ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.7, July 2006, pp.1501−1509
DOI: 10.1360/jos171501

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

RNA 二级结构预测中动态规划的优化和有效并行*

谭光明[1,2+], 冯圣中[1], 孙凝晖[1]

[1](中国科学院 计算技术研究所,北京 100080)

[2](中国科学院 研究生院,北京 100049)

An Optimized and Efficiently Parallelized Dynamic Programming for RNA Secondary Structure Prediction

TAN Guang-Ming[1,2+], FENG Sheng-Zhong[1], SUN Ning-Hui[1]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: Phn: +86-10-62565533 ext 5730, E-mail: tgm@ncic.ac.cn

Tan GM, Feng SZ, Sun NH. An optimized and efficiently parallelized dynamic programming for RNA secondary structure prediction. *Journal of Software*, 2006,17(7):1501−1509. http://www.jos.org.cn/1000-9825/17/1501.htm

Abstract: RNA secondary structure prediction based on free energy rules remains a major computational method in computational biology. Its basic dynamic programming algorithm needs $O(n^4)$ time to calculate the minimum free energy for RNA secondary structure, where n is the length of RNA sequence. There are two variants for handling this problem: either the internal loops are bounded by a maximal size k, giving a time complexity of $O(n^2 \times k^2)$, or one uses the trick of Lyngso, which makes use of the rules of loop energies, to reduce time complexity to $O(n^3)$ for suboptimal free energy without restriction. Only with additional $O(n)$ space, a new algorithm is proposed to eliminate the redundant calculation in the energy of internal loops and reduce the time complexity to $O(n^3)$ with unrestricted loop sizes for optimal free energy. While the optimized algorithm is time consuming, an efficient parallel algorithm with good load balancing in cluster systems is also proposed. The experimental results show that the parallel program achieves reasonable speedups.

Key words: minimum free energy; dynamic programming; redundant calculation; load balancing; speedup

摘 要: 基于最小自由能模型的方法是计算生物学中 RNA 二级结构预测的主要方法,而计算最小自由能的动态规划算法需要 $O(n^4)$的时间,其中 n 是 RNA 序列的长度.目前有两种降低时间复杂度的策略:限制二级结构中内部环的大小不超过 k,得到 $O(n^2 \times k^2)$算法;Lyngso 方法根据环的能量规则,不限制环的大小,在 $O(n^3)$的时间内获得近似最优解.通过使用额外的 $O(n)$的空间,计算内部环中的冗余计算大为减少,从而在同样不限制环大小的情况下,在 $O(n^3)$的时间内能够获得最优解.然而,优化后的算法仍然非常耗时,通过有效的负载平衡方法,在机群系统上实现并行程序.实验结果表明,并行程序获得了很好的加速比.

* Supported by the National Natural Science Foundation of China under Grant No.60372040 (国家自然科学基金); the Knowledge Innovative Project of the Chinese Academy of Sceiences under Grant No.KSCX2-SW-233 (中国科学院知识创新工程重大项目)

Received 2005-06-06; Accepted 2005-12-13

关键词: 最小自由能;动态规划;计算冗余;负载平衡;加速比
中图法分类号: TP301 文献标识码: A

RNA 二级结构预测已经成为计算生物学的主要研究领域之一.由于实验的方法,如 X 射线衍射等方法相当耗时,而且成本非常高,许多计算的方法预测 RNA 二级结构被提了出来[1,2].对不包括假结的二级结构预测,基于最小自由能[2]的动态规划算法是一种最实用的方法,逐渐成为预测 RNA 二级结构的主流.给定长度为 n 的 RNA 序列,使用基于最小自由能的动态规划算法预测包含假结[3]需要 $O(n^6)$时间和 $O(n^4)$空间,本文只讨论不包括假结的情况,后面不再加以区分.

RNA 序列 s 的二级结构可以表示为碱基对(i,j)的集合 S,其中:$1\le i<|s|;\forall(i,j),(i',j')\in S: i=i'\Leftrightarrow j=j'$.这就保证任何碱基最多与一个其他碱基配对,排除了假结的形成.RNA 二级结构 S 是带有一些外部未配对的碱基的环(loop)的集合(如图 1 所示).假设 $i<k<j,(i,j)\in S$,如果对所有的$(i',j')\in S$ 不满足 $i<i'<k<j'<j$,则称 k 是从(i,j)可访问的.如果包含碱基对(i,j)的环的所有碱基都可以访问(i,j),则碱基对(i,j)称为外部碱基对(external base);如果 i'和 j'是从(i,j)和(i',j')可访问的,则(i',j')称为环的内部碱基对并且从(i,j)是可访问的.没有内部碱基对的环称作发夹环(hairpin loop);对有一个内部碱基对的环,如果 $i'=i+1$ 而且 $j'=j+1$,称为堆积(stacked pair);否则称为内部环(internal loop),但有一种称作膨包(bulge)的特殊内部环,它满足 $i'=i+1$ 或者 $j'=j+1$;有多个内部碱基对的环称作多分支环(multibranched loop).未匹配的碱基和不能从任何碱基对访问的碱基对称作外部碱基.RNA 二级结构预测是对一条给定的序列决定其最稳定的结构.在最小自由能模型中,用自由能的大小评测结构的稳定性,自由能量最小的二级结构就是最优的二级结构.对最小自由能模型有如下假设[1]:二级结构的能量是组成结构的各个独立的环的能量总和,环的能量只依赖于环本身而与结构中的其他部分没有关系.下一节将详细描述计算最小自由能的基本动态规划算法,其时间和空间复杂度分别是 $O(n^4)$和 $O(n^2)$.

