

Mathematical model of the amoeba-inspired combinatorial optimization machine for physical implementation and its equivalence to recurrent neural networks

Yusuke Miyajima^{*} and Masahito Mochizuki[†]

Department of Applied Physics, Waseda University, Okubo, Shinjuku-ku, Tokyo 169-8555, Japan



(Received 21 October 2025; accepted 8 May 2026; published 9 June 2026)

Information-processing algorithms inspired by survival strategies of slime molds have been intensively studied to realize combinatorial optimization machines with high computational efficiency and low power consumption. However, previously proposed mathematical models are not suitable for physical implementation as they involved constraints due to the volume-conservation law of slime molds, multiple conditional branches, and numerous complex functions such as sigmoid functions. Here, we propose a mathematical model, the Amoeba TSP (traveling salesman problem) recurrence formula model, which simplifies the information-processing procedure and enables physical implementation without sacrificing optimization performance. Specifically, we demonstrate using numerical simulations that (1) the conservation-law constraints assumed in previous models can be removed, (2) several sigmoid functions can be eliminated, and (3) Gaussian-distribution random numbers that can be generated from thermal fluctuations can be employed as random fluctuations. These improvements broaden the range of candidate materials and phenomena for physical implementation and reduce the number of device components required. Furthermore, we discover that our mathematical model integrating these modifications is mathematically equivalent to a recurrent neural network with fixed weights that exploits nonlinear dynamics. Remarkably, despite its significant simplification, the proposed model exhibits superior solution-search performance compared with the previous models in terms of solution quality, convergence speed, and scalability. Our results open a practical pathway toward realization of highly efficient and high-performance combinatorial optimization machines based on non-von Neumann-type architectures and provide insights into bioinspired information processing by uncovering a connection between the computational principles of slime molds and recurrent neural networks.

DOI: [10.1103/zgyb-cfpg](https://doi.org/10.1103/zgyb-cfpg)

I. INTRODUCTION

In recent years, the demand for computational resources has increased exponentially in our information society, where artificial intelligence and large-scale combinatorial optimization are ubiquitous. However, conventional computers are predicted to reach a performance saturation in terms of computational efficiency and power consumption due to their underlying computational principles. To overcome this challenge, it is necessary to develop computers based on new computational paradigms beyond the von Neumann architecture [1]. One can regard emergent intelligence in natural phenomena, exemplified by living organisms, as a source of inspiration for establishing such new principles. Consequently, intensive research efforts have been devoted to constructing mathematical models that reproduce these phenomena

and to realizing their physical implementation as novel computing devices [2–8].

In our study, we focus on an optimization mathematical model inspired by survival strategies of slime molds [9,10]. The computational capability emerges from their deformation dynamics, through which they adapt to the surrounding environment by extending toward nutrients (attractive stimuli) and shrinking away from light (repellent stimuli). During this process, each part of the body attempts to acquire as many nutrients as possible while simultaneously minimizing the risk of illumination. However, these body parts do not deform independently but interact with one another as components of a single body, thereby exhibiting complex fluctuating dynamics [11,12]. This interaction is the key aspect underlying the computational capability of slime molds.

Previous experimental studies have demonstrated that slime molds are capable of solving problems such as mazes [13,14] and combinatorial optimization problems [15–17], despite being single-celled organisms. Unlike neural circuits in the brain, the information processing in slime molds is autonomous and decentralized, and is therefore expected to provide a potential principle for efficient parallel computation. However, living slime molds are difficult to directly utilize as computers because the timescale of their deformation dynamics is too slow ($\sim 10^{-5}$ m/s). Therefore, intensive research efforts have been devoted to constructing computational

^{*}Contact author: y.m-ichikawa@akane.waseda.jp

[†]Contact author: masa_mochizuki@waseda.jp

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

principles inspired by the survival strategies of slime molds and to implementing optimization machines, referred to as amoeba-inspired combinatorial optimization machines [18] based on these principles using physical devices.

To date, these studies have been carried out in three steps: (1) experimental examinations of the computational performance of living slime molds; (2) construction of a mathematical model to reproduce the experimental results and to explain the underlying mechanisms; and (3) physical implementation of the models using devices. There are two types of amoeba-inspired combinatorial optimization models: those addressing problems with only constraints such as the satisfiability problem (SAT) and those addressing problems with both constraints and a cost function such as the traveling salesman problem (TSP) [19]. For the former type, the so-called Amoeba SAT model has been proposed [20], and previous studies have demonstrated its physical implementations using various devices, including analog electronic circuits [21], digital electronic circuits [22], quantum nanostructures [23], superconducting logic circuits [24], and electronic Brownian ratchets [25].

In contrast, with respect to the latter type, implementation has so far been limited to analog electronic circuits [26,27], even though a well-established mathematical model, known as the Amoeba TSP model, has already been proposed [28]. A possible reason is that a constraint condition imposed in the Amoeba TSP model, referred to as the conservation law, restricts candidate devices and thereby makes physical implementation difficult. This constraint was introduced to represent the volume-conservation law of the slime-mold body, which has been naively believed to be essential for its computational capability. Analog electronic circuits can naturally satisfy this conservation law via the conservation of electric charges. (Note that the analog electronic circuits can also be applied to the cases without this constraint in principle.)

In our previous study [29], we investigated three conditions characterizing the Amoeba TSP model and analyzed how modifications of these conditions affect the optimization performance. We found that three modifications not only enhanced the physical implementability of the model but also improved the performance. Notably, we demonstrated that the conservation law mentioned above is not necessary for the solution-searching ability of the model. Moreover, we found that its absence even leads to better performance in contrast to the naive belief. We proposed a mathematical model named the Improved Amoeba TSP model by applying the three modifications to the original Amoeba TSP model. The proposed model is more suitable for physical implementation and can find feasible solutions with shorter route lengths in a much smaller number of iterations compared with the original model. However, the proposed model still involves several complicated information-processing steps such as counting operations and conditional branching, which may pose difficulties for physical implementation.

In this paper, we advance the development of amoeba-inspired combinatorial optimization machines by further improving the Amoeba TSP model in terms of both optimization performance and physical implementability. Building on insights from our previous study [29], we introduce a refined

mathematical formulation that eliminates conservation-law constraints, counting operations, and conditional branching, yielding a simple recurrence relation well suited for physical implementation. Using numerical simulations, we demonstrate that the proposed model achieves superior performance, consistently finding high-quality solutions with shorter route lengths in fewer iterations, while avoiding computational operations that are difficult to realize in physical systems. Furthermore, we establish the equivalence between the proposed model and a recurrent neural network, thereby situating it within a broader computational framework. This equivalence provides deeper theoretical insight and enables implementation using existing neuromorphic and physical neural-network hardware. These results not only strengthen the theoretical foundation of the amoeba-inspired optimization but also significantly enhance the feasibility of practical device-level implementations for next-generation computing architectures.

II. PREVIOUS MATHEMATICAL MODEL: AMOEBAS TSP MODEL

As a starting point of our study, we explain the previously proposed mathematical model of the amoeba-inspired combinatorial optimization machine known as the Amoeba TSP model [28]. We note that for comprehensive understanding of the related models and formulations, some equations and explanations overlap with those in our previous paper [29]. The model is based on the Hopfield-Tank-Amari formalism [30] and is described by Eqs. (1)–(7). Figure 1 shows the experimental setup for solving combinatorial optimization problems using a living amoeba [16,17]. An amoeba is confined within a circular device with numerous radially extending lanes. Each lane is indexed by a pair of indices (V, k) , where V and k represent the label of the city and the visiting order, respectively. The Amoeba TSP model contains two important variables, $X_{V,k}(t)$ and $L_{V,k}(t)$. The former represents the length of the branch in the lane (V, k) , while the latter is associated with the determination of whether the lane (V, k) is illuminated. Here, t denotes a unit of time, referred to as the iteration number in this model (as defined later).

First, the values of $L_{V,k}(t + 1)$ are calculated from the values of $X_{V,k}(t)$ for all $V (= 1, 2, \dots, n)$ and $k (= 1, 2, \dots, n)$ using Eq. (1). If $L_{V,k}(t + 1) > 0.5$ (≤ 0.5), the lane (V, k) should be (should not be) illuminated. This determination is carried out for all lanes. Here, $\sigma_{\gamma, \theta}(x)$ in Eq. (1) is a sigmoid function with parameters γ and θ defined in Eq. (3). The value $W_{V,k,U,l}$ in Eq. (1) given by Eq. (4) represents the cost associated with the distance between the cities and the penalty for breaking the TSP constraints. Specifically, $\text{dst}(U, V)$ in Eq. (4) denotes the distance between city U and city V , and the parameter ν is a proportionality coefficient. The parameters λ and μ correspond to the penalty for visiting one city twice and that for visiting two different cities simultaneously, respectively. The combination of the step function values $\theta(L_{V,k} + 0.5)$ for all (V, k) specifies the illumination pattern, that is, which lanes are illuminated in binary form. The details of the parameter values are explained in the next section.

