2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Framework of Systems for Creating Intelligent Behaviors of Imaginary Creatures for Humans

Kei Ohnishi
Kyushu Institute of Technology
Iizuka, Japan
ohnishi@csn.kyutech.ac.jp

Yusuke Kumano
Kyushu Institute of Technology
Iizuka, Japan
kumano.yusuke521@mail.kyutech.jp

***Abstract*—The paper proposes a framework of systems for creating intelligent behaviors of imaginary creatures for humans, which is built upon a framework of swarm intelligence optimization algorithms. The paper assumes and models an imaginary slime mold, which is an amoeboid unicellular creature, as an imaginary creature in the framework concretely. In addition, the paper conducts basic evaluations of the framework. Various swarm intelligence optimization algorithms have been proposed so far, but there is a common feature among them. That is that a set of search points in a search space, which are called individuals, move around in the space according to algorithm specific rules using fitness values of the individuals, and the differences among algorithms are in algorithm specific rules. Therefore, under the use of one swarm intelligence optimization algorithm, that is, one particular set of rules, if a fitness function is varied, behaviors of individuals are also varied. Based on this fact, the proposed framework optimizes a parametric fitness function for individuals to behave intelligently for a human. In the basic evaluation of the concrete system, a fitness function of computer program which returns a fitness value calculated with a distribution of individuals is used instead of a human, and it is demonstrated that optimization of a parametric fitness function indeed yields desired behaviors of individuals.**

***Index Terms*—imaginary creature, true slime molds, swarm intelligence optimization, interactive evolutionary computation, human subjective evaluation**

I. Introduction

In the field of swarm intelligence optimization algorithms, algorithms inspired by a variety of creatures have been proposed [1] [2] [3]. Especially, there are many algorithms inspired by social living things such as ants and bees of social insects. Swarm intelligent optimization algorithms move a swarm of individuals corresponding to search points in a search space according to rules utilizing fitnesses of the individuals in order to obtain better individuals. That is to say, the problem-solving is equivalent to producing behaviors of individuals in swarm intelligence optimization algorithms.

Since swarm intelligent optimization algorithms are based on intelligence of swarms of individuals in nature, people should feel intelligence for behaviors of the artificial swarms of individuals. For example, in the ant colony optimization algorithm based on ant colony in nature, the strengths of pheromone trails on paths are increasing or decreasing according to local behaviors of individuals corresponding to ants, and the change in the strengths of pheromone trails on the paths can be visualized dynamically. Humans would feel some intelligence for the visualization.

Thus, swarm intelligent optimization algorithms move individuals (search points in a search space) according to given rules using fitness values to find better solutions under a given fitness function. However, if we change a fitness function, movement of individuals under the use of given fixed rules is changed. In this way, we might be able to produce intelligent behaviors of individuals that we have never seen before. The key point here is to produce behaviors by changing not rules but a fitness function (environment).

Therefore, based on the idea that behaviors of individuals can change by changing environments, we propose a framework of systems for creating intelligent behaviors of imaginary creatures for humans in the paper. In addition, we assume imaginary slime molds as imaginary creatures and model imaginary slime molds as dynamically changing graphs. A true slime mold is an amoeboid unicellular creature. Also, to confirm basic validity of the proposed framework, we conduct simulations in which a human who evaluates behaviors of imaginary creatures is replaced by a computer program. Through the simulations, we demonstrate that our expected behaviors of imaginary creatures, that is, imaginary slime molds, can be obtained by optimizing a fitness function that affects how to activate their behavior rules.

The remaining parts of the paper are organized as follows. Section II describes the related work. We propose the framework of systems for creating intelligent behaviors of imaginary creatures in Section III. Section IV shows a model of imaginary slime molds. Section V presents simulation results for validating the framework. Section V describes the conclusions and future work.

II. Related Work

In the paper, in order to create intelligent behaviors of imaginary creatures, rules for behaviors are fixed, but a fitness function returning fitness values that the rules utilize is optimized. Meanwhile, to produce desired behaviors of soft robots which can be regarded as imaginary creatures in some sense, rules for behaviors of the soft robots, that is, controllers of the robots, have been optimized [4]. Also, in the theoretical biology field, behaviors of some real creatures have been

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

described by rules, and the rules have been optimized by an optimization method [5].

