最大集团问题的 DNA 计算机进化算法

李 源 方 辰 欧阳颀*

(北京大学理论生物中心, 北京大学物理系, 北京 100871. * 联系人, E-mail: qi@pku.edu.cn)

摘要 进化算法是克服 DNA 计算中穷举法极限的可能途径之一. 借用生物进化的概念, 设计了可用于 DNA 计算的进化算法来求解最大集团问题. 算法中所有的操作都可以在今天的分子生物技术水平上实现. 计算机模拟实验表明使用这种进化算法有可能由一个小的样本空间得到问题的解, 而不必穷举所有可能情况. 对于随机生成的问题, 这种进化算法能以高概率在很少的进化循环数内正确地给出问题的解. 结果显示这种进化算法所需的时间随问题的规模呈多项式增长, 这可能使 DNA 计算机在求解复杂问题时比传统电子计算机拥有更多的优势.

关键词 **DNA** 计算机 进化算法 **NP** 完全问题

近年来, 关于分子生物计算进展的报道频繁的出现(参见文献[1]以及里面的引文). 自从Adleman用DNA计算机解决Hamilton路径问题[2]以来, RNA或DNA计算机对其他NP完全问题, 如最大集团问题[3]、3-SAT问题[4]和骑士周游问题[5]的求解也相继被完成. 此外, 用于计算的分子生物学实验技术也取得了一系列重要进展[6~11]. 由于分子计算操作的高并行度与高密度, DNA计算机有可能在随问题规模多项式增长的时间内解决NP完全型的计算问题, 而传统的Turing机在解决这类问题时需要随问题规模指数增长的时间[12]. 然而, 目前所有的DNA计算的策略都是基于穷举的思想, 即先在样本中包含所有可能的解, 再通过筛选去掉非解部分. 这种算法要求初始样本数目随问题的规模指数增长, 从而限制了DNA计算机的计算容量. 以DNA计算机用穷举法解决最大集团问题为例, 记N为问题的节点数, 所需DNA分子的数目至少为 2^N. 以微摩尔量级的DNA浓度计算, 一个 35 节点的问题可以在一个普通的试管中解决, 然而一个 75 节点的问题则需要一个游泳池的容量. 为了克服穷举算法在这方面遇到的障碍, 我们需要为DNA计算机寻找新的算法.

1 基本原理介绍

一种克服存储容量限制的思路是在DNA计算中引入达尔文的进化思想. 大多数物种通过两种机制完成进化: 自然选择和有性繁殖. 第 1 种机制决定群体中哪一部分留下来继续繁衍; 第 2 种机制是通过杂交重组的繁殖过程保证群体的多样性. 进化算法的基本思想是在题目设定的进化压下对样本进行反复的选择—变异—扩增的循环操作. 这样可以将搜索的重点放在解空间的优势区域. 因为不断的复制和杂交会使样本中有越来越多的可能解处于这些区域, 有优势的样本将有更大的机会作为父代进行复制和杂交, 子代中自然也有更多的来自优势父代的样本[13]. 我们注意到DNA计算机高度并行的计算方式非常适合进化算法的应用. 在这篇文章里, 我们将给出关于使用进化算法解决最大集团(Maximum clique)问题的模拟实验结果.

集团(clique)是指所有节点两两之间都有边相连的节点的集合. 最大集团问题: 给定了一个有N个节点M条边的图, 找出其中含有最多节点的集团的节点数目. 在图 1(a)中给出了一个由 5 个节点 5 条边的问题. 显然, (2, 3, 4)这 3 个节点构成了最大的集团, 因此这个网络中的最大集团大小是 3. 最大集团问题已被证实是一个NP完全问题. 除了传统的在电子计算机和DNA计算机上的算法外, 人们还作了一些有趣的尝试[15].

