3D DNA self-assembly for maximum weighted independent set problem

Hong Wang[1*], Zicheng Wang[2], Yanfeng Wang[2], Guangzhao Cui[2]

[1]*School of Architectural and Environmental Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450002, China*

[2]*School of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450002, China*

Received 1 October 2014, www.cmnt.lv

Abstract

The problem of finding a maximum weighted independent set (MWIS) is a classical combinatorial optimization problem of graph theory, which has been proved to be NP-complete (NPC). A 2-D DNA tile self-assembly model for solving the problem has been proposed previously, but it still has many deficiencies. In this paper, we will propose a 3-D DNA tile self-assembly model based on 2-D DNA tile self-assembly model to solve the problem. This model includes two parts: the non-deterministic search system and the addition system. Firstly, we can find all the independent sets via the non-deterministic search system, and then get the total weight value of each independent set according to the addition system, and by comparison, the maximum weighted independent set will be found finally. Result shows that the operational time is linear, and the number of the tiles required in the process is constant.

Keywords: maximum weighted independent set problem, DNA computing, 3D self-assembly, DNA tile

1 Introduction

In 1994, Adleman [1] successfully solved a seven-vertex example of Hamiltonian path problem (HPP) by using the DNA molecules, which represented the birth of a new field – DNA computing. DNA computing is a new method that simulates biomolecular structure and completes the computing process with the help of molecular biology techniques, owing to its giant parallelism, high storage density, extremely low energy consumption etc that the traditional silicon computer can not match with, it has become the important research topic in the fields of computer, biology and nanotechnology.

From then on, many different DNA computing models emerged, for example, Lipton proposed a DNA computing model for solving the satisfiability (SAT) problem in 1995 [2]. Ouyang proposed a DNA computing model for solving the maximum clique problem (MCP) of graph in 1997 [3]. In 2000, Liu et al proposed a DNA computing method for the SAT problem based on the surface experiments [4]. In 2001, Wu improved the surface computing, making the operation of surface-based computing more feasible [5]. Gao et al gave a computing model for the maximum matching problem of graph based on the plasmid DNA molecules in 2002 [6] and so on.

DNA self-assembly, one important computing method of DNA computing, executes the computing process automatically according to the mutual matching of DNA molecules, and does not require manual intervention in each step. It developed in the mid-1990s through the large amounts of theoretical and experimental work by Winfree [7,8], Seeman[9], Reif [10], Rozenberg [11] et al. Due to the its following uniqueness, the constancy and specification in the interaction among DNA molecules, the reversibility of the assembly process and the predictability of the final structure of product etc, it gradually becomes a mainstream method for solving the NP and NPC problems. For example, DNA self-assembly for the minimum vertex cover problem [12], DNA 3-D self-assembly algorithmic model to solve the maximum clique problem [13], 3-D DNA self-assembly model for graph vertex coloring [14], DNA tile assembly model for 0-1 knapsack problem [15] and so on.

As the core issue of NPC class problems, MWIS problem has emerged for a long time. It has extensive use in the control of industrial process, network design, self-organizing network, design of large scale integrated circuit, analysis of economic model and many other aspects. Before the algorithm proposed in this paper, there have been some algorithms to be used for solving the maximum independent set problem, such as a genetic algorithm-based heuristic for solving the weighted maximum independent set and some other equivalent problems [16], a new hybrid genetic algorithm for maximum independent set problem [17], DNA algorithm based on plasmid for solving maximum independent set problem [18] and so on.

A 2-D DNA tile self-assembly model for solving the problem has been proposed previously, but it still has many deficiencies, for example, the large number of the tiles required in the self-assembly process etc. In this paper, we will propose a 3-D DNA tile self-assembly model for the MWIS problem based on 2-D DNA tile self-assembly model. The rest of this paper is arranged as follows: section 2 describes the basic knowledge of 3-D DNA tile self-

[*]***Corresponding author's*** e-mail: haorenlianghao@126.com

assembly model, section 3 gives the definition of MWIS problem, section 4 illustrates the 3-D DNA self-assembly for the MWIS problem, and finally the conclusion is organized in section 5.