Second, the values of $X_{V,k}(t + 1)$ are updated using Eq. (2) based on the illumination pattern determined in the previous

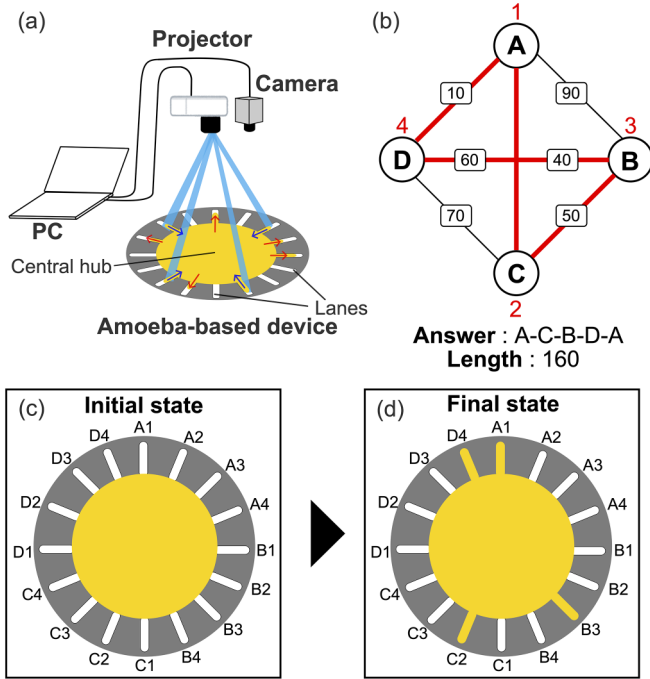


FIG. 1. (a) Schematic diagram of the experimental system for solving the TSP using a living slime mold. The system consists of an amoeba-based device and an illumination system comprising a projector, a video camera, and a personal computer (PC). The device consists of a central hub and n^2 lanes corresponding to the n^2 variables labeled by indices V and k that represent a solution of the TSP. Here, V ($= A, B, C, D$) denotes the alphabetical label of the cities and k ($= 1, 2, 3, 4$) denotes the visiting order. While integer numbers are used for the city label V in the text, we use alphabetical labels in this figure to avoid confusion between the city label and the visiting order. The slime-mold body in a lane is referred to as a branch. The branch length X_{V_k} is measured by the camera, and L_{V_k} is calculated by the PC according to Eq. (1). Illumination is applied (or not) to lane V_k if L_{V_k} is greater (or smaller) than 0.5. These operations are conducted for all lanes V, k within one iteration. (b) Typical four-city TSP map. The path highlighted in thick red lines indicates the feasible solution obtained after the search by the slime mold, corresponding to the state shown in panel (d). (c) Initial state of the slime mold. At first, the slime mold occupies only the central hub of the device and does not extend into any lanes. The mold is placed on an agar plate containing nutrients, and branches elongate into lanes to acquire nutrients if they are not illuminated, whereas branches contract when the corresponding lanes are illuminated. (d) Final state of the slime mold. The mold selects $n = 4$ lanes (B1, A2, C3, and D4) by elongating the branches, thereby representing a traveling route $B \rightarrow A \rightarrow C \rightarrow D \rightarrow B$.

step. The variables $O_{V_k}(t)$ and $I_{V_k}(t)$ in Eq. (2) denote the contracted and elongated branch lengths in the lane (V, k) under illumination and nonillumination, respectively. This equation describes the dynamics that a slime mold typically contracts (elongates) a branch to avoid light stimuli (to acquire nutrients), when the lane is (is not) illuminated. The variables $\xi_{V_k}(t) \in [-\delta, \delta]$ represent fluctuations in the deformation of the branch in the lane (V, k) , which is given by a uniform random number with the amplitude δ in the original Amoeba TSP model. Owing to this fluctuation $\xi_{V_k}(t)$, the model reproduces the unpredictable behaviors of slime mold in response to light stimuli and nutrients. The slime mold occasionally elongates (contracts) its branches even when the lanes are (are not) illuminated.

We can obtain an approximate solution to the TSP by iteratively evolving the system according to Eqs. (1) and (2) for all lanes (V, k) simultaneously. Here, we define one complete update of all the lanes (V, k) as a single iteration.

$$L_{V_k}(t+1) = 1 - \sigma_{1000, -0.5} \left(\sum_{U=1}^n \sum_{l=1}^n W_{V_k, U l} \sigma_{35, 0.6}(X_{U l}(t)) \right), \quad (1)$$

$$X_{V_k}(t+1)$$

$$= \begin{cases} X_{V_k}(t) - O_{V_k}(t+1) + \xi_{V_k}(t+1) & (L_{V_k}(t+1) > 0.5), \\ X_{V_k}(t) + I_{V_k}(t+1) + \xi_{V_k}(t+1) & (L_{V_k}(t+1) \leq 0.5), \end{cases} \quad (2)$$

$$\sigma_{\gamma, \theta}(x) = \frac{1}{1 + \exp(-\gamma(x - \theta))}, \quad (3)$$

$$W_{V_k, U l} = \begin{cases} -\lambda & (V = U \wedge k \neq l), \\ -\mu & (V \neq U \wedge k = l), \\ -\nu \cdot \text{dst}(V, U) & (V \neq U \wedge |k - l| = 1), \\ 0 & (\text{otherwise}). \end{cases} \quad (4)$$

The variables $O_{V_k}(t)$ and $I_{V_k}(t)$ are defined by Eqs. (5) and (6), respectively. The parameters Δ^{out} and Δ^{in} represent the degrees of contraction and elongation of a branch, respectively. The variable $L^{\text{off}}(t)$ denotes the number of lanes that are not illuminated at the t th iteration step. The variable $S(t)$ in Eq. (6) is defined by Eq. (7). Note that the value of $I_{V_k}(t)$ is independent of the indices (V, k) , which causes the slime mold to elongate its branches uniformly across the $L^{\text{off}}(t)$ nonilluminated lanes.

$$O_{V_k}(t+1) = \begin{cases} 2\Delta^{\text{out}} \sigma_{20, 0.6}(X_{V_k}(t)) & (L_{V_k}(t+1) > 0.5), \\ 0 & (L_{V_k}(t+1) \leq 0.5), \end{cases} \quad (5)$$

$$I_{V_k}(t+1) = \begin{cases} (\Delta^{\text{in}} + \sum_{V=1}^n \sum_{k=1}^n O_{V_k}(t+1) + S(t)) / L^{\text{off}}(t+1) & (L^{\text{off}}(t+1) > 0), \\ 0 & (L^{\text{off}}(t+1) = 0), \end{cases} \quad (6)$$

$$S(t+1) = \begin{cases} 0 & (L^{\text{off}}(t+1) > 0), \\ S(t) + \Delta^{\text{in}} + \sum_{V=1}^n \sum_{k=1}^n O_{V_k}(t+1) & (L^{\text{off}}(t+1) = 0). \end{cases} \quad (7)$$

The Amoeba TSP model is constructed on the hypothesis that the strong optimization capability of slime molds arises from the deformation dynamics of their bodies subject to a volume-conservation law. To incorporate this assumption, the following conservation condition is imposed on the branch lengths in the model:

$$\sum_{(V,k) \in \text{off}} \{I_{V_k}(t+1) + \xi_{V_k}(t+1)\} - \sum_{(V,k) \in \text{on}} \{O_{V_k}(t+1) + \xi_{V_k}(t+1)\} \approx \Delta^{\text{in}}. \quad (8)$$

Here, $\sum_{(V,k) \in \text{on}}$ and $\sum_{(V,k) \in \text{off}}$ denote the summations over lanes with and without illumination, respectively. On the left-hand side, the first term represents the total increase of branch lengths, while the second term represents the total decrease. This constraint indicates that the sum of branch lengths over all lanes increases by a constant value Δ^{in} per iteration. The validity of this relation can be confirmed by the following calculation:

$$\begin{aligned} & \sum_{(V,k) \in \text{off}} \{I_{V_k}(t+1) + \xi_{V_k}(t+1)\} - \sum_{(V,k) \in \text{on}} \{O_{V_k}(t+1) + \xi_{V_k}(t+1)\} \\ &= I_{V_k}(t+1)L^{\text{off}}(t+1) + \sum_{(V,k) \in \text{off}} \xi_{V_k}(t+1) - \sum_{(V,k) \in \text{on}} O_{V_k}(t+1) - \sum_{(V,k) \in \text{on}} \xi_{V_k}(t+1) \\ &\approx I_{V_k}(t+1)L^{\text{off}}(t+1) - \sum_{(V,k) \in \text{on}} O_{V_k}(t+1) \\ &= \left(\Delta^{\text{in}} + \sum_{(V,k) \in \text{on}} O_{V_k}(t+1) + S(t) \right) - \sum_{(V,k) \in \text{on}} O_{V_k}(t+1) \\ &= \Delta^{\text{in}} + S(t). \end{aligned} \quad (9)$$

In most cases, this value remains constant at Δ^{in} , because $S(t)$ is zero except when $L^{\text{off}}(t) = 0$. The sums, $\sum_{(V,k) \in \text{off}} \xi_{V_k}(t+1)$ and $\sum_{(V,k) \in \text{on}} \xi_{V_k}(t+1)$, are approximated to be zero for the transformation from the second line to the third line, because the expectation value of the uniform random number is zero and the number of index pairs (V, k) is assumed to be large. The variable $S(t)$ temporarily accumulates the sum of Δ^{in} , when all the lanes are illuminated and thus no branches cannot be elongated. In this special case, it is given by

$$S(t) = \sum_{(V,k) \in \text{on}} O_{V_k}(t+1) + \xi_{V_k}(t+1). \quad (10)$$

Once nonilluminated lanes appear again, the slime mold distributes the stored length $S(t)$ to them evenly in addition to the default elongation, according to Eq. (6). Therefore, the constraint in Eq. (8) is satisfied thanks to the variable $S(t)$ in all cases except in the case $L^{\text{off}}(t) = 0$.

III. CALCULATION CONDITIONS

A. Parameter values

In the numerical simulations, we adopt the parameter values listed in Table I. These values are determined in the same manner as in the previous study to facilitate direct comparison between the results of the proposed model and those of

TABLE I. Parameter values of mathematical models used for the numerical simulations.

λ	μ	δ	Δ^{out}	Δ^{in}
0.5	0.5	0.003	0.001	0.001

the original Amoeba TSP model. The distances between two cities in the n -city TSP are generated as normally distributed random numbers with a mean of 100 and a standard deviation of 17. The obtained solution must satisfy the constraints of TSP. Therefore, penalties imposed when these constraints are violated should be set to a value larger than the cost associated with the path length. Conversely, differences in route lengths should be weighted as strongly as possible to obtain a solution with a shorter total route. Based on these two considerations, the value of ν is determined as follows.