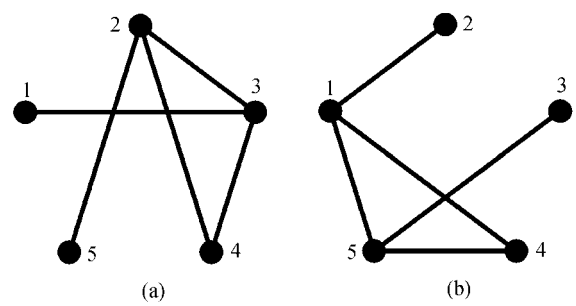

图 1 描述一个 5 节点、5 条边的最大集团问题的原图(a)以及它的补图(b)

我们的结果表明只用一个相对十分小的样本空间而不是穷举所有可能解是可行的. 对随机生成的问题, 进化算法能以大概率在不多的进化循环周期内正确地给出问题的解. 尽管与电子计算机相比, 当前的 DNA 计算机速度还比较慢, 但我们的模拟结果显示进化算法所需的循环周期数仅仅是问题节点数的线性函数. 这种特点可能会使 DNA 计算机在求解十分的复杂问题时有广阔的前景.

2 DNA 计算机的进化算法

我们所用来求解问题的数据结构与前人的工作一致[3]. 对于一个有N个节点的图, 用一个N位二进制数来表示一个可能的集团. 在这个二进制数中, 某一位为 1 表示相应的节点在集团中, 为 0 则表示不在集团中. 例如, 二进制数(01110)代表图 1(a)中的最大集团(2, 3, 4). 这样, 解决问题的任务就变成是在给定图对应的二进制数字串集合中寻找含有最多的为 1 的数. 对一个N节点的图, 我们可以构造它的补图. 补图包含所有节点以及所有在原图中缺少的边, 如图 1(b). 显然, 在原图中任意可能存在于一个集团中的两个节点在补图中都是不相连的, 即在补图中相连的两个节点对应的二进制位不能同时为 1(称之为条件©). 如果补图是由若干强连通子图构成, 可以就每个子图对应原图中的部分求解相应的部分最大集团问题, 而这些子图各自的最大集团的并集就是整个问题的最大集团. 从这个意义上说, 我们只需求解补图为强连通图的最大集团问题.

我们设计如下的进化算法来在 DNA 计算机上求解最大集团问题, 并且特别注意到算法中各个过程在当前生物技术水平上的可行性.

计算从空网开始, 对应的二进位数的每一位都是 0. 然后我们让样本群体一代代演化. 在每一代中, 首先使各个二进制数的每一位以某一个概率置 1, 不论该位原来是 0 或者 1. 我们称这种操作为变异. 然后我们从样品群体中删除所有不满足条件©的数. 例如, 在图 1(a)和 1(b)表示的问题中, 形如(11xxx)、(1xx1x)、(1xxx1)、(xx1x1)和(xxx11)的二进制数将被从样本群体中删去, 这里 x 代表 0 或 1 的任意一个. 如果样本群体中含有的二进制数在删除操作后变少了, 我们复制这些二进制数直到样本群体的大小达到初始的大小左右, 这称为扩增. 通过变异和删除, 样品群体中的集团将随着演化的进行越来越大.

为了检查在每一代中是否有进展, 在每次删除操作后, 我们将在二进制数串集合中找出含有的最多 1 的数串. 这代表了当前所得到的最大集团的大小. 如果这个数串中 1 的个数随着演化的进行不断增大, 则最大集团还没有被找到, 计算将继续进行下去. 如果它连续很多代都停滞不前, 我们则认为这表示了演化到了尽头, 计算将中止并且最后的集团大小被视作这次计算所给出的问题答案.

上面所说的所有计算步骤都可以在当今生物技术中找到相应实现的方法. 在下面的讨论中, 我们将介绍实现我们算法所需的生物技术.

首先, 我们需要将二进制数编码到DNA链中, 一个单链代表一个二进制数. 平行重叠合成(parallel overlap assembly)技术可以直接应用到DNA数据库建造上[16]. 为了编码一个N位二进制数, 我们需要N+1 种不同的DNA片段序列代表N+1 个位置和N个序列片段代表每一位的值. 代表位置的序列的长度都是一样的, 但代表不同 0、1 数值的序列是不一样长的, 比如代表 1 的序列要比代表 0 的短.