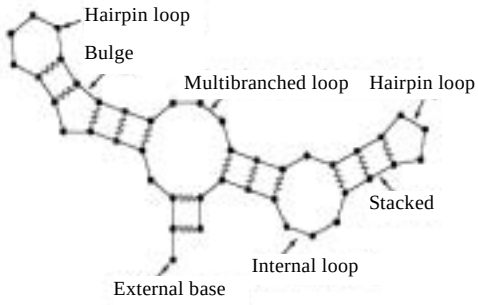

Fig.1 A example RNA structure
图 1 RNA 二级结构的组成

降低串行算法的时间复杂度是优化算法的行之有效的途径,但当优化后的串行算法仍然有较高的时间复杂度时,基于并行计算机体系结构的并行实现成为解决计算密集型问题的有效方法.通过冗余计算的消除,最小自由能的基本动态规划算法的时间复杂度从 $O(n^4)$减少到 $O(n^3)$.当 RNA 序列长度和规模增大时,$O(n^3)$的串行算法需要消耗大量的时间.动态规划算法需要计算动态规划矩阵,由于动态规划方法本身的特点,计算过程存在较强的数据依赖关系,增加了有效并行实现的难度,主要体现为并行算法的负载平衡策略.合理的负载平衡算法能够提高处理器的利用效率,从而获得更高的加速比.我们提出了一种动态负载平衡的策略,在机群系统上获得了更高的加速比.

本文第 1 节描述 RNA 二级结构的基本动态规划算法,分析其时间和空间复杂度.第 2 节介绍相关工作.第 3 节分析如何消除冗余计算,给出优化的算法.第 4 节在简单给出静态负载平衡的基础上,提出动态负载平衡并行算法.第 5 节列出实验结果并对算法的性能进行分析.最后是对本文工作的总结和下一步工作的展望.

1 RNA 二级结构预测基本算法

前面介绍了 RNA 二级结构的基本定义和假设.基于最小自由能模型的假设,给定一条 RNA 序列 s,可以得到计算序列 s 的最优结构的能量的递归式.4 个数组 W,V,VBI 和 VM 用于保存子序列 s 的某种结构的能量.不同环的类型对最小自由能的组成如下:

1. 子序列 $s_1\ldots s_i$ 的最优结构的自由能 $\mathbb{N}$

$$W(i)=\min\{W(i-1),\min_{1<j\le i}\{W(j-1)+V(j,i)\}\} \tag{1}$$

2. 包含碱基对(i,j)的子序列 $s_i\ldots s_j$ 的最优结构的自由能

$$V(i,j)=\min\{eH(i,j),eS(i,j)+V(i+1,j-1),VBI(i,j),VM(i,j)\} \tag{2}$$

其中:$eH(i,j)$是包含碱基对(i,j)的发夹环的能量;$eS(i,j)$是由碱基对(i,j)和$(i+1,j-1)$组成的堆积的能量;

3. 包含碱基对(i,j)的子序列 $s_i\ldots s_j$ 的最优结构的自由能,碱基对(i,j)形成膨包或者内部环

$$VBI(i,j)=\min_{\substack{i<i'<j'<j\\ i'-i+j-j'>2}}\{eL(i,j,i',j')+V(i',j')\} \tag{3}$$

其中:$eL(i,j,i',j')$是包含外部碱基对(i,j)和内部碱基对(i',j')的膨包或者内部环的能量;

4. 由碱基对(i,j)组成多分支环的子序列 $s_i\ldots s_j$ 的最优结构的自由能

$$VM(i,j)=\min_{i+1<k\le j-1}\{WM(i+1,k-1)+WM(k,j-1)+a\} \tag{4}$$

$$WM(i,j)=\min\{V(i,j)+b,WM(i,j-1)+c,WM(i+1,j)+c,\min_{i<k\le j}\{WM(i,k-1)+WM(k,j)\}\} \tag{5}$$

其中:$WM(i,j)$组成多分支环的子序列 $s_i\ldots s_j$ 的最优结构的自由能;a,b,c 是常量.

