Partial-update exponential chaotic tabu search for a large-scale analog/digital hybrid electronic system

Yoshihiko Horio\textsuperscript{1a)}, Masato Ozawa\textsuperscript{1}, and Takeshi Miura\textsuperscript{1}

\textsuperscript{1} Graduate School of Engineering, Tokyo Denki University
5 Senju-Asahi-cho, Adachi-ku, Tokyo 120-8551, Japan
\textsuperscript{a)} horio@eee.dendai.ac.jp

Received May 11, 2014; Revised July 30, 2014; Published January 1, 2015

Abstract: A salient feature of complex systems is their inherent robustness against poor and fluctuating characteristics of constituent elements, systematic offsets in parameter values, environmental changes, noise, and other such fluctuations. In spite of such unreliable, poor-quality, and highly variable elements, complex systems generally exhibit high-quality behavior as a whole. In this paper, we exploit this inherent robustness of complex systems to create a compact and efficient realization of a hardware system for exponential chaotic tabu search. We start from a synchronous exponential chaotic tabu search algorithm and develop a novel partial-update exponential chaotic tabu search algorithm. The proposed algorithm is suitable for a large-scale analog/digital hybrid hardware implementation. In addition, a hardware system architecture suitable for the partial-update exponential chaotic tabu search is proposed, and a switched-current chaotic neuron integrated circuit dedicated to the proposed system architecture is also designed. We investigate the feasibility of implementing the digital part of the system with a field-programmable gate array.

Key Words: complex systems, analog/digital hybrid hardware, quadratic assignment problems, chaotic tabu search, chaotic neuron, nonlinear analog integrated circuits

1. Introduction

A salient characteristic of complex systems, particularly complex networked systems, is that they are robust against variations and even malfunctions of constituent elements, environmental changes, noise, and other non-ideal fluctuations. In addition, even if the system consists of unreliable, poor-quality, and highly variable elements, it exhibits reliable, high-quality, and stable behavior as a whole. A good example is the human brain. The brain consists of a vast number of neurons, whose individual performance is not very good and differs a lot among neurons. Nevertheless, the brain realizes a fast, reliable, high-quality, and high-precision computational information processing that is efficient, stable, and tolerant of environmental changes, such as body conditions and temperature changes.

If we exploit the robustness that is inherent in complex systems, it is possible to construct hardware that is simple, high-performance, and robust because noise, large offsets, and mismatches among
devices are acceptable in such a system. For example, the $\beta$-encoder, which is highly robust against parameter changes, offsets, low-gain of amplifiers, and temperature changes, has been implemented as an integrated circuit that uses nano-scale semi-conductor devices with large characteristic variations and poor performance as analog circuit elements [1–3]. This novel analog-to-digital (A/D) converter exploits the properties of the chaotic $\beta$-map attractor, which is always confined to an invariant subspace, and the rich redundancy in conversion codes for the same input that results from a real-number (non-integer) conversion base $\beta$ [4, 5]. As a result, superb $\beta$-A/D converter performance, which ordinary binary A/D converters cannot achieve, has been obtained.

We have also conducted a course of research on hybrid computational systems, which fuse highly parallel analog real-number computations based on high-dimensional physical chaotic dynamics with serial digital computation based on algorithms; this was inspired by the computational style of the brain, which uses mutual interactions between conscious and sub-conscious processes [6–9]. Examples of our hardware systems are the chaotic tabu search system for quadratic assignment problems (QAPs), and the general-purpose chaotic neuro-computer system [6–10]. These hybrid computational hardware systems share the properties of large-scale complex systems mentioned above. That is, these systems robustly realize high-performance computational functions that are tolerant of noise, environmental change, low accuracy, variations and mismatches of electronic circuit devices.

In this paper, we propose and discuss several modifications to the previously constructed asynchronous exponential chaotic tabu hardware system with the aim of making an efficient large-scale system that is suitable for solving large-scale practical problems. These modifications leverage the robustness inherent in complex systems to use simple circuits, algorithms, and architecture. However, it is important to preserve the following core dynamics for an efficient chaotic tabu search through real-number processing. These core dynamics include 1) chaotic itinerancy dynamics with high-dimensional coupled chaotic elements [11], 2) “weak” chaotic states near periodic windows, or the “edge of chaos” [12], 3) exponentially decaying memory effects naturally realized by refractoriness in a chaotic neuron model, and 4) competition and cooperation between convergent dynamics toward stable (or low-dimensional attractor) states and divergent dynamics toward high-dimensional chaotic states [13]. Therefore, in the following sections, we exploit the robustness of the exponential chaotic tabu search as a networked complex system to develop an efficient hardware system while preserving the above-mentioned core dynamics.

