行政院國家科學委員會專題研究計畫　成果報告

在叢集電腦上進行演化樹之建構與實作(II)

計畫類別：個別型計畫
計畫編號：NSC93-2213-E-216-037-
執行期間：93 年 08 月 01 日至 94 年 07 月 31 日
執行單位：中華大學資訊工程學系

計畫主持人：游坤明
共同主持人：唐傳義
計畫參與人員：周嘉奕、陳啟修、黃立明

報告類型：精簡報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 94 年 10 月 31 日

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告
□期中進度報告

在叢集電腦上進行演化樹之建構與實作(II)

計畫類別：■ 個別型計畫　　□ 整合型計畫

計畫編號：NSC 93－2213 －E －216－037

執行期間：　93年8月1日至94年7月31日

計畫主持人：游坤明
共同主持人：唐傳義
計畫參與人員：周嘉奕、陳啟修、黃立明

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告
本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
■出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、
　　　　　列管計畫及下列情形者外，得立即公開查詢
　　　　□ 涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：中華大學資訊工程系

中　華　民　國　94　年　8　月　30　日

在叢集電腦上進行演化樹之建構與實作(II)

游坤明 [1]、唐傳義 [2,]

[1] 中華大學資訊工程系
[2] 清華大學資訊工程系

摘要:
將各個物種間的演化關係透過一個距離矩陣的方式來建立最佳的演化樹，以得知物種間的演化相似程度如何，在生物計算學中是一個相當重要的課題。本計畫的主要目的為針對距離矩陣提供一個有效率且富使用者親和力的建構最佳等距演化樹的平行系統。在此計畫執行的第一年中，我們已進行探討以分支與界定演算法來建構最佳演化樹的平行化策略與方法，尋求一個適合在叢集電腦上執行的平行分支與界定方法來建構等距演化樹，並已獲得不錯的結果。在本計劃執行的第二年，我們進行更快速的近似最佳解的策略與方法之研究，並實際發展出一個利用緊湊集合(Compact Set)關係的演化樹平行建構演算法，以提供生物學家選擇較適合的演化樹作實際的應用，並且已整理成論文(A Fast Technique for Constructing Evolutionary Tree with the Application of Compact Sets)在 PaCT 2005 國際研討會中發表，同時亦收錄於 Lecture Notes in Computer Science, Vol. 3606, pp. 346-354, Springer-Verlag (SCI Expanded)中。
關鍵詞：演化樹、分支與界定、等距演化樹、緊湊集合、叢集電腦

Abstract:
Bioinformatics has becoming one of the major research topics in the 21st century. The constitution of computer technology and molecular biology technology is evidently essentially in the future. The problem of constructing evolutionary tree from distance matrix is an important issue in Bioinformatics. In the first year of this project, we have proposed an effective parallel algorithm for constructing an optimal ultrametric tree from a given distance matrix by using Branch-and-Bound technique. In the second year of the project, we have proposed a heuristic algorithm to shorten the revolutionary tree’s construction time. Also, we have developed an efficient algorithm for near-optimal ultrametric tree’s construction by applying the compact set technique.
Keyword : Evolutionary tree 、Branch and Bound、Ultrametric Tree、Compact set、PC Cluster

一、前言

近年來由於 DNA 序列的定序與分子生物學技術的進步，因此生物學家對於未知的遺傳因子與訊息有更進一步的瞭解，一些相關的研究也如火如荼的展開。其中將各個物種間的演化關係透過一個距離矩陣（Distance Matrices）的方式來建立最佳的（Optimal）演化樹(Phylogenetic Trees)，得知物種間的演化相似程度如何，在生物計算學中是一個相當重要的課題，演化樹的建構是依據物種(objects)所具有的生物特徵來建立距離矩陣，進而分辨物種之間的親疏遠近，以建立其相對應之演化樹，在一些相關的研究中（[3]，[8]，[9]，[10]），雖然提出了各種的策略與方法，但這些方法卻無法提供良好的執行效率以讓生物學家廣為

使用。Ultrametric Tree 亦是由距離矩陣所建構出來的，它是一棵有根樹（Rooted Tree），其樹葉（Leaf）代表了某一個物種，內部節點（Internal Node）代表在其下面物種的共同祖先（Ancestor），並且假設每個物種的演化速率相等。如此於 Ultrametric Tree 的演化假設下，所建立出的樹由其各內部節點至其所屬的 leaf 距離為等距。但同樣地，給一群物種間的距離矩陣建立最小的 Ultrametric Tree（Minimum Size Ultrametric Tree，MUT）已被證明是 NP 的問題[1]。因此，許多的研究都是利用 approximation algorithms 或 heuristic algorithms 去求得近似解，但跟 MUT 畢竟還是有一些誤差。在[5]中提出一個演算法來建立最小演化樹，其物種數目只能達到 11 個。而在[14]中則提出了新的演算法，在合理的時間範圍內，以及輸入的距離矩陣不同的限制下，能夠算到 12 至 20 個物種。

目前用來建構演化樹所使用到的演算法大都是使用『分支與界定』（Branch and Bound）的方式[14,15,16]。當處理的資料量不大時，單一處理器（即 Sequential Processing）尚能負荷其計算量；但是，當處理的資料量太多或太大時，則單一處理器的計算模式便會出現記憶體不足或無法在有效的時間內給出答案的窘境。這意謂著以目前的技術而言，想要在單一處理器上發展一個有效率的演算法是不大可行的。[14]提出了一個以分支與界定演算法來建構 MUT。但是其結構與設計上是適合於單機運作，然而要於合理的時間內得到答案的條件要求下，可以計算的物種數目會被系統硬體效能所限制而無法實際的應用。在第一年的計畫執行中，我們已提出一個有效率的平行化的分枝與界定演算法，在我們所設計的平行化分枝與界定演算法中，所有計算節點同時對他們所擁有的候選樹做分枝 (branch) 的動作，當計算節點發現候選樹符合界定(bound) 的條件時便不再對此候選樹做分枝的動作。而當計算節點得到更好的 upper bound 值時便會將此值傳遞給其他所有計算節點，其他所有計算節點得到新的 upper bound 值便可以界定掉(除掉)更多的候選演化樹。基於這個原因，在平行化系統的解集合會少於單一處理器的系統，亦可以讓演算法的效率大幅提昇，所以我們提出的平行化分枝與定界演算法在效率提昇(speedup) 上在一些例子上可能會達到 super-linear 的速度。在此平行化演算法中， 我們同時使用了 global pool 及 local pool 做為一種負載平衡的機制,讓計算節點不至於有閒置的情況發生。在此研究中我們採用 master / slave 的架構來建構平行化最小等距演化樹的系統，運算資料是在執行期間由 master 指派。

因此，在本計畫執行的第二年的主要目的便是針對 MUT 平行化的建構，發展出更有效率的平行化 MUT 建構的模式及策略，有效地加速演算法在處理 MUT 的物種個數及執行速度，並且探討近似最佳解的可能演算法。

二、研究目的

本計畫的主要目的為針對距離矩陣提供一個有效率且富使用者親和力的建構最佳演化樹的平行系統，在此計畫中，我們在叢集電腦的架構下探討以分支與界定的平行化演算法來建構最佳演化樹的平行化策略與方法，尋求一個適合在叢集電腦上執行的分支與界定方法來建構最佳演化樹，並且進行近似最佳解的演算法，並且實際發展一套有效率的演化樹平行建構軟體工具系統，以提供生物學家選擇較適合的演化樹作實際的應用，最後我們將此套工具系統透過 Web 介面，提供給從事此研究領域的專家學者使用。

三、文獻探討

[17]本計畫第一年的執行成果之一，在[17]我們提出了平行化的分支與界定演算法來建構 Minimum Size Ultrametric Tree 的演算法，在此演算法，中我們主要是利用兩兩物種間之距離關係計算出距離最大的兩個物種將其放在樹的最左右兩側，以建立芻型樹(只有兩個 leaf 的 binary tree，Branch-and-Bound Tree)，再將剩餘的物種一一插入在中間，並用 UPGMM（從 UPGMA 演算法改良而來)演算法計算權重值,當節點 v 產生的的 LB（Lower Bound）大於 UB（Upper Bound）時，表示後面產生的樹其結果比前述 UB 還差而不符合要求，則將節點 v 都刪除掉，以獲得最後的最小代價的 MUT。

在平行處理進行運算過程中，其影響整體效能（Efficiency）的重要因素乃為負載平衡（Load Balance)。若各運算單元負載不均，則會浪費很多寶貴的計算資源於閒置的運算處理單元。而利用分支與界定演算法進行解決問題的重點在於分支（Branch）與界定（Bound）的方法選擇決定上。於平行分散式系統中，往往會將原始問題分成數個可行解（Feasible solution）的子問題（Subproblem)，而分支後的一個或數個可行解分配給數個運算處理單元計算，因此分支的動作將會影響負載平衡問題。因此，我們在研究的進行中使用了 global pool 及 local pool 做為一種負載平衡的機制，讓計算節點不至於有閒置。而在我們的系統架構中，我們用的是 master / slave 的架構，並且資料是在執行期間由 master 來指派，以提高系統整體運算效率。

四、計畫成果自評

本計畫之執行之成果將能於較短的時間內建構出 MUT，不但能夠增進執行效率，還能夠大幅提昇 ultrametric tree 的正確性與結果的可讀性，本年度的計畫執行不但順利完成第二年度預期的計畫目標，亦已將第二年度所獲得的研究成果整理成論文並且發表了二篇國際研討會論文(一篇為 SCI Extended，另一篇為 IEEE 研討會)，一篇國內研討會論文(NCS 2005) ，並且亦已將所得之成果整理成期刊論文形式(Title : Efficient Parallel Branch-and-Bound Algorithm for Constructing Minimum Ultrametric Tress from Distance Matrices)投稿至 IEEE TPDS special issue on High Performance Biology，本年度的計畫執行成果可說是相當豐碩。

參考文獻

[1] W.H.E Day, Computational complexity of inferring phylogenies from dissimilarity matrices, *Bulletin of Mathematical Biology*,Vol.49, No.4, pp. 461-467, 1987.

[2] M. Farach, S. Kannan and T. Warnow, A Robust Model for Finding Optimal Evolutionary Trees, *Algorithmica*, **13** (1995), pp. 155-179.

[3] J.S. Farris, Estimating phylogenetic trees from distance matrices, *Am. Nat.* ,106, pp.645-668

[4] J. Felsenstein, Evolutionary trees from DNA sequences: a maximum likelihood approach. *ournal of Molecular Evolution*, 17:368-376, 1981.

[5] M.D. Hendy and D. Penny,Branch and bound algorithms to determine minimal evolutionary trees, *Mathematical Biosciences*, 59:277-290, 1982

[6] Chia-Mao Huang and Chang-Biau Yang. Approximation Algorithms for Constructing Evolutionary Trees. *In Proc. of National Computer Symposium, Workshop on Algorithm and Computation Theory*, pages A099-A109, Taipei, Taiwan, Dec. 20-21, 2001.

[7] P. Kearney, R.B. Hayward, R. B. and H. Meijer, Inferring Evolutionary Trees from Ordinal Data, *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA97), 1997, pp. 418-426.

[8] N. Saitou and M. Nei, The neighbor –joining method: A new method for reconstructing phylogenetic trees, *Molecular Biology and Evolution*, 4 (1987), pp.406 – 425

[9] P.H.A. Sneath and R.R. Sokal, *Numerical Taxonomy*, (Freeman, San Francisco, CA), 1973, pp 230-234.

[10] R. Sokal and P. Sneath, *Numerical Taxonomy*, San Francisco: Freeman, 359.,1963

[11] M. Steel, The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees, *Journal of Classification*, **9** (1992), pp. 91-116.

[12] L. Wang and T. Jiang, On the Complexity of Multiple Sequence Alignment, *Journal of Computational Biology*, **1** (1994), pp. 337-348.

[13] M. S. Waterman, T. F. Smith, M. Singh, and W. A. Beyer. “Additive evolutionary trees.” *Journal of Theoretical Biology*, 64:199-213, 1977.

[14] Wu, B.Y., Chao, K.M., and Tang, C.Y. (1999), “Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices”, *Journal of Combinatorial Optimization*, vol. 3, pp. 199-211.

[15] B.Y. Wu, G. Lancia, V. Bafna, K.M. Chao, R. Ravi and C.Y. Tang, A polynomial time approximation scheme for minimum routing cost spanning trees”, *SIAM J. Computing*, **29** (1999), pp. 761-778.

[16] B.Y. Wu and C.Y. Tang, An O(n) algorithm for finding an optimal position with relative distances in an evolutionary tree, *Information Processing Letters*, **63** (1997), pp. 263-269.

[17] Jia-Yi Zhou, Kun-Ming Yu, Chun-Yuan Lin, and Chuan Yi Tang, “Efficient Parallel Algorithm for Constructing Evolutionary Trees of Human Mitochondrial DNA from Distance Matrices,” The 2004 International Conference of Bioinformatics (InCoB 2004), pp. 35 (Sep. 3-6, 2004, Auckland, New Zealand). (NSC92-2213-E-216-011)

[18]蔡京甫、游坤明, “一個具有兩階段動態負載平衡機制之高效能計算環境,”二００四數位生活與網際網路科技研討會(2004 Symposium on Digital Life and Internet Technologies), pp. 5, 2004. (NSC92-2213-E-216-011)

附錄一：研討會論文

國際研討會論文

1. “A Fast Technique for Constructing Evolutionary Tree with the Application of Compact Sets,” PaCT 2005- Lecture Notes in Computer Science, Vol. 3606, pp. 346-354, Springer-Verlag, Sep. 2005. (PaCT'05), (SCI Expanded), (NSC-93-2213-E-216-037)
2. “Parallel Branch-and-Bound Algorithm for Constructing Evolutionary Trees from Distance Matrix,” The 8th International Conference and Exhibition on High-Performance Computing in Asia-Pacific Region (HPCAsia2005),Accepted, (NSC-93-2213-E-216-037).
3. Chun-I Chen, Chang Wu Yu, Ching-Hsien Hsu, Kun-Ming Yu, and C.-K. Liang, “Irregular Redistribution Scheduling by Partitioning Messages,” *Computer Systems Architecture - Lecture Notes in Computer Science*, Vol. 3740, pp. 295-309, Springer-Verlag, Oct. 2005. (ACSAC’05) (Oct. 24-26, 2005, Singapore). (SCI Expanded)

國內研討會論文

1. 游坤明、徐蓓芳、賴威廷、謝一功、周嘉奕、林俊淵、唐傳義, “應用網格建立一個高效能演化樹平行建構環境,” 九十四年全國計算機會議, NCS2005, Accepted, (NSC-93-2213-E-216-037).

