[Invited Talk]

Amoeba-inspired Heuristic Search for NP-complete Problem Solution at the Nanoscale

Masashi Aono[†‡], Song-Ju Kim[♯], Seiya Kasai[♮], Hiroyoshi Miwa[♭], and Makoto Naruse[§]

† Earh-Life Science institute, Tokyo Institute of Technology, 2–12–1 Ookayama, Meguro, 152–8550, Japan
‡ PRESTO, Japan Science and Technology Agency, 4–1–8 Honcho, Kawaguchi-shi, 332–0012, Japan
♯ WPI Center for Materials Nanoarchitectonics, NIMS, 1–1 Namiki, Tsukuba, 305–0044, Japan
♮ Graduate School of Information Science and Technology, and
Research Center for Integrated Quantum Electronics, Hokkaido University, North 14, West 9, Sapporo 060–0814, Japan
♭ Graduate School of Science and Technology, Kwansei Gakuin University, 2–1 Gakuen, Sanda, 669–1337, Japan
§ Photonic Network Research Institute, NICT, 4–2–1 Nukui-kita, Koganei, 184–8795, Japan
Email: masashi.aono@elsi.jp

Abstract—We formulated a heuristic search algorithm, AmoebaSAT, inspired by the spatiotemporal oscillatory dynamics of a single-celled amoeboid organism that exhibits sophisticated computing capabilities in adapting to its environment efficiently. AmoebaSAT finds a solution to an NP-complete problem, the satisfiability problem (SAT), at a speed that is dramatically faster than one of the conventionally known fastest stochastic local search methods for randomly generated 3-SAT instances. By implementing AmoebaSAT using various nanodevices, we aim to develop ultra-compact and ultra-low-power-consuming devices with ultra-fast computational speed.

1. Introduction

The satisfiability problem (SAT) is stated as follows: Given a logical formula f involving N variables x_i, does there exist an assignment $x_i \in \{1, 0\}$ (i.e., a combination of N *true*/*false* values) that satisfies f, which ensures that the overall formula f is true? For example, a problem instance $f = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_3 \vee x_4 \vee \neg x_1) \wedge (x_3 \vee x_4 \vee \neg x_1) \wedge (\neg x_4 \vee x_1 \vee \neg x_2) \wedge (x_4 \vee x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3)$ has the unique solution $(x_1, x_2, x_3, x_4) = (1, 1, 1, 1)$. In general, SAT instances involve logical formulae that have multiple satisfying solutions (assignments) or that have no satisfying solution.

As the value of N increases, the total number of possible assignments grows exponentially as 2^N and no polynomial-time algorithm for finding a solution is known. SAT belongs to the particularly difficult class of problems known as NP (nondeterministic polynomial time). Moreover, SAT was the first problem shown to be NP-complete; this means that any problem in NP may be reduced to SAT in polynomial time [1]. For this reason, fast algorithms and systems capable of solving SAT may be applied to solve an extremely large number of application problems including automatic inference, software/hardware verification, and information security.

2. Amoeba-inspired Algorithm

Aono et al. formulated the AmoebaSAT algorithm [2, 3], which utilizes the spatiotemporal dynamics of a coupled system of $2N$ units corresponding to pseudopod-like branches of a single-celled amoeba, to solve the N-variable SAT problem. Each unit is assigned a variable name $i \in \{1, 2, \cdots, N\}$ and a *truth*/*false* value $v \in \{0, 1\}$ and is associated with three variables $X_{i,v}$, $Y_{i,v}$, $Z_{i,v}$. If, at discrete time step t, a resource is supplied to unit (i, v) (corresponding to the elongation of the amoeba branch), we denote this by $Y_{i,v}(t) = 1$, and we interpret this as meaning that the system is considering the assignment $x_i = v$. If no resource is supplied, we write this as $Y_{i,v}(t) = 0$. We put $L_{i,v}(t) = 1$ or $L_{i,v}(t) = 0$ to indicate the application or non-application, respectively, of a stimulus that "bounces back" the supply of resources to $Y_{i,v}$ (corresponding to an optical stimulus inhibiting the elongation of the amoeba branch). State transitions of the values $Y_{i,v} \in \{0, 1\}$ are determined by a dynamics involving the following fluctuations:

$$Y_{i,v}(t) = \begin{cases} 0 & (\text{if } L_{i,v}(t) = 1), \\ sgn(1 - \epsilon - Z_{i,v}(t)) & (\text{if } L_{i,v}(t) = 0), \end{cases} \tag{1}$$

where $\epsilon = 0.25$ and

$$sgn(z) = \begin{cases} 1 & (\text{if } z > 0), \\ 0 & (\text{otherwise}). \end{cases} \tag{2}$$

Here the variable $Z_{i,v} \in [0.0, 1.0]$ obeys the following logistic map and gives rise to chaotic oscillations:

$$Z_{i,v}(t+1) = 4\, Z_{i,v}(t)(1 - Z_{i,v}(t)). \tag{3}$$

In short, if the bounceback stimulus is applied ($L_{i,v}(t) = 1$), the resource is not supplied ($Y_{i,v}(t) = 0$). If the bounceback stimulus is not applied ($L_{i,v}(t) = 0$), the resource is allocated ($Y_{i,v}(t) = 1$), unless the value of the logistic map evolution exceeds 0.75, in which case the supply of the resource is blocked ($Y_{i,v}(t) = 0$).

We also define a variable $X_{i,v}(t) \in \{-1, 0, 1\}$ to represent the accumulated value of the resource-supply variable

$Y_{i,v}(t)$:

$$X_{i,v}(t+1) = \begin{cases} X_{i,v}(t)+1 & (\text{if } Y_{i,v}(t+1)=1, \\ & \text{ and } X_{i,v}(t)<1), \\ X_{i,v}(t)-1 & (\text{if } Y_{i,v}(t+1)=0, \\ & \text{ and } X_{i,v}(t)>-1), \\ X_{i,v}(t) & (\text{otherwise}). \end{cases} \tag{4}$$

The quantity $X_{i,v}$ may be understood as an abstract representation of the displacement from the equilibrium volume of the amoeba branches with one of the three values $\{-1,0,1\}$. In each step, the variables $X = (X_{1,0}, X_{1,1}, X_{2,0}, X_{2,1}, \cdots, X_{N,0}, X_{N,1})$ are transformed into the variable assignments $x = (x_1, x_2, \cdots, x_N)$ according to the following rule:

$$x_i(t) = \begin{cases} 0 & (\text{if } X_{i,0}(t)=1 \text{ and } X_{i,1}(t)\le 0), \\ 1 & (\text{if } X_{i,1}(t)=1 \text{ and } X_{i,0}(t)\le 0), \\ x_i(t-1) & (\text{otherwise}). \end{cases} \tag{5}$$

Any logical formula that defines a given SAT instance is expressed set-theoretically by replacing the literals x_i and $\neg x_i$ with i and $-i$, respectively. For example, the above mentioned formula f is expressed in the form $F = \{\{1,2,-3\}, \{-2,3,-4\}, \{2,3,-4\}, \{-3,4,-1\}, \{3,4,-1\}, \{-4,1,-2\}, \{4,1,-2\}, \{-1,2,-3\}, \{1,2,3\}\}$, where each "clause" in f is represented by each element C in the set F. SAT in which no clause contains more than three literals are known as 3-SAT. It has been proven that 3-SAT is NP-complete [1].