In Section 2, we briefly review quadratic assignment problems (QAPs) [14] and the asynchronous exponential chaotic tabu search hardware system for QAPs [9, 10]. In Section 3, we introduce the synchronous exponential chaotic tabu search algorithm [17], which is the starting point for our discussion. We then propose two modifications to the synchronous exponential chaotic tabu search in Sections 4 and 5: the first modification omits mutual connections among all neurons; the second simplifies the tabu effects. In Section 6, we propose partial updating, which is a novel updating scheme for neuronal states, and several additional modifications to the search algorithm. Finally, in Section 7, we propose a system architecture for the partial-update exponential chaotic tabu search. We also briefly introduce a dedicated switched-current (SI) chaotic neuron chip, and a field-programmable gate array (FPGA) implementation of the cost calculations.

2. Review of the asynchronous exponential chaotic tabu search hardware system for QAPs

In this section, we first give a short description of QAPs [14]. We then review the previously constructed asynchronous chaotic tabu search hardware system for QAPs [9, 10].

The size-$N$ QAP is given by two $N \times N$ matrices, $A$ and $B$, which are called the distance matrix and the flow matrix, respectively. The matrix $A$ gives mutual distances between $N$ elements (say, cities), while $B$ gives mutual relations between $N$ elements (e.g., factories) that are independent of those in $A$. The constraint of the QAP is that all factories should be assigned to distinct cities. A feasible solution of the QAP can be expressed by a permutation $p$ as
where the index of \( p \) corresponds to a city, and the element of \( p \) is the factory that is assigned to the city indicated by the corresponding index. The cost function \( F(p) \) of the QAP is given by

\[
F(p) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} b_{p(i)p(j)},
\]

where \( a_{ij} \) is the \((i,j)\)th element of \( A \), and \( b_{p(i)p(j)} \) is the \((p(i), p(j))\)th element of \( B \). The goal of the QAP is to obtain the assignment of factories to cities, \( p \), that gives the smallest value of \( F(p) \).

One heuristic method to solve the QAP is a tabu search with 2-opt exchange algorithm. In this method, two elements of \( p \) are swapped at each iteration, as shown in Fig. 1, if the resulting new assignment is a better solution. The assignment of element \( i \) to index \( j \) is called an \((i,j)\)-assignment, and the accompanying assignment, where the element \( p(j) \) is assigned to the index \( q(i) \), is called a \((p(j), q(i))\)-assignment. In tabu search, the assignments that were used during the previous \( s \) iterations (for some integer \( s \)) are not available for new assignments so as to mitigate the local minima problem.

The asynchronous exponential chaotic tabu search implements a tabu search on a chaotic neural network with \( N \times N \) chaotic neurons [15,16]. The dynamics of the \((i,j)\)th chaotic neuron in the network, where \( i \) is the row number and \( j \) is the column number, is given as

\[
\xi_{ij}(t+1) = \beta \{ F_1(t) - F_{ij}(t) \} = \beta \Delta F^p_{ij}(t),
\]

\[
\eta_{ij}(t+1) = -W \sum_{k=1}^{N} \sum_{l=1}^{N} y_{kl}(t) + k_f \eta_{ij}(t) - \alpha_\eta \{ y_{p(j)q(i)}(t) + z_{p(j)q(i)}(t) \} + (1 - k_f) \theta_\eta,
\]

\[
\zeta_{ij}(t+1) = k_r \zeta_{ij}(t) - \alpha_\zeta \{ y_{ij}(t) + z_{ij}(t) \} + (1 - k_r) \theta_\zeta,
\]

\[
x_{ij}(t+1) = \xi_{ij}(t+1) + \eta_{ij}(t+1) + \zeta_{ij}(t+1),
\]

\[
y_{ij}(t+1) = f(x_{ij}(t+1)),
\]

where \( F_1(t) \) is the current value of the objective function at time index \( t \), \( F_{ij}(t) \) is the value of the objective function after the \((i,j)\)-assignment, \( \Delta F^p_{ij}(t) \equiv F_1(t) - F_{ij}(t) \) is the gain in the cost function resulting from the \((i,j)\)-assignment, and \( \beta \) is the scaling parameter of the gain. The internal state \( \eta_{ij}(t) \) is a sum of the weighted feedback from other neurons with a coupling coefficient of \( W \), which regulates the firing rate of the network, and the tabu effect for the \((p(j), q(i))\)-assignment. In particular, \( z_{p(j)q(i)}(t) \) memorizes the \((p(j), q(i))\)-assignment that does not correspond to the index of a firing neuron\(^1\). The internal state \( \zeta_{ij}(t) \) is the tabu effect for the \((i,j)\)-assignment. In addition, \( \alpha_\eta \) and \( \alpha_\zeta \) are the scaling parameters for the tabu effect, and \( k_f \) and \( k_r \) are decay parameters for the tabu effect. Finally, \( x_{ij}(t) \) is the total internal state, \( y_{ij}(t) \) is the output of the neuron, \( f(\cdot) \) is a monotonically increasing continuous nonlinear output function, and \( \theta_\eta \) and \( \theta_\zeta \) are external biases.