Evolutionary art [6] is one type of art that generates better digital objects by adjusting parameter values of the generative system in an evolutionary manner. Humans usually become evaluation mechanisms for generated digital objects, and then, it can be said that the evolutionary art systems rely on interactive evolutionary computation, which is one type of evolutionary computation which uses a human as an evaluation system. The purpose of the proposed framework in the paper is not to generate artistic digital objects, but is similar to that of evolutionary art in a sense that novel objects for humans are created. In addition, systems for evolutionary art do not optimize fitness functions.

Also, regarding slime molds which are targets to be modeled in the paper, inspired by behaviors of true slime molds, a method for finding the shortest paths of maze has been proposed [7] [8]. In the method, an entire maze is represented as a true slime mold in a from of graph. The mechanism for finding the shortest path is represented by differential equations regarding materials' flow at nodes of its graph structure. Calculations of virtual materials' flow using the differential equations at each node bring the shortest path of the maze. However, this is not an swarm intelligence optimization approach in which behaviors of individuals are described by rules.

III. System Proposal

The overview of the proposed framework of systems is shown in Figure 1. The components of the proposed framework are (A) a model of imaginary creatures, (B) a fitness function which returns a fitness value determining how to activate behavior rules to each imaginary creature, and (C) an optimization method which optimizes targets using human subjective evaluations. A solution candidate for the optimization method of (C) is a fitness function of (B), which is a parametric functions that particular set of parameter values are given to. A fitness value given to an imaginary creature is assumed to be set from the position of the creature and the positional relationship with other ones. In the system, humans give higher evaluation to imaginary creatures which behave more intelligently for them.

We do not need to use a specific method as the optimization method of (C) using a human as an evaluation system, but one of them is interactive evolutionary computation. Interactive evolutionary computation is one type of evolutionary computation, in which a human or humans are used as its evaluation system. Humans as an evaluation system here give higher fitness to imaginary creatures which behave more intelligently for them as mentioned above, and evolutionary computation modifies parameter values of a fitness function, which is a solution candidate, according to the human subjective evaluations. This interaction between humans and evolutionary computation is expected to create intelligent behaviors of imaginary creatures.

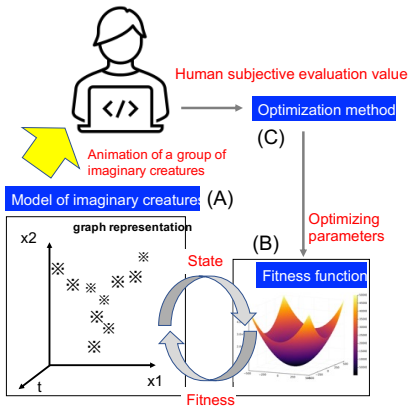


Fig. 1. Overview of the proposed system for creating intelligent behaviors of imaginary creatures.

IV. Model of Imaginary Slime Molds

A. Representation and Behavior Rules

A true slime mold is an amoeboid unicellular creature which moves, modifies itself, divides itself, and connects to others. An imaginary slime mold here is modeled as a directed graph whose nodes are placed in a two-dimensional Euclidian distance space. Therefore, nodes of a graph has a coordinate in the space. The number of nodes increases or decreases according to the rule mentioned later. Meanwhile, the number of directed edges is the same for every node and not changed.

Imaginary slime molds represented by graphs move in the space as the result of changing positions of their nodes, and modify themselves as the result of changing target nodes for which their edges are generated and increasing or decreasing their nodes. The rules for movement and modification of imaginary slime molds use a fitness value of each node in a graph or a fitness value of the entire graph. A fitness function used here and ways to calculate fitness values are explained in the next section. In the explanation of the algorithm below, it is assumed that fitness values have been assigned to both each node and the entire graph. However, the paper does not consider rules for connecting multiple imaginary slime molds and dividing an imaginary slime molds into multiple ones.

Algorithm 1 shows the algorithm for movement of imaginary slime molds. Algorithm 2 shows the algorithm for modification of nodes for which directed edges of imaginary slime molds are generated, that is, modification of edges. Algorithm 3 shows the algorithm of increasing or decreasing nodes of imaginary slime molds.