For now, we wish to focus on the leftmost clause $(x_1 \vee x_2 \vee \neg x_3)$ in f. If we have both $x_1 = 0$ and $x_2 = 0$, then we require $x_3 = 1$ in order for this clause to be *true*; indeed, otherwise we find $(x_1 = 0 \vee x_2 = 0 \vee \neg x_3 = 0) = 0$. For this reason, if at step t we have both $X_{1,0}(t) = 1$ and $X_{2,0}(t) = 1$, then at step $t+1$ we apply a bounceback stimulus to $Y_{3,1}$ (i.e., we choose $L_{3,1}(t+1) = 1$). We call this rule a "bounceback rule". Similarly, from the leftmost clause we can read off the bounceback rules $X_{1,0}(t) = 1 \wedge X_{3,1}(t) = 1 \Rightarrow L_{2,0}(t+1) = 1$ and $X_{2,0}(t) = 1 \wedge X_{3,1}(t) = 1 \Rightarrow L_{1,0}(t+1) = 1$. We proceed similarly to investigate all clauses in f to analyze mutual interdependencies between the variables and determine a set of all bounceback rules, which are formally defined as follows [2, 3]:

$$L_{i,v}(t+1) = \begin{cases} 1 & (\text{if } B \ni (P,Q) \text{ such that } Q \ni (i,v) \\ & \text{ and } \forall (j,u) \in P (X_{j,u}(t)=1)), \\ 0 & (\text{otherwise}), \end{cases} \tag{6}$$

where

$$B = INTRA \cup INTER \cup CONTRA \tag{7}$$

represents the set of all bounceback rules; each of which is a pair (P, Q) that is interpreted as signifying that "if P is *true*, then Q is forbidden."

The subset of B named $INTRA$ is defined as follows to reflect the fact that each variable x_i is forbidden from taking the values 0 and 1 simultaneously. For all $i \in I$, we have

$$INTRA \ni (\{(i,v)\}, \{(i,1-v)\}). \tag{8}$$

The set $INTER$ is defined as follows to express interference between the variables in each clause. For each variable i contained in each element $C \in F$, we have

$$INTER \ni \begin{cases} (P, \{(i,0)\}) & (\text{if } C \ni i), \\ (P, \{(i,1)\}) & (\text{if } C \ni -i). \end{cases} \tag{9}$$

where P includes the following elements for all $j \neq i$:

$$P \ni \begin{cases} (j,0) & (\text{if } C \ni j), \\ (j,1) & (\text{if } C \ni -j). \end{cases} \tag{10}$$

If f contains both clauses involving x_i and clauses involving $\neg x_i$, then some rules in the set $INTER$ will serve to prohibit x_i from being assigned either of the values 0 or 1. For example, the two leftmost clauses in f, $(x_1 \vee x_2 \vee \neg x_3)$ and $(\neg x_2 \vee x_3 \vee \neg x_4)$, generate the rules $X_{1,0}(t) = 1 \wedge X_{3,1}(t) = 1 \Rightarrow L_{2,0}(t+1) = 1$ and $X_{3,0}(t) = 1 \wedge X_{4,1}(t) = 1 \Rightarrow L_{2,1}(t+1) = 1$, respectively. If we were to operate the system under these rules, a state in which we have $X_{1,0}(t) = X_{3,1}(t) = X_{3,0}(t) = X_{4,1}(t) = 1$ would lead to the application of both bounceback stimuli $L_{2,0}(t+1) = L_{2,1}(t+1) = 1$, thus yielding an inconsistent state in which variable x_2 is prevented from being assigned either 0 or 1. To avoid such inconsistencies, for each $i \in I$ we check the set $INTER$ and define the set $CONTRA$ as follows:

$$\begin{aligned} &\text{If } (P, \{(i,0)\}) \in INTER \text{ and } (P', \{(i,1)\}) \in INTER, \\ &\text{then } CONTRA \ni (P \cup P', P \cup P'). \end{aligned} \tag{11}$$

For any problem instance involving N variables and M clauses with all clauses having three literals (3-SAT), we have $\#(INTRA) = 2N$, $\#(INTER) = 3M$, $\#(CONTRA) < M^2$ (where # is the number of elements in a finite set). Consequently, assuming that M typically grows as a linear function of N, the time required to generate all bounceback rules, the memory required to store them, and the time required to implement controls at each step based on these rules all grow polynomially like $poly(N^2)$.

