Effect of feedback delays on solution quality in amoeba-inspired computing system that solves traveling salesman problem

Kenta Saito[1*] and Seiya Kasai[1,2]

[1]*Research Center for Integrated Quantum Electronics and Graduate School of Information Science and Technology, Hokkaido University, N13 W8, Sapporo 060-0813, Japan*
[2]*Center for Human Nature, Artificial Intelligence, and Neuroscience, Hokkaido University, N12 W7, Sapporo 060-0812, Japan*

E-mail: k-saito@rciqe.hokudai.ac.jp

We investigate how feedback delays affect the quality of solutions from an amoeba-inspired analog electronic computing system that solves the "traveling salesman problem." Delays in the feedback process induce the oscillation of state variables. With an appropriate delay length, the system converges to the stable state that corresponds to the solution after oscillation. We find that the solution quality is improved by increasing the delay length. Consequently, delays bring a trade-off between the solution search time and the solution quality. Delay scheduling can further improve solution quality.

Delays are unavoidable in a physical system, and they decelerate the operations of conventional electronic computers. However, they sometimes play an important role in a feedback system such as a harmonic oscillator[1,2], which is indispensable for many electronic systems. The delayed feedback mechanism is also ubiquitous in human biological systems such as the neural network[3–5], the auditory system[6,7], multistability in a neuron[8], production of white blood cells[9,10], etc. All of this suggests that delays in a bio-inspired electronic system can play a constructive or positive role in its function. As one of such electronic systems, we have recently developed an amoeba-inspired, all-analog electronic computing system to solve optimization problems called the "electronic amoeba"[11,12]. This system consists of an instance-mapping circuit (IMC) and an amoeba core that represents the spatiotemporal dynamics of an amoeboid organism (shown in **Fig. 1**). Each unit in the amoeba core corresponds to state variables. The IMC feeds back into the amoeba core in accordance with a specific rule (the "bounce-back rule") derived from the given constraints[11,12]. This analog electronic system has a configuration similar to a recurrent neural network. It operates in an asynchronous way, with the state variables sometimes oscillating due to a delay in the feedback process, causing the system to destabilize. Therefore, we investigate the delay-induced instability in an amoeba-inspired analog electronic computing system that solves the traveling salesman problem (TSP) and its effect on solution quality. When the system destabilizes, the amoeba core employs trial and error using the oscillations, which provides an opportunity to improve solution quality. In this study, we use a numerical model for investigating the effect of delay feedback in the electronic amoeba. Our results show that delay feedback introduces a trade-off between the solution quality and the solution search time, and we find a way to change solution quality by tuning the delay time.

The electronic amoeba shown in **Fig. 1** electronically mimics the solution-searching behavior of the amoeboid organism *Physarum Polycephalum*[11]. The amoeba's core is a star topology network of pseudopod units with a hub current source. Using Kirchhoff's current law, the current conservation at the hub represents the volume conservation of the amoeboid organism. The charges in a capacitor of each pseudopod in the amoeba core represent a state variable, X_{Vk}. Each pseudopod unit has a sigmoid function on its end. The input of the sigmoid function is the amount of charges in a capacitor in the pseudopod unit, and the output of the function is sent to the IMC. The IMC is composed of a crossbar circuit with a resistor on each cross-point. The bounce-back rule is mapped by setting the resistance values that correspond to the weighting. The IMC performs product-sum operations of the weighted variables with an operational amplifier, carries out the threshold detection with a comparator, and returns the processed signal to the amoeba core to update the variables. In this process, a signal-propagation

delay occurs due to the parasitic capacitance (shown in **Fig. 1**). As a result, the state variables oscillate. So far, analog-digital hybrid, fully analog, superconducting, and fully digital electronic amoebae dedicated to solving optimization problems have been proposed[12–14].

The TSP is formulized as searching for a short route when a salesman should visit every city only once and then return to the city started at. Aono *et al*. have defined the bounce-back rule for the TSP and the satisfiability problem (SAT)[15–17]. In the TSP, when the number of cities N increases, the number of solution candidates increases in accordance with (N-1)!/2. This results in so called "combinatorial explosion"[18,19]. Nobody knows the exact algorithm for finding an optimal solution, the shortest route, in polynomial time. Algorithms that search for a semi-optimal solution in a short time (e.g., k-opt with simulated annealing, genetic algorithms, ant colony optimization, and particle swarm optimization) have already been widely studied[20–23].

