Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080143

Parallel Solution to the Dominating Set Problem by Tile Assembly System

Hongjiang Zheng[1,2], *Yufang Huang* [3,4,*], *Jianhua Xiao*[5], *Jian Liu*[2] *and Tao Song* [4]

[1] School of Information Engineering, Tarim University, Akesu 843300, P.R. China
[2] School of Information Engineering, Wuhan University of Technology, Wuhan 430070, P.R. China
[3] College of Mathematics, Southwest Jiaotong University, Chengdu 610031, P. R. China
[4] Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, P. R. China
[5] Research Center of Logistics, Nankai University, Tianjin 300071, P. R. China

Received: 28 Jun. 2013, Revised: 6 Nov. 2013, Accepted: 7 Nov. 2013
Published online: 1 Jan. 2014

Abstract: The dominating set problem is a well known NP hard problem. It means that as the instance size grows, they quickly become impossible to solve on traditional digital computers. Tile assembly model has been demonstrated as a highly distributed parallel model of computation. Algorithmic tile assembly has been proved to be Turing-universal. This paper proposes a tile assembly system for the dominating set problem. It only needs $\Theta(mn)$ tile types to solve such a complex problem in the time $\Theta(m+n)$ where n and m are the number of vertices and edges of the given graph, respectively.

Keywords: Parallel computing, Tile assembly model, Dominating set problem, NP hard problem

1 Introduction

Biological systems are a rich source for computing devices design. A lot of classical computing devices are inspired by biological systems such as automaton and Turing machine. In recent years, computing devices inspired by cells (or molecules inside cells such as DNA) are deeply investigated. Most of such computing systems inspired by cells, tissues and neural networks are theoretically proved to be universal [12,13,14] and computationally efficient [15,16,20,21,22,23,24,25,26,27,28,29,30]. This work focuses on computing systems based DNA tile assembly which is a prospective method to overcome the ultimate limits of silicon-based technology.

The dominating set problem is a well known NP hard problem. It means that as the instance size grows, they quickly become impossible to solve on traditional digital computers. The main goal of this paper is to utilize the computational power of tile assembly to solve the dominating set problem, while minimizing the dependency of the process on particular problem instance.

Parallel computation by tile assembly is largely different from the previous DNA computing [1], and even is regarded as overcoming the previous DNA computing in some sense. Winfree [17] extended Wang tilings [18] to the tile assembly model with a view to model self-assembly process of DNA motifs. Furthermore, the computation by self-assembly has been proved to be Turing-universal.

Therefore, DNA computation based on self-assembly becomes a significant role in bio-molecular computing. Since Winfree [19] first constructed the simple two-dimensional tiles using DNA strands to demonstrate the feasibility of computing through the self-assembling of tiles, Mao [10] came up with more complicated tiles (TX) to experimentally execute four steps of logical (cumulative XOR) operations. Barish et al. [2] proposed and experimentally demonstrated an algorithmic self-assembly to perform two primitive computations: copying and counting. Fujibayashi et al. [8] used tiles and DNA origami to grow crystals containing a cellular automaton pattern and proved that programmable molecular self-assembly may be sufficient to create a wide range of complex objects in one-pot reactions. Dwyer [6] proposed two architectures that are enable by self-assembly for implementations of molecular

* Corresponding author e-mail: huangyufang@home.swjtu.edu.cn

computers solving highly demanding computational problems.

In addition, a large number of researches have emerged into this field and applied this biological technique to implementing more intricate computation. Lagoudakis [11] presented an algorithmic design for solving the SAT problem by encoding of the algorithm in a general way separated the algorithm from the data. Jonoska [9] developed an algebraic representation of the self-assembly process and used it to prove that the model of self-assembly precisely captures NP-computability under certain conditions. Furthermore, the tile assembly model has been designed to do algebraic computation. Huang [7] gave an algorithmic factoring tile system on the basis of Euclids algorithm. Barua [3] demonstrated how the tile assembly process was used for computing the finite field multiplication and addition. In addition, Brun [4,5] developed how to make use of the tile assembly to compute the sums and products of two numbers and to factor non-deterministically an integer into the product of two integers.