First, in Algorithm 1, each node in a graph representing an imaginary slime mold becomes a start node for random walk, and a random walker produces new positions of nodes while moving between nodes a fixed number of times. The algorithm causes change in node positions, and the result of the change is regarded as movement of imaginary slime mold.

Specifically, in the third line of Algorithm 1, the j-th node of the i-th imaginary slime mold is obtained. In the fourth to

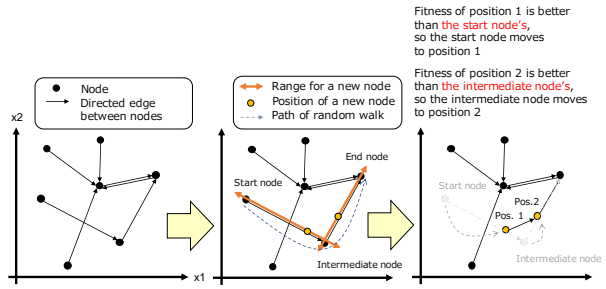


Fig. 2. Example of movement of imaginary slime mold.

the eleventh lines, a random walker moves between nodes H_M times and also produces H_M new nodes. In the fifth line, the destination node, enp, from the present node, snp, is chosen. In the sixth line, although the details are not described there, a line segment connecting the nodes snp and enp with the length of ℓ is enlarged α times by adding $\alpha/2$ times length of the line segment connecting the nodes snp and enp to those nodes, and then a tentative node, nnp, is randomly generated on the α times enlarged line segment. This α is a parameter of the algorithm. In the seventh to the ninth lines, if a fitness value of the generated node, nnp, is better than the node snp's, the node snp is replaced by the node nnp. In the tenth line, the next start node for movement between nodes is set to be the node enp. The example of movement of imaginary slime mold is shown in Figure 2.

Algorithm 1 Move nodes of imaginary slime molds.

Require: P: population of imaginary slime molds, S: size of P, N(i): # of nodes, H_M: # of allowed hops
Ensure: P: population
1: **for** $i = 1$ to S **do**
2: **for** $j = 1$ to N(i) **do**
3: snp ← GetNode(i, j)
4: **for** $k = 1$ to H_M **do**
5: enp ← RandomChooseNextNode(snp)
6: nnp ← GenerateNewNode(snp, enp)
7: **if** Fitness(nnp) is better than Fitness(snp) **then**
8: ReplaceNode(i, j, nnp)
9: **end if**
10: snp ← enp
11: **end for**
12: **end for**
13: **end for**

Next, in Algorithm 2, if nodes in a graph representing an imaginary slime mold meet some condition mentioned later, they become a start node for random walk, and a random walker moves between nodes a fixed number of times. Then, one node meeting another condition is obtained from among nodes that the random walker reached. Finally, the start node changes its directed edge's destination node to the obtained one. The algorithm causes change in node positions, and the result of the change is regarded as modification of imaginary slime mold. Change in destination nodes of directed edges causes change in a topology of the graph.

Specifically, in the second to the fourth lines of Algorithm 2, a cycle time of natural number, which is called an edge modification cycle, is given to each node. The maximum value of an edge modification cycle is C. If the fitness value of a node of focus, f, is better than the average fitness value over all nodes', f_a, the cycle time of the node is randomly chosen from $[1, C/2]$, where C is an even number and more than or equal to 2. If it is worse than or equal to the average, the cycle time of the node is randomly chosen from $[C/2+1, C]$. In the fifth line, the time t proceeds by one from 1 to T ($C \leqq T$). In the sixth and the seventh lines, at each time t, every node whose modification cycle time c meets $t \mod c \equiv 0$ is selected as target one that changes its edge. In the eighth to the sixteenth lines, while a walker is randomly moving between nodes the given H_L times from every target node to modify its edge, it finds a node with the best fitness value among all nodes at which it arrived. In the twenty-first to the twenty-fifth lines, the target node finds a node with the worst fitness value among nodes to which it has made directed edges, and then compares that best fitness value among the arrived nodes' and the worst one among the connected nodes', and the if that best fitness value is better than that worst one, it deletes the edge to the node with that worst fitness value and makes a new edge to the node with that best one. However, if the target node has already made A_e or larger edges to the taken one (the best one), it does nothing. Modification of imaginary slime molds occurs through the edge modification. An example of the edge modification of a node is shown in Figure 3.