We use N^2 variables to represent a solution to an N-city TSP instance as Hopfield recurrent neural networks[24,25]. X_{Vk} = 1 ($0 \le X_{Vk} \le 1$, $V = \{1, 2, ..., N\}$, $k = \{1, 2, ..., N\}$) corresponds to each unit in the amoeba core and denotes that a salesman visits V city at k-th order. In this study, the spatiotemporal dynamic behavior of the electronic amoeba is formularized as follows:

$$X_{Vk}(t+1) = \begin{cases} X_{Vk}(t) + \Delta_{in}/L_{off}(t+1) & (\text{if } L_{Vk}(t+1) = 0) \\ X_{Vk}(t) - X_{Vk}(t)/\Delta_{out} & (\text{if } L_{Vk}(t+1) = 1) \end{cases}, \quad (1)$$

$$L_{Vk}(t+1) = \sigma_{\alpha_1,\beta_1}\left(\sum_{Ul} W_{Vk,Ul} \cdot \sigma_{\alpha_2,\beta_2}\big(X_{Ul}(t-\tau)\big)\right), \quad (2)$$

$$L_{off}(t+1) = \sum_{Vk} \theta\big(1 - L_{Vk}(t+1)\big), \quad (3)$$

$$\sigma_{\alpha,\beta} = 1/\big(1 + \exp\big(-\alpha \cdot (x-\beta)\big)\big), \quad (4)$$

where Δ_{in} is the amount of resource supply and Δ_{out} is the amount of resource withdrawal, L_{off} ensures the current conservation at the network hub, τ is delay time larger than a time step, $\theta(x)$ is a threshold function with a threshold at 0.5, and $W_{Vk,Ul}$ is a variable interaction coefficient defined by the bounce-back rule. $\sigma_{\alpha,\beta}$ represents a sigmoid function where α is an inverse temperature and β is a threshold value. L_{Vk} is a bounce-back signal that decides whether X_{VK} is flipped from 1 to 0. Considering a delay in process, we formulate that the bounce-back signal controls the variable $X_{Vk}(t\text{-}\tau)$. When all the variables take either 0 or 1, we judge the system as having found a solution. In this study, X_{Vk} is 0 when $X_{Vk} < 0.2$, whereas X_{Vk} is regarded as 1 when $X_{Vk} > 0.8$. The variable interaction $W_{Vk,Ul}$ is defined as follows:

$$W_{Vk,Ul} = \begin{cases} 0.5 & (\text{if } V = U \text{ at } k \neq l \text{ or } V \neq U \text{ at } k = l) \\ v \cdot \text{dist}(V,U) & \big(\text{if } V \neq U \text{ and } (|k-l| = 1 \text{ or } |k-l| = n-1)\big), \\ 0 & (\text{otherwise}) \end{cases} \quad (5)$$

where dist(V,U) is the intercity distance between V and U, and v is a normalization coefficient that makes the system stable when it finds a solution[15,26]. The interaction $W_{Vk,Ul}$ is defined to forbid visiting the same city twice, forbid visiting different cities at the same time, and minimize the total path. v is defined as $v = 0.5/\max(\text{dist}(V,V') + \text{dist}(V',V''))$ ($V \neq V' \neq V''$), where max(x) returns the maximum value of x. This value depends on the instance, and it is pre-computed before the search for the solution. Its evaluation needs polynomial time $O(N^3)$. In the numerical simulation, we used the parameters $\Delta_{in} = 0.1$, $\Delta_{out} = 700$, $\alpha_1 = -1000$, $\beta_1 = 0.5$, $\alpha_2 = -35$, and $\beta_2 = 0.5$. In this study, we use negative logic for X_{Vk} and L_{Vk} because it is convenient for the physical circuit implementation. The sigmoid functions are simply represented using the inverter circuit (α_1 and $\alpha_2 < 0$) (shown in **Fig. 1**). The initial value of each element of $\{X_{Vk}\}$ was randomly assigned between $0.99 \pm 2.5\times10^{-3}$, and that of $\{L_{Vk}\}$ was assigned 1. In the numerical experiment, we used a randomly generated 10-city TSP instance whose average intercity distance was 100 and whose standard deviation was 17.