The rest of the paper is organized as follows: after introducing basic concepts used throughout the paper in Section 2, we will show the detail method of how to design the tile assembly system to solve the dominating set problem and discuss its computational complexity in Section 3. Section 4 will summarize the contributions of this work.

2 Preliminary

Formally, the standard tile assembly system introduced and validated as a universal computational framework will be our abstract model. Informally, a tile is modeled as a square with glues on each side. Two adjacent tiles will attach to each other if they have compatible glues. We recall some definitions of formal tile system as it was originally introduced in [4].

A tile is a four-tuple $< \sigma_E, \sigma_S, \sigma_W, \sigma_N >$ over a set of binding domains Σ such that: (1) Σ is a finite alphabet of binding domains and $null \in \Sigma$; (2) $D = \{E,S,W,N\}$ denotes a set of direction functions from positions to positions, i.e., $\mathbb{Z}^2$ to $\mathbb{Z}^2$ such that for all positions (x,y), $E(x,y)=(x+1,y)$, $S(x,y)=(x,y-1)$, $W(x,y)=(x-1,y)$ and $N(x,y)=(x,y+1)$; (3) σ_X represents the binding domain in the direction X.

If A is a configuration, then within system S, a tile t can attach to A at position (x,y) and produce a new configuration A' iff: (1) $(x,y) \in A$; (2) $\sigma_{d \in D} g(bd_d(t), bd_{d^{-1}}(A(d(x,y)))) \geqslant \tau$; (3) $\forall (u,v) \in Z^2$, $(u,v) \neq (x,y) \Rightarrow A'(u,v) = A(u,v)$; (4) $A'(x,y) = t$, where $E^{-1} = W$, $S^{-1} = N$, $W^{-1} = E$, $N^{-1} = S$. That is, a tile can attach a configuration only in empty positions and only if the total strength of the appropriate binding domains on the tiles in neighboring positions meets or exceeds the temperature τ.

3 Solving the Dominating Set Problem based on Tile Assembly System

3.1 Definition of the Dominating Set Problem

Given a graph $G = (V,E)$ where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $E = \{e_1, e_2, ..., e_m\}$ is the set of edges. In graph theory, a dominating set of a graph $G = (V,E)$, is a subset $V_1 \in V$ such that for all $u \in V - V_1$ there is a $v \in V_1$ with $(u,v) \in E$. The size of a dominating set is the number of vertices it contains. The dominating set problem is to find a minimum size dominating set in G. The dominating set problem has been proved to be a NP-complete problem.

3.2 Construction of Algorithmic Tile Assembly System

As was mentioned above, our design attempts to simulate a non-deterministic algorithm for finding the dominating set V_1. The algorithm is given below:

Non-deterministic Dominating set $(v_1, v_2, ..., v_n; e_1, e_2, ..., e_m)$
(1) Set all edges to be immarked
(2) for $(i = 1, 2, ..., n)$ do
(3) Assign a value (0 or 1) to the vertex v_i
(4) Mark all edges $e_j (j = 1, 2, ..., m)$ which include the vertex $v_i (v_i = 1)$
(5) if all edges are marked
(6) then return Yes and output $v_1, v_2, ..., v_n$ where $v_i = 0$ or 1
(7) else return Failure

To perform the algorithm above for finding the dominating set, we constructed five kinds of tile subsets which are T_1, T_2, T_3, T_4 and T_5. The tile set T_1 shown in Fig. 1 over the binding domain set $\Sigma_1 = \{i, =, \#, 0, 1\}$ $(i = 1, 2, ..., n)$ aims to assign a value 0 or 1 to the vertex $v_i (i = 1, 2, ..., n)$. If $v_i = 1$, then $v_i \in V_1$; otherwise $v_i \notin V_1$.

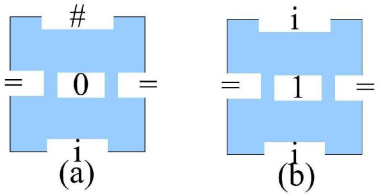


Fig. 1: Tile set T_1 over the symbol set Σ_1 for stochastically selecting vertex from the vertex set $V = \{v_1, v_2, ..., v_n\}$, where $\Sigma_1 = \{i, =, \#, 0, 1\} (i = 1, 2, ..., n)$.

