行政院國家科學委員會專題研究計畫 成果報告

設計與實作一個最佳演化樹之分散式建構環境(I)
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 94-2213-E-216-028-
執 行 期 間 ：94 年 08 月 01 日至 95 年 07 月 31 日
執 行 單 位 ：中華大學資訊管理學系

計 畫 主 持 人：游坤明
共 同 主 持 人：唐傳義
計畫參與人員：碩士班研究生-兼任助理：周嘉奕、蔡宜霖、黃立明

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 95 年 12 月 22 日

行政院國家科學委員會專題研究計畫　成果報告

設計與實作一個最佳演化樹之分散式建構環境(I)

計畫類別：個別型計畫
計畫編號：NSC94-2213-E-216-028-
執行期間：94 年 08 月 01 日至 95 年 07 月 31 日
執行單位：中華大學資訊管理學系

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、黃&#63991；明、蔡宜霖

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 95 年 10 月 16 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

設計與實作一個最佳演化樹之分散式建構環境

計畫類別：■ 個別型計畫　　□ 整合型計畫

計畫編號： NSC 94-2213-E-216 -028

執行期間：　94年8月1日至95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、黃立明、蔡宜霖

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　95　年　8　月　30　日

設計與實作一個最佳演化樹之分散式建構環境

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最佳等距演化樹的平行建構系統。在此計畫執行中，我們將有效利用 Compact Set 來做為物種資料的分群，藉以減少物種個數以縮短建構演化樹的時間，以利建構大型演化樹，並且考慮以 2nd best methodology、3-3 關係、4 點關係等技巧來探討快速建構最佳演化樹的平行化策略與方法。最後我們將會把計畫的執行成果整合成 Web 介面之執行環境，提供給從事此研究領域的專家學者使用以及提供給生物學家選擇較適合的演化樹作實際的應用（UTCE）。計畫進行中所得到的研究成果已整理成二篇論文在國際研討會中發表，以及二篇國內研討會論文，同時亦有一篇論文被收錄於 Lecture Notes in Computer Science(SCI), Springer-Verlag 系列專書中，除此之外，我們亦將所建構完成之網站（UTCE: Ultrametric Tree Construction and Evaluation platform）所提供的各項演化樹的建構相關功能整理成論文參加 The Evolutionary Genomics and Bioinformatics Symposium and Workshop (EGBS 2006)之壁報比賽並獲得 Excellent Poster Prize。
關鍵詞：等距演化樹、分支與界定、演化樹評估方法、叢集電腦、網格計算

Abstract:
The construction of evolutionary tree is an important problem in biology and taxonomy. The purposes for studying phylogenetics include (1) reconstructing the correct genealogical ties between species and (2) estimating the time when a divergence occurs between species from a common ancestor. Usually, these can be done by constructing evolutionary trees to obtain plenty of information. Often, we assume evolutionary tree is an ultrametric tree whose leaves are the present species and whose interior nodes represent the ancestors of the species. By the definition of an ultrametric tree, the distances from root to all the leaves are the same, and it means that the present species have the same progresses in evolution so far. In the project, we developed an effective parallel algorithm to construct an optimal ultrametric tree from a given distance vector by using Branch-and-Bound technique based on clustering technique from Compact Set. Also, we have studied some methods to speedup the ultrametric tree's construction, for example, 2nd best methodology, 3-3 relationship and 4 points relationship. In addition, we implemented an efficient ultrametric tree's construction algorithms for Cluster computing environment. Finally, we integrated the related results to provide an efficient as well as user friendly Web-base ultrametric tree construction environment（UTCE: Ultrametric Tree Construction and Evaluation platform）.
Keyword : Ultrametric Tree、Branch and Bound、Revolutionary Tree Evaluation、PC Cluster、Grid Computing

一、前言

由於分子生物學的迅速發展，DNA 已經能從各個物種取出來。物種之間的比較，也進入定量的計算。而當生物學家得到 DNA 序列後，如何計算序列之間的親疏程度是一個相當重要的議題。在計算序列之間的親疏程度上，我們通常先求得一個距離矩陣 (distance matrix) 代表兩兩序列之間的距離，爾後再用此 matrix 來建構演化樹 (evolutionary tree)。為了計算序列之間的親疏程度，生物學家希望將序列並排，相同的部份儘量對齊。這個問題稱 (多重序列排比) multiple sequence alignment，在得到 distance matrix 後，有許多不同的數學模型可以來建立演化樹，大多數的模型也都是難題，而這裡我們所指的難題、指的是問題難度在 NP-hard 以上。

目前在建立演化樹上有許多不同的方法，這些方法中、依據不同的假設、包含了許多的 heuristic algorithm；而根據不同的數學模型，有許多不同的 optimum algorithm。這些選擇、對生物學家而言沒有絕對的好壞、也沒有絕對的意義，生物學家往往在許多的測試、嘗試以後得到他們想要的答案。所以，當我們建構完演化樹後、我們也試著評估樹的好壞。

在這些建立演化樹的方法中，我們可以從距離短陣建立，也可以從 DNA 序列來建立，在建立一個最佳解的演化樹的問題上，大部份都已被證明為 NP-Complete 問題。在此計劃中我們主要選擇是從距離矩陣來建立演化樹；用距離矩陣來建構演化樹的 heuristic algorithm 常用有下列數種方法 (1) Unweighted Pair-Group Method with Arithmetic Mean (UPGMA), (2) Transformed Distance Method, (3) Neighbors-Relation Method, (4) Neighbor-Join Method，而我們想要求的是最佳解的樹，所以我們選擇了方法是 Minimum Size Ultrametric Tree。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。

目前來建構演化樹所使用的方法大部份為 “分枝與界限” (branch-and-bound) 的方式求最佳解。當處理的資料量不大時，單一處理器尚能負擔其計算量，但當資料量增多時，單一處理器便會出現記憶體不足或者無法在合理的時間內求出答案。所以在目前的技術下，如果要在合理的時間內得到答案的話，必需考慮使用多處理器或平行系統來處理所需的資料。前人曾提出了一個以分枝與界限的方法來建構 MUT (Minimum Size Ultrametric Tree)，在些方法中，我們可以有效的在單一機器上以分枝與界限來建構 MUT，在我們之前申請通過的二年計劃中，在此議題的研究上，我們有下述的心得及成果，在研究中，我們依據 [24] 作為發展的根據，依此為依據，發展了一個有效率的平行化演算法，在我們的平行化系統中，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 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 bound 掉更多的候選樹。基於這個理由，在叢集系統的解集合會少於單一處理器的系統，所以我們提出的平行化分枝與限界演算法在 speedup 上可能會達到 super-linear 的速度。我們使用了 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。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 指派。在我們的研究成果中可以知道，在單一處理器時，23 個物種數已經是在我們能接受的時間內能得到最佳解的物種數目了，如果要計算更多的物種