基于上面的递归关系式,通过动态规划算法在 $O(n^4)$时间内计算最优结构的能量.发夹环和碱基堆积能量的计算只需 $O(1)$的时间;内部环能量由递归关系式(3)给出,导致计算内部环需要 $O(n^4)$的时间复杂度;递归式(4)、式(5)计算多分支环的能量,时间复杂度是 $O(n^3)$.因此,整个基本动态规划算法的时间复杂度是 $O(1)+O(n^3)+O(n^4)=O(n^4)$.算法的数据结构都是二维数组,所以空间复杂度为 $O(n^2)$.

2 相关工作

直接实现上述 RNA 二级结构预测中的动态规划算法需要 $O(n^4)$时间,但当序列规模较大时,算法相当耗费时间.显然,基本动态规划算法的瓶颈是内部环的计算.为了提高内部环的计算速度,有两种不同的近似方法降低时间复杂度:HoHacker[4]通过限制内部环的最大长度,假设为 k(通常的实现中,k 设为 30),递归式(3)中 i',j' 变化区间的大小不超过 k,从而内部环的计算时间复杂度为 $O(n^2\times k^2)$,整个算法的时间复杂度降为 $O(n^3)$;根据内部环的分解规则[1,5],Lyngso[6]提出了一种计算近似最优二级结构的 $O(n^3)$算法,优化后的算法仍然有 $O(n^3)$时间复杂度.为了降低算法的运行时间,并行化是一种有效的方法.研究表明[7−10],对该类动态规划问题的并行算法,按动态规划矩阵的对角线计算可比按行或者列的计算更有效地开发并行性.然而,注意到这两种计算方法在填充动态规划矩阵时需要按行或者列的顺序,因此,为了更好地利用并行性,设计一种按对角线计算的 $O(n^3)$算法动态规划算法是必要的.

针对类似 RNA 二级结构预测的 $O(n^3)$动态规划算法,一些基于特殊体系结构的细粒度并行算法得到了大量的研究[7−9],而且,最近有基于机群系统对 RNA 类似二级结构预测的动态规划算法并行化的工作[10].与许多其他动态规划算法一样,RNA 二级结构预测算法中动态规划算法呈现了波前(wave-front)[11]的特点,即动态规划矩阵中的元素依赖于以前已经计算了的元素的值.也正是这种波前计算中的依赖关系,增加了并行化动态规划算法时数据和任务划分的难度,导致算法的计算负载严重不平衡.T. Liu[10]的并行实现采用静态负载平衡策略,并行程序的加速效果有待提高.

3 优化内部环计算

在上述的基本动态规划算法中,内部环的计算是瓶颈.为了计算内部环能量 $VBI(i,j)$,需要计算区域 $\{(i',j')|i<i'<j'<j,i'-i+j-j'>2\}$所有碱基对的能量,并且取其最小值.根据递归式(2),计算每个碱基对(i,j)形成的环的能量,都需要遍历该区域.从后面分析可以得出,这部分计算存在大量的冗余.在用二维数组实现的算法中,内部环的计算区域是一个逐渐扩大的上三角矩阵(如图 2 所示,图中右上角的两个点从左到右分别表示(i,j)和$(i,j+1)$,最右边的一列是计算$(i,j+1)$新增加的点,即每条对线上只增加了一个点.反对角线 i 和 $i+11$ 之间的区域是计算(i,j),为 asymmetry 区域;而反对角线 $i+1$ 和 $i+12$ 之间的区域是计算$(i,j+1)$,为 asymmetry 区域).

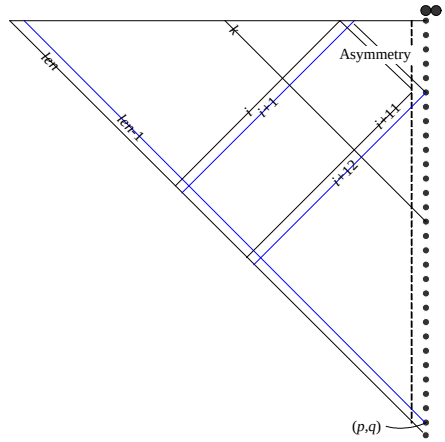

Fig.2 The dependent entries for computing $VBI(i,j)$ (excluding bugle loop)

图 2 $VBI(i,j)$的计算区域(不包括 bulge 区域)

Zuker[5]提出了一种内部环能量分解的规则,即计算内部环的能量分为 3 部分:(a) 依赖于环大小的熵值;(b) 与形成闭合的内(外)碱基对邻接的没有配对的碱基堆积能量;(c) 不对称环的惩罚能量.根据该内部环能量分解规则,内部环能量的计算公式可用式(6)表示.

$$VBI(i,j)=\min_{\substack{i<i'<j'<j\\ i'-i+j-j'>2}}\{size(i'-i+j-2)+asymmetry(i'-i-1,j-j'-1)+stacking(i,j)+stacking(i',j')+V(i',j')\} \quad (6)$$

