基于 Tile 自组装模型的最大匹配问题算法研究

周 旭[1,2],周炎涛[2,3],李肯立[2],欧阳艾嘉[2],潘 果[2]
(1.嘉兴学院数理与信息工程学院,浙江嘉兴 314001; 2.湖南大学信息科学与工程学院,湖南长沙 410082;
3.湖南大学电气与信息工程学院,湖南长沙 410082)

摘 要: Tile 自组装模型作为一种重要的 DNA 计算模型,在解决 NP 问题时展现出了巨大优势.文中针对现有最大匹配问题 DNA 计算算法实验操作复杂,错误率高的缺点,提出了一种基于 Tile 自组装模型的最大匹配问题新算法.算法所需的 Tile 分子种类为 $O(mn)$,所需生物操作数为 $O(1)$,计算时间为 $O(m)$,计算空间复杂度为 $O(mn)$(其中 m 为边数,n 为顶点数,且 $O(m)=O(n^2)$).与现有的最大匹配问题 DNA 计算算法相比,本算法不仅可靠性更好,而且更具可操作性.

关键词: DNA 计算; Tile 自组装模型; 最大匹配问题; NP 完全问题; 并行计算
中图分类号: TP3 **文献标识码:** A **文章编号:** 0372-2112 (2015)02-0262-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.02.009

Efficient Maximum Matching Problem Algorithms in the Tile Assembly Model

ZHOU Xu[1,2], ZHOU Yan-tao[2,3], LI Ken-li[2], OUYANG Ai-jia[2], PAN Guo[2]
(1. *College of Mathematics and Information Engineering, Jiaxing University, Jiaxing, Zhejiang* 314001, *China*;
2. *College of Information Science and Engineering, Hunan University, Changsha, Hunan* 410082, *China*;
3. *College of Electrical and Information Engineering, Hunan University, Changsha, Hunan* 410082, *China*)

Abstract: The tile self-assembly model is an important DNA computing model. It's useful for handling the NP problem. Currently, when using the DNA computing to solve the maximum matching problem, it will be hard to experiment and easy to make mistakes. Therefore, based on the tile self-assembly model, a new algorithm for the maximum matching problem is designed. The present algorithm needs $O(mn)$ types of tile molecular, its bio-operation is $O(1)$, the computing time is $O(m)$ and space complexity is $O(mn)$(where m is the number of edges, n is the number of vertices, $O(m)=O(n^2)$). Compared to the existed algorithms, the proposed algorithm is effectiveness and correctness.

Key words: DNA computing; Tile self-assembly model; maximum matching problem; NP-complete problem; parallel computing

1 引言

基于生物材料的计算模型和算法研究被称为生物计算.和以序优化为代表的一类启发式模型的类似[1],生物计算模型大多通过空间换计算,能有效求解计算困难问题[2~5].1994 年,Adleman 提出了一种新的生物计算模型-DNA 计算[3].DNA 计算作为一个新兴的研究领域,它通过 DNA 分子的生物化学性质实现各种计算[3].尽管 DNA 计算机具有巨大的发展潜力,但其在可靠性、可操作性及灵活性等方面存在的问题已经成为其进一步走向实际应用的巨大障碍[4].

Winfree 提出了二维自组装模型又称为 Tile 自组装模型[6].Tile 自组装模型以其高度的并行性,可编程性,自治性及纳米特性等优势推动 DNA 计算机进一步发展[4~7].现今 Tile 自组装模型已经被广泛应用于解决众多计算困难问题[5,8~12].李菲等人提出了一种新型 DNA 自组装磁珠光电检测系统[5];Burn 提出了基于 Tile 自组装的 3-SAT 算法[8];吴帆等人提出了一种求解 N 皇后问题 DNA 自组装算法[9];李肯立等人提出了一种基于自组装模型的最大团问题 DNA 计算算法[10].

收稿日期:2013-09-18;修回日期:2014-04-22;责任编辑:马兰英
基金项目:国家自然科学基金重点项目(No.61133005);国家自然科学基金(No.61173013,No.61202109);湖南省杰出青年基金(No.12JJ1011);浙江省教育厅科研计划项目(No.Y201226110);湖南省科技厅科技计划项目(No.2013GK3082,No.2014GK3043);湖南省教育厅项目(No.08D092,No.13C333)

最大匹配问题[13]作为一个著名的 NP 完全问题,其 DNA 计算算法已有相当研究[11,12].然而文献[11]中提出的算法基于表面模型、文献[12]中提出算法基于粘贴模型,此两种模型均很难克服实验操作难度较大,需要大量的人工干预的缺点.此外,从公开发表的刊物看,关于最大匹配问题 Tile 自组装模型的研究还比较少.为此,在文献[11,12]工作的基础上,本文首先基于 Tile 自组装模型的特性,提出了一种最大匹配问题 Tile 自组装模型;随后基于所提出模型给出了求解最大匹配问题的算法步骤并进行性能分析;最后通过 xgrow 软件[9,13]对算法进行实例模拟,进一步验证模型及算法的正确性和有效性.

2　Tile 自组装模型

本文使用文献[6]中提出的 Tile 自组装模型,以下将介绍其中的分子结构,相关生物操作及抽象描述.

2.1　Tile 分子的结构

Tile 自组装模型中的 Tile 分子可以抽象为一个带有标签的矩阵块,每个标签识别一个特定的粘性末端[6,8].如图 1,North,South,West,East 可用于标记该 Tile 分子的四个粘性末端,Value 为 Tile 分子的标识,通常定义 Tile 分子时可以省略.不同 Tile 分子之间的组装通过四个粘性末端的碱基互补配对来实现.

2.2　相关生物操作

本文模型在基本生物操作如退火,连接,荧光标记及凝胶电泳操作的基础上,引入了 Adleman-Lipton 模型中抽取,检测,读取等生物操作[10,12].

