Motif Finding 问题的分布式参数算法

张祖平，王 丽

(中南大学 信息科学与工程学院，湖南 长沙，410083)

摘 要：基于从 DNA 序列形成 *k* 分图的图理论算法和查找 *k*-clique 的理论算法，设计与实现了对 Motif Finding 问题求解的分布式参数算法。该算法的主要特点是：采用新的 1−3 树分枝算法并实现分布式计算机制，即任务可以随着计算过程的展开在每一阶段不断地分解并分布到不确定数量的申请参与计算的客户机上，服务器端负责任务均衡与结果整合。实验结果表明：分布式参数算法充分利用多台机器协同计算，能够正确、高效地得到计算结果，为求解生物计算中难解的 Motif Finding 问题提供了有效的解决手段。

关键词：Motif；*k* 分图；团；生物计算；分布式系统

中图分类号：TP311 **文献标识码**：A **文章编号**：1672-7207(2007)05−0943−07

Distributed parameterized algorithm for Motif Finding

ZHANG Zu-ping, WANG Li

(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: Based on the graph-theoretic algorithm, which is used to generate *k*-partite graph from DNA sequences, and a theoretic algorithm which is used to search *k*-clique in *k*-partite graph, a new distributed parameterized algorithm is designed and implemented to solve Motif Finding problem. The main characteristics of this algorithm are the introduction of 1−3 tree branching algorithm and the implementation of distributed mechanism, i.e., task can be constantly decomposed into several tasks as program proceeding at each phase, and then these tasks are distributed to several client computers which is applied for participation of calculation. On the server, the tasks are distributed and balanced as well as the results are received and combined. Experimental results show that this distributed parameterized algorithm is correct and efficient since it makes full use of computing capability of many computers, and provides an effective approach to solving the difficult problems in biocomputing such as Motif Finding.

Key words: Motif; *k*-partite graph; clique; biocomputing; distributed system

生物信息学是新兴的交叉学科，它综合运用数学、计算机科学和生物学的各种工具，来阐明和理解大量数据所包含的生物学意义[1]。其中，对基因突变热点区的检测[2]是当前研究的热点，而在扩展的多重序列比对方面还处于试验探索阶段。已经证明 Motif Finding 是 NP 难问题[3]，它是生物计算中较早被深入研究的问题之一，也是近年来研究的热点问题[4−6]，虽然已经出现一些优秀的软件工具，如：基于贪婪算法的 CONSENSUS[7]，基于 Gibbs 取样的 GibbsDNA[8] 和基于 EM 算法的 MEME[9]等，但是，这些算法并不能确保一定能找到确实存在的 Motif，因为它们都需要一个很好的开始点，所以，都是不确定算法；另外，基于详尽的列举所有可能的 Motifs 算法，虽然能保证找到最优的 Motif，但是，运行时间随着 Motif 长度 *l*

收稿日期：2006−12−15；**修回日期**：2007−01−08

基金项目：国家自然科学基金资助项目(60433020)

作者简介：张祖平(1966−)，男，湖南湘乡人，博士，教授，从事参数计算与应用及大型数据库技术研究

通信作者：张祖平，男，教授；电话：0731-8877936；E-mail: zpzhang@mail.csu.edu.cn

的增加而呈指数增长，在 l 较大的挑战问题中变得不可行。为解决上述问题，本文作者一方面引进参数算法求解 Motif Finding 问题的所有精确解，有效解决非确定算法遗漏结果的问题；另一方面，引进分布式计算方法与分布式求解系统[10]，将计算任务划分成较小的可以独立执行的单元，并分布到网络上有空闲计算资源的多台机器中去，系统将各机器返回的计算结果进行合并处理后形成问题的最终结果，从而加快了 Motif Finding 问题的求解过程。

1 关键算法设计与分析

1.1 问题分析

依照 Pevzner 的(l, d)-motif 模型[11]随机产生 k 个长度均为 n 的随机序列，在每个序列中随机挑选 1 个位置置入 1 个(l, d)-motif，该 Motif 基于 1 个长为 l 允许有 d 个突变的模式 P 产生。Motif Finding 问题要求在不知道模式 P 和置入位置的情况下找出这些置入的 Motif。