的話，勢必用平行化等方法讓在合理的時間內 (一般來說，我們期望在 1 天內能夠得到我們想要的演化樹)得到更多物種的最佳解，我們可以知道在我們提出的平行演算法中，可以在合理的時間內將物種數目推至 48 個物種，如果物種數目再增加時，時間會成指數成長，所以在研究過程中讓我們了解到，如何減少物種的數量 (如使用分群演算法對物種分群、以減少候選樹的數量) 或者如何例用其他限制式 (如 2nd Best Methodology、3-3 Relationship、4 Points Relationship) 加速 bound 的速度或減少候選樹的產生，將是我們完成 136 物種最佳化求解的方向。而相同的距離矩陣可能會產生許多不同的演化樹，於是如何用適當的數學模式或者適當的依據來對演化樹做評估將是我們研究的重點，並且也是讓生物評估的一個重要方向。在最後我們希望能提供一個人性化的使用者介面以及利用格網強大計算能力的特性加速我們演化樹的建構。

二、研究目的

由於生物學的迅速發展，許多的 DNA 序列已從各個生物中萃取出來，生物學家在得到許多 DNA 序列後，如何計算序列之間的親疏關係是一個相當重要的議題，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演化樹的平行系統，在此計畫中，我們希望能夠建立出更多物種最佳解的結果樹，但是在分枝與界限的過程中，當物種數目增多時，其候選樹的數目成長是非常快速的，在先前的研究中，我們在合理時間內得到最佳解的演化樹的物種數目依然有限 (在合理時間內，平行化程式能求得的最佳解演化樹在 48 個物種左右)，所以如何減少物種的數目或者適當的分群是解決本計劃一個主要的方向，在本計劃中，我們提出了許多方法如：緊湊集合 (Compact Set)、2nd Best Methodology、3-3 Relationship、以及 4 Points Relationship，進行有效縮短最佳解的演化樹的探討。並且我們希望能將所有方法整合在一個 Web-base 的使用者介面中，讓使用者可以自行選擇參數、設定運算模式，時間限制以及選擇所欲的方法來評估樹的好壞並可以在 Web 介面中直接顯示演化樹的樹型。

三、文獻探討

在建構最佳演化樹的問題中，當物種個數愈多時，在分枝與界限 (branch-and-bound) 的計算過程中所產生的分枝將會成長的非常快速，舉例來說 10 個物種的分枝數目會遠大於 10^7、15 個物種的分枝數目會遠大於 10^{13}、而 20 個物種的分枝數目更到達 10^{21}。如此一來，我們的最終目標求出人類粒腺體 136 個物種的最佳解，其分枝數目將會大於 10^{268}。在這個觀察下，要如何減少物種數目或抑合併物種都是一個重要的議題。在計畫的執行過程中，我們試著利用緊湊集合 (Compact Set) 方法來減少或合併物種的數目，緊湊集合分群法是屬於分群演算法中的階層演算法，基於圖形理論中緊密集合的觀念，使用緊湊集合分群法我們可以找出內聚力較強的集合。

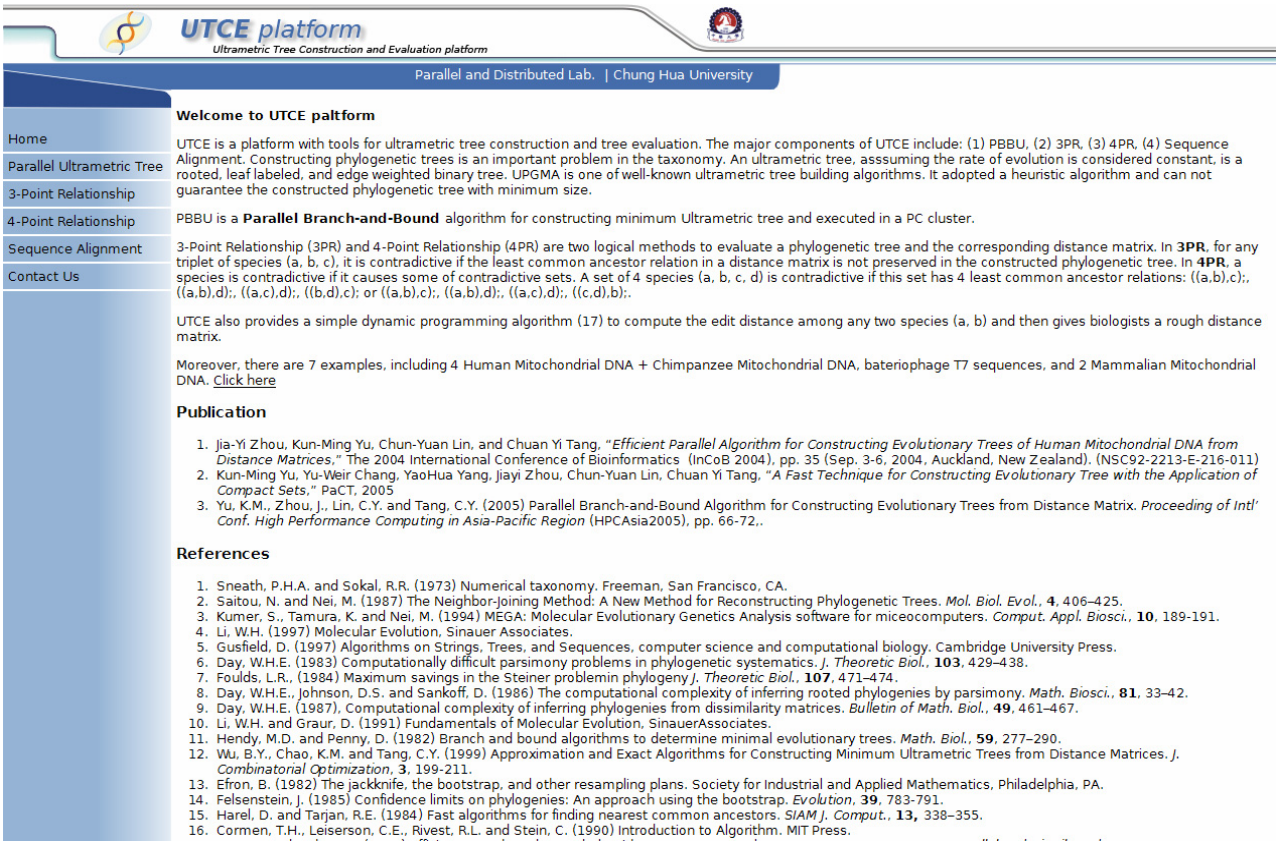
除此之外，我們還進行包括了 3-3 Relationship 以及 4 Points Relationship 的研究。在 3-3 Relationship 中，我們期望在建樹過程中參考原始的距離矩陣的 3 點間關係，期望在分枝時就能夠辨別及決定是否要做分枝的動作，如此一來，可以快速及大量的減少候選樹的數目；在 4 Points Relationship 中，我們希望能夠排除矛盾點，以減少物種的數目，希望能夠更快速的建立演化樹，也亦希望能夠評估一個結果樹的好壞。