Algorithm 2 Change edges of imaginary slime molds.

Require: P: population of imaginary slime molds, S: size of P, N(i): # of nodes, H_L: # of allowed hops
Ensure: P: population
1: **for** $i = 1$ to S **do**
2: **for** $j = 1$ to N(i) **do**
3: c(j) ← AssignCycle(j)
4: **end for**
5: **for** $t = 1$ to T **do**
6: **for** $j = 1$ to N(i) **do**
7: **if** $t \mod$ c(j) $\equiv 0$ **then**
8: snp ← GetNode(i, j)
9: **for** $k = 1$ to H_L **do**
10: enp ← RandomChooseNextNode(snp)
11: **if** $k = 1$ **then**
12: bn ← enp
13: bf ← Fitness(enp)
14: **else** $k > 1$ & Fitness(enp) is better than bf
15: bn ← enp
16: bf ← Fitness(enp)
17: **end if**
18: snp ← enp
19: **end for**
20: **end if**
21: **if** WorstNodeFitness(j) is worse than bf **then**
22: wn ← GetWorstLinkedNode(j)
23: DeleteWorstLink(j, wn)
24: MakeLink(j, bn)
25: **end if**
26: **end for**
27: **end for**
28: **end for**

Also, in Algorithm 3, a graph of imaginary slime mold which has better fitness value than others increases nodes and one which has worse fitness value decreases nodes. The total number of edges of all imaginary slime molds is a constant and not changed.

Specifically, in the second and the third lines of Algorithm 3, two graphs (imaginary slime molds) are randomly selected

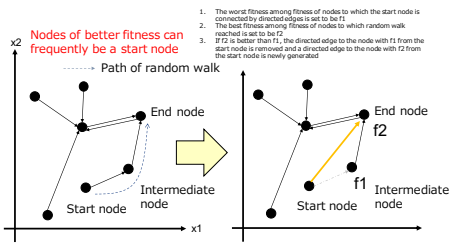


Fig. 3. Modification of an edge of graph representing an imaginary slime mold.

from all. In the fourth to the ninth lines, two selected graphs are compared in terms of fitness value, and then, better one increases nodes by one and worse one decreases nodes by one. However, in the case that the number of nodes of a graph becomes more than N_{max} by this procedure or in the case that the number becomes less than N_{min} by this procedure, the node increase or decrease does not occur on the graph. In order to increase nodes of graph by one, first, a new node first randomly decides a position in the two dimensional Euclidean space, and then, the new node at the position makes E directed edges to randomly chosen nodes in the graph. Meanwhile, in order to decrease nodes of graph by one, the node with the worst fitness value in the graph and the edges that the node makes to others are deleted. Every node which had made edges to the deleted node removes those edges and randomly selects nodes equal to the number of removed edges and makes edges to the selected nodes.

Algorithm 3 Increase or decrease nodes of imaginary slime molds.

Require: P: population of imaginary slime molds, S: size of P
Ensure: P: population
1: **for** $i = 1$ to S **do**
2: n1 ← RandomChooseGraph(P)
3: n2 ← RandomChooseGraph(P)
4: **if** GFitness(n1) is better than GFitness(n2) **then**
5: IncreaseNode(n1)
6: DecreaseNode(n2)
7: **else** GFitness(n2) is better than GFitness(n1)
8: IncreaseNode(n2)
9: DecreaseNode(n1)
10: **end if**
11: **end for**

B. Fitness Function as Optimization Target

Although a fitness function is not directly related to the model of imaginary slime molds, a fitness function as the optimization target is explained here because the a fitness value used by behavior rules of imaginary slime molds is given by this fitness function. The fitness function gives a fitness value to each node of a graph placed in the two dimensional Euclidian space, and is a real valued function on that space, $F(x, y)$. When the coordinate of a node in the space is (x_*, y_*), the fitness value of the node is $F(x_*, y_*)$. Larger fitness values are better.

