Amoeba-Based Emergent Computing: Combinatorial Optimization and Autonomous Meta-Problem Solving

MASASHI AONO\textsuperscript{1}, MASAHIKO HARA\textsuperscript{1}, KAZUYUKI AIHARA\textsuperscript{2} AND TOSHINORI MUNAKATA\textsuperscript{3}

\textsuperscript{1}Frontier Research System, RIKEN (The Institute of Physical and Chemical Research), Wako, Saitama 351-0198, Japan
E-mail: masashi.aono@riken.jp

\textsuperscript{2}Institute of Industrial Science, The University of Tokyo, Meguro, Tokyo 153-8505, Japan, and ERATO Aihara Complexity Modelling Project, JST, Shibuya-ku, Tokyo 151-0064, Japan
E-mail: aihara@sat.t.u-tokyo.ac.jp

\textsuperscript{3}Computer and Information Science Department, Cleveland State University, Cleveland, OH 44115, USA
E-mail: t.munakata@csuohio.edu

Received: December 26, 2007. Accepted: February 15, 2008.

Here we demonstrate a computing system employing an amoeba of a true slime mold \textit{Physarum} known as a model organism for studying cellular information processing. The system works as a neurocomputer that exhibits high optimization capability in solving various problems including the traveling salesman problem. Additionally, we present a new technique that we call “autonomous meta-problem solving.” In this approach, our system not only can solve a given problem but also can find new problems and then determine solutions by exploiting the amoeba’s unique searching ability and spontaneous behavior.

\textit{Keywords:} \textit{Physarum}, actomyosin, molecular computing, self-organization, neural network, optimization, fluctuations, spontaneous destabilization, chaos, meta-problem solving.

1 INTRODUCTION

1.1 Self-organizing molecular computing

There has been growing interest in unconventional computing schemes to implement information processing on a substrate of an excitable or oscillatory
medium composed of an assembled mass of interacting molecular-scale elements [1, 2]. The molecular elements capable of self-organizing behavior, for example, chemicals [3, 4], nanoparticles [5], and biomolecules [6], are expected to perform massively parallel (concurrent) computing without introducing elaborate microscopic control techniques. Moreover, if an appropriate biological system could be employed as a computing substrate, it may be possible to exploit the substrate’s self-organizing behavior for performing autonomous functions, such as self-modification and self-reparation of a program embedded in software and/or hardware [7], that would be advantageous when incorporated in autonomous agents to cope with unexpected situations.

With these expectations, we employ an amoeboid organism Physarum which is regarded as a kind of oscillatory medium [8–10] and is capable of sophisticated computing [11, 12]. Aono and Gunji proposed a computing system employing the amoeba to execute nonclassical cellular automata by concurrent operations of the amoeba’s photosensitive branches induced by a realtime optical feedback system [13]. Based on this proposal, some of the authors constructed the amoeba-based computing systems with the optical feedback system to implement recurrent neural network algorithms and demonstrated that the systems work as associative memory [14] and as constraint satisfaction problem solvers [15–17]. So far, employing the amoeba, some other researchers presented a number of computing devices/systems including logic operators [18], robot controllers [19], and unconventional problem solvers [20, 21].

In this paper, we propose two computing schemes employing the amoeba and show their specific embodiments in two kinds of experiments. The first one is for standard problem solving for combinatorial optimization. The second one, meta-problem solving, is an extended version of the first one and is oriented toward the above-mentioned autonomous functions.

1.2 True slime mold amoeba

A plasmodium of the true slime mold Physarum polycephalum is a huge unicellular multinucleated amoeboid organism (Fig. 1(A)). When we place the amoeba in a stellate container structure put on an agar plate containing no nutrient (Fig. 1(B)), the amoeba comes to have multiple branches and changes its shape by expanding or shrinking the branches concurrently, where its total volume is almost constant during the experiment.1

The amoeba’s body structure is simple and homogeneous. Figure 1(C) schematically shows the structure of an amoeba with three expanding branches. An individual amoeba has a single gel layer (a sort of cellular membrane) to encapsulate the intracellular sol, and has no other highly differentiated structures. Because numerous cell nuclei are distributed throughout

---

1The amoeba stores nutrients fed before the experiment as an internal energy source and survives for up to about a week solely by absorbing moisture from the agar plate containing no nutrient.
Combinatorial Optimization and Autonomous Meta-Problem Solving

FIGURE 1
(A) An individual unicellular amoeba (scale bar = 7 mm). (B) Au-coated plastic container structure on agar plate without nutrients (scale bar = 7 mm). The amoeba acts only inside the container where agar is exposed, because of its aversion to the metal surface. (C) Schematic illustration of the amoeba’s body structure. (D) Schematic illustration of the contraction-relaxation oscillation of actomyosin systems.

the homogeneous body of an individual amoeba, a part of the amoeba divided from the individual survives as another self-sustainable individual.

The gel layer is formed by masses of actomyosin systems (fibrous proteins contained in muscles) that are capable of taking contracting or relaxing states, as schematically illustrated in Fig. 1(D). These actomyosin systems can be regarded as the amoeba’s processing elements [22,23]. The interacting actomyosin population produces oscillatory behavior in which the gel layer alternately contracts and relaxes at a period of $\sim 1\sim 2$ min, and forms spatiotemporal oscillation patterns observed as the repeated increase and decrease of the vertical body thickness.