2 3-D tile assembly model

The same as 2-D tile assembly model, the 3-D tile assembly model is also a formal model of crystal growth and to be designed to imitate molecules self-assembly such as DNA. However, in order to assemble in 3-D space, the tile's construction of 3D assembly model should be different from that of the 2-D's. The 2D tile assembly model had been fully defined by Rothemund and Winfree[19], similarly, the 3-D tile assembly model had been studied by many researchers[20,21,22,23]. Finally Lin Minqi [14] gave a formal definition of 3-D tile assembly model based on the 2-D tile assembly model, 3D's molecular model and 3D's abstract model are all shown in Figure 1.

From Figure 1, we can intuitively see that the tile in the 3-D tile assembly model is hexahedral, and its six surfaces are corresponding to the direction in the 3-D space coordinate system respectively, that is, the positive and negative direction of X, Y, Z axes. Each surface of a tile has a binding domain, and the surfaces with binding domains may bind with each other or not according to their binding domains, only when the binding domains on the adjoining sides of those tiles match and the total strength of all the binding domains exceeds the current temperature, they can connect with each other. The strength among all the matching surfaces and the current temperature are defined to be "1" and "3" respectively, that is to say, only when the matching surfaces of the tiles are "3" at least, the tiles can be assembled in the appropriate position. The type of a tile is decided by its binding domain on the six surfaces.

In this definition, the tiles are not allowed to rotate. The details about the definition of 3-D tile assembly model are in the reference [14], here we will not state again.

a) b)

FIGURE 1 3D's molecular model and 3D's abstract model: a) 3-D molecular model, b) 3-D tile abstract model

3 MWIS problem

Given an undirected weighted graph G = (V, E), V(G) is a vertex set, E(G) is the edge set, U is a subset of the vertex set V(G). If any vertex in U is not adjacent to others of U, call U the independent set of graph G, if the vertex number of one independent set is the most, it is called the maximum independent set, if the vertex weight sum of one independent set is the largest. We call it the maximum weighted independent set of graph G. In Figure 2, the vertex set {v1, v3, v4, v6} is an independent set of the graph, as well as the maximum independent set, but not the maximum weighted independent set. Vertex set {v2, v4} is also an independent set, and it is not the maximum independent set but the maximum weighted independent set. Maximum weighted independent set problem is proved to be a NP-complete problem.

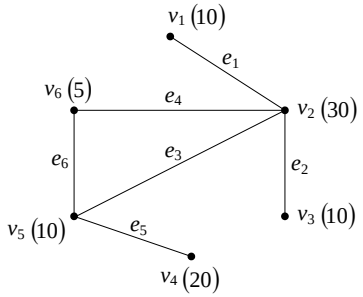

FIGURE 2 6 vertices weighted graph (weight value is in brackets)

In order to solve the problem better, here we present a formal description for the problem of finding all the independent sets first: for an undirected weighted graph G, V(G) and E(G) are the vertical sets and edge sets of graph G respectively, finding a function f: V(G)→M, which is the mapping from vertices to set M={0,1}, and meets the following condition: $\forall e_{mn} \in E(G)$, $m, n \in i$, the values of $f(m)$ and $f(n)$ are not to be "1" at the same time.

After all the independent sets are found, we begin to look for the MWIS, the details will be illustrated in the following chapters.

4 DNA self-assembly for the MWIS problem

In this paper, we found all the independent sets that satisfy the definition via the non-deterministic algorithm, then used the addition system [24] designed by Brun to get the vertex weight sum of each independent set, finally by comparison, get the optimal value.

