Amoeba-inspired Tug-of-War algorithms for exploration–exploitation dilemma in extended Bandit Problem

Masashi Aono, Song-Ju Kim, Masahiko Hara, Toshinori Munakata

A R T I C L E   I N F O

Article history:
Received 23 March 2013
Received in revised form 18 December 2013
Accepted 21 December 2013

Keywords:
Natural computing
Multi-armed Bandit Problem
Decision making
Resource allocation
Physarum polycephalum

A B S T R A C T

The true slime mold Physarum polycephalum, a single-celled amoeboid organism, is capable of efficiently allocating a constant amount of intracellular resource to its pseudopod-like branches that best fit the environment where dynamic light stimuli are applied. Inspired by the resource allocation process, the authors formulated a concurrent search algorithm, called the Tug-of-War (TOW) model, for maximizing the profit in the multi-armed Bandit Problem (BP). A player (gamblers) of the BP should decide as quickly and accurately as possible which slot machine to invest in out of the N machines and faces an "exploration–exploitation dilemma." The dilemma is a trade-off between the speed and accuracy of the decision making that are conflicted objectives. The TOW model maintains a constant intracellular resource volume while collecting environmental information by concurrently expanding and shrinking its branches. The conservation law entails a nonlocal correlation among the branches, i.e., volume increment in one branch is immediately compensated by volume decrement(s) in the other branch(es). Owing to this nonlocal correlation, the TOW model can efficiently manage the dilemma. In this study, we extend the TOW model to apply it to a stretched variant of BP, the Extended Bandit Problem (EBP), which is a problem of selecting the best M-tuple of the N machines. We demonstrate that the extended TOW model exhibits better performances for 2-tuple-3-machine and 2-tuple-4-machine instances of EBP compared with the extended versions of well-known algorithms for BP, the ε-Greedy and SoftMax algorithms, particularly in terms of its short-term decision-making capability that is essential for the survival of the amoeba in a hostile environment.

© 2013 Elsevier Ireland Ltd. All rights reserved.

1. Introduction

The speed and accuracy of the decision making are crucial but conflicted objectives for resource-limited gamblers and organisms to survive in uncertain environments. The multi-armed Bandit Problem (BP) (Sutton and Barto, 1998), a problem of finding the most rewarding machine from N slot machines, is a good example of why and how the difficulty of the decision making should be overcome in real-world situations. Suppose each machine i emits a reward, for example, a coin, with an individual probability $p_i$ but no player knows the reward probabilities of all the machines in advance. A player tries to maximize the total reward sum obtained after playing the machines for a certain number of trials. The player needs to drop coins into a bank of machines to correctly evaluate the best machine, but has to complete the evaluation quickly to minimize the waste. Thus, the player confronts the "exploration–exploitation dilemma" created with incompatible demands: one either "exploits" the rewards obtained using already collected knowledge or "explores" new alternatives for acquiring higher rewards involving risks. How can we optimize these conflicting objectives? The optimal solution to this problem is to determine the optimal strategy, i.e., an efficient algorithm for

* Corresponding author at: Earth-Life Science Institute, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan. Tel.: +81 357342854; fax: +81 357344116.
E-mail address: masashi.aono@elisi.jp (M. Aono).

1 Although the original definition of BP is stated with the term “bandit,” here we explain BP using “N slot machines” instead of “N-armed bandit” because a slot machine is equivalent to one-armed bandit.

2 In this study, we deal with the simplified variant of the general BP. We assume that all slot machines can only return a uniform reward, (i.e., at most a coin for a play), but their probabilities to emit the reward are different.
selecting the machine that yields maximum rewards by referring to past experiences.

Many algorithms for tree searches are subjected to this dilemma in practical situations. Thus, efficient algorithms for BP are useful for a wide range of applications requiring powerful tree search capability. We believe that living organisms generally encounter a similar trade-off between the speed and accuracy of their decision making and would have developed some efficient methods to overcome the dilemma for their survival in an unknown world.

A plasmodium of the true slime mold Physarum polycephalum (Fig. 1A) is an amoeboid multi-nucleated unicellular organism that has been studied actively in terms of its sophisticated computational capabilities. Nakagaki et al. showed that the organism is capable of connecting the optimal routes among foods, despite the absence of a central nervous system (Nakagaki et al., 2000). This capability can be applied to solving some geometric path planning problems (Tero et al., 2006, 2010; Nakagaki et al., 2000) As shown in Fig. 1B, when the organism is placed in a four-lane stellate chip resting on an agar plate, it elongates its four terminal branches and changes its shape by expanding or shrinking the branches. Although the shape of the organism can deform arbitrarily, its total volume remains almost constant during our experimental observations. Thus, the shape-changing behavior of the organism can be considered to be a process for allocating a constant amount of intracellular resource to its branches.