For any three cities labeled V_1 , V_2 , and V_3 , we consider the path from V_1 to V_3 via V_2 (i.e., $V_1 \rightarrow V_2 \rightarrow V_3$), whose length is given by $\text{dst}(V_1, V_2) + \text{dst}(V_2, V_3)$. When the combination of three cities that maximizes the path length is denoted as V_1^* , V_2^* , and V_3^* , the value of ν is chosen to be slightly smaller than

$$\nu^* = 0.5 / (\text{dst}(V_1^*, V_2^*) + \text{dst}(V_2^*, V_3^*)), \quad (11)$$

so that the following inequality is satisfied for any arbitrary three cities V_1 , V_2 , and V_3 ,

$$\nu \cdot \{\text{dst}(V_1, V_2) + \text{dst}(V_2, V_3)\} \leq \lambda = \mu = 0.5. \quad (12)$$

Both the parameters Δ^{out} and Δ^{in} are set to 0.001. The parameter δ , which represents the amplitude of the fluctuation $\xi_{V_k}(t)$, is set to 0.003. Considering that $O_{V_k}(t)$ takes values between 0 and $2\Delta^{\text{out}} (= 0.002)$, the magnitude relationship between $O_{V_k}(t)$ and $\xi_{V_k}(t)$ changes randomly. As a result, the fluctuating behavior of slime mold can be reproduced in this numerical simulation.

B. Initial state and termination condition

We set $X_{V_k}(0) = 0$ for all V and k as the initial state, corresponding to the experimental condition where the body of the slime mold is confined to the central hub of the device

and no branches extend into the lanes. Starting from this initial configuration, the quantities $L_{V_k}(t)$ and $X_{V_k}(t)$ are iteratively updated according to Eqs. (1) and (2) until $X_{V_k}(t)$ satisfies the termination condition. Specifically, we introduce binary variables $X_{V_k}^{\text{bin}} = \theta(X_{V_k} - 0.99)$ for all V and k to indicate whether (or not) a branch occupies a lane. The termination conditions, corresponding to the TSP constraints, are defined by the following equations:

$$\sum_{V=1}^n X_{V_k}^{\text{bin}} = 1, \quad \sum_{k=1}^n X_{V_k}^{\text{bin}} = 1. \quad (13)$$

A set of $X_{V_k}^{\text{bin}}$ satisfying these conditions is regarded as a feasible TSP solution, and the iteration is terminated immediately when these conditions are satisfied. This termination test is performed at every iteration step in the numerical simulation. The maximum number of iterations is set to 3000, and the process is forcibly terminated if no feasible solution is found within this limit.

C. Evaluation criteria of the solution-search performance

We evaluate the solution-search performance of the models using the following three criteria:

- (1) Success rate for finding a feasible solution among 1000 trials.
- (2) Average number of iterations required to find a feasible solution.
- (3) Average normalized route length of the obtained solutions.

Here, the normalized route length is calculated by dividing the route length of an obtained solution by the estimated mean route length, which is equal to $100n$ for the n -city TSP, since the distance between two cities is given as a normally distributed random number with a mean of 100. A solution with a shorter normalized route length can thus be regarded as a better solution.

D. Solution-search performance of the Amoeba TSP model and its necessity of physical implementation

According to the results of the previous study [28], the Amoeba TSP model is capable of finding feasible solutions in all 500 trials for the n -city TSP ($10 \leq n \leq 20$). Moreover, the number of iterations required to find a solution increases proportionally with the number of cities n , while the average normalized route length remains approximately 0.9 independent of n , even though the number of solution candidates increases exponentially. Note, however, that while the number of iterations grows proportionally with n , the computational time does not follow this scaling law. Computational cost for each time step is proportional to n^4 (see Appendix A). Consequently, the total computational time for the optimization grows proportionally to $n^5 (= n \times n^4)$. It is necessary to suppress the order of computational time per iteration from $O(n^4)$ to $O(1)$ by implementing the combinatorial optimization machine with devices in which the CPU-intensive computations are executed in parallel through the corresponding physical phenomena.

IV. PROPOSED MATHEMATICAL MODEL: THE AMOEBA TSP RECURRENCE FORMULA MODEL

A. Derivation of the Amoeba TSP recurrence formula

In this section, we construct an improved mathematical model, referred to as the Amoeba TSP recurrence formula model, by introducing several modifications to the original Amoeba TSP model. We aim to remove complex operations in the Amoeba TSP model such as counting the number of non-illuminated lanes $L^{\text{off}}(t)$ and conditional branching, which are difficult to represent using analog physical phenomena. This simplification enhances the feasibility of realizing amoeba-inspired combinatorial optimization machines. In addition, the proposed modifications are expected to improve the solution-search performance of the model.

The following five modifications are applied to the original Amoeba TSP model with performing several approximations. Modifications (1), (3), and (4) are found to improve the solution-search performance of the model in our previous study [29], while modification (2) is based on the empirically obtained insight that $L^{\text{off}}(t)$ usually equals zero in the solution-search process. Modification (5) is obtained from the results in our previous study that the steepness of the sigmoid function near the threshold value does not affect the optimization performance.

- (1) Replace the sigmoid function $\sigma_{20,0.6}(x)$ in Eq. (5) with a constant value of 1.
- (2) Extend the update formulas for $I_{V_k}(t)$ and $S(t)$ to the case of $L^{\text{off}}(t) = 0$, without conditional branching in Eqs. (6) and (7).
- (3) Substitute $L^{\text{off}}(t)$ in Eq. (6) with the number of cities n .
- (4) Generate the fluctuation $\xi_{V_k}(t)$ from a Gaussian random distribution instead of a uniform random distribution in Eq. (2).
- (5) Replace the sigmoid function $\sigma_{1000,-0.5}(x)$ in Eq. (1) with the step function $\theta(x + 0.5)$.

After some algebra (see Appendix B for the detailed derivation), we obtain an improved mathematical model defined by the following equations, which we refer to as the Amoeba TSP recurrence formula:

$$\begin{aligned} X_{V_k}(t+1) = & X_{V_k}(t) - 2\Delta^{\text{out}} + 2\alpha\Delta^{\text{out}}n\theta \\ & \times \left[\sum_{U=1}^n \sum_{l=1}^n W_{V_k,Ul} \sigma_{35,0.6}(X_{Ul}(t)) + 0.5 \right] \\ & + \xi_{V_k}(t+1), \end{aligned} \quad (14)$$

with the initial condition $X_{V_k}(0) = 0$. This equation represents the update rule of $X_{V_k}(t)$ describing the temporal evolution of the branch lengths in the lanes. Compared with the original Amoeba TSP model, the four conditional branches are reduced to a single step function. In addition, counting the number of nonilluminated lanes is no longer required, because the conditional branching based on $L^{\text{off}}(t)$ has been eliminated by modification (2). Therefore, the Amoeba TSP model is reduced to a simple recurrence formula, which represents the update rule, and now the proposed model is expected to be suitable for physical implementation. By iteratively updating

$X_{V_k}(t)$ according to Eq. (14), a solution to the TSP can be obtained from the converged (termination) values of $X_{V_k}(t)$.

B. Effects of modifications (1)–(5)

The modifications are intended to improve both the physical implementability and the solution-search performance of the model. In particular, the former is the highest priority for realizing the amoeba-inspired combinatorial optimization machine. Therefore, among several possible candidates, the modifications were selected so as to achieve the following two objectives. The first is to render the model more reproducible using a wider variety of physical materials and physical phenomena. The second is to simplify the model structure. When multiple candidate modifications satisfied these objectives, we adopted the one that most effectively improved the solution-search performance.

We identified above five modifications through the procedure described below. Concretely, we first consider modifying the fluctuations and removing the constraint referred to as the conservation law, which restricts the choice of device candidates for physical implementation. The random fluctuations $\xi_{V_k}(t)$ are changed from a uniform distribution to a Gaussian distribution, which is ubiquitous in nature. For example, we can exploit thermal fluctuations as a generator of $\xi_{V_k}(t)$ obeying the Gaussian distribution. This corresponds to modification (4). In our previous study, we demonstrated that this modification simultaneously contributes to improving the solution-search performance through reducing the number of iterations and shortening the route length of the obtained solution.

With respect to removing the conservation law and simplifying the model structure, several possible modifications can be considered. Among them, we selected a modification that most effectively improves the solution-search performance. In practice, in our previous study, we applied various candidate modifications to the Amoeba TSP model and examined the resulting changes in solution-search performance. As a result, modifications (1) and (3) were obtained. Modification (1) is based on an extrapolation inspired by the fact that the number of nonilluminated lanes necessarily becomes n in the final stage of the solution search. Owing to this modification, the conservation law imposed in the original Amoeba TSP model, which restricts the device candidates for physical implementation, can be removed. Moreover, it becomes unnecessary to count the number of nonilluminated lanes at each iteration step, and thus the model is further simplified. Due to modification (3), the value of the argument of the sigmoid function, $X_{V_k}(t)$, is no longer required when calculating the values of $I_{V_k}(t)$, which also leads to a simplification of the computation. Note that although the sigmoid function is replaced by the constant value 1 in this case, a more general formulation can be considered by introducing a tunable parameter α in the third term of the proposed model in Eq. (14), which allows the degree of contraction of the slime-mold body to vary. As shown in our previous study, this modification reduces the number of solution-search iterations due to the increased extension of the slime-mold body in a single iteration, while the route length of the obtained solution remains unchanged.

TABLE II. Solution-search performance of the proposed Amoeba TSP recurrence formula model in Eq. (14) for $\alpha = 1.0$.

n	Success rate	Average number of solution-search iterations	Normalized route length
20	1.00	70.8	0.916
30	1.00	93.9	0.869
40	1.00	94.6	0.849
50	1.00	105.9	0.847
60	1.00	127.4	0.848
70	1.00	164.4	0.851
80	1.00	159.5	0.836
90	1.00	198.9	0.852
100	1.00	245.1	0.844
110	1.00	291.2	0.845
120	1.00	266.6	0.830
130	1.00	296.6	0.827
140	1.00	332.5	0.835
150	1.00	370.5	0.831
160	1.00	406.6	0.834
170	1.00	446.2	0.838

In addition, we further applied modifications (2) and (5) to simplify the model structure without affecting the solution-search performance. Modification (2) eliminates a conditional branch in Eq. (7). By applying modification (5), two sigmoid functions are reduced to a single sigmoid function by exploiting the property of composite step functions, thereby simplifying the model formulation. Note that modification (5) is not merely a mathematical transformation but also provides an advantage in terms of reducing the number of components required for physical implementation. Specifically, as shown in Ref. [26], five circuit components are required to implement a single sigmoid function in an analog electronic circuit. Therefore, removing the sigmoid functions for all indices V and k reduces the total number of components by $5n^2$. Based on the above procedure, we identified the five modifications (1)–(5) and applied them to the Amoeba TSP model.

