2 Studies on Network Computational Machines Based on Molecular Interactions

SUZUKI Hideaki

Recent studies on network artificial chemistry (NAC) is surveyed. First, a model of active clusters created through the mathematical folding of node chains is presented, and next the studies on the rewiring rule of weak edges which constructs the base of the NAC's dynamics are summarized. The rule is formulated with the criterion of the minimization of newly defined network `energy´, and with experimental results, its effectiveness and limitation are discussed. Then we turn to a new scheme of the NAC that underwent the following modifications to the design. The former inactive solvent nodes are equipped with active (functional) programs, and finally the programs are implemented in agents that move through edges of the network and conduct the programs at nodes. This last model, "program-flow computing", can be applied to some computational problems.

Keywords
Artificial chemistry, Complex network, Agent, Self-organization, Program-flow computing

1 Creating a new algorithm based on molecular logic

1.1 Constructive approach

Each of the cells that make up our body is a gigantic machine in which several hundreds of millions of molecules (excluding water) interact in exquisite fashion in diverse biological processes, including genetic and metabolic activities. These machines are highly autonomous, adaptable, robust, reliable, and self-repairing, traits artificial machines have yet to approach. In recent years, researchers across a wide range of fields have pursued studies combining biology, chemistry, physics, and informatics to explore the design principles and mechanisms of the cell. Ambitious attempts in this area based on informatics include systems biology, which constructs detailed models of an entire living cell and performs simulations consuming vast computational resources. Nevertheless, even if we can assume infinite computational resources and copy all the events and processes that occur in a living cell to computer code, then run simulations, do we really understand the “design principle” of a cell? Do we understand the “reasons” for the diverse useful properties of the cells given above?

At the Bio-algorithms Project currently underway at the Kansai Advanced Research Center (KARC) within NICT, we pursue research using a bottom-up approach (the integration approach) (see Fig. 1). This approach diverges significantly from the techniques used to create the detailed model described above. The goal is to build a computational and communication device with the properties of a living creature. We investigate the details of cell activity at the molecular level and build

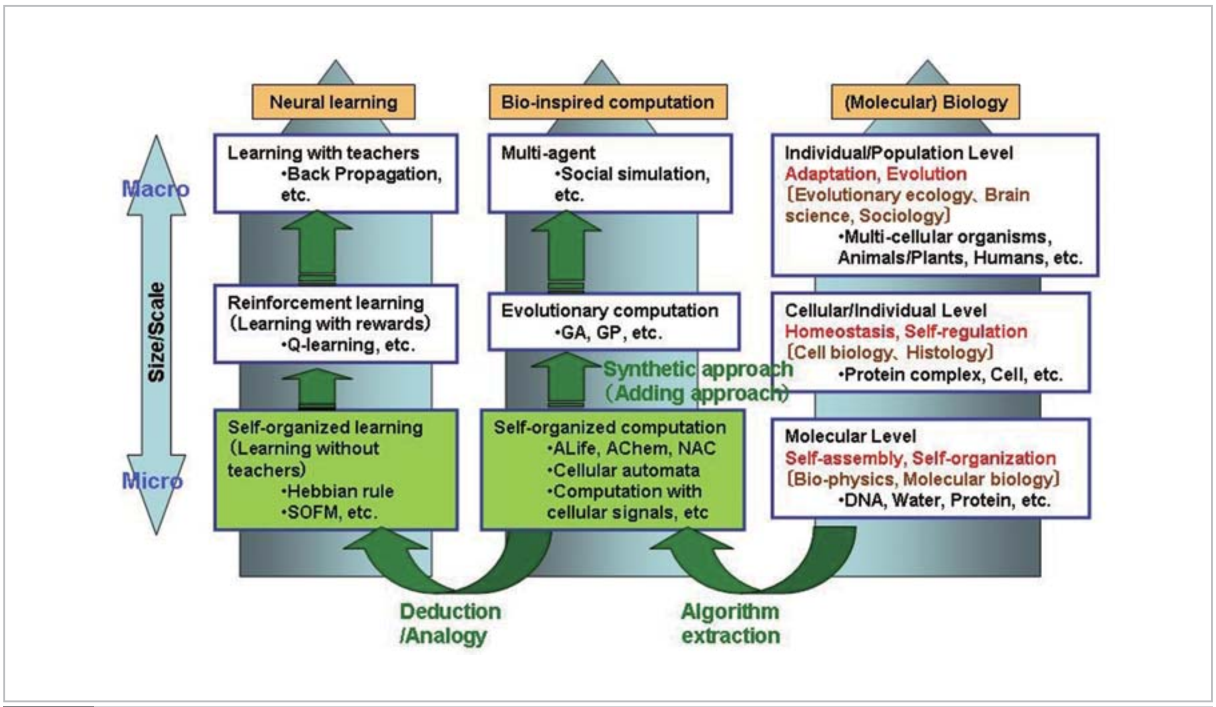
Neural learning
Bio-inspired computation
(Molecular) Biology
Macro
Size/Scale
Micro
Learning with teachers •Back Propagation, etc.
Reinforcement learning (Learning with rewards) •Q-learning, etc.
Self-organized learning (Learning without teachers) •Hebbian rule •SOFM, etc.
Multi-agent •Social simulation, etc.
Evolutionary computation •GA, GP, etc.
Self-organized computation •ALife, AChem, NAC •Cellular automata •Computation with cellular signals, etc
Synthetic approach (Adding approach)
Individual/Population Level Adaptation, Evolution [Evolutionary ecology, Brain science, Sociology] •Multi-cellular organisms, Animals/Plants, Humans, etc.
Cellular/Individual Level Homeostasis, Self-regulation [Cell biology, Histology] •Protein complex, Cell, etc.
Molecular Level Self-assembly, Self-organization [Bio-physics, Molecular biology] •DNA, Water, Protein, etc.
Deduction /Analogy
Algorithm extraction

Fig.1 *Conceptual diagram of bottom-up approach*

models of information processing and learning. The resulting model may be less fine-grained than a comprehensive model that describes every physical and chemical property of the molecules in question (at exorbitant computational cost), and yet is required to be detailed enough to reflect molecular logic and to exhibit desirable properties of bio-molecules such as self-assembly and self-organization. By constructing such models and assessing their suitability, we seek to better understand the essence of biological properties at the molecular level and perhaps to propose new (non-von Neumann) information processing and communication models.

1.2 Artificial life and artificial chemistry

Artificial life is among several historical efforts to capture the essence of life via a constructive approach[2]. In brief, artificial life seeks to introduce the principles of biological evolution into computing or other artificial systems and to perform simulations that shed light on the nature of life using a constructive approach, observing life-like behaviors that emerge in the simulations and applying them to design engineering systems. Artificial chemistry has recently emerged as a subfield whose goal is to model the conditions under which bio-chemical reactions and chemical evolution occurred on the primeval Earth, eventually leading to emergent phenomena such as cell evolution and genetic encoding[5]–[7]. In the design of self-organizing computational systems, the ideal model would be a chemical reaction system, confirmed to undergo self-organization or self-assembly and to produce complex structures and functions, composed of elemental processes whose mechanisms can be identified down to the level of quantum mechanics. Such a system could be used to design or evaluate models more accurate and reliable than the simplified systems (i.e., conventional artificial life) available now, which rely so heavily on human intuition.

In 2003, Suzuki et al. considered the environment required for such a system of artificial chemistry, distilling the essential characteristics to the five factors listed below[16]:

- Information space
- Elementary symbols
- Rules for transport of symbols

• Rules for reaction between symbols
• High-level manager that leads or modifies the elementary operations

Among the five factors, information space and the rules governing the transport of symbols in this space represent the most basic components, the foundations of the artificial environment. How these factors are modeled significantly affects model performance. Traditionally, the study of artificial life or chemistry used one-dimensional address spaces such as a core memory[12][14][15], two- or three-dimensional lattice spaces represented by cellular automata[10][9][13], or a tank structure with no difference in hierarchy between the encircled symbols[7][28]. These frameworks each have their advantages and disadvantages; none can flexibly express the formation of self-organizing walls or the transport or bonding of symbols[16].

1.3 Network artificial chemistry

To resolve these issues, Suzuki et al. have proposed network artificial chemistry (NAC), which expresses the relationship between the spatial positions of the symbols purely in terms of a topological network. Research driven by this approach is currently underway[17]–[27].

As is commonly known, solute molecules, which play the central role in bio-chemical reactions, undergo repeated collisions when moving through the solvent (water). Their movements are restricted by the physical properties of the three-dimensional solution space and the size and shape of each molecule. NAC replaces these constraints with rewiring rules for the edges of a network. Figure 2 compares and draws parallels between this solution system and NAC.

The networks used in NAC are graphs consisting of nodes and edges. The nodes represent molecules or atomic clusters, while the edges represent collisions or bonds. Since biological molecular interactions generally fall into one of four bonds — van del Waals, hydrogen, ionic, or covalent bonds — NAC has four edge types: **wa**, **hy**, **io**, and **cv**. The four edges are defined by different bond strengths. Weak edges are rewired one after another by a passive rule prepared in advance. Strong edges are formed by the active functions of the nodes and clusters.

With respect to representative research[23] for the early stages of NAC, Section **2** of this chapter presents the behavior of one-dimensional node chains converted into active clusters through folding. These function in the network like proteins, separating hydrophilic and hydrophobic regions or replicating molecular chains.

Section **3** presents research on the passive

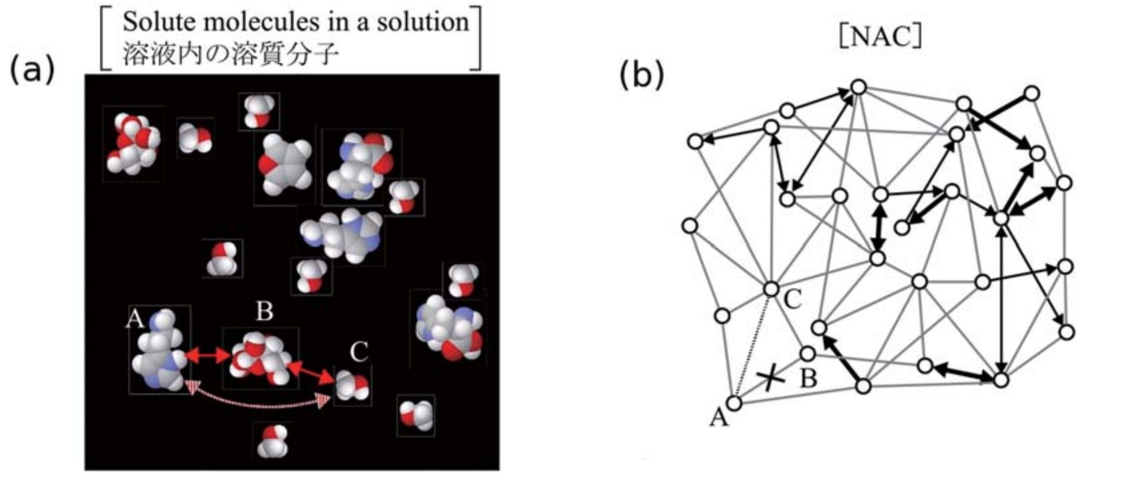

Fig.2 *Relationship diagram of (a) solute molecules in solution and (b) network artificial chemistry*

rewiring rule of the edges[24], a cornerstone of NAC dynamics. Bio-molecules corresponding to NAC nodes move and contact and collide with each other in a three-dimensional solution space due to Brownian motion. As indicated by the arrows (⇔) in Fig. 2 (a), a solute molecule is most likely to collide with a molecule located spatially close — in other words, with a molecule that has contacted or collided with the molecule with which it is now contacting or colliding. Remote molecules are unlikely to collide with each other within the near-future. To investigate how this spatial restriction affects network rewiring rules, we first perform a numerical simulation of a random walk for D-dimensional hard spheres. We then plot the probability that an edge is created or removed in the contact graph of the hard spheres as functions of degree, shortest distance, and second shortest path length. For the NAC graph, we define the "energy" of the graph. Using edge rewiring rules to minimize energy, we perform simulations and demonstrate that the probabilities of edge creation and removal by rewiring are in qualitative agreement with experimental results for the random walk. We also show that the values of the cluster coefficient (C) and the average path length (L) are brought closer to those of the contact graph of the hard spheres while maintaining the connectivity of the entire graph.

