最大匹配问题的 DNA 表面计算模型

刘文斌[1,2],高 琳[3],王淑栋[1,2],刘向荣[1],许 进[1]
(1. 华中科技大学控制科学与工程系,湖北武汉 430074; 2. 山东科技大学信息科学与工程学院,山东泰安 271019;
3. 西安电子科技大学雷达信号处理国家重点实验室,陕西西安 710071)

摘 要: 本文给出了一个最大匹配问题的 DNA 表面计算模型,我们在表面上逐步生成解空间的同时,利用酶切技术删除所产生的"不可行解",从而大大减少了最终生成的解空间. 最后,我们还研究了边的排列顺序对解空间的生成过程的影响. 结果表明,通过对图中的边进行合理的编排也能减小不可行解的生成.

关键词: DNA 计算; 表面方式; 最大匹配问题
中图分类号: TP18 **文献标识码**: A **文章编号**: 0372-2112 (2003) 10-1496-04

A Surface-Based DNA Algorithm for Maximal Matching Problem

LIU Wen-bin[1,2], GAO Lin[3], WANG Shu-dong[1,2], LIU Xiang-rong[1], XU Jin[1]
(1. *Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei* 430074, *China*;
2. *College of Information Science and Engineering, Shandong University of Science and Technology, Taian, Shandong* 271019, *China*;
3. *National Key Lab. of Radar Signal Processing, Xidian Univ., Xi'an, Shanxi* 710071, *China*)

Abstract: This paper first proposes a DNA algorithm based on surface for the maximal matching problem. The main idea is to generate the possible solution space step by step in the solid surface, and at the same time these fault solutions are deleted through the digestion process of restriction enzymes. Finally the influence of the edge order on the solution generating process is discussed. Our results show that the solution space can be efficiently reduced by a reasonable arrangement of the order of edges.

Key words: DNA computing; surface-based fashion; maximum matching problem

1 引言

1994 年,Southern California 大学的 Adleman 博士首次开创性的在试管中用 DNA 分子解决了一个简单的有向图的哈密尔顿路问题(Hamiltonian Path Problem,简记为 HPP)[1]. 随后,Lipton 及其他一大批科学家投身到 DNA 分子计算这一领域,对组合优化问题中的许多 NP - 完全问题提出了各种计算模型[2~5,7]. 尽管如此,DNA 计算机的真正实现还面临许多挑战. 主要原因是: (1) 目前 DNA 计算所采用的分子生物技术在执行速度上还难以与电子计算机相抗衡; (2) 虽然 DNA 计算具有无可比拟的并行性,但是却难以处理中等规模以上的问题,因为此时编码解空间所需的 DNA 分子的数量将以指数级的方式增长. 正如 Adleman 在文献[1]中所指出的,虽然他的方法成功的解决了一个 7 个顶点的有向哈密尔顿路问题,但是,对于规模比较大的情况则需要进一步的研究.

最大匹配问题(Maximum Matching Problem)是一个经典图论问题,在现实生活中,有许多问题,如工作分配问题,排课表问题,球队的比赛问题以及军事部署方案的制定等都可以化为求最大匹配. 本文首先给出最大匹配问题的 DNA 计算模型;然后,给出了其生物实现过程;最后,分析了基于表面方式的最大匹配问题解空间的生成过程.

2 最大匹配问题的 DNA 表面计算模型

2.1 问题描述[8]

在无向图 $G=(V, E)$ 中,对边集 E 的任一子集 $M \subseteq E$,如果 M 中任意两条边在 G 中都没有公共端点,则 M 称是 G 的一个匹配(Matching). 对 $v \in G(V)$,如果 v 的关联边都不属于 M,则称 v 是 M 的非饱和点;否则,称 v 是 M 的饱和点.