Depending on the spatiotemporal pattern, the gel layer’s contraction tension changes from site to site. The sol is led to stream horizontally (velocity = $\sim 1$ mm/sec) along the pressure difference (gradient) produced by the local contraction tension difference. We can observe that the direction of the sol streaming is repeatedly reversed in response to the gel layer’s oscillation. The repetition of the sol efflux-influx for a branch provides the driving force of the branch’s expansion or shrinkage. Thus, it is possible to think that the amoeba makes decisions about changing its horizontal shape by the self-organization of the spatiotemporal pattern in its vertical thickness oscillation [24].

Despite the absence of a central nervous system, the amoeba exhibits sophisticated computational capabilities to optimize its behavior under a given environmental condition. As an example, Nakagaki and coauthors showed that the amoeba is capable of searching for a solution to a maze [11]. Inside a container structure such as the maze, the amoeba changes its shape into a string-like configuration that is the shortest connection path between two nutrient sources placed at the entrance and exit of the maze. This shape changing can be regarded as an optimal behavioral selection for the amoeba’s
survival, because it maximizes the nutrient absorption efficiency under this experimental condition [12]. A mathematical model to reproduce the optimized behavior has been presented [20].

1.3 Why implement neurocomputing with an amoeba?

We construct a computing system in which its state is represented by a shape of the amoeba in the stellate container structure. The amoeba’s shape changing – expansion or shrinkage of multiple branches – corresponds to a transition between states. Because the amoeba's branch exhibits a photoavoidance response, the shape changing can be induced by applying optical stimulation with the projection of an appropriate image pattern. We introduce an optical feedback that updates the illumination pattern according to a certain rule in response to the amoeba’s shape changing. In this paper, adopting a recurrent neural network algorithm as the rule for updating the illumination pattern, we implement neurocomputing for combinatorial optimization.

One might wonder why the amoeba’s shape changing and neurocomputing, two seemingly unrelated matters, are connected. The reasons are as follows.

1. As mentioned above, the amoeba is capable of optimizing its shape under a given environmental condition. We estimated that this optimization capability would be exploited for solving diverse application problems, if each problem could be properly translated into an environmental condition provided for the amoeba.

2. There is a one-to-one mapping between the amoeba’s shape and a state of a neural network model, where we identify a state of a single neuron as a dimension (area or volume) of the amoeba’s branch.

3. Since extensive studies have been conducted in the field of neurocomputing, many useful techniques for implementing various types of applications including combinatorial optimization have already been developed in terms of neural network algorithms. According to these algorithms, many application problems are translated into environmental conditions provided for the amoeba.

4. We suppose that the fluctuations and instability in the amoeba’s oscillatory dynamics would be positively exploited for exploring a broader search space in terms of a recurrent neural network algorithm, as well as some other metaheuristics for combinatorial optimization.

We exploit all the above facts together. That is, given a shape of the amoeba, we encode it as a state of the neural network model. Applying the recurrent neural network algorithm, we determine an illumination pattern and feed it back to the amoeba to induce its shape changing. Iterating these procedures, the optical feedback translates a given problem into a spatiotemporally changed illumination condition that leads the amoeba to recognize and meet the constraints
in the problem by receiving optical stimulation. When the amoeba reaches a configuration in which states of all neurons become unchanged, we regard the stable configuration as an output result of the computation.

2 METHODS

2.1 Traveling salesman problem (TSP)
The traveling salesman problem (TSP) is stated as follows: Given a map of $N$ cities defining the travel distances from any city to any other city, search for the shortest circular trip route for a salesman who needs to minimize the total travel distance when visiting each city exactly once and returning to the starting city. An example of a map of four cities ($A, B, C,$ and $D$) is given in Fig. 2(A). In this case, there exist $4! = 24$ possible circular routes.

FIGURE 2
(A) The map of four cities used to solve the traveling salesman problem. The edge lengths do not precisely reflect the distances. (B) Left: The amoeba's configuration representing the circular trip route $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$. White rectangles indicate illuminated regions. Right: Optical feedback system. For transmitted light imaging using a video camera (VC), a surface light source (LS) beneath the sample amoeba (SM) was employed to emit low-intensity light. The recorded image was digitally processed using a PC to update the high-intensity monochrome image for illumination with a projector (PJ). See Appendix for more details on the experimental setups. (C) Visual simulation of how the illumination pattern inhibits the amoeba's invalid route selection. Left: Representation of condition 1, by which a once-visited city cannot be revisited. Right: Representation of condition 2, prohibiting simultaneous visits to more than one city. (D) Visual simulation of how condition 3, reflecting the difference between the shorter and longer routes, is represented in the transition of the illumination pattern. Left: The shorter route $B \rightarrow C$ is destabilized (inhibited) when the perturbing (conflicting) branch $A_2$ expands to $x_{A_2}(t'') = 0.3175$ (bottom). Right: The longer route $B \rightarrow D$ is destabilized more easily by the smaller perturbation $x_{A_2}(t') = 0.235$ (middle).
Each circular route takes the total distance 12, 20, or 24. For example, the circular route $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$ is one of the shortest (optimal) solutions, whereas $A \rightarrow C \rightarrow B \rightarrow D \rightarrow A$ and $A \rightarrow C \rightarrow D \rightarrow B \rightarrow A$ are the second-shortest and the longest solutions, respectively.

