Exploiting Parallelization for RNA Secondary Structure Prediction in Cluster

Guangming Tan, Shengzhong Feng, and Ninghui Sun

Institute of Computing Technology, Chinese Academy of Sciences
{tgm, fsz, snh}@ncic.ac.cn

Abstract. RNA structure prediction remains one of the most compelling, yet elusive areas of computational biology. Many computational methods have been proposed in an attempt to predict RNA secondary structures. A popular dynamic programming (DP) algorithm uses a stochastic context-free grammar to model RNA secondary structures, its time complexity is $O(N^4)$ and spatial complexity is $O(N^3)$. In this paper, a parallel algorithm, which is time-wise and space-wise optimal with respect to the usual sequential DP algorithm, can be implemented using $O(N^4/P)$ time and $O(N^3/P)$ space in cluster. High efficient utilization of processors and good load balancing are achieved through dynamic mapping of DP matrix to processors. As experiments shown, dynamic mapping algorithm has good speedup.

1 Introduction

Although computational methods such as phylogenetic comparative methods [2] to predict RNA secondary structure only provide an approximate RNA structural model, they have been widely used in the research of RNA secondary structures. In phylogenetic comparative methods, a stochastic context-free grammar (SCFG)[2] is used to model RNA secondary structure. The core of this method is dynamic programming (DP) algorithm [1][2][3]. Applications of SCFG-based RNA models require the alignment of an RNA sequence to a model. The computational complexity of polynomial-time dynamic programming algorithms for a sequence of length L and a model with K states is $O(K \cdot L^2 + B \cdot L^3)$, where B is the number of bifurcation states in model [3] or $O(L^4)$ when $B \sim M$. The alignment runtime on sequential computers is too high for many RNAs of interest such as ribosomal RNA (rRNA).

Cluster systems [4] have become very popular because of their cost/performance and scalability advantages over other parallel computing systems, such as supercomputers. Load balancing is very important to the performance of application programs in cluster systems. Good load balancing strategy improves the efficiency of processors, thus reduces the overall time of application programs. In this paper, we present an efficient parallel algorithm for aligning an RNA sequence to an SCFG model. The parallel algorithm consumes $O(N^4/P)$ time and $O(N^3/P)$ space, it is time-wise and space-wise optimal with respect to sequential algorithm.

The rest of paper is structured as follows. In section 2 we briefly review the SCFG model of RNA secondary structure and the DP algorithm. In section 3, we propose load balancing parallel algorithm through dynamic partitions of DP matrix. In section 4 we show experiment results and evaluate the performance of the parallel algorithm. Section 5 concludes this paper.

2 SCFG-Based Algorithm in RNA Secondary Structure Prediction

Eddy [3] described sequence-structure alignment algorithm in detail (The notations and definitions in this paper are referred to [2][3], we will not give their descriptions in detail). An RNA sequence can be aligned to a CM to determine the probability that the sequence belongs to the modeled family. The most likely parse tree generating the sequence determines the similarity score. Given an input sequence $x = x_1 \dots x_N$ of length N and a CM G with K states numbered in preorder traversal. The DP algorithm iteratively calculates a 3-dimensional DP matrix $M(i, j, v)$ for all $i = 1, \dots, j+1$, $j = 0, \dots, N$, $v = 1 \dots K$. $M(i, j, v)$ is the log-odds score of the most likely CM parse tree rooted at state k that generates the subsequence $x_i \dots x_j$. The matrix is initialized for the smallest subtrees and subsequence, i.e. subtrees rooted at E-states and subsequence of length 0. The iteration then proceeds outwards to progressively longer subsequences and larger CM subtrees.

$$M(i, j, v) = \begin{cases} \max_{\gamma \in \mathcal{V}} \{M(i, j, \gamma) + \log t_v(\gamma)\} & \text{if } S(v) \in \{DEL, BEG\} \\ e_v(x_i, x_j) + \max_{\gamma \in \mathcal{V}} \{M(i+1, j-1, \gamma) + t_v(\gamma)\} & \text{if } S(v) \in MP \text{ and } d \geq 2 \\ e_v(x_i) + \max_{\gamma \in \mathcal{V}} \{M(i+1, j, \gamma) + t_v(\gamma)\} & \text{if } S(v) \in \{ML, IL\} \text{ and } d \geq 1 \\ e_v(x_j) + \max_{\gamma \in \mathcal{V}} \{M(i, j-1, \gamma) + t_v(\gamma)\} & \text{if } S(v) \in \{MR, IR\} \text{ and } d \geq 1 \\ \max_{i-1 \leq mid \leq j} \{M(i, mid, kleft) + M(mid+1, j, kright)\} & \text{if } S(v) \in BIF \\ 0 & \text{if } S(v) \in E \\ -\infty & \text{otherwise} \end{cases} \quad (1)$$

At the end of the iteration $M(1, N, 1)$ contains the score of the best parse of the complete sequence with the complete model.

3 Parallel RNA Secondary Structure Prediction

As many other DP algorithms, the DP algorithm for RNA secondary structure prediction exhibits the character of a wave-front computation [5], that is, each matrix element depends on a set of previously computed elements. However, it is the wave-front computation that increases the difficulty of the parallelization of DP algorithm. Due to the character of a wave-front computation, our parallel algorithm focuses on DP matrix mapping to processors in the process of filling the DP matrix. The performance of parallel algorithm greatly depends on the strategy of the partition of DP matrix.