这种方法把膨包形式当成一种特殊内部环,膨包的能量计算规则不同于通常的内部环,但其值只依赖于与(i,j)在同一水平线$\{(i',j')|i=i',i<j'<j\}$和垂直线$\{(i',j')|j=j',i'<i<j\}$上(即内部环计算区域的边界)的所有碱基对能量的最小值.因此实际计算中,膨包可以单独计算.Papanicolaou[1,5]指出不对称环的惩罚能量只依赖于绝对值$|i'+j'-i-j|$,通过实验的方法进一步验证并观测到惩罚值当$|i'+j'-i-j|>5$ 时达到最大值(MAX_*ASY*).因此,当$|i'+j'-i-j|\le 5$ 时,可用式(6)单独计算.现在我们考虑其他区域的计算,其计算顺序采取沿对角线推进的方式.由于所有的碱基对的位置满足$|i'+j'-i-j|>5$,故式(6)中的不对称环的惩罚能量 $asymmetry(i'-i-1,j-j'-1)$用其最大值 MAX_*ASY* 代替.这样得到能量计算公式(7):

$$VBI(i,j)=\min_{\substack{i<i'<j'<j\\ i'-i+j-j'>2}}\{size(i'-i+j-j'-1)+stacking(i,j)+stacking(i',j')+V(i',j')+\mathrm{MAX\_}ASY\} \quad (7)$$

考虑两个连续碱基对(i,j)和$(i,j+1)$的内部环能量的计算,递归关系式(2)、式(3)形成了 $VBI(i,j+1)$的值依赖于 $VBI(i,j)$.然而,注意到二者的计算区域存在着重叠部分,即导致了计算上的冗余.如在计算 $VBI(i,j)$时,需要计算其计算区域中所有碱基对能量的最小值,但计算 $VBI(i,j+1)$仅仅增加了一列碱基对$\{(i,k)|i\le k\le j+1\}$,其他碱基对的能量在计算 $VBI(i,j)$时已经得到了.因此,如果没有一种策略保存已经计算了的重叠区域的碱基对的能量,将会导致大量的冗余计算.式(7)中 $stacking(i',j')+V(i',j')+$MAX_*ASY* 的值与内部环的长度没有关系,只有 $size(i'-i+j-j'-1)$ 随着环的长度发生变化.不考虑对内部环的长度的限制,设计算 $VBI(i,j)$时内部环的长度是 $len-1$,则计算 $VBI(i,j+1)$是内部环的长度增加为 *len*.引入一个以内部环的长度为索引的一维数组 $LOOP(len)$,定义见式(8).

$$LOOP(len)=\min_{i'-i+j-j'+2=len}\{stacking(i',j')+V(i',j')\} \quad (8)$$

$LOOP(len)$代表了形成长度为 *len* 的碱基对(i,j)的计算区域中所有碱基对的能量最小值,公式(8)只是数组的形式定义,从该定义中可以得到计算该数组的递推关系式.如前所述,当计算完碱基对(i,j)的内部环的能量后,计算下一个碱基对$(i,j+1)$时只需另外比较一列碱基对的能量,对应到对角线的计算方式,即向前推进时,每条对角线上的能量集合只增加了一个碱基对的能量.设新增的碱基对为(p,q),$LOOP(len)$的递推计算公式如下:

$$LOOP(len)=\min\{LOOP(len-1),stacking(p,q)+V(p,q)\} \quad (9)$$

通过公式(9),当计算 $VBI(i,j+1)$时,不需要遍历其整个计算区域,只需将计算 $VBI(i,j+1)$时得到的 $LOOP(len)$

数组和新增加的碱基对的能量进行一次比较操作.

因此,基于 Zuker 的内部环能量分解规则,我们将其计算区域划分为 3 个独立的部分,修改内部环的计算公式(6),得到新的内部环能量计算公式

$$VBI(i,j)=\min\begin{cases} size(i'-i+j-j'-2)+asymmetry(i'-i-1,j-j'-1)+ \\ \qquad stacking(i,j)+stacking(i',j')+V(i',j') \mid i'+j'-i-j \mid <5 \\ bulge(i'-i+j-j'-1)+stacking(i,j)+stacking(i',j')+V(i',j') i'-i-1=0 \text{ or } j-j'-1=0 \\ \min\{LOOP(len)+size(len)+stacking(i,j)+\mathrm{MAX\_}ASY\} 4<len\le n \end{cases} \tag{10}$$

公式(3)计算内部环的能量需要遍历整个二维计算区域,时间复杂度为 $O(n^2)$;而公式(10)只需对一维数组 $LOOP(len)$的每个元素进行一次比较操作,时间复杂度是 $O(n)$,引入的额外空间消耗是 $O(n)$.然而,使用公式(10)计算内部环的能量,使得整个算法的时间复杂度降为 $O(n^3)$.