Specifically, the fitness function is represented by Equation (1). The function is a superposition of N_f Gaussian functions. The value of N_f is fixed, and the values of $A_i, \mu_{xi}, \mu_{yi}, \sigma_{xi}, and \sigma_{yi}$ $(i = 1, 2, \ldots, N_f)$ are the optimization targets. However, if there are other N_D nodes within a small distance of D_{th} from a node of focus no matter which graphs the N_D nodes are included in, the fitness value of the node of focus is obtained by dividing the value of Equation (1) by N_D. This mechanism reduces fitness values of nodes which are close to each other and makes the nodes easier to be deleted, so that congestion of nodes is expected to be avoided. A fitness value of node is calculated by the function of Fitness() in Algorithms 1 and 2.

$$F(x, y) = \sum_{i=1}^{N_f} A_i \exp\left(-\frac{(x - \mu_{xi})^2 + (y - \mu_{yi})^2}{\sigma_{xi}^2 + \sigma_{yi}^2}\right) \quad (1)$$

Meanwhile, a fitness value of graph itself is given as the average fitness values over all nodes'. A fitness value of graph is calculated by GFitness() in Algorithm 3.

Algorithm 4 Behaviors of imaginary slime molds.

Require: P: population of imaginary slime molds, MaxGEN: maximum generations
Ensure: P: population
1: $P \leftarrow Initialization()$
2: **for** $generation = 1$ to MaxGEN **do**
3: $P \leftarrow FitnessAssignment(P)$
4: $P \leftarrow Movement(P)$
5: $P \leftarrow Reformation(P)$
6: $P \leftarrow NodeIncreaseDecrease(P)$
7: **end for**

C. Algorithm for Behaviors of Imaginary Slime Molds

The algorithm for behaviors of imaginary slime molds is shown in Algorithm 4. This algorithm consists of initialization of imaginary slime molds (Initialization), fitness calculation for imaginary slime molds mentioned above (FitnessAssignment in Section IV-B), movement of imaginary slime molds (Movement in Algorithm 1), graph topology modification (Reformation in Algorithm 2), and increase or decrease of nodes in graphs (NodeIncreaseDecrease in Algorithm 3).

In the first line of Algorithm 4, a population of imaginary slime molds, P, are initialized. The population size is S. All imaginary slime molds represented by graphs have the same number of nodes, N_{sm}, initially, and all nodes of each imaginary slime mold are randomly placed in the two dimensional Euclidian space. The number of nodes can increase or decrease during the algorithm run. The number of directed edges that every node makes is E and this is not changed during the algorithm run. The procedures after that are as mentioned above. This algorithm is executed until MaxGEN generation in the second line. One generation is equal to one execution of the fitness calculation, the movement, the graph topology modification, and the node increase or decrease.

V. Simulations of Behaviors of Imaginary Slime Molds

A. Purpose

The framework of systems proposed in Section III uses an optimization method using a human as an evaluation

mechanism such as interactive evolutionary computation to produce intelligent behaviors of imaginary creatures. A human as the evaluation mechanism is assumed to watch an animation of behaviors of imaginary slime molds and give higher fitness values to behaviors to which the human feels more intelligence.

However, in the paper, as basic validation of the proposed framework, we just confirm if desired behaviors of imaginary creatures are obtained through optimization of a fitness function that provides fitness values for the rules of behaviors. For this purpose, we use not a human but a computer program as an evaluation mechanism for behaviors of imaginary creatures. The computer program extracts features from distributions of a population of imaginary slime molds (graphs) in the two dimensional Euclidean space and uses them as the evaluation values.

B. Configurations

The evaluation mechanisms for behaviors of imaginary slime molds are described in the following section. Here the other configurations, that is, the parameter values of the algorithm for behaviors of imaginary slime molds, the parameter ranges of the fitness function, the settings of the optimization algorithm used for optimizing the fitness function. A genetic algorithm is used as this optimization algorithm.

First, the parameter values of the algorithm for behaviors of imaginary slime molds are shown in Table I. Here the ranges of the two dimensional Euclidean search space in which imaginary slime molds can exist are $-100 \leqq x \leqq 100$ and $-100 \leqq y \leqq 100$.