Under the bounceback rules defined above, if a system state $X = (X_{1,0}, X_{1,1}, X_{2,0}, X_{2,1}, \cdots, X_{N,0}, X_{N,1})$ satisfies, for all (i, v), either the condition $X_{i,v}(t) = 1 \Leftrightarrow L_{i,v}(t) = 0$ or the condition $X_{i,v}(t) \le 0 \Leftrightarrow L_{i,v}(t) = 1$, then the system is "stable". If this stability criterion is not satisfied, there is a high probability that the sign of $X_{i,v}(t+1)$ differs from that of $X_{i,v}(t)$ depending on $L_{i,v}(t)$, and the state X is unstable. Miwa et al. proved mathematically that the condition that AmoebaSAT "stabilizes" in a state satisfying the above stability criterion is equivalent to the condition that the state represents a "solution" to the SAT instance [4].

3. Performance evaluation

To evaluate the solution-searching performance of AmoebaSAT, we selected a group of problems known as Uniform Random-3-SAT, in which all clauses are formed from three literals from the benchmark problems offered by the online SATLIB library [5, 6]. We selected 100 instances

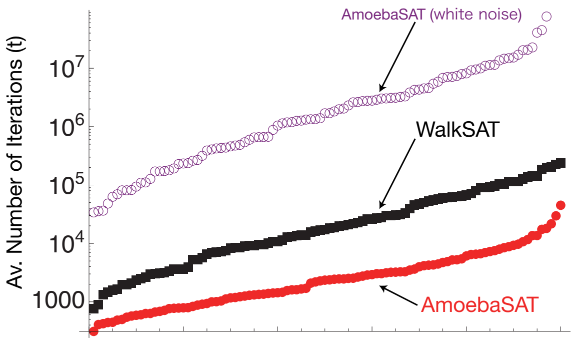

Figure 1: Performances for 100 randomly generated 3-SAT instances ($N = 50$).

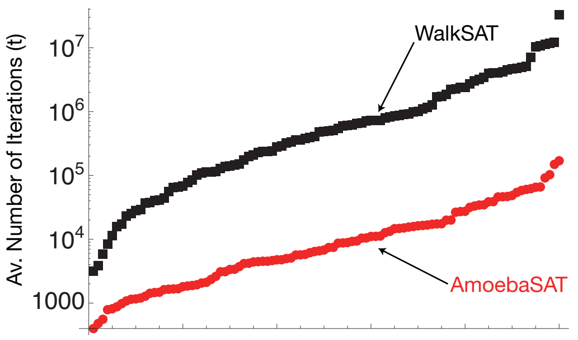

Figure 2: Performances for 100 randomly generated 3-SAT instances ($N = 75$).

with $N = 50$ variables and $M = 218$ clauses, and 100 instances with $N = 75$ variables and $M = 325$ clauses. The difficulty of finding a solution varies depending on the ratio of N and M; the presence of a phase transition in the vicinity of $M/N = 4.25$, near which the number of steps needed to find a solution suddenly skyrockets, has been previously reported [7]. Thus, the problem instances considered in this study, for which we have $M/N = 218/50 \simeq 4.36$ and $325/75 \simeq 4.333$, are considered to rank among the class of problems posing the high level of difficulty.

We considered WalkSAT, one of the categories of stochastic local search algorithms presently known to be the fastest heuristic methods for randomly generated 3-SAT instances [8]. WalkSAT configures its initial state by assigning all variables to random *true* or *false* values. Then the algorithm selects at random one clause from among the clauses that are not satisfied (i.e., are *false*) with the variable assignments at a given time and then chooses at random a single variable from within that clause to flip (changing 0 to 1 or 1 to 0). The algorithm then iterates this basic behavior.