附錄二：出席國際學術會議心得報告

A Fast Technique for Constructing Evolutionary Tree with the Application of Compact Sets*

Kun-Ming Yu[1,**], Yu-Wei Chang[1], YaoHua Yang[2], Jiayi Zhou[1], Chun-Yuan Lin[3,†], and Chuan Yi Tang[4]

[1] Department of Computer Science and Information Engineering, Chung Hua University
[2] Department of Information Management, Chung Hua University
[3] Institute of Molecular and Cellular Biology, National Tsing Hua University
[4] Department of Computer Science, National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
yu@chu.edu.tw

Abstract. Constructing an evolutionary tree has many techniques, and usually biologists use distance matrix on this activity. The evolutionary tree can assist in taxonomy for biologists to analyze the phylogeny. In this paper, we specifically employ the compact sets to convert the original matrix into several small matrices for constructing evolutionary tree in parallel. By the properties of compact sets, we do not spend much time and do keep the correct relations among species. Besides, we adopt both Human Mitochondrial DNAs and randomly generated matrix as input data in the experiments. In comparison with conventional technique, the experimental results show that utilizing compact sets can definitely construct the evolutionary tree in a reasonable time.
Keywords: computational biology, evolutionary tree, compact sets, branch-and-bound.

1 Introduction[1]

An evolutionary tree is a model of evolutional histories for a set of species. It is an important and fundamental model in bioinformatical field to observe livening species. A meaning evolutionary tree enhances biologists to evaluate the relationship of a set of species in taxonomy. Hence, many methods have been proposed to construct the evolutionary tree.

The majority of these methods are all based on two models, i.e., the sequences and a distance matrix. In the sequences model, they do multiple sequence alignment (MSA) for a set of species with corresponding DNA sequence first. Then an evolutionary tree is constructed according to the MSA result. However, the MSA problem is NP-hard. In a distance matrix model, they determine the distance as the edit distance for any two of species first. Then these distances are formed as a distance matrix. Finally, an evolutionary tree is constructed according to a distance matrix. Unfor-

* This work was supported in part by the NSC of ROC, under grant NSC93-2213-E-216-037.
** Corresponding author.
† Post doctor fellowship is supported by NSC under contract NSC92-3112-B-007-002 and NSC93-3112-B-007-008.

tunately, it is also an NP-hard problem to construct a minimum cost evolutionary tree from a distance matrix.

A category of evolutionary tree called ultrametric tree (UT) assumes that the rate of evolution is constant. An UT is a rooted and edge weighted binary tree in which every internal node has the same path length to all the leaves in its sub tree. The minimum UT for a distance matrix is an UT that the distance between any pair of leaves on the tree is no less than the given distance and the total weight on the tree edges is minimized.

In the distance matrix, shown in figure 1, each value represents the distance between two species. The distance matrix D is symmetric, i.e. for all $0 \leq i \leq n$, $D[i,i] = 0$. Also, the matrix D follows the triangle inequality, i.e. for all $1 \leq i, j, k \leq n$, $D[i,j] + D[j,k] \geq D[i,k]$.

	V1	V2	V3	V4	V5	V6
V1	0	3	1	12	6	13
V2		0	7	9	5	16
V3			0	11	4	15
V4				0	14	2
V5					0	8
V6						0

Fig. 1. An example of distance matrix

Some studies on constructing optimal evolutionary tree have been proven to be an NP-hard problem [3, 4, 6, 8, 9, 15]. The scientists could use the branch-and-bound technique to construct optimal evolutionary tree in a reasonable time [12] when the number of species is small. Although the branch-and-bound algorithm would detect an optimal solution, such capacities cannot effectively support the optimal evolutionary tree construction when the number of species exceeds 26.

In this paper, we specifically utilize the compact sets to convert the distance matrix into several small matrices for constructing an UT in parallel. We can not only obtain nearly optimal evolutionary tree but also keep the precise relations among species through compact sets by the property - the least common ancestor [14]. Of such an advantage, our work might contribute to the findings on the phylogeny.

The rest of the paper is organized as follows: section 2 proposes some preliminaries. Section 3 describes the methods for constructing the ultrametric tree in detail. The experimental results are presented in section 4. Finally, the conclusion is placed in section 5.

2 Preliminaries

An ultrametric tree is a rooted, leaf labeled binary tree, and each edge associates with a distance cost. The length from root to any leaf is equal. We can construct an UT through a distance matrix D representing a complete, weighted and undirected graph G. The graph $G = (V, E)$ includes vertices V and edges E. We give some definitions below:

Definition 1. Assume that P is a given topology and $i, j \in L(P)$. $LCA(i,j)$ represents the lowest common ancestor of i and j. Assume a and b are two vertices in P, we denote $a \longrightarrow b$ if and only if a is an ancestor of b.

Definition 2. Assume P is a tree topology. $R(P)$ is a relation - $\{(i,j,k\}|a,b,c\in L(P), LCA(i,k)=LCA(j,k) \longrightarrow LCA(i,j)\}$.

The compact set has been extensively studied [5] but have not been applied to the evolutionary tree construction problem. We will list some properties of compact sets below:

Lemma 1: Assume compact sets $\mathcal{C}$ exist in a tree T including elements i, j and k. The compact sets must satisfy a relation — least common ancestor. If and only if the relations $((i, j), k)$ and $LCA(i, j) < LCA(i, k) = LCA(j, k)$ exist, then there is an adjacency relation in T like figure 2.

Lemma 2: Let C be a subset of vertices V. If C is compact, then the maximum edge in C should be smaller than any edges between an element in C and that in V but not in C.

Lemma 3: Let A and B be two different compact sets of V_I. If A and B have intersection, then either $A \subset B$ or $B \subset A$[5].

Lemma 4: If sub graph g is compact set, then the sub tree in g also belongs to the minimum spanning tree $\overline{T}$.

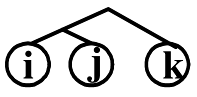


Fig. 2. An example of least common ancestor

3 Proposed Solutions

To construct nearly optimal UT for mass spices in reasonable time, we utilize the idea of compact set in our work. Firstly, we will find the compact sets from distance matrix D and explore them to create several small distance matrices D'. Then we input the smaller distance matrices D' to parallel branch-and-bound algorithm. Finally we can obtain sub trees T' and merge them into an ultrametric tree T. We describe the details in the subsection.

3.1 Compact Sets

As above, we explore compact sets to separate the distance matrix D into several small distance matrices D'. If the elements in a subset S of X are closer than those outside S but in X, then S is a compact set. Also we could continuously find compact sets in S until exploring all sub sets. In this work we can find all the compact sets to classify the organisms by collecting the more relative species on the graph[17]. The found

found groups will keep the correct relations and could conduce to analyze the phylogeny. Thus we utilize compact sets to construct a more precise ultrametric tree.

Initially we must find the minimum cost spanning tree to converge the closest groups and can probe the elements inside each group to discover all the compact sets. Take the figure 3 for example; if using the Kruskal's algorithm, we can locate a minimum spanning tree $\overline{T}$ like figure 4, and the compact sets are {(1,3),(4,6),(1,2,3,5)}. We will continue using the algorithm below to verify the subsets in $\overline{T}$ to discover all the compact sets.

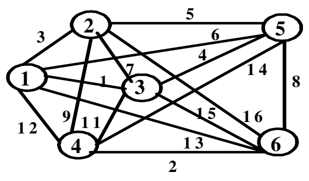


Fig. 3. The complete, weighted, undirected graph

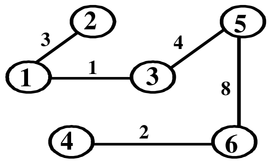


Fig. 4. The minimum spanning tree

```
Algorithm Compact Sets
Input: A graph $G$ = ($V$, $E$) with the vertex set $V$ ={$V_1$,
       $V_2$, …, $V_n$} and edge set $E$. Each edge has a weight.
Output: All of the compact sets on the graph $G$.
Step 1. Find the minimum spanning tree $\overline{T}$ on the graph
        $G$. )     //here we use Kruskal's algorithm.
Step 2. Sort the edges in $\overline{T}$ in ascending order, which is
        marked as ($e_1$, $e_2$, …, $e_{n-1}$).
Step 3. $P \leftarrow$ {{$V_1$}, {$V_2$,…,$V_n$}.
Step 4. for $i$ := 1 to $n$-2
        {
         1. Let $a$ and $b$ to be the end vertices of edge
            $e_i$, i.e., $e_i$ = ($a$, $b$).
         2. Find $A$, $B$ in $P$ such that $a$ belongs to $A$ and $b$
            belongs to $B$
         3. $A \leftarrow$ merge $A$ and $B$
         4. Delete $B$ from $P$
         5. Find the maximum edge in $A$, denoted Max($A$).
         6. Find the minimum edge between a vertex in $A$
            and a vertex not in $A$, denoted Min($A$, $!A$).
         7. If Max($A$) < Min($A$, $!A$), then $A$ is a compact
            set.
}
```

According to the algorithm, the order of edges is (1, 3), (4, 6), (1, 2), (3, 5) and (5, 6) after sorting by the weights. The population P includes all the vertices in $\overline{T}$, i.e. P = {(1), (2), (3), (4), (5), (6)}. We will firstly merge (1) and (3) together while coming to step 4. After the mergence, the P becomes {(1, 3), (2), (4), (5), (6)}. Continuously, we will find compact sets, (1, 3) and (4, 6). Worthy to be noticed is when we merge (1, 2) with (1, 3), we must examine if (1, 2, 3) satisfies the lemma 2. The maximum distance in (1, 2, 3) is less than the minimum distance between vertices in (1, 2, 3) and (4, 5, 6). Thus, (1, 2, 3) is a compact set. In the end, all the compact sets are (1, 3), (4, 6), (1, 2, 3) and (1, 2, 3, 5) like figure 5.

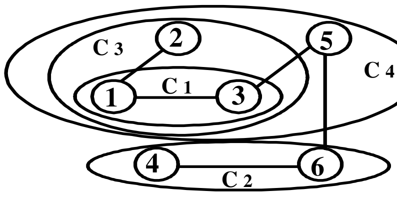


Fig. 5. Compact sets for the example

We then create several small distance matrices *D'* of three types which differ in the distance lengths stored in *D'*. These three matrices separately called *maximum*, *minimum*, and *average*. In this paper, we only study the ultrametric tree constructed from *maximum* matrix. The construction procedure is as follows. While creating the *maximum* matrix of C_4, we will examine the distances between elements in C_4, i.e. (C_1, C_3, 5). When considering C_3 and (5), we must select the maximum distance, which is 6, between (5) and any element in C_3, i.e. (1), (3) or (2). The resulted *maximum* matrix of C_4 shows in figure 7.

We shall discuss a situation that if there more than one $\overline{T}$ exists. In the previous step when finding $\overline{T}$, we need to examine and will obtain another $\overline{T}$ while replacing the edge of $\overline{T}$ with that holding the same weight on the graph. Indeed the new $\overline{T}$ should satisfy all conditions after the replacement. Figure 7(a) and (b) provides an example that two $\overline{T}$ *s* coexist in a graph.

Maximum

	C_1	C_2	5
C_1	0	7	6
C_2		0	6
5			0

Fig. 6. Maximum matrix of C_4

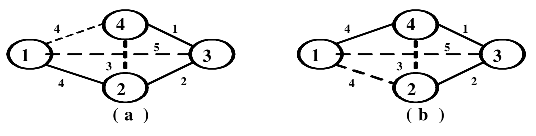


Fig. 7. Two minimum spanning trees in a graph

We can keep the precise relations among species by discovering all the compact sets on the graph. Thus we could ensure the relationship of every species in the ultrametric tree is precisely preserved by the characteristics of compact set. Then we can use the parallel branch-and-bound technique to construct an ultrametric tree from the small matrices *D'*. The following is an introduction to parallel branch-and-bound technique.

3.2 Parallel Branch-and-Bound Algorithm

We input several small distance matrices *D'* to the parallel branch-and-bound algorithm to find sub trees *T'*. Branch-and-bound algorithm is an efficient tree search algorithm for NP-hard problems. Some results about ultrametric trees have been proposed in [2]. In the previous researches, Wu et al., [19] had proposed a sequential branch-and-bound algorithm to construct minimum ultrametric trees from distance matrices.

For the parallel branch-and-bound algorithm, we utilize a heuristic algorithm UPGMM (Unweighted Pair Group Method with Maximum), which is altered from algorithm UPGMA [15], to find the cost values as bound values in our algorithm. If any computing nodes are notified that the branching unable to create any better solution, we then remove the branch. Compared with the single processor system, the solution space in the multi-processor system will decrease greatly. Thus, the parallel branch-and-bound algorithm could achieve super-leaner speedup.

The parallel branch-and-bound algorithm in the master and slave paradigm is listed as follows.

```
Parallel Branch-and-Bound Algorithm
Input: An $n * n$ distance matrix $D$.
Output: The minimum ultrametric tree for $D$.
Step 1: Master control node re-label the species such
        that (1, 2, …, $n$) is a maxmin permutation.
Step 2: Master control node creates the root of the BBT
        (branch-and-bound Tree).
Step 3: Master control node run UPGMM and using the
        result as the initial UB (upper bound).
Step 4: Master control node branches the BBT until the
        branched BBT reach 2 times of total nodes in
        the computing environment.
Step 5: Master control node broadcasts the global UB
        and send the sorted matrix the nodes cycli-
cally.
Step 6: while number of UTs in LP (Local Pools) > 0 or
        number of UTs in GP (Global Pools) > 0 do
          if number of UTs in LP = 0 then
            if number of UTs in GP <> 0 then
              receive UTs from GP
            end if
          end if
          $v$ = get the tree for branch using DFS
          if LB($v$) > UB then
            continue
          end if
          insert next species to $v$ and branch it
          if $v$ branched completed then
            if LB ($v$) < UB then
              update the GUB (Global Upper Bound) to
              every nodes
              add the $v$ to results set
            end if
          end if
          if number of UTs in GP = 0 then
            send the last UT in sorted LP to GP
          end if
        end while
Step 7: Gather all solutions from each node and output.
```

When obtaining the sub tree *T’* from the small matrix *D’*, each node will return it to the master control node. Finally, the master control node will merge all the sub trees *T’* into the ultrametric tree *T*.

4 The Experimental Results

The experimental environment is built by a Linux-based cluster incorporating one master control node and 16 computational nodes. Computational nodes have the same hardware specification and connect with each other at 100Mbps and 1Gbs to server. Human Mitochondrial DNAs and randomly generated species matrix are the data instances stored in the distance matrix. The experiments will process in two conditions: To construct ultrametric tree (1) with application of compact sets and (2) without utilizing compact sets. We will compare the differences in computing time and total tree cost. We can find compact sets on a graph and determine the *maximum* distances of elements in each compact set as the total tree cost while considering the ultrametric tree based on *maximum* matrix. The following experimental results of compact sets are shown based on the data of *maximum* matrix.