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。

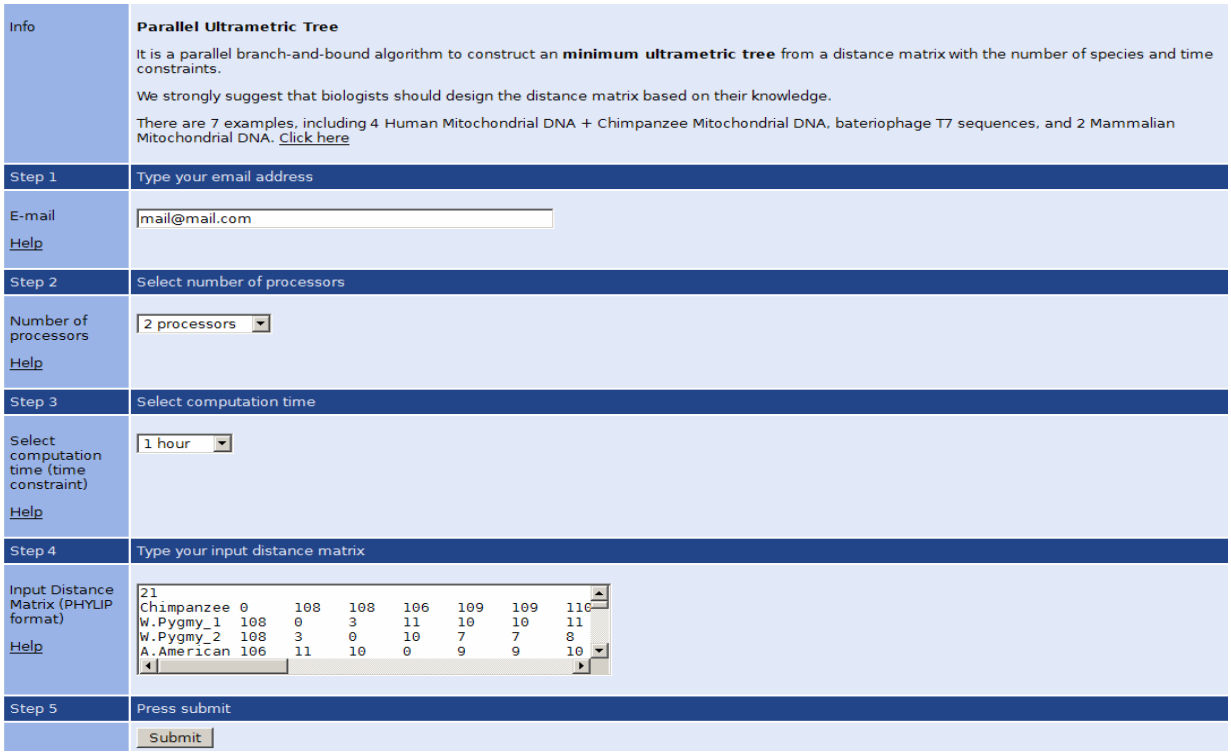
而利用分支與界定演算法進行解決問題的重點在於分支（Branch）與界定（Bound）的方法選擇決定上。於平行分散式系統中，往往會將原始問題分成數個可行解（Feasible solution）的子問題（Subproblem），而分支後的一個或數個可行解分配給數個運算處理單元計算，因此分支的動作將會影響負載平衡問題。因此，我們在研究的進行中使用了 global pool 及 local pool 做為一種負載平衡的機制，讓計算節點不至於有閒置。而在我們的系統架構中，我們用的是 master / slave 的架構，並且資料是在執行期間由 master 來指派，以提高系統整體運算效率。

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已順利完成預期的計畫目標，亦已將所獲得的研究成果整理成論文並且發表了三篇國際研討會論文(二篇為 SCI)，一篇國內研討會論文(EGBS 2006) ，並且亦已將所得之最佳演化樹的建構成果撰寫工具程式並以網站的形式提供給相關領域之專家學者研究之用(UTCE: Ultrametric Tree Construction and Evaluation platform)，有效分享本計畫之執行成果，UTCE 之操作畫面如圖一至五所示，本年度的計畫執行成果可說是相當完整且豐碩。