\(^1\)Equation (4) is a combination of \( \eta_{ij}(t+1) \) and \( \gamma_{ij}(t+1) \) originally defined in [15,16]. See these references for their roles in more detail.
Note that $z_{ij}(t)$ and $z_{p(j)q(i)}(t)$ are additional variables used for asynchronous (sequential) updates of the neuronal states [15, 16].

Neuronal states are updated one by one (asynchronous update). When the $(i, j)$th neuron in the network fires, then the $(i, j)$ and $(p(i), q(j))$-assignments are executed. The tabu effect is naturally included through the exponentially decaying refractoriness of the chaotic neuron, so that the tabu effect also exponentially decays. Furthermore, the chaotic search with chaotic itinerancy realizes an efficient search in a high-dimensional solution space. Therefore, asynchronous exponential chaotic tabu search provides a powerful method to solve QAPs.

We faithfully implemented the asynchronous exponential chaotic tabu search algorithm as an analog/digital hybrid electronic system with an SI chaotic neuron integrated circuit [9, 10]. We then illustrated superb ability of the hardware system to solve QAPs. However, the size of QAPs that the hardware system can handle is limited to 10 because of hardware constraints. Therefore, we need a larger-scale hardware system to cope with practical problems. Unfortunately, for large-scale systems, we cannot directly employ the same system architecture used in the previous hardware system because a simple extension will result in a huge and impractical implementation. Moreover, the computational time for such a simply extended system will grow according to the square of the problem size.

3. From asynchronous dynamics to synchronous dynamics

To solve the problem mentioned above, we adopted a synchronous update method that simultaneously updates all neuronal states in the network in one iteration [17, 18]. In addition, $z_{ij}(t)$ and $z_{p(j)q(i)}(t)$ are omitted from Eqs. (5) and (4), respectively, because these terms were originally added to support the asynchronous updating of neuronal states. For this method, the processing time is ideally constant, regardless of the problem size.

However, the following difficulty arises from using a synchronous update. In the original asynchronous system, the 2-opt exchange is evaluated at a particular iteration if the currently updated neuron, which is only one neuron, fires. However, for synchronous updating, more than one neuron may fire simultaneously, so that we have to decide which one of the firing neurons we use for the 2-opt exchange. To solve this problem, we have proposed several methods to select one particular neuron for the 2-opt exchange from among all firing neurons [17–19]. One of the neuron selection methods is “Method-A,” detailed in [19], which can be summarized as follows:

Method-A:

MA-1 $\Delta F_{p}^{ij}(t)$ is calculated for all neurons. Then, all neurons are updated simultaneously according to $\Delta F_{p}^{ij}(t)$.

MA-2 We sort fired neurons according to the internal state values in descending order.

MA-3 We select the first $U$ neurons from the sorted list, where $U$ is a parameter.

MA-4 The current permutation $p_{\text{current}}$ is tentatively updated separately with each of the selected $U$ neurons, resulting in $U$ different permutations $p_k$ ($1 \leq k \leq U$).

MA-5 We select the permutation $p_M$ from $p_k$ that gives the largest value of $\Delta F_{M}^{ij}(t)$ (defined as $\Delta F_{M}^{ij}(t)$). If $\Delta F_{M}^{ij}(t) > 0$, then $p_{\text{current}}$ is replaced with $p_M$. Otherwise, we keep $p_{\text{current}}$. Then, we go back to MA-1.

However, the above modifications will result in synchronous dynamics different from the original asynchronous dynamics [20–23]. Therefore, we confirmed through numerical simulations that the proposed synchronously updated chaotic neuro-dynamics is also effective for exponential chaotic tabu search [17–19].

The above synchronous exponential chaotic tabu search with Method-A (we will now refer to this algorithm as SECT-MA) still has several drawbacks for hardware implementation. Therefore, in the following sections, we will further modify and improve the SECT-MA algorithm by exploiting the
robustness of a networked complex system as much as possible while preserving the core dynamics that we mentioned in Section 1.

4. Omitting direct mutual couplings among neurons

The all-to-all direct connections of analog neuronal outputs via the synaptic weight $W$, as in Eq. (4), will impose a heavy burden on an electronic hardware system. Therefore, our first modification is to omit these direct synaptic couplings among all neurons. That is, Eq. (4) is modified to

$$\eta_{ij}(t + 1) = k_f \eta_{ij}(t) - \alpha_\eta y_{p(j)}(t)\eta_{ij}(t) + (1 - k_f)\theta_\eta. \quad (8)$$

As a consequence, the mutual interactions between the chaotic neuro-dynamics and the 2-opt exchange algorithm through $\xi_{ij}(t + 1)$ in Eq. (3) dominate in the chaotic search dynamics. Note, however, that indirect mutual interactions among neurons still exist via $\xi_{ij}(t + 1)$.