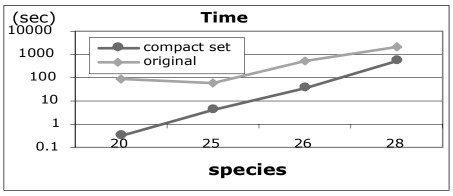


Fig. 8. The computing time for random data set

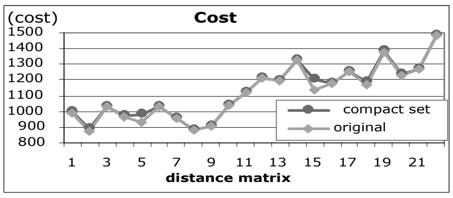


Fig. 9. The total tree cost for random data set

As the experiments on the randomly generated sequences, the averages computing time is shown in figure 8. Figure 8 illustrates the more species the more computing time we spend. In comparison with the method without applying compact set, the most time we save is about 99.7% and the least is 77.19% while using compact sets. Also we present the differences in cost between condition 1 and 2 in figure 9 and the results are based on randomly generated sequences. Figure 9 illustrates the total tree costs under two conditions are almost equal and the difference is less than 5%.

As the experiments on Human Mitochondrial DNAs, we use 15 data set containing 26 species for each and the total tree cost is presented in figure 10. The results show the maximum difference is 1.5%. In other words, the results demonstrate compact sets have the same effect not only on generated sequences but also on Human Mitochondrial DNAs. Figure 11 shows the computing time. Using compact sets can definitely save time but unexpectedly the experiments without compact sets also take little time except the last data.

We also experiment with 30 DNAs and figure 12 represents the costs of 10 data set each including 30 DNAs. As figure 12, using compact sets could keep the cost down when we experiment on 30 DNAs as well as generated data or 26 DNAs. According to figure 13, for computing time, the performances of the experiments on both 26 and 30 DNAs are alike.

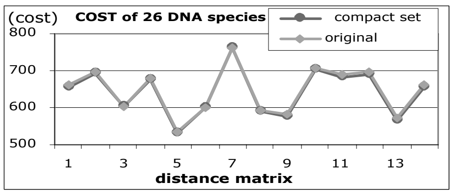


Fig. 10. The total tree cost for 26 DNAs

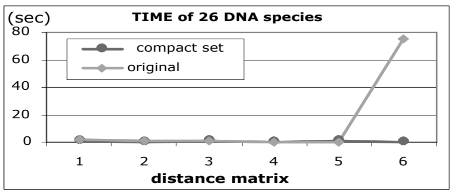


Fig. 11. The computing time for 26 DNAs

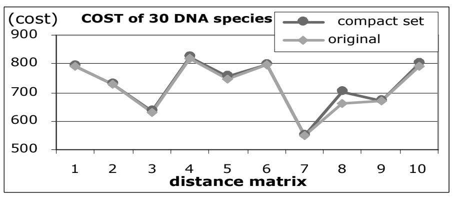


Fig. 12. The total tree cost of 30 DNAs

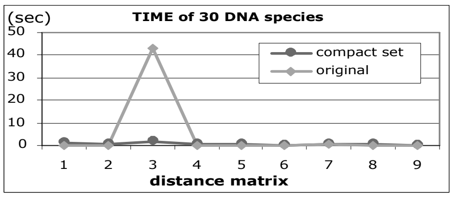


Fig. 13. The computing time of 30 DNAs

No matter how many species on which we experiment, the computing speed is still extremely rapid without using compact sets. Although the experiments using compact sets do not take much less time, we suppose the phenomenon is relevant to the population of the data. The computing time resulted from the experiment with randomly generated data can be a reference for any circumstance.

5 The Conclusions

In this paper, we employ the compact sets to convert the original matrix into several small matrices for constructing ultrametric tree in parallel. Of the compact sets, the precise phylogeny remains and facilitates biologists to analyze the species in taxonomy. Although we experiment with both Human Mitochondrial DNAs and randomly generated sequences, the results from generated data can represent any real instance. Therefore our technique could be applied in any condition.

References

1. H.J. Bandelt, “Recognition of tree metrics,” *SIAM Journal on Discrete Mathematics*, vol. 3, no. 1, pp.1-6, 1990.
2. E. Dahlhaus, “Fast parallel recognition of ultrametrics and tree metrics,” *SIAM Journal on Discrete Mathematics*, vol. 6, no. 4, pp. 523-532, 1993.
3. W.H.E. Day, ”Computationally difficult parsimony problems in phylogenetic systematics,“ Journal of Theoretic Biology, vol. 103, pp. 429-438, 1983.
4. W.H.E. Day, “Computational complexity of inferring phylogenies from dissimilarity matrices,” Bulletin of Mathematical Biology, vol. 49, no. 4, pp. 461-467, 1987.

5. E. Dekel, J. Hu, and W. Ouyang. An optimal algorithm for finding compact sets. Information Processing Letters, 44:285-289, 1992.
6. M. Farach, S. Kannan, and T. Warnow, “A robust model for finding optimal evolutionary trees,” *Algorithmica*, vol. 13, pp. 155-179, 1995.
7. W.M. Fitch, “A non-sequential method for constructing trees and hierarchical classifications,” *Journal of Molecular Evolution*, vol. 18, pp. 30-37, 1981.
8. L.R. Foulds, “Maximum savings in the Steiner problem in phylogency,” *Journal of theoretic Biology*, vol. 107, pp.471-474, 1984.
9. L.R. Foulds and R.L. Graham, “The Steiner problem in phylogeny is NP-complete,” *Advances in Applied Mathematics*, vol. 3, pp. 43-49, 1982.
10. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman: San Fransisco, 1979.
11. D. Gusfield, “Algorithms on Strings, Trees, and Sequences, computer science and computational biology,” *Cambridge University Press, 1997*.
12. M.D. Henry and D. Penny, ”Branch and bound algorithms to determine minimal evolutionary trees,” *Mathematical Biosciences*, vol. 59, pp. 277-290, 1982.
13. R.M. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher (Eds.), Plenum Press: New York, 1972, pp. 85-103.
14. SungKwon Kim, “A note on finding compact sets in graphs represented by an adjacency list” Information Processing Letters, vol. 57, pp. 335-238, 1996.
15. M. Krivanek, “The complexity of ultrametric partitions on graphs,” *Information Processing Letter*, vol. 27, no. 5, pp. 265-270, 1988.
16. W.H. Li and D.Graur, Fundamentals of Molecular Evolution, Sinauer Associates, 1991.
17. Chiou-Kuo Liang, "An O(n^2) Algorithm for Finding the Compact Sets of a Graph," BIT, vol. 33, pp 390-395, 1993.
18. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular Biology and Evolution, 4:406-425, 1987.
19. B.Y. Wu, K.M. Chao, C.Y. Tang, “Approximation and Exact Algorithms for Constructing minimum Ultrametric Tree from Distance Matrices,” Journal of Combinatorial Optimization, vol. 3, pp.199-211, 1999.

Parallel Branch-and-Bound Algorithm for Constructing Evolutionary Trees from Distance Matrix [*]

Kun-Ming Yu[1 †], Jiayi Zhou[1], Chun-Yuan Lin[2 ‡], and Chuan Yi Tang[3]

[1]Department of Computer Science and Information Engineering, Chung Hua University
[2]Institute of Molecular and Cellular Biology, National Tsing Hua University
[3]Department of Computer Science, National Tsing Hua University
Hsinchu, Taiwan 300, ROC

[1] yu@chu.edu.tw, jyzhou@pdlab.csie.chu.edu.tw
[2] cyulin@mx.nthu.edu.tw
[3] cytang@cs.nthu.edu.tw

Abstract

An ultrametric tree is an evolutionary tree in which the distances from the root to all leaves in the tree are equal. The Minimum Ultrametric Tree construction problem is the problem of constructing an ultrametric tree from distance matrices with minimum cost. It is shown that to construct a minimum cost ultrametric tree is NP-hard. In this paper, we present an efficient parallel branch and bound algorithm to construct a minimum ultrametric tree with less cost. The experimental results show that our proposed algorithm can discover optimal solutions for 38 species within reasonable time with 16 computing nodes.

Keyword: Parallel computing, branch-and-bound, evolutionary tree, distance matrices, minimum ultrametric trees.

1. Introduction

An evolutionary tree is a model of evolutional histories for a set of species. It is a very important and fundamental model in computational biology field to observe livening species. A meaning evolutionary tree is helpful for biologists to evaluate the relationship of a set of species in taxonomy.

However, it is hard to know the constructed evolutionary tree is meaning or not since the real evolutionary process is unknown. Hence, many methods have been proposed to construct the evolutionary tree.

The majority of these methods are all based on two models, the sequences and the distance matrix. In the sequences model, researchers do multiple sequence alignment (MSA) for a set of species with corresponding DNA sequence first. Then an evolutionary tree is constructed according to the MSA result. However, the MSA problem is NP-hard. In the distance matrix model, they calculate the distance as the edit distance for any two of species first. Then these distances are formed as a distance matrix. Finally, an evolutionary tree is constructed according to a distance matrix. Unfortunately, it is also a NP-hard problem to construct a minimum cost evolutionary tree from a distance matrix.

Some heuristic algorithms, such as Unweighted Pair Group Method with Arithmetic Mean (UPGMA) and Neighbor Joining Method, have been proposed and popularly used by biologists. However, the constructed evolutionary tree from them is not optimal. Moreover, it is still worthy to construct an optimal evolutionary tree for a set with small number of species.

There is a category of evolutionary tree called ultrametric tree, in which we assume that the rate of evolution is constant. An ultrametric tree is a rooted and edge weighted binary tree in which every internal node has the same path length to all the leaves in its sub tree. However, the number of an ultrametric tree $A(n)$ grows very rapidly when the number of species n increases. For example, $A(20) > 10^{21}$, $A(25) > 10^{29}$,

[*] This work was supported in part by the NSC of ROC, under grant NSC-93-2213-E-216-037
[†] Corresponding author
[‡] Post doctor fellowship is supported by NSC under contract NSC92-3112-B-007-002 and NSC93-3112-B-007-008

$A(30) > 10^{37}$. The problem of constructing a minimum ultrametric tree has been shown to be NP-hard. The branch-and-bound algorithms are very well-known techniques to avoid exhaustive search. It is a partition algorithm to decompose a problem into smaller subproblems and then repeatedly decomposes them until infeasibility is proved or a solution is found [17]. Theoretically, a branch-and-bound algorithm cannot ensure polynomial time complexity in the worst case. However, it has been used to solve some NP-hard problems, such as *Traveling Salesman, Knapsack, Vertex Covering, Integer Programming*, and so on [17]. In addition, a branch-and-bound algorithm can often find the near optimal solutions as well as an optimal one.

In our previous work, we have proposed a parallel branch-and-bound technique to construct a metric minimum ultrametric tree. Our technique can drastically reduce the solution space. However, it is not enough to construct a metric minimum ultrametric tree with a numerous number of species. In this paper, we utilize the concept of 3-3 relationship in our proposed parallel branch-and-bound algorithm to reduce the solution space and may reduce the execution time significantly.

This paper is organized as follows. In Section 2, we introduce the metric minimum ultrametric tree problem and the 3-3 relationship. Section 3 describes the proposed parallel branch-and-bound algorithm with the 3-3 relationship. The experimental results and the conclusions will be given in Sections 4 and 5.

2. Related Work

Most of the optimization problems for evolutionary tree construction are NP-hard [3, 7, 12, 15]. There are many models of evolutionary tree and one of them is called ultrametric tree (UT) which assumes the rate of evolution is constant [310, 1]. A UT is a rooted, leaf labeled, and edge weighted binary tree in which every internal node has the same path length to all the leaves in its sub-tree [15]. Distance matrix is most frequently used to construct an evolutionary tree. For an n by n distance matrix M, the minimum UT for M is an UT that the distance between any pair of leaves on the tree is no less than the given distance and the total weight on the tree edges is minimized. There are some results about UT which have been presented in [1, 4, 7, 15].

As it is an NP-hard problem to construct a minimum ultrametric tree from distance matrix, branch-and-bound technique is a good candidate to reduce the solution space effectively. Wu et al., [15] proposed a sequential branch-and-bound algorithm for constructing minimum ultrametric trees from distance matrices. We denote their algorithm as Algorithm BBU for brevity. Initially, Algorithm BBU uses a heuristic algorithm UPGMM (Unweighted Pair Group Method with Maximum), which modifies from algorithm UPGMA, to find a feasible solution. Then, Algorithm BBU repeatedly searches the branch-and bound tree (BBT) for better solutions until an optimal solution is found. For any node, say v, in the BBT, compute the value of LB(v), which is a lower bound on the weight of any ultrametric tree. Below is a formal description of Algorithm BBU.

```
Algorithm BBU

Input: An n x n distance matrix M.
Output: The minimum ultrametric tree for M.

Step 1: Relabel the species such that (1, 2, …, n) is
maxmin permutation.
Step 2: Create the root v of the BBT such that v
        represents the only topology with leaves 1 and
        2.
Step 3: Run UPGMM to find a feasible solution and
        store its weight in UB (the weight of current
        best UT).
Step 4:
while there is a node in BBT do
        if LB(v) >= UB or all the children of v have
been deleted then
            delete all nodes v from BBT
        end if
        Select a node s in BBT, whose children has
not been generated.
        Generate the children of s by using the
branching rule.
        if find a better solution then
          update UB
        end if
end while.
```

The readers can refer to [15] for the correctness and time complexity issues of algorithm BBU.

In this paper, G = (V, E) represents an unweighted graph with vertex set V and edge set E and G = (V, E, w) denotes an edge weighted graph. To simplify the presentation, notations and terminologies used in this paper are prior defined as follows.

Definition 1: A distance matrix of n species is a symmetric n×n matrix M such that *M[i,* *j]* $\geq$ *0* for all $0 \leq$ i, j $\leq$ n, and *M[i, i]=0* for all $0 \leq$ i $\leq$ n [15].

Definition 2: A *M* is a *metric* if the distances obey the triangle inequality, i.e., *M[i, j]+M[j, k]* $\geq$ *M[i, k]* for all $1 \leq i, j, k \leq n$ [15].

Definition 3: A metric M is an *ultrametric* if and only if *M[i, j]* $\leq$ *max{M[i, k], M[j, k]}* for all $1 \leq$ i, j, k $\leq$ n [2].

Definition 4: Let T = (V, E, ω) be an edge weighted tree and u, v $\in$ V. The path length from u to v is denoted by $d_T(u,v)$. The weight of T is defined by ω(T)= $\sum_{e\in E} \omega(e)$ [15].

Definition 5: Let T be a rooted tree and r be any node of T. We use *Tr* to denote the subtree rooted at r, and *L(T)* to denote the leaf set of *T* [15].

Definition 6: An *ultrametric tree T* of {1, …, n} is a rooted and edge-weighted binary tree with *L(T)* = {1, …, n} and root r such that $d_T(u,r)$ = $d_T(v,r)$ for all u, v $\in$ L(T) [15].

Definition 7: Let T = (V, E, ω) be an UT. For any r $\in$ V, the height of r, denoted by height(r), is the distance from r to any leaf in the subtree *Tr*, i.e., *height(r)* $= d_T(r,v)$ for any v $\in$ L(Tr) [15].

Definition 8: For any M, *MUT* for M is T with minimum ω(T) such that L(T)={1, …, n} and $d_T(i,j) \geq$ *M[i, j]* for all $1 \leq$ i, j $\leq$ n. The problem of finding MUT for M is called *MUT problem* [7].

Definition 9: The *metric minimum ultrametric tree* (Δ *MUT*) problem has the same definition as *MUT* problem except that the input is a metric [15].