为了实现删除操作, 目前至少有两种可行的生物技术. 令代表变量值为 1 的数值序列的长度为零, 则在DNA链中每逢有变量取值为 1 的地方, 最近的两个位置序列就会直接相邻. 设计连接点的两端各为DNA内切酶位点的各一半, 则其连接处都恰好是一个酶切位点. 这样就可以用内切酶破坏特定位置上有 1 的DNA链. 如果希望破坏所有在位置a和位置b同时取值为 1 的DNA链, 可以将样本分为A, B两份, 将A中的位置a处为 1 的DNA链全部破坏, B中位置b处为 1 的全部破坏, 这样A, B剩下的部分合并后将不再含有a, b处同时为 1 的DNA链, 但只有其中一位为 1 的DNA链仍可以存活下来(图 2(a))[3]. 在另一种技术中, 可令代表变量值为 0 的数值序列的长度为零. 这样每当 0 出现的地方, 最近的两个位置序列就直接相邻. 为了删除不符合要求的DNA链, 我们只需选出并保留符合要求的DNA. 这可以通过在电泳中用原位杂交技术捕捉特定位置序列的方法完成(图 2(b))[4].

变异操作可以通过如下步骤完成. 如果值为 0 的数值序列长度为 0, 我们可以利用内切酶将 DNA 从该位置切断, 插入位置取值为 1 的数值序列, 再通过连接酶使样本中的 DNA 链重组, 这样取值为 1 的数值序列就有可能插入到原来切断的位置上, 从而完

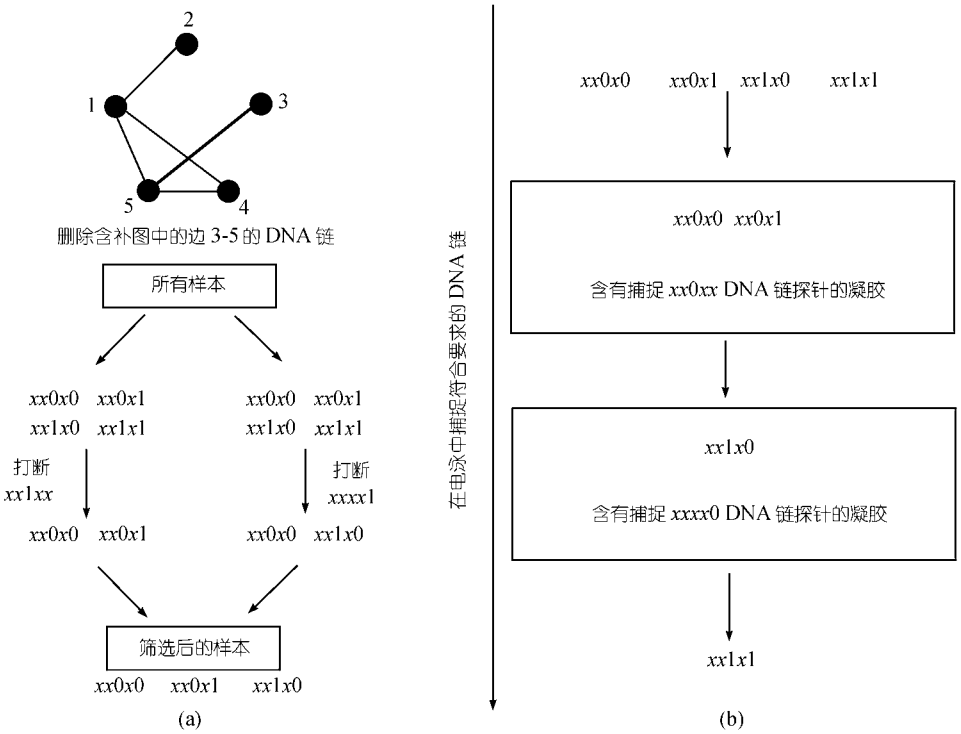

图 2　可用于删除样本群体中 DNA 的两种不同方法

(a)用内切酶破坏待删除的 DNA, (b)用探针捕捉电泳中满足要求的 DNA

成 0 到 1 的变异操作. 我们的模拟表明, 算法所需的变异率不超过 5%, 因而这种技术对我们的算法应该已经足够. 克隆或者 PCR 可以完成扩增的操作. 由于取值为 1 和 0 的数值序列长度不同, 通过电泳可以得到关于其中集团大小的信息.