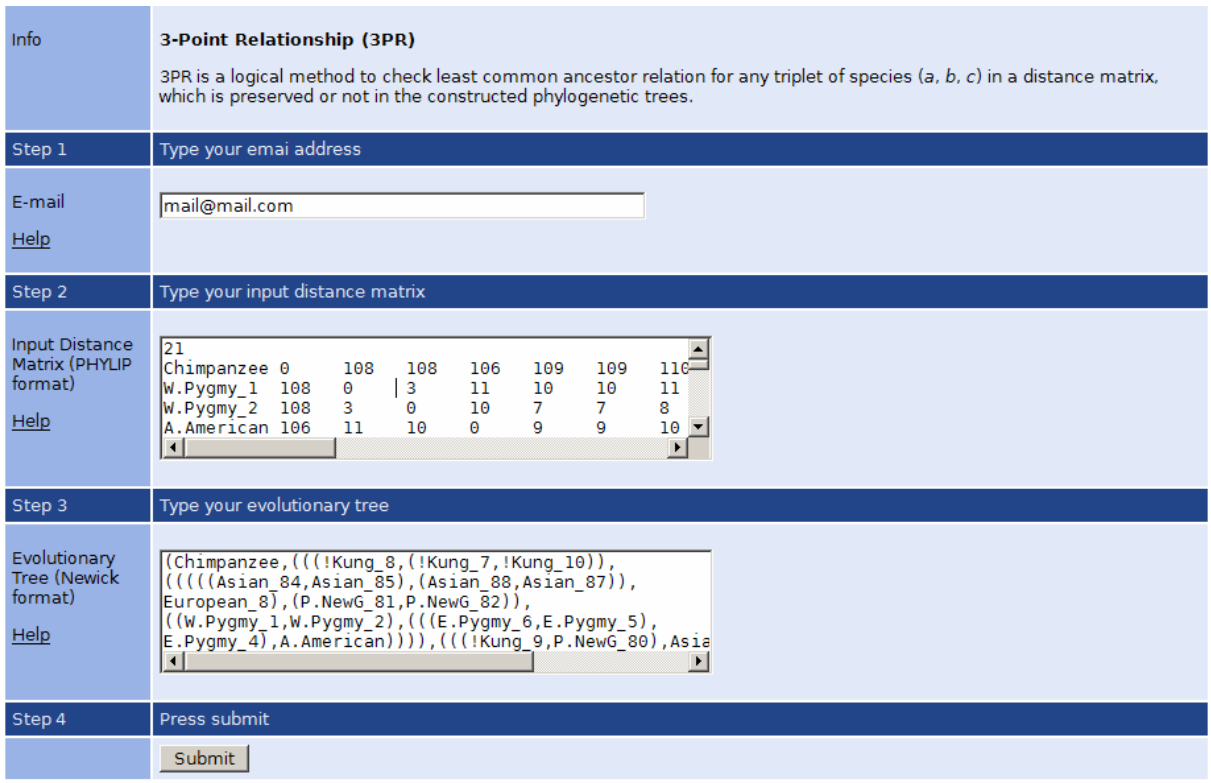
圖一：Ultrametric Tree Construction and Evaluation platform 首頁畫面



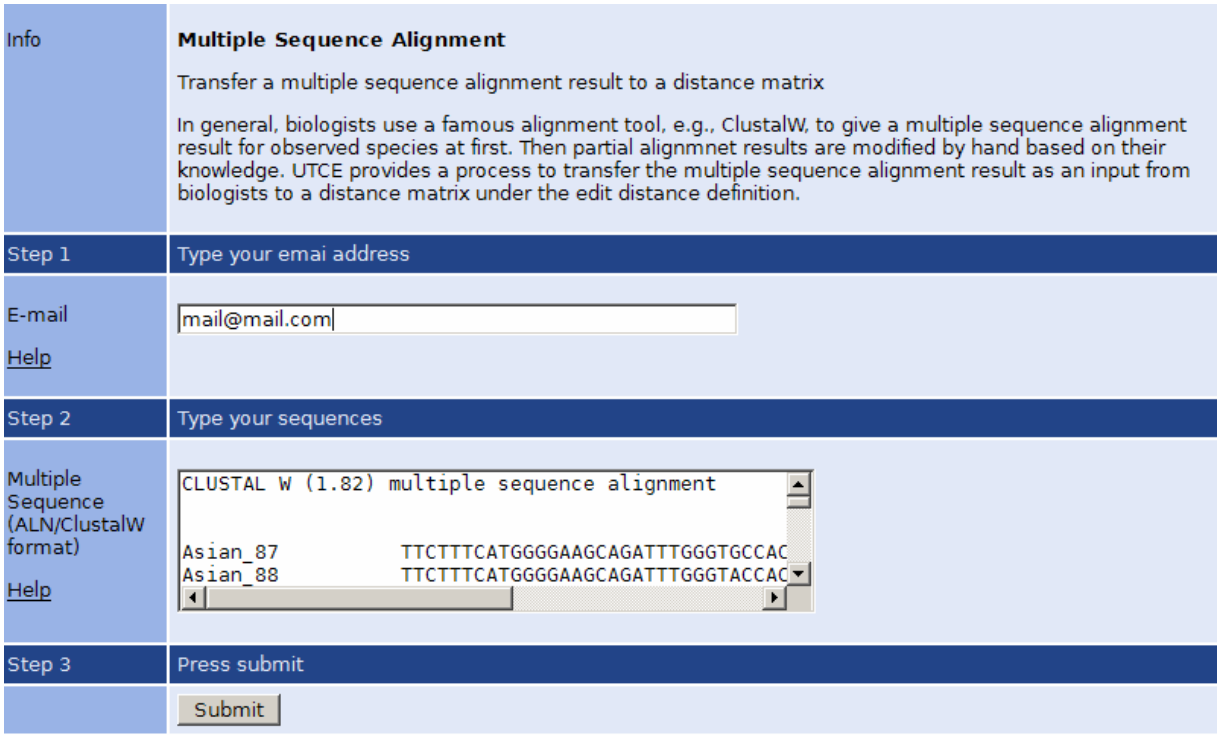
圖二：Parallel Ultrametric Tree 建構使用者輸入介面

```
Initial bound got from UPGMM: 542
The output is optimal ultrametric tree.

----- Ultrametric Tree -----
(Chimpanzee,(((!Kung_8,(!Kung_7,!Kung_10)),(((((Asian_84,Asian_85),(Asian_88,Asian_87)),European_8),(P.NewG_81,P.N
(0,(((8,(7,10)),(((((15,16),(19,18)),20),(12,13)),((1,2),(((6,5),4),3)))),(((9,11),17),14)));
Cost: 542
(Chimpanzee,(((!Kung_8,(!Kung_7,!Kung_10)),(((((Asian_84,Asian_85),European_8),(Asian_88,Asian_87)),(P.NewG_81,P.N
(0,(((8,(7,10)),(((((15,16),20),(19,18)),(12,13)),((1,2),(((6,5),4),3)))),(((9,11),17),14)));
Cost: 542
(Chimpanzee,(((!Kung_8,(!Kung_7,!Kung_10)),((((Asian_84,Asian_85),(European_8,(Asian_88,Asian_87))),(P.NewG_81,P.N
(0,(((8,(7,10)),((((15,16),(20,(19,18))),(12,13)),((1,2),(((6,5),4),3)))),(((9,11),17),14)));
Cost: 542

----- Input distance matrix -----
21
Chimpanzee      0   108   108   106   109   109   110   102   112   121   101   125   106   106   135   107   107
W.Pygmy_1     108     0     3    11    10    10    11    17    27    34    17    38    23    25    48    22    22
W.Pygmy_2     108     3     0    10     7     7     8    18    28    35    18    37    22    24    45    19    21
A.American    106    11    10     0     9     9    10    19    29    35    19    37    23    25    47    26    22
E.Pygmy_4     109    10     7     9     0     2     3    18    26    35    18    35    22    24    43    21    21
E.Pygmy_5     109    10     7     9     2     0     1    18    28    36    18    36    22    24    44    21    21
E.Pygmy_6     110    11     8    10     3     1     0    19    29    37    19    37    23    25    45    22    22
!Kung_7       102    17    18    19    18    18    19     0    12    21     2    29    16    19    42    17    15
!Kung_8       112    27    28    29    26    28    29    12     0    27    14    35    26    29    48    27    25
!Kung_9       121    34    35    35    35    36    37    21    27     0    23    10    35    38    23    36    34
!Kung_10      101    17    18    19    18    18    19     2    14    23     0    29    16    19    42    17    15
P.NewG_80     125    38    37    37    35    36    37    29    35    10    29     0    27    33    19    30    26
P.NewG_81     106    23    22    23    22    22    23    16    26    35    16    27     0     8    42    15    13
P.NewG_82     106    25    24    25    24    24    25    19    29    38    19    33     8     0    38    17    17
Hadza_83      135    48    45    47    43    44    45    42    48    23    42    19    42    38     0    41    40
Asian_84      107    22    19    26    21    21    22    17    27    36    17    30    15    17    41     0    10
Asian_85      107    22    21    22    21    21    22    15    25    34    15    26    13    17    40    10     0
Asian_86      127    43    42    44    39    41    42    36    38    17    36    11    34    38    23    29    27
Asian_87      107    26    23    26    23    23    24    19    29    38    19    30    17    21    43    14    12
Asian_88      111    22    19    24    21    21    22    17    27    36    17    28    15    18    40    12    10
European_8    111    25    22    24    21    21    22    18    28    33    18    25    16    20    38    13    11
```