Theorem 1: *The* Δ*MUT problem is NP-hard* [15].

*Definition 10:*Let P be a topology, and $a, b \in L(P)$. $LCA(a,b)$ denotes the lowest common ancestor of a and b. If x and y are two nodes of P, we write $x \rightarrow y$ if and only if x is an ancestor of y.

*Definition 11:*We denote the distance between distance matrix and rooted topology of evolutionary trees is consistent if $M[i,j] < \min\{M[i,k], M[j,k]\}$ *if and only if* $LCA(i,j) < LCA(i,k) = LCA(j,k)$ for any $1 \leq i, j, k \leq n$. Otherwise is *contradictory.*

Fan [5] proposed an idea to evaluate the evolutionary trees by using distance relations between distance matrix and evolutionary trees for any 3 species. The idea was as follows, choosing three species i, j, k arbitrary, if i, j relates closely in distance matrix, then on evolutionary trees should also present relation of i, j. Otherwise, it is contradiction, if the number of contradictions is more, expresses the method of evolutionary tree construction is insufficiently good, and it cannot faithfully reflect the relation of the original distance matrix.

For the purpose of reducing the solution space in branch-and-bound strategy; we observe that the characteristic of 3-3 relationship between distance matrix and evolutionary tree can be utilized.

3. Main title

In this section, we will describe the system framework we developed in detail, including parallel algorithm, load balancing strategy, data structure, and how to use 3-3 relationship to construct evolutionary trees.

The same level of evolutionary tree can be divided into independent parts, therefore parallel branch-and-bound is a very suitable technique to solve evolutionary tree problem without considering the data-dependent problem between computing nodes. Each computing node only needs to handle or solve a sub-problem with sequential algorithm regardless of data-dependent problem.

In our proposed parallel branch-and-bound algorithm, every node in the same level of branch-and-bound tree represents respective solution. Every computing node branches one of the nodes in the same time. When some computing nodes find the branching solution satisfies the bounding rule then we don’t need to branch any more. It will pass a message to notify other computing nodes that the branching will not produce a better solution and then we can delete the branch. For this reason, the solution space in multi-processor system will be less than the solution space in the single processor system. Thus, our proposed parallel branch-and-bound algorithm may achieve super-linear speedup.

The load balancing strategy is important in our proposed system. Because the solution space in each computing node may differ a lot after bounding, this may result in the situation that some computing nodes idle. We apply the global pools design, which located in the Master processor. When local pools of computing nodes empty, it can request some branching data from global pools if it is not empty. Even through the global pools empty, it will poll branching data form the heavily loaded computing nodes.

The data structure is also an important issue in the parallel computing. An unsuitable data structure may take unnecessary time during the exchange of information between computing nodes so that we shall consider whether the data structure performs well in parallel computing. Therefore, we develop a data structure, which is called UT node, including every internal node's left children, right children, parents, leaves which were sorted by array and the UT node's low bound. All necessary information is stored in a branch and bound tree (BBT) which combined with UT.

In the proposed algorithm, the master processor (MP) will create initial nodes and then dispatch most of them to slave computing processors. The MP is also used to do the same work in slave computing processors and try to balance the nodes among MP and slave computing processors.

In MP, in Step 1, it reorders the input metric distance matrix M to form a max-min permutation and then re-label the species as a leaf set {1, 2, …, n}. This work could be done in parallel. In Step 2, a root v of BBT is created by MP which v represents the only topology with leaves 1 and 2. In Step 3, MP will run UPGMM to find a feasible solution and store its weight in a global variable UB as an initial upper bound. In the Step 4, MP applies the 3-3 relationship constrain to insert the third species, which can reduce the solution space significantly. In order to dispatch nodes to slave computing processors, some nodes of BBT should be generated. Therefore, in Step 5, MP will do parts of Step 5 in BBU to generate some nodes of BBT. Note that the value of LB(v) for each node v generated by MP is lower than or equal to UB. Now, the number of nodes is set to be double of the number of processors p. Similarly, this step could be done in parallel, but it is done by MP with the same reason.

Since each node v in each slave computing processor may be bounded quickly or not, we try to balance the work among processors before the dispatching procedure. In Step 6, for each node v generated by MP, a global UB is computed first, and then broadcasts to slave computing processors. According to the sorting results, each corresponding node will be stored sequentially into the Global pool (GP). Afterward, MP dispatches most of them to slave computing processors by the cyclic partition method. In the dispatching procedure, UB and M with a max-min permutation are also sent to Slave computing processors. Since MP is also used to do the same work in Slave computing processors, it needs to preserve some nodes in GP. Now, MP preserves 1/p nodes in GP. By Step 7, a potential effect may be existed to balance the work among MP and slave computing processors. After dispatching most of nodes from MP to Slave computing processors, parallel branch-and-bound algorithm tries to find the optimal solution.

The parallel branch-and-bound algorithm in the master-slave paradigm is presented as follows.

Table 1 Parallel Branch-and-Bound with 3-3 Relationship

```
Input: An n x n distance matrix M.
Output: The minimum ultrametric tree for M.

Step 1: Master processor re-label the species such that
        (1, 2, …, n) is a maxmin permutation.
Step 2: Master processor creates the root of the BBT.
Step 3: Master processor run UPGMM and using the
        result as the initial UB (upper bound).
Step 4: Under rooted base tree with 2 species.
        Referring the original distance matrix to insert
        the third species according to the 3-3
        relationship constraint.
Step 5: Master processor branches the BBT until the
        branched BBT reaches 2      times of total
        nodes in the computing environment.
Step 6: Master processor broadcasts the global UB and
        sends the sorted matrix the slave computing
        processors cyclically.
Step 7:
while number of UTs in LP (Local Pools) > 0 or
      number of UTs in GP (Global Pools) > 0 do
    if number of UTs in LP = 0 then
        if number of UTs in GP <> 0 then
            receive UTs from GP
        end if
    end if
    v = get the tree for branch using DFS
    if LB(v) > UB then
        continue
    end if
    Insert next species to v and branch it
    if v branched completed then
        if LB (v) < UB then
            Update the GUB (Global
Upper Bound) to every node
```

```
                Add the v to results set
            end if
        end if
        if number of UTs in GP = 0 then
            Send the last UT in sorted LP to GP
        end if
end while
Step 8: Gather all solutions from each node and output the optimal solution.
```

4. Experimental Results

The experimental environment is built by a Linux-based cluster; it consisted of one Master processor and 16 slave computing processors. All slave computing nodes have the same hardware specification and connected with each other at 100Mbps and 1Gbs to server. One computing node (single processor) is designated as the sequential platform in contrast with the parallel computation.

The data instances we used are the distance matrix constructed from Human Mitochondrial DNA (HMDNA), and each number of species we run 20 instances to reduce the factor influenced by distance matrix.

The computing time for 16 slave computing nodes and single node is shown in figure 1 and 2. From figure 1 and 2, we can observe that our proposed parallel algorithm is effective when the number of species is getting large. Also, we can observe that the computing time will be unendurable when the number of species greater than 26 for single processor. On the other hand, the parallel branch-and-bound algorithm can find optimal ultrametric tree within reasonable time for 38 species. The speedup ratio is shown in Figure 3, and we can find our proposed parallel branch-and-bound algorithm achieve super linear speedup ratio. Figure 4 depicts that 3-3 relationship can reduce computing time when number of species grows. Also, in our experimental results, the result trees with 3-3 relationship are a subset of result without 3-3 relationship. It indicates that applying 3-3 relationship can not only reduce the solution space but also have the same results.

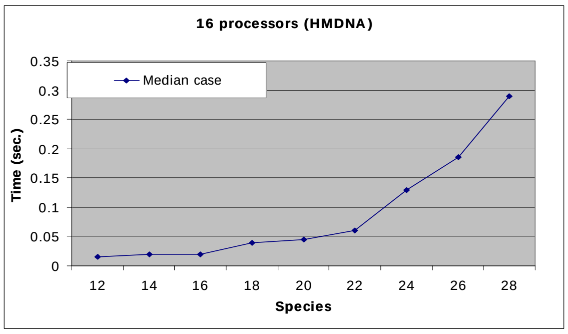


Figure 1 The computing time for 16 processors, HMDNA.

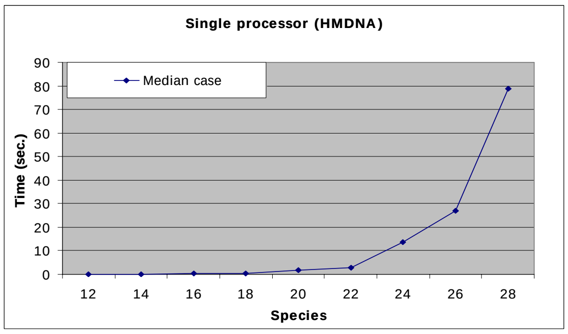


Figure 2 The computing time for single processor, HMDNA.

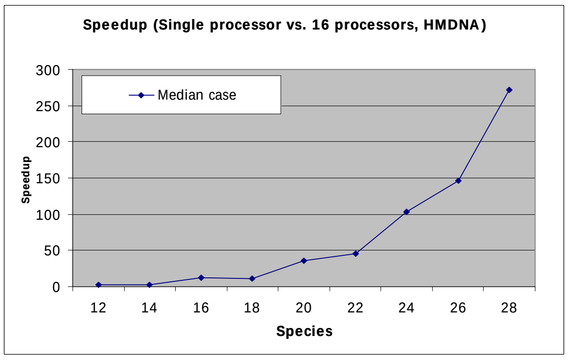


Figure 3 Speedup (16 processors vs. single processor, HMDNA).

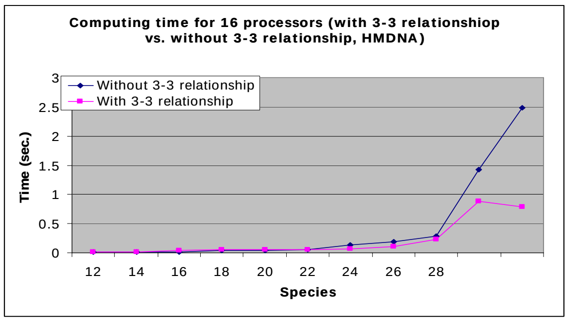


Figure 4 The computing time for 16 processors (with 3-3 relationship vs. without 3-3 relationship, HMDNA).

Figure 5, 6, 7 and 8 show the computing time as well as speedup ratio for randomly generated data sample set, the range of the data values is from 0 to 100. Also, our proposed algorithm has supreme performance and can obtain optimal evolutionary tree within reasonable time. Our proposed parallel branch-and-bound algorithm can achieve super linear speedup ratio.

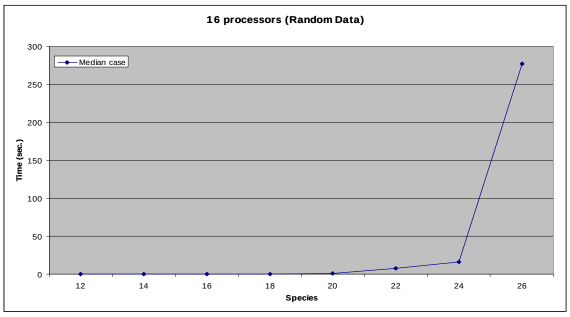


Figure 5 The computing time for 16 processors, Random Data.

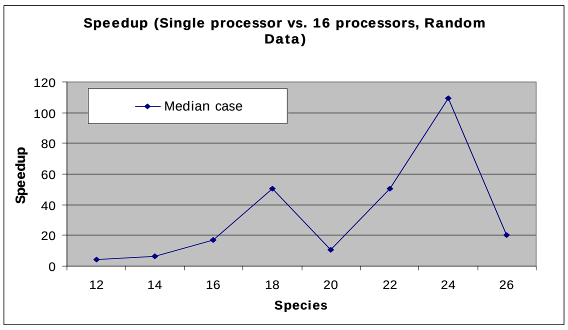


Figure 6 Speedup (16 processor vs. single processor, Random Data).

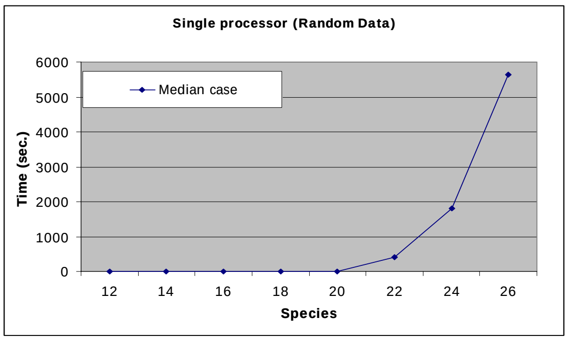


Figure 7 The computing time for single processor, Random Data.

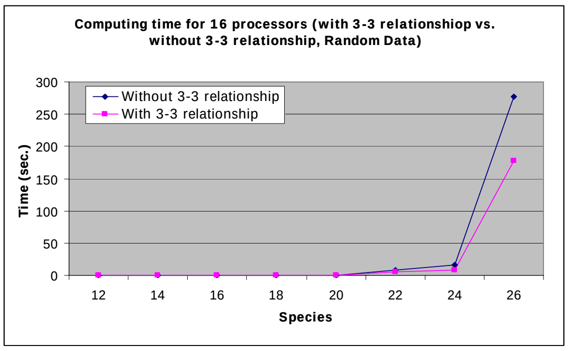


Figure 8 The computing time for 16 processors (with 3-3 relationship vs. without 3-3 relationship, Random Data).

5. Conclusions

In this paper, we have proposed a parallel branch-and-bound algorithm that runs in a master/slave paradigm to resolve the minimum ultrametric trees construction problem, and we adopt the 3-3 relationship in our algorithm. Experimental results show that the performance of our algorithm, running on a personal computer cluster with 16 slave computing processors, is extraordinary in comparison with single processor. Moreover, our proposed parallel algorithm can find an optimal solution for 38 species within reasonable time. To the best of our knowledge, there are no reported algorithms which can find the optimal ultrametric tree with the number of species exceeding 25.

From experimental results, we can see that the performance of the sequential and parallel algorithms will be influenced by the number of species, the number of processors and the distance matrix. (Hint: different distance matrices with the same number of species lead to different performance). With 3-3 relationship, we found it can reduce the computing time when number of species grows, but we only used it in the initial step. In our future work, we can extend this feature and speedup the process of constructing evolutionary trees.

References

[1] H.J. Bandelt, “Recognition of tree metrics,” SIAM Journal on Discrete Mathematics, vol. 3, no. 1, pp.1-6, 1990.

[2] E. Dahlhaus, “Fast parallel recognition of ultrametrics and tree metrics,” SIAM Journal on Discrete Mathematics, vol. 6, no. 4, pp.523-532, 1993.

[3] W.H.E. Day, D.S. Johnson and D. Sankoff, “The computational complexity of inferring rooted phylogenies by parsimony,” Mathematical Biosciences, vol. 81:33-42, 1986.

[4] W.H.E. Day, “Computation complexity of inferring phylogenies from dissimilarity matrices,” Bullotin of Mathematical Biology, vol. 49, no. 4, pp. 461-467, 1987.