3　模拟实验结果

为了评估这个算法, 我们随机生成一些不同规模的最大集团问题, 通过对这些问题的求解研究算法的时间代价与问题规模的关系. 问题是用 80%的连接率产生的, 即每一对节点都有 80%的概率被一条边连接起来. 在模拟中, 样本群的个数固定为 10^6. 执行变异操作的时候, 把某一位置变为 1 的概率(即变异率)为 $2/N$, 这是为了让集团以大约每代两个节点的速率增大. 如果连续 30 代集团的大小都没有变化, 计算就会停止. 为了检验计算的结果并与传统算法进行关于时间代价的比较, 我们还用了穷举的递归算法求解问题. 表 1 里列出了我们的一些计算结果. 可以看到, 进化算法可以正确地给出结果, 所需的循环数大致上是问题节点数目的线性函数(见图 3(a)). 当连接率固定不变时, 补图中的边的数目大致正比于N^2. 在每一代中, 补图里的一条边对应一次删除的操作, 如果所需循环数随节点个数N线性增长, 则这个进化算法总的时间代价是N^3级的. 另一方面, 递归算法的总时间代价是N的指数函数, 如图 3(b)所示. 尤其重要的是相对于问题所有可能解的数目, 使用进化算法只需要使用很小的样本群体. 一个 40 节点问题的所有可能解的数目是 2^{40}, 即 10^{12}, 而 100 节点问题则是 2^{100}, 即 10^{30}, 而在样本群体的大小只有 10^6 时, 依然在绝大多数场合下可以正确地求解问题(见表 1). 因此我们认为这种DNA算法克服了DNA计算中穷举算法所遇到的存储空间障碍[2~5].

问题的难度不但取决于网络中节点的数目, 还取决于节点间的连接率. 从表 1 中看出在 80%的连接率下, 最大集团的大小随总节点数目的增长很缓慢. 这表明对于随机生成的问题, 80%的连接率不足以使网络中出现很大的集团. 因此, 我们研究了在固定节点数的情况下(40 节点和 70 节点)解题时间随连接率的变化. 计算的主要结果被列在表 2 中. 可以看到当连接率接近 100%时最大集团的大小迅速增长. 随着边的数目的增多, 传统递归算法的时间代价指数增

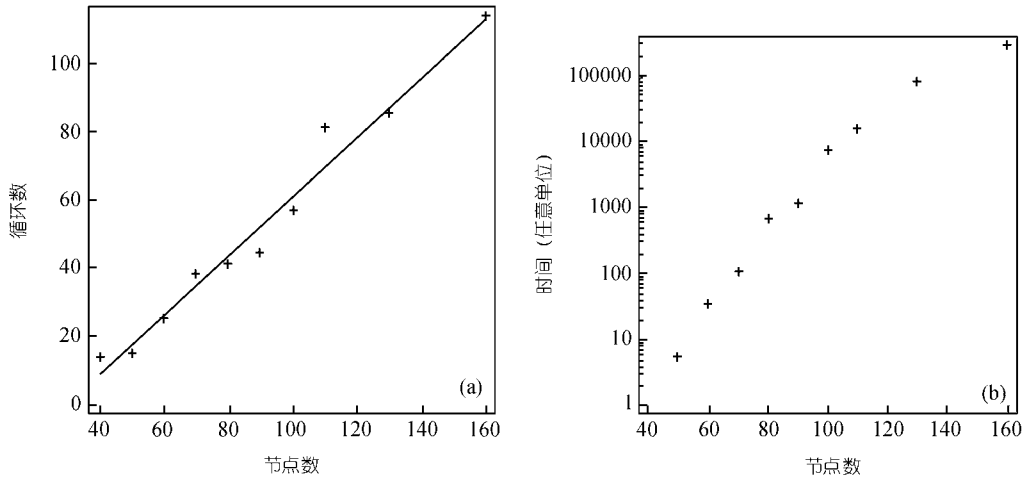

图 3 进化算法(a)和传统递归算法(b)求解不同规模问题的相对时间代价(任意的时间单位)