For each problem instance, we ran 500 Monte Carlo simulations of both the AmoebaSAT and WalkSAT algorithms and compared the average number of time steps (number of iterations) t required to arrive at the solution. For the instances of $N = 50$ and $N = 75$, Figures 1 and 2, respectively, plot the number of steps needed to arrive at the solution, sorted in ascending order. The plots indicate that AmoebaSAT is able to find the solution with a speed orders of magnitude greater than that of WalkSAT. The difference in the solution-search performance between the two algorithms is more significant for $N = 75$ than for $N = 50$. Indeed, we observed that the impressive search performance of AmoebaSAT becomes even more remarkable as N increases. Moreover, although Figures 1 and 2 are comparisons of iteration counts, comparisons of the computation time required to run the two algorithms on identical machines also reveal that the time required for AmoebaSAT to find solutions is orders of magnitude less than that required for WalkSAT. These results will be reported elsewhere.

Understanding the origins of the high performance exhibited by AmoebaSAT is a subject of current investigation. Whereas WalkSAT only updates one variable in each step, AmoebaSAT incorporates many processes, which collectively update multiple variables and evolve simultaneously while interfering with each other through the bounceback control mechanism. Analytical results have been obtained that suggest that this unique "concurrent search" feature of the algorithm is the source of its high performance.

The statistical properties of the chaotic fluctuations that AmoebaSAT utilizes exhibit a significant influence on its performance. When the logistic map evolution $Z_{i,v}$ in equation (3) exceeds the threshold $1 - \epsilon = 0.75$ in equation (1), "error" occurs; even though the bounceback stimulus is not applied ($L_{i,v} = 0$) the resource is not supplied ($Y_{i,v} = 0$). The probability that $Z_{i,v}$ exceeds 0.75 is approximately $1/3$, which is understood as an "error probability" of resource supply when $L_{i,v} = 0$. In this paper we set ϵ to 0.25 not because the error probability needs to be $1/3$ but because the logistic map has an unstable fixed point at 0.75. As $Z_{i,v}$ tends to cross the fixed point repeatedly, when $L_{i,v} = 0$, it frequently happens that the value of $Y_{i,v}$ flips between 0 and 1; i.e., $Y_{i,v}(t + 1) = 1 - Y_{i,v}(t)$. Thus,$Y_{i,v}$ evolves in a negatively autocorrelated manner. Previously, some authors demonstrated the usefulness of fluctuations with "negative temporal correlations" for combinatorial optimization and Monte Carlo computation [9, 10]. Indeed, when the logistic map $Z_{i,v}$ in equation (1) was replaced by uncorrelated white noise, the performance of AmoebaSAT declined dramatically even though the error probability remained unchanged (i.e., we set $\epsilon \simeq 1/3$), as shown in Figure 1 (circles).

4. Nanodevice Implmentations

By developing information processing devices that make use of the physical properties and dynamics of nanomaterials and structures other than traditional CMOS devices, and by using these hardware devices to implement the amoeba-inspired algorithm, we can expect even greater efficiency, miniaturization, and reductions in energy consumption. In fact, spatiotemporal dynamics of photoexcitation (exciton) transfer, mediated by near-field optical interactions between multiple quantum dots, can be described by dynamics exhibiting nonlocal spatial correlations and bearing several similarities to the dynamics of intercellular resource flow in the amoeba; these authors showed that, by introducing bounceback control to the nanophotonic system of this type, the system can be made to search for solutions to constraint satisfaction problems and SAT [3]. The photoexcitation transfer in nanophotonics systems of this type can be realized with $1/10^4$ the power consumption of bit-flipping circuits in traditional electronic devices [11]. Applying a similar nanophotonics system to the multi-armed bandit problem, a decision problem whose objective is to maximize profit, the system was capable of efficiently and adaptively identifying the choice maximizing the probability of obtaining compensation [12]. On the other hand, in electronic systems, by utilizing charging and discharging processes in a network of nonlinear elements and capacitive elements, it is possible to search for solutions to constraint satisfaction problems [13].