Fig. 1c shows a schematic illustration of the architecture of the body of the organism (Kessler, 1982). In the body, the intracellular sol (resource) shuttles through tubular channels (branches), while the extracellular gel layer, like a sponge, rhythmically oscillates contraction tension to squeeze and absorb the sol. Depending on the phase difference of the oscillation, the contraction tension of the gel layer varies from site to site, and the sol is made to flow along the pressure difference (gradient) produced by this difference in local contraction tension. Each branch expands further as the sol influx raises the growth rate of the volume of the branch.

The branch of the organism shrinks when illuminated by visible light. The light stimuli are believed to enhance the contraction tendency of the stimulated gel layer and intensify the sol efflux (extrusion) from the illuminated branch. This negative phototactic behavior enables us to make the organism withdraw its branches from illuminated lanes, and expand toward non-illuminated lanes.

By introducing an optical feedback system (Fig. 1c), which automatically updates the illumination condition with changes in the shape of the organism, Aono et al. created a biocomputer that exploits the capability of the organism to search for solutions to combinatorial optimization problems (Aono and Gunji, 2003; Aono et al., 2007; Aono and Haro, 2008). The amoeba-based computer can be used to solve the N-city traveling salesman problem (TSP) when we place the organism in a N2–lane stellate chip and update the illumination condition according to a recurrent neural network model (Hopfield and Tank, 1986). The optimal solution is obtained when the organism succeeds in exclusively elongating the correct combination of the N branches to maximize its body area for maximal nutrient absorption from the agar plate and minimize the risk of being illuminated. The amoeba-based computer showed a high probability of deriving the optimal solution to the TSP (Aono et al., 2009a; Zhu et al., 2013). These results indicate the optimization capability of the organism to allocate efficiently the intracellular resource to the elongated branches that are most adaptive to the dynamic illumination condition.

To extract the physical essence of the resource allocation process of amoeba, Kim et al. formulated a discrete-time-state dynamical system, called the Tug-of-War (TOW) model (Kim et al., 2009, 2010a, b, 2011: Kim and Aono, 2014). As in the illustration shown in Fig. 1c, the TOW model is a star network consisting of amoeba-like branches connected to a hub node, where the total sum of volumes is kept constant.

The TOW model can be used as an algorithm to solve BP. In the TOW model, each branch plays machine i when the volume displacement of branch i is positive. The positive branch i is stimulated by light with a probability 1 − p_i, i.e., “punished” instead of being “rewarded” with a probability p_i. Namely, BP is translated equivalently into a problem of finding the least frequently punished lane. This problem setting can be compared to a situation created in the amoeba-based computer in which the amoeba changes its shape to maximize the energy (nutrient) acquisition from the agar plate by elongating the most appropriate branches that can minimize the probability of being illuminated. The amoeba needs to elongate its branches in the least frequently illuminated lanes because the branches consume certain energy for their withdrawal movements when illuminated. To correctly evaluate the least frequently illuminated lanes, the amoeba has to elongate its branches in all the lanes. However, at the same time, the amoeba has to complete the evaluation quickly to minimize the energy consumption caused by the elongation movements. Thus, the amoeba encounters the exploration–exploitation dilemma, which arises as a difficulty of achieving the maximal energy acquisition through the minimal energy consumption.
The TOW model is a concurrent algorithm in which the branches play more than one machine simultaneously at each time step, whereas in any sequential algorithm the player is allowed to play only one machine during each trial according to the original definition of BP. To make unbiased comparisons among the performances of the sequential and concurrent algorithms, if the concurrent algorithm played \( Y \leq N \) machines at a time, the scores should be evaluated as the results obtained after \( Y \) trials. Even under this comparison guideline, the TOW model was evaluated as more efficient and adaptive for 2-machine and 3-machine instances compared with well-known sequential algorithms such as the modified \( \epsilon \)-Greedy algorithm and modified SoftMax algorithm (Kim et al., 2009, 2010a,b).

In our previous studies on amoeba-based computing for the \( N \)-city TSP, the problem was to find the optimal combination of the \( N \) branches out of \( N^2 \) branches, and the organism succeeded in elongating the optimal branches with a high probability (Aono et al., 2009a; Zhu et al., 2013). We expect that the TOW model can be extended to predict the elongation of more than one branch efficiently and can be used to explore the essential dynamics that produce the problem-solving capability of the organism. In this study, we revise the settings of the BP and consider a new problem termed the “Extended Bandit Problem (EBP),” a problem of finding the optimal \( M \)-tuplet out of \( N \) machines.