尽管通过增加 $O(n)$的额外空间消耗降低了内部环能量的计算时间复杂度,但仍需要 $O(n^3)$的时间.因此,如第 5 节中的实验所示,当序列很长时,计算二级结构的运行时间非常高.进一步降低算法的时间复杂度似乎比较困难,而在现有的并行计算机体系结构上实现有效的并行算法却提供了一种很好的解决办法.

4 并行动态规划算法

并行算法将围绕在填充动态规划矩阵过程中矩阵划分与处理器的映射,设计一种提高处理器利用率的负载平衡划分算法.设给定 p 个处理器:0,1,…,p–1,序列的长度为 n,不失一般性,假设 n 被 p 整除.一种简单的动态规划矩阵的划分沿着行平均划分,即处理器 i 计算行:$i\times n/p$+1,…,(i+1)×n/p,我们称这种方法为静态划分算法.如图 3 所示,波前计算沿着对角线的方向推进,每个处理器计算的矩阵块大小相等.静态划分的并行算法可以分为 p 个阶段,每个阶段由两个步骤组成.第 1 阶段的第 1 步处理器 i 计算三角形块 $M(i+1,1)$,在阶段 s($2\le s\le p$),处理器 i($i\le p-s$)计算矩阵块 $M(i+1,s)$,因此,在阶段 s 只有 $p-s+1$ 个处理器处于计算状态.在处理器 i 计算 $M(i+1,s)$时,其中 $2\le s\le p-i$,依赖于自身和处理器 i+1 的计算结果,因此需要在适当的时候进行数据传输.在每个阶段的第 2 步,处理器 i 将计算结果和从处理器 i+1 接收的数据传送到处理器 i–1.这种简单的静态划分策略的关键路径是计算最后一个矩阵块的处理器 0 的运行时间.整个并行程序的运行时间分为两部分:第 1 步的计算时间 T_{comp} 和第 2 步的通信时间 T_{comm},其中的 $T_{comp}=\sum_{s=1}^{p} s\times n\times(n/p)^2=O(n^3/p)$, $T_{comm}=\sum_{s=1}^{p-1}2s\times(n/p)^2=O(n^2/p)$,整个算法的时间复杂度等于 $O(n^3/p)$.

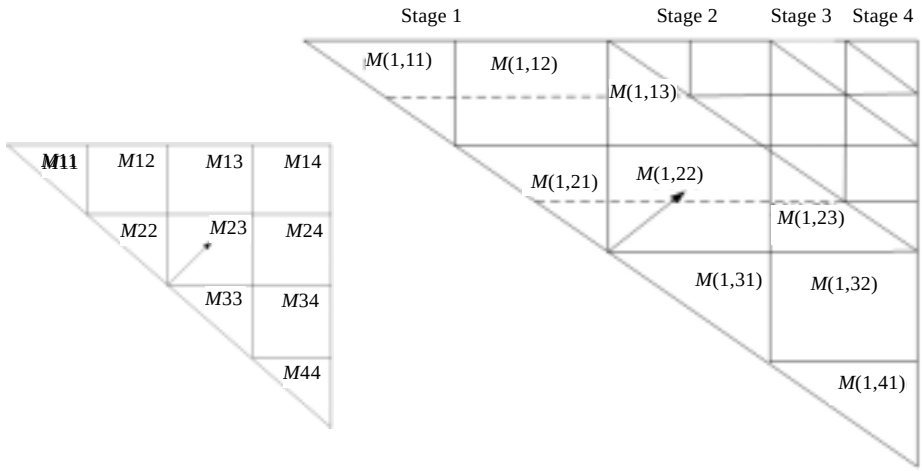

Fig.3 The wavefront computation in the static and dynamic partition algorithm

图 3 静态划分和动态划分中的波前计算策略

显然,静态划分算法在每个阶段结束后都增加一个空闲的处理器,导致处理器的利用率非常低.我们的解决

办法是:当空闲的处理器个数大于给定的阈值 t 时,对未计算的动态规划矩阵重新划分,阈值 t 的大小决定了算法的性能.当 $t=1$ 时,每个阶段都需要重新划分,会导致处理器之间的频繁通信;当 $t=p$ 时,只划分一次,变成了静态划分算法.这种引入重新划分策略的算法成为动态划分算法,对动态划分策略,必须考虑由于重新划分增加的通信开销的影响.在提供处理器的利用率的同时,恰当选取 t 的值,这里只考虑 $t=p/2$ 的情况,保证划分的次数不会太频繁也不会太少:太频繁会增加通信的开销,太少又会降低处理器的利用率.