To confirm the robustness of the SECT-MA against the lack of mutual connections among neurons, we examined the performance of SECT-MA systems with different strengths of mutual synaptic connections $W$. The results are shown in Figs. 2 and 3 for the benchmark problems Had20, Lipa20a, Lipa20b, Tai20a, and Tai20b, taken from the QAP Library [14]. In the figures, the average gap from the optimal solution $GAP$ and the average rate of success in finding the optimal solution $AR$ are respectively defined as

$$GAP = \frac{\sum_{n=1}^{TR} (F_{\text{best}}(n) - F_{\text{opt}})}{TR} \times 100\%, \quad \text{and}$$
$$AR = \frac{\text{A Number of Success Trials}}{TR} \times 100\%, \quad (10)$$

where $TR$ is the total number of trials, $F_{\text{best}}(n)$ is the smallest value of the cost function in Eq. (2) obtained in the $n$th trial, and $F_{\text{opt}}$ is the value of the cost function for the optimal solution. For the results in Figs. 2 and 3, we used $TR = 100$ with different initial conditions, and each trial consisted of 10,000 iterations. The parameter values used in the simulations were determined through the following procedure. We first preliminarily picked up some good candidates for the parameter sets through coarse simulations with 5 trials for each parameter set with 2,000 iterations per trial. In these coarse parameter searches, we used 2 values for $W$; $W = 0$ and $W = 0.2$. In addition, we swept the values of $\alpha_\eta$, $\alpha_\zeta$, $\beta$, $k_f$, $k_r$, $R_\eta \equiv (1 - k_f)\theta_\eta$, and $R_\zeta \equiv (1 - k_r)\theta_\zeta$ from 0.1 to 1 in step of 0.1 considering the precision of the analog hardware implementation of these parameters. According to the results of the coarse simulations, we selected the preliminary parameter sets that give the average

![Fig. 2. Average gaps from the optimal solution $GAP$ for different values of $W$ in SECT-MA.](image-url)
Fig. 3. Average rates of success in finding the optimal solution AR for different values of W in SECT-MA.

As shown in these figures, SECT-MA is tolerant of changes in the value of W. In other words, the mutual synaptic connections contribute only weakly to efficient chaotic search, so we can omit W from Eq. (4). We refer to the SECT-MA algorithm without synaptic connections among neurons (i.e., W = 0) as SECT-MA-woW in the following.

5. Simplification of tabu effects

The next obstacle for a large-scale hardware system is the input from the \((p(j), q(i))\)th neuron to the \((i, j)\)th neuron, \(y_{p(j)q(i)}(t)\) in Eq. (8). Because \(p(j)\) and \(q(i)\) change iteration by iteration, we need to provide an all-to-all connection capability via a switch matrix circuit to accommodate the different one-to-one connection paths selected in each iteration.

The simulation results in Fig. 4 compare the performance, measured by GAP and AR, with and without the \(\eta_{ij}\) term of Eq. (8) for Had20, Lipa20a, and Lipa20b problems from the QAP Library. The parameters used in the simulations for SECT-MA-woW are the same as in Table I, while those for the simulations without \(\eta\) term were determined through the similar procedure described in Section 4, and are listed in Table II. In this parameter search, however, we set \(W = 0\), and we varied the value of \(R_\zeta\) with a step of 0.002 since we set the value of \(R_\zeta\) through the external voltage reference in a hardware system so that we can control it precisely.

These results show that the tabu effect for \((i, j)\)-assignment is a key factor in the exponential tabu search, while the searching dynamics is tolerant to changes of \((p(j), q(i))\)-assignment. Therefore, as
the second modification, we completely omit Eq. (8) from the chaotic neuron model used in our hybrid hardware system. This leads the system to rely only on the tabu effect for the \((i, j)\)-assignment.

The resulting chaotic neuron model for the hybrid exponential chaotic tabu search hardware system can be summarized as

\[
\xi_{ij}(t+1) = \beta \Delta F_p^{ij}(t),
\]

\[
\zeta_{ij}(t+1) = k_r \zeta_{ij}(t) - \alpha \zeta y_{ij}(t) + (1 - k_r) \theta \zeta,
\]

\[
x_{ij}(t+1) = \xi_{ij}(t+1) + \zeta_{ij}(t+1),
\]

\[
y_{ij}(t+1) = f(x_{ij}(t+1)).
\]

In the following, we shall refer to the SECT-MA-woW with the above simplified tabu effect as SECT-MA-woW/Eta.