2.3　Tile 自组装模型的抽象描述

本文对传统 Tile 自组装模型进行扩展.扩展后的 Tile 自组装模型定义为 5 元组 < TileSet, g, t, Configuration, BioOperations >,其中 TileSet, g, t, Configuration 的定义见文献[8],BioOperations 表示自组装模型中的基本生物操作.

3　问题描述

定义 1　最大匹配问题[12]　无向图 $G=(V,E)$ 中具有顶点集 $V=\{v_1,v_2,v_3,\cdots,v_n\}$,边集 $E=\{e_1,e_2,e_3,\cdots,e_m\}$ 其中 m 为边数,n 为顶点数.图 G 的子图 $G_{Sub}=(V_{Sub},E_{Sub})$ 满足 $V_{Sub}\subseteq V$,$E_{Sub}\subseteq E$.若图 G_{Sub} 中任意两条边 e_k,e_d 在 G 中没有公共的端点,则 E_{Sub} 为 G 的一个匹配.具有最多边的匹配称为

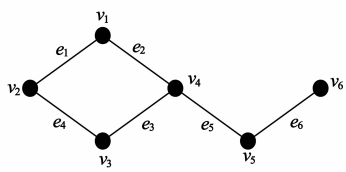


图2　简单无向图G

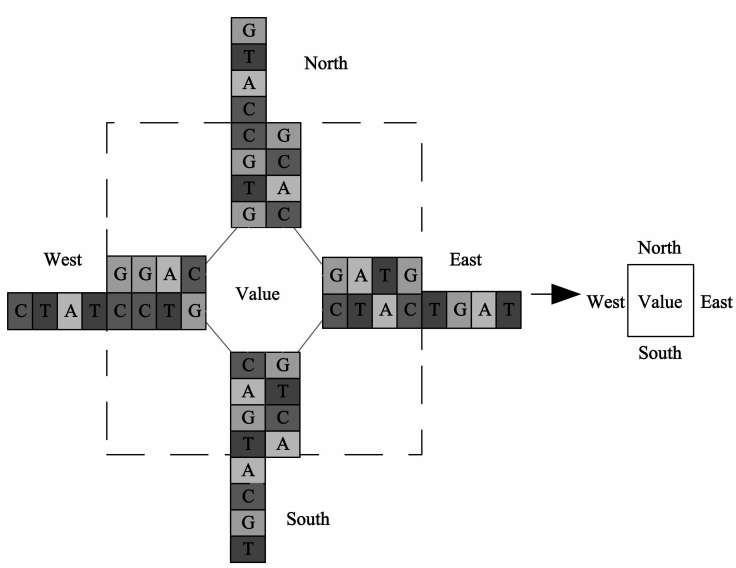


图1　Tile分子及其抽象结构

图 G 的最大匹配.求解图的最大匹配是一个经典的 NP 完全问题.可用一个 m 位二进制串 $e_1^i e_2^i\cdots e_m^i$ 表示图 G 中的一种可能的匹配方案,i 标识边是否在匹配中($i\in\{0,1\}$).如图 2 中的最大匹配 $\{e_1,e_3,e_6\}$ 可分别表示为 "$e_1^1e_2^0e_3^1e_4^0e_5^0e_6^1$".

4　基于 Tile 自组装模型的最大匹配问题算法

4.1　基于 Tile 自组装模型的最大匹配问题算法框架

基于 Tile 自组装模型的最大匹配问题算法框架如下:

(1)根据所求解最大匹配问题的基本信息(图 G 中边信息,顶点信息)设计基本 Tile 分子.

(2)初始子系统中 Tile 分子充分自组装后将生成问题的输入即种子配置.

(3)选择子系统中将其 Tile 分子充分自组装到种子配置上后将输出解空间配置.

(4)检测子系统中输入解空间配置后,基于其用于检测的 Tile 分子充分自组装后,生成最终配置.

(5)通过凝胶电泳、PCR 扩增放大及 DNA 标准序列的测定操作读取最终结果.

4.2　最大匹配问题 Tile 自组装模型

最大匹配问题 Tile 自组装模型由初始子系统、选择子系统和检测子系统组成.

4.2.1　初始子系统

初始子系统将建立在 L 型配置上,将下边界及右边界编码最大匹配问题信息的输入,充分自组装后,左边界 Tile 分子携带输出信息.

定理 1　定义 $\sum_S=\{V,E,\#,=\}$,V、E 分别为图 G 中顶点集合及边集合.初始子系统 SystemSeed = $\langle T_{\text{Seed}},g_{\text{Seed}},\tau_{\text{Seed}},C_{\text{Seed}},\text{BioOperations}\rangle$,其中 T_{Seed} 表示

Tile 分子的集合，$g_{\text{Seed}}=1$，$\tau_{\text{Seed}}=2$，C_{Seed}代表最大匹配问题输入信息的种子配置，BioOperations 表示所需的生物操作. SystemSeed 中 T_{Seed}作为输入，输出种子配置 C_{Seed}.

证明 基本 Tile 分子集合 T_{Seed}如图 3，其中 $e_{\hat{m}}$ 表示图 G 中的第 $\hat{m}$ 条，$v_{\hat{n}}$ 表示图 G 中的第 $\hat{n}$ 个顶点，($1\leqslant\hat{m}\leqslant m, 1\leqslant\hat{n}\leqslant n$，$m$，$n$ 分别代表图 G 中的边数和顶点数). 初始子系统中通过将 T_{Seed}集合中的 Tile 分子充分自组装后将生成种子配置 C_{Seed}，C_{Seed}中存在一些位置 $(x,y)\in\mathcal{Z}^2$，满足如下约束：

(1) $C_{\text{Seed}}(0,0)=<\text{|},\text{null},\#,\text{null}>$.

(2) For all $1\leqslant y\leqslant n-1$,

$C_{\text{Seed}}(0,y)=<\text{|},\text{|},e_{\hat{m}},\text{null}>$.