After stochastically guessing a subset $V_1 \subseteq V$, the next operation is to decide whether the set V_1 can satisfy the conditions of the dominating set. Our design is to make V_1 be compared with each edge of E. That is to say,

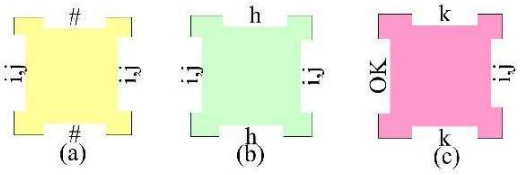


Fig. 2: Tile set T_2 over the symbol set Σ_2 to decide whether the vertex chosen at random can dominate the edges, where $\Sigma_2 = \{(i,j),\#,h,k,OK\}(i,j=1,2,...,n)$.

for each edge $e_ij \in E$, it needs to compare e_{ij} with each vertex $v_k \in V_1$. If for all edges $e_{ij} \in E$, there always exist $v_i \in V_1$ or $v_j \in V_1$, then the subset V_1 is accepted as a dominating set. The tile subset T_2 shown in Fig. 2 over the binding domain set $\Sigma_2 = \{(i,j),\#,h,k,OK\}$ $(i,j=1,2,...,n)$ focuses on performing the functions above. The binding domain σ_E represents the edge e_{ij}, and $\sigma_S = h$ $(h=1,2,...,n)$ denotes the vertex v_h. If $(\sigma_E,\sigma_S) = ((i,j),h)$ and $h \notin \{i,j\}$, then it makes $(\sigma_W,\sigma_N) = (\sigma_E,\sigma_S)$ in Fig. 2(b). Similarly, the tile in Fig.2(a) performs the same function. If $(\sigma_E,\sigma_S) = ((i,j),k)$ and $k \in \{i,j\}$, then it makes $(\sigma_W,\sigma_N) = (OK,\sigma_S)$ in Fig. 2(c).

The tile set T_3 illustrated in Fig. 3 is designed to transmit the data in the process of computation. T_3 is over the binding domain set $\Sigma_3 = \{\#,OK,i,=,*\}$ $(i=1,2,...,n)$ where σ_E, σ_S denote the input and σ_W, σ_N denote the output binding sides. Then we use the tile subset T_4 in Fig. 4 over the binding domain set $\Sigma_4 = \{S,i,\#,0,i1\}(i=1,2,...,n)$ to extract the subset V_1. It decides whether $v_i \in V_1$ according to the input σ_S. If $\sigma_S = i$, then the vertex $v_i \in V_1$ and outputs the value $i1$ in the centre of the tile. Otherwise, it outputs the value 0 in the centre of the tile. Finally, it needs to identify which final tile assembly configuration represents the dominating set. The tile set T_5 shown in Fig. 5 over the binding domain set $\Sigma_5 = \{S,\#,Yes\}$ aims to mark the one that denotes the dominating set and output Yes in the centre of the tile.

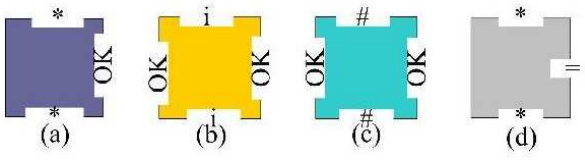


Fig. 3: Tile set T_3 over the symbol set Σ_3 for implementation of data transmission, where $\Sigma_3 = \{\#,OK,i,=,*\}(i=1,2,...,n)$.

As discussed above, we construct the tile assembly system including five tile subsets. They are proved to find the dominating set in Theorem 1 as follows.

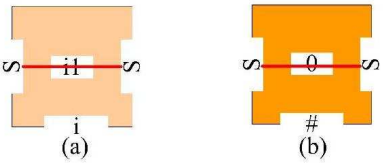


Fig. 4: Tile set T_4 over the symbol set Σ_4 for distracting the final vertex sub-set, where $\Sigma_4 = \{S,i,\#,0,i1\}(i=1,2,...,n)$.

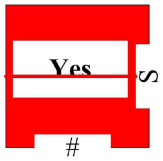


Fig. 5: Tile set T_5 over the symbol set Σ_5 to identify the dominating set, where $\Sigma_5 = \{S,\#,Yes\}$.