V. SOLUTION-SEARCH PERFORMANCE OF THE AMOEBA TSP RECURRENCE FORMULA MODEL

In this section, we evaluate the solution-search performance of the proposed Amoeba TSP recurrence formula model by performing 1000 trials for each n -city TSP ($n = 20, 30, \dots, 180$) using the three criteria introduced in Sec. IV. In the following, these criteria are referred to as the success rate, average number of solution-search iterations, and normalized route length, respectively.

A. Performance for $\alpha = 1.0$

Table II summarizes the results for $\alpha = 1.0$ when n ranges from $n = 20$ to $n = 170$. Figures 2(a) and 2(b) show the n dependence of the average number of solution-search iterations and that of the normalized route length, respectively. Feasible solutions are successfully obtained in all 1000 trials for all values of n . The number of solution-search iterations is smaller by one to two orders of magnitude compared with

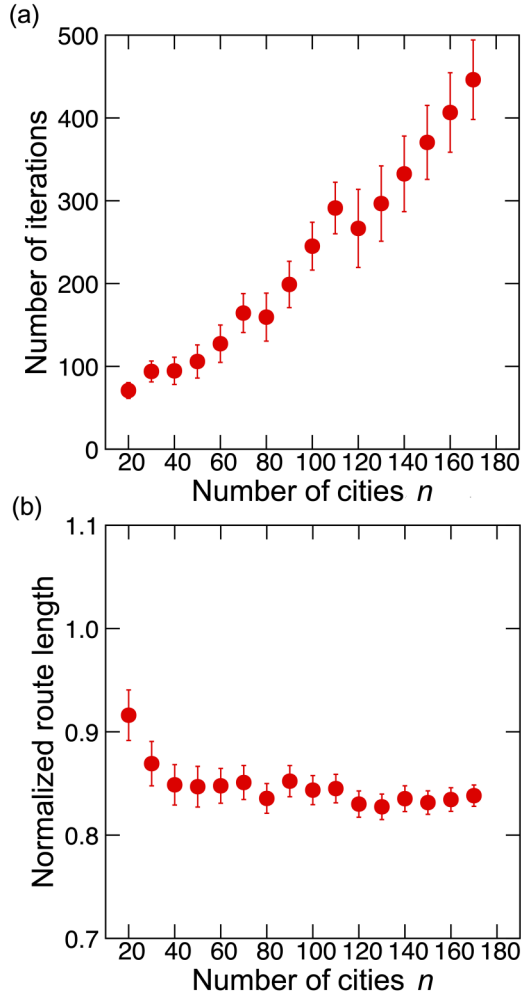


FIG. 2. Dependence of (a) the average number of solution-search iterations and (b) the normalized route length on the number of cities n for the Amoeba TSP recurrence formula model in Eq. (14) for $\alpha = 1.0$.

that of the original Amoeba TSP model. The normalized route length remains approximately constant at around 0.85 when $n \geq 40$ and is smaller than that of the original Amoeba TSP model when $n \geq 30$.

We compare the solution-search performances of the original Amoeba TSP model [28], the Improved Amoeba TSP model proposed in our previous study [29], and the present Amoeba TSP recurrence formula model with $\alpha = 1.0$ in Figs. 3(c) and 3(d). Feasible solutions are obtained in all trials for all models. The number of solution-search iterations in the present model is much smaller than those of the other models. Moreover, the normalized route length of the present model is shorter than those of the others when $n \geq 30$. Therefore, the proposed Amoeba TSP recurrence formula model exhibits better solution-search performance than the previous models. It should be noted that the performances of the Improved Amoeba TSP model and the Amoeba TSP recurrence formula model differ, even though most of the applied modifications are identical. This difference arises from the following reason. In the derivation of the Improved Amoeba TSP model, the variable $L^{\text{off}}(t+1)$ was replaced by n in the first equation on

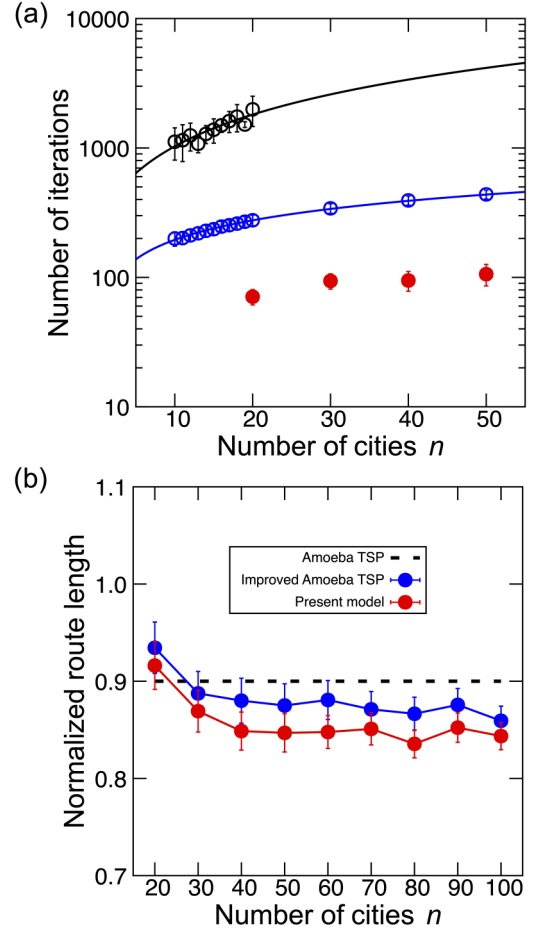


FIG. 3. Comparison of (a) the average number of solution-search iterations and (b) the normalized route length among the original Amoeba TSP model [28], the previously proposed Improved Amoeba TSP model [29], and the present Amoeba TSP recurrence formula model with $\alpha = 1.0$.

the right-hand side of Eq. (B5). In other words, $I_{V_k}(t)$ was defined not by Eq. (B8) but by the equation

$$I_{V_k}(t+1) = \frac{\Delta^{\text{in}} + 2\Delta^{\text{out}}(n^2 - L^{\text{off}}(t+1))}{n}, \quad (15)$$

as shown in the fourth line on the right-hand side of Eq. (B5).

B. α Dependence of the performance

We further examine the solution-search performance of the Amoeba TSP recurrence formula model in Eq. (14) for several values of the parameter α ($= 0.03, 0.05, 0.1, 0.3$, and 0.5). In this equation, the elongation length of a branch in a nonilluminated lane, $2\alpha\Delta^{\text{out}}nL_{V_k}(t+1)$, is proportional to the number of cities n , whereas the contraction length of an illuminated branch remains constant at $2\Delta^{\text{out}}$. The parameter α controls the elongation length in nonilluminated lanes and thus tunes the balance between elongation and contraction.

Figures 4(a)–4(c) show the dependence of the three evaluation criteria on the number of cities n for various values of α . According to Fig. 4(a), the model with a small α fails to find

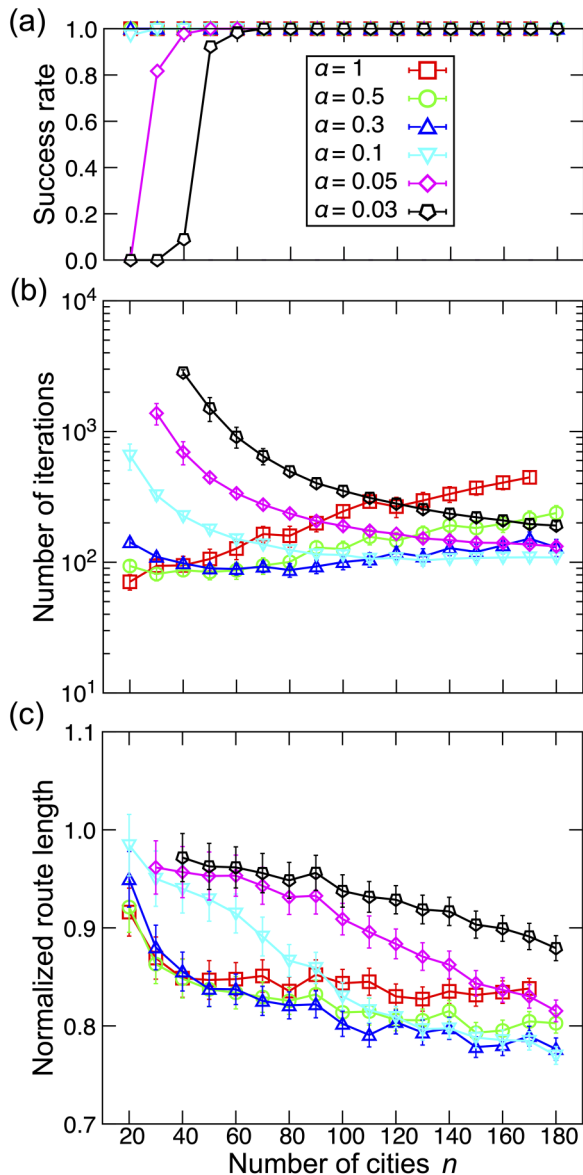


FIG. 4. Dependence of the solution-search performance on the number of cities n , i.e., (a) the success rate, (b) the average number of solution-search iterations, and (c) the normalized route length, of the Amoeba TSP recurrence formula model for several values of α .

feasible solutions for smaller city numbers ($n \leq 60$), because the elongation length of a branch in each iteration becomes too short, and thus the branches cannot satisfy the termination condition in Eq. (13) within 3000 iterations. In contrast, for larger n , feasible solutions are successfully obtained for all α values.