1.4 Revised network artificial chemistry

As discussed above, the primary factor in NAC dynamics that determines self-organization capabilities is the rewiring rule that governs weak edges (**wa** and **hy**). Although rewiring rules based on the network energy described in Section **3** have reproduced the qualitative nature of the graph observed in the contact graph of hard spheres, this achievement has been unsatisfactory when we view the resulting network as a virtual space that completely replaces Euclidean space. An example is liquid crystallization: Water at room temperature forms a network in which water molecules are loosely bonded by hydrogen bonds. At temperatures below 0 °C, this network becomes regular, forming a crystal structure (ice). This crystal structure is formed by the physical properties of the water molecules and three-dimensional spatial constraints. However, it is difficult to create such a regular structure in a graph without spatial information (angles, positions, etc.) based only on the energy minimization principle. The values of the cluster coefficient and the average path length obtained in the numerical simulation in Section **3** are also limited to the values of the small world[30] or slightly more regular value. We cannot make the values converge to a larger L value while maintaining the connectivity of the overall graph[21][24].

One of the reasons for this difficulty may be the specialty of the water molecules, ignored in the past NAC solvent node. Unlike organic solvents such as benzene, water is an abnormal liquid that establishes a pseudo-regular lattice over the space that it occupies based on hydrogen bonds. This nature is also the main cause of other behavior, including hydrophobic interactions and lipid bilayer formation. (Hydrophobic interactions are generally described as a phenomenon in which nonpolar molecules cannot join the regular network of the water molecules, but are instead repelled and thereby segregated.) Intermolecular interactions based on hydrophilic and hydrophobic properties constitute one of the basic dynamics of the bio-molecules. Ultimately, we might say that life underwent self-organization and evolution because water molecules are not static or passive but have properties that change under differing conditions. In the past, we considered only the data-flow cluster (⇔ proteins) as the active device in NAC (as in Section **2**), assuming no specific functions for the solvent node (⇔ water). However, for the graph to develop self-organized regularity (crystal structure) and a large L value, we must inevitably assign active functions to solvent nodes and control the peripheral edges.

Based on this idea, we try edge rewiring by executing a program (active function) assigned to the NAC nodes as described in Section **4**. The program design is modeled on the complex functions of water molecules. We investigate the possibility that a structure is formed in the network by executing the program[25][27]. As a target structure, we consider a two-dimensional square lattice and implement the node program in Java as a node instance to investigate how independent, parallel execution of the node program at all nodes creates a pseudo-regular structure.

Section **5** presents a new framework[26] for network artificial chemistry developed as an extension of these studies. Figure 3 shows the new relationship between the graph and the bio-chemical reaction system in the revised network artificial chemistry. In contrast to past frameworks, the nodes do not represent the molecules or atomic clusters directly. Rather, they represent minute spatial regions at the nanometer scale containing several to several dozen molecules. Molecules or atomic clusters are expressed as agents moving within the network. Collectively, the edges represent contact or bonding between molecules or atomic clusters belonging to the spatial regions in question. The functional programs are not assigned to fixed notes but implemented in agents moving along the edges within the network. When an agent arrives at a node, the program fires (reacts) or is executed, rewiring the edges. The experiment designs programs for three types of molecules, based on this framework, and demonstrates how hydrophilic cluster structures are organized, then split, by centrosomes. If we consider the nodes to be CPUs, the edges communication paths, and the agents programs, this model is a computational model, a “program-flow computing” model in which many CPUs execute in parallel many types of programs moving through communication paths connecting the CPUs.

2 Control flow cluster as active machine

2.1 Introduction

Section **2** describes a model in the network artificial chemistry in which one-dimensional node chains are converted to form clusters by “folding” and function as an active machine[23]. The clusters created are bound with strong edges and function as data-flow machines by operating in parallel while exchanging tokens along the edges. This data-flow machine can perform the active processing in NAC, such as node modification and edge rewiring. The folding proposed here is a type of conversion from a genotype (node chain) to a phenotype (data-flow machine) and provides a method for introducing a large functional machine into NAC.

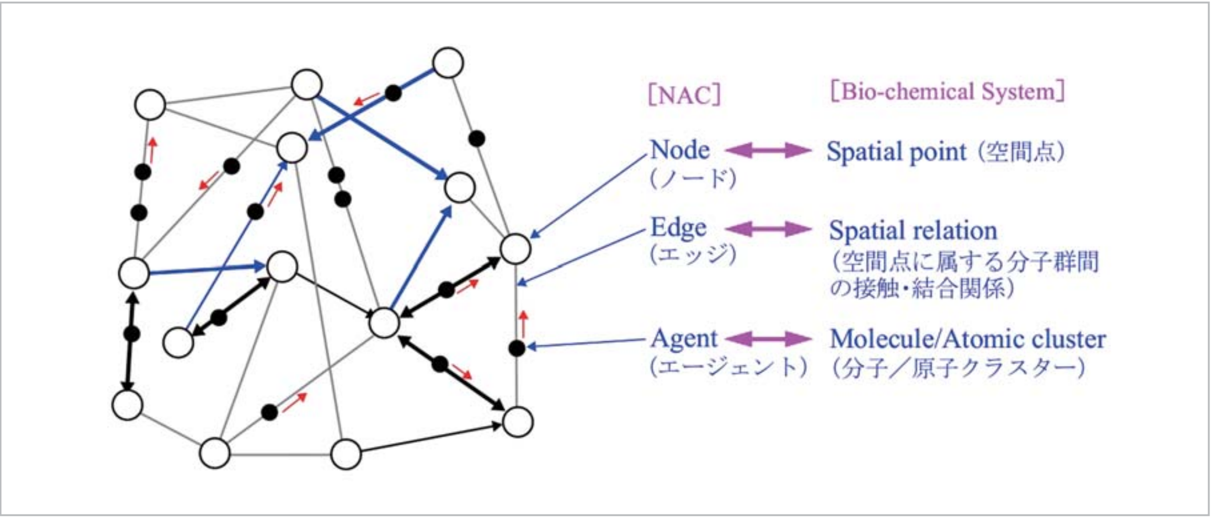

Fig.3 *Relationship diagram of revised network artificial chemistry and bio-chemical reaction system*

Section **2.2** below briefly describes the basic design of NAC. Section **2.3** describes the folding algorithm of the node chain in detail; Section **2.4** describes the operation of the data-flow machine in detail; Section **2.5** discusses several experimental results; and Section **2.6** presents a conclusion and summarizes the significance of the model.

2.2 Basic model

The NAC graph used in this section consists of two types of nodes and three types of edges. Using an analogy drawn from living creatures, the nodes are classified as hydrophilic and hydrophobic nodes and the edges as Covalent (**cv**, directional, covalent bond), Hydrogen (**hy**, directional, hydrogen bond), or van del Waals (**wa**, non directional, van del Waals bond) edges, in order of decreasing strength.

The **wa** edges of NAC are locally rewired by repeating the process sequences given in Steps (1) through to (4) below.

(1) Reference node, A, is randomly selected from the network.

(2) Node, B, with maximum degree is selected from the adjacent nodes of A, connected by **wa** edges.

(3) Node, C, with minimum degree is selected from nodes connected by two or three **wa** edges from A.

(4) The hydrophilic and hydrophobic properties of Nodes A and C are investigated. If they satisfy specific conditions for bonding, Edge A-B is cut and Edge A-C created.

Here, the bonding conditions for Step (4) are set so that the bonding is formed preferentially in the order of hydrophilic- hydrophilic, hydrophobic- hydrophobic, and hydrophilic-hydrophobic[11]. Under this rewiring rule, the graph evolves to have strong small-world properties[30] and uniform degree distribution.

2.3 Mathematical folding of node chain

NAC expresses the genetic information in terms of node chains, as shown in Fig. 4. Each node has a character or a character string composed of functional characters (**a**, **b**, ...) and template characters (**0**, **1**, ...). Each of the functional characters has a pre-defined function, but they do not operate in the form of the node chain.

Such node chains are folded after processing as shown in Fig. 5. First, the "agglomeration" process divides the adjacent node sequences directly before a functional charac-

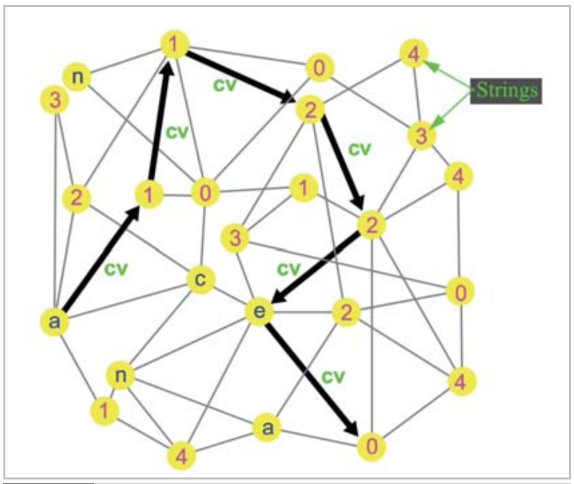

Fig.4 Example of NAC node chain

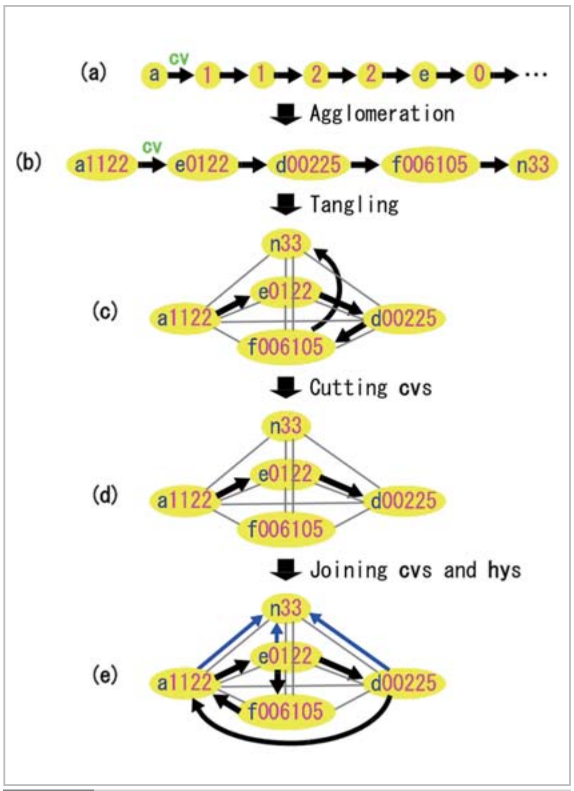

Fig.5 Node chain folding process

ter and converts each of the divided node chain sections into a single node with character strings such as **a1122** or **e0122** (Fig. 5 (b)). Then, the **wa** edges link each of the node pairs in the node chain in (b), tangling the node chain ("tangling"; Fig. 5 (c)). The template characters in each node are checked, and if "5" is included, the **cv** edge directly after is cut (cutting **cvs**; Fig. 5 (d)).