圖三： Parallel Ultrametric Tree 建構完成之使用者輸出畫面



圖四：3PR 關係限制之建構使用者輸入介面



圖五：ＭＳＡ使用者輸入介面

附錄一：研討會論文

國際研討會論文

1. Kun-Min Yu, Jiayi Zhou, Chun-Yuan Lin and Chuan Yi Tang, “An Efficient Parallel Algorithm for Ultrametric Tree Construction Based on 3PR,” Parallel and Distributed and Processing and Applications, - Lecture Notes in Computer Science, Vol. 4331, pp. 215 – 220, Springer-Verlag, Dec. 2006, (SCI). (NSC-94-2213-E-216-028).
2. The 2006 IAENG International Workshop on Computer Science , 優秀論文(The Certificate of Merit) 論文名稱 ：“An Efficient Scheduling Algorithm for Irregular Data Redistribution,” Authors: Kun-Ming Yu and Yi-Lin Tsai. (IMECS 2006, pp. 270-275, 6/20 ~ 6/22, 2006, Hong Kong).

國內研討會論文

1. The Evolutionary Genomics and Bioinformatics Symposium and Workshop, Excellent Poster Prize, 論文名稱 ：“UTCE: Ultrametric Tree Construction and Evaluation platform,” Authors: Kun-Ming Yu, Jiayi Zhou, Chun-Yuan and Chuan Yi Tang. (EGBS 2006, pp. 27, 8/15 ~ 8/17, 2006, Taipei, Taiwan). (NSC-94-2213-E-216-028)
2. 游坤明、徐蓓芳、賴威廷、謝一功、周嘉、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, NCS’2005, Abs. pp. 51. (台南, 崑山科技大學, 12/15 ~ 12/16, 2005), (NSC-93-2213-E-216-037、NSC-94-2213-E-216-028).

An Efficient Parallel Algorithm for Ultrametric Tree Construction Based on 3PR*

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

[1] Department of of Computer Science and Information Engineering, Chung Hua University
[2] Institute of Engineering Science, Chung Hua University
[3] Institute of Molecular and Cellular Biology, National Tsing Hua University
[4] Department of Computer Science, National Tsing Hua University
300, Hsinchu, Taiwan, R.O.C
[1] yu@chu.edu.tw, [2] jyzhou@pdlab.csie.chu.edu.tw,
[3] cyulin@mx.nthu.edu.tw, [4] cytang@cs.nthu.edu.tw

Abstract. In the computational biology and taxonomy, to construct phylogenetic tree is an important problem. A phylogenetic tree can represent the relationship and histories for a set of species and helpful for biologists to observe existent species. One of popular model is ultrametric tree, and it assumed the evolution rate is constant. UPGMA is one of well-known ultrametric tree algorithm. However, UPGMA is a heuristic algorithm, and it can not guarantee the constructed tree is minimum size. To construct minimum ultrametric tree (MUT) has been shown to be an NP-hard problem. In this paper, we propose an efficient parallel branch-and-bound algorithm with 3-Point Relationship (3PR) to reduce the construction time dramatically. 3PR is a relationship between a distance matrix and the constructed phylogenetic tree. The main concept is for any two species closed to each other in a distance matrix should be also closed to each other in the constructed phylogenetic tree. We use this property to mark the branching path with lower priority or higher, then we move the lower ranked branching path to delay bound pool instead of remove it to ensure the optimal solution can be found. The experimental results show that our parallel algorithm can save the computing time and it also shows that parallel algorithm with 3PR can save about 25% of computing time in average.