Figure 2 shows an example of the time evolutions of the variables $\{X_{Vk}\}$ in solving 10-city TSP for various lengths of delay τ. In this example, we used $v = 19.04 \times 10^{-4}$. Although all variables (100 variables) are displayed in the figure, they are almost overlapped. When $\tau = 0$, the variables did not oscillate; they immediately reached the stationary state, finding a solution (**Fig. 2(a)**). In contrast, when $\tau = 100$ and $\tau = 200$, the variables obviously oscillated and could reach the stationary state, shown in **Figs. 2(b)** and **2(c)**. From detailed observation, it was found that initially all the variables oscillated similarly, then gradually phased out from each other, and bifurcated. The variables involved in the same city bifurcated almost at the same time and followed the same curve. When τ was increased, both the number of oscillations and the oscillating period increased, shown in **Figs. 3(a)** and **3(b)**. These results confirmed that the search time to find a solution depended on the length of the delay. **Figure 3(c)** shows the solution quality and **Figure 3(d)** shows the number of iterations to find a solution as a function of the length of a delay. We observed that the solution quality improved and the number of iterations increased as τ increased. When using the metaheuristic algorithm to solve the optimization problem, solution quality is often proportional to the solution search time[20–23]. However, when $\tau = 300$, the variables continued oscillation without reaching the stationary state, as shown in **Fig. 2(d)**. In this case, the system could not find a solution. The results indicated that introducing appropriate feedback delays made it possible for the amoeba-inspired

computing system to improve solution quality through trial and error.

To further improve solution quality, we investigated a process of stabilizing the system that was in the oscillatory state during a long feedback delay. Here, we examined the scheduling of the delay length, inspired by the annealing schedule in the simulated annealing (SA)[27]. The delay length was initially set to be long enough to avoid the variables converging at a low-quality solution. Then, the delay length was decreased step by step, as shown in **Fig. 4(a)**. In this study, we defined 3 parameters for the scheduling: the maximum delay length and the step width and height in decreasing delays (hereafter called "width" and "height," respectively), as shown in **Fig. 4(a)**. **Figure 4(b)** shows an example of the time evolutions of $\{X_{Vk}\}$ when we scheduled the delays. The variables oscillated when the system started; however, the oscillations disappeared, and the system reached a stable state. **Figures 4(c)–4(f)** show the simulation results of changing the width and the height at the fixed maximum delay length. The scheduling parameters we used in **Figs. 4(c)** and **4(d)** are the maximum length of a delay of 300 and a height of 3 at fixed width, and those in **Figs. 4(e)** and **4(f)** are the maximum length of a delay of 300 and a width of 15,000 at fixed height. We improved the solution quality by increasing the width or decreasing the height. On the other hand, the number of iterations increased when the width was increased or the height was decreased, as shown in **Figs. 4(d)** and **4(f)**. The behavior we obtained was analog to the temperature scheduling in the SA[27]. By scheduling the delays, the system made a trade-off between the solution quality and the number of iterations (i.e., search time).

Finally, we investigated a possible approach to shifting the balance point of the trade-off between the solution quality and the search time. We expected the parameters in the system, Δ_{in}, Δ_{out}, α_1, β_1, α_2, β_2 and v, to affect the balance point. We examined the normalization coefficient v because we found that v significantly affected the solution quality. By increasing v, the optimality of the solution increased while the system sometimes reached an illegal candidate. **Figure 5** shows the simulation results of the solution searchability when v was offset as $v = 19.04 \times 10^{-4} + v_{offset}$. The delay scheduling parameters were the maximum delay of 300, the width of 25,000, and the height of 3. Increasing v_{offset} improved the solution quality, but the number of iterations remained almost unchanged (see **Figs. 5(a)** and **5(b)**). As **Fig. 5(c)** shows, the success rate of finding a solution was maintained when $v_{offset} \leq 4 \times 10^{-4}$. On the other hand, when $v_{offset} > 4 \times 10^{-4}$, the success rate decreased sharply, and the system could not find a legal solution. Therefore, the balance point of the trade-off between the solution quality and the number of iterations could be shifted by changing v_{offset}. The optimal value of v_{offset} was bounded by the legality of the solution, like a penalty term of Ising machines[28,29].