TSP is a particularly hard problem among typical combinatorial optimization problems [25]. Because the number of all possible circular routes $N!$ runs into astronomical numbers when the number of cities $N$ becomes larger, often one has to give up the search for the exact optimal solution. Therefore, various approximation algorithms for quickly obtaining good solutions have been proposed so far. Genetic algorithm [26], ant colony optimization [27], simulated annealing, and the recurrent neural network algorithm [28] we adopt here, are examples of popular approximation algorithms known as metaheuristics.

2.2 Neural network representation of route selection

According to the recurrent neural network algorithm proposed by Hopfield and Tank [29], the $N$-city TSP can be solved with $N \times N$ neurons. To implement the four-city TSP solution, we fabricated a container structure having 16 radial lanes (grooves) shown in Fig. 1(B), where each lane is called a “neuron” to be distinguished from the amoeba’s “branch” expanding in the lane. Each neuron is labeled with $i \in \{Pn \mid P \in \{A, B, C, D\}, n \in \{1, 2, 3, 4\}\}$ to indicate the city name $P$ and its visiting order $n$. When the amoeba sufficiently expands its branch in the neuron $Pn$, it indicates that $P$ was selected as the $n$th visiting city. For example, Fig. 2(B) (left) represents the selection of the circular route $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$.

2.3 State transition induced by optical stimulation

For each neuron $i$ at time $t$, the state $x_i(t) \in [0.0, 1.0]$ is defined as the fraction of the area occupied by the amoeba’s branch inside the corresponding neuron (i.e., $x_i = \text{area of branch} / \text{area of the entire region of neuron} i$). At each time step, the numerical value of each state is calculated by means of digital image processing of a transmitted light image, as shown in Fig. 2(B) (right). All numerical calculations are performed at double precision (16 decimals).

When light illumination for neuron $i$ is turned on, we represent this status as $y_i(t) = 1$, otherwise $y_i(t) = 0$ (turned off). The amoeba’s branch inherently expands and tends to occupy the entire region of the corresponding neuron in principle when $y_i = 0$. Namely, if no illumination were applied, all sixteen branches would fully expand to fill all of the neuron regions. Note that the state $x_i(t)$ changes in a non-monotonous manner because the amoeba’s branch alternates between small-scale expansion and shrinkage at each period of the oscillation (Fig. 1(C)). Only when the accumulation of several periods of the small-scale changes in $x_i(t)$ becomes a positive value, the amoeba’s branch comes to perform the large-scale expansion (growth) to fill the entire neuron, where the growth velocity is at most about 1 cm/h.
On the other hand, the amoeba’s branch exhibits a photoavoidance response. It is said that a branch is shrunk by optical stimulation because the light-induced contraction enhancement of the gel layer intensifies the sol efflux (extrusion) from the stimulated part [30]. Accordingly, when the neuron \( i \) is illuminated as \( y_i = 1 \), the large-scale increase of the state \( x_i \) can be inhibited and its large-scale decrease can be promoted.

2.4 Neural network algorithm in optical feedback

It is possible to input an arbitrary initial configuration \( \langle x_{A1}(0), x_{A2}(0), \ldots, x_{D4}(0) \rangle \), because the amoeba’s shape is freely deformable. In our experiments, the time evolution starts from the initial configuration in which all neurons are set to \( x_i(0) = 0 \).

In our optical feedback system, each neuron’s illumination status \( y_i \in \{0, 1\} \) is updated automatically in accordance with the following neural network dynamics at every 6 sec:

\[
\begin{align*}
\sigma(x; a, b, c) &= a/(1 + \exp\{-b(x-c)\}) \quad \text{(2)} \\
f(X) &= \begin{cases} 
0 & \text{if } X < \theta \\
1 & \text{otherwise} 
\end{cases} \quad \text{(3)} \\
w_{ij} &= \begin{cases} 
-\alpha & \text{if } i = Pn, j = Pm, \text{ and } n \neq m \\
-\beta & \text{if } i = Pn, j = Qn, \text{ and } P \neq Q \\
-\gamma \text{dist}(P, Q) & \text{if } i = Pn, j = Qm, P \neq Q \text{ and } |n-m| = 1 \\
0 & \text{otherwise} 
\end{cases} \quad \text{(4)}
\end{align*}
\]

We designed these dynamics to fit our experimental setups by making some modifications to the original dynamics given by Hopfield and Tank [29]. We newly introduced the sigmoid function \( \sigma \) to enhance the adjustability of the system’s sensitivity, where its parameters are set as \( a = 1, b = 35, \) and \( c = 0.25 \) in the experiments. The step function \( f \) is defined with a negative threshold \( \theta = -0.5 \).

2.5 Representing TSP by optical feedback as inhibitory coupling

As shown in Eqs. (1) and (4), each neuron \( i \) is connected to every neuron \( j \) with the coupling weight \( w_{ij} \) defined to be nonpositive and symmetric (i.e., \( w_{ij} = w_{ji} \leq 0 \)). A negative weight \( w_{ij}(=w_{ji} < 0) \), an inhibitory coupling weight, creates a conflict (i.e., exclusive relationship) between the neurons \( i \) and \( j \) in which the increase of \( x_j \) results in the decrease of \( x_i \), and vice versa. That is, when the amoeba’s branch in neuron \( j \) expands to take a certain threshold value \( x_j(t) \), it becomes a trigger for illuminating neuron \( i \) as \( y_i(t + \Delta t) = 1 \). If the absolute value of the weight \( |w_{ij}| \) becomes larger, the
threshold (minimum) value of \( x_j(t) \) required for triggering \( y_i(t + \Delta t) = 1 \) becomes smaller.