The n dependence of the average number of solution-search iterations exhibits different behaviors depending on α . It decreases (increases) monotonically with n for $\alpha \leq 0.1$ ($\alpha \geq 0.5$). In the former case, the elongation length of a branch increases proportionally with n , allowing the branch states to reach the termination condition within a smaller number of iterations. In the latter case, a larger number of iterations is required for the branches to satisfy the termination condition for large n , because the elongation length becomes

too long compared to that of contraction. For $\alpha = 0.3$, the number of solution-search iterations decreases (increases) monotonically with n for $n \leq 80$ ($n > 80$), showing intermediate behavior between the cases of $\alpha \leq 0.1$ and $\alpha \geq 0.5$. The normalized route length decreases monotonically with n for all parameter values. These results indicate that better solution-search performance for larger n can be achieved by appropriately tuning the value of α .

The proposed model can find a feasible solution with much smaller number of iterations compared with previous models, including the Amoeba TSP model and the Improved Amoeba TSP model. Owing to the acceleration of solution search, we can address the problems with up to 180 cities, whereas the Improved Amoeba TSP model and the circuit simulation of device system physically implementing the Amoeba TSP model were practically limited to around 100 cities and 30 cities, respectively, because of long computational time. Therefore, the obtained results motivate the physical implementation of amoeba-inspired combinatorial optimization machines by demonstrating the scalability of the proposed model that it is effective for larger number of cities by tuning the parameter value α .

C. Map dependence of the performance

We investigate the solution-search performance of the proposed model with $\alpha = 0.3$ using five maps constructed from intercity distances $d(U, V)$ generated according to different random distributions. All other parameter values are the same as those described in Sec. III. Here, we set the parameter value to $\alpha = 0.3$, because the solution-search iteration exhibits relatively weak dependence on the number of cities n for this value, which makes it suitable for clearly elucidating changes in performance due to other parameters. Here, $N(P, Q)$ denotes a Gaussian (normal) distribution with the mean P and the standard deviation Q , while $U(R, S)$ denotes a uniform distribution with the minimum R and the maximum S . The latter distribution has a mean $(R + S)/2$ and a standard deviation $(S - R)/2\sqrt{3}$. We set the parameter $a = 17\sqrt{3}$ so that the distribution $U(100 - a, 100 + a)$ has the same mean and standard deviation as $N(100, 17)$.

The proposed model can find a feasible solution in all trials for almost all maps and fails once only for the 180-city map constructed from $N(100, 34)$. The n dependence of the number of the solution-search iterations and that of the normalized route length are shown in Figs. 5(a) and 5(b), respectively. These profiles exhibit generally similar behavior across the five maps derived from different random distributions. The normalized route length becomes shorter as the standard deviation of the distance distribution increases. When the standard deviation is the same, the normalized route length is approximately identical (blue and red plots), suggesting that it is basically independent of the specific form of the distribution. The normalized route length approaches asymptotically to 0.9 in the smallest deviation cases (green and cyan plots). This behavior resembles that of the original Amoeba TSP model. Although the number of solution-search iterations shows a larger variance at $n = 160$ for the distribution $N(100, 34)$, the model successfully finds a feasible solution within 200

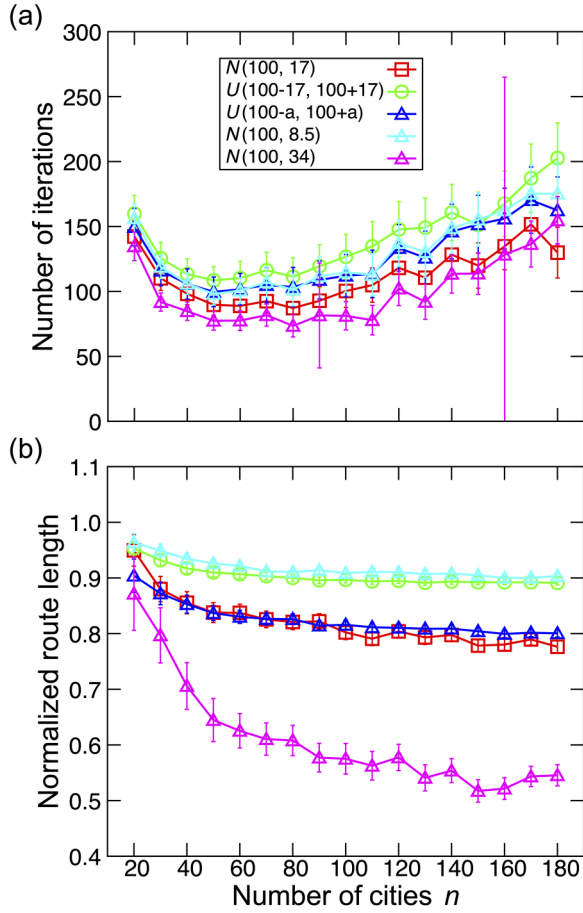


FIG. 5. Dependence of (a) the average number of solution-search iterations and (b) the normalized route length on the number of cities n for the Amoeba TSP recurrence formula model in Eq. (14) when $\alpha = 0.3$ for maps where distances between cities are generated from different random distributions.

iterations in 995 out of 1000 trials. Only for a few trials, more than 1000 iterations are required to obtain a feasible solution.

D. Fluctuation dependence of the performance

We examine the performance of the proposed model for various standard deviations D of the random fluctuation obeying a Gaussian distribution, that is, $D = 0.001, 0.003$, and 0.005 . All other parameter values are the same as those described in Sec. III. The value of α is set to 0.3, and a map constructed from $N(100, 34)$ is used. The proposed model can find a feasible solution in every trial irrespective of the value of D . Figure 6 shows the n dependence of the normalized route length. The value generally decreases with increasing n for all parameter values. These results imply that fine tuning of the deviation D results in the better solution-search performance. However, the influence of D on the performance is not elucidated yet, and therefore, further investigation is required to clarify this relationship, which is left for future study.

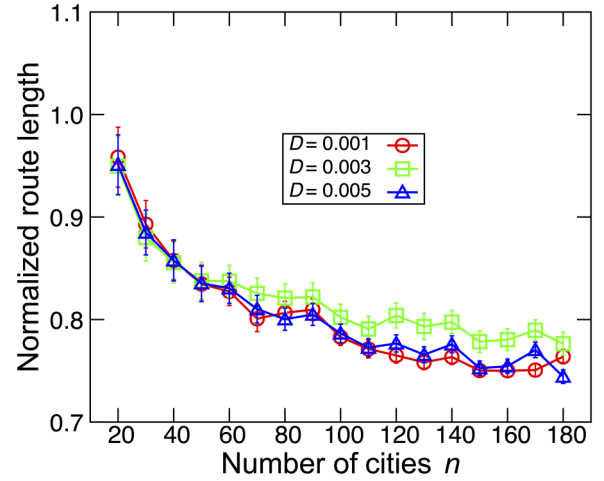


FIG. 6. Dependence of the normalized route length on the number of cities n for the Amoeba TSP recurrence formula model with $\alpha = 0.3$ in Eq. (14) for random fluctuations with different standard deviations D of the normal distribution.

VI. EQUIVALENCE WITH A RECURRENT NEURAL NETWORK

The Amoeba TSP recurrence formula in Eq. (14) is equivalent to the information-processing dynamics of a simple recurrent neural network, as illustrated in Fig. 7. The network consists of three layers: an input layer, a hidden layer, and an output layer. Each layer contains n^2 nodes corresponding to

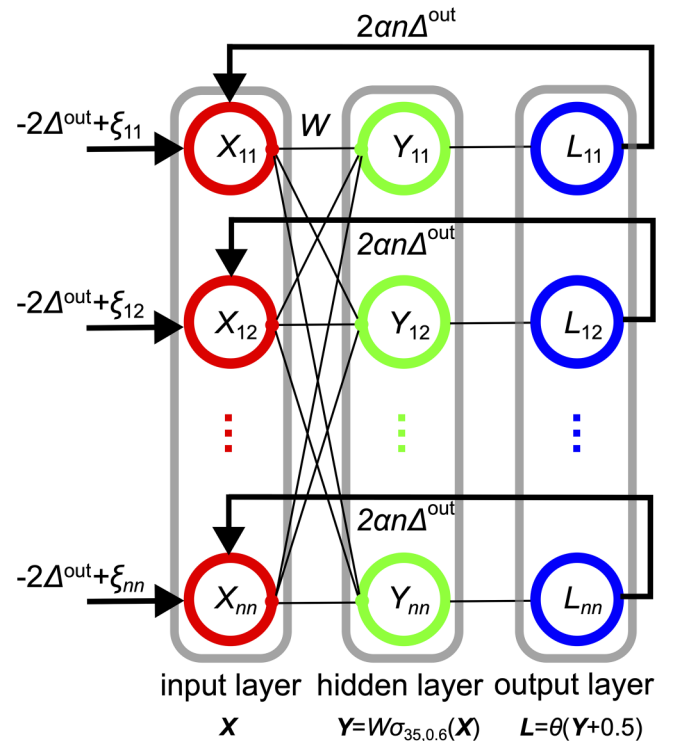


FIG. 7. Neural network representation of the information processing of Amoeba TSP recurrence formula model in Eq. (14). It is a recurrent neural network with the fixed weight because the outputs from the final layer are fed back to the input layer.

the number of lanes specified by two indices V and k . The input layer is fully connected to the hidden layer through the weight matrix $W = \{W_{Vl}\}$, which has $n^2 \times n^2$ elements. Here, W_{Vl} represents the connection between node Vl in the input layer and node Ul in the hidden layer.