EBP represents a common situation in which one should determine the optimal strategy for allocating resource to a specified number of options simultaneously. An example is a situation that occurs in the internet advertising. An ad agent should maximize the number of total click-through counts of banner advertisements of \( N \) companies, but in a targeted website there are only \( M \) spaces for the banners to be placed simultaneously. The agent, therefore, should try many combinations of the \( M \) banners and find the maximally clicked one. Another example of EBP is the optimization of the usability of telecommunication lines in terms of cognitive radio technology. Suppose there are \( N \) lines that are preferentially allocated to licensed users, where each line is assumed to be occupied with a fixed probability. To make effective use of communication resources, cognitive radio technology allows public access to unoccupied lines for \( M \) unlicensed users. In this case, the problem is to find a combination of \( M \) lines that are the least frequently occupied ones. Thus, efficient algorithms for EBP will be useful for a lot of real-world applications.

This paper is organized as follows. In the next section, we review the definition of the original TOW model referred to as TOW1, after introducing two well-known sequential strategies for the original BP, i.e., the modified \( \epsilon \)-Greedy algorithm (EGR1) and modified SoftMax algorithm (SMX1). Then we extend these three algorithms so that they can be applied to EBP. The extended algorithms are referred to as TOW2, EGR2, and SMX2, respectively. In Section 3, first we show the results on the performance comparisons among TOW1, EGR1, and SMX1 for 3-machine and 4-machine instances of BP. Then we compare the performances of TOW2, EGR2, and SMX2 for 2-tuple-3-machine and 2-tuple-4-machine instances of EBP. TOW2 exhibits better short-term decision capability than EGR2 and SMX2. We conclude this paper, by discussing some implications of the results from biological and physical perspectives.

2. Models

2.1. Multi-armed Bandit Problem

The multi-armed Bandit Problem (BP) is stated as follows. Suppose there are \( N \) slot machines, and their probabilities of emitting a reward are independent and unknown to a player. Let \( \mathcal{I} = \{1, 2, \ldots, N\} \) be a set of all slot machines and \( p_i \in [0.0, 1.0] \) be a reward probability of machine \( i \in \mathcal{I} \). Here we assume that all the machines release at most a unit of reward, for example, a coin. The player should maximize the total sum of rewards obtained after playing the machines for a certain number of trials. Thus, the problem is to find as quickly and accurately as possible the optimal machine \( i^\star \) such that its reward probability is maximum, i.e., \( i^\star = \arg\max_{i \in \mathcal{I}} p_i \).

BP was originally described by Robbins (1952), although the same problem, in essence, was studied by Thompson (1933). A different version of the Bandit Problem has also been studied where the reward distributions are assumed to be known to the player. In this version, the optimal strategy is known only for a limited class of problems (Gittins and Jones, 1974; Gittins, 1979). In the original version, a popular measure for the performance of an algorithm is “regret,” i.e., the expected loss of rewards for not selecting the correct machine at all times. Lai and Robbins first showed that regret has to increase at least logarithmically in the number of selections (Lai and Robbins, 1985). They defined the condition where an optimal strategy must satisfy asymptotically. However, the computation of their algorithm is generally difficult due to the Kullback–Leibler divergence. Agrawal proposed algorithms where the index could be expressed as a simple function of the total reward sum obtained from a machine (Agrawal, 1995). These algorithms are considerably easier to compute than those developed by Lai and Robbins; however, the regret retains the asymptotic logarithmic behavior albeit with a larger leading constant. Auer et al. proposed a simple algorithm called Upper Confidence Bound 1 (UCB1) that achieved logarithmic regret uniformly over time, rather than only asymptotically (Auer et al., 2002). In addition, they proved that some family of the modified \( \epsilon \)-Greedy algorithm (the \( \epsilon \)-decreasing algorithm) also achieves logarithmic regret. Vermorel and Mohri concluded that the most naive approach, the modified \( \epsilon \)-Greedy algorithm with carefully chosen parameter, is the best (Vermorel and Mohri, 2005). It is also known that the performance of the modified SoftMax algorithm (the decreasing SoftMax algorithm) is comparable to that of the modified \( \epsilon \)-Greedy algorithm (Vermorel and Mohri, 2005). Therefore, we evaluate these two algorithms for the performance comparisons in this study.

\( 2.1.1. \text{ EGR1: modified } \epsilon \text{-Greedy algorithm} \)

The greedy algorithm is one of the most popular sequential algorithms for solving BP (Sutton and Barto, 1998). In this algorithm, a player switches between “random exploration” and “greedy exploitation” in a probabilistic manner.

We write \( u_i^t = 1 \) when machine \( i \) is played at time \( t \), i.e., a player drops a coin into machine \( i \).

\[
u_i^t = \begin{cases} 
1 & \text{if machine } i \text{ is played,} \\
0 & \text{otherwise.} 
\end{cases} \tag{1}
\]

When the player obtains a reward released from machine \( i \), we write this as \( u_i^t = 1 \).

\[
u_i^t = \begin{cases} 
1 & \text{if rewarded,} \\
0 & \text{otherwise.} 
\end{cases} \tag{2}
\]

The player accumulates past experiences and calculates the estimates \( g_i^t \in \mathbb{R}^+ \) for all the machines:

\[
g_i^t = \frac{R_{i}^{t}}{U_{i}^{t}}, \tag{3}
\]

\[
U_{i}^{t+1} = U_{i}^{t} + u_i^t, \tag{4}
\]
$R^i_t = R^{i-1}_t + r^i_t$.  