(3) $C_{\text{Seed}}(0,n)=<\text{null},\text{|},\#,\text{null}>$.

(4) For all $-n+1\leqslant x\leqslant 2$, $C_{\text{Seed}}(x,0)=<v_{\hat{n}},\text{null},\#,\#>$.

(5) $C_{\text{Seed}}(n,0)=<\text{|},\text{null},\text{null},\#>$.

(6) $C_{\text{Seed}}(-1,0)=<\text{Choose},\text{null},\#,\#>$.

(7) 对于其他的位置 $(x,y)\in\mathcal{Z}^2$，$(x,y)\notin C_{\text{Seed}}$.

如图 3，初始子系统中共需 Tile 分子类型为 $m+n+3$.

4.2.2 选择子系统

选择子系统负责生成最大匹配问题的解空间即所有的匹配方案. 该系统中通过将种子配置及相关 Tile 分子充分自组装生成解空间配置.

定理 2 定义 $\sum_C=\{\text{Choose},\approx,\#,e_m^1,e_m^{\cdot}\}$，选择子系统 SystemChoose $=\langle T_{\text{Choose}},g_{\text{Choose}},\tau_{\text{Choose}},C_{\text{Result}},\text{BioOperations}\rangle$，其中 T_{Choose}表示基本选择子系统 Tile 分子的集合，$g_{\text{Choose}}=1$，$\tau_{\text{Choose}}=2$，C_{Result}表示解空间配置，BioOperations 表示所需的生物操作. SystemChoose 将生成最大匹配问题的解空间配置 C_{Result}，而在某一解空间配置基础上能够产生唯一的最终配置.

证明 基本 Tile 分子集合 T_{Choose}如图 4，其中 $e_{\hat{m}}$ 表示图 G 中的第 $\hat{m}$ 条边($1\leqslant\hat{m}\leqslant m$)，$e_m^1$ 表示边 $e_{\hat{m}}$ 在当前匹配方案中，$\approx$表示边 $e_{\hat{m}}$ 不在当前匹配方案中. SystemChoose 中通过将 T_{Choose}中的选择分子自组装到种子配置 C_{Seed}上生成解空间配置 C_{Result}. C_{Result}中存在一些位置 $(x,y)\in\mathcal{Z}^2$，满足如下约束：

(1) For all $1\leqslant y\leqslant n-1$, $C_{\text{Result}}(0,y)=\langle\text{Choose},\text{Choose},\approx,e_{\hat{m}}\rangle$或$\langle\text{Choose},\text{Choose},e_m^1,e_{\hat{m}}\rangle$.

(2) $C_{\text{Result}}(0,n)=<\text{null},\text{Choose},\#,\#>$.

(3) 对于其他的位置 $(x,y)\in Z^2$，$(x,y)\notin C_{\text{Result}}$.

如图 4，初始子系统中共需 Tile 分子类型为 $2m+1$.

4.2.3 检测子系统

检测子系统用于筛选出最大匹配问题的所有合法解，主要通过用于检测的 Tile 分子充分自组装后得到最终配置，若最终配置中含有特定标识的 Tile 分子，则其上携带的输出信息为最大匹配问题的合法解.

定理 3 定义 $\sum_D=\{v_{\hat{n}}^1,v_{\hat{n}},e_{\hat{m}}^1,\approx,\text{|},\#\}$，检测子系统 SystemDetect $=\langle T_{\text{Detect}},g_{\text{Detect}},\tau_{\text{Detect}},C_{\text{Final}},\text{BioOperations}\rangle$，其中 T_{Detect}表示检测子系统中基本 Tile 分子集合，$g_{\text{Detect}}=1$，$\tau_{\text{Detect}}=2$. 解空间配置 C_{Result}与 T_{Detect}中的检测分子充分自组装后得到最终配置 C_{Final}. 含有 SUC 标识 Tile 分子的 C_{Final}为合法最终配置，其上携带的输出信息为最大匹配问题的合法解.

证明 如图 5，检测子系统 SystemDetect 中检测分子集合 T_{Detect}，其中 $e_{\hat{m}}$ 表示图 G 中的第 $\hat{m}$ 条边，$v_{\hat{n}}$ 表示图 G 中的第 $\hat{n}$ 个顶点，$e_{\hat{m}}^1$ 表示边 $e_{\hat{m}}$ 在当前匹配方案中，$\approx$表示边 $e_{\hat{m}}$ 不在当前匹配方案中，$v_{\hat{n}}^1$ 表示顶点 $v_{\hat{n}}$ 被当前匹配中的边覆盖($1\leqslant\hat{m}\leqslant m, 1\leqslant\hat{n}\leqslant n$). 基于解空间配置 C_{Result}，SystemDetect 中通过将检测 Tile 分子自组装到种子配置 C_{Result}上生成最终配置 C_{Final}. C_{Final}中存在一些位置 $(x,y)\in\mathcal{Z}^2$，满足如下约束：

(1) For all $-m-1\leqslant x\leqslant -2, 1\leqslant y\leqslant m$, $C_{\text{Final}}(x,y)=\langle v_{\hat{n}}^1,v_{\hat{n}},e_{\hat{m}}^1,e_{\hat{m}}^1\rangle$、$\langle v_{\hat{n}},v_{\hat{n}},e_{\hat{m}}^1,e_{\hat{m}}^1\rangle$、$\langle v_{\hat{n}}^1,v_{\hat{n}}^1,e_{\hat{m}}^1,e_{\hat{m}}^1\rangle$、$\langle v_{\hat{n}},v_{\hat{n}},\approx,\approx\rangle$或$\langle v_{\hat{n}}^1,v_{\hat{n}}^1,\approx,\approx\rangle$.

(2) For all $-2\leqslant x\leqslant -m-1$, $C_{\text{Final}}(x,y=m+1)=\langle\text{null},v_{\hat{n}}^1,\#,\#\rangle$或$\langle\text{null},v_{\hat{n}},\#,\#\rangle$.