A simple matrix mapping method is to statically partition the matrix along the diagonal as depicted in [4][5]. The utility efficiency of processors is very low in the static partitioning algorithm since one processor becomes idle at each stage of p stages. A solution for $p < n$ is to repartition the matrix among the processors when the number of idle processors exceeds some threshold t . The t value directly determines the performance of algorithm. When $t = 1$, repartitioning occurs at every stage. The communication among processor is too frequent. When $t = p$, it never repartitions and corresponds to the static partitioning algorithm. In SCFG model based RNA secondary structure prediction algorithm the case where $t = p/2$ is discussed. This threshold can avoid the extremes of repartitioning too frequently and too infrequently.

Without loss of generality assume that p evenly divides into $\log(n/p)+1$ stages, each of which is further divided into $p/2 + 1$ phases, each phases is composed of two steps. The phases and steps are analogous to the stages and steps of the static partitioning algorithm, respectively. Repartitioning occurs between stages. Along the diagonal the DP matrix is partitioned into strips numbered $1, 2, \dots, \log(n/p)+1$ from the left of M such that strip i has $n/2^i$ diagonals for $1 \leq i \leq \log(n/p)$. The last strip has p diagonals. Starting with the first strip, the i th strip is filled at stage i using the static partitioning algorithm. Then previously computed entries are repartitioned among the p processors and stage $i+1$ begins. Processors i ($i \leq p-1$) evenly partitions the entries stored in local into two halves by rows. The low half entries are sent to processor 2^i+1 , except for processor 0, the high half entries are sent to processor 2^i . Accordingly, processor i receive entries from processor $i/2$.

Actually, the matrix is evenly partitioned by p both vertically and horizontally at each stage. For all $1 \leq s \leq \log(n/p), 1 \leq j \leq p/2, M_{(i+1)j}^{(s)}$ denotes the diagonal block computed in processor i in j phrase of stage s . At the last stage $\log(n/p)+1$, each processor computes one row along diagonal and processor 0 computes the last cell. Both DP matrix and entries in the first phrase of each stage are triangular. (see Fig. 1)

Fig. 1. The wavefront computation in dyanmic portioning algorithm. The dashed represents the repartition

The overall time can again be decomposed two parts: a computation time T_{comp} , a communication time T_{comm} , the communication time comprises a static communication time T_{stcomm} and a repartition time $T_{\text{repartition}}$. The computation time T_{comp} and the static communication time T_{stcomm} are $O(n^4/p)$ and $O(n^3)$, respectively. At each stage s , processor i sends no more than $(n/2^{s-1}p)(n-n/2^s)$ entries to processor 2^i .

Hence, the total repartitioning time $T_{\text{repartition}} = \sum_{s=1}^{\log(n/p)} (n/2^{s-1}p)(n-n/2^s)\log(p) = O(n^2\log(p)/p)$. The overall time is $T_{\text{comp}} + T_{\text{stcomm}} + T_{\text{repartition}} = O(n^4/p) + O(n^3) + O(n^2\log(p)/p) = O(n^4/p)$. The maximum space occurred at the first stage, it is $O(n^3/p)$.

4 Performance Evaluation

We have implemented the presented two parallel algorithms using MPI and experimentally evaluated their performance on a NUMA cluster systems. Each node consists of four 2.4GHz AMD Opteron CPUs and 8GB memory. The nodes are connected by 2Gbits/s Myrinet switch.

As the experiment shown, The experimental evidence indicates that the computation time dominates the communication time for all of the runs. The dynamic partitioning algorithm greatly reduces computation time. Table.1 give the results for RNA sequences length 120(5S_ecoli), 377(RnaseP), 1542 (lsu_ecoli) and 2904 (ssu_ecoli). The maximum speedup occurred using 32 processors for two algorithms. Due to relative contribution of the communication time is great, the maximum speedup for all input length is difference. The lower speedup is caused by the more communication cost. For the dynamic partitioning algorithm, the repartitioning time contribute significantly to the communication time. However, thanks to the increasing efficiency of processors, the speedup of the dynamic partitioning algorithm is fine.

Table 1. The run time of dynamic partitioning algorithm. Time: seconds

Length	processors	overall	compute	communication	speedup
120	2	0.724	0.702	0.024	1.47
	4	0.445	0.355	0.09	1.64
	8	0.328	0.078	0.25	2.20
	16	0.238	0.172	0.066	4.63
377	2	0.210	0.051	0.158	4.03
	4	10.063	10.798	0.165	1.61
	8	5.075	5.075	0.001	2.22
	16	2.501	2.588	0.113	5.52
1542	2	1.565	1.324	0.241	10.59
	4	0.98	0.675	0.304	15.92
	8	169.734	169.565	0.175	1.55
	16	85.175	84.609	0.565	3.08
2904	2	48.707	48.873	0.434	6.16
	4	88.051	81.113	0.938	11.94
	8	15.883	10.343	5.54	19.91
	16	8774.886	8771.047	3.838	1.54
	2	1445.59	1359.274	86.316	11.98
	4	859.889	705.519	154.37	4.97
	8	515.57	347.525	168.047	8.34
	16	318.558	77.507	141.051	19.57

5 Conclusion

We have given parallel DP algorithm for RNA secondary structure prediction in NUMA cluster systems. The parallel algorithm is cost-optimal with respect to the most efficient sequential algorithm. The experimental results shows that employing a good load balancing technique yields superior performance. The RNA secondary structure prediction algorithms mostly are DP algorithms. Our parallel DP algorithm skeleton can be used in other DP algorithms in computational biology by little modification.

References

1. E Rivas and S Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. J. Mol. Biol. 285:2053-2068. 1999

2. S Eddy and R Durbin. RNA sequence analysis using covariance models. Nucl. Acids Res. 22(11):2079-2088. 1994

3. S Eddy. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. BMC Bioinformatics, 3:18, 2002

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

5. Parallel Dynamic Programming. PhD dissertation by Phillip Gnassi Bradford, 1994.