The inhibitory coupling weights are introduced to establish the following three constraints that should be satisfied when representing a solution of TSP.

1. Prohibition of revisiting a once-visited city.

   If city \( P \) is the \( n \)th city visited, \( P \) cannot be the \( m(\neq n) \)th city visited, either before or after that. This constraint is represented by the inhibitory coupling weight \( w_{ij} = -\alpha \) between the neuron \( i = Pn \) and \( j = Pm \).

2. Prohibition of simultaneous visits to more than one city.

   If city \( P \) is the \( n \)th city visited, no other city \( Q(\neq P) \) can be visited at the same time \( n \). The inhibitory coupling \( w_{ij} = -\beta \) between neurons \( i = Pn \) and \( j = Qn \) represents this constraint.

3. Reflection of travel distance between cities.

   If city \( P \) is the \( n \)th city visited, and right before or right after that city \( Q(\neq P) \) is the \( m(= n \pm 1) \)th city visited, the cost of traveling the edge \( P \rightarrow Q \) (or \( Q \rightarrow P \)) proportionally reflects the distance between \( P \) and \( Q \) written as \( dst(P, Q)(= dst(Q, P)) \). This constraint is reflected in the inhibitory coupling \( w_{ij} = -\gamma dst(P, Q) \) between neurons \( i = Pn \) and \( j = Qm \), where if \( n = 4 \) then \( m = n + 1 =: 1 \), and if \( n = 1 \) then \( m = n - 1 =: 4 \).

Let us see separately how each of the above constraints is represented with the transition of the illumination pattern. In Fig. 2(C) (left), for convenience in explaining constraint 1 exclusively, we set the parameters as \( \alpha = 0.5 \) and \( \beta = \gamma = 0 \). When the state of the neuron \( A1 \) increases beyond the threshold \( x_{A1} = 0.4825 \) to trigger the illuminations, the neurons \( A2, A3, \) and \( A4 \) conflicting with constraint 1 are inhibited by the illuminations. Figure 2(C) (right) shows the case in which the parameters are set as \( \beta = 0.5 \) and \( \alpha = \gamma = 0 \) for explaining constraint 2. When neuron \( A1 \) exceeds the threshold \( x_{A1} = 0.4825 \), the conflicting neurons \( B1, C1, \) and \( D1 \) are inhibited. Any configuration representing a circular route by satisfying both constraints 1 and 2 gives a valid solution of TSP.

In the actual experimental condition, the parameters are set as \( \alpha = 0.5, \beta = 0.5, \) and \( \gamma = 0.025 \). Note that we set the parameters so that the amoeba can never be illuminated when all its expanding branches select a valid combination of edges satisfying constraints 1 and 2 without any conflict. To recognize information on edge distances defined by constraint 3, however, the amoeba needs to be illuminated. Therefore, to select a more favorable route with a shorter total distance, it is necessary for the amoeba to examine many invalid combinations of edges by temporarily expanding some conflicting branches.

In Fig. 2(D), we compare two situations in parallel in the left and right panels, where the parameters are set as identical to the actual experimental
condition. In the left panel the amoeba attempts to select the edge $B \rightarrow C$ with the shorter distance $dst(B, C) = 2$, and in the right panel the amoeba attempts to select the edge $B \rightarrow D$ with the longer distance $dst(B, D) = 15$. The states of the neurons $B1, C2,$ and $D2$ are equally fixed at $x_{B1} = x_{C2} = x_{D2} = 0.3$ for all configurations in Fig. 2(D), whereas branch $A2$, conflicting with the former ones, is gradually expanded. Let us confirm how the transition of the illumination pattern reflects the relative difference between the shorter and longer edge distances in response to the expansion of the conflicting branch $A2$.

In the top row of Fig. 2(D), the conflicting branch $A2$ does not exist (i.e., $x_{A2}(t) = 0$). Even if the branches $B1, C2,$ and $D2$ selecting valid edge combinations expand further in the absence of other branches, they can never be illuminated directly as mentioned above. In the middle row of Fig. 2(D), the conflicting branch $A2$ expanded to be $x_{A2}(t') = 0.235$. This triggered the illumination for branch $D2$ (right). Namely, the longer edge $B \rightarrow D$ became unstable due to the perturbation by branch $A2$, and the edge selection was urged to be withdrawn. On the other hand, despite branch $A2$’s perturbation of the identical dimension, branches $B1$ and $C2$ selecting the shorter edge $B \rightarrow C$ remain nonilluminated (left). As shown in the bottom row of Fig. 2(D), the illumination for the branch $C2$ (left) was triggered after the branch $A2$ expanded to $x_{A2}(t'') = 0.3175$.