Keywords: phylogenetic tree, minimum ultrametric tree, parallel branch-and-bound algorithm, 3-point relationship, 4-point relationship.

1 Introduction

To construct phylogenetic trees is an important problem in the computational biology and in taxonomy, the phylogenetic tree can represent the histories for a set of species and helpful for biologists to observe existent species or evaluate the relationship of them. However, the real evolutionary histories are unknown in practice. Therefore, many methods had been proposed and tried to construct a meaningful phylogenetic tree, which is closing to the real one.

* The work is partially supported by National Science Council. (NSC 94-2213-E-216 -028).
** The corresponding author.

G. Min et al. (Eds.): ISPA 2006 Ws, LNCS 4331, pp. 215 – 220, 2006.

In the input of distance matrix, a phylogenetic tree is constructed according to the distance matrix [10,11]. In general, these values are edit distances between two sequences of any two species. There are many different models and motivated algorithmic problems were proposed [1,9]. However, most of optimization problems for phylogenetic tree construction have been show to be NP-hard [2-4,6,7]. An important and commonly used model is assumed that the rate of evolution is constant. Based on this assumption, the phylogenetic tree will be an ultrametric tree (*UT*), which is rooted, leaf labeled, and edge weighted binary tree. Because many of these problems are intractable and NP-hard, biologists usually construct the trees by using heuristic algorithm. The Unweighted Pair Group Method with Arithmetic mean (*UPGMA*, [1]) is one of the popular heuristic algorithms to construct *UTs*.

Although construct *MUTs* is an NP-hard problem, it is still worthy to construct for middle-size of species. Thus, it seems possible to find an optimal tree using exhaustive search. Nevertheless, for n species, the number of rooted and leaf label tree is, it grows very rapidly. For example, $A(10) > 10^7$, $A(20) > 10^{21}$, $A(30) > 10^{37}$. Hence, it is impossible to exhaustively search for all possible trees even n are middle-size. Wu *et al*. [13] proposed a branch-and-bound algorithm for constructing *MUTs* to avoid exhaustive search. The branch-and-bound strategy is a general technique to solve combinatorial search problems.

In this paper, 3-Point Relationship (3PR) is used to construct *MUTs* more efficiently. 3PR is the relationship between a distance matrix and the constructed phylogenetic tree. The concept is that in triplet of species (a, b, c), any of two species which is closed to each other in the distance matrix should aslo be closed to each other in the constructed phylogenetic tree in a distance matrix. The experimental results show that *PBBU* with 3PR can reduce about 25% computation time both in sequential and parallel algorithms.

The paper is organized as follows. In section 2, some preliminaries for sequential branch-and-bound algorithm and 3PR are given. Parallel algorithm is described in section 3. Section 4 shows our experimental results, and final section is our conclusions.

2 Preliminaries

In this paper, we present *PBBU* with 3PR for construct minimum ultrametric tree. In the following, we denote an unweighted graph G=(V,E,w) with a vertex set V, an edge set E, and an edge weight function w. Some definitions are given as follows:

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

***Definition* 2:** Let $T = (V, E, w)$ 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 $w(T) = \sum_{e \in E} w(e)$.

***Definition* 3:** For any M (not necessarily a metric), *MUT* for M is T with minimum $w(T)$ 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.

***Definition* 4:** 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 $x \to y$ if and only if x is an ancestor of y.

***Definition* 5:** The distance between distance matrix and rooted topology of phylogenetic 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 $1 \le i, j, k \le n$. Otherwise is contradictory.

2.1 Sequential Branch-and-Bound Algorithm for MUTs

In the *MUT* construction problem, the branch-and-bound is a tree search algorithm and repeatedly searches the branch-and-bound tree (*BBT*) [8,14] to find a better solution until optimal one is found. The *BBT* is a tree which can represent a topology of *UTs*. Assume that the root of *BBT* has depth 0, hence each node with depth i in BBT represents a topology with a leaf set $\{1,...,i+2\}$.

2.2 3-Point Relationship (3PR)

3PR is a logical method to check the *LCA* relation for any triplet of species (a, b, c) in a distance matrix, which is preserved or not in the constructed phylogenetic trees. For any two species (a, b), $LCA(a, b)$ denotes the least common ancestor of (a, b). If (x, y) are two nodes in a phylogenetic tree, $x \to y$ is written if x is an ancestor of y. For a triplet of species (a, b, c) in the distance matrix M, if the distance $M[a, b]$ of species a and b is less than $M[a, c]$ and $M[b, c]$, $LCA(a, c)=LCA(b, c) \to LCA(a, b)$ (as ((a, b), c); in Newick tree format). For a triplet of species (a, b, c), it is contradictive if the least common ancestor relation in a distance matrix is not preserved in the constructed phylogenetic tree. *3PR* can be used to evaluate the qualities of constructed phylogenetic trees. A phylogenetic tree is considered unreliable if the number of contradictive triplets is large. The evaluated result may be useful for biologists to choose a feasible phylogenetic tree construction tool.

3 Parallel Branch-and-Bound Algorithm with 3PR

Parallel Branch-and-Bound Algorithm with *3PR* (*PBBU* with *3PR*) is designed on distributed memory multiprocessors and the master-slave architecture. The *PBBU* uses a branch-and-bound technique to avoid exhaustive search of possible trees. For load-balance purpose, the master processor (*MP*) contains a *Global Pool* and each slave processor (SP) has Local Pool, moreover we use new data structure instead of the link list to store *BBT*.

In [5], *3PR* is applied as a tree evaluation method. We use this property to put lower rank branching path to Delay Bound Pool (*DBP*) when selecting branch path in the branch-and-bound algorithm. For example, Table 1 is the distance matrix and Figure 1 shows two candidates when inserting the third species c. In *PBBU* without *3PR*, both (a) and (b) candidates need to be added to the pool when branching. However, topology of (b) is closing to distance matrix, it obtained higher rank, and (a) has lower rank. In *PBBU* with *3PR*, only (b) (with higher rank) candidate will be

selected due to the distance of *a* and *c* is greater than the distance of *b* and *c*. This result is based on the conception that in a triplet of species (*a*, *b*, *c*), any of two species which is closed to each other in the distance matrix should also be closed to each other in the corresponding phylogenetic tree in a distance matrix. However, it cannot be directly used to bound another branching path, and *PBBU* with *3PR* put others candidates to the *DBP* to ensure the optimal solution can be found.