We now explain the information-processing procedure in this neural network. First, the initial values $X_{Ul}(0) = 0$ are set and stored in all input-layer nodes Ul ($U, l = 1, 2, \dots, n$). Here, the stored values are assumed to decrease by constant value $2\Delta^{\text{out}}$ with time interval corresponding one iteration and to fluctuate with external noise. Each node applies the sigmoid function $\sigma_{35,0.6}(x)$ to its stored value and outputs $\sigma_{35,0.6}(X_{Ul}(0))$. The outputs are weighted by the corresponding elements of the weight matrix W_{Vl} , and the weighted sum $\sum_{U=1}^n \sum_{l=1}^n W_{Vl} \sigma_{35,0.6}(X_{Ul}(0))$ is fed into the node Vk in the hidden layer. The hidden nodes apply the step function $\theta(x + 0.5)$ to these input values, and the resulting binary outputs, which represent the illumination pattern, are sent to the corresponding nodes in the output layer. Each output-layer node Vk directly outputs the value

$$\theta \left(\sum_{U=1}^n \sum_{l=1}^n W_{Vl} \sigma_{35,0.6}(X_{Ul}(0)) + 0.5 \right). \quad (16)$$

This value is then multiplied by a constant factor $2\alpha n \Delta^{\text{out}}$ and fed back to the corresponding input-layer node Vk . The stored value in the input node is then updated from $X_{Vl}(0)$ to

$$X_{Vl}(1) = X_{Vl}(0) - 2\Delta^{\text{out}} + 2\alpha n \Delta^{\text{out}} \theta \left(\sum_{U=1}^n \sum_{l=1}^n W_{Vl} \sigma_{35,0.6}(X_{Ul}(0)) + 0.5 \right) + \xi_{Vl}. \quad (17)$$

This formula consists of the previous stored value, the feedback from the output layer, and a random fluctuation term. By iterating this update process for $X_{Vl}(t)$, the system evolves until the termination condition in Eq. (13) is satisfied.

The elements of the weight matrix W are fixed throughout the solution-search process, unlike in the training procedure of neural networks, because each element W_{Vl} represents either the cost associated with the distance between two cities or the penalty for violating the TSP constraints. Moreover, the nonlinearities introduced by the sigmoid and step functions play an essential role in the solution-search dynamics. Therefore, Amoeba TSP recurrence formula is equivalent to a recurrent neural network with fixed weights [31–33].

Owing to its characteristics, the proposed model has two advantages from the perspective of its physical implementation. The first advantage is that various physical devices can be employed for implementation, because the model has a simple information processing structure and is free from the constraint due to the volume-conservation law. By adopting suitable devices, combinatorial optimization machines with low energy consumption can be realized, in addition to the intrinsic efficiency derived from the survival strategies of slime molds. The second advantage is that rapidly advancing neural network circuit technologies can be directly applied to the implementation of Amoeba TSP recurrence formula, thereby increasing the feasibility of its realization. Moreover, this equivalence inspires research interest in understanding the

information-processing mechanisms of slime molds from the perspective of recurrent neural networks.

VII. SUMMARY AND DISCUSSION

In this paper, we constructed an optimization mathematical model inspired by the survival strategies of slime molds, named Amoeba TSP recurrence formula model, described by Eq. (14), aiming to realize combinatorial optimization machines with high computational efficiency and low power consumption. We emphasize that the improving the physical implementability of the model is the highest priority in our study to this end. Specifically, we derived the Amoeba TSP recurrence formula by applying the five modifications to the original Amoeba TSP model, including the three proposed in the previous work under the appropriate approximations to enhance the solution-search performance and to simplify the model form toward physical implementation. As a result, the Amoeba TSP model is reduced to a single recurrence formula in Eq. (14). The modifications are first chosen from the viewpoint of enhancing the physical implementability of the model and are further selected based on solution-search performance among several candidates. Notably, the conservation law imposed in the original Amoeba TSP model often restricts device candidates for physical implementation, and therefore it was removed in the modification. Moreover, we introduce a tunable parameter α to the proposed model, with which we can control and/or optimize the solution-search performance.

We studied the TSP as a representative and versatile combinatorial optimization problem and evaluated the solution-search performance of the proposed model via three criteria, that is, the success rate, the number of iterations required for the solution search, and the normalized route length. The Amoeba TSP recurrence formula with $\alpha = 1$ exhibits the better performance than those of the original Amoeba TSP model and the Improved Amoeba TSP model. We further examined the solution-search performance of the model for the several parameter values $\alpha = 0.03, 0.05, 0.1, 0.3$, and 0.5 . The model successfully finds feasible solutions in all trials for all α for the larger n . The number solution-search iteration decreases (increases) monotonically for the number of cities n for $\alpha \leq 0.1$ ($\alpha \geq 0.5$). The normalized route length decreases monotonically for the number of cities n for all parameter values. Therefore, we can perform the efficient solution search even for the larger number of cities by choosing an appropriate parameter value of α .

The proposed model achieves a feasible solution with a significantly smaller number of iterations than previous models, including the original Amoeba TSP model and the Improved Amoeba TSP model. Owing to this accelerated solution search, the proposed model can handle problems with up to 180 cities. In contrast, the Improved Amoeba TSP model and the circuit-level simulation of a device system physically implementing the Amoeba TSP model were practically limited to approximately 100 and 30 cities, respectively, due to computational time constraints. These results highlight the scalability of the proposed model and motivate the physical implementation of amoeba-inspired combinatorial optimization machines, demonstrating that the model remains effective

for larger problem sizes by appropriately tuning the parameter value α .

Moreover, we demonstrated that Amoeba TSP recurrence formula in Eq. (14) is equivalent to the information processing in the simple recurrent neural network. Notably, the weight matrix W in the neural network represents the distance between two cities and the penalty for breaking the constraints of TSP and thus is fixed throughout the solution-search procedure. The nonlinearity introduced by the sigmoid function and the step function also plays an important role for the solution search. Interestingly, this result implies that the information processings in slime molds can be understood from the perspective of recurrent neural networks.

We note that not all nonlinear maps with feedback can be regarded as a recurrent neural network. For an information-processing model to be regarded as a recurrent neural network, it must be composed of product-sum operations and activation functions (e.g., the sigmoid, ReLU, and tanh functions). The equivalence between our Amoeba TSP recurrence formula and the recurrent neural network is ensured because they have the same form in this regard. Note also that it is highly nontrivial that the solution-search process of a slime mold can be described by one recurrence formula composed of the product-sum operations and the sigmoid function. This finding indicates that the recurrent-neural-network structure is hidden behind the complex and elusive dynamics of slime molds when they solve the TSP.

From the above discussion, we succeed in enhancing the possibility for realization of combinatorial optimization machines with both the computational efficiency from the smart survival strategies of slime molds and the low power consumption with choosing appropriate hardware for the implementation. The insight established by the present study that the Amoeba TSP recurrence formula can be regarded as a form of recurrent neural network opens up new possibilities for its physical realization. Below, we discuss potential device candidates and future research directions for realizing such Amoeba-inspired combinatorial optimization machines. From the viewpoint of device implementation, several promising candidates can be considered. Photonic circuits offer ultra-fast parallel signal propagation with inherently low energy dissipation, enabling high-speed analog computation. Memristor crossbar arrays provide nonvolatile storage and tunable conductance, which can naturally emulate the weighted summation process in Eq. (14). Spintronic devices, such as magnetic domain-wall and skyrmion-based systems, are also attractive because their nonlinear dynamical responses and mutual interactions can mimic the sigmoid and threshold operations in the present model. Furthermore, mechanical or fluidic systems that directly reproduce the adaptive morphological behavior of slime molds may offer an intuitive and energy-efficient route to realization. Future studies will focus on constructing prototype physical implementations based on these device concepts and verifying their solution-search performance experimentally. Another promising direction is to explore hybrid approaches combining numerical simulations with the amoeba-inspired combinatorial optimization machine, which may further enhance both efficiency and scalability. We believe that the Amoeba TSP recurrence formula will serve as a bridge between soft-matter-inspired

computation and recurrent neural network, providing a universal framework for energy-efficient analog optimization.

ACKNOWLEDGMENTS

We are indebted to Masashi Aono for insightful discussions. This work was financially supported in part by a Waseda University Grant for Special Research Projects (Projects No. 2023E-026, No. 2024C-155, and No. 2025C-491) and Waseda Research Institute for Science and Engineering, Grant-in-Aid for Young Scientists (Early Bird) awarded to Y.M. and by Japan Society for the Promotion of Science KAKENHI (Grants No. JP24H02231 and No. JP25H00611), CREST, the Japan Science and Technology Agency (Grant No. JPMJCR20T1), and a Waseda University Grant for Special Research Projects (Projects No. 2023C-140, No. 2024C-153, and No. 2025C-133) awarded to M.M.

DATA AVAILABILITY

There are no publicly available research data or software supporting this manuscript. Requests for further information or data should be sent to the authors.

APPENDIX A: COMPUTATIONAL COST OF THE ORIGINAL AMOEBA TSP MODEL

In this Appendix, we explain the reason why the computational time for each iteration step is $O(n^4)$ for the original Amoeba TSP model and discuss how it can be reduced by the physical implementation. The solution search using the original Amoeba TSP model consists mainly of the following four processes, i.e., calculation of the illumination pattern, update of $L_{V_k}(t)$, deformation of the slime-mold body, and update of $X_{V_k}(t)$. To compute each $L_{V_k}(t)$, the summation $\sum_{V=1}^n \sum_{k=1}^n W_{V_k,UI} \sigma_{35,0.6}(X_{UI}(t))$ must be performed. This calculation requires n^2 sequential operations, since the sigmoid function $\sigma_{35,0.6}(x)$ is applied to each $X_{UI}(t)$, multiplied by the corresponding weight $W_{V_k,UI}$, and then summed. Because this computation is performed for n^2 values of $L_{V_k}(t)$, the total number of operations required for determining the illumination pattern is proportional to n^4 ($= n^2 \times n^2$). Therefore, when executed sequentially on a CPU without parallelization, the computational time scales as $O(n^4)$.

The deformation of the slime-mold body, i.e., the update of $X_{V_k}(t)$, mainly consists of the calculations of $O_{V_k}(t)$ and $I_{V_k}(t)$. The computation of $O_{V_k}(t)$ does not involve any product-sum operations and thus requires only $O(1)$ time for each pair (V, k) . In contrast, the computation of $I_{V_k}(t)$ requires evaluating the sum $\sum_{V=1}^n \sum_{k=1}^n O_{V_k}(t)$, which involves $L^{\text{off}}(t)$ sigmoid terms, when $L^{\text{off}}(t) \neq 0$. Since $L^{\text{off}}(t)$ is at most n^2 , the computation of $I_{V_k}(t)$ requires up to $O(n^2)$ operations for each (V, k) . If this calculation is naively performed for all n^2 variables $X_{V_k}(t)$ without exploiting the fact that $I_{V_k}(t)$ is independent of V and k , the total number of operations is again proportional to n^4 . Consequently, each iteration step of the solution search requires computational time proportional to $O(n^4)$, when performed sequentially on a single CPU.