Previously, noises and/or errors have been used to perform global searches in various solution search machines. However, these approaches require many random signal generators that correspond to the number of the variables, which is an obstacle to physical implementation[30]. The delay feedback approach does not need such generators, and compact implementation of the system is feasible. There are several options for time-dependent control of the lengths of delays in a physical electronic system, the simplest being to control the current of the current source in the amoeba core. It is noted that, in the case of the electronic system using an asynchronous analog electronic circuit, the time constant of the capacitance in the amoeba core effectively defines one iteration in the computer model and the feedback delay τ is controlled by the parasitic capacitance mainly in the wiring between the amoeba core and the crossbar. Thus, the increase of the current in the network hub reduces the capacitance's charging time and results in relative increase of the feedback delay. Thereby, to reproduce the delay scheduling in Fig. 4(a), the current of the current source is gradually decreased with time. Similarly, gradual increase of the resistances in the amoeba core is also possible for scheduling.

In conclusion, we investigated the effect of feedback delays in an amoeba-inspired computing system, solving the TSP by forming an electronic amoeba with feedback delays and numerical simulations using this model. We showed that feedback delays caused the oscillations of the state variables; however, they improved the quality of the solution of the TSP. We found that feedback delay made a trade-off between the solution quality and the solution search time. We demonstrated that delay scheduling could further improve solution quality. The solution search without scheduling is one of the advantages of the amoeba-inspired computing system against the annealing machines[31,32], however, it turns out to be a disadvantage in the case of solving TSP because of the difficulty in controlling the quality of solutions. The delay scheduling should provide a feasible option to overcome the disadvantage of the amoeba-inspired system.

Acknowledgements

The authors would like to thank Dr. Masashi Aono from Amoeba Energy Co., Ltd. for his valuable discussion and comments. This work was partly supported by JSPS KAKENHI Grant Number JP18H0148700, Japan.

Reference

1) K. Pyragas, Phys. Lett. A **170**, 421 (1992).
2) F. M. Atay, J. Sound Vib. **218**, 333 (1998).
3) L. Appeltant, M. C. Soriano, G. Van Der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Nature Commun. **2**, 1 (2011).
4) D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, Nature Commun. **4**, 1 (2013).
5) G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Neural Networks **115**, 100 (2019).
6) B. S. Lee, J. Acoust. Soc. Am. **22**, 824 (1950).
7) T. Toya, D. Ishikawa, R. Miyauchi, K. Nishimoto, and M. Unoki, J. Signal Process. **20**, 197 (2016).
8) J. Foss and J. Milton, J. Neurophysiol. **84**, 975 (2000).
9) E. Thomas, *Applied Delay Differential Equations* (Springer, New York, 2009).
10) M. C. Mackey and L. Glass, Science **197**, 287 (1977).
11) S. Kasai, M. Aono, and M. Naruse, Appl. Phys. Lett. **103**, 163703 (2013).
12) K. Saito, N. Suefuji, S. Kasai, and M. Aono, Proc. 48th Int. Symp. Mult. Val. Log., 2018, p.127.
13) N. Takeuchi, M. Aono, and N. Yoshikawa, Phys. Rev. Appl. **11**, 044069 (2019).
14) A. H. N. Nguyen, M. Aono, and Y. Hara-Azumi, IEEE Access **4**, 49053 (2020).
15) L. Zhu, S. J. Kim, M. Hara, and M. Aono, R. Soc. Open Sci. **5**, 180396 (2018).
16) M. Aono, M. Naruse, S. J. Kim, M. Wakabayashi, H. Hori, M. Ohtsu, and M. Hara, Langmuir **29**, 7557 (2013).
17) M. Aono, S. Kasai, S. J. Kim, M. Wakabayashi, H. Miwa, and M. Naruse, Nanotechnology **26**, 234001 (2015).
18) J. B. Kruskal, Proc. Am. Math. Soc., 1956, p.48.
19) T. Bektas, Omega **34**, 209 (2006).
20) G. Reinelt, *The Traveling Salesman-Computational Solutions for TSP Applications* (Springer, Berlin, 1994).
21) D. Whitley, Stat. Comput. **4**, 65 (1994).
22) M. Dorigo and L. M. Gambardella, IEEE Trans. Evol. Comput. **1**, 53 (1997).
23) K. Wang, L. Huang, C. Zhou, and W. Pang, Proc. Second Int. Conf. Mach. Learn. Cybern., 2003, p.1583.
24) J. J. Hopfield, Proc. Natl. Acad. Sci. USA, 1982, p.2554.