根据文献[12]，算法分为 2 步：第 1 步，应用 Pevzner 的图理论算法[11]从 DNA 序列产生 k 分图：在每个序列 S_i 中的第 j 个位置，建立 1 个顶点 S_{ij}，表示 1 个从位置 $j(1\leqslant j\leqslant n-l+1)$开始的长为 l 的子串，若 $i\neq p$ 并且 S_{ij} 和 S_{pq} 之间的距离不超过 $2d$(突变元素的数量)，则用 1 条边将 2 个顶点 S_{ij} 和 S_{pq} 相连；第 2 步，参考查找 k-clique 的理论算法[12]，应用分治方法从 k 分图中求解 k-clique：将得到的 k 分图细分为 $k'(k'<k)$ 分图直到 k'小到容易解决为止，求得 k'-clique 之后再合并为所求的 k-clique。求得的 k-clique 即为所求的 Motif 及其置入位置。

1.2 k 分图形成算法

1.2.1 算法思想

根据图理论，算法形成 k 分图的关键在于判断 2 个顶点间是否存在边，即计算 2 个字符串之间的不匹配个数是否在允许范围之内，而图理论算法并未给出有效的方法。通过研究，已有算法如 PROJECTION[13]，WINNOWER，SP-score[11]及最新的判据搜索算法[14]等都只考虑了简单的替换突变，不能处理存在插入和删除突变的序列。为解决这个问题，本文从 Mismatch Tree[15]算法中得到启发，设计了 1－3 树分枝算法。因为这种算法在逻辑上是一个树的形式。其中，“1”代表匹配时只产生 1 个分枝，“3”代表不匹配时产生 3 个分枝。

当 2 个字符串 A 与 B 进行比较时，若在某一位置出现了不匹配，则该位置上的字符可能产生了替换、插入或删除突变。首先，对 3 种突变进行算法抽象。

a. 替换。无论 A 与 B 中任何 1 个产生了替换突变，甚至都产生了突变，其结论都是跳过 A 与 B 在该位置上的字符，并将不匹配个数计数器增加 1。

b. 插入和删除。将 2 个字符串进行比较，插入和删除突变都是相对于另一条字符串而言的。若在某个位置上 A 相对 B 发生了插入突变，则相当于 B 相对 A 发生了删除突变，这时，应该忽略该位置上 A 的字符，用 A 下一个位置上的字符继续与 B 的该位置上字符比较，同时不匹配个数计数器加 1。

1−3 树分枝算法为：

```
boolean done=false;    /* 结束标志 */
int mis=0;             /* 不匹配个数 */
int pos1=0;   /*字符串 A 将要进行比较的字符位置*/
int pos2=0;   /*字符串 B 将要进行比较字符的位置*/
while (true) {
    if (done) then return;
    if (pos1 到了字符串 A 的尽头 or pos2 到了字符串 B 的尽头 or mis>所允许的最大不匹配个数) then break;
    if (pos1 位置上的字符与 pos2 上的字符一样) then {
        pos1++;
        pos2++;
        continue;
    }
    else {
        mis++;
        产生假设是插入突变的线程，传值 pos1+1, pos2, mis;
        产生假设是删除突变的线程，传值 pos1, pos2+1, mis;
        假设是替换突变，pos1++, pos2++;
        continue;
    }
}
if (mis <= 允许的最大不匹配个数) then{
    if (done) then return;
    synchronize {
        if (done) then return;
        done = true;
        将这条边加入 k 分图;
    }
}else return;
```

为了更好地理解这个算法，下面给出比较字符串 actta…… 和 attag……时程序产生的线程树，其中 a, c, t, g 为 4 个碱基，如图 1 所示。

线程树是一个逻辑上的概念，实际上并不存在，可以看成是线程运行的轨迹。1 个分枝代表 1 个线程，每个线程都拥有自己的一些资料，如不匹配个数计数器，与 2 个字符串各自比较到了哪个字符等。