The player randomly “explores” which machine to play with the probability $\epsilon$ or performs a “greedy” action with the probability $1 - \epsilon$. In the greedy mode, the player selects the known best machine $i$ which has the highest value, i.e., $i = \arg\max_{j \in I} (g^j_t)$. When machine $i$ is played ($u^i_t = 1$), other ones cannot be played ($u^k_t = 0$ for all $k \in I \setminus \{i\}$).

In the original $\epsilon$-Greedy algorithm, $\epsilon \in [0.0, 1.0]$ is constant. However, in this study, we use the time-dependent $\epsilon(t)$ given as follows:

$$\epsilon^t = \frac{1}{1 + \tau \cdot t},$$  

where $\tau$ is the decay rate.\(^3\) Hereafter we refer to this modified $\epsilon$-Greedy algorithm as EGR1.

2.1.2. SMX1: modified SoftMax algorithm

The SoftMax algorithm is another well-known sequential algorithm for BP. Some studies have reported that this is the best algorithm in the context of decision making (Daw et al., 2006; Cohen et al., 2007).

In this algorithm, the player always selects which machine to play in a probabilistic manner. The probability of selecting machine $i$, $e^i_t \in [0.0, 1.0]$, is given by the following Boltzmann distributions:

$$e^i_t = \frac{\exp(\beta \cdot g^i_t)}{\sum_k \exp(\beta \cdot g^k_t)}$$  

where $\beta$ is the growth rate, and the estimate $g^i_t$ defined by Eq. (3) is the same as the one used in EGR1. Note that $\sum_i e^i_t = 1$.

Similar to $\epsilon$ in EGR1, $\beta$ was modified to a time-dependent form in this study as follows:

$$\beta^t = \beta \cdot t,$$  

where $\beta = 0$ corresponds to a random selection, and $\beta \to \infty$ corresponds to a greedy algorithm. We call this modified SoftMax algorithm SMX1.

2.1.3. TOW1: Tug-of-War model

The original Tug-of-War model, TOW1, represents the amoeba’s body as a star network consisting of $N$ terminal branches connected to a hub node (Fig. 1c). For each branch $i \in I$ at time $t$, let $x^i_t \in \mathbb{N}$ be the displacement of the volume from an arbitrarily assumed normal value, where that of the hub node is denoted by $x^0_t$.

We consider that branch $i$ pulls the lever of slot machine $i$ when taking a positive volume displacement $\theta(x) = \begin{cases} 1 & \text{(if $x > 0$),} \\ 0 & \text{(otherwise).} \end{cases}$

where $\theta$ is a step function:

$$\theta(x) = \begin{cases} 1 & \text{(if $x > 0$),} \\ 0 & \text{(otherwise).} \end{cases}$$  

Because the amoeba is allowed to play up to $N$ machines at a time $t$, there exist $2^N$ possible moves that are included in a set $\{(\theta(x_1), \theta(x_2), \ldots, \theta(x_N)) | \theta(x_k) \in \{0, 1\}\}$.

If machine $i$ is played, branch $i$ is stimulated by light with the probability $1 - p_i$, as a “punishment,” i.e., an effect opposite to a “reward.” For each branch $i$, the light condition $l^i_t$ is given as follows:

$$l^i_t = \begin{cases} 1 & \text{(if $\theta(x^i_t) = 1$), light ON with a probability $1 - p_i$),} \\ -1 & \text{(otherwise, light OFF).} \end{cases}$$  

\(^3\) Instead of $\epsilon^t = \min\{1, 1/(t+1)\}$, which is well known as the “$\epsilon$-decreasing algorithm,” we used this $\epsilon^t$ for simplicity.

### Table 1

<table>
<thead>
<tr>
<th>Acceleration $a^i$ defined by Eq. (13),</th>
<th>$s^i_t &gt; 0$</th>
<th>$s^i_t = 0$</th>
<th>$s^i_t &lt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l^i_t = -1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$l^i_t = 1$</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

### Table 2

Signals from the reward detector $\rho^i$ and punishment detector $\pi^i$

<table>
<thead>
<tr>
<th>State</th>
<th>Rewarded</th>
<th>Punished</th>
<th>Not played</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho^i_t$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\pi^i_t$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The volume displacement $x^i_t$ is updated according to the following difference equations:

$$x^i_{t+1} = \begin{cases} x^i_0 + v^i_t & \text{(i.e., $i \in I$),} \\ x^i_0 - \sum_{j=1}^{N} v^j_t & \text{(i.e., $i = 0$),} \end{cases}$$  

$$v^i_t = v^{i-1}_t + v^i_t,$$  

where $v^i_t$ and $v^i_t$ denote the velocity and acceleration of the corresponding volume displacement, respectively. Owing to Eq. (11), the conservation law holds as $\sum_{i=0}^{N} x^i_t = \sum_{i=0}^{N} x^i_0 = \text{const., i.e., the total volume remains unchanged from the initial condition}.$