4.1 NON-DETERMINISTIC ALGORITHM

A non-deterministic algorithm means that there are some non-deterministic choices at some steps of the algorithm (as if some oracle could tell you what to choose). To solve the MWIS problem, we introduce the non-deterministic algorithm which was described as follows:

Non-Deterministic Algorithm (G, f):
1) For each $v_i \in V(G)$ *{*
2) Assign vi : $f(i) \rightarrow \{0,1\}$
3) Check all $e_{uv} \in E(G)$ *if exist (* $u \neq v$ *) and f(u) = f(v) = 1*
4) Break and return failure
5) }
6) If all $V_i \in V(G)$ *are assigned*
7) Return success and output vi
8) Else return failure

We take Figure 2 as an example to illustrate the

nondeterministic search system for an independent set. From the Figure 2, we can get the following conclusion: edge e1 is connected by vertex v1 and v2, edge e2 is connected by vertex v2 and v3. Edge e3 is connected by vertex v2 and v5. Edge e4 is connected by vertex v2 and v6. Edge e5 is connected by vertex v4 and v5. Edge e6 is connected by vertex v5 and v6.

According to the relationship between the vertices and edges of Figure 2, the adjacency matrix A is produced:

$$A=\begin{bmatrix}1&1&0&0&0&0\\0&1&1&0&0&0\\0&1&0&0&1&0\\0&1&0&0&0&1\\0&0&0&1&1&0\\0&0&0&0&1&1\end{bmatrix}.$$

In order to see the relationship between vertices and edges more intuitively, give the corresponding adjacency Table 1.

In the matrix and adjacency Table 1, 0 represents the assumption that the vertex is not on the edge, and 1 represents the assumption that the vertex is on the edge.

TABLE 1 Adjacency table

	v1	v2	v3	v4	v5	v6
e1	1	1	0	0	0	0
e2	0	1	1	0	0	0
e3	0	1	0	0	1	0
e4	0	1	0	0	0	1
e5	0	0	0	1	1	0
e6	0	0	0	0	1	1

4.2 DESIGN OF 3D DNA TILES

Adjacency tiles: (σZ = 0,1; σX = σ–X = σY = σ–Y = σ–Z = null); include two tile types (shown in Figure 3). σZ = 0 represents the assumption that the vertex is not on the corresponding edge; σZ = 1 represents the assumption that the vertex is on the corresponding edge. The adjacent tiles can constitute different seed configurations according to different graphs. The number of the adjacency tiles is 2 for a graph with n vertices and m edges.

Passing Tiles: (σX =σ–X = 0,1; σY =σ–Y = 0,1; σZ = null; σ–Z = 0); include four tile types (show in Figure 4). The bottoms of them are all symbolized with "0"; the values on the surfaces σY and σ–Y represent the information of the vertices with the passing function; the values on the surfaces σX and σ–X represent the information of the corresponding edges, which also have the passing function. The number of the passing tiles is 4 for a graph with n vertices and m edges.

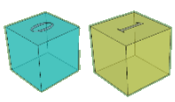

FIGURE 3 Adjacency tiles

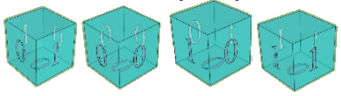

FIGURE 4 Passing tiles

Checking Tiles: (σX =σ–X = σY = σ–Y = 0; σZ = null; σ–Z = 1), (σX =σ–X = 1; σY = σ–Y = 0; σZ = null; σ–Z = 1), (σX = 1; σ–X = 0; σY = σ–Y = 1; σZ = null; σ–Z = 1); include three tile types (shown in Figure 5). The bottoms of them are all symbolized with "1"; the values on the surfaces σY and σ–Y represent the information of the vertices, which have the passing function; the values on the surfaces σX and σ–X represent the information of the corresponding edges.

(σX = σ–X = σY = σ–Y = 0; σZ = null; σ–Z = 1) denotes the vertex is not on the corresponding edge. (σX = 1; σ–X = 0; σY = σ–Y = 1; σZ = null; σ–Z = 1) denotes the vertex is on the corresponding edge. (σX =σ–X = 1; σY = σ–Y = 0; σZ = null; σ–Z = 1) denotes the vertex is not on the corresponding edge, while the vertex before it is on the corresponding edge. The number of the checking tiles is 3 for a graph with n vertices and m edges.

Input Tiles: (σX =σ–X =&; σY = 0, 1; σ–Y = σ–Z = #; σZ = null); include two tile types (shown in Figure 6). The surfaces σX and σ–X are all symbolized with "&"; the value of surface σY is "0" or "1", which represents the value of the vertices; the surfaces σ–Y and σ–Z are all symbolized with "#". The number of the input tiles is 2 for a graph with n vertices and m edges.

