrapping, and transporting. The task goal is to transport all targets in the scenario back to the base station in the shortest possible time.

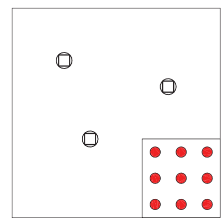


Fig. 6. The simulation experiment scenario.

On the basis of the tasks, we design the fitness function as

$$f = \sum_{i=1}^{100} (c_i/3) \times 100\% \tag{1}$$

where c_i represents the number of targets which have been transported to the base station at the i^{th} time step. This equation means the fitness value after running 100 time steps.

According to the information of the swarm and task scenario mentioned above, we conclude the conditional states as shown in Table I. Thus, the set of conditional states is constructed as S = {A1B1C1, A1B1C2, A1B2C1, A1B2C2, A2B1C1, A2B1C2, A2B2C1, A2B2C2}. Similarly, according to executable motion rules, the set of motion rules is constructed as A = {a, b, c}.

TABLE I
Conditional states of the simulation

Condition	Conditional symbols	Conditional states	Symbols
Is there a target detected?	A	Detect a target	A1
		No target detected	A2
Is the target entrapped?	B	The target is entrapped	B1
		The target is not entrapped	B2
Is the target at base station?	C	The target is at base station	C1
		The target is not at base station	C2

The next step is to use the modified GP algorithm to design a logical transformation method of the motion rules for swarms. The concrete parameters of this algorithm are set in Table II, where the size of the initial population is 10, the probability of selection is 0.1, the probability of crossover is 0.05, the probability of mutation is 0.1, and the probability of replication is 0.05.

TABLE II
PARAMETERS OF THE MODIFIED GP ALGORITHM IN THE EXPERIMENT

Parameters	Values
Size of the initial population	10
Probability of selection	0.1
Probability of crossover	0.05
Probability of mutation	0.1
Probability of reproduction	0.05

According to the processes of the modified GP algorithm, we apply it to the simulation experiment scenario. Result of the experiment is shown in figure 7. The broken line with asterisks represents the best fitness values of each generation during evolution. The dotted line with dots represents the mean fitness values of each generation during evolution. And the fitting curve of the mean fitness value is represented by the dot dash line. It is clear that in the fifth generation, the best fitness value reaches the optimal value 58.33. The mean fitness value of the population is continuously increasing from the third generation to the 15^{th} generation. And then the mean fitness value tend to be stabilized after 15^{th} generation. Screenshots of the optimized logical transformation method of the motion rules used in the simulation scenario are shown in figure 8 in order. Any target in base station is hidden. In figure 8(d), all the targets are disappeared. It is attributed to that all of the targets in this scenario nave been transported to the base station in the end. These results show the feasibility, good optimization ability and convergence of the modified GP algorithm.

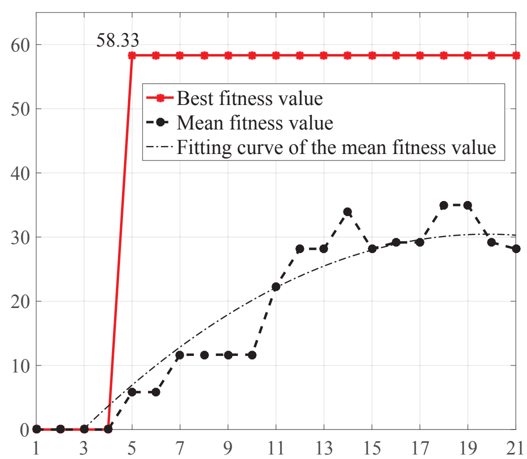


Fig. 7. Changes in fitness values during evolution.

VI. CONCLUSION

In this paper, we proposed a logical transformation method of the motion rules for swarms. There are two steps to the method. The first step is to construct the set of conditional state and the set of motion rules. The second step is to use the modified GP algorithm and the simulation scenario to find an appropriate logical transformation scheme. After that, the feasible optimized scheme can be applied to the scenario.

In fact, the scenario is not so complicated as to use the evolutionary algorithm to find the best possible solution. We

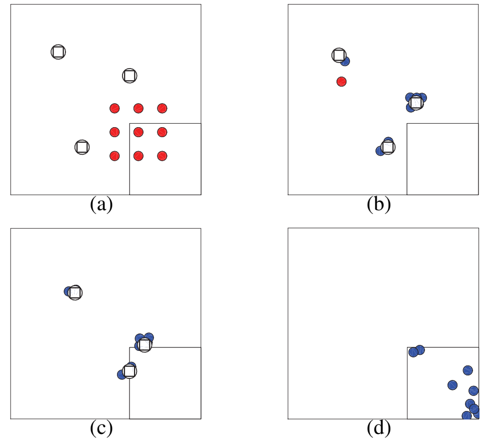


Fig. 8. Screenshots of the application of the optimized results.

are just here to verify the feasibility of the proposed method. On condition that there are too many conditional states, the situation may change and the adaptability of the proposed algorithm may need to be reconsidered. Therefore, our next work is considered to design a method with better adaptability and evolutionary efficiency for more complex task scenarios.