6. From synchronous update to partial update

Because real-number computation through physical high-dimensional chaotic dynamics is important, we use analog circuitry to implement Eqs. (11)–(14). However, the term \(\Delta F_p^{ij}(t)\) in Eq. (11) involves matrix manipulations, so it is a feasible option to implement this term with digital circuitry, such as an FPGA, as will be described in Section 7.3.

To calculate \(\Delta F_p^{ij}(t)\) in procedure MA-1 of Section 3, manipulations of large matrices \(A\) and \(B\) are necessary. Then, we need to deliver the value of \(\Delta F_p^{ij}(t)\) to all neurons in the network. This incurs a huge hardware overhead when the problem size \(N\) is large. For MA-2, on the other hand, we first measure the all analog internal state values. This requires high-precision measurements of these analog values, which needs huge and slow hardware. In addition, sorting these internal state values consumes a lot of time, if \(N\) is large. Therefore, the evaluation of \(\Delta F_p^{ij}(t)\) for the synchronous update in SECT-MA-woW/Eta will require a lot of hardware resources, including both analog circuits (A/D and digital-to-analog (D/A) converters, sense amplifiers, etc.) and digital circuits (memory, multipliers, comparators, etc.), and this will result in a very slow system when \(N\) is large.

To alleviate the above problems, we employ a “partial update” method that updates only a part of neuronal states simultaneously, unlike the synchronous update. To do so, we first determine the

---

**Fig. 4.** The GAP and AR with and without the \(\eta_{ij}\) term of Eq. (8).

**Table II.** Parameter sets used in the simulations for Fig. 4 without \(\eta \) term.

| Instance | \(\alpha\) | \(\beta\) | \(k_r\) | \(R_\zeta\) |
|----------|----------|----------|----------|----------|
| Had20    | 0.9      | 0.5      | 0.3      | 0.224    |
| Lipa20a  | 0.9      | 0.3      | 0.3      | 0.244    |
| Lipa20b  | 0.7      | 0.5      | 0.5      | 0.63     |
minimum number of neurons in the network that should be updated at once. The performance of the chaotic search has certain relations to the average firing rate (AFR) of the network as shown in Figs. 5 and 6. The parameter values for these figures are the same as listed in Table II. These figures illustrate that chaotic neuro-dynamics with an AFR of about 10%–15% is effective. This result may be universal for high-dimensional chaotic systems, and actually is consistent with our former results with different systems and settings [6–9]. The average firing rate can be easily controlled through the external threshold $\theta_\zeta$, which we can tune with simple analog circuitry. As a consequence, we always keep $AFR \approx 10$–15% by tuning $\theta_\zeta$ in the following.

![Graph showing GAP and AR for SECT-MA-woW/Eta for the Had20 problem from the QAP Library when we change the average firing rate AFR. AFR $\approx 10$–15% shows good performance.](image1)

![Graph showing GAP and AR for SECT-MA-woW/Eta for the Lipa20a problem from the QAP Library when we change the average firing rate AFR. AFR $\approx 10$–15% shows good performance.](image2)

6.1 The number of simultaneously updated neurons
We already know that $U = 5$ is a good choice from our previous research [17, 18]. This means that we want to have five firing neurons on average in each iteration. Given $U = 5$, and $AFR = 10\%$ for the worst case where the largest number of neurons are required for $AFR \approx 10$–15%, we can conclude that at most 50 neurons should be updated simultaneously in the partial update. In this case, we
calculate only 50 values of $\Delta F_{ij}^p(t)$ out of a possible $N^2$. As a result, a simple and fast system will be feasible with the partial update scheme.

6.2 Systematic selection of firing neurons
For simpler hardware realizations, we use the first 5 (as $U = 5$) firing neurons in ascending order of the neuron index to completely avoid the sorting of fired neurons according to their internal state values. As a result, only binary information (fire and non-fire) is required, which greatly reduces required hardware resources.

6.3 SLIDE parameter
The procedure for choosing 50 neurons from the network for the partial update, and the simple selection of the first 5 firing neurons by index number will greatly affect the chaotic search dynamics. Therefore, we introduce a “SLIDE” parameter, which indicates how many of these 50 neurons we should replace with new ones in the next iteration, where $1 \leq \text{SLIDE} \leq 50$. If SLIDE is small, local search might be realized, while if SLIDE is large, the search would be global. Therefore, we can properly control the searching dynamics through SLIDE.

6.4 Quasi-randomization of neuron indices
The avoidance of mismatches and noise is a common practice in ordinary system (or circuit) design, particularly for linear systems. In contrast, the collective dynamics and functions of complex systems have a tendency to be tolerant of and robust against mismatches and noise. Moreover, complex systems, e.g., animals and plants, may use these non-ideal conditions to obtain a rich diversity for evolving better performance. Thus, we positively exploit both the mismatches among characteristics of analog chaotic neuron circuits and noise to effectively diversify the chaotic search dynamics.