显然,一般来说,G 的匹配不是唯一的. 若 G 中没有另外的 M 使得 $|M'|>|M|$,则称 M 为 G 的最大匹配;如果 G 中不含关于匹配 M 的非饱和点,则称 M 为 G 的完美匹配. 完美匹配都是最大匹配;反之,则不真. 容易看出,图 1 包含二个最大匹配,分别为 $M_1=\{e_1, e_3, e_6\}$ 和 $M_2=\{e_2, e_6, e_8\}$.

2.2 算法步骤

对任一给定的 n 阶图 $G=(V, E)$,令 $G(E)=\{e_1, e_2, \Lambda, e_m\}$,则其可能的匹配可以很方便的用 m 位(依边的顺序进行编码)二进制字符串来表示;若二进制串的某位的值为 1,则其对应的边在匹配中;否则,则不在. 对于图 1(a),其中的二个最大匹配 $M_1=\{e_1, e_3, e_6\}$ 和 $M_2=\{e_2, e_6, e_8\}$,可以分别用二进制串表示为 10100100 和 01000101. 因此,最大匹配问题可以转化为求满足条件且含 1 最多的二进制串. 本文的算法步骤可以叙述如下:

收稿日期:2001-11-29;修回日期:2003-04-05
基金项目:国家自然科学基金(No. 60103021;60144047)

Step 1　对图 G 中的 m 位二进制串进行编码;

Step 2　生成图 G 中满足条件的所有匹配;

Step 3　找出其中含 1 最多的串,即为对应的最大匹配并输出结果.

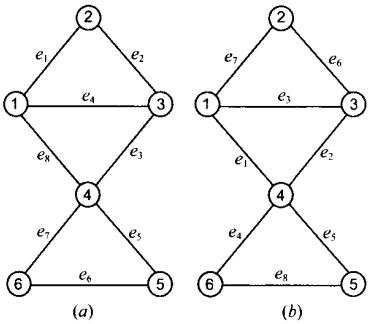

图 1　最大匹配图

3　算法的生物实现

3.1　表面计算方式的概念

目前,DNA 计算主要有试管方式(test tube-based)和表面方式(surface-based).试管方式就是在一个或多个试管的溶液里进行生化反应;而表面方式则是将对应于问题解空间的 DNA 分子固定在一块固体载体上,然后进行各种生化反应;或是在其表面上逐步生成解空间,最后获得运算结果.

表面方式常用的载体主要有玻璃片、金属片及各种有机高分子制作的薄膜等,如 Wisconsin 大学的 Smith 等是在一块镀金的表面上进行可满足问题(Satisfiability,简称 SAT)的 DNA 计算研究[5,6].载体在使用之前必须进行一定的修饰,即通过化学反应用不同的活化试剂在载体表面键合上各种各样的活性基团,以便与配基共价结合,形成具有不同生物特异性的亲和载体,用来固定不同的活性生物分子,如蛋白质、核酸等.通过这种方法固定在载体表面上的 DNA 分子,能够承受在表面上进行的各种加热、清洗及其他生化反应的作用[9].

和试管方式相比,表面方式主要有如下优点:(1)操作简单,易于实现自动化操作;(2)减少了人为操作过程中造成的损失;(3)减少了表面分子间的相互作用,增强了分子间的特异性结合;(4)结果易于纯化.因此,表面方式的研究将对 DNA 计算的真正实现有重大意义.

3.2　编码方法

我们用单链 DNA 分子编码图 1(*a*)的 8 条边,每条边的编码由二部分组成:(1)一部分表示位值,当位值为"1"时用 15bp 表示,为"0"时用 8bp 的序列表示且所有位均相同;(2)另一部分表示根部(RS)或尾部(TS),各用 20bp 表示;(3)根部或尾部与位值连接处分别设有 Eco RI 和 Bam HI 酶切位点;(4)在根部的 51 端加有生物素分子,它是用来将 DNA 分子固定在载体表面上的.