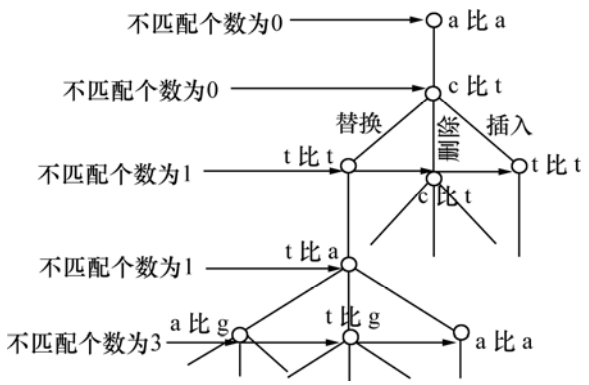

图 1　应用 1−3 树分枝算法比较 2 个字符串示例

Fig.1　Comparison between two sequences with 1−3 tree branching algorithm

1.2.2　算法分析

比较 2 个长为 l 的字符串，允许不匹配个数为 $2d$，该算法的时间复杂度在最好情况下为 l，最坏情况下为 3^{2d}，d 一般不大。在一条长为 n 的序列中，把其中长为 l 的字符串当做 k 分图中的 1 个顶点，则共有 $n-l+1$ 个顶点。

第 1 条序列中的顶点与其后的 k–1 个序列中的所有顶点进行比较，即进行$(k–1)(n–l+1)^2$次顶点比较；

第 2 条序列中的顶点与其后的 k–2 个序列中的所有顶点进行比较，即进行$(k–2)(n–l+1)^2$次顶点比较；

……

将倒数第 2 条序列即第 k–1 条序列中的顶点与最后那条序列中的所有顶点进行比较，即进行$(n–l+1)^2$次顶点比较；

最后，那条序列即第 k 条序列不再与任何序列进行比较。

所以，比较次数共为：

$$(k-1)(n-l+1)^2+(k-2)(n-l+1)^2+\cdots+(n-l+1)^2=$$
$$(n-l+1)^2[(k-1)+(k-2)+\cdots+1]=$$
$$(n-l+1)^2k(k-1)/2=$$
$$O(k^2(n-l)^2)。$$

因而，形成 k 分图在最好情况下的时间复杂度为：$O(lk^2(n–l)^2)$；最坏情况下的时间复杂度为：$O(3^{2d}k^2(n–l)^2)$。

由于该算法运行时产生的线程不确定，最好和最坏情况的发生概率均趋于 0，经实验验证，在 k, n 和 d 不大(k = 10, n = 82, d = 4)但要求精确解的情况下，实际计算可行。

1.3　*k*-clique 寻找算法

1.3.1　算法思想

查找 k-clique 的理论算法采用了从上至下的递归方法来实现分治策略，实际上更为有效的方式是采用自下而上的循环方法来实现分治，这样可以获得更高的效率，便于将算法进行分布式处理，并且只需将算法稍作修改便可求解更加复杂的最大团问题[12]。具体方法如下。

将 k 分图划分为一些 k'分图($k'<k$)，从 k'分图中求解 k'-clique 后再将 k'-clique 组合成 k''-clique($k'<k''<k$)，直至最后组合成 k-clique。若最后无法将 k''-clique 组合成 k-clique，则可以判定 k 分图至少存在大小为 k”的 clique，修改程序，利用这些中间结果可以方便地求出 k 分图的最大团。k'的选择应该使得子图足够稀疏以至于很容易解决。由于受存储结构的制约(存储的是边)，k'选择 4 最恰当。

当序列个数不是 4 的倍数时，若只剩 1 条序列，则把它的所有顶点看成 1-clique；若余下 2 条序列，则把这 2 条序列间的边看作 2-clique；若余下 3 条序列，则把 2 条序列间的 2-clique 和 1 条序列的 1-clique 结合得到 3-clique。