### The acceleration $a^i_t$ is determined by the following function that is equivalently expressed by Table 1:

$$a^i_t = -l^i_t \cdot (1 - \theta[l^i_t \cdot s^i_t]),$$

where $s^i_t$ denotes the deviation of the amount of intracellular sol allocated to branch $i$. The positive acceleration ($a^i_t = 1$) drives branch $i$ to grow, whereas the negative acceleration ($a^i_t = -1$) results in the withdrawal of the branch. Table 1 shows that branch $i$ grows when the light stimulation is not applied ($l^i_t = 1$) and the sol is sufficiently allocated ($s^i_t > 0$). The sol, therefore, can be considered to be a resource required for growth. The photoavoidance behavior of the branch is expressed as the negative acceleration ($a^i_t = -1$) in response to the illumination ($l^i_t = 1$). However, if the sol is abundant ($s^i_t > 0$), the branch cannot withdraw even when illuminated, because the acceleration cannot be negative ($a^i_t = 0$). The movements of the branches, therefore, cannot be determined solely by the external light stimuli. The decision to expand and shrink particular branches is jointly determined by the external stimuli and internal resource-allocating dynamics.

The sol deviation $s^i_t$ is defined as a function of the hub volume displacement $x^0_t$ and the accumulated information $q^i_t$ regarding past events:

$$s^i_t = x^0_t + q^i_t - \mathrm{mean}_{j \in I} |q^{i-1}_j|,$$  

$$q^i_t = q^{i-1}_t + \mu \cdot (\rho^i_t + \omega \cdot \sum_{j=1}^{N} \pi^j_t),$$

where $\rho^i_t = \theta(x^i_t) - \theta(l^i_t)$ detects the rewarded (unpunished) play, $\pi^j_t = \theta(l^j_t)$ detects the punished play, $\mu \in \mathbb{R}^+$ is a parameter for adjusting the accumulation rate, and $\omega \in \mathbb{R}^+$ is a parameter for emphasizing information transmitted from punished branches. Table 2 shows how the reward detector $\rho^i_t$ and punishment detector $\pi^i_t$ operate. The parameter $\omega$ is always set to $\omega = 1$ in this study.

The intrinsic dynamics of TOW1 are deterministic. However, the accelerations $a^i_t$ are determined stochastically, as the external light stimuli are applied in a probabilistic manner. Fig. 2 shows a typical time evolution of the four-branch TOW1, where the initial
2.2.1. EGR2: extended ε-Greedy algorithm

Only by substituting Eqs. (3)–(5), with the following equations, EGR1 can be naturally extended to EGR2:

\[
\begin{align*}
 g^t_{i,j} &= R^t_{i,j} - U^t_{i,j}, \\
 U^t_{i,j} &= U^{t-1}_{i,j} + \frac{1}{t} \cdot q^t_{i,j}, \\
 R^t_{i,j} &= R^{t-1}_{i,j} + \frac{1}{t} \cdot r^t_{i,j}.
\end{align*}
\]

That is, if \(\{i, j\} = \arg\max_{a,b,\{a,b\} \in I} \{g^t_{i,j}\}\), EGR2 takes a greedy action to play machines \(i\) and \(j\) simultaneously (i.e., \(u^t_i = u^t_j = 1\) and \(u^t_k = 0\) for all \(k \notin I\setminus\{i,j\}\)) with the probability \(1 - \epsilon\) or explores a randomly chosen pair with the probability \(\epsilon\) given by Eq. (6).

2.2.2. SMX2: extended SoftMax algorithm

SMX1 can be developed to SMX2 by replacing Eq. (7) with the following equation:

\[
\epsilon^t_{i,j} = \frac{\exp(\beta \cdot g^t_{i,j})}{\sum_{k,l \in I} \exp(\beta \cdot g^t_{k,l})},
\]

where \(\beta = \beta \cdot t\) is the time-dependent decay rate and \(g^t_{i,j}\) is calculated by Eqs. (16)–(18). At each time \(t\), SMX2 plays machine \(i\) and \(j\) simultaneously with the probability \(\epsilon^t_{i,j}\).

2.2.3. TOW2: extended Tug-of-War model

We revised three points for extending TOW1 to TOW2. First, we set an upper limit on the absolute value of the velocity \(v^t_i\) in Eq. (12) as follows:

\[
v^t_i = L(u^{t-1}_i + a^t_i),
\]

where \(L(v) = y \cdot |v\) when \(v < y\); \(-y\) when \(v \leq -y\); \(v\) (otherwise) and \(y \geq 0\) is the limit value. In this study, \(y\) is fixed at \(y = 10\).