ACKNOWLEDGMENT

First and foremost, I would like to show my deepest gratitude to my supervisors, Professor Zhu Xiomin, a respectable, responsible and resourceful scholar, who has provided me with valuable guidance of this thesis. Second, I gratefully acknowledge the suggestions of senior fellow apprentice and lecturer Wang Ji on the content and structure of this paper. In addition, I would like to express my heartfelt gratitude to Dr. Ma Li who has instructed and helped me a lot in the past two years. Last, my thanks would go to my team mate Wu Meng who gave me a lot of guidance on programming. Above all, I also own my sincere gratitude to my schoolmate Niu Liyuan who gave me a lot of care and encouragement.

REFERENCES

[1] S. S. GE and Y. J. Cui, "Dynamic Motion Planning for Mobile Robots Using Potential Field Method," Proceedings of the 8th IEEE Mediterranean Conference on Control and Automation, pp. 207-222, 2000.
[2] C. W. Reynolds, "Flocks, Herds and Schools: A Distributed Behavioral Model," ACM SIGGRAPH Computer Graphics, pp. 25-34, 1987.
[3] M. Wu, Y. Zhou, X. M. Zhu, et al. "Cooperation-based gene regulatory network for target entrapment," International Conference on Swarm Intelligence, pp. 60—69, 2019.
[4] J. Y. Zhang, Z. P. Zhao, and D. Liu, "A path planning method for mobile robot based on articial potential field," journal of Harbin Institute of Technology, pp. 136-139, 2006.
[5] A. A. Paranjape, S. Chung, Kyunam Kim, et al, "Robotic Herding of a Flock of Birds Using an Unmanned Aerial Vehicle," IEEE Transactions on Robotics, pp. 901-915, 2018.
[6] Y. C. Jin, H. L. Guo, and Y. Meng, "A Hierarchical Gene Regulatory Network for Adaptive Multirobot Pattern Formation," IEEE Transactions on Systems, Man, and Cybernetics, pp. 805-816, 2012.
[7] H. Oh and Y. C. Jin, "Adaptive Swarm Robot Region Coverage Using Gene Regulatory Networks," Conference Towards Autonomous Robotic Systems, pp. 197-208, 2014.

[8] H. Oh, A. R. Shirazi, C. L. Sun, et al, “Bio-inspired Self-organising Multi-robot Pattern Formation: A review,” Robotics and Autonomous Systems, pp. 83–100, 2017.
[9] J. J. Tao, X. M. Zhu, L. Ma, et al, “Benign: An Automatic Optimization Framework for the Logic of Swarm Behaviors,” 2020 IEEE International Conference on Systems, Man, and Cybernetics, pp. 2999–3005, 2020.
[10] S. Yang, X. J. Mao, S. Wang, et al, “Extending Behavior Trees for Representing and Planning Robot Adjoint Actions in Partially Observable Environments,” Journal of Intelligent and Robotic Systems, PP. 102, 2021.
[11] K. Y. Scheper, S. Tijmons, C. C. de Visser, et al, “Behavior Trees for Evolutionary Robotics,” Artificial Life, pp. 23–48, 2016.
[12] Y. P. Wang, S. Li, Q. W. Chen, et al, “Biology Inspired Robot Behavior Selection Mechanism: Using Genetic Algorithm,” Bio-inspired Computational Intelligence and Applications, pp. 777–786, 2007.
[13] M. Colledanchise, R. Parasuraman, and P. Ogren, “Learning of Behavior Trees for Autonomous Agents,” IEEE Transactions on Games, pp. 183-189, 2019.
[14] Z. X. Cai, M. L. Li, W. R. Huang, et al, “BT Expansion: a Sound and Complete Algorithm for Behavior Planning of Intelligent Robots with Behavior Trees” The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), pp. 6058-6065, 2021.
[15] P. G. Espejo, S. Ventura, and F. Herrera, “A Survey on the Application of Genetic Programming to Classification,” IEEE Transactions on Systems, Man, and Cybernetics—part c: applications and reviews, pp. 121-144, 2010.