RS - e_1^1:5′- biotin - aaacgtctagatgcaGAATT - Cggcctacttaagag - 3′

e_2^1 - TS:5′- ttgttacgcagcccG - CATCCataggagactaagag - 3′

e_3^1 - TS:5′- attgccgaggtactG - GATCCataggagactaagag - 3′

e_4^1 - TS:5′- taccgaaagtatgaG - GATCCataggagactaagag - 3′

e_5^1:5′- tcgtgtgccgtgcaG - GATCCataggagactaagag - 3′

e_6^1:5′- tgtgaggttcgagaG - GATCCataggagactaagag - 3′

e_7^1:5′- ttgagcttgaaactG - GATCCataggagactaagag - 3′

e_8^1:5′- aatgcacttgtctcG - GATCCataggagactaagag - 3′

RS - e_1^0:5′- biotin - aaacgtctagatgcaGAATT - Cgatcatg - 3′

e_1^0:5′- cgatcatG - GATCCataggagactaagag - 3′($i \neq 1$)

其中大写字母为限制性内切酶识别序列,中间的"-"为酶切位点

图 2　边的编码

3.3　算法的操作过程

(1)合成(MAKE)　按照图 2 所示的编码合成各边及反应过程所需要的引物、探针等.

(2)固定(ATTACH)　在载体表面加入 RS - e_1^1 和 RS - e_1^0,因为 RS 的 5 端附有生物素分子(biotin),它会和载体表面的活性基团结合,从而将 RS - e_1^1 和 RS - e_1^0 固定在载体表面.

(3)延伸(EXTEND)　加入边 e_2^1 - TS、e_2^0 - TS 及 Rev(e_1^1 - e_2^1)、Rev(e_1^1 - e_2^0)、Rev(e_1^0 - e_2^1)、Rev(e_1^0 - e_2^0)(Rev(e_1^1 - e_2^1)表示与 e_1^1 - e_2^1 互补的序列,其方向为 3′→5′);为了防止碱基错配,将温度加热到 94 ℃,保持 2 分钟;然后加入 Taq DNA 连接酶,在 65 ℃保持 10 分钟进行连接反应;最后,加热到 94 ℃解链,并清洗干净载体表面.延伸反应完成后将温度降至 37 ℃

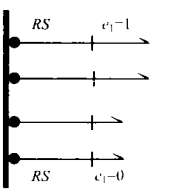

图 3　载体表面固定的 DNA 分子示意图

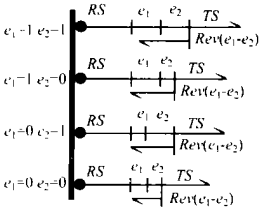

图 4　延伸反应示意图

(4)删除(DELETE)　步骤(3)完成后,载体表面上的所有 DNA 分子的末端均为 e_2 - TS,加入 Rev(e_1^1 - e_2^1 - TS)以及 dNTP,Rev(e_1^1 - e_2^1 - TS)将会与载体表面上所有与 e_2 关联的 DNA 分子杂交.然后,将温度升至 94 ℃,加入 Taq DNA 聚合酶,再在 80 ℃保持 1 分钟,等聚合反应完成后,所有与 e_2 关联的边将会形成双链.在室温下加入 Eco RI 酶,水解所有的双链并进行清洗,这样表面上剩下的就都是与 e_2 互不关联的边.当延伸的边数大于 3 时,杂交反应有可能形成"发夹"结构,且加入的 Rev(e_i^1 - e_j^1 - TS)可能不止一个($i < j$ 且 e_i 为与 e_j 关联的所有边).