不失一般性,我们假设 p 整除 n,n 是 2 的幂.整个并行算法被分成 $\log(n/p)+1$ 个阶段,每个阶段又分成 $p/2+1$ 个子阶段,每个子阶段由两个步骤组成.子阶段和步骤对应于静态划分算法中的阶段和步骤,在两个阶段之间进行重新划分.DP 矩阵沿着对角线划分成 $\log(n/p)+1$ 个条带,标记为 $1,2,\ldots,\log(n/p)+1$,每个对角线条带包含了 $n/2^i$ 条对角线,最后一个条带包含了 p 条对角线.从第 1 个条带开始,第 i 个条带将在第 i 个阶段使用静态划分算法计算完成.然后将 DP 矩阵在处理器之间重新划分,开始第 $i+1$ 个阶段.重新划分时,对 $0<i<p$,将处理器 i 的数据分成两半分别发送给处理器 $2\times i$ 和 $2\times i+1$,处理器 0 只需将下半部分发送给处理器 1,处理器 p 仅需接受来自 $p/2$ 的数据(如图 3 所示,其中的左图和右图分别是静态和动态划分算法的示例,右图中的虚线代表重新划分).

在每个阶段,对 DP 矩阵水平和垂直方向平均划分,当 $1\le s\le\log(n/p),1\le j\le p/2$ 时,$M^s_{i+1,j}$ 表示 s 阶段处理器 i 在第 j 个子阶段要计算的对角线块.最后一个阶段,即 $\log(n/p)+1$ 阶段,每个处理器沿着对角线方向分别计算一行,处理器 0 计算最后一个矩阵单元.由于我们只需计算 DP 矩阵的上三角部分,因此,在每个阶段的第 1 个子阶段计算的也是上三角形的对角线块.

动态划分算法的描述见算法 1.

算法 **1**.

```
for all processors i in 0,…,p–1 do in parallel
    for all stages in 1,…,log(n/p)
        for all phase ss in 1,…,p/2
            //step 1
            compute M^s(i+1,ss) from M^s(i+1,j) and M^s(i+j+1,ss–j),1≤j≤ss–1;
            //step 2
            save M^s(i+1,ss);
            if i=0
                receive M^s(i+j+1,ss–j+1) from processor i+1,1≤j≤ss;
            else if i=p–1
                send M^s(i+1,ss) and M^s(i+j+1,ss–j) to processor i–1,1≤j≤ss–1;
            else
                receive M^s(i+j+1,ss–j+1) from processor i+1,1≤j≤ss;
                send M^s(i+1,ss) and M^s(i+j+1,ss–j) to processor i–1,1≤j≤ss–1;
        endfor
        //repartition
        partition DP matrix into two halves block by row;
        if i<p/2
            if i=0
            send lower half block to processor 2*i+1;
            else if i<p
                send upper half block to processor 2*i;
                send lower half block to processor 2*i+1;
            if i>0
            receive block from processor i/2
```

endfor

endfor

整个程序的运行时间也分为计算时间 T_{comp} 和通信时间 T_{comm},但由于对动态规划矩阵进行重新划分,通信时间进一步分为静态通信时间 T_{stcomm} 和重新划分通信时间 $T_{repartition}$.计算时间和静态通信时间等于静态划分算法相应部分,增加了重新划分通信时间.在每个阶段,处理器 i 最多发送$(n/2^{s-1}p)(n-n/2^{s})$项给处理器 $2\times i$,因此整个重新划分导致增加的通信时间为

$$T_{repartition}=\sum_{s=1}^{\log(n/p)}(n/2^{s-1}p)(n-n/2^{s})\log(p)=O(n^2\log(p)/p)\ .$$

从时间复杂度的分析来看,动态划分算法并没有改变算法的时间复杂度,只是增加了额外的通信时间的开销.尽管引入了额外的通信时间,但动态划分算法极大地提高了处理器的利用率.后文的实验表明:动态划分算法降低了并行程序的运行时间,而且加速比相对于静态划分算法更高.

5 实验结果

对算法的性能评价分为两部分:首先给出 $O(n^3)$优化的动态规划算法的运行时间和现有的流行 RNA 二级结构预测软件的运行时间的比较;然后对基于 $O(n^3)$优化的动态规划算法的两种不同的并行算法进行详细的时间测试和加速比的分析.

5.1 $O(n^3)$优化的动态规划算法

ViennaRNA[4]实现了对内部环的最大长度限制在 30 的 $O(n^4)$算法(事实上,其算法复杂度是 $O(n^3)$,这里说 $O(n^4)$是指其采用的基本的动态规划算法,但对内部环的长度做了限制),我们基于 ViennaRNA 二级结构预测软件包实现了 $O(n^3)$动态规划算法,称为 IctRNA.测试平台是 1.6GHz Opteron 处理器,内存 3GB.实验结果表明,IctRNA 相对于 ViennaRNA 获得了明显的加速比.

首先,给出对内部环最大长度限制下时间消耗的比较.表 1 给出了两者的时间消耗比较(内部环的最大长度限制在 30).随着 RNA 序列长度的增加,优化内部环能量计算获得的加速比减少,这是由于内部环的长度受到了限制,使得计算内部环能量的时间占总时间的比例减少.这种情形下,算法的瓶颈是多分支环能量的计算.如在计算 *lsu_ecoli* 的二级结构时,多分支环能量计算时间几乎是内部环能量计算时间的两倍.