1.3.2　算法分析

当 k 为 2 的整数(>1)次幂时，时间复杂度最大。这里分析这种时间复杂度最大的情况。考虑最坏的情况，即 k 分图是完全图，这意味着每 2 个顶点间都有边，每条序列有 $n-l+1$ 个顶点，则每 2 条序列间有$(n–l+1)^2$条边，每 4 条序列中有$(n–l+1)^4$个 4-clique。

算法首先用 2 条序列间的所有边(即 2-clique)与另 2 条序列间的所有边进行比较，得到一组 4-clique；然后，用一组 4-clique 与另一组 4-clique 进行比较，得到一组 8-clique，……。直到最后比较 2 组 k/2-clique，检查是否存在 k-clique。

第 1 步：求出所有 4-clique，时间复杂度为：

$$(k/4)\cdot 2^2(n-l+1)^2(n-l+1)^2。$$

第 2 步：求出所有 8-clique，时间复杂度为：

$$(k/8)\cdot 4^2(n-l+1)^4(n-l+1)^4。$$

…

最后 1 步即第 $\log_2 k-1$ 步：求出所有 *k*-clique，时间复杂度为：

$$(k/2)^2(n-l+1)^{k/2}(n-l+1)^{k/2}。$$

推出通用公式为(*m* 代表第 *m* 步)，时间复杂度为：

$$(k/2^{m+1})\cdot 2^{2m}(n-l+1)^{2^{m+1}}=$$
$$k\cdot 2^{m-1}(n-l+1)^{2^{m+1}}。$$

总的时间复杂度为：

$$k[(n-l+1)^4+2(n-l+1)^8+2^2(n-l+1)^{16}+\cdots+$$
$$2^{\log_2 k-2}(n-l+1)^k]<$$
$$k(2^0+2^1+2^2+\cdots+2^{\log_2 k-2})(n-l+1)^k=$$
$$O(k^2(n-l)^k)。$$

实际计算的时间复杂度远小于此时间复杂度，并且由于是应用分布式系统并行求解，故运行时间在可承受范围之内。

2 关键数据结构设计

2.1 *k* 分图

k 分图共有 $k\cdot(n-l+1)$ 个顶点，为方便读取顶点信息，在存储文件的头部存储 1 个顶点索引。

k 分图的数据结构采用邻接表，即在 1 个顶点中存储另 1 个顶点的位置来表示 1 条边，在序号较小的顶点中存储所有与之相连的序号较大的顶点位置，即在顶点 S_{ij} 里存储数组(*p*, *q*)，其中 $i<p$。若顶点 S_{ij} 与 S_{pq} 间有边，则进一步将顶点按其属于哪个序列进行分类，只存储在此序列中的位置，即只存储 *q*，节省了一半空间。

顶点采用 Vector 数据结构。下面给出顶点(0, 3)的存储结构示例图。假设该顶点与(1, 2)，(1, 5)，(1, 6)，(2, 4)，(3, 8)，(3, 9)等顶点相连，则该顶点的数据结构如图 2 所示。

2.2 团

由于团的大小与序列个数是一样的，因此，用 1 个确定长度的数组来存储；团中的元素是顶点，用长度为 2 的数组表示。综上所述，团的数据结构是 1 个二维数组，第 1 维长度为计算的序列个数，第 2 维长度为 2。结果中团的个数不确定，因此，用 Vector 来存储这些团。

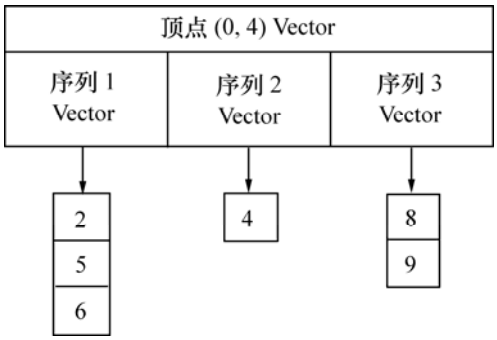

图 2 顶点 Vector 的数据结构

Fig.2 Data structure of vertex

3 算法分布式思想及其实现

参数算法虽然能够用于有效求解精确解，但是，时间复杂度仍然随问题规模呈指数增长，因此，为加快求解速度，引入分布式机制，将算法进行分布式处理，使之能在分布式平台上运行，并设计动态均衡模型，保证负载均衡。