Once the node chain is ready for folding in this manner, the "folding" process creates the **cv** and **hy** edges, based on complementary matching between the template character strings. Here, the template characters, **0**, **1**, **2**, and **3**, are matched complementarily with the relationship of **0** ⇔ **1**, and **2** ⇔ **3**. Nodes with matched templates are newly linked to **cv** or **hy** edges. The directions of the edges go from a template starting with **0** to the template starting with **1** and from the template starting with **2** to the template starting with **3**. The **cv** or **hy** edges are created in this direction. In the example of Fig. 5, new **cv** edges are formed from **00** to **11** and **01** to **10**, and new **hy** edges are created from **22** to **33**.

2.4 Operation of data-flow clusters

When the node chain is folded into a node cluster, it operates as a data-flow machine functioning in parallel while transmitting tokens along the **cv** edges. In general, when a node receives a token, it fires if the value is "true," executing a routine assigned to the functional character (**a**, **d**, ...). The value of the token is set according to the result and passed on to the next node.

Figure 6 shows an example of a data-flow machine, called "*splitase*," generated by folding in the previous section. This machine consists of an active site (**n33**) and three operation nodes (**a1122**, **e0122**, **d00225**, and **f006105**). The operation nodes share a single active site via the hy edges and execute various processes to external nodes through this site. The processing starts with Node **a1122** firing and the active site rewiring the hy edge to another operand node. Then, node **e0122** fires, and it is checked whether the new operand node is

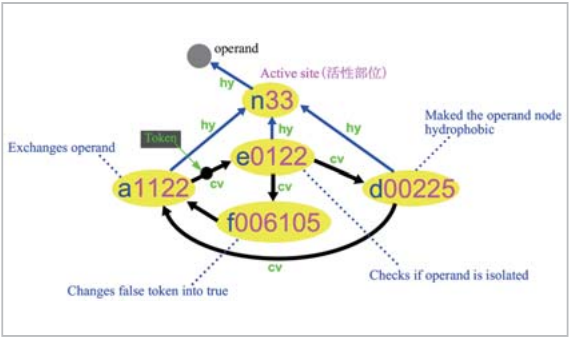

Fig.6 *Example of data-flow machine, "splitase"*

isolated with respect to **cv**. If so, the "true" token is transmitted from Node **e0122** along the **cv** edges. If it is not isolated, a "false" token is transmitted instead. When Node **d00225** receives the true token, it operates on the operand node, changes its property to hydrophobic, and then returns the token to Node **a1122**. On the other hand, Node **f006105** has the function to fire when it receives a false token. If this node fires, the token value is converted to "true" and transmitted to Node **a1122**. This data-flow cluster has the capability of changing the operand nodes adjacent to Active site **n33** to hydrophobic, one after another. With this capability, oily molecules are rapidly produced one after another and cause cell division.

Figure 7 shows the example of designing the "*replicase*" data-flow cluster. The cluster consists of three active site nodes and 16 operation nodes. The basic replication processing of the node chain is based on that proposed by Hutton[8]. The **cv** loop composed of Nodes 11 through to 17 on the right half of this machine plays the main role. With the nodes in this part operating one after another by transmitting the tokens, the active site randomly creates **hy** edges to nearby nodes, compares the character strings with the chain node, creates **cv** edges if they are the same, and moves one step along the chain. When the replication of the node chain is completed, the active site moves away from the node chain (cuts the **hy** edge) and searches for another node chain.

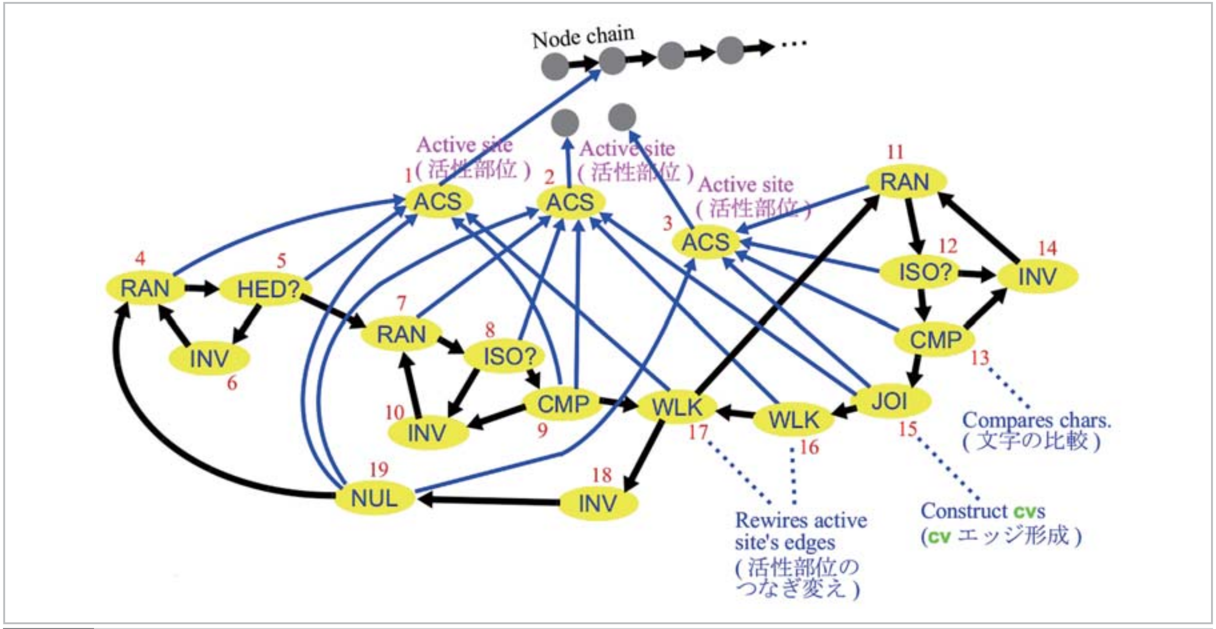

Fig.7 Design example of data-flow machine, "replicase"

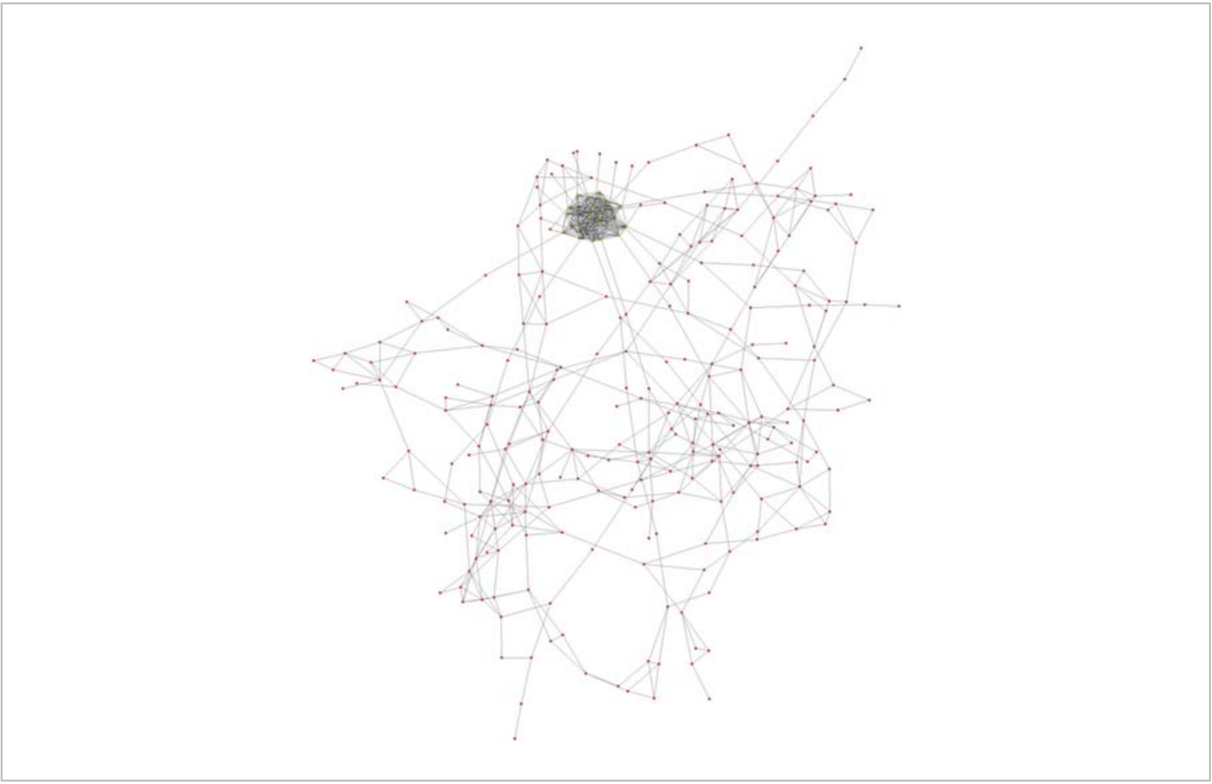

Fig.8 Example of separation of network into hydrophilic and hydrophobic regions

2.5 Experiment

2.5.1 Network separation

We perform a network separation experiment using the *splitase* shown in Fig. 6. A node chain consisting of 26 nodes is introduced into an initial random network. After agglomeration, tangling, and **cv** cutting, the following three processes are performed: (1) rewiring of passive **wa** edges; (2) folding by **wa** edge connection; and (3) active processing by token transmission. Repeating these three processes changes the network topology and

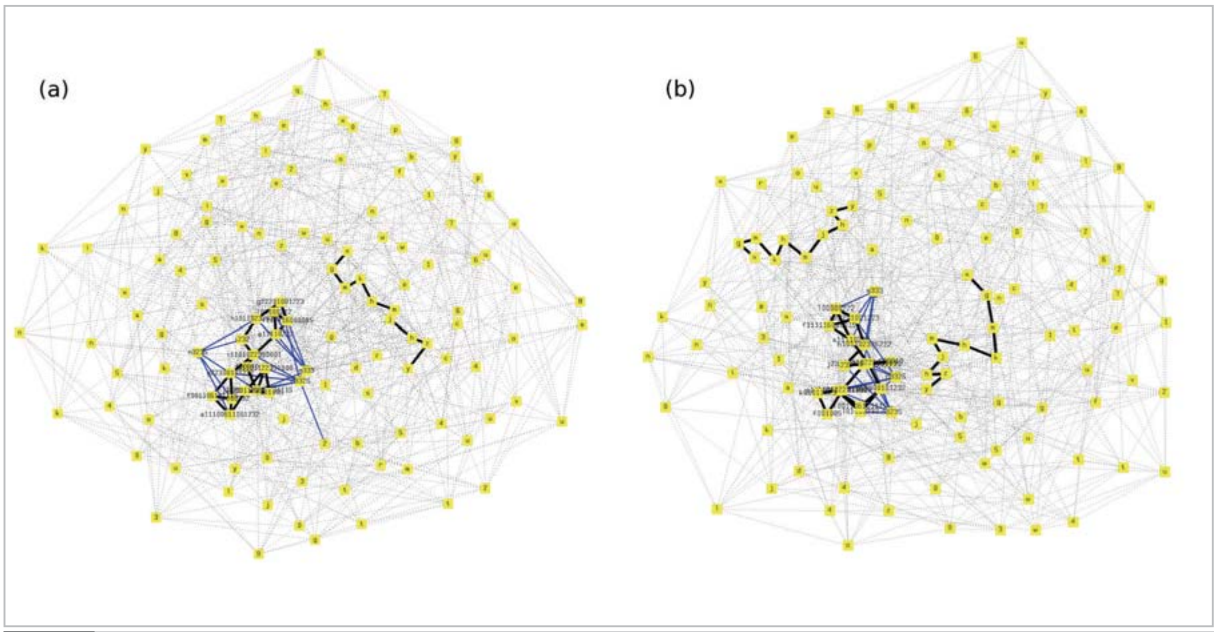

Fig.9 *Example of node chain replication by replicase*

causes its structure to self-organize. Figure 8 shows a typical result of this experiment. As the hydrophobic nodes are generated by the *splitase*, the graph is separated into hydrophilic and hydrophobic regions and results in the structure with the hydrophilic cluster enclosed in the hydrophobic network.