Theorem 1. Given the graph $G=(V,E)$. Let $\Sigma_D = \cup_{i=1}^5 \Sigma_i$, $g_D = 1, \tau_D = 2$, and $T_D = \cup_{i=1}^5 T_i$ be a set of tiles over Σ_D with T_i and Σ_i shown in Fig.i $(i=1,2,...,5)$. Then the tile system $S_D = (T_D, g_D, \tau_D)$ can find all the dominating sets of the graph G.

Proof. For the input graph, let $V = \{v_1,v_2,...,v_n\}$ and $E = \{e_1,e_2,...,e_m\}$. For simplicity of discussion, let $e_i = e^i_{j_i k_i}$ such that $e_i = (v_{j_i}, v_{k_i})$, where $v_{j_i}, v_{k_i} \in V$.

Let the tile set Γ_D be constructed as follows,

$$\begin{aligned}
\Gamma_D = \{&\alpha_0 =< null, null, \%, 11 >,\\
&\alpha_i =< \%, null, \%, i > (i=1,2,...,n),\\
&\alpha_{n+1} =< \%, null, null, * >,\\
&\beta_0 =< null, 11, =, 11 >,\\
&\beta_i =< null, 11, (j_i,k_i), 11 > (i=1,2,...,m),\\
&\beta_{m+1} =< null, 11, S, null > \}.
\end{aligned}$$

Then the seed configuration $S_D : \mathbb{Z}\times\mathbb{Z} \to \Gamma_D$ is such that
(1) $S(-i,0) = \alpha_i (i=0,1,2,...,n+1)$,
(2) $S(0,i+1) = \beta_i (i=0,1,2,...,m+1)$,
(3) for all other positions $(x,y) \in \mathbb{Z}^2$, $S(x,y) = empty$.

According to the rules above, the seed configuration can be constructed so that V and E are denoted in the seed configuration.

As for the tile $t_{T_1} \in T_1$, the following conditions are satisfied:
(1)$bd_W(F(0,1)) = bd_E(t_{T_1})$
(2)$bd_N(F(-1,0)) = bd_S(t_{T_1})$.
Then some tile $t_{T_1} \in T_1$ will non-deterministically attach to the seed configuration on the position (-1, 1), i.e. $F(-1,1) = t_{T_1}$. Similarly, some tiles $t'_{T_1} \in T_1$ can attach to the positions $(-i,1)(i=2,3,...,n)$. These tiles growth aims to produce a subset $V_1 \subseteq V$ at random.

Therefore, from the second row to the $(m+1)-th$ row, the tiles $t \in T_i (i = 2,3)$ will decide whether the subset $V_1 \subseteq V$ can include one vertex of the edge $e^i_{j_i k_i} (i = 1,2,...,m)$.

In the $(m+2)-th$ row, the system will extract the subset V_1. Moreover, the identifier tile T_5 will attaches to the position $(-(n+1), m+2)$ to identify the dominating sets. Then, the tile system $S_D = (T_D, g_D, \tau_D)$ can find all the dominating sets of the graph G. □.

It is noticed that the tile assembly system $S_D = (T_D, g_D, \tau_D)$ can find all the dominating sets. Furthermore, in order to find the minimum dominating set, it needs to combine the tile assembly system with the gel electrophoresis technique. As designed in Fig.4, we make a red single DNA strand denote the existence of the vertex v_i so that $v_i \in V_1$ which has shorter oligonucleotide than that representing the absence of a vertex. After gel electrophoresis, we can see that the strands running fastest are our desired answers to the minimum dominating set problem.