分布式系统采用一个服务器控制多台客户机的模式，其中服务器只负责产生任务与接收客户机传回结果及整合，客户机对分给自己的任务进行计算。客户机端的程序思想如前所述，但计算任务由从服务器得到的任务参数决定，规模比总的任务小得多。我们下面介绍的分布式处理算法都是指服务器端程序采用的算法。

3.1 生成 *k* 分图

为了使分解出的任务规模相近，采用对序列进行垂直平分的方法，如图 3 所示，其中水平的粗线条代表 1 条 DNA 序列，前端的实线部分代表要比较的顶点，长为(*n*−*l*+1)。为了平均分割任务，这里将实线部分用垂直的虚线进行平分，划分出的每 1 截对应于比较时的一个顶点。

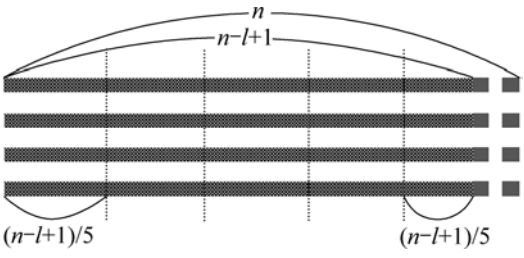

图 3 垂直平分 DNA 序列里的顶点比较任务

Fig.3 Vertically average division of comparing task

当 $n-l+1$ 不能被所分份数整除时，将余数平均分给前几个任务，保证大的任务数仅比小的任务数大 1，并且先产生先被取走执行，减少了对系统效率的影响。

将分解后的任务分配给客户机进行计算。以图 3 所示序列为例。假如 1 台客户机分到了第 2 个任务，那么，它将用第 1 个序列由虚线分成的第 2 个部分的顶点与第 2，3 和 4 个序列的全部顶点进行比较，然后，用第 2 个序列的第 2 部分顶点与第 3 和 4 个序列的全部顶点进行比较，最后，用第 3 个序列的第 2 部分顶点与第 4 个序列的全部顶点进行比较。比较完后把结果传回服务器，服务器将其与其他客户机传回结果进行整合，得到 k 分图。

整合操作是当服务器得到一个返回结果时，把结果文件中除了索引以外的部分全部拷贝(为防止内存溢出，采用依次拷贝结果文件定长的一部分策略)到一个指定的新文件里，并用结果文件的索引对新文件的索引进行修正。这个新文件在第 1 个结果返回时创建，并且在开始部分放入一个空的 long[$k-1$][$n-l+1$]数组用来索引全部顶点。

3.2　查找 *k*-clique

在这一步里，如何分解任务并使得任务的粒度合适是很关键的。因为经验表明，如果粒度过小，将使系统频繁建立连接传输文件，严重影响系统效率；若粒度过大，则无法充分利用分布式系统的资源。另外，对 k'-clique 的结合方法将影响系统负载均衡和计算资源的利用。这里介绍一个动态的均衡模型以解决这个问题。

为了充分利用分布式系统资源，在应用分治法将序列分成 4 个 1 组的同时，还将每个组的任务细分。具体做法如下。

以每组的第 1 个序列作为基准序列，从第 1 个顶点开始，把所有与这个顶点有关的计算任务放在一起，一直到最后 1 个顶点，最终形成 $n-l+1$ 个任务。有关的任务是指该组中与这个顶点的判定有关的顶点和边。具体来说，第 1 个任务是第 1 个序列的第 1 个顶点和第 2 个序列顶点间的所有边与第 3 和第 4 个序列间所有边的组合判定，直到第 $n-l+1$ 个任务为止。经测试，这样的粒度是合适的，若有 1 种合适的检测控制粒度的算法将更加有效地提高系统效率。