2.5.2 Replication of node chain

A *replicase* node chain consisting of 191 nodes and an unfolded operand node chain consisting of 10 nodes are introduced into an initial random network and processed in a manner similar to the previous sub-section. Figure 7 shows a typical result. In this example, the *replicase* cluster creates replicated node chains with the same character strings as the operand node chain in the network.

2.6 Conclusion and discussion

Here we proposed data-flow clusters as active machines operating in network artificial chemistry and presented algorithms for constructing them by mathematical folding of one-dimensional node chains. Based on this proposed method, we designed *splitase* and *replicase* and experimentally demonstrated their operation.

This mathematical folding, which is analogous to the folding of bio-protein molecules, has the following common features with actual protein molecules:

[Completeness of expression]

When clusters are constructed based on the algorithm indicated in this section, a node cluster (partial network) of any topology can be formed in principle if appropriate sequences are prepared. This means that based on the difference in one-dimensional sequences, mathematical folding can construct clusters of varying shapes, analogous to the diversity of protein functions, which causes the fantastic variation in the shapes of bio-proteins or life forms.

[Use of complementary matching]

The biological and mathematical foldings are based on the complementary matching of shapes and characters, respectively. The folding of a biological protein, which refers to the shape, size, and physical and chemical properties of the amino-acid residue constituting it, uses complementary matching for determining a matched residue. If the chain is folded with better matching, the resulting protein is thermodynamically stable. The folding shown in Fig. 5 (b) to (c) is a simplified counterpart of this process. In the study of artificial life, several systems use such methods to determine relationships between character strings (for example[12]). This algorithm is one such system.

3 Network rewiring rule based on energy minimization

3.1 Introduction

As an example of research on the passive rewiring rule for edges, this section describes research achievements associated with a model that rewires edges based on the criterion of minimizing the energy defined for the network[24].

Section **3.2** below simulates a random walk of hard spheres moving in Euclidean space and investigates how the spatial structure restricts the contact relationship of the spheres. Section **3.3** formulates the rewiring rule of the NAC edges based on the energy; Section **3.4** discusses the results of NAC experiments using this formulation; and Section **3.5** presents a summary of Section **3**.

3.2 Random walk simulation of hard spheres

3.2.1 Experimental method

To model molecular diffusion in a solution (Brownian motion) by a random walk in Euclidean space, we consider a D-dimensional continuous space that satisfies the periodic boundary condition. In the space, we introduce N D-dimensional hard spheres of radius R_1. For each iteration, each hard sphere moves its center coordinates (hops) by a randomly-chosen hopping vector. If the hopping of the sphere reduces the distance between the center points of the two hard spheres to below $2R_1$, the hopping is cancelled. To simplify the problem and reduce computational costs, R_1 and hopping vector length Δ are assumed to be constant for all spheres and all hops. Additionally, the direction of the hopping vector is not arbitrary, but randomly selected from $2D$ discrete values (a pair of positive and negative values for each dimension).

Collisions and contacts involving hard spheres are defined using contact radius R_2. When Euclidean distance d between the center points of a pair of hard spheres satisfies the condition $2R_1 \le d \le 2R_2$, the pair is regarded to be in contact. In particular, when only a further hard sphere cannot enter between the two hard spheres, the two spheres are regarded to be in contact, and R_2 is assumed to equal $2R_1$ ([4]; p. 890 of Japanese translation). While the contact graph thus created changes over time according to the random walk model, we measure several graph characteristics.

First, we measure cluster coefficient C and average path length L based on the usual definition[30]. In addition, the probability of edge joining P_{join} and the probability of edge cutting P_{cut} are calculated as follows. Assume that Node **A** hops at certain intervals and a new edge is created between Nodes **A** and **C**. Here, we measure the shortest path length l between Nodes **A** and **C**, and the degree k of Node **C** directly before the edge is created, then add 1 to the frequency matrix $N_{\text{join}}(l, k)$. Here, we measure l and k not just for Node **C**, but for all nodes in the graph, adding 1 to the frequency matrix $N(l, k)$. At the end of the simulation, we calculate $P_{\text{join}}(l, k) = N_{\text{join}}(l, k)/N(l, k)$ to obtain the expectation value of the edge creation probability P_{join}. Similarly, when the hopping of Node **A** at certain intervals removes the edge **A-B**, we measure the second shortest path length, l_2, between Nodes **A** and **B** and degree k of Node **B** directly before the edge removal, then add 1 to the frequency matrix $N_{\text{cut}}(l_2, k)$. We measure l_2 and k for all nodes adjacent to Node **A** and add 1 to the frequency matrix $N_2(l_2, k)$. At the end of the simulation, we calculate $P_{\text{cut}}(l_2, k) = N_{\text{cut}}(l_2, k)/N_2(l_2, k)$ to obtain the expectation value of edge removal probability P_{cut}.

3.2.2 Parameter setting

Among the important parameters required for the experiment are the typical values for the ratio R_1/Δ of the radius of the hard spheres and hopping vector length, which we calculate based on the assumption that the hydrophilic head of the lipid molecule (solute) is surrounded by water (solvent) and is characterized by Brownian motion. If we denote the number of collisions between the solute molecule and the solvent molecule per second as z, the radius of the solute molecule as r_1, and the average relative velocity of the solute molecule with respect to the solvent molecule as Vr. Here, z equals the average number of solvent molecules in a cylinder with radius $R_1 + r_1$ and height $|V\text{r}|$ and can be expressed as shown in Fig. 10 (a). Here, ρ is the number of solvent molecules per unit volume. Vr is the composite vector of the average velocity vector V of the solute molecules and the average velocity vector v of the solvent molecules. In this case, we can assume that the directions of the solvent molecules are completely random. Thus, v and V can be assumed to be orthogonal to each other. The resulting expression is as given in Fig. 10 (b). Here, we use the principle of equipartition of energy. $(1/2)M|V|^2 = (1/2)m|v|^2$, to estimate the velocity ratio as shown in Fig. 10 (c). $M = 255$ and $m = 18$ are the molecular weights of the lipid head ($C_8O_6PNH_{18}$) and water (H_2O), respectively. The equations shown in Fig. 10 (a) and (b) give the R_1/Δ ratio shown in Fig. 10 (d). Based on this reference value, this experiment selects values for radius, contact radius, and hopping vector length as $R_1 = 4.0$, $R_2 = 8.0$, and $\Delta = 0.1$, respectively.

We can evaluate average degree $\bar{k}$ of the contact graph if the hard spheres are randomly distributed in space as follows. Denote the volume of the D-dimensional sphere with radius r as $V_D(r) = \pi^{D/2} r^D/(D/2)!$. The state in which N hard spheres with volume $V_D(R_1)$ are found in a cube with sides of length X can be approximated by the state in which N points with zero volume are in the volume of $X^D - N \bullet V_D(R_1)$ based on number density. Here, number density is expressed as shown in Fig. 10 (e). $\bar{k}$ is this density multiplied by the volume of the D-dimensional spherical shell of inner radius greater than or equal to $2R_1$ and outer radius less than or equal to $2R_2$. This is obtained as shown in Fig. 10 (f). From this

(a)
$$z = \pi(R_1 + r_1)^2 \cdot |\boldsymbol{V}_\mathrm{r}| \cdot \rho$$

(b)
$$|\boldsymbol{V}_\mathrm{r}| = \sqrt{|\boldsymbol{V}|^2 + |\boldsymbol{v}|^2} = \sqrt{1 + \frac{255}{18}} \cdot |\boldsymbol{V}| = 3.89 \cdot |\boldsymbol{V}|$$

(c)
$$\frac{|\boldsymbol{v}|}{|\boldsymbol{V}|} = \sqrt{\frac{M}{m}} = \sqrt{\frac{255}{18}}$$

(d)
$$\begin{aligned} \frac{R_1}{\Delta} &= \frac{R_1}{|\boldsymbol{V}|/z} \\ &= \frac{R_1}{|\boldsymbol{V}|} \cdot \pi(R_1 + r_1)^2 \cdot |\boldsymbol{V}_\mathrm{r}| \cdot \rho \\ &= 3.89 \cdot R_1 \cdot \pi(R_1 + r_1)^2 \cdot \rho \\ &= 3.89 \times (3.63 \times 10^{-10}[m]) \times 3.14 \\ &\quad \times (3.63 \times 10^{-10}[m] + 1.5 \times 10^{-10}[m])^2 \times 3.34 \times 10^{28}[1/m^3] \\ &\simeq 39.0 \end{aligned}$$

(e)
$$\frac{N}{X^D - N \cdot V_D(R_1)}$$

(f)
$$\bar{k} = \frac{N}{X^D - N \cdot V_D(R_1)} \times (V_D(2R_2) - V_D(2R_1))$$

Fig.10 *Setting of parameters for hard sphere random walk simulation*

equation, for example, under the condition $D = 3$, $X = 93.0$, $N = 200$, $R_1 = 4.0$, and $R_2 = 8.0$, the average degree is estimated to be $\overline{k} = 4.0$.

3.2.3 Simulation results

The experiment is performed with both $D = 2$ and $D = 3$. In both cases, the C and L values are confirmed to have the following characteristics. The C value increases to 10 to 20 times that for the random graph. The L value maintains a value approximately two-fold that for the random graph. These results indicate that the contact graph of the hard spheres is in the middle of a small world network[30] ($C \gg C_{\text{rand}}$, $L \approx L_{\text{rand}}$) and the regular network ($C \gg C_{\text{rand}}$, $L \gg L_{\text{rand}}$).

Figure 11 shows results for edge joining and cutting probabilities for the experiment with $D = 3$. (We confirmed that the results for $D = 2$ are similar.) Based on the results, where l is small, P_{join} (l, k) decreases exponentially as l increases and decreases linearly as k increases. This result is intuitive based on what we know of molecular collisions. The collision probability is regarded to decrease as intermolecular distance increases due to the increase in l and with declining effective cross sections of collisions due to contact resulting from increased k. Here, according to Fig. 11 (a), in the region of $l = 2$ and k ~ large, P_{join} increases slightly. This result is only observed under these particular conditions and is regarded to be attributable to noise. In the region with large l, P_{join} increases once, due to confined space. If the size of the space and the number of nodes in the graph are infinite, N (l, k) in the denominator increases infinitely as l increases. However, in a finite graph, N (l, k) decreases in a region with l ~ large, and P_{join} is thus regarded to increase based on the actions of a small number of exceptional nodes.

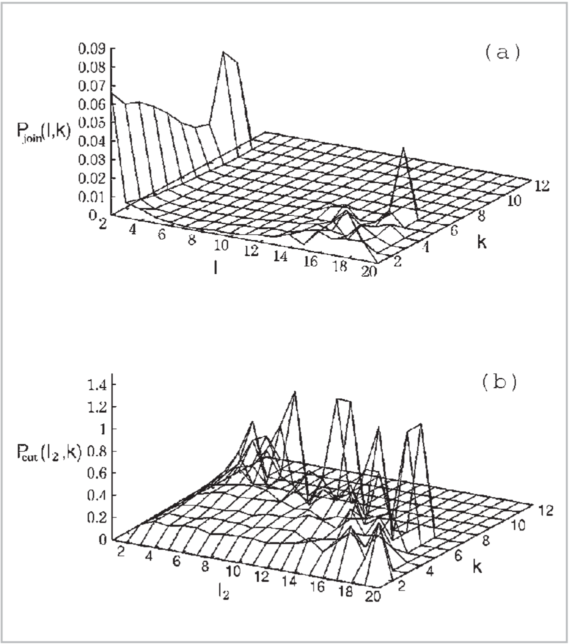

Fig.11 Result of hard sphere random walk simulation

Figure 11 (b) shows that P_{cut} (l_2, k) is kept small in the region with l_2 ~ small but rapidly increases when l_2 is above a certain value. The threshold value of l_2 decreases as k increases. For example, when $k = 8$, P_{cut} (l_2, k) is extremely large, even when $l_2 = 3$. This result can be understood by the “unnaturalness” of the graph topology with spatial restrictions considered. As the C and L values discussed above indicate, the contact graph of hard spheres has a type of regularity attributable to spatial restrictions. One conspicuous characteristic of a D-dimensional regular graph is that the nodes are distributed almost evenly in D-dimensional space and therefore contains many paths of similar path length as the shortest path between a node pair. (In other words, the frequency is quite large near the shortest path in the histogram of the path length between node pairs.) An edge with a large second shortest path length (l_2 ~ large) surrounded by many nodes (k ~ large) contradicts this characteristic, and cutting such an edge with high probability can bring the contact graph of hard spheres closer to the regular graph.