Table 1 The runtime in minutes of ViennaRNA and the IctRNA

表 1 IctRNA 和 ViennaRNA 的运行时间比较

Programs	ViennaRNA		IctRNA	
Sequences	VBI	Overall	VBI	Overall
trna_yeast_phe.fa (83bp)	0.010	0.016	0.007	0.013
5S_ecoli.fa (154bp)	0.036	0.078	0.023	0.032
SRP_human.fa (312bp)	0.401	0.454	0.217	0.262
RNaseP_ecoli.fa (387bp)	0.560	0.643	0.336	0.423
ssu_ecoli.fa (1556bp)	11.061	16.390	6.359	11.618
lsu_ecoli.fa (2918bp)	37.642	69.123	22.477	50.073

我们更关心的是对内部环长度不受限制情况下优化算法的性能.所谓长度不受限制并不是说可以无穷大,显然,组成二级结构的各种环的长度不可能超过序列的长度 n,因此,在算法的实现中设内部环的最大长度是 n.由于这时内部环的计算时间占了总时间的绝大部分,表 2(内部环的长度无限制时)中只给出内部环的计算时间和总时间.从表 2 的时间比较可以看出,优化的算法获得了巨大的加速比.

Table 2 The runtime in minutes of ViennaRNA and the IctRNA

表 2 IctRNA 和 ViennaRNA 的运行时间比较

Programs	ViennaRNA		IctRNA	
Sequences	VBI	Overall	VBI	Overall
trna_yeast_phe.fa (83bp)	0.009	0.017	0.005	0.012
5S_ecoli.fa (154bp)	0.075	0.083	0.019	0.028
SRP_human.fa (312bp)	3.358	3.381	0.253	0.289
RnaseP_ecoli.fa (387bp)	7.359	7.412	0.251	0.523
ssu_ecoli.fa (1556bp)	3 204.177	3 208.005	31.786	35.482
lsu_ecoli.fa (2918bp)	51 166.314	51 194.369	213.668	237.222

5.2 并行算法的性能

我们用 MPI 实现了静态划分和动态划分的并行动态规划算法,测试平台是中国科学院计算技术研究所国家智能中心开发的高性能计算机系统 DAWNING 4000A,每个节点由 2.4GHz AMD Opteron 处理器、8GB 内存组成,计算节点之间由 2Gb/s Myrinet 网络互连.测试中只考虑对内部环长度不限制的情况,一个处理器上运行一个程序,采用两种划分策略的并行程序的运行时间见表 3.测试序列分别是 312(*SRP_human.fa*),1556(*ssu_ecoli.fa*),2918(*lsu_ecoli.fa*),串行程序的运行时间分别是 1.08s,263.32s,4273.26s.

Table 3 The runtime in seconds distribution of two partition algorithms (s)

表 3 静态和动态划分算法的运行时间分布和比较 (秒)

Len	Procs	Overall		Computation		Communication		Speedup	
Algorithms		Static	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic
312	2	0.949	0.734	0.914	0.703	0.035	0.031	1.14	1.47
	4	0.664	0.445	0.656	0.355	0.008	0.09	1.62	2.42
	8	0.402	0.328	0.395	0.078	0.007	0.25	2.68	3.29
	16	0.352	0.238	0.215	0.172	0.137	0.066	3.06	4.53
	32	0.266	0.219	0.121	0.051	0.145	0.168	4.06	4.93
1 556	2	220.59	169.734	220.578	169.562	0.012	0.172	1.19	1.55
	4	160.621	85.172	160.48	84.609	0.141	0.563	1.63	3.09
	8	97.09	42.707	96.891	42.273	0.199	0.434	2.71	6.16
	16	53.406	22.051	53.129	21.113	0.277	0.938	4.93	11.94
	32	28.426	13.223	27.859	10.543	0.567	2.68	9.26	19.91
2 918	2	3 596.637	2 774.266	3 596.469	2 771.047	0.168	3.219	1.18	1.54
	4	2 677.004	1 442.59	2 676.445	1 379.574	0.559	63.016	1.59	2.96
	8	1 633.156	859.289	1 628.141	702.219	5.015	157.07	2.61	4.97
	16	912.562	512.27	895.922	347.523	16.64	164.747	4.68	8.34
	32	483.582	218.258	465.457	77.207	18.125	141.051	8.83	19.57

尽管动态划分策略增加了处理器之间的通信开销,但由于更好的、复杂的平衡使得处理器的利用率提高;另一方面,实验表明计算时间占整个运行时间的很大部分,动态划分策略明显降低了计算时间.这样,整个程序的运行时间减少.由于通信时间的相对比例不同,不同序列出现的最大加速比也不同.两种算法都是在 32 个处理器时出现了最大加速比.通信时间的增加导致加速比的降低,在动态划分策略中,重新划分引入的通信时间是通信时间的主要部分.然而,动态负载平衡提高了处理器利用率,动态划分算法仍然获得了相当于静态划分算法的 2 倍的加速比.