由于得到 k'-clique 后需要再将 k'-clique 组合成 k''-clique($k'<k''<k=$，则需要将结合 2 个 k'-clique 得到 k''-clique 作为新的判定任务并分配给客户机进一步计算。如果只是简单地返回了 2 个组就将其结合进入下一层循环，那么，必定使得下一层的后一组比前一组晚很多才能产生，这样，势必影响更下一层的结合，最终使系统效率下降。为了克服这个时间缺陷，设计了一个动态结合组的模型，为了便于理解，在介绍这个模型之前先介绍克服上述缺陷的静态模型。如图 4 所示，假设产生了奇数个分组并且各个组的任务其结果是按发出去的顺序依次返回的，其中有斜线填充的 2 个矩形代表同一个组，这个组是第 1 层中最后 1 个完成的(这里称从上到下的一排排矩形为第 1 层、第 2 层，……)，而又比第 2 层的所有任务先返回，那么，该组将移到第 2 层的最前面与其他组结合。这个模型能够克服时间缺陷的原因是，每层先完成的组在选择与其进行结合的组时，选择的是没有结合过的组中完成顺序居中的那个。

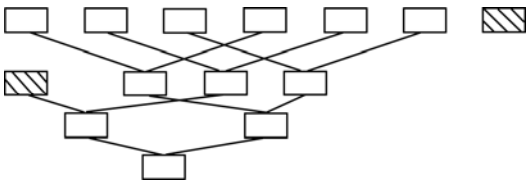

图 4　克服了时间缺陷的静态模型

Fig.4　Static combining model without time flaw

每个组分成 $n-l+1$ 个小任务，可能返回 0~ $n-l+1$ 个结果，若为 0，则程序结束运行，要寻找的 k-clique 不存在；若为 $n-l+1$，直接将这些结果与另外一组结果进行结合，则最多可能有$(n-l+1)^2$ 个任务；若继续结合下去，则任务数将呈指数增长，必须将一个组里的结果先进行合并，然后，再与另一组里的任务结合，这样，结合后的组里最多只有 $n-l+1$ 个任务。另外，不必等 2 个组的任务全部返回才开始结合，在其中一个组的任务全部返回合并后就可以与另一个只有部分返回结果的组进行结合。

在实现动态模型时，采用文件夹来存储任务的返回结果，文件夹可以动态创建与释放。下面用一个假设的例子来说明这个动态模型的工作过程。如图 5 所示，当 1 个组里的结果全部返回并且合并为 1 个时就用横线填充矩形来表示。完成顺序是否居中，用剩余任务(即已经产生但还没有结果返回的任务)的多少来衡量。假设一共分了 7 组，前 6 组都是 4 个序列 1 组，分解成小任务之后分发出去，最后，1 组只有 2 个序列，直接从最后 2 个序列里得到的所有边作为 2-clique 放入第 6 个文件夹，这时第 6 个文件夹已满，它去寻找能够与它进行结合的文件夹。但是，没有一个文件