25) J. J. Hopfield and D. W. Tank, Biol. Cybern. **52**, 141 (1985).
26) L. Zhu, M. Aono, S. J. Kim, and M. Hara, BioSystems **112**, 1 (2013).
27) S. Kirkpatrick, J. Stat. Phys. **34**, 975 (1984).
28) K. Takehara, D. Oku, Y. Matsuda, S. Tanaka, and N. Togawa, IEEE Int. Conf. Consum. Electron., 2019, p.64.
29) K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, J. Phys. Soc. Japan **88**, 061010 (2019).
30) Y. Hara-Azumi, N. Takeuchi, K. Hara, and M. Aono, Jpn. J. Appl. Phys. **59**, 040603 (2020).
31) M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, IEEE J. Solid-State Circuits **51**, 303 (2016).
32) M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, Front. Phys. **7**, 1 (2019).

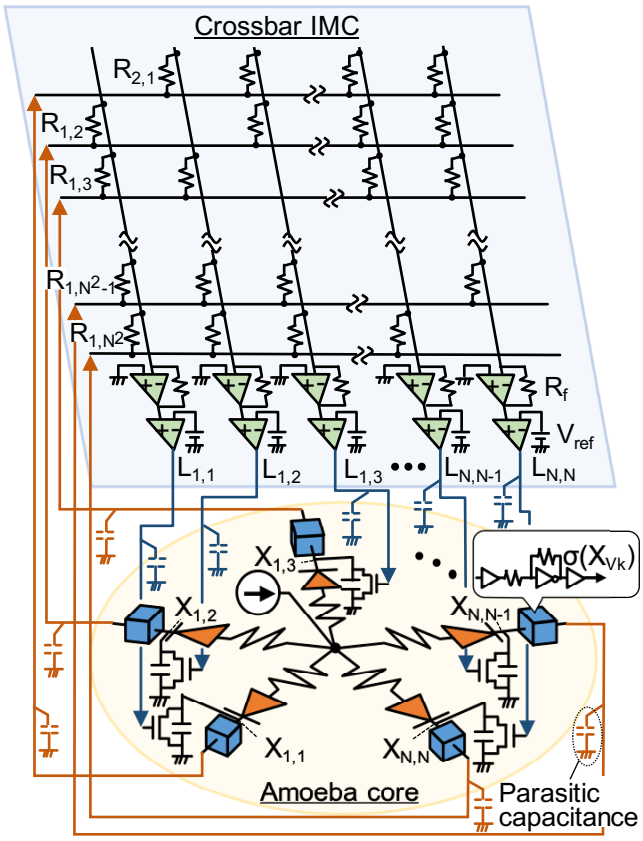

Fig. 1: Schematic view of the amoeba-inspired electronic computing system "electronic amoeba" for solving TSP. The electronic amoeba consists of an amoeba core that searches for a solution and an IMC that returns feedback signal to the amoeba core in accordance with a bounce-back rule. A cube in each branch of the amoeba core represents a sigmoid function.