TABLE I
PARAMETER VALUES OF THE ALGORITHM.

Parameter	Brief explanation	Value
S	population size	4
N_{sm}	initial # of nodes	30
N_{max}	max # of nodes	45
N_{min}	min # of nodes	15
E	# of edges	2
A_e	# of overlapped edges	1
α	expansion rate of edge	0.2
H_M	# of hops for movement	2
C	max edge modification cycle	10
T	time duration for edge modification	20
H_L	# of hops for edge modification	2
D_{th}	crowded distance	10
N_D	size of crowded individuals	1
MaxGEN	max generations	10

Also, the parameter ranges of the fitness function are as follows. The number of Gaussian functions, N_f, is 20, which is fixed. A_i, μ_{xi}, μ_{yi}, σ_{xi}, and σ_{yi} are all in $[-100, 100]$, which are the parameters of Gaussian function.

The genetic algorithm (GA) used for optimizing the fitness function takes real-valued representation of solutions, uses Minimal Generation Gap (MGG) [9] as a generation gap model, uses BLX-α (α=0.36) [10] as a crossover operator, and uses uniform random number based mutation as a mutation operator. The crossover and the mutation rates are 1.0 and 0.1, respectively. The population size of the GA is 200 and the stop condition of algorithm run is to reach 2×10^4 function evaluations.

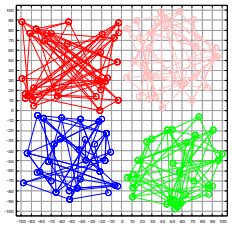


Fig. 4. Initial distribution of imaginary slime molds for every GA individual in every simulation scenario.

C. Simulation Scenarios

As mentioned above, the purpose of the evaluations here is to confirm that optimization of the fitness function using a computer program as the evaluation mechanism for desired behaviors of imaginary slime molds indeed yields desired ones. However, the fitness function used here can be much different from a human as a fitness function. Therefore, the validation here can be positioned as the basic validation.

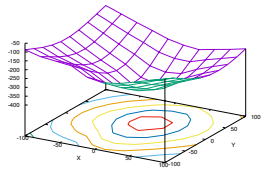
Two kinds of simulation scenarios are used, which have different fitness functions. Considering an unit of movements, topology modification, increase or decrease of nodes, and fitness calculation to be a generation, a generation is iterated MaxGEN times to produce behaviors of imaginary slime molds. The two types of fitness functions use features obtained from the final distribution of imaginary slime molds right after the MaxGEN-th generation. The obtained feature is the average distance between nodes in all imaginary slime molds.

Then, the first scenario is that the evaluation function for imaginary slime molds considers larger average distances to be better. The second scenario is that the evaluation function considers smaller average distances to be better. For every GA individual (every set of parameter values of the fitness function) in every simulation scenario, the identical initial distribution of imaginary slime molds is used. The distribution is shown in Figure 4.

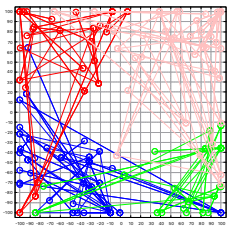
The expected behaviors of imaginary slime molds in the two scenarios are as follows.

In the first scenario, it should occur that nodes are finally located as much apart as possible in the two dimensional square space to maximize the average distance between nodes. Since the maximum distance direction in a square area is diagonal direction, regions of good fitness values are expected to be produced around both ends of two diagonal segments in the square area, and as a result, nodes of four imaginary slime molds are expected to be located around the ends of the two diagonal segments.

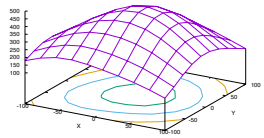
In the second scenario, it should occur that nodes are finally located as much close as possible in the two dimensional square space to minimize the average distance between nodes. Therefore, regions of good fitness values are expected to be produced around one point in the square area, and as a result,



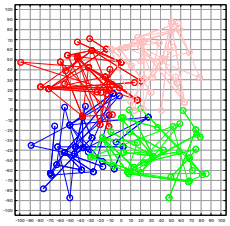
(a) Scenario 1: Best fitness function.



(b) Scenario 1: Final distribution.