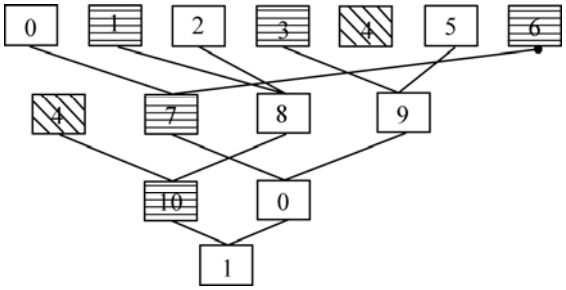

图 5 动态结合组模型

Fig.5 Dynamic combining model

夹里有返回的结果，所以，它填写注册信息，以期将来有某个文件夹发现它并与它结合。当第 0 号文件夹的第 1 个结果返回时系统扫描注册信息，发现第 6 号文件夹并与之结合，动态创建文件夹 7 来存放将来的返回结果。假设当第 1 号文件夹满时第 2，3 和 4 号文件夹分别还有 3，2 和 5 个任务没有返回，若第 5 号文件夹还是空的，则第 1 号文件夹就选择文件夹 2 进行结合，创建文件夹 8；当文件夹 3 满时，文件夹 5 的剩余任务数小于文件夹 4 的剩余任务数，则选择与文件夹 5 结合，生成文件夹 9；当文件夹 4 满时，由于它是本层的最后一个，无法与本层的文件夹结合，因此，将它的层数加 1，把它放入下一层与下一层的文件夹结合。根据图 5，它与文件夹 8 结合，生成文件夹 10。当文件夹 7 与文件夹 9 结合时，由于文件夹 7 已经完成所有任务，文件夹 0 和 6 都已被释放，所以，重复利用文件夹 0 来存放将来的结果。最后，文件夹 10 与 0 结合，重复利用文件夹 1，将最终结果存放在文件夹 1 中。

4 实验结果验证与讨论

4.1 判断 2 个顶点间是否有边

3 个序列组匹配的测试结果如表 1 所示。其中，距离指编辑距离。匹配序列组中大写字符代表匹配的字符，小写字符代表不匹配的字符，插入和删除突变的则对应“-”。

从验证 1 看出，1−3 树分枝算法对于只存在替换突变的序列比对可以胜任；验证 2 表明，当存在插入和删除突变时，1−3 树分枝算法同样有效；验证 3 表明，在处理 1 条序列产生了几个连续的插入突变这样的极端情况时，1−3 树分枝算法仍然有效，而 Mismatch Tree[15]算法不能有效处理这种情况。

表 1 序列组匹配的验证实例

Table 1 Test results of comparing sequences

	原序列组	匹配序列组
验证 1	agaagaaaggttggg	agAagAAAgGttGGG
距离 7	caataaaacggcggg	caAtaAAAcGgcGGG
验证 2	tttgagtatccctgg	tTtGAGtATccc-TGg
距离 8	ctgagaattggatgt	cT-GAGaATtggaTGt
验证 3	accgtatggcaattg	AccgtATGGCAATTG----
距离 8	aatggcaattgccaa	A----ATGGCAATTGccaa

4.2 寻找 *k*-clique

测试基于在 k=10, n=82 的模拟范例序列中寻找 (15, 4)-motif，性能结果如图 6 所示。

从测试性能结果可以看出，分布式计算机制大大缩短了程序运行时间。虽然本文的算法要求第 1 步求 k 分图的任务全部完成返回后才能开始第 2 步的计算，但是，在该测试中，由于在第 1 步分解任务时有几台客户机参与计算就平均分解成几个任务，大的任务比小的任务最多大 1，大的任务先被取走计算，所以，空等待时间很少。

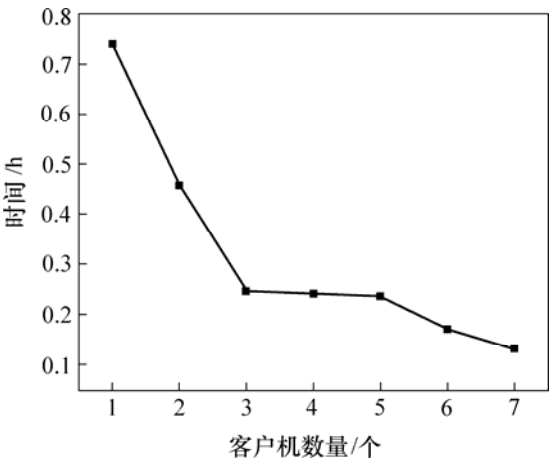

图 6 多台客户机运行时间测试结果

Fig.6 Test results of multiple clients

对这个范例序列进行实际计算后，得到 3 000 多个团，而实际上范例里标出的团只有 2 个。经过分析验证，确定本文的算法与程序是正确的，产生这样的结果是因为 Motif Finding 问题本身就是一个结果数量众多的问题，用精确算法得到结果后，还要用生物学知识进一步选择处理才能找出真正正确、有效的结果。