The complexity of the design is considered in terms of computation time and the number of distinct tiles required. It is obvious from the design of the tile sets shown in Fig.$i (i = 1,2,...,5)$ that the computation time $T(S_D)$ is equal to the depth of the final assembly configuration. In fact, it is induced from the Theorem 1 that

$$T(S_D) = (n+1) + (m+2) - 1 = m + n + 2 = \Theta(m+n)$$

where n and m are the number of vertices and edges of the given graph respectively. And the number of tile types $N(S_D)$ can be concluded according to the Fig.$i (i = 1,2,...,5)$ that

$$N(S_D) = 2n + (m + m(n-2) + 2m) + (3+n) + (n+1) + 1 = 4n + mn + m + 5 = \Theta(mn).$$

4 Conclusions

In this paper, we proposed an algorithmic tile assembly system to solve the dominating set problem. The model presented here is based on the automatic growth of tile assembly units and therefore, inherit several advantages from it. The dominating set problem is a NP-complete problem and it takes exponential time to solve it in a traditional digital computer, while in tile assembly based supercomputing, it only needs $\Theta(mn)$ tile types to solve such a complex problem in the time $\Theta(m+n)$ where n and m are the number of vertices and edges of the given graph respectively.

A great deal of experimental researches has demonstrated the feasibility of construction for large shapes based on tile assembly. We have reasons to believe that the algorithmic establishment for solving the dominating problem by applying tile assembly system as developed here will provide some helps to solve other NP hard problems.

Acknowledgement

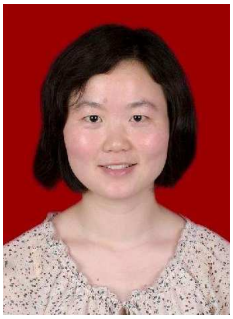
The work was supported by National Natural Science Foundation of China (Grant Nos. 51268051, 61170016, 61202204, 61074169, 60903105, 61033003 and 91130034), Postdoctoral Science Foundation of China (2012M521427), Foundation of Southwest Jiaotong University (SWJTU11ZT29, SWJTU11CX157), Innovation Fund of Huazhong University of Science and Technology (2011TS005) and the National High Technology Research and Development Program of China under Grant 2013AA122403.

References

[1] L. M. Adleman, Molecular Computation of Solutions to Combinatorial Problems, Science, **266**, 1021-1024 (1994).
[2] R. Barish, P. Rothemund and E. Winfree, Two Computational Primitives for Algorithmic Self-assembly: Copying and Counting, Nano Lett., **5**, 2586-2592 (2005).
[3] R. Barua and S. Das, Finite Field Algorithmetic Using Self-assembly of DNA Tilings. IEEE, 2529-2536 (2003).
[4] Y. Brun, Arithmetic Computation in the Tile Assembly Model: Addition and Multiplication. Theor. Comput. Sci., **378**, 17-31 (2007).
[5] Y. Brun, Nondeterministic Polynomial Time Factoring in the Tile Assembly Model, Theor. Comput. Sci., **395**, 3-23, (2008).
[6] C. Dwyer, J. Poulton, R. Taylor and L. Vicci, DNA Self-assembled Parallel Computer Architectures, Nanotechnology, **15**, 1688-1694 (2004).
[7] Y. Huang, J. Xu and Z. Cheng, Integer Factorization Based on the Tile Assembly Model, Journal of Computational and Theoretical Nanoscience, **8**, 1-12 (2011).
[8] K. Fujibayashi, R. Hariadi, S. Park, E. Winfree and S. Murata, Toward Reliable Algorithmic Self-assembly DNA Tiles: A Fixed-width Cellular Automaton Pattern, Nano Lett., **8**, 1791-1797 (2007).
[9] N. Jonoska and G. McColm, A Computational Model for Self-assembling Flexible Tiles, Berlin Heidelberg, Germany: Springer-Verlag, 142-156 (2005).
[10] C. Mao, T. Labean, J. Reif and N. Seeman, Logical Computation Using Algorithmic Self-assembly of DNA Triple-Crossover Molecules, Nature, **407**, 493-496 (2000).
[11] M. Lagoudakis and T. LaBean, 2D DNA Self-assembly for Satisfiability, DIMACS Series in DISCRETE Mathematics and Theoretical Computer Science, 139-152 (2000).
[12] L. Pan, and G. Paun, Spiking Neural P Systems: An Improved Normal Form, Theor. Comput. Sci., **411**, 906-918 (2010).
[13] L. Pan, X. Zeng and Y. Zhang, Time-free Spiking Neural P Systems, Neural Computation, **23**, 1-23 (2011).
[14] L. Pan, and X. Zeng, Small Universal Spiking Neural P Systems Working in Exhaustive Mode, IEEE Trans. on Nanobioscience, **10**, 99-105 (2011).
[15] L. Pan and M. J. Prez-Jimnez, Computational Complexity of Tissue-like P Systems, J. Complex., **26**, 296-315 (2010).
[16] L. Pan, D. Diaz-Pernil, and M. J. Prez-Jimnez, Computation of Ramsey Numbers by P Systems with Active Membranes, Int. J. Found. Comput. Sci., **22**, 29-38 (2011).