Second, we updated the acceleration \(a^t_i\) determined discretely by Eq. (13) to be modulated continuously depending on the volume displacements of the hub \(x^0_i\) and branch \(x^0_i\):

\[
a^t_i = -L_i \cdot (1 - \theta [L_i \cdot x^0_i]) + \eta \cdot (x^0_i - x^0_i),
\]

where \(\eta\) is a parameter for adjusting the restoring effect which narrows the gap between \(x^0_i\) and \(x^0_i\). We also fix this parameter as \(\eta = 0.002\).

Third, we modified Eq. (15) so that TOW2 can accumulate the information on coinciding rewards and punishments:

\[
q^t_i = q^{t-1}_i + \sum_{j=1}^{M} \mu \cdot (\rho^t_i \cdot \rho^t_j + \omega \cdot \sum_{k=1}^K \pi^t_k \cdot \pi^t_k).
\]

A typical time evolution of the four-branch TOW2 is shown in Fig. 3. TOW2 finally stabilized to play only two machines 3 and 4 while the two volume displacements oscillate in performing antiphasic synchronization. Comparing Figs. 2 and 3, we can confirm that there is a sharp contrast between the movements of TOW1 and TOW2. That is, the volume displacements of the latter are confined in a bounded state space owing to the velocity limit and the restoring effect, whereas that of the former appears to diverge.

Although in this study we consider a case where \(M=2\), TOW2 can be extended for an arbitrary \(M\) using the following generalized form of Eq. (22):

\[
q^t_i = q^{t-1}_i + \sum_{j=1}^{M} \mu \cdot (\prod_{j \neq i} \rho^t_j + \omega \cdot \prod_{k=1}^K \pi^t_k),
\]

conditions \((x^0_i, v^0_i, q^0_i)\) are always set to zero for all \(i\) in this study. TOW1 comes to play only machine 4, which has the highest reward probability, although initially all machines were played.

2.2. Extended Bandit Problem

We define the Extended Bandit Problem (EBP) as an extended notion of the multi-armed Bandit Problem (BP). The only difference between EBP and BP is that a player of the former should pick the optimal M-tuple out of the N slot machines, whereas a player of the latter is solely required to select the best one.

In this section, we deal with the case where \(M=2\) for simplicity in describing the definitions of EGR2, SMX2, and TOW2 that are extended versions of the previously introduced algorithms. The optimal 2-tuple \((i^\star, j^\star)\) is a pair of machines such that the sum of their reward probabilities is maximum, i.e., \((i^\star, j^\star) = \arg\max_{i,j,\{i,j\} \in I} \{i = l \in I, l < j\}\) is a set of all pairs. The number of all possible pairs is \(N \cdot (N-1)/2\).
where \( I_M \) is a set of all \( M \)-tuples, and \( I_{M-1} \) is a set of all \( M - 1 \)-tuples that do not contain \( i \). Using Eq. (23), we confirmed that the model elongates \( M \) branches exclusively in principle.

2.3. Average accuracy rate

As a measure for evaluating the performances of the above introduced algorithms, in this study we use the “average accuracy rate” instead of “regret” which is commonly used for analyzing logarithmic asymptotic behavior, mentioned in Section 1. The logarithmic regret behavior generally belongs to a long-term behavior. However, the long-term behavior can continue in constant environments that are rare in natural worlds. We are more interested in a short-term decision capability required for surviving in the dynamic environments that commonly occur. The average accuracy rate that we define in this section allows us to focus on the early-stage performances of the algorithms.

First we define the measure AverageAccuracyRate for evaluating EGR1, SMX1, and TOW1. Let \( \mathcal{I} = \{ \mathcal{I}^* \mid \mathcal{I}^* = \arg\max_{i \in I} \left( \hat{a}_i \right) \} \) be a set of all correct machines and \( \mathcal{I}^t = \{ i \in I \mid u_i^t = 1 \text{ or } \theta(x_i^t) = 1 \} \) be a set of all played machines at time \( t \). The AccuracyRate\(^t\) at time \( t \) is calculated as follows:

\[
\text{Accuracy Rate}^t = \frac{\sum_{s=1}^{t} \text{Correct}^s}{\sum_{s=1}^{t} \text{Play}^s},
\]

(24)

\[
\text{Correct}^t = \#(\mathcal{I}^t \cap \mathcal{I}^*),
\]

(25)

\[
\text{Play}^t = \#(\mathcal{I}^t),
\]

(26)

where \( \#(\mathcal{I}) \) counts the number of elements in the set \( \mathcal{I} \).