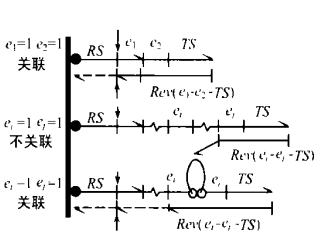

图 5　删除过程示意图(箭头处为酶切位点)

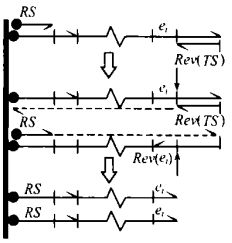

图 6　扩增过程示意图(箭头处为酶切位点)

(5)扩增(AMPLIFY)　因为在步骤(4)中删除了所有的不可行解,所以需要将载体表面剩余的 DNA 分子进行扩增.加入引物 biotin-RS 和 Rev(*TS*)、dNTP、以及 Taq DNA 聚合酶进行 20 个循环的 PCR 扩增反应.PCR 反应完成后,在室温下加入 Bam HI 酶,水解掉 DNA 分子双链末端的尾序列(TS)并将其清洗干净;然后,加热到 94 ℃进行解链,并再次清洗载体表

面.然后对剩余的边重复步骤(3) (4) (5).

(6)读结果(READOUT) 当在最后一次 PCR 扩增完成后,表面上就会生成图 1 中的所有匹配(除一个空串 00000000 外).在室温下加入 Eco RI 酶,将所得结果从表面上分离出来,并清洗多次;然后对其进行凝胶电泳,分离出其中最长的链(15 ×3 + 8 ×5 + 20 = 105bp)并多次纯化.显然,这些分子应该包括图 1 中的二个最大匹配.

因为最大匹配不止一个,为了区分出这二个最大匹配,我们可以用 e_i^0 和 Rev(*TS*)为引物,对长度为 105bp 的 DNA 分子进行 PCR 扩增反应.由于 e_i^0 为相同序列,我们就会得到以 e_i^0 开始结尾 *TS* 的各种长度的 DNA 分子.然后进行凝胶电泳,最后根据长度可以得出所有可能为 1 的位置.

在实际问题中,最大匹配的个数一般不会很大.我们可以根据图中各边之间的相互关联关系,判断其是否可能在一个匹配 *M* 中,最后就可以得出结果.对于图 1(*a*),显然 e_1 与 e_2 和 e_8、e_2 与 e_3 不可能在一个匹配中;其次,e_6 与 e_1、e_2、e_3 和 e_8 均不关联,所以可以视为公用;结果如表 1 所示:

表 1

位置	1	2	3	4	5	6	7	8
有 1 的位	1	1	1			1		1
M_1	1		1			1		
M_2		1				1		1

4 解的构造过程分析

我们在表 2、表 3 中用二进制字符串来表示图 1(*a*)和 1(*b*)的解空间的生成过程,左边带方框的为将要删除的串,右边黑体为最终所得最大匹配串,“ * ”表示还未生成的位.首先,在生成解空间的过程中时,每向后延伸一位,都要检查前面的位是否与其关联,如果关联,则删除掉;否则保留,用来进一部构造解空间.这样通过随时删除所生成的不可行解,大大减小了表面上的解空间.对于图 1 这样的简单问题,其最终生成的解空间仅为 24 个(原解空间为 2^8).

我们现在设有 *m* 位,则需要 *m* - 1 次循环操作,显然,第一次循环只能删除 1 个,而它代表的总数位 2^{m-2},再假说以后每次循环将会删除 2 个不可行解,则其删除的总数将为

$$S = 2^{m-2} + 2^{m-2} + 2^{m-3} + \Lambda 2^2 + 2 = 2^{m-1} + 2^{m-2} - 2 \quad (1)$$

$$\text{当 } M \to \infty \text{ 时}, S/2^m = 3/4. \quad (2)$$

其次,我们可以看出图 1(*a*)、1(*b*)唯一区别只是边的表示顺序不同.虽然其最终解空间都为 24 个,但是在同一计算步它们所生成的解的数量却不一样.显然,图 1(*a*)要优于 1(*b*),其主要原因是如果一个不可行解串删除的越早,其所代表的不可行解的数目越大.而删除的数量主要取决于当前边与其前面的边关联的数量,因此,在边的排列顺序上应尽可能使得与当前边所关联的边数最多,如何使得边的排列顺序最优,还有待于尽一步研究.虽然这样不能减少最终生成的解空间的数量,但是却能降低生物计算过程的难度.