The hardware neuron circuits have inherent mismatches and noise, so we can diversify the dynamics by randomly reassigning neuron indices to neuron circuits in every trial. Hardware structure changes in each trial to accommodate these index number reassignments are impractical. Therefore, we use a fixed hardware structure instead, but realize the random index reassignments by randomizing the elements of $A$ and $B$ [24]. In this element randomizing process, however, we have to keep the original relationships between corresponding elements in $A$ and $B$. Therefore, we apply standard row and column substitution matrices several times to $A$ and $B$ for this purpose. We cannot, of course, obtain truly randomized matrices by applying these substitution matrices; nevertheless, we can exploit the mismatches and noise to diversify the search dynamics by the reassignment of the neuron indices. So, we call this scheme the “quasi-randomization” of neuron indices.

6.5 Partial-update exponential chaotic tabu search
The following summarizes the modified SECT-MA-woW/Eta discussed above, which we refer to as the “partial-update exponential chaotic tabu” (PUECT) search.

PUECT:

PU-1 We quasi-randomize matrices $A$ and $B$ by applying row and column substitution matrices to them several times.

PU-2 We use $U = 5$, and set $AFR \approx 10–15\%$ by tuning the value of $\theta_{\zeta}$.

PU-3 A group of 50 neurons is updated simultaneously.

PU-4 We select the first 5 fired neurons in ascending order of neuron index.

PU-5 The current permutation $p_{\text{current}}$ is updated with the selected 5 neurons resulting in 5 different permutations: $p_1, p_2, \ldots, p_5$.

PU-6 We select the permutation $p_M$ from the resulting 5 permutations that gives the largest value of $\Delta F_M^p(t)$. If $\Delta F_M^p(t) > 0$, then $p_{\text{current}}$ is replaced with $p_M$. Otherwise, we keep $p_{\text{current}}$.  

79
We offset the starting index of the group of 50 neurons by the value specified by $SLIDE$, and we go back to PU-3.

Figures 7 and 8 compare the values of $GAP$ and $AR$, respectively, for SECT-MA-woW/Eta and PUECT with different values of $SLIDE$. For SECT-MA-woW/Eta, we used the same parameter sets as shown in Table II. On the other hand, we used those listed in Table III for PUECT, which we determined through the similar procedure described in Sections 4 and 5. From these figures, PUECT shows better performance if we properly set the value of $SLIDE$. For the cases shown in Figs. 7 and 8, $SLIDE = 5$ or 10 is a good choice. In other words, Figs. 7 and 8 confirm that we can control the balance between the local and global search dynamics through $SLIDE$ as we expected (see Section 6.3).

![Fig. 7. GAP for the SECT-MA-woW/Eta and PUECT with different values of SLIDE. The Lipa20b problem from the QAP Library was used.](image1)

![Fig. 8. AR for SECT-MA-woW/Eta and PUECT with different values of SLIDE. The Lipa20b problem from the QAP Library was used.](image2)

**Table III.** Parameter sets used in the simulations for PUECT in Figs. 7 and 8.

| Instance | $\alpha_C$ | $\beta$ | $\kappa_r$ | $R_\zeta$ | $SLIDE$     |
|----------|------------|---------|------------|------------|-------------|
| Lipa20b  | 0.7        | 0.5     | 0.5        | 0.9        | 1, 5, 10, 25 |
7. Practical considerations for hardware implementation

In this section, we propose an analog/digital hybrid hardware system architecture for the PUECT algorithm. As was mentioned in Section 6, $\Delta F_{ij}^{P}(t)$ in Eq. (11) is implemented using digital circuitry, and the rest of the model equations are realized by an analog SI chaotic neuron integrated circuit. In the following, the SI chaotic neuron IC designed for the proposed system architecture is briefly introduced. In addition, an FPGA implementation scheme for the digital part is briefly described.

7.1 System architecture for PUECT

We employ a chip-parallel architecture to realize PUECT as an analog/digital hybrid electronic circuit system, as shown in Fig. 9. Because we update 50 neurons simultaneously, we use 50 SI chaotic neuron chips in parallel, each of which includes many chaotic neurons (we assume Q neurons in the following). The digital calculations for $\Delta F_{ij}^{P}(t)$ are accomplished by one FPGA.