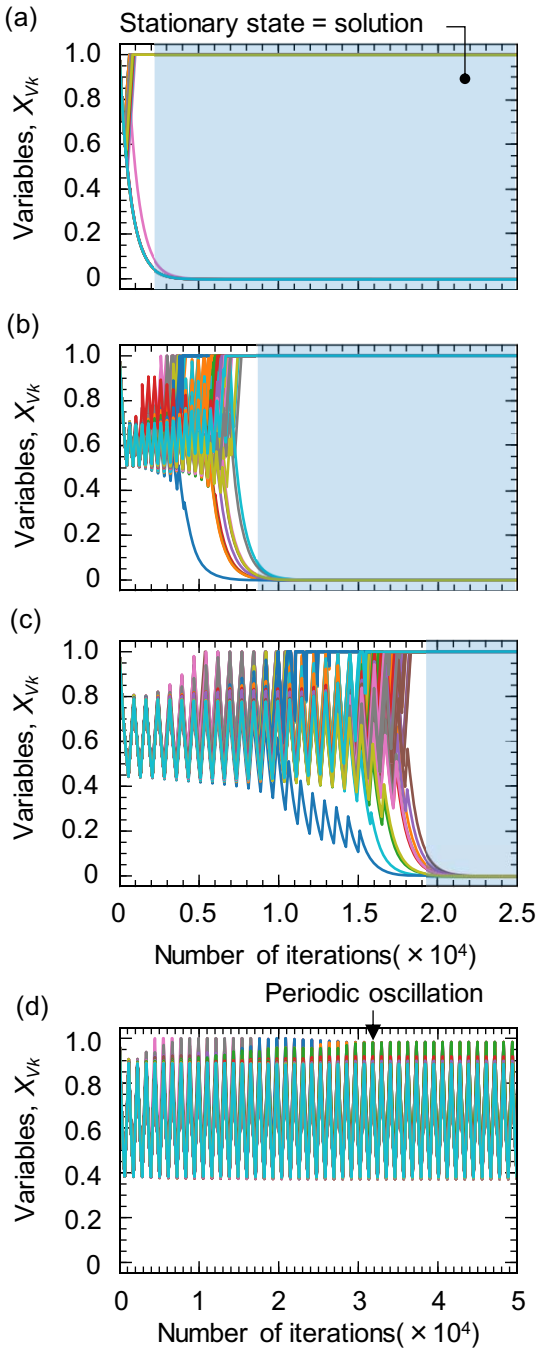

Fig. 2: Example of time evolution of X_{Vk} at (a) $\tau = 0$, (b) $\tau = 100$, (c) $\tau = 200$, and (d) $\tau = 300$. There were 10 cities. The system reached a stationary steady state (indicated by hatching) corresponding to a solution in (a)–(c). It fell into the oscillatory state and could not find a solution in (d). All variables (100 variables) are displayed in each plot.

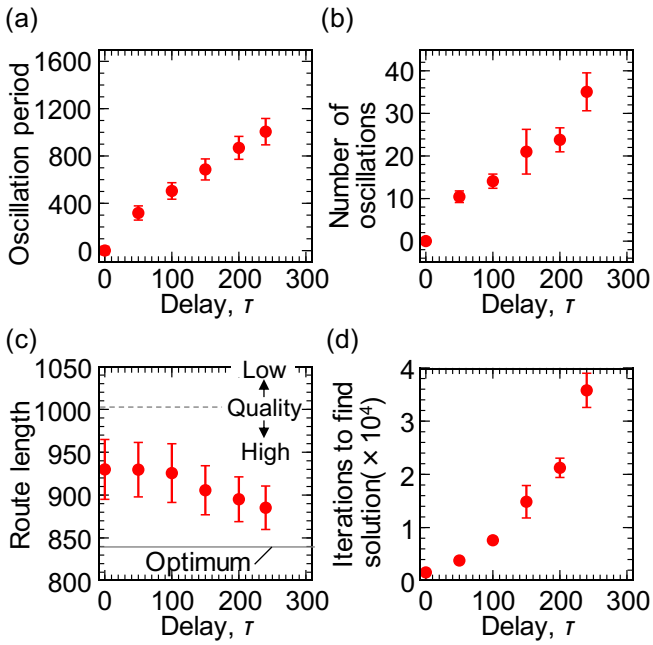

Fig. 3: Solution search performance as a function of the length of delay evaluated by solving 10-city TSP instance. The average number of (a) periods and (b) oscillations. (c) Solution quality. Solid line indicates the optimal route length and broken line indicates an average length of possible routes obtained from the random sampling algorithm after 10,000 trials. (d) The number of iterations to find a solution. Error bars are the standard deviation derived by 100 trials for each delay length.