Output Tiles: (σX =σ–X = * ; σY = σ–Y = 0, 1; σZ = null; σ–Z = out); include two tile types(show in figure 7), which can lead to the final result. The bottoms of them are all symbolized with "out"; the surfaces σX and σ–X are all symbolized with "*"; the value of surface σY and σ–Y are "0" or "1" which represents the value of the vertices. As a result, the output tiles also have the function of passing. The number of the output tiles is 2 for a graph with n vertices and m edges

Boundary Tiles: (σ–X = &; σY = σ–Y = #;σ–Z = E; σX =σZ = null), (σ–X = 0, 1; σY = σ–Y = #; σ–Z = E; σX =σZ = null), (σ–X = * ; σ–Y = #; σZ = SS; σ–Z = E; σX =σY = null); include four tile types(show in Figure 8). The bottoms of them are all symbolized with "E"; if we count from the left, the first tile is labeled with "&" on its surface σ–X, "#" on surface σY and σ–Y; the second tile is labeled with "0" on its surface σ–X, "#" on surface σY and σ–Y; the third tile is labeled with "1" on its surface σ–X, the symbol of its other surface is same as the second tile; the fourth tile is labeled with "*" on itsσ–X, "#" on surface σ–Y, "E" on surface σ–Z, "SS" on surface σZ, which represents the completion of the self–assembly process. The number of the boundary tiles is 4 for a graph with n vertices and m edges.

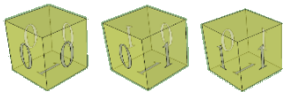

FIGURE 5 Checking tiles

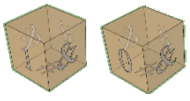

FIGURE 6 Input tiles

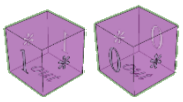

FIGURE 7 Output tiles

4.3 SEED CONFIGURATION

The seed configuration is shown in Figure 9. It is constituted by the adjacency tiles and some other tiles, in which the orange surfaces labeled with symbol "#" are the input positions. The pink surfaces labeled with symbol "out" are the output positions.

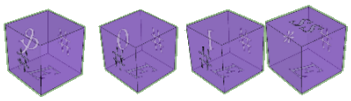

FIGURE 8 Boundary tiles

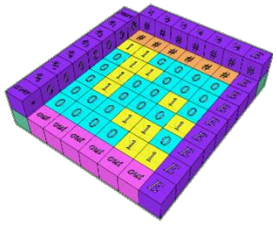

FIGURE 9 The seed configuration

The vertices and the edges of the graph G are encoded on the first row and the first column (count from left to right, top to bottom) respectively, here we encode the vertices and the edges one by one according to the order. In addition to the adjacency tiles of the seed configuration, the number of the remaining tiles required in the seed configuration is n + m + 7 for a graph with n vertices and m edges.

4.4 THE SELF-ASSEMBLY EXAMPLE

4.4.1 A successful self-assembly example

The growth of self-assembly follows the process described in the non-deterministic algorithm. Figure 10 shows the process of a successful self-assembly example, from step 1 to step 6, six input tiles (show in Figure 6) will be assembled on the input positions in turn, and the input tiles are non-deterministic. Here we take the vertex set {v1, v3, v4, v6} as the successful self-assembly example, and it can be written to be the equivalent form "101101". Owing to the length of the paper, here we will not present all the self-assembly steps, but just list the first three steps and the last step to illustrate the problem.

Step 1: in this step, just one tile will be assembled. Here we choose the input tile (σX =σ–X =&; σY = 1; σ–Y = σ–Z = #; σZ = null) as the first random input tile, which is assembled onto the first input position (count from left to right), and it represents that the vertex v1 is in the independent set that we will input.