Recall that we should compare fairly the sequential and concurrent algorithms. For that purpose, if the concurrent algorithm played \( Y \) \( (\leq N) \) machines at time \( t \), we consider that AccuracyRate\(^t\) was subsequently continued over a \( Y \)-play span. More formally, AccuracyRate\(^t\) in concurrent time is mapped to sequential time as follows:

\[
\text{Accuracy Rate Series} = \text{Append}^t_{t=1} (\text{Copy} (\text{Accuracy Rate}^t, \text{Play}^t)),
\]

(27)

where \( \text{Copy}(x, m) \) gives \( m \) copies of \( x \). \( \text{Append}^t_{t=1} (X^t) \) generates the conjunction of the series \( \{X^1, X^2, \ldots, X^t\} \), and \( T = 1000 \) is the maximum observation time.

After Monte Carlo simulations for each algorithm, we obtain a collection of AccuracyRateSeries. Finally, the measure AverageAccuracyRate is calculated as a series averaged over 1000 samples of AccuracyRateSeries.

To evaluate EGR2, SMX2, and TOW2, AverageAccuracyRate can be extended by replacing the set of all correct machines \( \mathcal{I}^* \) and the set of all played machines \( \mathcal{I}^t \) in Eqs. (25) and (26) with \( \mathcal{I}^t = \{ (i^*, j^*) \mid \langle i^*, j^* \rangle = \arg\max_{i,j \in \mathcal{I}} \left( \hat{a}_i + \hat{p}_j \right) \} \) and \( \mathcal{I}^t = \{ (i,j) \mid i, j \in I^t, \ i < j \} \), respectively.

3. Results

We were interested in the early-stage performances of the algorithms because many real-world situations in general do not allow the algorithms to collect information for long periods of time. Therefore, we focused on the initial rise of AverageAccuracyRate. For each algorithm, we optimized the performance by changing a single parameter, i.e., \( \tau \) for EGRs, \( \beta \) for SMXs, and \( \mu \) for TOWs, so that the maximal AverageAccuracyRate can be achieved at 100 and 200 Plays.

In this study, we demonstrate the results for four illustrative sets of reward probabilities: \( P3E = \{ p_1, p_2, p_3 \} = \{ 0.2, 0.5, 0.8 \} \), \( P3H = \{ 0.4, 0.5, 0.6 \} \), \( P4E = \{ p_1, p_2, p_3, p_4 \} = \{ 0.2, 0.4, 0.6, 0.8 \} \), and \( P4H = \{ 0.35, 0.45, 0.55, 0.65 \} \). We chose these probability sets because of the following two reasons. First, they are “symmetric” in a sense that the differences of all values from the average value 0.5 are symmetrically distributed. In our previous study, we confirmed that TOW1 exhibited the best performances for symmetric probability sets of 3-machine instances by setting the parameter \( \omega = 1 \). The symmetric probability sets, therefore, allow us to skip the parameter optimization for \( \omega \). Second, these sets enable to evaluate the dependence of the performances on the difficulty of the problem instances. Because BP becomes difficult when the differences among the probabilities are small, P3E is “easier” than P3H, and P4H is “harder” than P4E.

3.1. Multi-armed Bandit Problem

Fig. 4 shows the performances of TOW1, EGR1, and SMX1 at 200 Plays for the four instances of BP. For each instance, we determined
the optimal parameter $\mu$ of TOW1 by comparing the maximal AverageAccuracyRates with the changes $\Delta \mu = 1$, where $\omega = 1$. The optimal parameters for EGR1 and SMX1 were selected with the changes $\Delta \tau = \Delta \beta = 0.05$.

For all the instances, AverageAccuracyRates of the optimized TOW1 at 100 and 200 Plays were higher than that of the optimized SMX1 and EGR1. That is, a player of BP using TOW1 can obtain a larger total reward sum after 100 and 200 trials, compared with SMX1 and EGR1 users. Fig. 4 also shows that AverageAccuracyRates of all the algorithms degrade as the problems become harder.

The initial rise of AverageAccuracyRates of SMX1 was higher than that of EGR1 in most cases.  

As well as the previous cases, AverageAccuracyRates of the optimized TOW2 at 100 and 200 Plays were larger than that of the optimized SMX2 and EGR2. We could also confirm that the difference in the performance between TOW2 and other algorithms becomes greater as the problem becomes difficult.

3.3. Summary

In summary, TOWs were stronger in their early-stage performances compared with EGRs and SMXs for the symmetric probability sets. However, we have to report not only strong points but also weak points of TOWs. TOWs were sometimes overtaken by EGRs and SMXs after long periods of observation time.  

In addition, TOW2 tends to be good at solving harder problem instances but weak for easier ones. We confirmed that sometimes SMX2 defeated TOW2 by narrow margins for very easy problems, for example, (0.05, 0.35, 0.65, 0.95).