Table 1. Distance matrix

	a	*b*	*c*
a	0	25	20
b	25	0	15
c	20	15	0

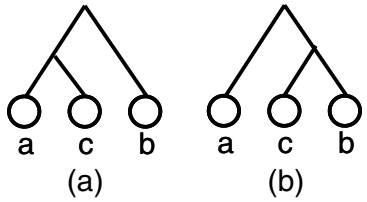


Fig. 1. Candidate *BBT*

4 Experimental Results

In the experimental results, we implement *PBBU* and *PBBU* with *3PR* on a Linux based PC cluster. Each computing node is an AMD Athlon PC with a clock rate of 2.0 GHz and 1GB memory. Each node is connected with each other by 100Mbps network. There are two data sets used to test our algorithms. One is a random data set, which is generated randomly. The distance matrix in the random data set is metric and the range of distances is between 1 and 100. Another is a data set composed of 136 Human Mitochondrial DNAs (*HMDNA*), which is obtained from [12]. Its distance matrix is metric and the range of distances is between 1 and 200. In order to eliminate the problems of data dependence, for each testing data, we run 10 instances. Then we compare the average, median, and worst cases.

Figure 2 and 3 show that *PBBU* with *3PR* and delay bound technique can find the optimal solution and save about 25% of computation time than *PBBU* without *3PR*. Because *3PR* technique move lower ranking candidates which disaccording to *3PR* to delay bound pool, after that, the better bounding value can be found early. Afterward it can bound more candidates to decreasing computation time.

Figure 4 is the speed-up ratio of *HMDNA* data set. We observed that the speed-up ratio of *3PR* is better than it without *3PR*. Furthermore, the difference between *3PR* and without *3PR* is larger when the number of processors increasing. Because of the tighter bounding value can be found quickly with more processors. It also shows that our algorithm is scalable in large number of computing resources. Figure 5 shows the computation time of 16 processors of *PBBU* with *3PR* for different number of species. We can observe that the computation time grow rapidly when the number of species increasing. Moreover, the reduced proportion between *PBBU* and *PBBU* with *3PR* is increasing with larger number of species. We consider that large number of species contains more candidates that a tighter bounding value which can be obtained from *3PR* technique can also bound grater number of candidates; it can decreasing the computation time.

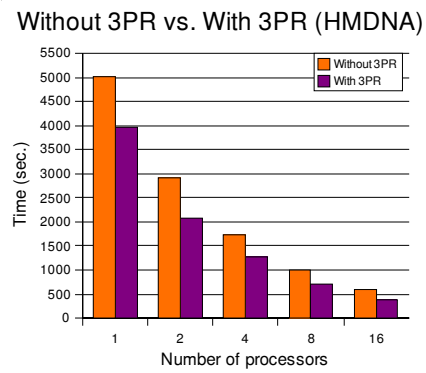


Fig. 2. 3PR vs. Without 3PR (HMDNA)

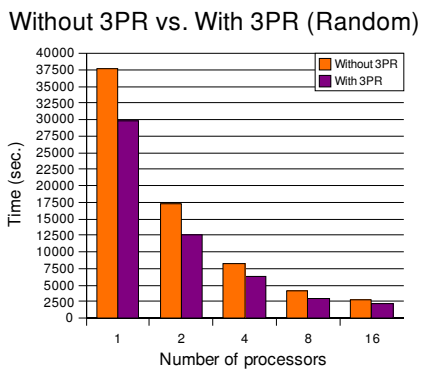


Fig. 3. 3PR vs. Without 3PR (Random)

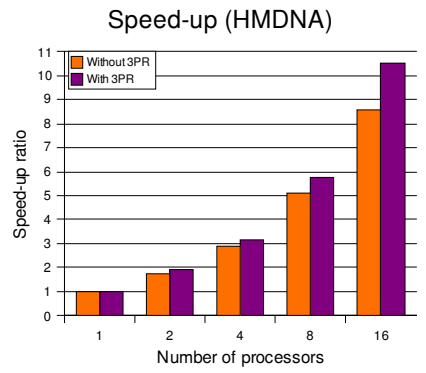


Fig. 4. Speed-up ratio (HMDNA)

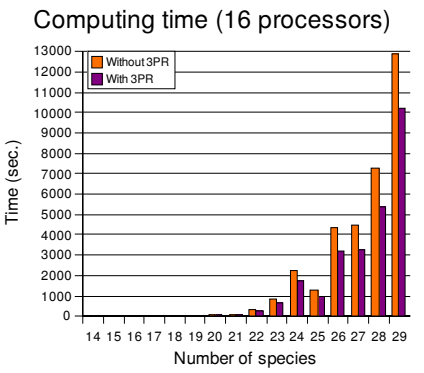


Fig. 5. Computing time (16 processors)

5 Conclusions

In this paper, we have designed *PBBU* with *3PR* for constructing *MUTs* problem. The *3PR* is the relationship between distance matrix and constructed evolutionary tree. It moves candidates which do not fit *3PR* to delay bound pool in branch-and-bound algorithm. After that, we can obtain the tighter bounding value quickly and uses it to bound more candidates. In order to evaluate the performance of our proposed algorithm, a random data set and a practical data set of *HMDNA* are used. The experimental results show that *PBBU* with *3PR* can find optimal solution for 36 species within a reasonable time on 16 PCs. Furthermore, the speed-up ratio shows the performance of our algorithm is good in our PC cluster environment. Moreover, the results also show that *PBBU* with *3PR* can save about 25% in average of computing time than *PBBU* without *3PR*, and it assured the results are optimal with the delay bound technique.