Step 2: in this step, two tiles will be assembled. One of them is the second random input tile. Here we choose the input tile (σX =σ–X =&; σY = 0; σ–Y = σ–Z = #; σZ = null) as the second random input tile, which is assembled onto the second input position (count from left to right), and it represents that the vertex v2 is not in the vertex set that we will input. Another tile is the checking tile, which will be chosen according to complementary relationship among tiles, and the checking tile here is (σX = 1; σ–X = 0; σY = σ–Y = 1; σZ = null; σ–Z = 1).

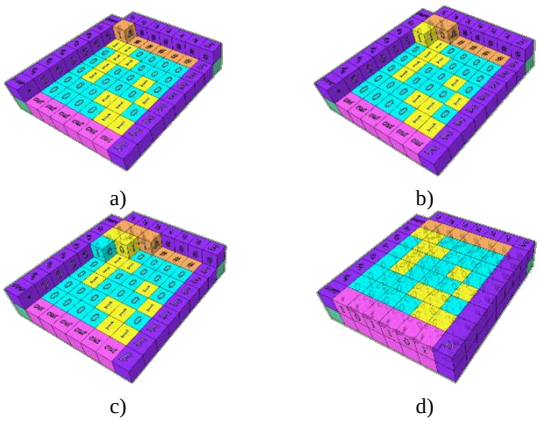

FIGURE 10 A successful self-assembly example: a) The first step of the self-assembly process, b) The second step of the self-assembly process, c) The third step of the self-assembly process, d) The last step of the self-assembly process

Step 3: in this step, three tiles will be assembled. One of them is the third random input tile, here we choose the input tile (σX =σ–X =&; σY = 1; σ–Y = σ–Z = #. σZ = null) as the third random input tile, which is assembled onto the third input position (count from left to right), and it represents that the vertex v3 is in the vertex set that we will input. The remaining tiles are checking tile and passing tile respectively, which are also to be chosen according to complementary relationship among tiles, the checking tile here is (σX =σ–X = 1; σY = σ–Y = 0; σZ = null; σ–Z = 1), the passing tile here is (σX =σ–X = 0; σY =σ–Y = 1; σZ = null; σ–Z = 0).

Step 4: this step is the last step of the self–assembly process; in this figure, we can get the following conclusion: the successful assembly of the tile with symbol "SS" on The surface σZ represents the successful completion of the whole assembly process. And here we can get the output result "101101", which denotes the vertex set {v1, v3, v4, v6} is one independent set of graph G.

4.4.2 An unsuccessful self-assembly example

The vertex sets we input randomly are not always to be an independent set of graph G, that is to say, the processes are not always successful. An unsuccessful self-assembly example is shown in the Figure 11.

In this example, we choose {v1, v3, v4, v5} as the random vertex set, its equivalent form is "101110". From Figure 11, we can see a transparent tile labeled with red cross on its σZ surface, which represents that no tile can be attached to the position, that is to say, from all the checking tiles we design, we can not find a tile to satisfy the

following conditions: (σX =σ–X = σY =σ–Y = 1. σZ = null; σ–Z = 1). So when the process carries out to this location, the assembly terminates, and we can say that the vertex set {v1, v3, v4, v5} is not an independent set of graph G.

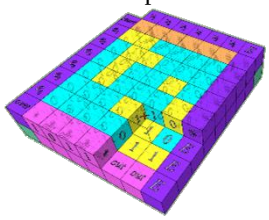

FIGURE 11 An unsuccessful self-assembly example

4.5 ADDITION SYSTEM

In this study, we refer to the addition system that Brun designs in reference [25] to solve f(a, b) = a + b (a, b are positive). The tiles that the system requires are shown in Figure 12, four sides of each tile represent different sticky ends, the south and east sides are input values, the west side is the carry value, the north side is the output value.