Fig. 4. Average accuracy rate of the original Tug-of-War model (TOW1: solid red line), modified SoftMax algorithm (SMX1: blue dotted line), and modified $\epsilon$-Greedy algorithm (EGR1: green broken line) for the multi-armed Bandit Problem (BP). The horizontal and vertical axes denote the number of Plays and AverageAccuracyRate of 1000 samples, respectively. For each algorithm, a single parameter was optimized, i.e., $\mu$ for TOW1, $\beta$ for SMX1, and $\tau$ for EGR1. The parameter $\omega$ of TOW1 was fixed at $\omega = 1$, (a) A relatively easier 3-machine BP instance $\mathcal{P}E = (0.2, 0.5, 0.8)$, where the optimized parameters were $(\mu, \beta, \tau) = (3.0, 0.2, 0.25)$; (b) A relatively harder 3-machine instance $\mathcal{P}H = (0.4, 0.5, 0.6)$, where $(\mu, \beta, \tau) = (3.0, 0.25, 0.05)$; (c) a relatively easier 4-machine instance $\mathcal{P}E = (0.2, 0.4, 0.6, 0.8)$, where $(\mu, \beta, \tau) = (1.0, 0.20, 0.15)$; and (d) a relatively harder 4-machine instance $\mathcal{P}H = (0.35, 0.45, 0.55, 0.65)$, where $(\mu, \beta, \tau) = (3.0, 0.20, 0.05)$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

4 The performances of EGR1 and EGR2 are likely to be improved when we change the $\epsilon'$ into $\epsilon' = \min\{1, 1/(\tau - \tau')\}$.  

5 However, EGRs and SMXs require significant changes in their optimal parameters to achieve their best performances when the observation time gets longer. In contrast, TOWs do not need large parameter changes even if the observation time were extended.
4. Discussion

There is a study that connects an efficient algorithm for BP with human decision-making capability (Shinohara et al., 2007; Takahashi et al., 2010). The TOW models will be useful for studying the origin of the efficient resource-allocating capability of the amoeboid organism, because it was formulated on the basis of mechanics to be grounded in physical laws. The learning term of TOW (Eq. 22), which enables the "exploitation," represents a physically plausible process of accumulating information on coinciding events between two branches of the organism. Indeed, this learning process, which is realized in a manner similar to Hebbian reinforcement learning, has been experimentally confirmed, as the organism was found to be capable of strengthening the connectivity between the two parts that are in contact with food sources (Nakagaki et al., 2000; Tero et al., 2006, 2010).

Unlike many nature-inspired metaheuristics, the TOW model is able to "explore" without needing a random number generator that leaves the question of the origin of the decision to the external environment. The intrinsic dynamics of the TOW model are capable of spontaneously switching between the "exploitation" mode and "exploration" mode. This spontaneous mode-switching behavior was observed experimentally (Takamatsu, 2006) and was reproduced by the authors’ ordinary differential equation model (Aono et al., 2009b, 2011; Hirata et al., 2010).

The TOW model maintains a constant volume while collecting environmental information by concurrently growing and withdrawing its branches. We are interested in the effect of this conservation law on the computational capabilities of the organism because it yields a nonlocal correlation among the oscillating branches in terms of their spatiotemporal dynamics. In our previous study, we showed that the conservation law enhances efficiency and adaptability in solution-searching, as the resource increment information in a branch is instantaneously transmitted from one branch to the other so that they can immediately decrease their resources to compensate for the increment (Kim et al., 2010b). It is an interesting subject to verify the problem solving skill of the organism, by investigating the correlation between branches and the leaning processes assumed in the TOW algorithms.

5. Conclusion

In this study, we proposed two concurrent search algorithms that extract the physical nature of the efficient resource-allocating process of an amoeboid organism, the true slime mold *P. polycephalum*. The Tug-of-War algorithm (TOW1) and its extended version (TOW2) were applied to solving the multi-armed Bandit Problem (BP) and Extended Bandit Problem (EBP), respectively. Two well-known algorithms for the BP, the modified $\epsilon$-Greedy algorithm (EGR1) and modified SoftMax algorithm (SMX1), were
also extended to EGR2 and SMX2 respectively, so that they can be applied to EBP.

Optimizing a single parameter for each algorithm, we compared the performances of TOWs, EGRs, and SMXs in terms of their short-term decision-making capabilities represented by average accuracy rates. Although TOWs have more than one parameter, they exhibited better performances than the optimized EGRs and SMXs by adjusting solely a parameter $\mu$. Moreover, it was noteworthy that, even when the parameter $\mu$ was fixed, TOWs did not degrade significantly their early-stage performances. Indeed, TOW1 with $\mu = 3$ and TOW2 with $\mu = 7$ outperformed other algorithms for almost all the problem instances examined in this study. We will report the parameter robustness of TOWs elsewhere.

The proposed algorithms for BP and EBP are good at managing the exploration–exploitation dilemma, which is a trade-off between the speed and accuracy of the decision making that are vital but incompatible objectives for achieving successful business and quick adaptation in unpredictable worlds. Thus, we believe that our TOW models will be exploited for a broad range of real-world applications (Kim et al., 2013) and will be useful for exploring the physical nature of biological information processing.

References