[17] E. Winfree, Algorithmic Self-assembly of DNA, Ph.D thesis, California Institute of Technology, Aug. (1998).
[18] H. Wang, Dominoes and the AEA Case of the Decision Problem, In Proc. Symp. Math. Theory of Automata, New York, Polytechnic Press, 23-55 (1963).
[19] E. Winfree, F. Liu, L. Wenzler and N. Seeman, Design and Self-Assembly of Two-dimensional DNA Crystals. Nature, **394**, 539-544 (1998).
[20] X. Zhang, J. Wang and L. Pan, A note on the generative power of axonSystems, Int. J. Comput. Commun., **4**, 92-98 (2009).
[21] L. Pan, G. Paun, Spiking neural P systems: an improved normal form. Theor. Comput. Sci., **411** (2010).
[22] L. Pan, X. Zeng, Small universal spiking neural P systems working in exhaustive mode. IEEE Trans. on Nanobioscience, **10**, (2011).
[23] L. Pan, X. Zeng, X. Zhang, Time-free spiking neural P systems. Neural Comput., **23**, (2011).
[24] L. Pan, X. Zeng, X. Zhang, Y. Jiang, Spiking neural P systems with weighted synapses. Neural Process. Lett., **35**, (2012).
[25] X. Zeng, X. Zhang, L. Pan, Homogeneous spiking neural P systems. Fund. Inform., **97**, (2009).
[26] X. Zhang, X. Zeng, L. Pan, On string languages generated by asynchronous spiking neural P systems. Theor. Comput. Sci., **410**, (2009).
[27] X. Zhang, B. Luo, X. Fang, L. Pan, Sequential spiking neural P systems with exhaustive use of rules, BioSystems, **108**, (2012).
[28] X. Zhang, Y. Jiang, L. Pan, Small universal spiking neural P systems with exhaustive use of rules. Int. J. Comput. Commun., **7**, 5 (2010).
[29] J. Wang, H.J. Hoogeboom, L. Pan, G. Paun, M. J. Prez-Jimnez, Spiking neural systems with weights. Neural Comput., **22**, 10 (2010).
[30] L. Pan, G. Paun, M. J. Prez-Jimnez, Spiking neural P systems with neuron division and budding. Sci. China Inform. Sci., **54**, 8 (2011).

Hongjiang Zheng was born in Anhui, China, in 1980. He received the B.S. and M.S. degrees from Wuhan University of Technology, China, in 2004 and 2008, respectively. He is currently working towards the Ph.D. degree in communication and information system at the same university. He is a lecturer with Tarim University, Akesu. His current interests include cooperative communications, wireless networks and computational theory.



Yufang Huang received the Ph.D. degree from Huazhong University of Science and Technology in 2010. She is now on the Post Doctor research of the Department of Control Science and Engineering at Huazhong University of Science and Technology from 2011. Currently, she is a lecturer of the College of Mathematics at Southwest Jiaotong University. Her research interests include combinatorial optimization and molecular computation.

Jianhua Xiao is received the Ph. D. degree in System Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2008. He is currently a lecture at Nankai University, Tianjin, China. His research interests include combinatorial optimization, Bio-inspired computation and logistics optimization etc.

Jian Liu received the M.S. degree in communication and information system from Wuhan University of Technology, China, in 2010. He is currently working towards the Ph.D. degree in communication and information system at the same university. His research interests include satellite navigation and computer networks.

Tao Song received his Ph.D in Systems Analysis and Integration from Huazhong University of Science and Technology in 2013. He is now a post-doctor in Huazhong University of Science and Technology. His research interests include DNA computing, DNA encoding and membrane computing. He has published more than 30 papers on DNA computing and membrane computing.