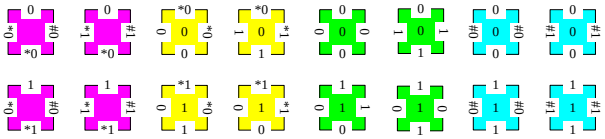

FIGURE 12 Addition operation tiles

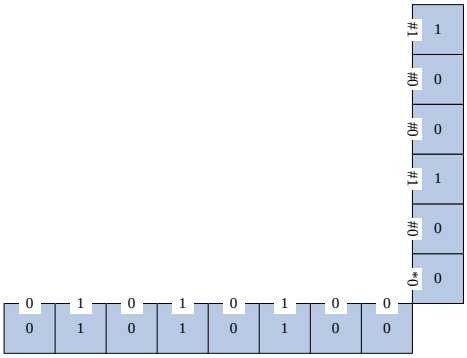

FIGURE 13 Seed configuration of addition system

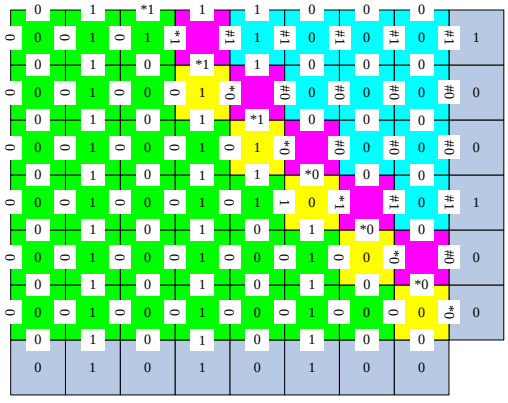

FIGURE 14 An example of addition operation

Now we illustrate the system according to an example in Figures 13 and 14, we take a = 10101002 = 84, b = 1001002 = 36, by computing, we obtain f(a, b) = 1111002 = 120.

4.6 COMPLEXITY ANALYSIS

Like 2-D DNA self-assembly model, the complexity of 3-D DNA self-assembly model also mainly includes three aspects: computation time T, computation space S, and the total tile's number N required in the assembly process. Since the complexity of the addition system has been discussed in reference [24], here we will just discuss the complexity of the independent set search system for a graph with n vertices and m edges.

From the Figure 10, we can obviously see that the computation time T is equal to the width (depth) of the assembly, in fact:

$$T \le n+ m + 2 = \Theta(n). \quad (1)$$

The computation space S is the volume of the assembly map:

$$S \le (m + 3) \times (n + 2) = \Theta(mn). \quad (2)$$

The total number of the tiles required during the assembly is:

$$N = 2 + 4 + 3 + 2 + 2 + 4 + n + m + 7 = n + m + 24. \quad (3)$$

5 Conclusions

In this paper, we design a 3-D DNA self-assembly model to solve the MWIS problem of graph based on 2D DNA self-assembly model, and confirm that the assembly process is feasible in theory. Compared with the 2-D DNA self-assembly model, the computing ability of 3-D DNA self-assembly model is more powerful. For example, the 3D DNA self-assembly model ensures the number of the tiles required in the process is constant, which is far smaller than the 2D DNA self-assembly model, the 3-D DNA tiles can express more information etc. However, we may encounter some technical obstacles in the actual operation, such as restrictions on the concentration of DNA chains, AFM, the restrictions of electrophoresis and so on. So in the future we will work harder to overcome the difficulty.

Acknowledgements

The work for this paper was supported by the National Science Foundation of China (Grant No.60773122, 60970084, 61070238), Basic and Frontier Technology Research Program of Henan Province (Grant No. 082300413203, 092300410166) and Innovation Scientists and Technicians Troop Construction Projects of Henan Province (Grant No. 094100510022).