In every iteration, one of the Q neurons in each SI chaotic neuron chip is selected, resulting in a total of 50 neurons selected from 50 chips. The FPGA calculates $\Delta F_{ij}^{P}(t)$ for those 50 selected neurons, and sends their values to the corresponding neuron through the latched D/A converter (DAC) circuit. Then, the FPGA updates the states of the 50 neurons simultaneously, by providing proper clock signals to the chaotic neuron ICs. After that, the firing states of the 50 currently selected neurons are checked by the FPGA in ascending order of chip number. From the first 5 fired neurons, the FPGA finds the largest value of $\Delta F_{ij}^{P}(t)$, denoted by $\Delta F_{M}^{P}(t)$. If $\Delta F_{M}^{P}(t) > 0$, then the FPGA replaces the current permutation $p_{current}$ by the permutation $p_{M}$ that gives $\Delta F_{M}^{P}(t)$. This concludes one iteration. Before the next iteration, the FPGA shifts the indices of the selected neurons in each SI chaotic neuron chip according to the value of the SLIDE parameter. After this, the same process will be repeated until the end of one trial. When the trial has finished, the main control quasi-randomizes the matrices $A$ and $B$ and stores new matrices in the FPGA. This starts the next trial.

7.2 SI chaotic neuron chip

We have designed and fabricated a dedicated SI chaotic neuron chip that includes 736 neurons, as shown in Fig. 10 [25]. With the SI chaotic neuron circuit, we can tune the value of the important
Fig. 10. The layout of the SI chaotic neuron chip designed for the PUECT system using TSMC 180 nm CMOS semiconductor process technology. The figure on the right enlarges one of the SI chaotic neurons. The nominal clock frequency is 100 MHz, and a total of 736 chaotic neurons are included in one chip.

parameter \( k_r \) in Eq. (12) from 0.2 to 0.8 in steps of 0.1. Because we have 50 neuron chips on the system board, as shown in Fig. 9, a system with this chip can handle QAPs of size up to \( N = 191^2 \).

Since the chip has only just been fabricated, the details of the chip and measurement results will be reported elsewhere.

7.3 Calculation of \( \Delta F_{ij}^P(t) \) using FPGA

We need a digital device that provides quick multiplication with fast memory access to calculate \( \Delta F_{ij}^P(t) \). We have considered the possibilities of using an FPGA or a DSP (digital signal processor) in the proposed chip-parallel architecture and we have concluded that using an FPGA with pipelined processing is most feasible [24]. We are currently making the FPGA circuits using Xilinx XC7K325T. Details will be reported elsewhere.

8. Conclusions

We have proposed a partial-update exponential chaotic tabu search algorithm for QAPs by modifying and improving the synchronous exponential chaotic tabu search algorithm. In the modification process, we positively exploited the inherent robustness of the original chaotic tabu search system, which arises because it is a complex networked system, to design compact, fast, and efficient hardware. The proposed algorithm is suitable for a large-scale analog/digital hybrid hardware implementation. In addition, a possible system architecture for the hardware system has been proposed. An SI chaotic neuron integrated circuit has also been designed for the proposed system architecture. Feasibility studies have confirmed that an FPGA with a pipelined processing should be the first candidate for a component to implement the digital calculation of \( \Delta F_{ij}^P(t) \).

We will, as a future problem, construct a prototype hardware system for partial-update exponential chaotic tabu search with the proposed system architecture using the SI chaotic neuron ICs and FPGAs.

Acknowledgments

This research was partially supported by the Aihara Innovative Mathematical Modelling Project, the Japan Society for the Promotion of Science (JSPS) through the “Funding Program for World-Leading

\(^2\)If we mount more than 50 chips on the board but still use 50 neurons for the partial update, larger problems can be solved by the same system although the system control will be complicated.
Innovative R&D on Science and Technology (FIRST Program)" initiated by the Council for Science and Technology Policy (CSTP).