[5] Chen-Tai Fan, “The evolved tree appraises the pattern the establishment and applies, “ Master Thesis, National Tsing Hua University, 2000.

[6] L.R. Foulds and R.L. Graham, “The Steiner problem in phylogeny is NP-complete,” Advances in Applied Mathematics, vol. 3, pp. 43-49, 1982.

[7] M. Frach, S. Kannan, and T. Warnow, “A robust model for finding optimal evolutionary trees,” Algorithmica, vol. 13, pp.155-179, 1995.

[8] L.R. Foulds, “Maximum savings in the Steiner problem in phylogeny,” Journal of theoretic Biology, vol. 107, pp. 471-474, 1984.

[9] L.R. Foulds and R.L. Graham, “The Steiner problem in phylogeny is NP-complete,” Advances in Applied Mathematics, vol. 3, pp. 43-49, 1982.

[10] D. Gusfield, “Algorithms on Strings, Trees, and Sequences, computer science and computational biology,” Cambridge University Press, 1997

[11] M.D. Hendy and D. Penny, “Branch-and-bound algorithms to determine minimal evolutionary trees,” Mathematical Biosciences, vol. 59, pp. 277-290, 1982.

[12] M. Krivanek, “The complexity of ultrametric partitions on graphs,” Information Processing Letter, vol. 27, no. 5, pp. 265-270, 1988.

[13] W.H. Li and D. Graur, “Foundomentals of Molecular Evolution,” Sinauer Associates, 1991.

[14] Yuji Shinano, Kenichi Harada and Ryuichi Hirabayashi,” Control Schemes in a Generalized Utility for Parallel Branch-and-Bound Algorithms,’’ Parallel Processing Symposium Proceedings, pp. 621 –627, 1997.

[15] B.Y. Wu, K.M. Chao, C.Y. Tang, “Approximation and Exact Algorithms for Constructing Minimum Ultrametric Tree from Distance Matrices,” Journal of Combinatorial Optimization, vol 3, pp. 199-211, 1999.

[16] Albert Y. Zomaya, Senior Member, IEEE, and Yee-Hwei Teh, “Observations on Using Genetic Algorithms for Dynamic Load-Balancing,” IEEE Transaction on Parallel and Distributed Systems, vol. 12, no. 9, pp.899-911, 2001.

[17] C.F. Yu and B.W. Wah, “Efficient Branch-and-Bound Algorithms on a Two-Level Memory System,” *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no.9, 1988, pp. 1342-1356.

Contention-Free Communication Scheduling for Irregular Data Redistribution in Parallelizing Compilers*

Kun-Ming Yu, Chi-Hsiu Chen, Ching-Hsien Hsu, Chang Wu Yu, and Chiu Kuo Liang

Department of Computer Science and Information Engineering,
Chung Hua University, Hsinchu, Taiwan 300, ROC
Tel: 886-3-5186412, Fax: 886-3-5329701
yu@chu.edu.tw

Abstract. The data redistribution problems on multi-computers had been extensively studied. Irregular data redistribution has been paid attention recently since it can distribute different size of data segment of each processor to processors according to their own computation capability. *High Performance Fortran Version 2* (HPF-2) provides *GEN_BLOCK* data distribution method for generating irregular data distribution. In this paper, we develop an efficient scheduling algorithm, Smallest Conflict Points Algorithm (SCPA), to schedule HPF2 irregular array redistribution. SCPA is a near optimal scheduling algorithm, which satisfies the minimal number of steps and minimal total messages size of steps for irregular data redistribution.

Keywords: Irregular data redistribution, communication scheduling, GEN_BLOCK, conflict points.

1 Introduction

More and more works had large data or complex computation on run-time in most scientific and engineering application. Those kinds of tasks require parallel programming on distributed system. Appropriate data distribution is critical for efficient execution of a data parallel program on a distributed computing environment. Therefore, an efficient data redistribution communication algorithm is needed to relocate the data among different processors. Data redistribution can be classified into two categories: the regular data redistribution [2, 3, 6] and the irregular data redistribution [1, 4, 10, 11, 12]. The irregular distribution uses user-defined functions to specify unevenly data distribution. High Performance Fortran version 2 (HPF2) provides GEN_BLOCK data distribution instruction which facilitates generalized unequal-size consecutive segments of array mapping onto consecutive processors. This makes it

* The work is partially supported by National Science Council of Taiwan, under grant number NSC-93-2213-E-216-029.

possible to let different processors dealing with appropriate data quantity according to their computation capability. In this scenario, all processors must send and receive message, even if send and receive on the same processor.

In the irregular array redistribution, *Guo et al.* [11] proposed a Divide-and-Conquer algorithm, they utilize Divide and Conquer technique to obtain near optimal scheduling while satisfied minimize the total communication messages size and minimize the number of steps.

In this paper, we present a smallest-conflict-points algorithm (SCPA) to efficiently perform GEN_BLOCK array redistribution. The main idea of the SCPA is to schedule the conflict messages with maximum degree in the first step of data redistribution process. SCPA can effectively reduce communication time in the process of data redistribution. SCPA is not only an optimal algorithm in the term of minimal number of steps, but also a near optimal algorithm satisfied the condition of minimal message size of total steps.

The rest of this paper is organized as follows. In Section 2, a brief survey of related work will be presented. In section 3, we will introduce communication model of irregular data redistribution and give an example of GEN_BLOCK array redistribution as preliminary. Section 4 presents smallest-conflict-points algorithm for irregular redistribution problem. The performance analysis and simulation results will be presented in section 5. Finally, the conclusions will be given in section 6.

2 Related Work

Many data redistribution results have been proposed in the literature. These researches are usually developed for regular or irregular problems [1] in multi-computer compiler techniques or runtime support techniques.

Techniques for communication optimizations category provide different approaches to reduce the communication overheads [5, 7] in a redistribution operation. The communication scheduling approaches [3, 12] avoid node contention and the strip mining approach [9] overlaps communication and computational overheads.

In irregular array redistribution problem, some works have concentrated on the indexing and message generation while some has addressed on the communication efficiency. Guo et al. [10, 11] proposed a divide-and-conquer algorithm for performing irregular array redistribution. In this method, communication messages are first divided into groups using Neighbor Message Set (NMS), messages have the same sender or receiver; the communication steps will be scheduled after those NMSs are merged according to the relationship of contention. Yook and Park [12] presented a relocation algorithm, while their algorithm may lead to high scheduling overheads and degrade the performance of a redistribution algorithm.

3 Preliminaries and Redistribution Communication Models

Data redistribution is a set of routines that transfer all the elements in a set of source processor S to a set of destination processor T. The sizes of the messages are specified by values of user-defined random integer for array mapping from source processor to destination processor. Since node contention considerably influences, a processor can only send messages to other one processor in each communication step. Use the same rule, a processor can only receive messages from other one processor.

To simplify the presentation, notations and terminologies used in this paper are prior defined as follows.

Definition 1：GEN_BLOCK redistribution on one dimension array A[1:N] over P processors. The source processor is denoted as SP_i, the destination processor is denoted as DP_j, where 0 ≦ i, j ≦ P-1.

Definition 2： The time of redistribution separator the time of startup is denoted as t_s, and the time of communication is denoted as t_{comm}.

Definition 3： To satisfy the condition of the minimum steps and the processor sends/receives one message at each steps, some messages can not be scheduled in the same communication step are called conflict tuple [11].

Data redistribution implements have two methods: non-blocking scheduling algorithm and blocking scheduling algorithm. The non-blocking scheduling algorithm is faster than the blocking scheduling algorithm. But need more buffer and be better control synchronization. In this paper, we discuss on blocking scheduling algorithm.

Irregular data redistribution is unlike regular has a cyclic message passing pattern. Every message transmission link is not overlapping. Hence, the total number of message links *N* is $numprocs \le N \le 2 \times numprocs - 1$, where *numprocs* is the number of processors. Figure 1 shows an example of redistributing two GEN_BLOCK distributions on an array *A*[1:101]. The communications between source and destination processor sets are depicted in Figure 2. There are totally fifteen communication messages, m_1, m_2, m_3…, m_{15} among processors involved in the redistribution. In this example, {m_2, m_3, m_4} is a conflict tuple since they have common source processor SP_1; {m_7, m_8, m_9} is also a conflict point because of the common destination processor DP_4. The maximum degree in the example is equal to 3. Figure 3 shows a simple schedule for this example

Source distribution

Source Processor								
SP	SP_0	SP_1	SP_2	SP_3	SP_4	SP_5	SP_6	SP_7
Size	12	20	15	14	11	9	9	11

Destination distribution

Destination Processor								
DP	DP_0	DP_1	DP_2	DP_3	DP_4	DP_5	DP_6	DP_7
Size	17	10	13	6	17	12	11	15

Fig. 1. An example of distributions

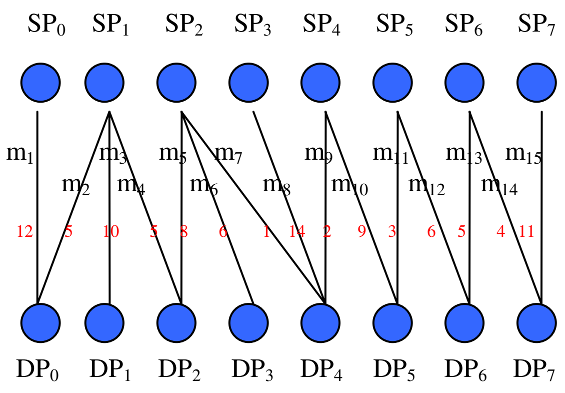

SP₀ SP₁ SP₂ SP₃ SP₄ SP₅ SP₆ SP₇
m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 m13 m14 m15
12 5 10 5 8 6 1 14 2 9 3 6 5 4 11
DP₀ DP₁ DP₂ DP₃ DP₄ DP₅ DP₆ DP₇

Fig. 2. The communications between source and destination processor sets

Schedule Table	
Step 1	m_2 m_5 m_9 m_{12} m_{14}
Step 2	m_1 m_3 m_6 m_8 m_{11} m_{15}
Step 3	m_4 m_7 m_{10} m_{13}

Fig. 3. A simple schedule

3.1 Explicit Conflict Point and Implicit Conflict Point

The total communication time of a message passing operation using two parameters: the startup time t_s and the unit data transmission time t_m. The startup time is once for each communication event and is independent of the message size to be communicated. The data transmission time is relationship of a message size, size(m). The communication time of one communication step is the maximum of the message in this step. The total communication time of all steps is summary of each the communication time of step. The length of these steps determines the data transmission overheads. The minimum step is equal to maximum degree k, when message can not put into any step of minimum step it must relate to the processor has maximum degree transmission links. Figure 4 shows the maximum degree of figure 1. SP_1, SP_2 and DP_4 had maximum degree (K = 3) from messages m_2~m_9. Because of each one processor can only send/receive at most one message to/from other processor in each communication step. First, we concentrate all processors which have maximum degree transmission links messages. For the sake of simplicity, such messages are referred to as "Maximum Degree Message Set" (MDMS) in the paper, as shown in figure 4. If the messages in MDMSs can put into k steps with no conflict occur, other messages of the processors' degree less than maximum degree will be easier to put into the rest of step without increasing the number of steps.

We say a message to be an explicit conflict point if it belongs to two MDMSs. There exists at most one explicit conflict point between two MDMSs. In figure 4, m_7 is a explicit conflict point since it belongs to two MDMSs {m_5, m_6, m_7} and {m_7, m_8, m_9}. On the other hand, if two MDMSs do not contain the same message, but the neighbor MDMSs each has a message been sent by the same processor, or been received by the same processor. We call this kind of message as an implicit conflict point. As shown by figure 5, m_4 and m_5 are contained by the different MDMSs. DP_2 only receives m_4 and m_5 two messages, so it can not form an MDMS. But m_4 and m_5 are also owned by different MDMSs. Therefore, m_4 is an implicit conflict point. Although, m_5 is also covered by two MDMSs, but it is restricted by m_4. Hence m_5 will not cause conflict. Figure 7 depicts all MDMSs for the example shown in Figure 1.

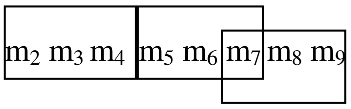


Fig. 4. Maximum Degree Messages Set

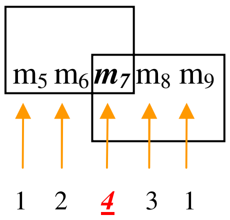


Fig. 5. Example of explicit conflict point

4 Scheduling Algorithm

The main goal of irregular array distribution is to minimize communication step as well as the total message size of steps. We select the smallest conflict points which will really cause conflict to loose the schedule constraint and to minimize the total message size of schedule.

Smallest conflict points algorithm consists of four parts:

(1) Pick out MDMSs from given data redistributed problem.

(2) Find out explicit conflict point and implicit conflict point. And schedule all the conflict point into the same schedule step.

(3) Select messages on MDMSs in non-increasing order of message size. Schedule message into similar message size of that step and keep the relation of each processor send/receive at most one message to/from the processor. Repeat above process until no MDMSs' messages left.

(4) Schedule messages do not belong to MDMSs by non-increasing order of message size. Repeat above process until no messages left.

From Figure 1, we can pick out four MDMSs, $MDMS_1 = \{m_2, m_3, m_4\}$, $MDMS_2 = \{m_4, m_5\}$, $MDMS_3 = \{m_5, m_6, m_7\}$ and $MDMS_4 = \{m_7, m_8, m_9\}$, shown in Figure 8. We schedule m_4 and m_7 into the same step. Then schedule those messages on

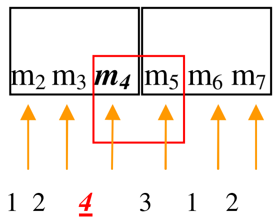


Fig. 6. Example of implicit conflict point

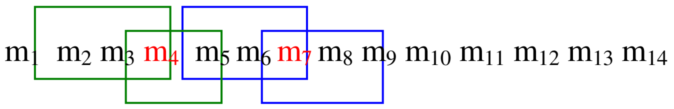


Fig. 7. All MDMSs for the example in Figure 1

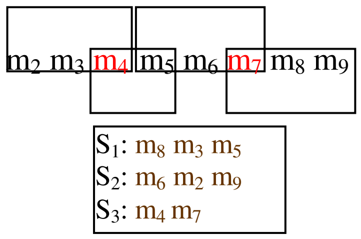


Fig. 8. Results of MDMSs for Figure 1

S_1: m_8 m_3 m_5 m_1 m_{15} m_{10} m_{12}
S_2: m_6 m_2 m_9 m_{13} m_{11}
S_3: m_4 m_7 m_{14}

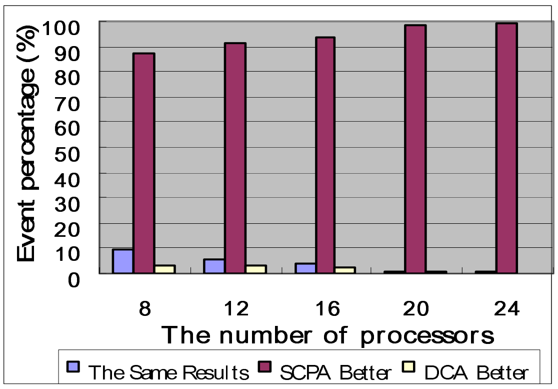
Fig. 9. The schedule obtained form SCPA

MDMSs by non-increasing order of message size as follows: m_8, m_3, m_5, m_6, m_2, m_9. After that, we can schedule the rest messages that are not belong to any MDMSs by non-increasing order of message size as follows: m_1, m_{15}, m_{10}, m_{12}, m_{13}, m_{14}, m_{11}. Figure 9 shows the final schedule obtained form smallest conflict points algorithm.