(c) Scenario 2: Best fitness function.



(d) Scenario 2: Final distribution.

Fig. 5. Best fitness function obtained by one run of GA and final distribution of imaginary slime molds under the use of the best fitness function.

nodes of four imaginary slime molds are expected to be located around the point.

If behaviors of imaginary slime molds became as expected above, although it is basic demonstration, we consider that our desired behaviors could be obtained through the optimization of the fitness function.

D. Results and Discussions

Figures 5 (a)(c) show the best fitness functions obtained by one run of GA. Also, Figures 5 (b)(d) show the final distributions of imaginary slime molds under the use of the best fitness functions.

It can be observed from Figure 5(a) that the fitness function in which good fitness regions exist around the four corners of the square area and bad region exits around the center was obtained for the scenario 1. In addition, it can be observed from Figure 5(b) that nodes of the four imaginary slime molds were finally distributed around the four corners of good fitness regions. Therefore, we can conclude that the result was as expected.

It can be observed from Figure 5(b) that the fitness function in which a good fitness region exits around the center of the square area was obtained for the scenario 2. If the four imaginary slime molds can gather around any one point, the average distance between nodes can be smaller. However, the center of the square area would be the smallest moving distance on average over all nodes, so that around the center would become a good fitness region. In addition, it can be observed from Figure 5(d) that nodes of the four imaginary slime molds were finally distributed around the center. Although the gathering point was not able to be predicted, we can conclude that the result was as expected in Section IV-C.

Thus, although it is basic demonstration, we conclude that our desired behaviors could be obtained through the optimization of the fitness function.

VI. Concluding Remarks and Future Work

In the paper, we proposed the framework of systems for creating intelligent behaviors of imaginary creatures for humans, and developed the concrete system within the framework, and conducted the basic validation for the system. The proposed framework regards swarm intelligent optimization algorithms conceptually as a framework of systems that produce behaviors of imaginary creatures under a given fitness function, and added an optimization method using a human as an evaluation mechanism to the framework of the swarm intelligent optimization algorithms. The optimization method optimizes a fitness function for imaginary creatures. In the basic validation, we assumed and modeled imaginary slime molds as imaginary creatures, and used not a human but a computer program as an evaluation mechanism for behaviors of imaginary slime molds. The results demonstrated that desired behaviors of imaginary slime molds could be produced through optimization of the fitness function.

In the future work, we will develop a web application for creating intelligent behaviors of imaginary slime molds for humans and observe and analyze what behaviors of imaginary slime mold many humans feel intelligence to.

Acknowledgement

This work is supported by the Japan Society for the Promotion of Science through a Grant-in-Aid for Transformative Research Areas (A) (Publicly Offered Research) (21A402).

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942–1948.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," *IEEE Trans. on System, Man, and Cybernetics-Part B*, vol. 26, no. 2, pp. 29–41, 1996.

[3] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," in *Technical Report-TR06*, 2006, pp. 1–10.

[4] J. S. Bhatia, H. Jackson, Y. Tian, J. Xu, and W. Matusik, "Evolution gym: A large-scale benchmark for evolving soft robots," 2022.

[5] T. Krink and F. Vollrath, "Analysing spider web-building behaviour with rule-based simulations and genetic algorithms," *Journal of Theoretical Biology*, vol. 185, no. 3, pp. 321–331, 1997.

[6] J. Romero and P. Machado, Eds., *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, ser. Natural Computing Series. Springer Berlin Heidelberg, November 2007.

[7] T. Nakagaki, H. Yamada, and A. Toth, "Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 6803, p. 470, 2000.

[8] A. Tero, R. Kobayashi, and T. Nakagaki, "A mathematical model for adaptive transport network in path finding by true slime mold," *Journal of Theoretical Biology*, vol. 244, no. 4, pp. 553–564, 2007.

[9] H. Satoh, M. Yamamura, and S. Kobayashi, "Minimal generation gap model for GAs considering both exploration and expolation," in *Proceedings of the International Conference on Fuzzy Systems, Neural Networks and Soft Computing (Iizuka'96)*, 1996, pp. 494–497.

[10] L. J. Eshelman and J. D. Schaffer, "Real coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, 1993, pp. 187–202.