This comparison reveals that the difference in the edge distances creates the difference in the conflicting branch’s threshold values (i.e., $x_{A2}(t'') - x_{A2}(t') = 0.0825$) for triggering the illuminations to destabilize the edge selections. To destabilize a valid edge combination, often it is necessary for some conflicting branches to expand in illuminated neurons, but this entails a certain degree of difficulty due to the amoeba’s photoavoidance. Therefore, in general, it is more difficult for the amoeba to destabilize a shorter edge as larger dimensions of perturbations by the conflicting branches are necessary. Conversely, a longer edge can be destabilized more easily with smaller perturbations. A shorter-route solution was expected to be more stable than a longer-route solution, because the shorter-route solution is more tolerant to the perturbations triggered by the amoeba itself.

### 2.6 Valid solution as stable equilibrium

The original neural network algorithm [29] was designed to naturally reach a stable equilibrium in which all neurons become unchanged, and was formulated in such a way that the stable equilibrium in principle represents a valid solution of TSP. A solution corresponds to one of the minimum energy points in the potential landscape established by the network dynamics. Any medium that can only relax toward equilibrium, therefore, cannot spontaneously bootstrap itself out of a once-reached solution without an external energy supply. In our system, when the system reaches a solution, the illumination pattern no longer forces the amoeba to reshape, and the amoeba fully completes the inherent expanding movements of branches inside all nonilluminated neurons.
Thus, we expected that a solution would be maintained stably as a kind of equilibrium. When selecting the shortest solution, the amoeba can minimize the probability of being illuminated by aversive optical stimuli evoked by the perturbations. The shortest solution, therefore, would be the most “comfortable” configuration for the amoeba to maintain under this environmental condition.

3 RESULTS

3.1 Experiment 1: Four-city TSP solution

3.1.1 Solution-searching process

In experiment 1, the map shown in Fig. 2(A) was given as a problem (i.e., an instance of TSP) to be solved. Figure 3 shows an example of the computing process observed experimentally. The computing was started by putting

![Problem-solving process of four-city TSP given by the statically fixed map in Fig. 2(A).](image)

FIGURE 3

Problem-solving process of four-city TSP given by the statically fixed map in Fig. 2(A). (A) Initial configuration recorded as a transmitted light image before digital image processing. (B) Early stage of solution-searching process. Three panels show successive time evolution within a period of oscillation. By means of digital image processing, the phase of vertical thickness oscillation is binarized into the relaxing (thickness increasing) and contracting (decreasing) states, represented by the black and gray pixels, respectively. (C) Final stage of solution-searching process. Two panels show time evolution within a half period of oscillation. (D) First-reached solution $D \rightarrow C \rightarrow B \rightarrow A \rightarrow D$ with the shortest travel distance 12. (Duration $\approx 5$ h) (E) Spontaneous destabilization. The newly emerged branch $D_2$ started to invade the illuminated region, contrary to its photoavoidance response. (F) Second-reached solution $B \rightarrow C \rightarrow D \rightarrow A \rightarrow B$ with the shortest distance. (Duration $\approx 1$ h).
In the early stage, the spherical amoeba flattened into a disc-like shape and expanded its thin periphery horizontally with circular symmetry. We observed that the amoeba exhibited various spatiotemporal oscillation patterns, including highly symmetric ones and symmetry-broken chaotic ones, and the patterns varied in a nonperiodic and stochastic manner.

Figure 3(B) shows the state transition in a period of oscillation at the time when the amoeba came to have some expanding branches that were about to reach their own threshold values for triggering the illuminations. It should be recalled that the state $x_i(t)$ increases in a non-monotonous manner as the amoeba’s branch alternates between small-scale expansion and shrinkage at each oscillation period. Compared with the left and right panels, the center panel has the largest number of illuminated neurons, because at that moment most of the neuron states increased to take their maximum values for that oscillation period. At this stage, the illuminations blinked at short intervals due to the non-monotonous changes in neuron states. Because some conflicting branches performed their movements by invading illuminated neurons in a fluctuating manner, a wide variety of the illumination patterns were evoked within a short time. Through a trial-and-error process to examine diverse illumination patterns, the amoeba continued its deformation in search of an infrequently illuminated configuration; that is, a more stable solution with a shorter total distance.

Figure 3(C) shows the state transition in a half period of the oscillation at the time when the amoeba entered the final stage of the solution-searching process. Although the amoeba was about to reach a valid solution, the transition of the illumination pattern was still observed. The expanding branches $D_1, C_2, B_3,$ and $A_4$ were the least frequently illuminated ones, whereas others were in the middle of their shrinking.

Figure 3(D) shows that the amoeba reached an optimal solution $D \rightarrow C \rightarrow B \rightarrow A \rightarrow D$ with the shortest total distance 12. A successful solution can be recognized as the stabilization of the illumination pattern. The branches $D_1, C_2, B_3,$ and $A_4$ selecting the solution sustained their expanded states for about 5 h.

3.1.2 Searching for multiple solutions via spontaneous destabilization

Figure 3(E) shows the state transition in a half period of the oscillation about 5 h after the situation shown in Fig. 3(D). At this time, the long maintained stabilizing mode of the first solution, however, was spontaneously switched to the destabilizing mode even though no explicit external perturbation was applied. Interestingly, the amoeba spontaneously destabilized the solution
$D \rightarrow C \rightarrow A \rightarrow D$ as a branch newly emerged and suddenly started to invade the illuminated neuron $D2$ contrary to its photoavoidance response. The expansion of the branch $D2$ triggered the illuminations for neurons $D1$, $C2$, and $B3$. This perturbation of a large dimension induced some branches to start their large-scale expanding and shrinking movements, and the solution-searching process involving the transition of the illumination pattern was restarted.