表 2(图 1(*a*))

循环	延伸反应所得结果	总数	删除不可行解所得结果	总数
0	1******* 0*******	2		0
1	[11******] 10****** 01****** 00******	4	10****** 01****** 00******	3
2	101***** 100***** 010***** [011*****] 001***** 000*****	6	101***** 100***** 010***** 001***** 000*****	5
3	[1011****] 1010**** [1001****] 1000**** [0101****] 0100**** [0011****] 0010**** 0001**** 0000****	10	1010**** 1000**** 0100**** 0010**** 0001**** 0000****	6
4	[10101***] 10100*** 10001*** 10000*** 01001*** 01000*** [00101***] 00100*** 00011*** 00010*** 00001*** 00000***	12	10100*** 10001*** 10000*** 01001*** 01000*** 00100*** 00011*** 00010*** 00001*** 00000***	10
5	101001** 101000** [100011**] 100010** 100001** 100000** [010011**] 010010** 010001** 010000** 001001** 001000** [000111**] 000110** 000101** 000100** [000011**] 000010** 000001** 000000**	20	101001** 101000** 100010** 100001** 100000** 010010** 010001** 010000** 001001** 001000** 000110** 000101** 000100** 000010** 000001** 000000**	16
6	[1010011*] 1010010* [1010001*] 1010000* [1000101*] 1000100* [1000011*] 1000010* 1000001* 1000000* [0100101*] 0100100* [0100011*] 0100010* 0100001* 0100000* [0010011*] 0010010* [0010001*] 0010000* [0001101*] 0001100* [0001011*] 0001010* 0001001* 0001000* [0000101*] 0000100* [0000011*] 0000010* 0000001* 0000000*	32	1010010* 1010000* 1000100* 1000010* 1000001* 1000000* 0100100* 0100010* 0100001* 0100000* 0010010* 0010000* 0001100* 0001010* 0001001* 0001000* 0000100* 0000010* 0000001* 0000000*	20
7	[10100101] 10100100 [10100001] 10100000 [10001001] 10001000 10000101 10000100 [10000011] 10000010 10000001 10000000 [01001001] 01001000 01000101 01000100 [01000011] 01000010 [01000001] 01000000 [00100101] 00100100 [00100001] 00100000 [00011001] 00011000 [00010101] 00010100 [00010011] 00010010 [00010001] 00010000 [00001001] 00001000 00000101 00000100 [00000011] 00000010 00000001 00000000	40	10100100 10100000 10001000 10000100 10000010 10000000 01001000 01000101 01000100 01000010 01000001 01000000 00100100 00100000 00011000 00010100 00010010 00010000 00001000 00000101 00000100 00000010 00000001 00000000	24

表 3(图 1(*b*))

循环	延伸反应所得结果	总数	删除不可行解所得结果	总数
0	1******* 0*******	2		0
1	11****** 10****** 01****** 00******	4	10****** 01****** 00******	3
2	101***** 100***** 010***** 011***** 001***** 000*****	6	100***** 010***** 001***** 000*****	4
3	1011**** 1000**** 0101**** 0100**** 0011**** 0010**** 0001**** 0000****	8	0100**** 1000**** 0010**** 0011**** 0001**** 0000****	6
4	10001*** 10000*** 01001*** 01000*** 00101*** 00100*** 00111*** 00110*** 00011*** 00010*** 00001*** 00000***	12	01000*** 00110*** 10000*** 00101*** 00100*** 00010*** 00001*** 00000***	8
5	010001** 010000** 001101** 001100** 100001** 100000** 001011** 001010** 001001** 001000** 000101** 000100** 000011** 000010** 000001** 000000**	16	010000** 001100** 100001** 100000** 001010** 001000** 000101** 000100** 000011** 000010** 000001** 000000**	12
6	0100001* 0100000* 0011001* 0011000* 1100001* 1000010* 1000001* 1000000* 0010101* 0010100* 0100001* 0010000* 0001011* 0001010* 0001000* 0001000* 0000111* 0000110* 0000101* 0000100* 0000011* 0000010* 0000001* 0000000*	24	0100001* 0100000* 0011000* 1000010* 1000000* 0010100* 0010000* 0001010* 0001001* 0001000* 0000110* 0000101* 0000100* 0000010* 0000001* 0000000*	16
7	01000011 01000010 01000001 01000000 00110001 00110000 10000101 10000100 10000001 10000000 00101001 00101000 01001001 00100000 00010101 01000100 00010011 00010010 00010001 00010000 00001101 00001100 00001011 00001010 00001001 00001000 00000101 00000100 00000011 00000010 00000001 00000000	32	01000011 01000010 01000001 0100000 00110000 10000101 10000100 10000001 10000000 00101000 00100001 00100000 00010100 00010010 00010000 00001100 00001010 00001000 00000101 00000100 00000011 00000010 00000001 00000000	24