As an illustrative example, let us consider the implementation by the analog electronic circuit proposed in Ref. [26]. In this implementation, the variables $L_{V_k}(t)$ and $X_{V_k}(t)$ are represented by terminal voltages, and a system with bidirectional feedback between $L_{V_k}(t)$ and $X_{V_k}(t)$ is constructed. When voltages corresponding to $X_{V_k}(t)$ (or $L_{V_k}(t)$) are simultaneously applied to the n^2 input terminals labeled by V and k , the voltages at the corresponding n^2 response terminals change in parallel to represent $L_{V_k}(t)$ (or $X_{V_k}(t)$). The solution search is performed by iterating this procedure while alternately switching the roles of the input and response terminals. Roughly speaking, the model is implemented by constructing a circuit in which the input-output relationship corresponds to the update equations of the proposed model. In this framework, the computational time can be significantly reduced because the variables $L_{V_k}(t)$ and $X_{V_k}(t)$ are updated simultaneously through physical phenomena, namely, the flow of electric current rather than being computed sequentially as in conventional numerical simulations on multiple CPUs.

APPENDIX B: DERIVATION OF THE AMOEBA TSP RECURRENCE FORMULA

In this Appendix, we present the derivation processes of the proposed model by applying modifications (1)–(5) to the Amoeba TSP model step by step. First, we introduce new binary variables $\tilde{L}_{V_k}(t)$ to represent the illumination pattern instead of $L_{V_k}(t)$ for notational simplicity:

$$\begin{aligned} \tilde{L}_{V_k}(t+1) &= 1 - \theta \left[\sigma_{1000, -0.5} \left(\sum_{U=1}^n \sum_{l=1}^n W_{V_k, U l} \sigma_{35, 0.6}(X_{U l}(t)) \right) - 0.5 \right] \\ &= 1 - \theta(0.5 - L_{V_k}(t+1)) = \theta(L_{V_k}(t+1) - 0.5). \end{aligned} \quad (\text{B1})$$

Here, $\tilde{L}_{V_k}(t) = 1(0)$ corresponds to the case that the lane (V, k) is (is not) to be illuminated. Using the variables $\tilde{L}_{V_k}(t)$, the update formula for $X_{V_k}(t)$ in Eq. (2) can be rewritten in a simpler form:

$$\begin{aligned} X_{V_k}(t+1) &= X_{V_k}(t) - O_{V_k}(t+1)\tilde{L}_{V_k}(t+1) \\ &\quad + I_{V_k}(t+1)(1 - \tilde{L}_{V_k}(t+1)) + \xi_{V_k}(t+1). \end{aligned} \quad (\text{B2})$$

The definition of $O_{V_k}(t)$, the branch length contracted by illumination, is reformulated to be independent of $X_{V_k}(t)$ by applying modification (1):

$$O_{V_k}(t+1) = 2\Delta^{\text{out}}\tilde{L}_{V_k}(t+1). \quad (\text{B3})$$

Owing to modification (2), the update formulas for $I_{V_k}(t)$ and $S(t)$ in Eqs. (6) and (7), originally defined for $L^{\text{off}}(t) \neq 0$, are extended to the case of $L^{\text{off}}(t) = 0$. The following equations are applied for all values of $L^{\text{off}}(t)$:

$$\begin{aligned} I_{V_k}(t+1) &= \frac{\Delta^{\text{in}} + \sum_{V=1}^n \sum_{k=1}^n O_{V_k}(t+1) + S(t)}{L^{\text{off}}(t+1)} \\ &= \frac{\Delta^{\text{in}} + \sum_{V=1}^n \sum_{k=1}^n O_{V_k}(t+1)}{L^{\text{off}}(t+1)} \\ &= \frac{\Delta^{\text{in}} + 2\Delta^{\text{out}}n^2}{L^{\text{off}}(t+1)} - 2\Delta^{\text{out}}, \end{aligned} \quad (\text{B4})$$

$$S(t) = 0. \quad (\text{B5})$$

Applying modification (3) to Eq. (6), $L^{\text{off}}(t)$ is replaced with n , and the equation is simplified as follows:

$$\begin{aligned} I_{V_k}(t+1) &\approx \frac{\Delta^{\text{in}} + 2\Delta^{\text{out}}n^2}{n} - 2\Delta^{\text{out}} \\ &= \frac{\Delta^{\text{in}}}{n} + 2\Delta^{\text{out}}n - 2\Delta^{\text{out}} \\ &\approx 2\Delta^{\text{out}}n - 2\Delta^{\text{out}}. \end{aligned} \quad (\text{B6})$$

Here, we approximate $\Delta^{\text{in}}/n \approx 0$ because $n \gg \Delta^{\text{in}} (= 0.001)$. The conservation law in Eq. (8) is no longer imposed owing to this modification.

Substituting the obtained expressions for O_{V_k} and I_{V_k} in Eqs. (B4) and (B8) into the update equation of X_{V_k} in Eq. (B3), we obtain

$$\begin{aligned} X_{V_k}(t+1) &= X_{V_k}(t) - O_{V_k}(t+1)\tilde{L}_{V_k}(t+1) + I_{V_k}(t+1) \\ &\quad \times (1 - \tilde{L}_{V_k}(t+1)) + \xi_{V_k}(t+1) \\ &= X_{V_k}(t) - 2\Delta^{\text{out}} + 2\Delta^{\text{out}}n(1 - \tilde{L}_{V_k}(t+1)) \\ &\quad + \xi_{V_k}(t+1). \end{aligned} \quad (\text{B7})$$

Here, we use the relation $L_{V_k}(t+1)^2 = L_{V_k}(t+1)$ for $L_{V_k}(t+1) = 0, 1$ when transforming from the second to the third line on the right-hand side. The fluctuation term $\xi_{V_k}(t)$ is generated from a normal random distribution, following modification (4). Substituting the definition of $\tilde{L}_{V_k}(t)$, the update equation can be rewritten as

$$X_{V_k}(t+1) = X_{V_k}(t) - 2\Delta^{\text{out}} + 2\Delta^{\text{out}}n\theta \left[\sigma_{1000, -0.5} \left(\sum_{U=1}^n \sum_{l=1}^n W_{V_k, U l} \sigma_{35, 0.6}(X_{U l}(t)) \right) - 0.5 \right] + \xi_{V_k}(t+1). \quad (\text{B8})$$

From modification (5) and the identity $\theta(\theta(x) - 0.5) = \theta(x + 0.5)$, the update equation for $X_{V_k}(t)$ becomes

$$X_{V_k}(t+1) = X_{V_k}(t) - 2\Delta^{\text{out}} + 2\Delta^{\text{out}}n\theta \left[\sum_{U=1}^n \sum_{l=1}^n W_{V_k, U l} \sigma_{35, 0.6}(X_{U l}(t)) + 0.5 \right] + \xi_{V_k}(t+1). \quad (\text{B9})$$

Finally, introducing a control parameter α , the equation can be written in a generalized form as

$$X_{V_k}(t+1) = X_{V_k}(t) - 2\Delta^{\text{out}} + 2\alpha\Delta^{\text{out}}n\theta \left[\sum_{U=1}^n \sum_{l=1}^n W_{V_k, U l} \sigma_{35, 0.6}(X_{U l}(t)) + 0.5 \right] + \xi_{V_k}(t+1). \quad (\text{B10})$$

APPENDIX C: POSSIBLE SPINTRONIC IMPLEMENTATION

The mathematical model of the amoeba-inspired combinatorial optimization machine proposed in this study is superior to conventional models in that it significantly facilitates physical implementation, which has previously been difficult to achieve. In particular, the absence of the constraint originating from the volume conservation law of the amoeba as well as the fact that Gaussian-distributed random fluctuations exhibit excellent solution-search performance greatly expands the range of candidate materials and physical phenomena that can be utilized for physical implementation. In this Appendix, as a representative example, we describe the concept of a physical implementation using spintronic devices based on current-driven domain-wall motion in ferromagnetic materials.

We first consider a device fabricated from a thin ferromagnetic film consisting of a circular disk from which multiple radial lanes extend as shown in Fig. 1. The red regions represent areas with out-of-plane upward magnetization, while the blue regions represent areas with out-of-plane downward magnetization. This device has the same geometrical structure as the confinement device for the amoeba shown in Fig. 1, where the red region can be regarded as corresponding to the body of the amoeba. We apply an external magnetic field to this ferromagnetic device. The magnetic field may be generated either by placing a permanent magnet near the device or by injecting an electric current through a conductive coil located above or below the device. When an upward magnetic field is applied, the region with upward magnetization (red region) expands in order to increase the Zeeman energy gain. In particular, the domain walls separating the central circular region and the lane regions move away from the center of the disk, resulting in the expansion of the red regions into the lanes. This behavior corresponds to the extension of amoeba body into the lanes during the process of exploring nutrient sources.

In the amoeba-based combinatorial optimization, when a candidate solution violates constraints or increases the cost function, the amoeba legs extending into the corresponding lanes are selectively retracted according to the prescribed rules. This operation is experimentally realized by illuminating the corresponding lanes with light that the amoeba avoids. An analogous function can be readily implemented using ferromagnetic domain walls. Specifically, by attaching electrodes to the ends of the lanes and injecting an electric current into selected lanes, the domain walls can be driven so as to retract the upward magnetized region within those lanes. This motion originates from the spin-transfer torque mechanism, in which spin angular momentum carried by conduction electrons is transferred to the local magnetization, causing translational motion of domain walls. By utilizing this mechanism, domain walls that have advanced under an applied magnetic field can be selectively pushed back by electric currents.