(3) For all $1\leqslant y\leqslant m-1$, $C_{\text{Final}}(-m-2,y)=\langle\text{|},\text{|},\text{null},e_{\hat{m}}^1\rangle$或$\langle\text{|},\text{|},\text{null},\approx\rangle$.

(4) $C_{\text{Final}}(-m-2,0)=\langle\text{|},\text{null},\text{null},\#\rangle$.

(5) $C_{\text{Final}}(-m-2,m+1)=\langle\text{null},\text{|},\text{null},\#\rangle$.

(6) 其他的位置 $(x,y)\in\mathcal{Z}^2$，$C_{\text{Final}}(x,y)=\text{empty}$.

从以上定义可以发现 $\forall\, t\in T_{\text{Detect}}$，$(bd_{\text{South}}(t),bd_{\text{West}}(t))$是唯一的，因此检测子系统 SystemDetect 中针对某一解空间配置 C_{Result}将生成唯一的最终配置 C_{Final}.

如图 5，检测子系统中共需 Tile 分子类型为 $3mn+4n+m+2$.

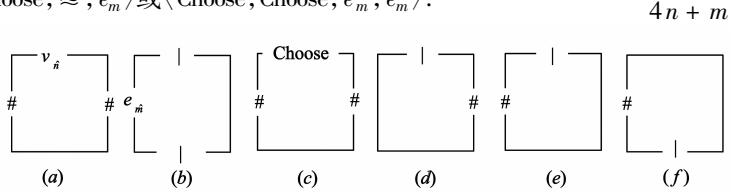


图3 初始子系统Tile分子抽象结构

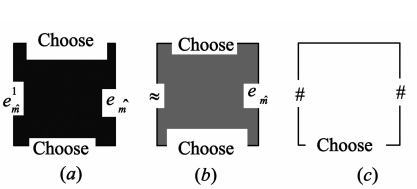


图4 选择子系统中Tile分子抽象结构

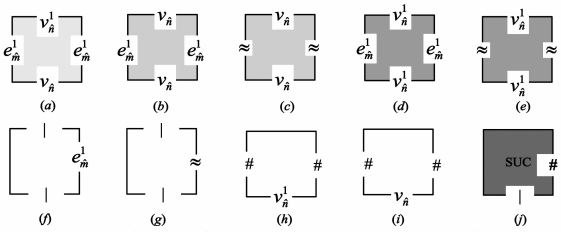


图5　检测子系统中的基本Tile分子抽象结构

4.4　基于 Tile 自组装模型的最大匹配问题算法

基于 4.3 节中的最大匹配问题 Tile 自组装模型,提出了基于 Tile 自组装模型的最大匹配问题算法如下:

算法 1　Maximum_Matching_algorithm(G, T_{tileseed}, $T_{\text{tilechoose}}$, $T_{\text{tiledetect}}$).
Input　图 $G=\{V,E\}$及试管 T_{tileseed}, $T_{\text{tilechoose}}$, $T_{\text{tiledetect}}$;
Output　最大匹配问题的解.
(1)生成初始子系统,选择子系统及检测子系统所需的基本 Tile 分子,并存储于试管 T_{tileseed}, $T_{\text{tilechoose}}$, $T_{\text{tiledetect}}$;
(2)Amplify(T_{tileseed}), Amplify($T_{\text{tilechoose}}$)and Amplify($T_{\text{tiledetect}}$);
(3)Anneal(T_{tileseed});
(4)Ligate(T_{tileseed});
(5)Merge($T_{\text{tilechoose}}$, T_{tileseed}, $T_{\text{tilechoose}}$);
(6)Anneal($T_{\text{tilechoose}}$);
(7)Ligate($T_{\text{tilechoose}}$);
(8)Merge($T_{\text{tiledetect}}$, $T_{\text{tilechoose}}$, $T_{\text{tiledetect}}$);
(9)Anneal($T_{\text{tiledetect}}$);
(10)Ligate($T_{\text{tiledetect}}$);
(11) T_{Match} = Extract_strand($T_{\text{tiledetect}}$, $\text{tile}_{\text{Succ}}$);
(12)Gel-electrophoresis(T_{Match});
(13) T_{maxMatch} = Extract_maxLength(T_{Match});
(14)IF (Detect(T_{maxMatch}) == "yes")Then
(15)　　Read(T_{maxMatch})
(16)EndIF

定理 4　算法 1 可并行求解最大匹配问题.

证明　算法 1 中,首先步骤(1)并行生成基本 Tile 分子,并分别存储于试管 T_{tileseed}, $T_{\text{tilechoose}}$和 $T_{\text{tiledetect}}$中;步骤(2)中通过 Amplify()操作,生成试管 T_{tileseed}, $T_{\text{tilechoose}}$和 $T_{\text{tiledetect}}$中 Tile 分子的大量拷贝;步骤(3)、(4)通过 Anneal()和 Ligate()操作,试管 T_{tileseed}中的 Tile 分子充分的自组装后将生成种子配置 C_{Seed};步骤(5)中通过 Merge()操作,将试管 T_{tileseed}和 $T_{\text{tilechoose}}$中 Tile 分子合并到试管 $T_{\text{tilechoose}}$中;步骤(6)、(7)通过 Anneal()和 Ligate()操作,试管 $T_{\text{tilechoose}}$中的 Tile 分子充分的自组装后将生成解空间配置 C_{Result};步骤(8)中通过 Merge()操作,将试管 $T_{\text{tiledetect}}$中的检测 Tile 分子加入到试管 $T_{\text{tilechoose}}$中;步骤(9)、(10)通过 Anneal()和 Ligate()操作,试管 $T_{\text{tiledetect}}$中的检测 Tile 分子充分的组装到种子配置后将生成最终配置 C_{Final};步骤(11)中通过操作,抽取出含有 $\text{tile}_{\text{Succ}}$ = <null, |, null, # > 的 Tile 组装体,并存放于试管中 T_{Match},由此将得到即图 G 中的所有匹配方案.步骤(12)、(13)中通过凝胶电泳技术分离出不同长度的 Tile 输出链,通过抽取操作提取最长的 Tile 输出链;步骤(14)首先通过 Detect()操作检测试管中是否包含 Tile 组装体,如果返回"yes" 则通过 Read()操作得到最大匹配问题的解.