3.3 Edge rewiring rule

3.3.1 Mean free path of nodes

Mean free path as known in the world of statistical physics is the quantity defined as the average distance a molecule propagates without colliding with another molecule. It is a parameter determined solely by the size and density of the molecules and does not depend on temperature. The NAC graph is a model of the contact graph of molecules or atomic clusters in three-dimensional space. NAC expresses the collision of molecules and atoms as edge generation. Here, we assume that the spatial distance between a pair of molecules or atoms can be emulated by the shortest path length l between the node pair. We also assume that the node pair for which the new edge generation is attempted is selected in proportion to the free path function, $P_{\rm fp}\ (l) = \exp\ (-l\ /\ \lambda)$. Here, λ is the NAC mean free path. If we use only this condition, we need to search node pairs with extremely large path length l on the graph, although this is extremely rare. We can use restriction $l \leq L_{\max}$ to reduce computational costs.

3.3.2 Network energy

The energy of the NAC graph is defined as the sum of the spatial restriction energy $E_{\rm s}$ related to topology and edge bonding energy $E_{\rm b}$ as shown in Fig. 12 (a).

$E_{\rm s}$ is expressed as the sum of the term (μ-term) to suppress fluctuations in the degree of each node and the term (ν-term) to produce regularity, as shown in Fig. 12 (b). Here, $\sum_n$ indicates the sum over all nodes n while $\sum_{<mn>}$ indicates the sum over all adjacent node pairs mn. k_n is the true degree of Node n, $\overline{k_n}$ the expectation value of the degree of Node n (potentially differing depending on the node); $\overline{k}$ the average of the degree over all nodes; and $(l_2)_{mn}$ the second shortest path length between the node pair mn. When the constant α is positive, the μ-term has the function of making the node degree closer to the target value, $\overline{k_n}$ (When hard spheres of the same size are distributed in space, a sphere cannot randomly contact many other spheres), and the ν -term has the function of cutting the edges between nodes with large degrees on both ends and a large second shortest path length with high probability.

The bonding energy of edge $E_{\rm b}$ is

(a)
$$E = E_{\rm s} + E_{\rm b}.$$

(b)
$$E_{\rm s} = \frac{\mu}{N}\sum_{n}|k_n - \overline{k_n}|^{\sigma} + \frac{\nu}{N}\sum_{<mn>}\left\{\left(\frac{k_m k_n}{\overline{k}^2}\right)^{\gamma}(l_2)_{mn}\right\}^{\alpha}$$

(c)
$$E_{\rm b} = -\frac{1}{N}\sum_{<mn>} e_{mn}$$

(d)
$$e_{mn} = \begin{cases} \epsilon^{(\textsf{cv})} & \text{if } mn \text{ is } \textbf{cv} \\ \epsilon^{(\textsf{io})} & \text{if } mn \text{ is } \textbf{io} \\ \epsilon^{(\textsf{wa})} & \text{if } mn \text{ is } \textbf{wa} \end{cases}$$

(e)
$$\Delta E_{\rm b} = \begin{cases} \frac{1}{N}\epsilon^{(\textsf{cv-act})} & \text{for joining a } \textbf{cv} \text{ edge} \\ \frac{1}{N}\left(\epsilon^{(\textsf{cv-act})} + e_{mn}\right) & \text{for cutting a } \textbf{cv} \text{ edge} \\ -\frac{1}{N}e_{mn} & \text{for joining an edge other than } \textbf{cv} \\ \frac{1}{N}e_{mn} & \text{for cutting an edge other than } \textbf{cv} \end{cases}$$

(f)
$$P(\Delta E) = \begin{cases} \kappa\exp(-\beta\Delta E) & \text{if } \Delta E \geq 0 \\ \kappa & \text{if } -dE_{\rm th} \leq \Delta E < 0 \\ 1 & \text{if } \Delta E < -dE_{\rm th} \end{cases}$$

Fig. 12 *(a)-(e) Network energy equations and (f) acceptance probability of modified metropolis method* expressed as shown in Fig. 12 (c). Here, e_{mn} is the bonding energy of Edge *mn* (the energy required to cut the edge) and is defined as shown in Fig. 12 (d) according to the type of the edge ($0 \leq \varepsilon^{(wa)} \leq \varepsilon^{(io)} \leq \varepsilon^{(cv)}$).

E is used in the next sub-section to calculate the change ΔE associated with joining or cutting of the edge. In particular, when calculating the joining or cutting of the **cv** edges, we add activation energy $\varepsilon^{(cv\text{-}act)}$ and evaluate the entire ΔE_b as shown in Fig. 12 (e).

3.3.3 Modified Metropolis algorithm

To minimize network energy defined in the previous sub-section stochastically, here we modify the commonly used Metropolis method. This method regards the edge joining and cutting as chemical reactions and applies a rule similar to one formulated based on the results of the chemical reaction velocity theory[4][29].

The edges are rewired by the two iterations below repeated at every iteration.

(1) [Edge joining]

A node is randomly selected in the graph and another node is randomly selected with a probability proportional to $P_{fp}(l)$ where *l* is the path length between the two nodes. Based on the change in network energy ΔE before and after an edge is created between the two nodes, we can calculate acceptance probability $P(\Delta E)$ and join the edge with this probability.

(2) [Edge cutting]

An edge is randomly selected from the graph and the change in network energy ΔE in the cut edge is calculated. From this change in energy, we calculate acceptance probability $P(\Delta E)$, with which the edge is cut.

Here, for probability $P(\Delta E)$, we slightly modify the acceptance probability of the Metropolis method, as shown in Fig. 12 (f). By setting the probability ($\kappa < 1$) for $\Delta E \geq -dE_{th}$ to differ from the probability for $\Delta E < -dE_{th}$, we can approximately express the rapid increase in probability, as observed in Fig. 11 (b).

The change in energy ΔE associated with the edge joining or cutting can be calculated from the equations shown in Fig. 12 (a) through to (e). Strictly speaking, the change in the ν-term in the equation shown in Fig. 12 (b) is a non-local quantity that requires evaluation over all edges in the graph. Calculating this quantity accurately demands a large load in view of computational costs. Thus, this section considers the partial graph shown in Fig. 13. To evaluate the $\sum_{<mn>}$ when joining or cutting edge **A-B**, we add only the contributions from the edges that appear in the partial graph. The preliminary results of the NAC rewiring experiment confirm that $\Delta E'$ calculated as such gives a good approximation sufficiently close to the true ΔE. To reduce computational costs, we also restrict the value of l_2 to be $l_2 \leq L_{max}$.

3.4 NAC simulation

This section performs NAC rewiring simulations based on the energy minimization criterion as indicated in the previous sub-section. The parameter values used in the experiment are determined as follows, based on the results of the preliminary experiment: $\lambda = 2$, $L_{max} = 15$, $\mu = 0.01$, $\sigma = 4$, $\nu = 0.02$, $\bar{k} = 4$, $\gamma = 0.1$, $\alpha = 2$, $\kappa = 0.2$, $\beta = 1500$, and $dE_{th} = 0.0015$. Since this section focuses on spatial restriction energy, use only the **wa** edges and set bonding energy to $\varepsilon^{(wa)} = 0$.

As the initial state, we start from a random graph with the number of nodes $N = 200$ and average degree $\bar{k} = 4$. Figure 14 plots the clus-

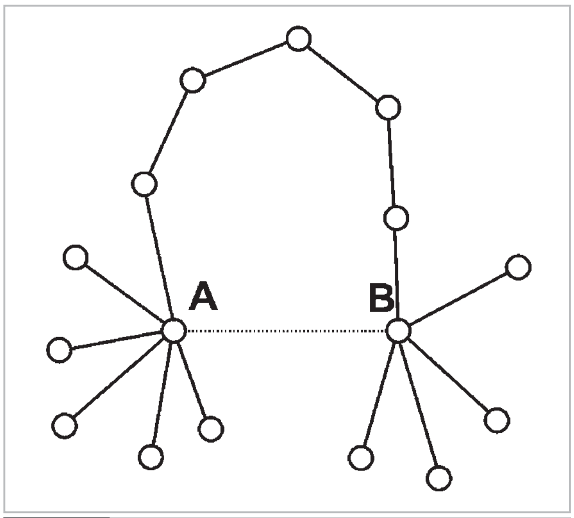

Fig.13 Nearby section graph for approximate calculation of change in ν-term of spatial restriction energy

ter coefficient (C) and average path length (L) with respect to time. According to Figure 14, after approximately 100 thousand iterations, the C value of the NAC graph is approximately 19.4 times the initial value ($C \approx 0.018$) for the random graph, and the L value is approximately 1.5 times the initial value ($L \approx 3.95$) for the random graph. This shows that the graph has characteristics midway between small world and regular graphs.

Figure 15 also shows the expectation values of P_{join} and P_{cut} measured in the simulation. The results qualitatively agree with the results of the random walk simulation (Fig. 11).

Figure 16 shows the initial NAC graph and the NAC graph after 80 thousand iterations projected onto a plane. The graph after rewiring is bunched with single links and contains fewer edge crossings, resulting in a topology easier to project onto a plane.

3.5 Conclusion and discussion

We derived and formulated the edge rewiring rule in the "network artificial chemistry," which describes the relationship between molecules and primitive clusters by a network, based on the criterion of minimizing network energy. We performed a rewiring simulation based on the rule obtained and found that cluster coefficients and average path length values are close to those for the contact graph of hard spheres. We also found that the dependence of the edge creation and removal

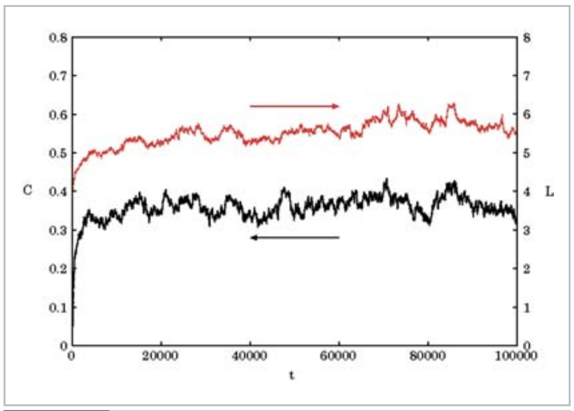

Fig. 14 Measurement of changes in cluster coefficient (C) and average path length (L) during 100,000 steps in NAC rewriting simulation