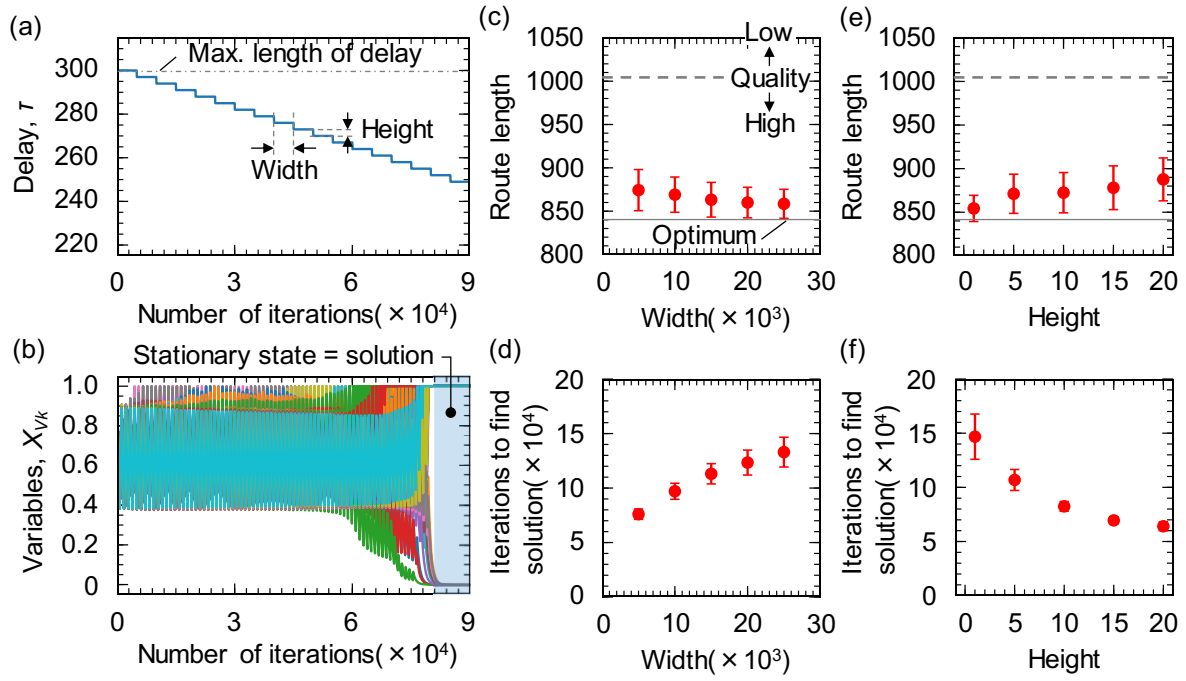

Fig. 4: (a) Delay scheduling and (b) example of time evolution of variables X_{Vk} under the delay scheduling with the maximum delay of 300, width of 5000, and height of 3. (c) Solution quality and (d) the number of iterations to find a solution as a function of width. (e) Solution quality and (f) the number of iterations to find a solution as a function of height. In (c) and (e), broken line is the average route length of the random sampling and solid line is the optimal route length of an optimal solution. Each data point was obtained from 100 trials.

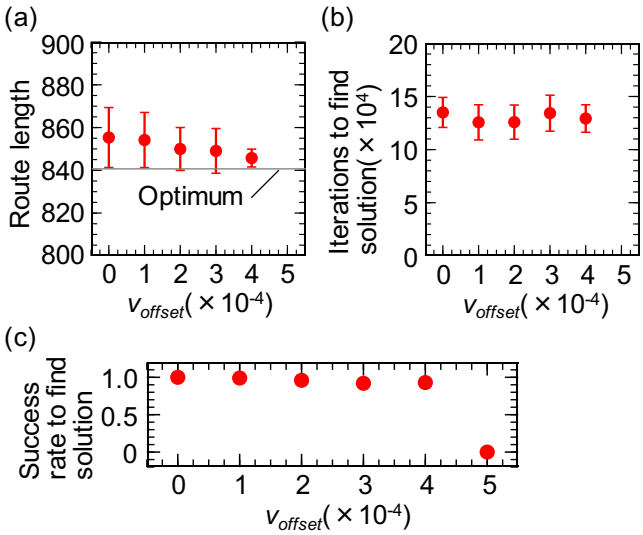

Fig. 5: Normalization coefficient offset v_{offset} dependence of solution-searching performance: (a) solution quality, (b) the number of iterations to find a solution, and (c) success rate of finding a solution. The scheduling parameters are the maximum delay of 300, width of 25000, and height of 3. Solid line in (a) is the optimal route length. Average solution quality was evaluated from the successful trials.