As shown in Figure 3(F), the destabilizing mode was switched to the stabilizing mode again since the amoeba subsequently reached another shortest solution $B \rightarrow C \rightarrow D \rightarrow A \rightarrow B$. This solution was maintained for about 1 h.

Afterwards, the spontaneous destabilization of the solution occurred once more, and the solution-searching process was restarted again (figures not shown). Consequently, the amoeba reached one more shortest solution, $B \rightarrow A \rightarrow D \rightarrow C \rightarrow B$, maintained for about 1.5 h. During this 16 h experimental trial, eventually the amoeba found three different optimal solutions.

The effective running time of the computing in the present experimental condition was limited to within about 16 h. After the time limit, the system became unstable enduringly (i.e., not transiently), as the amoeba’s photoavoidance response became irrecoverably insensitive.

3.1.3 Statistical results

We carried out 13 experimental trials with the parameter setup almost identical to the above one. Figure 4(A) shows the frequencies of reaching the shortest, the second-shortest, and the longest solutions. We certified that a solution had been reached only when two conditions were met; i) the configuration calculated by inverting the illumination pattern (i.e., $1 - y_i$ for all $i$) represented a valid solution, and ii) the illumination pattern was stabilized without any change for more than 30 min. White bars indicate the results of the first solution

![Figure 4](image-url)
reached for each trial. In 10 of 13 trials, the amoeba reached the shortest solution, whereas the longest solution was never reached.

In some cases, more than one solution was found in a trial, as the spontaneous destabilization of once-reached solution occurred several times. Including such cases, black bars indicate the results counting all solutions found in all observations, where we counted it as one if an identical solution was reached more than once in a trial. We can confirm that both white and black bars represent success rates of more than 70% of reaching the shortest solution.

Figure 4(B) shows the duration distributions of the shortest (white) and the second-shortest (gray) solutions. Note that the results are shown in a stacked manner. For example, 1 to 1.5 h maintenance of the shortest solution was observed six times, and that of the second-shortest one was observed once. We can confirm that the shortest solution was maintained longer than the second-shortest one. This suggests that the shortest one is more stable than the second-shortest one and may be the most “comfortable” condition for the amoeba to stay in.

3.2 Experiment 2: Autonomous meta-problem solving

3.2.1 Modifications in experimental setups

The search space explored by the amoeba in experiment 1 is a solution space, a set $X^{N \times N}$ covering all possible combinations of neuron states, where $X = [0.0, 1.0] \subset \mathbb{R}$ and $N = 4$. On the other hand, an instance of $N$-city TSP is given by a city map defining travel distances of $N(N-1)/2$ edges. In the map of four cities $A, B, C$, and $D$, there are six edges $AB, AC, AD, BC, BD, and CD$. When each edge distance is defined arbitrarily within a finite real interval $E = [dst_{Min}, dst_{Max}] \subset \mathbb{R}$, a space consisting of all possible instances of $N$-city TSP that we call a problem space is represented by a set $E^{N(N-1)/2}$ covering all possible combinations of edge distances.

In experiment 2, to extend the concept of standard problem solving, we modify the setups of experiment 1 in such a way that the search space is expanded to include not only the solution space $X^{16}$ but also the problem space $E^6$. We additionally set up 6 neurons labeled with corresponding names of the edges and fabricated a new container structure (Fig. (5)) that also carries over the original 16 neurons from experiment 1. Hereafter, for a total of 22 neurons, the newly added 6 and the original 16 are called problem-finding neurons and solution-searching neurons, respectively.

Applying an arbitrarily chosen mapping from the neuron states to the edge distances, we modify the dynamics in the optical feedback in such a way that the edge distances are altered in response to the amoeba’s movements. At every 6 sec, we update the edge distance between cities $P$ and $Q$, written as $dst(P, Q, t)$, to become inversely proportional to the state of problem-finding neuron $PQ$ as defined by the following mapping:

$$dst(P, Q, t + \Delta t) = (dst_{Max} - dst_{Min})(1.0 - x_{PQ}(t)) + dst_{Min},$$

where the maximum and minimum distances are given as $dst_{Max} = 11$ and $dst_{Min} = 1$, respectively. Accordingly,
the problem (i.e., instance of four-city TSP) to be solved evolves with the changes in problem-finding neurons. In other words, the amoeba is allowed to find and select the problem to solve. It is also possible to consider that the amoeba is allowed to modify the program by evolving the edge distances defining the coupling weights $w_{ij}$, because a set of the weights defining the interactions among neurons can be regarded as representing a program code in neurocomputing.

For solution-searching neurons, excluding the newly introduced setup for the distance updating, all setups including the parameter setup for updating the illumination pattern are kept identical to those in experiment 1. On the other hand, we manage not to illuminate problem-finding neurons.

3.2.2 Meta-problem solving

With the above setups, the amoeba concurrently executes two different levels of computing processes, the problem solving in solution-searching neurons and the distance updating in problem-finding neurons. Note that the two processes are able to interact through two kinds of channels. Through the optical feedback system, the state transition in problem-finding neurons influences the transition of the illumination pattern in solution-searching neurons. Through the amoeba’s body, some form of information exchange between problem-finding branches and solution-searching branches can be performed by streaming the sol in a shuttle-wise manner. We say that the two processes are consistent when a configuration selected by solution-searching neurons is a valid solution to a problem defined by problem-finding neurons. When the computing processes are consistent and all neurons are stabilized without large changes, we obtain a pair of the problem and its solution represented by the amoeba’s shape at that time as an output result of this computation. We estimated that the amoeba’s unique capability of spontaneously switching between the stabilization and destabilization in order to search for multiple solutions would be suited for exploring more than one consistent problem-solution pair.