5. Conclusions

In this paper, we showed that a solution-searching algorithm inspired by the behavior of an amoeboid organism is able to search for a solution to the satisfiability problem, an NP-complete problem known to be related to a variety of applications, with a speed orders of magnitude greater than that of a traditional stochastic local search method. Note that no algorithm for solving NP-complete problems in polynomial time has yet been discovered, and indeed the number of iterations required for AmoebaSAT to arrive at a solution do grow exponentially, as shown in Figures 1 and 2. However, the exponential growth of the number of iterations required to find a solution is dramatically reduced as compared to that of other algorithms; moreover, an additional advantage of AmoebaSAT is the possibility of allowing even faster computations to be implemented by ultra-miniature nanoscale devices with ultra-low power consumption. We hope this work will help to establish future computing paradigms.

Acknowledgments

This work was supported in part by the Core-to-Core Program from the Japan Society for the Promotion of Science.

References

[1] M.R. Garey, D.S. Johnson, "*Computers and Intractability: A Guide to the Theory of NP-Completeness*," W. H. Freeman and co., New York, 1979.

[2] M. Aono, S.-J. Kim, L. Zhu, M. Naruse, M. Ohtsu, H. Hori, M. Hara, "Amoeba-inspired SAT solver," *Proceedings of the 2012 International Symposium on Nonlinear Theory and its Applications*, pp.586–589, 2012.

[3] M. Aono, M. Naruse, S.-J. Kim, M. Wakabayashi, H. Hori, M. Ohtsu, M Hara, "Amoeba-inspired nanoarchitectonic computing: Solving intractable computational problems using nanoscale photoexcitation transfer dynamics," *Langmuir*, vol.29, pp.7557–7564, 2013.

[4] H. Miwa, M. Aono, M. Naruse, S. Kasai, "Amoeba-inspired Algorithm and Its Realization Using Nanodevice", *Technical Report of IEICE*, COMP2013-71, pp.77–82, 2014.

[5] http://www.cs.ubc.ca/ hoos/SATLIB/benchm.html

[6] H. H. Hoos, T. Stutzle, "SATLIB: An online resource for research on SAT," In: I. P., Gent, H. v. Maaren, T. Walsh (eds.) *SAT2000*, IOS Press, Amsterdam, pp.283–292, 2000.

[7] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, "Determining computational complexity from characteristic 'phase transitions", *Nature* vol. 400, pp.133–137, 1999.

[8] K. Iwama, S. Tamaki, "Improved upper bounds for 3-SAT", *Proceedings of 15th ACM-SIAM Symposium on Discrete Algorithms*, ACM, pp.328–328, 2004.

[9] M. Hasegawa, K. Umeno, "Solvable performance of optimization neural networks with chaotic noises and stochastic noise with negative correlation," *Lecture Notes in Computer Science*, vol.4984, pp.693–702, Springer, 2008.

[10] K. Umeno, "Performance of chaotic Monte Carlo computation and chaos codes for communications: Theory and experiments," *AIP Conf. Proc.*, vol.1339, pp.197–209, 2011.

[11] M. Naruse, H. Hori, K. Kobayashi, P. Holmstrom, L. Thylen, M. Ohtsu, "Lower bound of energy dissipation in optical excitation transfer via optical near-field interactions," *Opt. Express*, vol.8, pp.A544–A553. 2010.

[12] S.-J. Kim, M. Naruse, M. Aono, M. Ohtsu, M. Hara, "Decision maker based on nanoscale photo-excitation transfer," *Scientific Reports*, vol.3, 2370, 2013.

[13] S. Kasai, M. Aono, M. Naruse, "Amoeba-inspired computing architecture based on electron charge dynamics in parallel capacitance network," *Appl. Phys. Lett.*, vol.103, pp.163703, 2013.