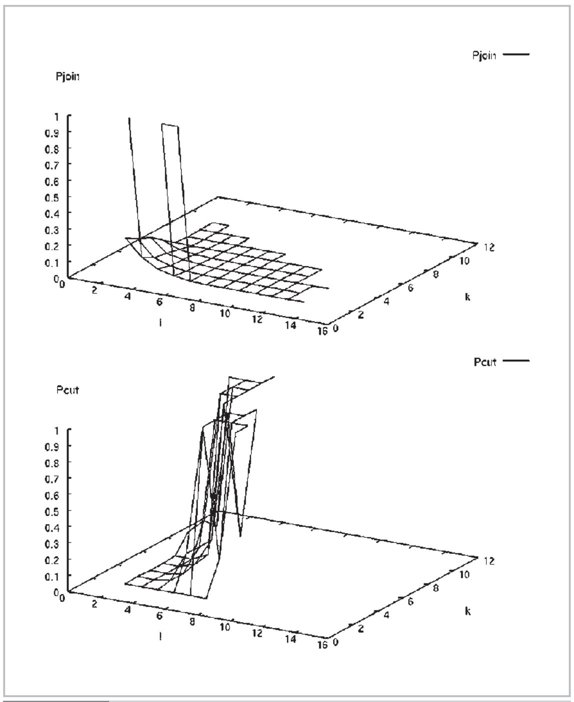

Fig. 15 Edge joining and cutting probabilities calculated based on NAC rewriting simulation result

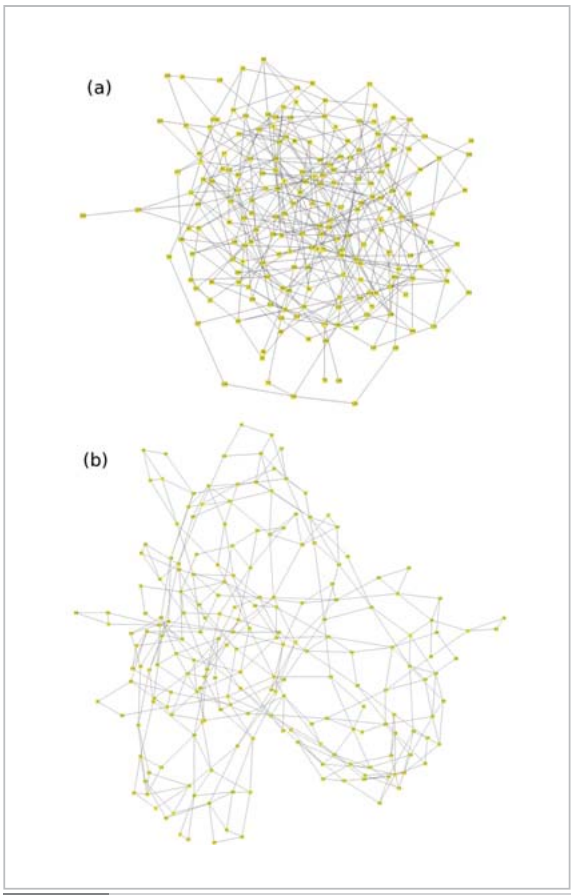

Fig. 16 Two-dimensional rendering of (a) initial random graph and (b) NAC graph after 80,000 rewriting steps

probabilities on the parameters (degree and path length) qualitatively agree with those of a random walk simulation of hard spheres in *D*-dimensional space. We drew the resulting graph using commercially available software and found that the graph has fewer edge crossings when projected onto a plane than at the initial state. The graph becomes one with less tangling, one more easily embedded in space.

We can conclude that the rewiring rule formulated reverses the Watts-Strogatz scenario[30] and increases *C* and *L* values to create a graph with a certain type of spatial restriction.

4 Organization of network structure by active node program

4.1 Introduction

This section considers NAC solvent nodes as active nodes and designs the node program. We let all the solvent nodes have the program designed with reference to the complex functions of water molecules and demonstrate how its parallel execution rewires the edges and forms a pseudo-regular structure in the network[25][27].

Below, Section **4.2** gives an overview of the model, Section **4.3** shows the experimental results, and Section **4.4** concludes with a discussion of the problem.

4.2 Model

The NAC program is implemented in Java. Each node or edge is expressed as an instance of a Java node class or edge class. These instances are linked to each other with instance variables. The node instance has a pointer array **hy** [ ] of size 4 and can indicate up to 4 **hy** edges connected to the node. Figure 17 shows the node program algorithm used to implement rewiring.

This algorithm essentially extracts the path **nde** - **this** - **nda** - **ndb** - **ndc** - **ndd** with reference to **this** node and creates a new edge for the path to form a loop of length 4. Here, **dp** is the variable that specifies the direction for searching the path. The two hy edges adjacent to each other in a loop are registered in the **hy** [ ] of the intermediate node to places differing by **dp**.

4.3 Experimental results

We start from a random graph with number of nodes $N = 512$ and uniform degree $K = 4$ and run **conduct_nd_prog** ( ) *K* times in each iteration in each node. Figures 18 and 19 show typical results.

As these figures show, rewiring by **conduct_nd_prog** ( ) gradually creates regularity within the graph. After approximately 100 iterations, the graph assumes the structure of a two-dimensional square lattice sheet folded as shown in Fig. 19. Beyond this stage, the

```
public void conduct_nd_prog()
{
  // choosing pointer's displacement
  int dp = [either -1 or 1];

  // choosing this node's initial
  //   adjacent node randomly
  Node nda = ad_rand();

  // choosing the next node
  Node ndb = nda.ad_next(this,dp);
  if(ndb==null) return;

  // choosing the second next node
  Node ndc = ndb.ad(nda,dp);

  // choosing this node's another
  //   adjacent node
  Node nde = ad_next(nda,-dp);
  if(ndc==null OR ndc==this){
    if(nde==null) return;
    if(nde==ndb) return;

    remake_edge(ndb,nde);
    return;
  }

  // choosing the third next node
  Node ndd = ndc.ad_next(ndb,dp);
  if(ndd==this) return;

  remake_edge(this,ndc);
}
```

Fig.17 Algorithm of node program

sheet is gradually unfolded while maintaining regularity; after approximately 1,000 iterations, it assumes a structure combining sheets and strings, as shown in Fig. 20. The current algorithm **conduct_nd_prog** ( ) specifies only the local links of the graph and does not specify the overall structure. It does not distinguish between two-dimensional sheets and one-dimensional strings as the overall structure. Several experimental results show that the current algorithm tends to converge the graph to a string, not a sheet.

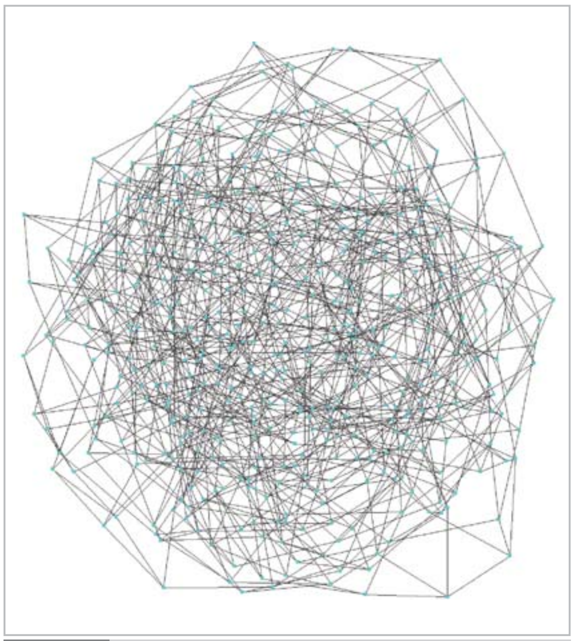
Fig. 18 Two-dimensional rendering of initial random graph based on the number of nodes N = 512 and uniform degree K = 4

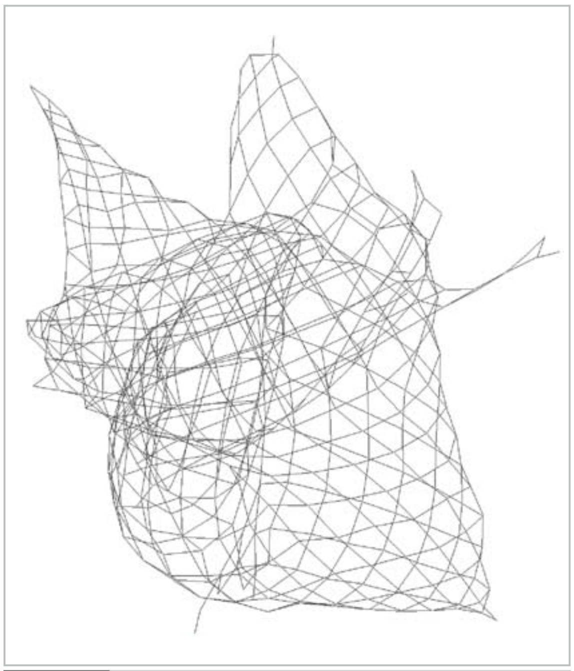
Fig. 19 NAC graph after 100 iterations

4.4 Conclusion and discussion

Whereas past NAC studies assumed solvent nodes had only passive functions, this section implements an active program at the solvent nodes to rewire surrounding edges based solely on local information. We ran this program in parallel in a simulation experiment. By doing so, we succeeded in self-organizing a regular structure for the entire graph corresponding to the pseudo-lattice structure of water.

In past NAC studies, we classified edges into four types from van del Waals to covalent, assuming that the weakest, van del Waals edge, would be rewired one after another based on the passive rewiring rule, reflecting the physical and spatial restrictions[21][24]. We also assumed solvent nodes were generally inactive and that active rewiring in the network is performed only by data flow clusters (corresponding to proteins) formed by the folding of the node chain[23]. However, the results discussed in this section indicate that the simplest and most direct method for forming an organized structure in the network is to implement an active program at each node

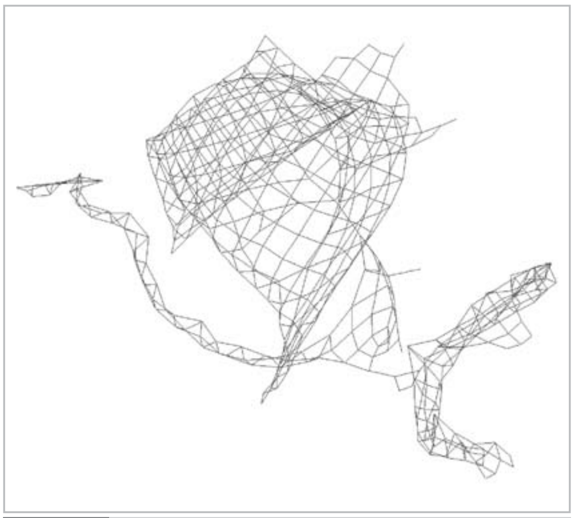
Fig.20 NAC graph after 1,000 iterations

with functions as complex as water molecules. This method of implementing a local artificial program in each processing unit to create the global function or structure is the same as that used in "amorphous computing"[1].

5 Molecular agent and program flow computing

5.1 Introduction

Extending the model in the previous section further, we introduce the framework of "revised network artificial chemistry," which expresses the molecule as an agent and the reaction by calculations of the agent program[26]. The system basically operates with the movement of the molecular agent along the edges and the firing and execution of the program at the nodes. Through these local processes, the edges are rewired and the topology of the entire network is changed. We use three designed agent programs to form and split hydrophilic clusters with a pseudo-lattice structure.

Section **5.2** below describes the experimental method; Section **5.3** presents the results; and Section **5.4** concludes this discussion of the problem.

5.2 Experimental method

5.2.1 Graph elements

The experimental program is coded in Java, with nodes, edges, and agents represented as classes. Here, a node represents a point in space, an edge represents the relationship between points in space, and an agent represents a molecule (or atomic cluster). As described below, there are several different types of edges, based on the strength of a bond. However, this strength does not represent the collision or bonding between the molecules. Rather, it collectively represents the bonding relationship between agent groups belonging to specific points in space. This section classifies nodes into hydrophilic and hydrophobic and edges into covalent bonds (covalent, **cv**, directional), hydrogen bonds (hydrogen, **hy**, directional), and van del Waals bonds (van del Waals, **wa**, non-directional) and agents into Type **0** (for splitting clusters; centrosomes), Type **1** (for **hy** rewiring), or Type **2** (for **wa** rewiring).