Because which problem will be solved is unknown before starting the computation, the scheme of this computation differs from the concept of standard problem solving: Initially given a problem, find a solution to the problem. It also differs from another common scheme to solve an “inverse problem”: Initially given a solution, infer a problem whose solution is the given one. In our scheme, both the problem and its solution are not given initially, but our system has to output finally both at the same time. In this sense, our system solves what we call a “meta-problem”, the problem of what problem should be solved. Although in this paper we demonstrate an extended version of the TSP solution as an example of meta-problem solving, this does not mean that the scheme of meta-problem solving is limited in this form of embodiment. The scheme can be applied to arbitrary application problems if the problem space can be defined appropriately as a set of potentially-generated program...
codes that deserve to be examined. Our view on a possible application of meta-problem solving is described later.

### 3.2.3 Concurrent processes of problem finding and solution searching

As in experiment 1, the computation was started from the all-zero initial configuration. That is, the initial map is given with all edges equally taking the maximum distance $d_{t_{max}} = 11$.

Figure 5(A) shows that the amoeba’s concentric deformation proceeded and some expanding branches triggered the illuminations about 30 min after the computation had started. At this stage, all problem-finding neurons still had close values, and thus the total travel distances for all valid solutions were almost even.

Shortly afterward, the amoeba selected a valid solution. Figure 5(B) shows the consistent pair of the problem and its solution $A \rightarrow D \rightarrow B \rightarrow C \rightarrow A$, that is, the output of this meta-problem solving. The solution gives the longest route in the map whose edge distances became uneven due to nonuniform expanding movements of problem-finding branches. The problem-solution pair was stably maintained for about 1 h.

Afterward, problem-finding branches continued the elongation of the edge $AD$ and the shortening of the edge $AC$ in the map. This behavior progressively lowered the solution’s stability (i.e., tolerance to perturbations) as its total travel distance was elongated.

**FIGURE 5**
Meta-problem-solving process of four-city TSP in which the map dynamically changes in response to the amoeba’s deformation. For each map (i.e., problem) shown in the right panels, we rounded off the numeric values of the distances to two decimal places, where the edge lengths are not precise. (A) Early stage of computing process. (B) First-reached consistent problem-solution pair. The solution $A \rightarrow D \rightarrow B \rightarrow C \rightarrow A$ has the longest distance, about 27.6. (Duration ≃ 1 h). (C) Spontaneous destabilization. The newly emerged branch $D4$ started to invade the illuminated region, and the problem-solution pair became inconsistent. (D) Second-reached consistent problem-solution pair. The solution $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A$ has the shortest distance, about 30.4. (Duration ≃ 1 h).
Figure 5(C) shows that a large-scale spontaneous destabilization of the solution occurred about 1 h after the situation shown in Fig. 5(B). As the newly emerged branch $D_4$ spontaneously invaded the illuminated region, the branches $A_1$, $C_4$, and $D_2$ selecting the solution started to be illuminated. This perturbation destroyed the consistency of the problem-solution pair, and some branches restarted their unstable large-scale expanding and shrinking movements for searching for another problem-solution pair.

After inconsistent problem-finding and solution-searching processes continued for about 4.5 h, subsequently the amoeba reached the consistent pair of the problem and its shortest solution $A \rightarrow B \rightarrow D \rightarrow C \rightarrow A$ shown in Fig. 5(D). This problem-solution pair was stably maintained for about 1 h.

The spontaneous switching between the stabilization and destabilization, however, occurred several times afterwards. Consequently, the amoeba found a number of consistent problem-solution pairs.

4 DISCUSSION AND CONCLUSION

We proposed two computing schemes employing the amoeboid organism by introducing optical feedback based on a recurrent neural network dynamics. We conclude this paper with summaries of the results together with discussions on our proposal’s advantages and disadvantages.

For the first scheme based on the concept of standard problem solving, the results of experiment 1 are summarized as follows:

1. Our system had a success rate of more than 70% in reaching the optimal solution of four-city TSP.

2. As the amoeba spontaneously switches between the stabilization and destabilization of once-reached solutions, our system was found to be capable of searching for more than one solution in a trial without applying external perturbation.

In this scheme, it is possible to implement not only combinatorial optimization but also various types of information processing that can be translated into search problems. For example, logical operation, associative memory, and pattern recognition are implementable only by modifying some experimental setups such as the number of neurons and the parameters to determine the coupling weights [28].

It would be possible to scale up our system to examine its capability of TSP solution with a larger number of cities. The size of an individual amoeba is adjustable from several tens of micrometers to several meters, and its shape is freely deformable to fit into arbitrary network topology. As we often witness a ten-centimeter amoeba having hundreds of branches even under
natural conditions, the experiment with a container structure having hundreds of neurons would be implementable in principle if its network topology is properly designed.