从实验测试的时间分布来看,当问题规模(序列长度)增加时,计算时间占据了绝大部分的运行时间.因此,这种并行动态规划算法的瓶颈是计算而不是通信.动态划分策略侧重于用负载平衡提高处理器的利用率,降低了计算时间.问题规模固定,当处理器规模增加时,每个处理器上的问题规模变小,导致处理器之间的数据传输减少,但处理器之间的通信次数更加频繁.由于启动通信的开销较大,导致总的通信开销超出了计算开销.通信启动开销是一个与具体系统相关的参数,影响该值的因素有网络传输协议和运行环境中间件(如 MPI 编程环境).在本工作的测试平台中采用的是 Myrinet[12]网卡和 MPICH-GM,Myrinet 采用的是虚拟界面架构(virtual interface architecture,简称 VIA)[13],启动延迟相对于 TCP/IP 协议架构要小.因此,尽管动态划分算法的通信操作更加频繁,但所消耗的时间对整个程序的运行时间影响不大.对于使用 TCP/IP 架构的网络互连,尽管由于启动延迟开销增加导致通信开销会明显增多,但从表中可以看到,动态划分策略的并行算法在问题规模增大时仍然

保持较高的加速效果,说明有较好的可扩展性加速比,而且计算时间还是主要部分.因此,对于大规模问题,动态划分算法仍然能够取得很好的加速效果.

6 结论和下一步工作

基于最小自由能模型的 RNA 二级结构预测动态规划算法是时间复杂度很高的耗时方法,利用计算中空间依赖关系消除计算过程中冗余计算部分,我们将其中的 $O(n^4)$基本动态规划算法优化为 $O(n^3)$算法.实验表明,实现优化算法的 IctRNA 软件比现有的流行二级结构预测软件 ViennaRNA 速度要快.但优化后的算法还是比较耗时的,利用现有性价比较高的机群系统,我们实现了具有较高加速比的并行动态规划算法.同时,对其中的负载平衡策略进行了研究.对于通信代价不是很高的并行动态规划算法,动态的负载平衡比静态负载平衡算法更能提高处理器的利用率、降低程序的运行时间,并且提高整个的加速比.

研究如何进一步降低 RNA 二级结构预测动态规划算法的时间复杂度,是一个有趣而又具有挑战性的课题.我们注意到:在生物信息学中存在大量的如采用动态规划算法的组合优化问题,而且进一步降低这些算法的时间复杂度相当困难.能否利用现代计算机体系结构的特点,如存储层次、指令级并行等对这些在计算理论上很难优化的算法,采用实验的方法进行适应不同体系结构的优化,将成为下一步的一个重要的研究方向.

References:

[1] Tinoco I, Borer PN. Improved estimation of secondary structure in ribonucleic acids. Nature New Biology, 1973,246(150):40−41.

[2] Gardner PP, Giegerich R. A comprehensive comparison of comparative RNA structure prediction approaches. BMC Bioinformatics, 2004. 1−32.

[3] Rivas E, Eddy S. A dynamic programming algorithm for RNA structure prediction including pseudoknots. Journal of Molecular Biology, 1999,285(5):2053−2068.

[4] Hacker I. 2006. http://www.tbi.univie.ac.at/~ivo/RNA/

[5] Zuker M. 2006. http://www.ibc.wustl.edu/~zuker/rna/energy

[6] Lyngso RB, Zuker M. Fast evaluation of internal loops in RNA secondary structure prediction. Bioinformatics, 1999,15(6): 440−445.

[7] Rodriguez C, Roda J, Almeida, F, Gonzalez D. Paradigms for parallel dynamic programming. In: Proc. of the EUROMICRO. IEEE Computer Society, 2002. 553−560.

[8] Hsuna S, Huang S, Liu F. Parallel dynamic programming. IEEE Trans. on Parallel and Distributed Systems, 1994,5(3):326−328.

[9] Parallel dynamic programming [Ph.D. Thesis]. Phillip Gnassi Bradford: The University of Alabama, 1994.

[10] Liu T, Schmidt B. Parallel RNA sequence-structure alignment. In: Proc. of the 18th Int'l Parallel and Distributed Processing Symp. New Mexico: IEEE Computer Society, 2004. 1034−1041.

[11] Grama A, Gupta A, Karypis G, Kumar V. Introduction to Parallel Computing. Addison Wesley, 2003.

[12] http://www.myri.com/. 2006.

[13] Dunning D, Rengnier G, McAlpine G. The virtual interface architecture. IEEE Micro, 1998,18(1):46−57.

谭光明(1980－),男,湖南衡阳人,博士生,主要研究领域为算法设计,并行计算.

孙凝晖(1968－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为计算机组织,系统结构.

冯圣中(1968－),男,博士,高级工程师,CCF 高级会员,主要研究领域为高性能计算,生物信息学,网格计算.