References

[1] H. San, T. Kato, T. Maruyama, K. Aihara, and M. Hotta, “Non-binary pipeline analog-to-digital converter based on β-expansion,” IEICE Trans. on Fundamentals of Electron., Commun. & Computer Sciences, vol. E96-A, no. 2, pp. 415–421, 2013.
[2] R. Suzuki, T. Maruyama, H. San, K. Aihara, and M. Hotta, “Robust cyclic-ADC architecture based on β-expansion,” IEICE Trans. on Electronics, vol. E96-C, no. 4, pp. 553–559, 2013.
[3] R. Sugawara, R. Suzuki, H. San, K. Aihara, and M. Hotta, “A β-expansion based 10-bit CMOS cyclic ADC with radix-value self-correction technique,” in Proc. Int. Conf. on Analog VLSI Circuits, CD-ROM, October 2013.
[4] T. Kohda, Y. Horio, Y. Takahashi, and K. Aihara, “Beta encoders: Symbolic dynamics and electronic implementation,” Int. Journal of Bifurcation and Chaos, vol. 22, no. 9, DOI: 10.1142/S02181274123000315, 2012.
[5] T. Kohda, Y. Horio, and K. Aihara, “β-expansion attractors observed in A/D converters,” AIP Chaos, vol. 22, no. 4, DOI: 10.1063/1.4758813, 2012.
[6] Y. Horio and K. Aihara, “Chaotic neuro-computer,” in Chaos in Circuits and Systems, pp. 237–255, G. Chen and T. Ueta eds., World Scientific Publishers, June 2002.
[7] Y. Horio, O. Yamamoto, and K. Aihara, “Neuron-synapse IC chip-set for large-scale chaotic neural networks,” IEEE Trans. Neural Networks, vol. 14, no. 5, pp. 1393–1404, 2003.
[8] Y. Horio, T. Ikeguchi, and K. Aihara, “A mixed analog/digital chaotic neuro-computer system for quadratic assignment problems,” Neural Networks, vol. 18, no. 5-6, pp. 505–513, June/July, 2005.
[9] Y. Horio and K. Aihara, “Analog computation through high-dimensional physical chaotic neurodynamics,” Physica D, vol. 237, no. 9, pp. 1215–1225, 2008.
[10] S. Matsui, Y. Kobayashi, K. Watanabe, and Y. Horio, “Exponential chaotic tabu search hardware for quadratic assignment problems using switched-current chaotic neuron IC,” in Proc. Int. Joint Conf. Neural Networks, vol. 3, pp. 2221–2226, 2004.
[11] K. Kaneko and I. Tsuda, “Chaotic itinerancy,” AIP Chaos, vol. 13, no. 3, pp. 926–936, 2003.
[12] C.G. Langton, “Computation at the edge of chaos: Phase transitions and emergent computation,” Physica D, vol. 42, pp. 12–37, 1990.
[13] K. Aihara, “Chaos engineering and its application to parallel distributed processing with chaotic neural networks,” Proc. IEEE, vol. 90, no. 5, pp. 919–930, 2002.
[14] R.E. Burkard, S.E. Karisch, and F. Rendl, “QAPLIB - A quadratic assignment problem library,” J. of Global Optimization, vol. 10, pp. 391–403, 1997, http://www.opt.math.tu-graz.ac.at/qaplib/.
[15] M. Hasegawa, T. Ikeguchi, and K. Aihara, “Exponential and chaotic neuromodular tabu searches for quadratic assignment problems,” Control and Cybernetics, vol. 29, no. 3, pp. 773–788, 2000.
[16] M. Hasegawa, T. Ikeguchi, K. Aihara, and K. Itoh, “A novel chaotic search for quadratic assignment problems,” European J. of Operational Research, vol. 139, pp. 543–556, 2002.
[17] N. Yokota, Y. Horio, M. Hasegawa, and K. Aihara, “A synchronous exponential chaotic tabu search for quadratic assignment problems,” in Proc. Int. Workshop on Nonlinear Circuits and Signal Processing, pp. 212–215, 2008.
[18] N. Yokota, Y. Horio, K. Aihara, and M. Hasegawa, Tech. Rep. IEICE, vol. 107, no. 561, pp. 49–54, 2008 (in Japanese).
[19] T. Kawamura, Y. Horio, and M. Hasegawa, “Mutual information analysis of chaotic neuodynamics driven by neuron selection methods in synchronous exponential chaotic tabu search for quadratic assignment problems,” in Neural Information Processing –Theory and Algorithms-, K.W. Wong, B. Sumudo, U. Mendis, and A. Bouzerdoum eds., Lecture Notes in Computer Science, vol. LNCS 6433, part 1, Springer, pp. 49–57, 2010.
[20] M.A. Arbib ed., *The Handbook of Brain Theory and Neural Networks*, The MIT Press, 1995.
[21] J.A. Anderson and E. Rosenfeld eds., *Neurocomputing*, The MIT Press, 1988.
[22] Y. Horio, T. Taniguchi, and K. Aihara, “An asynchronous spiking chaotic neuron integrated circuit,” *Neurocomputing*, vol. 64, pp. 447–472, 2005.
[23] M.D. Shrimali, S. Sinha, and K. Aihara, “Asynchronous updating induces order in threshold coupled systems,” *Physical Review E*, vol. 76, 046212, 2007.
[24] T. Miura, M. Ozawa, T. Orima, and Y. Horio, “Digital circuit implementation of a cost calculation for partial-update exponential chaotic tabu search hardware systems,” *Tech. Rep. IEICE*, vol. 114, NLP2014-34, pp. 17–22, 2014 (in Japanese).
[25] S. Tanaka, A. Toyoda, Y. Horio, and K. Aihara, “A switched-current chaotic neuron IC prototype for synchronous exponential chaotic tabu search,” *Tech. Rep. IEICE*, vol. 113, no. 341, pp. 65–69, 2013.