Figure 21 schematically shows the functions of these class instances with major variables and methods. Each instance has variables to link to other instances, in addition to basic variables such as **label** and **type**. Changing the variables by executing the methods changes the bonding relationship in the graph.

5.2.2 Execution operations

The simulation progresses by repeating the following four-step operations:

1. Node.produce_agents ( )
[Agent creation by node]:
This operation creates or removes the agents at each node to the target number specified in advance. The target number is separately determined according to the types of nodes and agents.
2. Agent.conduct_prog ( )
[Program execution by agent]:
This operation executes the agent program, rewiring edges.
3. Node.delete_edges ( )
[Edge cutting by node]:
This operation cuts the edges so that the edge degree in each node does not exceed the upper limit. The upper limit of the degree is determined by the types of nodes and edges.
4. Agent.move ( )
[Agent movement]:
This operation moves all agents from the current to the next node. The edge that an agent passes through is determined by agent variables: edge_type, DirSensitive, and template.

Among the four operations above, the three operations other than **Agent.conduct_prog** ( ) are common to all nodes and all agents. However, the algorithm that **Agent.conduct_prog** ( ) executes differs depending on the type of agent. Roughly speaking, Type **1** or **2** agents function locally to form **hy**-squares and **wa**-triangles, whereas Type **0** agents recognize differences between the **labels** of the other Type **2** agents in the same node and remove

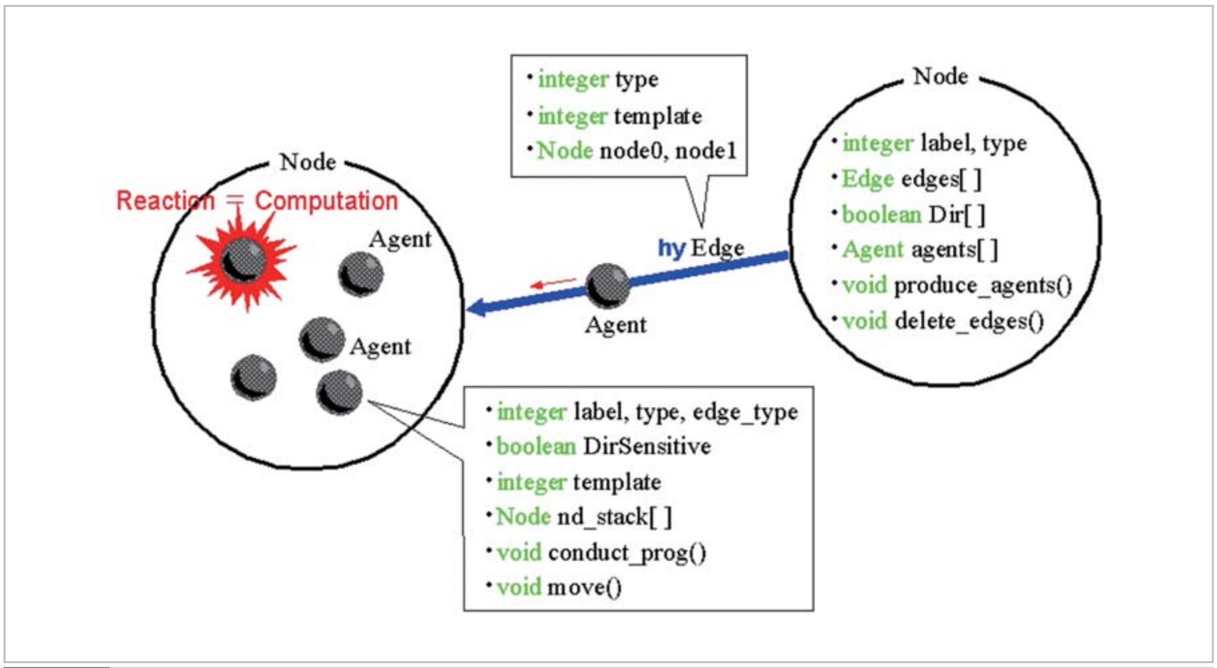

Fig.21 *Schematic of nodes, edges, agents, and their main instance variables and methods*

the agents with different **labels** and cut the edges through which they passed. For further discussion of this algorithm, refer to[26].

The target number and upper limit degree used in Steps 1 and 3 specify the type of the node (spatial point). For example, the degree of the **hy** edge and the number of Type **1** agents for a hydrophilic node are 4 independent of the **direction** or **template** and set to zero for a hydrophilic node. This specifies the hydrophobic nature of the node (the nature that it does not have **hy** edges). Figure 22 summarizes the criteria used when the agent selects the edges in Step 4.

5.3 Experimental results

5.3.1 Basic settings

The experiment starts from an initial regular random graph (with a uniform degree) containing N = 2000 nodes (1400 nodes among them are hydrophilic and 600 nodes are hydrophobic). The degrees of the **hy** and **wa** edges are set to K = 0 and K = 4, respectively, in all nodes. They do not include the number of agents. Each of the 600 hydrophobic nodes is linked to a hydrophilic node with a **cv** edge to form 600 amphipathic dipoles in a manner analogous to biological lipid molecules. Since this experiment does not use operations for cutting and creating **cv** edges, these dipoles are maintained throughout the experiment.

The four operations are repeatedly applied to the graph and the changes in graph topology observed. After 100 iterations, two **Type 0** agents with different **labels** are prepared and placed as substitutes at two randomly chosen hydrophilic nodes.

5.3.2 Formation and splitting of hydrophilic clusters with pseudo-regular structures

Figure 23 shows the typical results obtained. The initial random graph shown in Fig. 23 (a) undergoes topology changes by the Type **1** and **2** agents created and by their creation and rewiring of **hy** and **wa** edges. After 100 iterations, the graph changes to assume a shell structure. This shell structure contains hydrophilic nodes densely connected by the **hy** network and accumulated at the center and hydrophobic nodes pushed out to peripheral areas (Fig. 23 (b)). The **hy** network here has a structure close to a pseudo-two-dimensional square lattice, due to the functions of the Type **1** agent. The large average path length (L) of this structure forms the basis for the split that occurs next. (Another experiment

Edge type	Agent type		
	cv	hy	wa
type0	×	○	×
type1	×	○†	×
type2	○	○	○

Fig.22 *Edge selection scheme ○/X for Agent.move ( ) expresses whether or not each agent can pass through edges of the indicated type*

performed with N = 512 nodes, without hydrophobic nodes, using the Type **1** algorithm, produced an average path length of the resulting **hy** network reaching $L \approx 6.7$, which exceeds the value L = 6.01, for the three-dimensional cubic lattice containing the same number of nodes.)

Figure 23 (b) and figures thereafter show the growth and percolation of the Type **2** agents injected. This agent acts in a manner similar to the centrosome in a living cell and clusters (splits) the hydrophilic nodes by the **label** values they have. This split occurs between the iterations of 150 and 250. Figure 23 (c) shows the intermediate graph. The state with split clusters continues through to the final iteration, 500, until the simulation is stopped (Fig. 23 (d)).

This experiment is performed 10 times with different random number seeds. Figure 24 compares the graph after 300 iterations projected onto a plane in each case. As the figures show, the sizes of the clusters formed differ in each case, but all experiments produce the same qualitative result.

5.4 Conclusion

As a new framework for network artificial chemistry, we propose a model in which molecular agents move within the network and fire programs at nodes. The execution of these programs changes the network structure. As an example of the self-organization of the net-

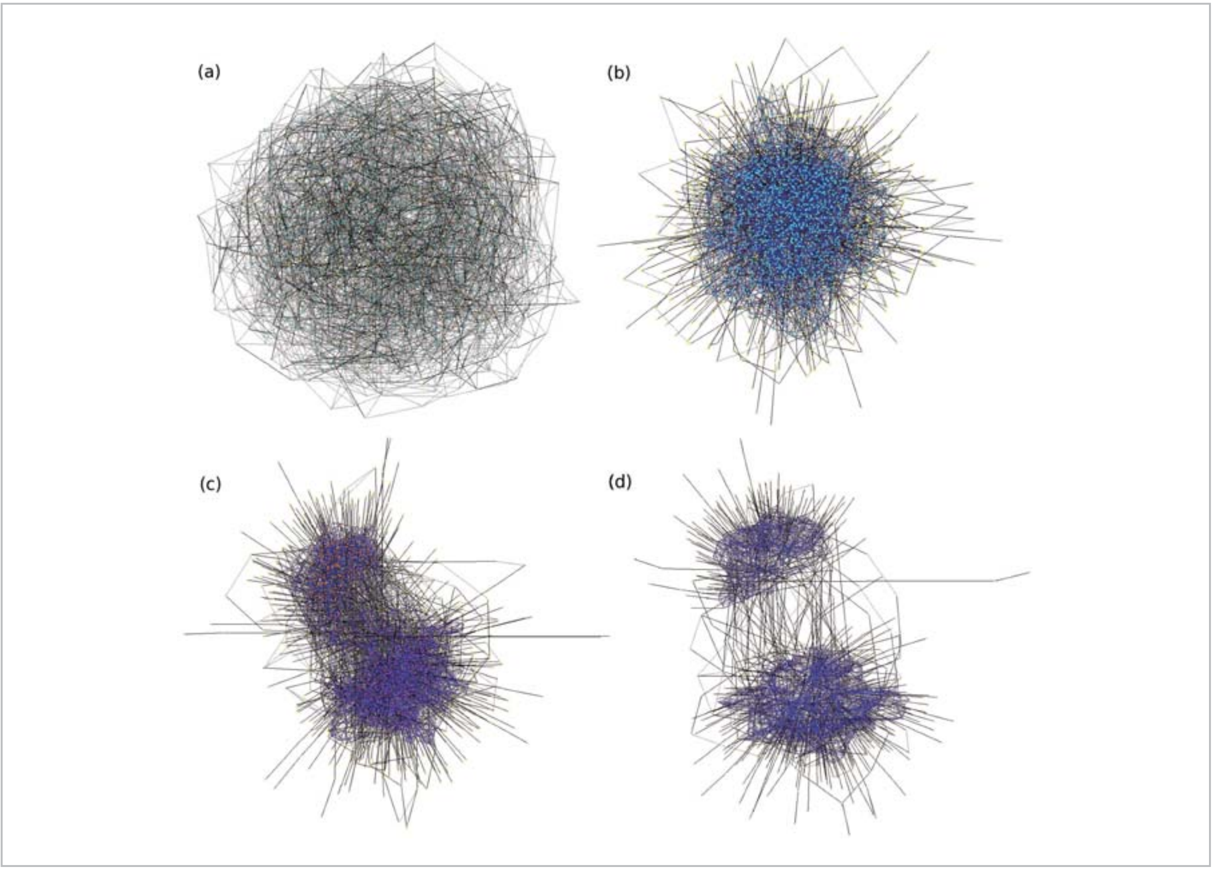

Fig.23 *Two-dimensional rendering of (a) initial regular random NAC graph and NAC graphs of (b) 100 iterations, (c) 180 iterations, and (d) 500 iterations*