4.5　性能分析

引理 1　本文提出的最大匹配问题 Tile 自组装高效模型中,需要的 Tile 分子种类为 $O(mn)$、需要的计算时间为 $O(m)$、计算空间为 $O(mn)$及所需生物操作数为 $O(1)$.

证明　本文模型主要由初始子系统、选择子系统及检测系统 3 个部分组成,所需 Tile 分子种类分别为 $m+n+3$、$2m+1$ 和 $3mn+4n+m+2$.综上分析可知本模型中所需 Tile 分子共为 $3mn+5n+4m+6$,因而 Tile 分子种类数为 $O(mn)$.

本文模型中 Tile 自组装体的深度为 $m+2$,因而计算时间复杂度为 $O(m)$.

本文模型中 Tile 自组装体的最大面积为$(m+2)(n+2)=mn+2m+2n+4$,因而计算空间复杂度为 $O(mn)$.

从最大匹配问题 Tile 自组装模型的算法步骤分析发现,算法共需 16 次生物操作.可见本文算法所需生物操作数为 $O(1)$.

4.6　算法模拟

如图 2,图 G 中顶点集合 $V=\{v_1,v_2,v_3,v_4,v_5,v_6\}$ 边集合 $E=\{e_1,e_2,e_3,e_4,e_5,e_6\}$.以下将以求解图 G 的最大匹配问题为例,对文中提出的算法进行模拟.

4.6.1　Tile 分子编码

本文最大匹配问题 Tile 自组装模型算法中首先根据所求最大匹配问题信息生成所需的 Tile 分子.

(1)初始子系统中基本 Tile 分子有:$\langle v_{\hat{n}}, \text{null}, \#, \# \rangle$, $\langle |, |, e_{\hat{m}}, \text{null} \rangle$、$\langle \text{Choose}, \text{null}, \#, \# \rangle$、$\langle |, \text{null}, \text{null}, \# \rangle$、$\langle |, \text{null}, \#, \text{null} \rangle$和$\langle \text{null}, |, \#, \text{null} \rangle$其中 $v_{\hat{n}} \in V$, $e_{\hat{m}} \in E$.

(2)选择子系统中基本 Tile 分子有:$\langle \text{Choose}, \text{Choose}, e^{1}_{\hat{m}}, e_{\hat{m}} \rangle$、$\langle \text{Choose}, \text{Choose}, \approx, e_{\hat{m}} \rangle$和$\langle \text{null}, \text{Choose}, \#, \# \rangle$其中 $e_{\hat{m}} \in E$.

(3)检测子系统中基本 Tile 分子如下:

①若边 $e_{\hat{m}}$ 被选中.当边 $e_{\hat{m}}$ 中一个顶点为 $v_{\hat{n}}$ 且 $v_{\hat{n}}$ 未被其它边覆盖,设计了 Tile 分子

$\langle v^1_1, v_1, e^1_1, e^1_1 \rangle$, $\langle v^1_2, v_2, e^1_1, e^1_1 \rangle$, $\langle v^1_1, v_1, e^1_2, e^1_2 \rangle$,
$\langle v^1_4, v_4, e^1_2, e^1_2 \rangle$, $\langle v^1_3, v_3, e^1_3, e^1_3 \rangle$, $\langle v^1_4, v_4, e^1_3, e^1_3 \rangle$,
$\langle v^1_2, v_2, e^1_4, e^1_4 \rangle$, $\langle v^1_3 1, v_3, e^1_4, e^1_4 \rangle$, $\langle v^1_4, v_4, e^1_5, e^1_5 \rangle$,
$\langle v^1_5, v_5, e^1_5, e^1_5 \rangle$, $\langle v^1_5, v_5, e^1_6, e^1_6 1 \rangle$, $\langle v^1_6, v_6, e^1_6, e^1_6 \rangle$;

当边 $e_{\hat{m}}$ 与顶点为 $v_{\hat{n}}$ 无关且 $v_{\hat{n}}$ 未被其它边覆盖,设计 Tile 分子

$\langle v_3, v_3, e_1^1, e_1^1\rangle$, $\langle v_4, v_4, e_1^1, e_1^1\rangle$, $\langle v_5, v_5, e_1^1, e_1^1\rangle$, $\langle v_6, v_6, e_1^1, e_1^1\rangle$, $\langle v_2, v_2, e_2^1, e_2^1\rangle$, $\langle v_3, v_3, e_2^1, e_2^1\rangle$, $\langle v_5, v_5, e_2^1, e_2^1\rangle$, $\langle v_6, v_6, e_2^1, e_2^1\rangle$, $\langle v_1, v_1, e_3^1, e_3^1\rangle$, $\langle v_2, v_2, e_3^1, e_3^1\rangle$, $\langle v_5, v_5, e_3^1, e_3^1\rangle$, $\langle v_6, v_6, e_3^1, e_3^1\rangle$, $\langle v_1, v_1, e_4^1, e_4^1\rangle$, $\langle v_4, v_4, e_4^1, e_4^1\rangle$, $\langle v_5, v_5, e_4^1, e_4^1\rangle$, $\langle v_6, v_6, e_4^1, e_4^1\rangle$, $\langle v_1, v_1, e_5^1, e_5^1\rangle$, $\langle v_2, v_2, e_5^1, e_5^1\rangle$, $\langle v_3, v_3, e_5^1, e_5^1\rangle$, $\langle v_6, v_6, e_5^1, e_5^1\rangle$, $\langle v_1, v_1, e_6^1, e_6^1\rangle$, $\langle v_2, v_2, e_6^1, e_6^1\rangle$, $\langle v_3, v_3, e_6^1, e_6^1\rangle$, $\langle v_4, v_4, e_6^1, e_6^1\rangle$;