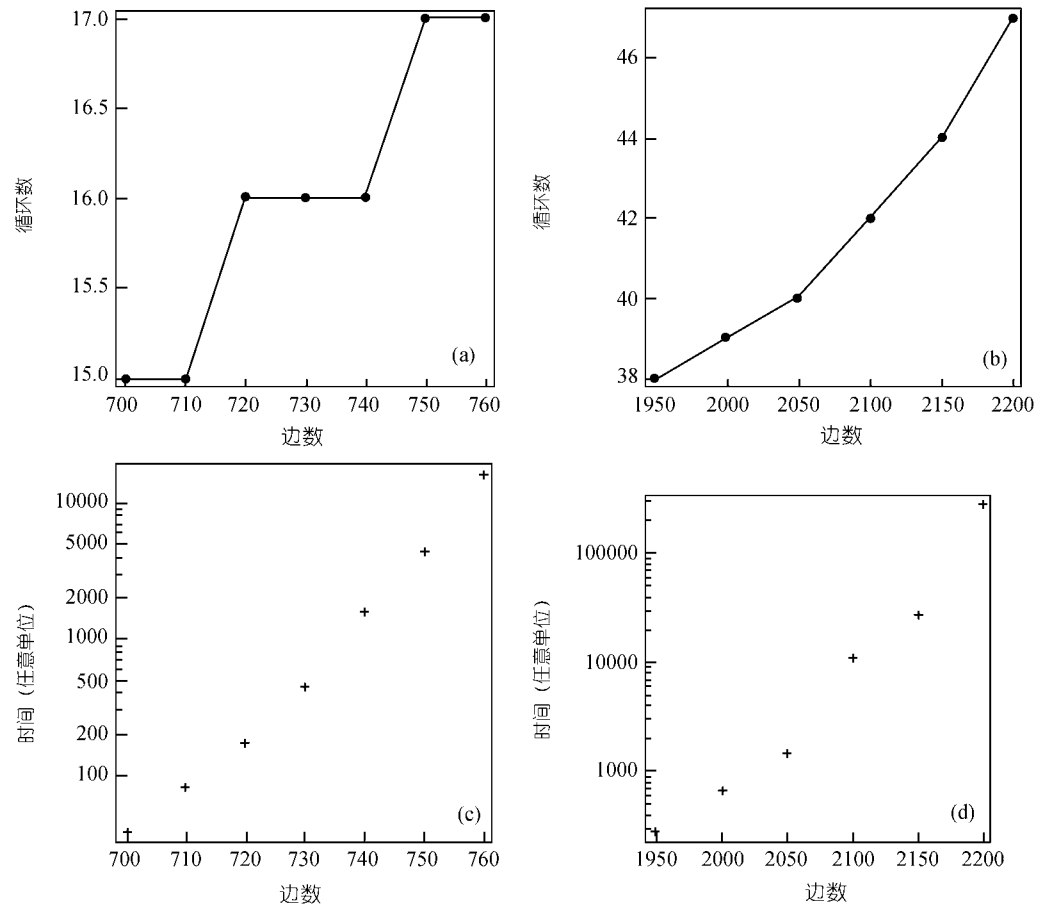

图 4 求解含有不同数量的边的问题时进化算法和递归算法的相对时间代价的比较

进化算法所需代数随问题的边的数目的增加几乎不变(a)(b), 递归算法的时间代价随边的数目的增加指数增长(c)(d)

表 1

节点数(N)	边数	最大集团大小	递归算法的时间代价	进化算法得到正确解的概率	进化算法的平均所需循环数
40	624	14	1	100% (6/6)	14
50	980	14	5.7	100% (6/6)	15
60	1420	16	34.2	100% (6/6)	25
70	1932	18	108	100% (6/6)	38
80	2530	19	703	100% (6/6)	41
90	3200	19	1130	100% (6/6)	44
100	4000	20	7440	100% (6/6)	57
110	4800	21	15800	100% (6/6)	81
130	6700	21	80800	100% (6/6)	85
160	10000	22	281000	83% (5/6)	113

表 2 计算主要结果

节点数(N)	边数	最大集团大小	递归算法的时间代价(任意单位)	进化算法所需的代数
40(可能存在的总边数: 780)	700	19	38	15
	710	20	80	15
	720	21	173	16
	730	22	440	16
	740	25	1570	16
	750	26	4250	17
	760	27	15700	17
70(可能存在的总边数: 2415)	1950	18	268	38
	2000	20	661	39
	2050	20	1430	40
	2100	23	11300	42
	2150	24	26800	44
	2200	27	283000	47

长, 如图 4(c)和 4(d)所示. 令人惊讶的是, 进化算法求解这种问题所需的代数并没有太多变化, 如图 4(a)和 4(b)所示. 在我们的计算中, 由于补图中的边的数目随着原图中边数目的增加而减少, 这将减少每一代中的删除操作数, 因此实际的总时间代价可能变得更少.