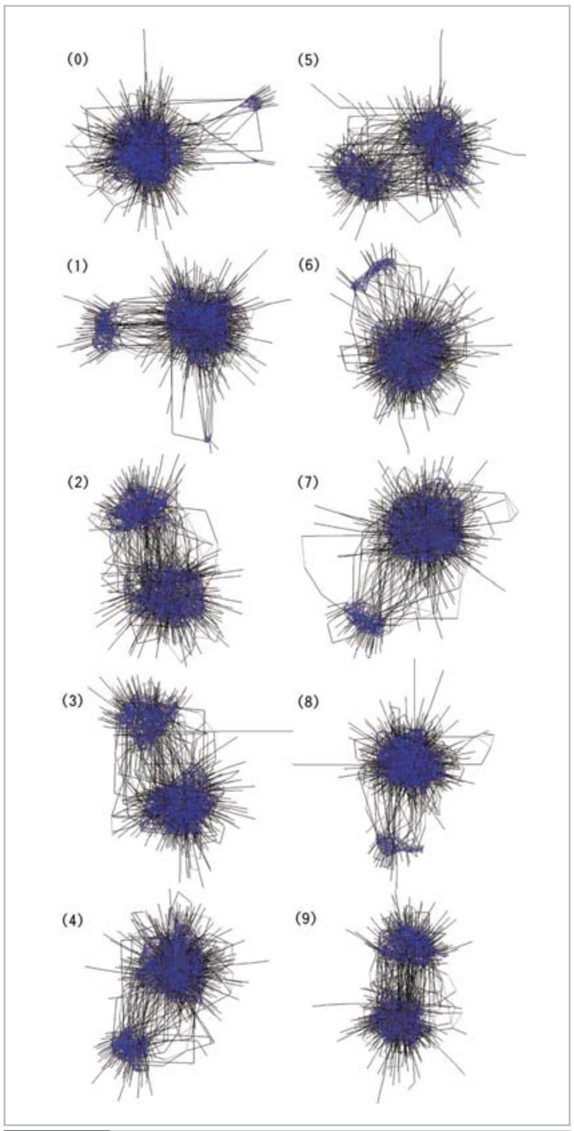

Fig.24 2-dimensional rendering of NAC graph after 300 iterations in 10 experiments conducted with different random sequences

work structure, we designed three types of agent programs, representing van del Waals edge rewiring, hydrogen edge rewiring, and cluster splitting. We verified that hydrophilic clusters with pseudo-regular structures form and split.

6 Future prospects

For the most part, this chapter presents progress with network artificial chemistry studies undertaken over the past five years. This section discusses future research problems and potential industrial applications of the new "program flow computing" computational model emerging from this research.

In short, program flow computing is a network computational model in which CPUs (nodes) connected by communication lines (edges) execute in parallel many programs that pass through them. An obvious application is finding optimal solutions to problems involving graphs. Figure 25 shows an example. The graph node coloring problem is solved by the parallel execution of many agents with simple programs. The agent program prepared here changes the color of the current node if this color is identical to the color of the previous node. Through parallel execution (e.g., the example shown in Fig. 25), a graph starting with randomly-colored nodes (Fig. 25 (a)) gradually eliminates competition between the nodes (Fig. 25 (b)) to acquire different colors on the ends of all edges (Fig. 25 (c)). This example finds the global optimal solution for graph coloring based on the local selection of simple programs. Similar methods can likely be applied to combination optimization problems for graphs.

Another example involves application to neural network information processing. The current mathematical modeling of neural networks is based on signal processing in animal brains. Information processing in or between living neurons is basically handled by molecules. In artificial neural networks, which model neurons with nodes and axons with edges, implementing molecular agents in the network and assigning various functions (programs) to these molecular agents may help in developing an integrated account of information processing for signal propagation and learning (Fig. 26).

Building and testing such models will help advance the state of research, potentially leading to the proposal of new (non-von Neumann) information processing and communication models.

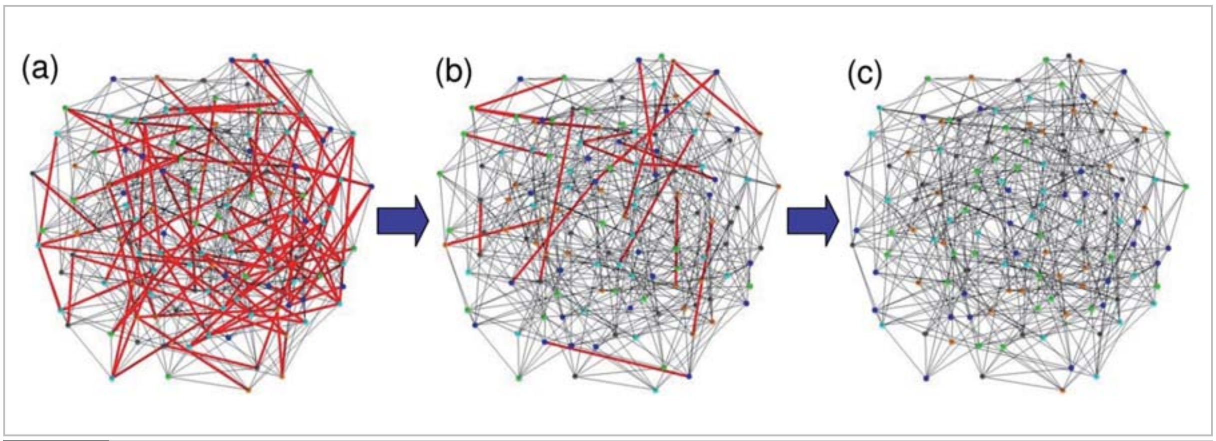

Fig.25 *Example of application of program-flow computing to node coloring problem*

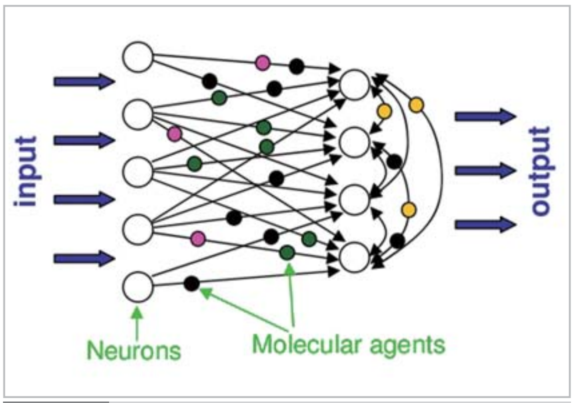

Fig.26 *Concept of application of program-flow computing to neural network information processing*

Acknowledgments

The research described in this chapter is supported in part through the Academic Frontier Research Project at Doshisha University when the author mainly belonged to both the ATR Network Informatics Laboratories (Director: Dr. Katsunori Shimohara) and the National Institute of Information and Communications Technology.

References

1 Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T. F., Nagpal, R., Rauch, E., Jay Sussman, G. J., and Weiss, R, “Amorphous computing”, Communications of the ACM 43(5), 74-82, 2000.

2 Adami, C, “Introduction to Artificial Life”, Springer-Verlag, Santa Clara, CA ,1998.

3 aiSee: Commercial software for visualizing graphs with various algorithms such as rubberband. http://www.aisee.com/

4 Barrow, G. M, “Physical chemistry”, Chapters 15-17. McGraw-Hill Education, 1988.

5 Workshop and Tutorial Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX). Bedau, M., Husbands, P., Hutton, T., Kumar, S., Suzuki, H. (eds.), 2004.

6 Dittrich, P., Ziegler, J., and Banzhaf, W, “Artificial chemistries-a review”, Artificial Life 7, 225-275, 2001.

7 Fontana, W, “Algorithmic chemistry”, In: Langton, C.G. et al. (eds.): Artificial LifeⅡ: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (Santa Fe Institute Studies in the Sciences of Complexity, Vol.10), Addison-Wesley, 159-209, 1992.

8 Hutton, T. J, “Evolvable self-replicating molecules in an artificial chemistry”, Artificial Life 8, 341-356, 2002.

9 Imai, K., Hori, T., and Morita, K, “Self-reproduction in three-dimensional reversible cellular space”, Artificial Life 8(2), 155-174, 2002.

10 Langton, C. G, “Self-reproduction in cellular automata”, Physica D 10, 135-144, 1984.

11 Ono, N., and Ikegami, T, “Artificial chemistry: computational studies on the emergence of self-reproducing units”, In: Kelemen, J., Sosik, P. (eds.): Advances in Artificial Life (6th European Conference on Artificial Life Proceedings), Springer-Verlag, Berlin, 186-195, 2001.

12 Ray, T. S, “An approach to the synthesis of life”, In: Langton, C. G. et al. (eds.): Artificial Life II: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (Santa Fe Institute Studies in the Sciences of Complexity, Vol.10), Addison-Wesley, 371-408, 1992.

13 Sayama, H, “A new structurally dissolvable self-reproducing loop evolving in a simple cellular automata space”, Artificial Life 5(4), 343-365, 2000.

14 Suzuki, H, “An approach to biological computation: unicellular core-memory creatures evolved using genetic algorithms”, Artificial Life 5(4), 367-386, 1999.

15 Suzuki, H, “Evolution of self-reproducing programs in a core propelled by parallel protein execution”, Artificial Life 6(2), 103-108, 2000.

16 Suzuki, H., Ono, N., and Yuta, K, “Several necessary conditions for the evolution of complex forms of life in an artificial environment”, Artificial Life 9(2), 537-558, 2003.

17 Suzuki, H, “Artificial chemistry on small-world networks”, In: Proceedings of the 18th Annual Conference of JSAI 2H4-03, 2004.

18 Suzuki, H, “Spacial representation for artificial chemistry based on small-world networks”, In: Pollack, J., Bedau, M., Husbands, P., Ikegami, T., Watson, R. A. (eds.): Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Artificial Life Ⅸ), 507-513, 2004.

19 Suzuki, H, “Network artificial chemistry - molecular interaction represented by a graph”, In: Bedau, M., Husbands, P., Hutton, T., Kumar, S., Suzuki, H. (eds.): Workshop and Tutorial Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife Ⅸ), 63-70, 2004.

20 Suzuki, H, “Computational folding of molecular chains in network artificial chemistry”, In: Proceedings of the 32th SICE Symposium on Intelligent Systems. The Society of Instrument and Control Engineers, 383-386, 2005.

21 Suzuki, H., and Ono, N, “Statistical mechanical rewiring in network artificial chemistry”, In: The 8th European Conference on Artificial Life (ECAL) Workshop Proceedings CD-ROM. Canterbury, UK, 2005.

22 Suzuki, H., and Ono, N, “Network rewiring rules representing molecular diffusion”, In: Proceedings of the 11th Emergent System Symposium (ESS) “Emergence Summer School 2005” Held in Toyama, Japan. The Society of Instrument and Control Engineers, 127-130, 2005.

23 Suzuki, H, “Mathematical folding of node chains in a molecular network”, BioSystems 87, 125-135, 2007.

24 Suzuki, H, “An approach toward emulating molecular interaction with a graph”, Australian Journal of Chemistry 59, 869-873, 2006.

25 Suzuki, H, “A node program that creates regular structure in a graph”, In: International Conference on Morphological Computation, Conference Proceedings. March 26-28, 2007, ECLT, Venice Italy.

26 Suzuki, H, “A network cell with molecular tokens that divides from centrosome signals”, In: Crook, N., Scheper, T. (eds.): Proceedings of the Seventh International Workshop on Information Processing in Cells and Tissues (IPCAT). Oxford Brookes University, 293-304, 2007.

27 Suzuki, H, “Structural organization in network artificial chemistry by node programs and token flow”, In: SICE Annual Conference 2007 Proceedings. The Society of Instrument and Control Engineers (SICE), Japan 1C10, 884-889, 2007.

28 Suzuki, Y., and Tanaka, H, “Chemical evolution among artificial proto-cells”, In: Bedau, M.A. et al. (eds.): Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life, MIT Press, Cambridge, 54-63, 2000.

29 Vemulapalli, G. K, “Physical chemistry”, Chapters 23-33. Prentice-Hall Inc, 1993.

30 Watts, D. J., and Strogatz, S. H, “Collective dynamics of “small-world” networks”, Nature 393, 440-442,

SUZUKI Hideaki, *Ph.D.*

Expert Researcher, Kobe Advanced ICT Research Center, Kobe Research Laboratories

Artificial Life, Artificial Chemistry, Bio-Inspired Computation