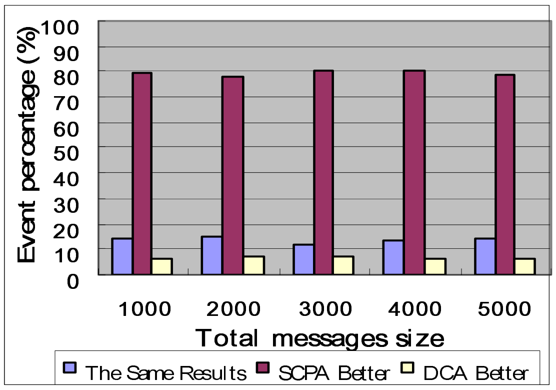
5 Performance Evaluation and Analysis

To evaluate the performance of the proposed methods, we have implemented the SCPA along with the divide-and-conquer algorithm [11]. The performance simula tion is discussed in two classes, even GEN_BLOCK and uneven GEN_BLOCK distributions. In even GEN_BLOCK distribution, each processor owns similar size of data. Contrast to even distribution, few processors might be allocated grand volume of data in uneven distribution. Since array elements could be centralized to some specific processors, it is also possible for those processors to have the maximum degree of communications.

The simulation program generates a set of random integer number as the size of message. To correctly evaluate the performance of these two algorithms, both programs were written in the single program multiple data (SPMD) programming paradigm with MPI code and executed on an SMP/Linux cluster consisted of 24 SMP nodes. In the figures, "SCPA Better" represents the percentage of the number of



(a)



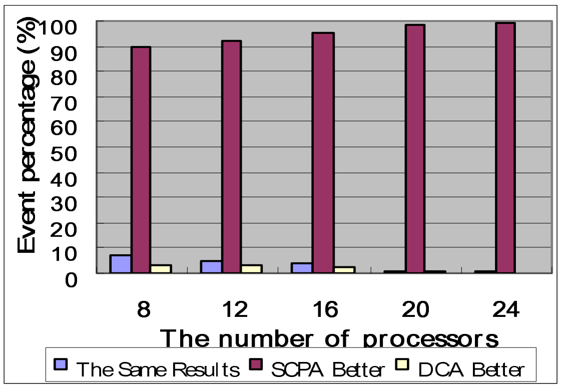
(b)

Fig. 10. The events percentage of computing time is plotted (a) with different number of processors and (b) with different of total messages size in 8 processors, on uneven data set

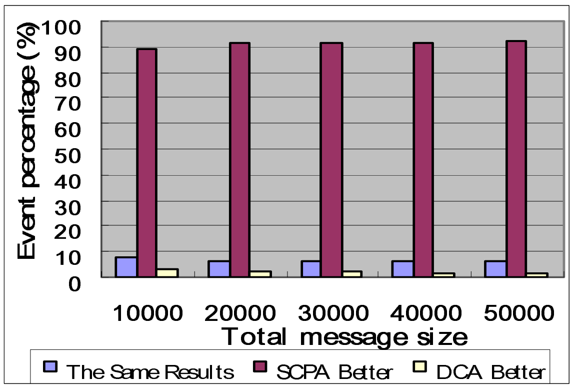
events that the SCPA has lower total steps of messages size than the divide-and-conquer algorithm (DCA), while "DCA Better" gives the reverse situation. In the uneven distribution, the size of message's up-bound is set to (totalsize/numprocs)*1.5 and low-bound is set to (totalsize/numprocs)*0.3, where totalsize is total size of messages and numprocs is the size of processor. In the even distribution, the size of message's up-bound is set to (totalsize/numprocs)*1.3 and low-bound is set to low-bound is (totalsize/numprocs)*0.7. The total messages size is 1M.

Figure 10 shows the simulation results of both the SCPA and the DCA with different number of processors and total message size. We can observe that SCPA has better performance on uneven data redistribution compared with DCA.

Since the data is concentrated in the even case, from figure 11, we can observe that SCPA have the better performance compared with uneven case. Figure 11 also



(a)



(b)

Fig. 11. The events percentage of computing time is plotted (a) with different number of processors and (b) with different of total messages size in 8 processors, on even data set

illustrates that SCPA has at least 85% supreme than DCA in any size of total messages and any number of processors In both even and uneven case, SCPA performs slightly better than DCA.

6 Conclusion

In this paper, we have presented an efficient scheduling algorithm, smallest conflict points algorithm (SCPA), for irregular data distribution. The algorithm can effectively reduce communication time in the process of data redistribution. Smallest-conflict-points algorithm is not only an optimal algorithm in the term of minimal number of steps, but also a near optimal algorithm satisfied the condition of minimal message size of total steps. Effectiveness of the proposed methods not only avoids node contention but also shortens the overall communication length.

For verifying the performance of our proposed algorithm, we have implemented SCPA as well as the divide-and-conquer redistribution algorithm. The experimental results show improvement of communication costs and high practicability on different processor hierarchy. Also, the experimental results indicate that both of them have good performance on GEN_BLOCK redistribution. But also both have advantages and disadvantages. In many situations, SCPA has better than the divide-and-conquer redistribution algorithm.

References

1. Minyi Guo, “Communication Generation for Irregular Codes,” The Journal of Supercomputing, vol. 25, no. 3, pp. 199-214, 2003.
2. Minyi Guo, I. Nakata and Y. Yamashita, “Contention-Free Communication Scheduling for Array Redistribution,” Parallel Computing, vol. 26, no.8, pp. 1325-1343, 2000.
3. Minyi Guo, I. Nakata and Y. Yamashita, “An Efficient Data Distribution Technique for Distributed Memory Parallel Computers,” JSPP'97, pp.189-196, 1997.
4. Minyi Guo, Yi Pan and Zhen Liu, “Symbolic Communication Set Generation for Irregular Parallel Applications,” The Journal of Supercomputing, vol. 25, pp. 199-214, 2003.
5. S. Lee, H. Yook, M. Koo and M. Park, “Processor reordering algorithms toward efficient GEN_BLOCK redistribution,” Proceedings of the ACM symposium on Applied computing, pp. 539-543, 2001.
6. Ching-Hsien Hsu, Kun-Ming Yu, Chi-Hsiu Chen, Chang Wu Yu, and Chiu Kuo Liang, "Optimal Processor Replacement for Efficient Communication of Runtime Data Redistribution," Lecture Notes in Computer Science (ISPA’04), Vol. 3358, pp. 268-273, Dec. 2004.
7. C.-H Hsu, Dong-Lin Yang, Yeh-Ching Chung and Chyi-Ren Dow, “A Generalized Processor Mapping Technique for Array Redistribution,” IEEE Transactions on Parallel and Distributed Systems, vol. 12, vol. 7, pp. 743-757, July 2001.
8. S. Ramaswamy, B. Simons, and P. Banerjee, “Optimization for Efficient Data redistribution on Distributed Memory Multicomputers,” Journal of Parallel and Distributed Computing, vol. 38, pp. 217-228, 1996.

9. Akiyoshi Wakatani and Michael Wolfe, “Optimization of Data redistribution for Distributed Memory Multicomputers,” short communication, Parallel Computing, vol. 21, no. 9, pp. 1485-1490, September 1995.
10. Hui Wang, Minyi Guo and Wenxi Chen, “An Efficient Algorithm for Irregular Redistribution in Parallelizing Compilers,” Proceedings of 2003 International Symposium on Parallel and Distributed Processing with Applications, LNCS 2745, 2003.
11. Hui Wang, Minyi Guo and Daming Wei, "Divide-and-conquer Algorithm for Irregular Redistributions in Parallelizing Compilers”, The Journal of Supercomputing, vol. 29, no. 2, pp. 157-170, 2004.
12. H.-G. Yook and Myung-Soon Park, “Scheduling GEN_BLOCK Array Redistribution,” Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems, November, 1999.

應用網格建立一個高效能演化樹平行建構環境*

游坤明 [1], 徐蓓芳 [1], 賴威廷 [1], 謝一功 [1], 周嘉奕 [1], 林俊淵 [2], 唐傳義 [3]

[1] 中華大學資訊工程學系
[2] 國立清華大學分子與細胞生物研究所
[3] 國立清華大學資訊工程學系

[1] yu@chu.edu.tw, {b9102042, b9004060, b9102004}@cc.chu.edu.tw,
jyzhou@pdlab.csie.chu.edu.tw
[2] cyulin@mx.nthu.edu.tw
[3] cytang@cs.nthu.edu.tw

摘要

以平行處理方式來計算龐大的資料運算是近年來一個非常重要的應用觀念。有許多不同的環境架構伴隨著不同的應用。網格 (Grid) 是一種建立在網際網路上的架構，網格可透過網際網路與其他網格互相分享資源，因此可以視為在使用龐大的且容易增減的資源來運算；與傳統的叢集式系統相比，傳統的叢集式系統 (Cluster) 若要增加運算能力，則必需花費比網格多的費用，因此運算能力有限。在一般所見的網格中，必須要有相同的協定、彼此認同的認證、安全性的考量以及合理的資源存取，才能讓網格在網路上互相溝通。使用網格運算我們所要處理的資料及程式，並且在合理的時間內得到正確的結果。本論文使用平行化演算法並以人類粒腺體為例，在單機、網格與叢集電腦環境中建構演化樹，並比較其效能差異。

關鍵詞：等距演化樹，叢集電腦計算, 網格計算, Globus Toolkit

1. 簡介

生物資訊研究領域中，科學家常常需要從演化樹的結果以了解物種間的親疏關係。從距離矩陣中建造演化樹在生物學和分類法方面是一個重要的議題，因此也產生許多不同的模型及相對應的演算法。而大部份的最佳解問題都已被証明為 NP-hard。

其中在許多不同的模型中有一個重要的模型便是假定演化的速度是一致的 [5, 17]。在這種前提下，利用距離矩陣算出的演化樹將會是一個等距演化樹(ultrametric tree)。

本論文使用一種高效能的平行化分枝界限演算法(branch-and-bound) 建立最小距離演化樹。這個平行演算法是建立在 master-slave centralize 的架構上，並且加入了有效的負載平衡、節點與節點間通訊的策略，以解決最小權值等距演化樹建構的問題，使得時間在可容忍的範圍內完成。

近年來，對於許多以電腦輔助來求解的問題越來越多，且個人電腦的計算能力已無法滿足在合理的時間內得到結果。於是分散式的計算技術便是下一個發展的層次。本論文以人類粒腺體為例建構出演化樹，建構演化樹是一種非常複雜且耗時的計算過程，使用一般的個人電腦，將耗費大量的時間以求得結果，有時還會因資源不足造成等待許久的運作中斷，因此，要在合理的時間內得到滿意的結果，必須具有高效能的電腦，如超級電腦，但在經濟的考量下，我們可使用叢集電腦或網格來達到近似的效能。

叢集電腦可有大小不同規模，此做法的最大優點是「可擴充性」 (scalability) ：只要增加新的個人電腦，就可以提高叢集電腦的效能。在某些情況下資料是分布在不同的地區中需要互相存取，而網格是透過網路連線將好幾個在不同地區的叢集電腦串聯成的，更可以有效的利用這樣的優點來保持最新的訊息，所以在使用資源效率方面更遠勝於叢集電腦 [19]。

在網格上發展的技術為中介軟體，是用來整合網格分散的計算資源，主要角色是擔任機器間協調功能的任務。在網格的使用者和資源提供者之

* This word was supported in part by the NSC of ROC, under grant NSC-93-2213-E-216-037 and NSC-94-2213-E-216-028

間，擔任資源分配的協調工作，幫助使用者找到適合其使用的機器，並完成資料存取的交易 [19]。其中一個重要的組成要素，就是後設資料。

網格的優點之一，是有效率的使用閒置中的電腦，若是再長時間運算比較下，網格可以更有效率的使用資源。使用平行處理的環境，像是叢集計算或網格計算，必須用平行化的演算法以及使用平行化的溝通工具，例如 MPI，以幫助程式在該平臺上順利運作。

目前我們已成功的在網格的環境上執行平行化演算法，並且建構出演化樹，從網格與叢集電腦的實驗數據可看出，網格擁有與叢集電腦相似的效能。在本論文中，比較使用單機、叢集電腦及網格三種環境下的效能，在實驗結果中可顯示出，單機運算能力遠不如叢集電腦及網格；叢集電腦與網格之間的比較，若在相同節點數計算下，兩種環境效能是差不多的。

2. 背景

2.1 等距演化樹

在建立演化樹上有許多模型，其中一種為等距演化樹。等距演化樹為假設各物種的演化速率一致 [5, 13]，而等距演化樹的特性為有共同的父節點，物種存在葉節點而且在邊上有權重值的一個二元樹，在每個內節點的子樹中有同樣的路徑長到每一個葉節點上 [4]。對於一個 n * n 的距離矩陣 M 來說，定義最小的等距演化樹指的是兩兩葉節點的邊上權重總合為最小的。因為等距演化樹可以很容易的轉換為二元樹且不需要改變葉節點的距離 [13]，所以，等距演化樹是一個非常適合給電腦計算的模型。

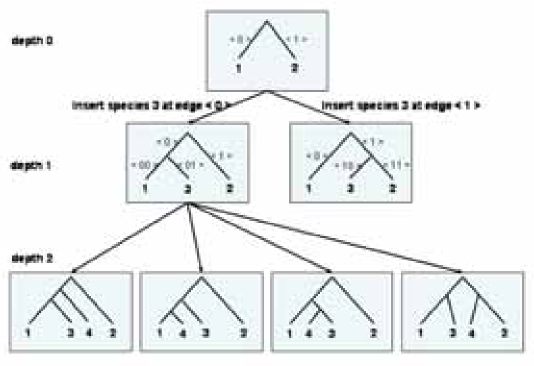


圖1. 建立分支界限樹 (BBT) [3]

如圖 1，我們可知，等距演化樹的數目 A(n)，隨著 n 的增加，演化樹的數量也快速的增加。有一些有關等距演化樹的研究先前已被提出 [6, 7, 15]。由於這些問題往往是不易解的，所以這些研究大都是基於 heuristic 演算法。舉例來說，像 UPGMA(Unweighted Pair Group Method with Arithmetic mean) [17]就是一個很常被用來建立等距演化樹的演算法。