References

[1] Adleman L M 1994 Molecular Computation of Solutions to Combinatorial Problems *Science* **266**(11) 1021-4
[2] Lipton R J 1995 *DNA* Solution of Hard Computation Problems *Science* **268**(4) 542-5
[3] Ouyang Q, Kaplan P D, Liu S, Libchaber A 1997 *DNA* Solution of the Maximal Clique Problem Science **278**(17) 446-9
[4] Liu Q, Wang L, Frutos A, Condon A E, Corn R M, Smith L M 2000 DNA Computing on Surfaces *Nature* **403** 175-9
[5] Wu H Y 2001 An Improved Surface Based Method for DNA Computation *Biosystems* **59**(1) 1-5
[6] Gao L, Xu J 2002 DNA Solution of Vertex Cover Problem Based on Sticker Model *Chinese Journal of Electronics* **11**(2) 280-4 *(in Chinese)*
[7] Winfree E, Eng T, Rozenberg G 2001 String tile models for DNA computing by self-assembly *Proceeding of the DNA'00 Revised Papers from the 6th International Workshop on DNA-Based Computers: DNA Computing* **2054** 63-88
[8] Winfree E 1998 Algorithmic self-assembly of DNA *PhD thesis* California Institute of Technology Pasadena CA
[9] Seeman N C 1998 DNA nanotechnology: novel DNA constructions *Annual Review of Biophysics and Biomolecular Structure* **27** 225–48.
[10] Reif J F 2002 Computing: successes and challenges *Science* **296** 478-9
[11] Rozenberg G, Spaink H 2003 DNA computing by blocking *Theoretical Computer Science* **292** 653-65
[12] Wang Y F, Hu P P, Zhang X C, Cui G Z 2011 DNA Self-Assembly for the Minimum Vertex Cover Problem *Advanced Science Letters* **4**(1) 74-9
[13] Ma J J, Li J, Dong Y F 2011 DNA 3D Self-assembly Algorithmic Model to Solve Maximum Clique Problem *Image, Graphics and Signal Processing* **3** 41-8
[14] Lin M Q, Xu J, Zhang D F, Chen Z, Zhang X, Cheng Z, Huang Y, Li Y 2009 3D DNA Self-Assembly Model for Graph Vertex Coloring *Computational and Theoretical Nanoscience* **7**(1) 246-53
[15] Wang Y F, Lu W L, Zhang X C, Cui G Z 2010 DNA Tile Assembly Model for 0-1 Knapsack Problem *Proceedings 2010 Fifth International Conference on Bio-Inspired Computing: Theories and Applications* 180-4
[16] Hifi M 1997 A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems *Journal of the Operation Research Society* **48**(6) 612-22
[17] Mehrabi S, Mehrabi A, Mehrabi A D 2009 A new hybrid genetic algorithm for maximum independent set problem *Conference: ICSOFT 2009 – Proceedings of the 4th International Conference on Software and Data Technologies* **2**
[18] Head T, Rosenberg G, Blagergroen R S, Breek C K, Lommerse P H, Spaink H P 2000 Computing with DNA by operating on plasmids *Biosystems* **57**(2) 87-93
[19] Rothemund P, Winfree E 2000 The program-size complexity of self-assembled squares(extended abstract) *Proceedings of the thirty-second annual ACM symposium on Theory of computing* 459-68
[20] Brun Y 2008 Self-Assembly for Discreet, Fault-Tolerant, and Scalable Computing on Internet-Sized Distributed Networks *[dissertation] California* University of Southern California 7-111
[21] Reif J H 1999 Local parallel biomolecular computation *DNA Based Computers III: DIMACS Workshop* Providence 217-54
[22] Pelletier O, Weimerskirch A 2002 Algorithmic Self-Assembly of DNA Tiles and its Application to Cryptanalysis *Proceedings of the GECC0-2002* 139-46
[23] Vieira F R J, Barbosa V C 2011 Optimization of supply diversity for the self-assembly of simple objects in two and three dimensions *Natural Computing* **10**(1) 551-81
[24] Brun Y 2007 Arithmetic computation in the tile assembly model: Addition and multiplication *Theoretical Computer Science* **378** 17-31

Authors

Hong Wang, August 1977, Henan, Pingdingshan, China.

Current position, grades: lecturer.
Scientific interests: computer control, non-linear theory and the biological information processing.
Publications: 10 papers, 2 patents.

Zicheng Wang, August 1976, Henan, Pingdingshan, China.

Current position, grades: associate professor.
Scientific interests: biological information processing.
Publications: 15 papers, 2patents.

Yanfeng Wang, March 1973, Henan, Nanzhao, China.

Current position, grades: professor.
Scientific interests: biological information processing.
Publications: 20 papers, 3 patents.

Guangzhao Cui, Septmeber 1957, Henan, Luoning, China.

Current position, grades: professor.
Scientific interests: biological information processing.
Publications: 30 papers, 5 patents.