5 结束语

本文给出了一个基于表面方式的最大匹配问题的 DNA 计算模型.其主要思想是通过在表面上逐步生成解空间的过程中,随时删除所生成的不可行解;另外,我们发现对于同一个图,在生成解空间的过程中对边的顺序进行适当的的编排,也可以减少在表面上不可行解生成的数量;最后,通过合适的编码,用 PCR 和电泳技术可以将解集中的不同解逐步分辨出来.

当然,该模型适用的问题规模还是有限的,因为 DNA 计算所依赖的生物技术所能处理的 DNA 分子数量及长度均是有限的.如目前 PCR 技术所允许的 DNA 分子的最大长度为 10000bp,而表面上所允许最大长度为 5000bp,用一条 DNA 分子代表一个可能解,其最终数量是有限的.所以,基于表面方式的 DNA 计算的研究,一方面要借助于现代生物技术的发展;另一方面,在计算模型上还有待更深入的研究.

参考文献:

[1] L Adleman. Molecular computation of solution to combinatorial problems [J]. Science ,1994 ,266 (11) :1021 - 1024.

[2] R J Lipton. DNA solution of hard computational problems [J]. Science , 1995 ,268(4) :542 - 545.

[3] Q Ouyang ,et al. DNA solution of the maximal clique problem [J]. Science ,1997 ,278(17) :446 - 449.

[4] T Head ,et al. Computing with DNA by operation on plasmids [J]. Biosystems 2000 ,57 :87 - 93.

[5] M Smith et al. A surface-based approach to DNA computation [J]. Journal of Computational biology ,1998 ,5 :255 - 267.

[6] Q Liu ,et al. DNA computing on surfaces[J]. Nature ,2000 ,403 :175 - 179.

[7] H Wu. An improved surface-based method for DNA computation [J]. Boisystem ,2001 ,59 :1 - 5.

[8] J A Bondy , USR Murty. Graph Theory with Application [M]. the Macmillan Press LTD. London :Basingtoke and New York ,1976.

[9] 马立人,蒋中华. 生物芯片[M]. 北京,化学工业出版社,1999.

作者简介:

刘文斌　男,1969 年生于陕西,1993 获华北电力学院工学学士学位,1996 年获东南大学工学硕士学位,现为山东科技科技大学信息科学与工程学院副教授,目前在华中科技大学系统工程研究所攻读博士学位,感兴趣的研究领域为 DNA 分子生物计算、神经网络、遗传算法及其在组合优化问题中的应用. E-mail :wbliu69 @sohu. com

高　琳　女,1964 年生于陕西省西安市,1987 获西安交通大学数学系理学学士学位,1990 年获西安电子科技大学数学系硕士学位,现为西安电子科技大学计算机学院副教授,该校雷达信号处理国家重点实验室博士生,感兴趣的研究领域为 DNA 分子生物计算、神经网络、遗传算法及其在组合优化问题中的应用.