在本論文中，我們使用 Exact Algorithms for Constructing Minimum branch-and-bound’s from Distance Matrices [4]的演算法為基礎，並將之平行化。在上述方法中，使用分支界限法的策略作為找尋最小距離演化樹的方法。為了求得最小距離演化樹我們會將所有可能的樹型都找出並一一求值，但隨著物種數的增加，等距演化樹 A(n)的增加是非常快的，例如：$A(20) > 10^{21}$ ，$A(25) > 10^{29}$ ，$A(30) > 10^{37}$ ，於是上述方法中使用了分支界限法的策略來避免完全的搜尋。在本論文中，使用有效率平行化的分支界限演算法建立最小距離演化樹，在我們提出的方法中，是一個主從且集中式的平行化架構，並在此架構中加上了 loading-balancing, bounded 和 communication strategies 等機制，以增加程式的效率。

2.2 叢集計算

叢集計算(cluster computing)在隨著目前的科技下，處理器和周邊設備的普及，我們可以用低成本連接出高效能的叢集計算機。叢集計算機是以高速網路連接個人電腦或工作站而成的，可提供高效能的計算能力而且降低原來達到此效能的成本。在運作上，既然是由許多台電腦連接的，所以普通的應用程式也無法在上面發揮作用，必須設計適合在平行及分散式環境中的演算法，而且同時配合像是 MPI 這種專門用來做平行溝通的軟體，來設計應用程式。

現今在電腦和網路普及下，幾乎是可以看成所有電腦都與網際網路相連，如果把叢集電腦更廣義的角度來看，每台電腦就好像被網際網路連接的大型區網，全球就是一個大型的叢集電腦，但是事實並非如此，因為無法做到資源互相分享、計算互相分擔，所以為了達到更廣義的資源活化運算，於是網格計算的理念被提出。

2.3 網格計算

網格計算(Grid Computing)可讓分散於各地的虛擬組織，協調彼此的資源分享，同時滿足大量運算的需求。而集合分散的運算資源之外，網格計算能夠經由網路管理組織內任何一個可使用的運算資源，進而降低伺服器的閒置時間。

網格計算可以解決在同一時間內使用網路上很多資源去解決一個問題或者當一個問題需要大量處理器計算或是需要存取大量分佈不同地方的資料。耳熟能詳的例子像是 SETI (Search For Extraterrestrial Intelligence)@home 它讓上千人的電腦在閒置時的處理器中去幫助計算資料。而且這些電腦都是獨立性工作，指的是說無論有些工作需

花較長的時間，或者沒有回傳資料，都沒有關係，因為有此狀況時，它會在暫停一段時間後，自動把工作分派給其他電腦做處理。

2.4 Globus Toolkit

Globus [8, 14, 20]對訊息安全、資源管理、訊息服務、數據蒐集管理以及應用開發環境等網格關鍵理論和技術進行廣泛的研究，並且開發出可以在多種平台上執行的 GlobusToolkit，用來幫助規劃和建造大型網格試驗和應用平台，開發大型網格系統可以執行的應用程式。Globus Toolkit 同時提供了好幾種語言模式給程式設計師選擇，就類似像物件導向的方式。程式開發者更可以由 Globus Toolkit 中所提供的服務任意選取最符合需求的工具去與現存的軟體作整合。例如: GRAM 提供資源管理的協定、MDS 提供資訊服務的協定、GridFTP 提供了資料傳輸的協定…等，這些全部都有使用 GSI 安全協定在他們的連接層 [20]。

表 1. GlobusToolkit 所提供的服務

Service	Name	功能
Resource managrment	GRAM	資源分配與工作管理
Communication	Nexus	單一或多重溝通服務
Security	GSI	認證與聯繫上的安全服務
Information	MDS	分散式存取和狀態的資訊
Health and status	HBM	監測系統零件健康狀況
Remote data access	GASS	遠程存取資料經由連續及平行的連繫裝置
Executable management	GEM	結構、讀取技術與狀態執行管理
Information	GRIS	查詢計算資源現有的設定、能力及狀態
GridFTP	GridFTP	提供高效能、安全，以及健全的資料傳輸機制

2.5 MPICH-G2

MPI 是訊息傳送介面(Message Passing Interface)用來撰寫 message-passing programs 和可以廣泛的使用於平行運算的一種基礎 API。在網格應用程式上 message-passing 的優點是它提供比通訊協定 TCP/IP sockets 更高層的介面，讓我們可以直接使用通訊結果而不必知道中間是如何溝通。Globus 服務已被用來發展成 Grid-enable MPI 以 MPICH library 為基礎，Nexus 為通訊基礎，GRAM 服務為資源分配和 GSI 來做安全認證。

MPICH-G2 是 Grid-enable 以 MPI v.1.1 為基礎在網格上的實作。它使用了 Globus Toolkik(像是資源分配、安全性)的服務。MPICH-G2 准許以連接不同平台的機器來執行 MPI 的程式。MPICH-G2 會自動作資料轉換當在兩個不同平台時的傳輸和自動的選擇 TCP 以提供多重協定通訊的訊息給網路上機器及傳出有 MPI 提供的訊息給區域內機器。

2.6 UniGrid

網格計算的目的是用來整合大型網路環境下的各種資源。UniGrid 是連結國內七所大學及國家高速網路中心之電腦網格系統，建置一「國家計算網格實驗平台」，以協助推廣網格計算的觀念到各產學領域。UniGrid 將著重在使用網格計算領域最常用之程式集及工具套件 Globus。並且有提供隨時每個節點的 CPU、RAM 狀況監看。

Globus[8]提供了網格中使用的協定，可以讓使用者充分利用分散於各處的資源中建出網格計算的架構。

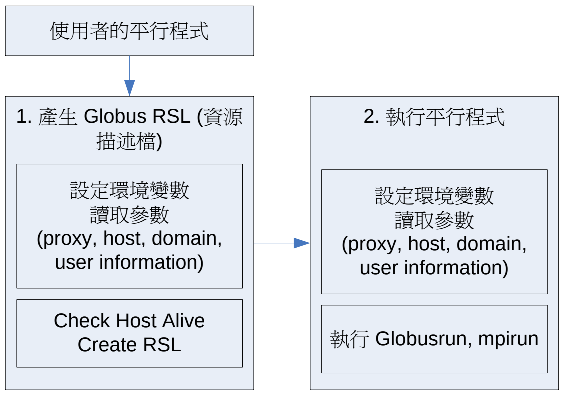


圖 2. UniGrid 上的程式流程

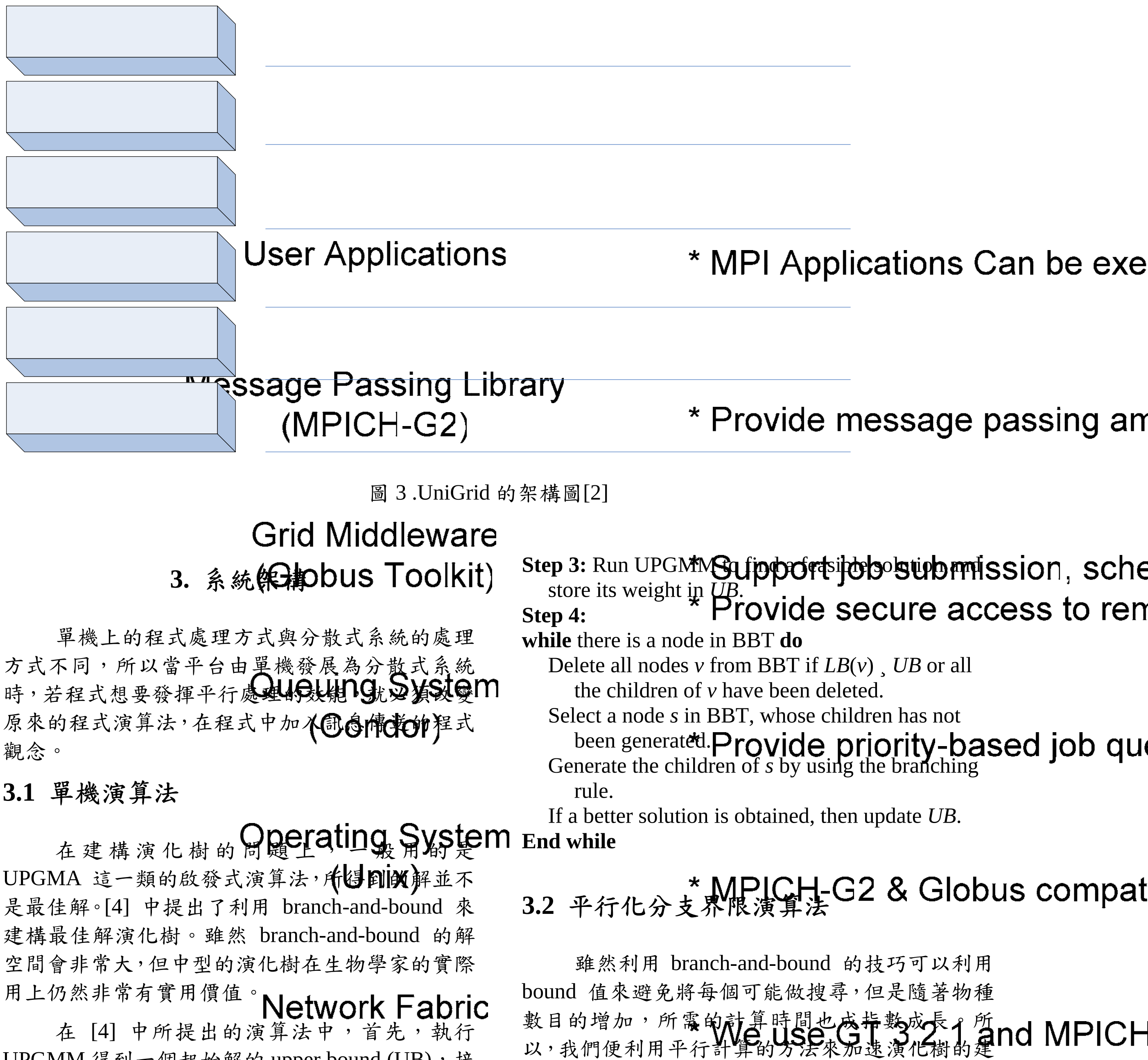

User Applications
* MPI Applications Can be exe
Message Passing Library (MPICH-G2)
* Provide message passing am
Grid Middleware (Globus Toolkit)
* Support job submission, sche
* Provide secure access to rem
Queuing System (Condor)
* Provide priority-based job qu
Operating System (Unix)
* MPICH-G2 & Globus compat
Network Fabric
* We use GT 3.2.1 and MPICH

圖 3 .UniGrid 的架構圖[2]

3. 系統架構

單機上的程式處理方式與分散式系統的處理方式不同，所以當平台由單機發展為分散式系統時，若程式想要發揮平行處理的效能，就必須改變原來的程式演算法，在程式中加入訊息傳遞的程式觀念。

3.1 單機演算法

在建構演化樹的問題上，一般用的是 UPGMA 這一類的啟發式演算法，所得到的解並不是最佳解。[4] 中提出了利用 branch-and-bound 來建構最佳解演化樹。雖然 branch-and-bound 的解空間會非常大，但中型的演化樹在生物學家的實際用上仍然非常有實用價值。

在 [4] 中所提出的演算法中，首先，執行 UPGMM 得到一個起始解的 upper bound (UB)，接著開始建立 branch-and-bound tree (BBT) 如果建立時 lower bound (LB) 大於目前的 UB 時就刪除此節點，選擇下一個位置繼續建立，當計算到 UB 比目前的 UB 低時就更新。直到所有物種都建立完畢，最後，權值最小的樹即是我們所要求的解。其演算法如下：

```
Algorithm BBU
Input: An n £n distance matrix M.
Output: The minimum ultrametric tree for M.

Step 1: Relabel the species such that (1, 2.…. n) is a
    maxmin permutation.
Step 2: Create the root v of the BBT such that v
    represents the only topology with leaves 1 and 2.
Step 3: Run UPGMM to find a feasible solution and
    store its weight in UB.
Step 4:
while there is a node in BBT do
    Delete all nodes v from BBT if LB(v) ¸ UB or all
        the children of v have been deleted.
    Select a node s in BBT, whose children has not
        been generated.
    Generate the children of s by using the branching
        rule.
    If a better solution is obtained, then update UB.
End while
```

3.2 平行化分支界限演算法

雖然利用 branch-and-bound 的技巧可以利用 bound 值來避免將每個可能做搜尋，但是隨著物種數目的增加，所需的計算時間也成指數成長。所以，我們便利用平行計算的方法來加速演化樹的建構。

考慮在平行計算的環境上的特性，所以針對資料結構和演算法做了些改變和增加。在資料結構上，為了減少節點與節點之間的溝通，因此所定義的資料結構包含了每個內結點的左子節點、右子節點、父節點，與子結點的路徑。在演算法上，為了更能發揮平行處理的環境，必須讓每個節點的計算量平衡，故必須加上如 Global Pools、Local Pools、等機制讓節點與節點間可以達到動態的負載平衡，並且我們為了減低不必要的計算，在算出一個比原來標準用的上限還低時，就會一直把資訊傳給全部的節點以達到提升計算效率。而平行化架構採用的是主從式架構，起始化時分配的資料與計算過程中所需動態分配的資料都是由 master 來做分配。

在負載平衡的問題上，一般來說可以分為靜態與動態的負載平衡 [3]。靜態的負載平衡指的是在資料的分配只在程式一開始的期間做分配，而程式執行期間不做任何的資料牽移；相對的動態負載平衡指的是會依需求而在節點間搬動及牽移資料。動態負載平衡可以分為集中式的 (centralized) 與分散式的 (decentralized) [3]。集中式的負載平衡是由一台管理主機 (管理節點) 來做調控，每個節點藉由把資料送至管理節點後再由管理節點來決定資料要如何分配。相對的分散式負載平衡則是由節點彼此間互相溝通後再彼此間一同決定的機制。一般來說，集中式的架構能夠有更好的負載平衡，因為管理節點可以知道所有節點的狀態並決定一個更好的分配，但在一個大型的平行系統下，集中式的負載平衡會因為管理節點的瓶頸而效能不佳。

```
Input: A n * n distance matrix M
Output: The minimum ultrametric trees

Step 1: Master computing node re-label the species such that feasible maxmin permutation.
Step 2: Master computing node creates the root of the BBT.
Step 3: Master computing node run UPGMA and using the result as the initial UB (upper bound).
Step 4: Master computing node branches the BBT until the branched BBT reach 2 times of total nodes in the computing environment.
Step 5: Master computing node broadcasts the global UB and send the sorted matrix the nodes cyclically.

Step 6:
while number of UTs in LP (Local Pools) > 0 or number of UTs in GP (Global Pools) > 0 do
    if number of UTs in LP = 0 then
        if number of UTs in GP <> 0 then
            receive UTs from GP
        end if
    end if
    v = get the tree for branch using DFS
    if LowerBound(v) > UB then
        continue
    end if
    insert next species to v and branch it
    if v branched completed then
        if Cost(v) < UB then
            update the GUB (Global Upper Bound) to every nodes
            add the v to results set
        end if
    end if
    if number of UTs in GP = 0 then
        send the last UT in sorted LP to GP
    end if
end while

Step 7: Gather all solutions from each node and output it.
```