5 结 论

a. 研究了 Motif Finding 问题的精确算法，提出了 1−3 树分枝算法，有效地解决了问题向图转换时存在突变时难以表达的关键问题。

b. 设计了 Motif Finding 问题的分布式参数算法，采用对序列垂直划分的方法将计算任务分布化，同时设计了静态与组合模型解决任务负载均衡的问题。

c. 实现了精确求解 Motif Finding 问题的分布式计算系统。实验结果表明，设计与实现的分布式算法与计算系统能有效求解问题实例，并在多机共同计算时具有较好的协同性。

参考文献：

[1] Krawetz S A, David D. Introduction to bioinformatics: A theoretical and practical approach[R]. Totowa: Humana Press, 2003.

[2] 孙　霞, 殷鑫浈, 邬玲仟, 等. 弥漫性掌跖角化病家系角蛋白 9 基因突变热点区的检测[J]. 中南大学学报: 医学版, 2005, 30(5): 521−524.
SUN Xia, YIN Xin-zhen, WU Ling-qian, et al. Hotspot of the mutations of keratin 9 gene in a diffuse palmoplantar keratoderma family[J]. Journal of Central South University: Medicine Science, 2005, 30(5): 521−524.

[3] YU Jiang-sheng. NP completeness [R]. Beijing: Peking University, Institute of Computational Linguistics, 2003.

[4] Chin F, Leung H, Yiu S M, et al. Finding motifs for insufficient number of sequences with strong binding to transcription factor[C]//RECOMB04. San Diego: ACM Press, 2004: 125−132.

[5] Rajasekaran S, Balla S, Huang C H. Exact algorithms for planted motif problems[J]. Journal of Computational Biology, 2005, 12(8): 1117−1128.

[6] Shapiro J, Brutlag D. FoldMiner: Structural motif discovery using an improved superposition algorithm[J]. Protein Science, 2004, 13: 278-294.

[7] Hertz G Z, Stormo G D. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences[J]. Bioinformatics, 1999, 15(7): 563−577.

[8] Lawrene C E, Altschul S F, Boguski M S, et al. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment[J]. Science, 1993, 262(5131): 208−214.

[9] Bailey T L, Elkan C. Fitting a mixture model by expectation maximization to discover motifs in biopolymers[C]//Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology. California: AAAI Press, 1994: 28−36.

[10] 王建新, 莫秋菊. 基于 Internet 的通信系统虚拟实验环境设计与实现[J]. 中南大学学报: 自然科学版, 2006, 37(2): 128−133.
WANG Jian-xin, MO Qiu-ju. Design and implementation of communication system virtual environment based on Internet[J]. Journal of Central South University: Science and Technology, 2006, 37(2): 128−133.

[11] Pevzner P A, Sze S H. Combinatorial approaches to finding subtle signals in DNA sequences[C]//Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology. La Jolla: AAAI Press, 2000: 269−278.

[12] Sze S H, CHEN Jian-er. Find specific motifs in DNA sequences via cliques in *k*-partite graphs[R]. Texas A&M University: Departments of Computer Science and Biochemistry & Biophysics, 2003.

[13] Buhler J, Tompa M. Finding motifs using random projections[C]//Proceedings of the 5th Annual International Conference on Computational Molecular Biology. Montreal: ACM Press, 2001: 69−76.

[14] 李冬冬, 王正志, 杜耀华, 等. DNA 序列中模式发现的一种快速算法[J]. 生物物理学报, 2005, 21(2): 121−129.
LI Dong-dong, WANG Zheng-zhi, DU Yao-hua, et al. A fast Motif Finding algorithm for dna sequence[J]. Acta Biophysica Sinica, 2005, 21(2): 121−129.

[15] Eskin E. Sparse sequence modeling with applications to computational biology and intrusion detection[D]. New York: Columbia University, 2002.