An important point is that the expansion and contraction of the upward magnetized regions under an applied magnetic field do not obey a volume conservation law. In contrast, in real amoebae, the extension of legs in some lanes is accompanied by contraction in others such that the total volume

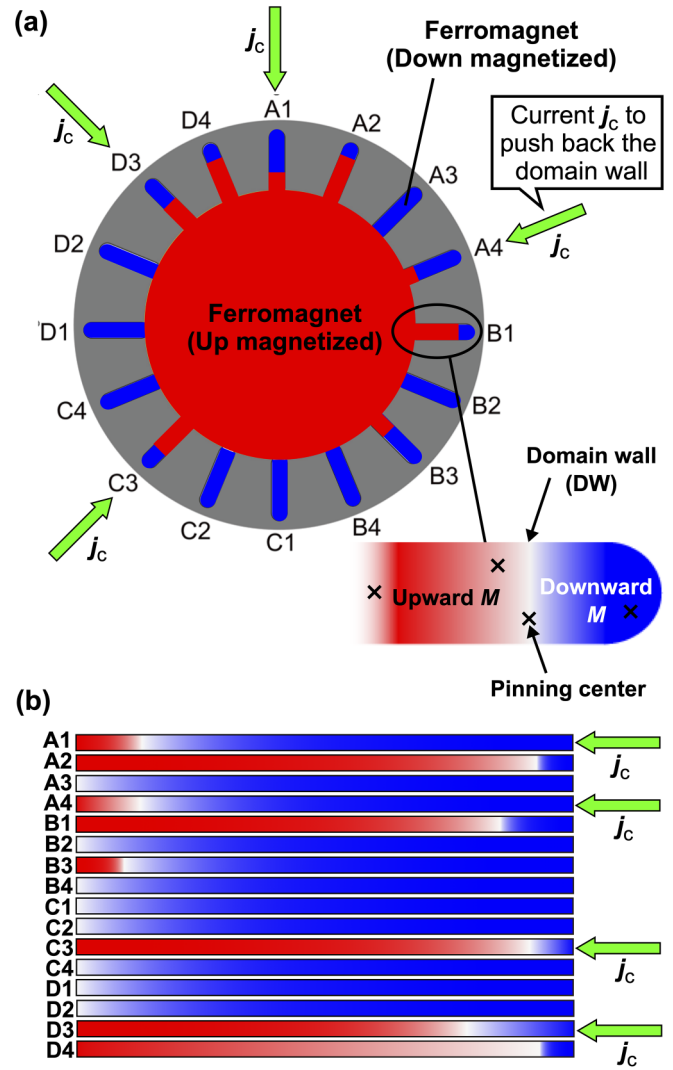


FIG. 8. (a) Schematics of the spintronic implementation of the Amoeba TSP recurrence formula model based on current-driven domain-wall motion in ferromagnetic materials. (b) Possible simplification of the device form with parallel configuration of multiple ferromagnetic nanowires.

remains conserved. In ferromagnetic systems, however, domain-wall motion in each lane occurs independently, and the total magnetized volume is not conserved. Therefore, if the optimization algorithm required strict volume conservation, physical implementation using ferromagnetic materials would be difficult. In our proposed model, however, volume conservation is not required, which allows irreversible state changes driven by external fields toward lower-energy configurations, which is a ubiquitous feature of physical phase-evolution phenomena, to be directly exploited for information processing.

Furthermore, the absence of volume-conservation constraints implies that the radial geometry shown in Fig. 8(a) is not essential. For example, as shown in Fig. 8(b), a device consisting of parallel ferromagnetic nanowires can also perform the same information-processing function. Such structures are advantageous in terms of scalability as they enable

high-density integration and three-dimensional architectures more easily than radial geometries. In addition, domain walls driven by external magnetic fields or electric currents are often pinned by impurities and defects inherently present in ferromagnetic materials. At finite temperatures, however, thermal fluctuations enable domain walls to escape from pinning sites probabilistically and resume motion. This thermally activated

depinning process provides a natural physical origin for the Gaussian random fluctuations incorporated in our model.

To summarize this Appendix, the model proposed in this study greatly broadens the range of materials and physical phenomena that can be used for physical implementation and provides a powerful foundation for realizing amoeba-inspired combinatorial optimization machines.

-
- [1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed. (Morgan Kaufmann Publishers, Boston, MA, 2017).
 - [2] T. Kadowaki and H. Nishimori, Quantum annealing in the transverse Ising model, *Phys. Rev. E* **58**, 5355 (1998).
 - [3] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, Quantum annealing with manufactured spins, *Nature (London)* **473**, 194 (2011).
 - [4] A. Lucas, Ising formulations of many NP problems, *Front. Phys.* **2**, 1 (2014).
 - [5] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing, *IEEE J. Solid-State Circuits* **51**, 303 (2016).
 - [6] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, A coherent Ising machine for 2000-node optimization problems, *Science* **354**, 603 (2016).
 - [7] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, Physics-inspired optimization for quadratic unconstrained problems using a digital annealer, *Front. Phys.* **7**, 48 (2019).
 - [8] H. Goto, K. Tatsumura, and A. R. Dixon, Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems, *Sci. Adv.* **5**, eaav2372 (2019).
 - [9] D. Kessler, Plasmodial structure and motility, in *Cell Biology of Physarum and Didymium*, edited by H. C. Aldrich and J. W. Daniel (Academic Press, New York, 1982), Vol. 1, p. 145.
 - [10] A. Adamatzky (ed.), *Advances in Physarum Machines: Sensing and Computing with Slime Mould* (Springer, Cham, 2016).
 - [11] K. Iwayama, L. Zhu, Y. Hirata, M. Aono, M. Hara, and K. Aihara, Decision-making ability of *Physarum polycephalum* enhanced by its coordinated spatiotemporal oscillatory dynamics, *Bioinspiration Biomimetics* **11**, 036001 (2016).
 - [12] S. Bajpai, M. Aono, and P. Kurian, Tracking and distinguishing slime mold solutions to the traveling salesman problem through synchronized amplification in the non-equilibrium steady state, *arXiv:2504.03492*.
 - [13] T. Nakagaki, H. Yamada, and Á. Tóth, Maze-solving by an amoeboid organism, *Nature (London)* **407**, 470 (2000).
 - [14] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki, Rules for biologically inspired adaptive network design, *Science* **327**, 439 (2010).
 - [15] M. Aono, M. Hara, and K. Aihara, Amoeba-based neurocomputing with chaotic dynamics, *Commun. ACM* **50**, 69 (2007).
 - [16] M. Aono, Y. Hirata, M. Hara, and K. Aihara, Amoeba-based chaotic neurocomputing: Combinatorial optimization by coupled biological oscillators, *New Gener. Comput.* **27**, 129 (2009).
 - [17] L. Zhu, M. Aono, S.-J. Kim, and M. Hara, Amoeba-based computing for traveling salesman problem: Long-term correlations between spatially separated individual cells of *Physarum polycephalum*, *Biosystems* **112**, 1 (2013).
 - [18] M. Aono, Amoeba-inspired combinatorial optimization machines, *Jpn. J. Appl. Phys.* **59**, 060502 (2020).
 - [19] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study* (Princeton University Press, Princeton, NJ, 2006).
 - [20] M. Aono, S.-J. Kim, L. Zhu, M. Naruse, M. Ohtsu, H. Hori, and M. Hara, Amoeba-inspired SAT solver, in *Proceedings of the 2012 International Symposium on Nonlinear Theory and its Applications (NOLTA2012)* (IEICE, Majorca, 2012), pp. 586–589.
 - [21] S. Kasai, M. Aono, and M. Naruse, Amoeba-inspired computing architecture implemented using charge dynamics in parallel capacitance network, *Appl. Phys. Lett.* **103**, 163703 (2013).
 - [22] Y. Hara-Azumi, N. Takeuchi, K. Hara, and M. Aono, Digital bio-inspired satisfiability solver leveraging fluctuations, *Jpn. J. Appl. Phys.* **59**, 040603 (2020).
 - [23] M. Aono, M. Naruse, S.-J. Kim, M. Wakabayashi, H. Hori, M. Ohtsu, and M. Hara, Amoeba-inspired nanoarchitectonic computing: Solving intractable computational problems using nanoscale photoexcitation transfer dynamics, *Langmuir* **29**, 7557 (2013).
 - [24] N. Takeuchi, M. Aono, and N. Yoshikawa, Superconductor amoeba-inspired problem solvers for combinatorial optimization, *Phys. Rev. Appl.* **11**, 044069 (2019).
 - [25] M. Aono, S. Kasai, S.-J. Kim, M. Wakabayashi, H. Miwa, and M. Naruse, Amoeba-inspired nanoarchitectonic computing implemented using electrical Brownian ratchets, *Nanotechnology* **26**, 234001 (2015).
 - [26] K. Saito, M. Aono, and S. Kasai, Amoeba-inspired analog electronic computing system integrating resistance crossbar for solving the travelling salesman problem, *Sci. Rep.* **10**, 20772 (2020).
 - [27] K. Saito and S. Kasai, Effect of feedback delays on solution quality in amoeba-inspired computing system that solves traveling salesman problem, *Appl. Phys. Express* **13**, 114501 (2020).
 - [28] L. Zhu, S.-J. Kim, M. Hara, and M. Aono, Remarkable problem-solving ability of unicellular amoeboid organism and its mechanism, *R. Soc. Open Sci.* **5**, 180396 (2018).

- [29] Y. Miyajima and M. Mochizuki, Proposed modified computational model for the amoeba-inspired combinatorial optimization machine, [NOLTA, IEICE 15, 824 \(2024\)](#).
- [30] J. J. Hopfield and D. W. Tank, Computing with neural circuits: A model, [Science 233, 625 \(1986\)](#).
- [31] H. Jaeger, The “echo state” approach to analysing and training recurrent neural networks-with a erratum note, GMD Technical Report 148, German National Research Center for Information Technology, Bonn, Germany, 2001, p. 34.
- [32] K. Nakajima, Physical reservoir computing—An introductory perspective, [Jpn. J. Appl. Phys. 59, 060501 \(2020\)](#).
- [33] K. Nakajima and I. Fischer, *Reservoir Computing: Theory, Physical Implementations, and Applications* (Springer, Berlin, 2021).