4. 實驗結果

4.1 實驗環境及結果

在實驗的環境中，我們使用了單機、以及叢集電腦與網格的系統。單機及叢集電腦的系統如表 2。網格實驗環境使用的是 UniGrid 系統。

在實驗數據中，我們挑選人類粒腺體做為實驗數據，並以物種數目 12、14、16、18、20、22 一一執行，每一物種數目有 10 組測試資料。我們從 10 組資料中分別取中位數、平均數、最差情況來做實驗結果比較，以期消除資料相依所產生執行時間的差異。

表 2. 實驗環境

單機	
處理器數目	1
環境	中華大學平行分散實驗室
硬體設備	AMD 2000+、2GB DDR RAM
叢集電腦	
處理器數目	16
環境	中華大學平行分散實驗室
硬體設備	AMD 2000+、1GB DDR RAM
網格	
處理器數目	12
環境	國家網格計算實驗平台
硬體設備	AMD 1.3G、2GB DDR RAM
處理器數目	4
環境	東海大學高效能計算實驗室
硬體設備	AMD MP 2000+ ‘2、512MB DDR RAM’2

如表 3 及圖 4 分別為執行時間中位數的結果，我們可以發現，當物種數目增加時，計算時間也相對增加，而在圖中也可以了解，不論是叢集電腦或者是網格系統，都能夠有效的降低執行時間。

表 3. 中位數時間比較表

物種數目	單機	叢集電腦	網格
12	0.113313	0.146947	0.130881
14	2.936615	0.889956	1.180245
16	36.1053	29.2515	11.6873
18	1003.268	324.5231	332.807
20	138.684	157.6269	99.2526
22	9873.82	5911.42	2625.637

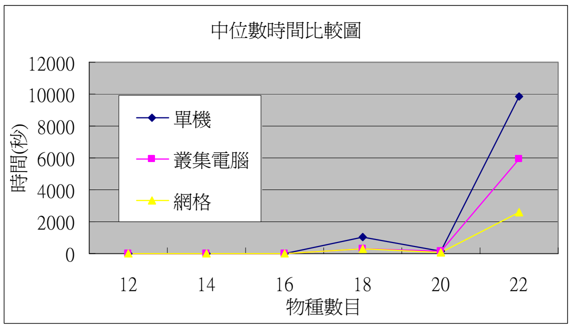


圖 4. 中位數時間比較圖

表 4 以及圖 5 為平均計算時間；表 5 以及圖 6 為最差計時間，兩個相比較，我們可以了解、平均計算時間可能被最差計算時間所影響，因為建構演化樹的問題有資料相依的情形，所以我們會選擇中位數計算時間做為我們主要的比較依據。而從圖中也可以觀察到，計算時間隨著數種數目的成長有相當快速的增加。

表 4. 平均數時間比較表

物種數目	單機	叢集電腦	網格
12	0.344878	0.166051	0.236581
14	41.99103	7.537349	7.390265
16	390.8962	207.8525	58.34718
18	2598.467	983.583	1031.805
20	1114.028	4705.797	249.0695
22	9873.82	5911.42	2625.637

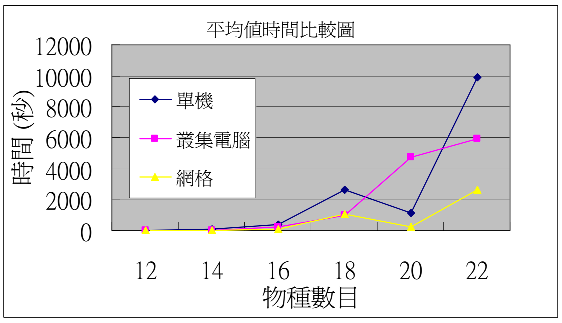


圖 5. 平均數時間比較圖

表 5. 最差狀況時間比較表

物種數目	單機	叢集電腦	網格
12	0.785476	0.395494	0.927229
14	303.738	40.4603	35.1285
16	1387.6	911.781	203.611
18	9339.67	4327.96	4606.76
20	5009.17	25064.1	1028.86
22	9873.82	5911.42	5068.7

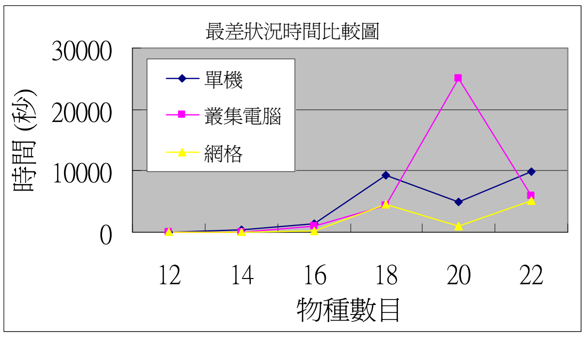


圖 6. 最差狀況時間比較圖

4.2 結果討論

我們從數據中取出中位數、平均數、最差情況來做比較。由圖表明顯看出隨著物種數目增加、計算相同物種時，單機效能最差，叢集電腦次之，網格效能最佳。正常情況下，叢集電腦的效能應比網格好，因為使用內部溝通為高速網路的叢集電腦，其效能遠高於使用網際網路溝通的網格。但是實驗結果與理論不符，這是因為實驗中所使用的叢集電腦設備較網格所使用的電腦設備差。

最初使用單機運算樣本以繪出演化樹，雖然成功建出演化樹，但是花費時間非常驚人，且運算樣本的大小有限，因此進度緩慢、效率不佳，之後採用叢集電腦。叢集電腦環境為 16 顆處理器，因此效率提高許多。但因為考慮到經濟成本以及為了應付更巨大的計算，我們考慮了更有效率的平行處理環境:網格。

所以我們開始把平行化建立演化樹的程式以網格平台來做實驗。我們以 10 組物種數 20 的資料去實驗，實驗結果見表 6 和圖 7。實驗結果發現網格計算效能，如果在相同的節點數目，計算效能似乎較叢集電腦差了一點，但是如果網格使用 24 節點，則效能遠超過叢集電腦 16 節點。

而現今已有許多網格平台的建立，像是 UniGrid 就是聯合國內七所大學以及國家高速網路中心的叢集實驗室所成的網格實驗平台，我們可以使用更多的資源去執行程式，並且透過網格運算的技術，他會到網路中尋找閒置的電腦，並將工作依據適當的比例分配，送到這些電腦上執行，然後將結果送回，這樣做可以更有效率。

而且我們考慮了未來萬一資料是放置在世界各處或者是隨時都會更動的，那叢集電腦就顯得不適合，且叢集電腦資源有限，如果遇到一個龐大的問題也可能需要計算很久的時間，所以即使資料分布在世界各地也可以輕鬆的應付並保持資料的最新狀況。

表 6. 叢集電腦與網格使用不同節點數比較

	叢集電腦 16 節點	網格 16 節點	網格 24 節點
1	1629	1652.96	885.517
2	10273	10691.6	6838.49
3	750	764.104	750.752
4	561	566.25	458.426
5	249	258.197	256.616
6	199	199.96	148.454
7	54	57.693	83.55
8	126	128.596	110.922

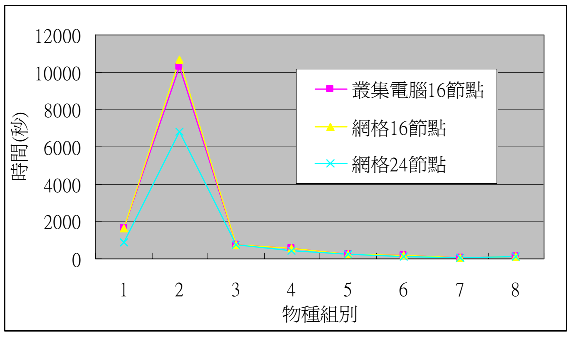


圖 7. 叢集電腦與網格使用不同節點數比較圖

5. 結論

實驗中測試的網格所使用的處理器 16 顆，與叢集電腦相比，數據中發現，網格與叢集電腦使用處理器個數相同，網格並無任何優勢，網格效能較叢集電腦差，因為叢集電腦內部溝通速度遠大於網格間所使用的網際網路溝通。未來我們可以考慮建立更有效率的網格溝通機制，相信可以大幅改善網格的效能。

我們的目標將是不只侷限於計算人類粒腺體的資料，推廣至以網格來運算蛋白質的樣本，甚至其他的資料在正規化之後皆可用網格來運算以得到結果。

目前是用網格來運算人類粒腺體的樣本，雖然花費的時間非常的多，因為剛開始時需要找出在網格上的最佳效率，但是，未來中，我們將更有效率的平行演算法及網格 API，目標在使用方面，只要將要處理的資料整理成我們目前資料輸入的形式，就可以得到我們要求的數據，所以，未來可能運用此種方法來運算類似的龐大資料，像蛋白質等等的樣本，應該跟目前的運作方式相同，在網格上運算即可得到結果。

參考文獻 ：

[1] 全球網格(World Wide Grid)發展趨勢

[2] 格網計算平台架設實例簡介格網計算平架設實例簡介 Introduction to Constructing Introduction to Constructing Computational Computational Grid Grid, 王順泰

[3] Barry Wilkinson, Michael Allen, “Parallel Programming”, *P.H.*

[4] B.Y. Wu, K.M. Chao, C.Y. Tang, “Approximation and Exact Algorithms for Constructing Minimum Ultrametric Tree from Distance Matrices,” Journal of Combinatorial Optimization 3, pp. 199-211

[5] D. Gusfield, “Algorithms on Strings, Trees, and Sequences, computer science and computational biology,” Cambridge University Press, 1997

[6] E. Dahlhaus, “Fast parallel recognition of ultrametrics and tree metrics,” *SIAM Journal on Discrete Mathematics*, 6(4):523-532, 1993

[7] H.J. Bandelt, “Recognition of tree metrics,” *SiAM Journal on Discrete Mathematics.*, 3(1):1-6, 1990

[8] The Globus Project: A Status Report. I. Foster, C. Kesselman. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18, 1998.

[9] The Anatomy of the Grid: Enabling Scalable Virtual Organizations. I. Foster, C. Kesselman, S. Tuecke. *International J. Supercomputer Applications*, 15(3), 2001.

[10] The Nexus Approach to Integrating Multithreading and Communication. I. Foster, C. Kesselman, S. Tuecke, J. *Journal of Parallel and Distributed Computing*, 37:70--82, 1996.

[11] A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. I. Foster, N. Karonis. *Proc. 1998 SC Conference*, November, 1998.

[12] A Secure Communications Infrastructure for High-Performance Distributed Computing. I. Foster, N. Karonis, C. Kesselman, G. Koenig, S. Tuecke. *6th IEEE Symp. on High-Performance Distributed Computing*, pp. 125-136, 1997.

[13] M.D. Hendy and D. Penny, “Branch-and-bound algorithms to determine minimal evolutionary trees,” Mathematical Biosciences, 59:277-290, 1982.

[14] M. Frach, S. Kannan, and T. Warnow, “A robust model for finding optimal evolutionary trees,” *Algorithmica*, 13:155-179, 1995.

[15] M. Krivanek, “The complexity of ultrametric partitions on graphs,” *Information Processing Letter*, 27(5):265-270, 1988.

[16] Chuan Yi Tang, Solomon K.C. Wu, "Chee Kane Chang, “A scalable Fully Distributed Parallel Branch & Bound Algorithm on PVM cluster”

[17] W.H. Li and D. Graur, “Foundomentals of Molecular Evolution,” *Sinauer Associates*, 1991.

[18] Yuji Shinano, “Kenichi Harada and Ryuichi Hirabayashi,” *Control Schemes in a Generalized Utility for Parallel*

Branch-and-Bound Algorithms, Parallel Processing Symposium, 1997. Proceedings., 11th International , 1-5 Apr 1997, Page(s): 621 -627

[19] GridCafé (http://www2.twgrid.org/gridcafe)

[20] The Globus Project (http://www.globus.org/)

附錄二：出席國際學術會議心得報告

第二屆平行分散處理與應用國際研討會

Second International Symposium on Parallel and Distributed Processing and Applications (ISPA'2004)

一、前言

第二屆平行分散處理與應用國際研討會(ISPA 2004)於西元 2004 年十二月十二日至十二月十五日在香港之香港科技大學舉行。本次會議共錄取 106 篇平行分散研究領域之優秀論文，分為二十七個議題進行討論，分別為” Parallel Algorithms and Systems I、II” 、” Data Mining and Management” 、” Distributed Algorithms and Systems”、” Fault Tolerant Protocols and Systems 、” Sensor Networks and Protocols”、”Cluster Systems and Applications”、” Grid Applications and Systems”、”Peer-to-Peer and Ad-Hoc Networking”、” Data Replication and Caching ”、” Software Engineering and Testing”、” Grid Protocols”、” Context-aware and Mobile Computing”、” Grid Scheduling and Algorithms I、II”、” Cluster Resource Scheduling and Algorithms”、” Distributed Routing and Switching Protocols I、II”、” High Performance Processing and Applications”、” Security I、II” 、” Artificial Intelligence Systems and Applications”、” Networking and Protocols I、II”、” Hardware Architectures and Implementations”、” High Performance Computing and Architecture”以及” Distributed Processing and Architecture ”。

平行分散處理與應用國際研討會是國際平行計算界領域的學者、專家相互交流研究成果和資訊技術開發經驗的年會，在世界各地輪流舉辦。第一屆會議於 2003 年在日本愛知大學(The University of Aizu)舉行；今年是第二屆，雖然僅舉辦二年，但是卻已成為分散平行計算研究界領域的重要會議，而且研討會論文已被 LNCS 收錄發行(SCI Extend) ，所以 ISPA 已是目前在平行暨分散計算研究領域中相當具有代表性之會議。

二、參加會議經過

會議的開幕典禮由主辦單位與會議的委員會主席簡單的致歡迎詞後，隨即展開，本屆會議分別於三天的會議議程中安排了三個場次之平行計算最新趨勢之專題報告: (1). Present and Future Supercomputer Architectures”(2). "Challenges in P2P Computing" (3). "Multihop Wireless Ad Hoc Networking: Current Challenges and Future Opportunities"，分別由 Prof. Jack Dongarra、Prof. Lionel Ni 以及 Prof. David B. Johnson 作精彩的專題報告，正式之論文報告分別於三天三個場次的專題報告後進行，本人之論文『Optimal Processor Mapping Scheme for Efficient Communication of Data Realignment』被安排在第一天的”Parallel Algorithms and Systems II”(Session 3A)之場次發表，本人並且擔任此場次之會議主席。

三、與會心得

此次會議有來自世界各地的學者發表了相當多優秀的平行計算各領域的論文，國內大約有六位學者參加此次會議並同時發表論文，由會議的進行過程中可以看出主辦單位對會議的流程安排相當用心，不過在專題報告的時間控制上不良，超出預定之時程太多，這些都是整個會議美中不足的地方。

四、攜回資料

1. 大會議程
2. Second International Symposium on Parallel and Distributed Processing and Applications 研討會論文集