当边 $e_{\hat{m}}$ 与顶点 $v_{\hat{n}}$ 无关且 $v_{\hat{n}}$ 已被覆盖,设计 Tile 分子

$\langle v_3^1, v_3^1, e_1^1, e_1^1\rangle$, $\langle v_4^1, v_4^1, e_1^1, e_1^1\rangle$, $\langle v_5^1, v_5^1, e_1^1, e_1^1\rangle$, $\langle v_6^1, v_6^1, e_1^1, e_1^1\rangle$, $\langle v_2^1, v_2^1, e_2^1, e_2^1\rangle$, $\langle v_3^1, v_3^1, e_2^1, e_2^1\rangle$, $\langle v_5^1, v_5^1, e_2^1, e_2^1\rangle$, $\langle v_6^1, v_6^1, e_2^1, e_2^1\rangle$, $\langle v_1^1, v_1^1, e_3^1, e_3^1\rangle$, $\langle v_2^1, v_2^1, e_3^1, e_3^1\rangle$, $\langle v_5^1, v_5^1, e_3^1, e_3^1\rangle$, $\langle v_6^1, v_6^1, e_3^1, e_3^1\rangle$, $\langle v_1^1, v_1^1, e_4^1, e_4^1\rangle$, $\langle v_4^1, v_4^1, e_4^1, e_4^1\rangle$, $\langle v_5^1, v_5^1, e_4^1, e_4^1\rangle$, $\langle v_6^1, v_6^1, e_4^1, e_4^1\rangle$, $\langle v_1^1, v_1^1, e_5^1, e_5^1\rangle$, $\langle v_2^1, v_2^1, e_5^1, e_5^1\rangle$, $\langle v_3^1, v_3^1, e_5^1, e_5^1\rangle$, $\langle v_6^1, v_6^1, e_5^1, e_5^1\rangle$, $\langle v_1^1, v_1^1, e_6^1, e_6^1\rangle$, $\langle v_2^1, v_2^1, e_6^1, e_6^1\rangle$, $\langle v_3^1, v_3^1, e_6^1, e_6^1\rangle$, $\langle v_4^1, v_4^1, e_6^1, e_6^1\rangle$;

②若某边未被选中设计了 Tile 分子$\langle v_{\hat{n}}, v_{\hat{n}}, \approx, \approx\rangle$,$\langle v_{\hat{n}}^1, v_{\hat{n}}^1, \approx, \approx\rangle$,其中 $v_{\hat{n}} \in V$.

③边界 Tile 分子主要有$\langle |, |, \text{null}, e_{\hat{m}}^1\rangle$,$\langle |, |, \text{null}, \approx\rangle$、$\langle \text{null}, v_{\hat{n}}, \#, \#\rangle$、$\langle \text{null}, |, \text{null}, \#\rangle$,其中 $e_{\hat{m}} \in E, v_{\hat{n}} \in V$.

④设计了含有 SUC 标识的 Tile 分子$\langle \text{null}, |, \text{null}, \#\rangle$.

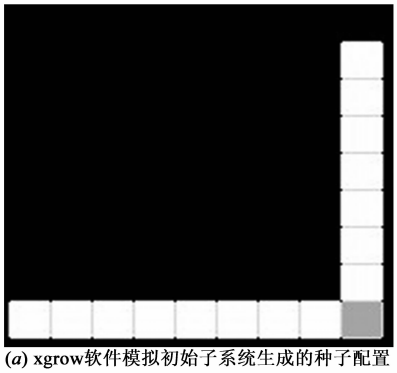
4.6.2 算法的模拟实验

本文在 Linux 环境下,采用通用的仿真软件 xgrow[9,13]对文中提出算法进行模拟实现.

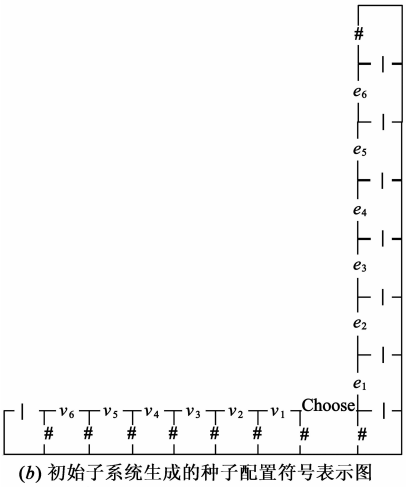
采用了 4.6.1 节的 Tile 编码方案.通过 Amplify()操作产生各类 Tile 分子的大量拷贝.求解过程中首先初始子系统中通过 Anneal()和 Ligate()操作即加入连接酶并通过退火操作生成种子配置.xgrow 模拟生成的种子配置见图 6(*a*),其符号表示见图 6(*b*).

其次,选择子系统中以种子配置为输入,通过 Anneal()和 Ligate()操作,Tile 分子充分自组装后,生成解空间配置即所有的匹配方案.图 7(*a*)、7(*b*)为 xgrow 模拟软件生成的子图$\{e_1, e_2, e_6\}$及$\{e_1, e_3, e_6\}$对应的解空间配置,其相关符号表示见图 7(*c*)、7(*d*).