4 讨论

DNA 计算机是针对解决传统图灵机难以解决的 NP 问题而设计的有效方法. 但是, 目前文献中讨论的 DNA 计算机算法都是基于穷举的思想, 其实质是将 NP 问题的时间代价转化为储存器的空间代价. 这种算法使 DNA 计算机在应用上受到极大的限制. 本文的目的是将分子进化的思想引入 DNA 计算机算法. 我们的模拟实验表明, 只用一个相对十分小的样本空间而不是穷举所有可能解是可行的. 进化算法所需的循环周期数仅仅是问题节点数的线性函数. 这种特点可能会使 DNA 计算机在求解十分的复杂问题时有广阔的前景.

致谢 感谢钱敏平教授启发性的讨论, 并感谢徐芦平提供的关于进化算法所要用到的生物技术的细致介绍. 本工作为国家“八六三”计划(批准号: 2002AA221011)以及“*著* 政”科研基金资助项目.

参 考 文 献

1 Ruben A J, Landweber L F. The past, present and future of molecular computing. Nature Reviews Molecular Cell Biology, 2000, 1: 69~72 [DOI]
2 Adleman L. Molecular computation of solutions to combinatorial problems. Science, 1994, 266: 1021~1024
3 Ouyang Q, Kaplan P D, Liu S, et al. DNA solution of the maximal clique problem. Science, 1997, 278: 446~449 [DOI]
4 Braich R S, Chelyapov N, Johnson C, et al. Solution of a 20-variable 3-SAT problem on a DNA computer. Science, 2002, 296: 499~502 [DOI]
5 Faulhammer D, Cukras A R, Lipton R J, et al. Molecular computation: RNA solutions to chess problems. Proc Natl Acad Sci USA, 2000, 97: 1385~1389 [DOI]
6 Benenson Y, Paz-Elizur T, Adar R, et al. Programmable and autonomous computing machine made of biomolecules. Nature, 2001, 414: 430~434 [DOI]
7 Liu Q, Wang L, Frutos A G, et al. DNA computing on surfaces. Nature, 2000, 403: 175~179 [DOI]
8 Ogihara M, Ray A. DNA computing on a chip. Nature, 2000, 403: 143~144 [DOI]
9 Sakamoto K, Gouzu H, Komiya K, et al. Molecular computation by DNA hairpin formation. Science, 2000, 288: 1223~1226 [DOI]
10 Wang L, Hall J G, Lu M, et al. A DNA computing readout operation based on structure-specific cleavage. Nat Biotechnol, 2001, 19: 1053~1059 [DOI]
11 Zimmermann Karl-Heinz. On applying molecular computation to binary linear codes. IEEE Trans Inform Theory, 2002, 48: 505~510 [DOI]
12 Impagliazzo R, Paturi R, Zane F. Which problems have strongly exponential complexity? J Comput Syst Sci, 2001, 23: 512~ 530 [DOI]
13 Holland J H. Generic algorithm. Scientific American, 1992, July: 66~72
14 Foster J A. Evolutionary computation. Nat Rev Genet, 2001, 2: 428~436 [DOI]
15 Chiu D T, Pezzoli E, Wu H, et al. Using three-dimensional microfluidic networks for solving computationally hard problems. Proc Natl Acad Sci USA, 2001, 98: 2961~2966 [DOI]
16 Kaplan P D, Ouyang Q, Thaler D S, et al. Parallel overlap assembly for the construction of computational DNA libraries. J Theor Biol, 1997, 188: 333~341 [DOI]

(2003-09-26 收稿, 2004-01-14 收修改稿)