There may be a view that the optical feedback system, an extrinsic factor that gives the amoeba hints to select the optimal route, is a major contributor to the optimization, but the amoeba’s intrinsic behavior is not. However, we have already confirmed that the optimization rate dramatically decreases if the optical feedback system is reconfigured to be incapable of sensing the small-scale oscillatory movements of the amoeba’s branches (i.e., non-monotonous changes in neuron states) in the searching process. That is, the amoeba’s oscillatory behavior is essential for enhancing our system’s optimization capability. We will report this result elsewhere.

On the other hand, there are some performance limitations derived from employing a living organism for computing, such as its slow processing speed and limited running time. However, it may be possible to implement our schemes with other faster oscillatory media, once we identify key factors within the spatiotemporal dynamics of the amoeba’s oscillatory behavior.

The spontaneous destabilization – the invasion of the amoeba’s branches into inhibited (illuminated) regions – is a notable behavior in our system. It occurs nonperiodically at stochastically distributed sites, when observed macroscopically. Although its mechanism is currently under investigation, we speculate that intrinsic fluctuations at the microscopic level are amplified in some way to influence the destabilization at the macroscopic level. Possible sources of the intrinsic fluctuations include thermal noise, local nonuniformity of chemical distributions, and local unevenness of the gel layer’s stiffness distribution. Some form of positive feedback effect produced by the coupling of chemical and hydrodynamic processes may be responsible for expanding the tiny fluctuations into extensive and sustained movements.

As an example of the unstable dynamics capable of amplifying the fluctuations, we can cite chaos characterized as nonperiodic but nonrandom behavior. Some chaotic dynamics with properly tuned parameters can intermittently switch between the stabilization and destabilization of macroscopic spatiotemporal patterns by amplifying microscopic fluctuations [31]. It may be possible to consider that the oscillation dynamics of the actomyosin population generates high-dimensional spatiotemporal chaos. Indeed, the system’s behavior is chaotic as its time evolution is unstable and unreproducible. Its capability for spontaneous destabilization, however, is robustly maintained and qualitatively reproducible. This resembles the robustness of strange attractors of chaotic systems. On the other hand, the efficiency of chaotic dynamics for combinatorial optimization has already been demonstrated theoretically with chaotic neural network models, where both stabilizing and destabilizing effects contribute to the efficient search [32, 33].
For the second scheme named *meta-problem solving*, the results of experiment 2 are summarized as follows:

1. In *meta-problem solving* of four-city TSP defined as *concurrent processes* in which a *problem-finding process* and a *solution-searching process* are executed simultaneously, we confirmed that our system can select a consistent pair of a problem (i.e., an instance of TSP) and its solution.

2. Due to the amoeba’s capability of spontaneous switching between the stabilizing and destabilizing modes, our system was found to be capable of generating more than one consistent problem-solution pair.

As mentioned, this scheme can be applied to arbitrary application problems if the *problem space* can be defined appropriately. In experiment 2 we confirmed that our system could find consistent problem-solution pairs for TSP, which is a particularly hard optimization problem. This suggests that our system will be able to work well even when dealing with other complex problems.

When applied to some techniques in automated reasoning and machine learning [34], *meta-problem solving* may be a useful concept to create autonomous agents that are capable of coping with unexpected situations. Suppose that our computing system is developed to be embedded in a control system of an autonomous agent such as a mobile robot. The standard *problem-solving process* to find a *solution* to a given *problem* would correspond to the agent’s computing process to select an appropriate *behavior* in an expected situation by following instructions of a given *program* prescribed in advance by a programmer. However, the *program* prescribed only for expected situations would always be insufficient for the agent to act in an actual environment where unexpected situations arise very often. The agent may unexpectedly be stuck in a stalematized situation in which the given *program* is helpless for finding a way out of the stalemate. In such a case, if a *meta-problem solver* was incorporated in the agent’s control system, the agent evolves the *program* and may succeed in finding an emergent *solution* to overcome the stalemate owing to the *meta-problem solver’s* capability of the spontaneous destabilization. This is a desirable scenario that we picture in our minds as a potential application of *meta-problem solving*.

**APPENDIX: EXPERIMENTAL SETUPS**

The amoeba was fed oat flakes (Quaker Oats, Snow Brand Co.) on a 1% agar gel at 25°C in the dark. The stellate container structure (thickness approximately 0.2 mm) is made from an ultrathick photoresist resin (NANO™ XP SU-8 3050, MicroChem Corp.) by a photolithography technique, and was coated with Au using a magnetron sputterer (MSP-10, Shinkuu Device
Co., Ltd.). The experiments were conducted in a dark thermostat and humidistat chamber (27 ± 0.3°C, relative humidity 96 ± 1%, THG062PA, Advantec Toyo Kaisha, Ltd.). For transmitted light imaging, the sample was placed on a surface light guide (MM80-1500, Sigma Koki Co., Ltd.) connected to a halogen lamp light source (PHL-150, Sigma Koki Co., Ltd.) equipped with a bandpass filter (46159-F, Edmund Optics Inc.), which was illuminated with light (intensity 2µW/mm²) at a wavelength of 600 ± 10 nm, which does not affect the amoeba’s behavior [30]. The intensity of the white light (monochrome color R255;G255:B255) illuminated from the projector (3000 lm, contrast ratio 2000:1, U5-232, PLUS Vision Corp.) was 123µW/mm². The outer edge of the circuit (the border between the structure and the agar region) was always illuminated to prevent the amoeba from moving beyond the edge.

REFERENCES