随后,检测子系统中检测 Tile 分子通过 Anneal()和 Ligate()操作充分自组装后得到最终配置.xgrow 模拟软件生成匹配$\{e_1, e_3, e_6\}$及$\{e_1, e_2, e_6\}$对应的最终配置如图 8(*a*)、8(*b*),其符号表示见图 8(*c*)、8(*d*).其中包

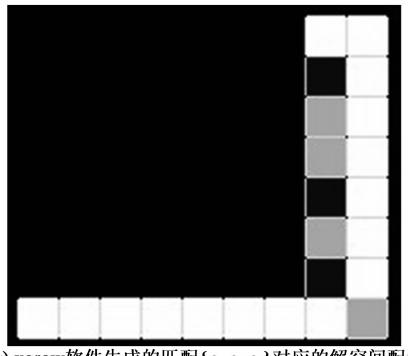


(*a*) xgrow软件模拟初始子系统生成的种子配置

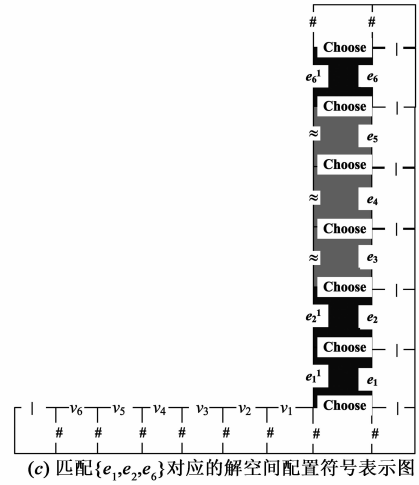


(*b*) 初始子系统生成的种子配置符号表示图

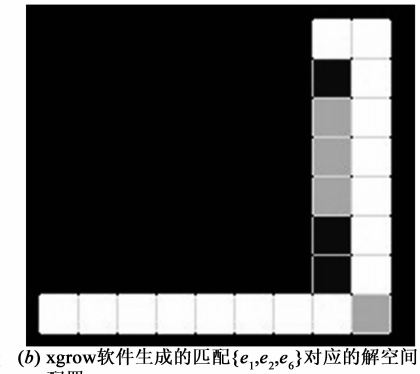
图6 初始子系统生成的种子配置



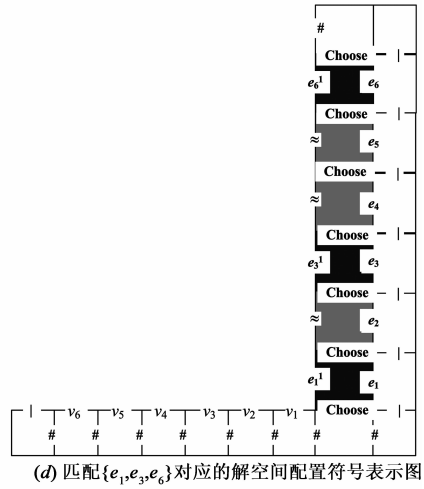
(*a*) xgrow软件生成的匹配$\{e_1,e_3,e_6\}$对应的解空间配置



(*c*) 匹配$\{e_1,e_2,e_6\}$对应的解空间配置符号表示图



(*b*) xgrow软件生成的匹配$\{e_1,e_2,e_6\}$对应的解空间配置

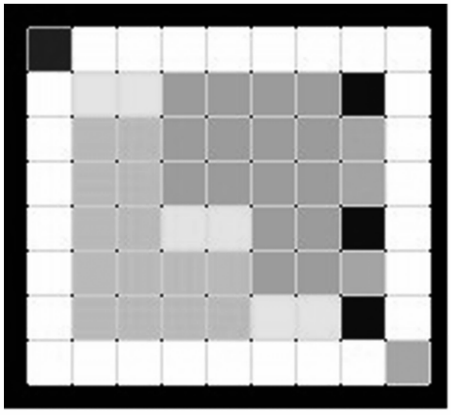


(*d*) 匹配$\{e_1,e_3,e_6\}$对应的解空间配置符号表示图

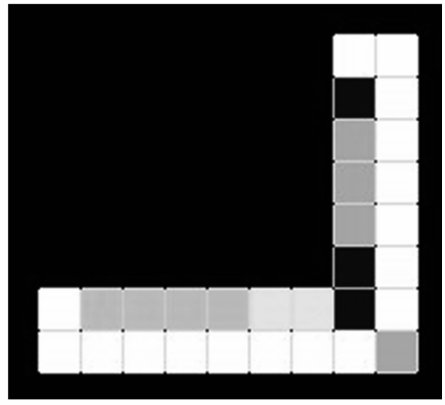
图7 选择子系统生成的两个解空间配置实例图

含检测 Tile 分子〈null,|,null,#〉的最终配置为一完整的 Tile 自组装体,表示合法的匹配方案;

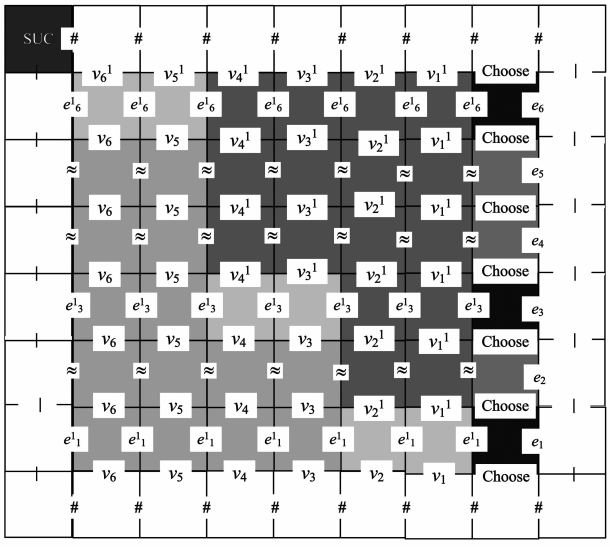
最后,抽取出包含检测 Tile 分子〈null,|,null,#〉的 Tile 自组装体,获得图 G 最大匹配问题的可满足解.并通过凝胶电泳技术抽获取最大匹配问题的最终解.



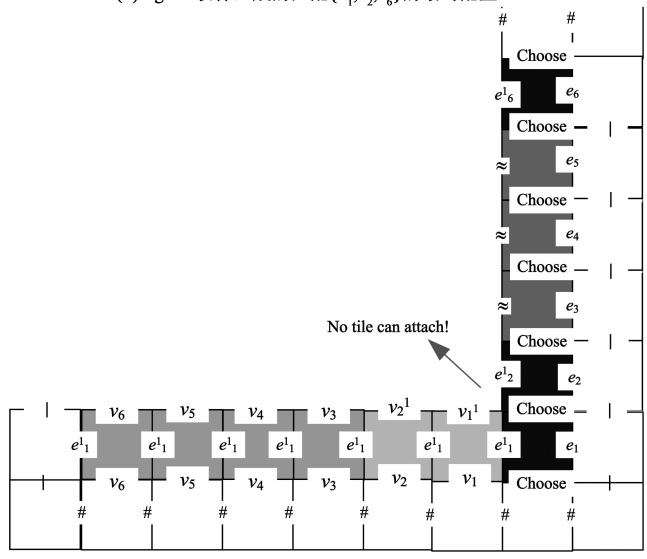
(*a*) xgrow软件生成的匹配{e_0,e_3,e_6}的最终配置



(*b*) xgrow软件生成的匹配{e_1,e_2,e_6}的最终配置



(*c*) 匹配{e_1,e_3,e_6}的最终配置符号表示图



(*d*) 匹配{e_1,e_2,e_6}的最终配置符号表示图

图8　检测子系统中两个最终配置实例图

5　结论

随着生物技术的进步,Tile 自组装模型凭借其自组装、可编程等特性而得到广泛关注.为此,本文将 Tile 自组装模型引入最大匹配问题 DNA 计算算法的设计中.首先提出了一种最大匹配问题的 Tile 自组装模型;其次,基于提出的模型,设计了基于 Tile 自组装的最大匹配问题算法.与已有最大匹配问题 DNA 计算算法相比,该算法仅需常数级生物操作,更加可靠,且更具可操作性.

参考文献

[1] ZhangF, Cao J, Hwang K, et al. Ordinal optimized scheduling of scientific workflows in elastic compute clouds[A]. Proc 3rd IEEE International Conference on Cloud Computing Technology and Science, 2011[C]. Athens: IEEE, 2011. 9 – 17.

[2] ZhangX Y, Wang S, Niu Y Y, et al. Tissue P systems with cell separation: attacking the partition problem[J]. Science China Information Sciences, 2011, 54(2): 293 – 304.

[3] Adleman L M. Molecular computation of solutions to combinatorial problems[J]. Science, 1994, 266(11): 1021 – 1024.

[4] 俞洋,陆建华,王东方,等.基于 DNA/RNA 的逻辑门与逻辑运算[J].科学通报,2013,58(2):131 – 140.
Yu Yang, Lu Jian-hua, Wang Dong-huang, et al. DNA/RNA based logic gates and computing[J]. Chinese Science Bulletin, 2013, 58(2): 131 – 140. (in Chinese)

[5] 李菲,许进.一种新型 DNA 自组装磁珠光电检测系统及其在 DNA 计算机研制中的应用[J].计算机学报,2013,36(9):1826 – 1833.
Li Fei, Xu Jin. A new photoelectric DNA-Detection platform

with assembled magnetic beads and its application on DNA computer[J]. Chinese Journal of Computers, 2013, 36(9): 1826 – 1833. (in Chinese)

[6] Winfree E. Algorithmic self-assembly of DNA[D]. Pasadena: California Institute of Technology, 1998.

[7] Seeman N C. DNA nanotechnology: Novel DNA constructions [J]. Annual Review of Biophysics and Bio-molecular Structure, 1998, 27(1): 225 – 248.

[8] Brun Y. Effective 3-SAT algorithms in the tile assembly model [J]. Natural Computing, 2012, 11(2): 209 – 229.

[9] 吴帆,李肯立.基于自组装的 N 皇后问题 DNA 计算算法[J].电子学报,2013,41(11):2174 – 2180.
Wu Fan, Li Ken-li. An algorithm in tile assembly model for N queen problem[J]. Acta Electronica Sinica, 2013, 41(11): 2174 – 2180. (in Chinese)

[10] 李肯立,罗兴,吴帆,等.基于自组装模型的最大团问题 DNA 计算算法[J].计算机研究与发展,2013,50(3):666 – 675.
Li Ken-li, Luo Xin, Wu Fan, et al. An algorithm in tile assembly model for maximum clique problem[J]. Journal of Computer Research and Development, 2013, 50(3): 666 – 675. (in Chinese)

[11] 陈治平,李小龙,等.最佳匹配问题的 DNA 表面计算模型[J].计算机研究与发展,2005,42(7):1241 – 1246.
Chen Zhi-ping, Li Xiao-long, et al. A surface-based DNA algorithm for the perfect matching problem[J]. Journal of Computer Research and Development, 2005, 42(7): 1241 – 1246. (in Chinese)

[12] 周旭,李肯立,乐光学,等.一种最大匹配问题 DNA 计算算法[J].计算机研究与发展,2011,48(11):2147 – 2154.
Zhou Xu, Li Ken-li, Yue Guang-xue, et. al. A volume molecular solution for the maximum matching problem on DNA-based computing[J]. Journal of Computer Research and Development, 2011, 48(11): 2147 – 2154. (in Chinese)

[13] MaX J, Lombardi F. Synthesis of tile sets for DNA self-assembly[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2008, 27(5): 963 – 967.

作者简介

周 旭 女,1983 年生于江苏宿迁,湖南大学博士研究生,研究方向为 DNA 计算和并行计算.
E-mail: zhouxu2006@126.com

周炎涛(通信作者) 男,1963 年生于湖南汉寿,湖南大学电气信息与工程学院教授,博士生导师,研究方向为 DNA 计算和并行计算.
E-mail: yantao_z@hnu.edu.cn