References

1. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein “Introduction to Algorithm,” MIT Press, 1990.
2. W.H.E. Day “Computationally difficult parsimony problems in phylogenetic systematics,” J. Theoretic Biol., 103, 1983, pp.429-438.
3. W.H.E. Day “Computational complexity of inferring phylogenies from dissimilarity matrices,” Bulletin of Math. Biol., 49, 1987, pp.461-467.
4. W.H.E. Day, D.S. Johnson, and D. Sankoff “The computational complexity of inferring rooted phylogenies by parsimony,” Math. Biosci., 81, 1986, pp.33-42.
5. C.T. Fan, “The evaluation model of evolutionary tree,” Master Thesis, Nationa Tsing Hua University, 2000.
6. L.R. Foulds “Maximum savings in the Steiner problemin phylogeny,” J. Theoretic Biol., 107, 1984, pp.471-474.
7. D. Gusfield “Algorithms on Strings, Trees, and Sequences, computer science and computational biology,” Cambridge University Press, 1997.
8. M.D. Hendy and D. Penny “Branch and bound algorithms to determine minimal evolutionary trees,” Math. Biol., 59, 1982, pp.277-290.
9. S. Kumer, K. Tamura, M. Nei “MEGA: Molecular Evolutionary Genetics Analysis software for miceocomputers,” Comput. Appl. Biosci., 10, 1994, pp.189-191.
10. W.H. Li “Molecular Evolution,” Sinauer Associates, 1997.
11. R.D.M. Page “TreeView: An application to display phylogenetic trees on personal computers,” Comput. Appl. Biosci., 12, 1996, pp.357-358.
12. L. Vigilant, M. Stoneking, H. Harpending, K. Hawkes and A.C. Wilson “African Populations and the Evolution of Human Mitochondrial DNA,” Science, 253, 1991, pp.1503-1507.
13. B.Y. Wu, K.M. Chao, and C.Y. Tang “Approximation and Exact Algorithms for Constructing Minimum Ultrametric Trees from Distance Matrices,” J. Combinatorial Optimization, 3, 1999, pp.199-211.
14. C.F. Yu and B.W Wah “Efficient Branch-and-Bound Algorithms on a Two-Level Memory System,” IEEE Trans. Parallel and Distributed Systems, 14, 1988, pp.1342-1356.

An Efficient Scheduling Algorithm for Irregular Data Redistribution

Kun-Ming Yu and Yi-Lin Tsai

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

Abstract. Dynamic data redistribution is used to enhance the performance of an algorithm and to achieve data locality in parallel programs on distributed memory multi-computers. Therefore, the data redistribution problem has been extensively studied. Previous results focus on reducing index computational cost, schedule computational cost, and message packing/unpacking cost. However, irregular data redistribution is more flexible than regular data redistribution; it can distribute different sizes of data segments of each processor to those processors according to their own computation capability. High Performance Fortran 2 (HPF-2), the current version of HPF, provides an irregular distribution functionality, such as GEN_BLOCK which addresses some requirements of irregular applications for the distribution of data in an irregular manner and explicit control of load balancing. In this paper, we present a degree-reduction-and-coloring (DRC) algorithm for scheduling HPF2 irregular array redistribution. We devoted to obtain the minimal number of transmission steps as well as to reduce the overall redistribution time. The proposed algorithm intends to reduce the number of maximum transmission messages in the first phase and then applies graph-coloring mechanism to obtain the final schedule. The proposed method not only avoids node contention, but also shortens the overall redistribution time. To evaluate the performance of DRC algorithm, we have implemented DRC algorithms along with the Divide-and-Conquer algorithm. The simulation results show that DRC algorithm has significant improvement on communication costs compared with the Divide-and-Conquer algorithm.

Keywords: Irregular redistribution, communication scheduling, GEN_BLOCK, degree-reduction

1. Introduction

Parallel computing systems have been widely adopted to solve complex scientific and engineering problems. To efficiently execute a data-parallel program on distributed memory multi-computers, an appropriate data distribution is critical to the performance. Appropriate distribution can balance the computational load, increase data locality, and reduce inter-processor communication. Array redistribution is crucial for system performance because a specific array distribution may be appropriate for the current phase, but incompatible for the subsequent one. Many parallel programming languages thus support run-time primitives for rearranging the array distribution of a program. The data redistribution problem has been widely studied in the literature. In general, data redistribution can be classified into two categories: the regular data redistribution [1,5,6,7,9,11,13,15,18] and the irregular data redistribution [4,8,22-24]. The regular data redistribution decomposes data of equal sizes into processors. There are three types of this data redistribution, called BLOCK, CYCLIC, and BLOCK-CYCLIC(n). The irregular data distribution employs user-defined functions to specify data distribution unevenly. High Performance FORTRAN 2 (HPF-2) provides GEN_BLOCK functionality and makes it possible to handle different processors dealing with appropriate data size according to their computation capability. Previous works emphasized the minimal steps of data redistribution and scheduled the ordering of messages with minimal total transmission size. In the regular array redistribution, [15] proposed an Optimal Processor Mapping (OPM) scheme to minimize the data transmission cost for general BLOCK-CYCLIC regular data realignment. Optimal Processor Mapping (OPM) utilized the maximum matching of realignment logical processors to achieve the maximum data hits for reducing the amount of data exchange transmission cost. In the irregular array redistribution problem, [22, 23] proposed a greedy algorithm to utilize the Divide-and-Conquer technique to obtain near optimal scheduling while attempting to minimize the size of total communication messages as well as the number of steps.

In this paper, we bring up the Degree-Reduction-and-Coloring (DRC) algorithm to efficiently perform GEN_BLOCK array redistribution. In section 2, we define the data communication model of irregular data redistribution and give an example of GEN_BLOCK data redistribution as the preliminary. Section 3 describes the DRC algorithm for the irregular redistribution problem. The performance analysis, simulation results and practical transmission with MPI on SMP/Linux cluster are presented in section 4. Finally, the conclusions are given in section 5.

2. Data communication models

In this section, we present some properties of irregular data redistribution with GEN_BLOCK functionality. There are no repetitive communication

patterns in the irregular GEN_BLOCK array redistribution. A data redistribution can be represented by a bipartite graph, called a *redistribution graph*. To simplify the presentation, notations and terminologies used in this paper are defined in the following.

***Definition* 1**: Given an irregular GEN_BLOCK redistribution on array $A[SPi]$ and $A[DPi]$ over P processors, the *source processors* of array data elements $A[SPi]$ are denoted as SPi; the *destination processors* of array elements $A[DPi]$ are denoted as DPi where $1 \le i \le P$.

***Definition* 2**: A bipartite graph $G = (V, E)$ is used to represent the communications of an irregular data redistribution between source and destination processors. Vertices of G are used to represent the processors. Edge e_{ij} in G denotes the message sent from SPi to DPj, where $e_{ij} \in E$. $|E_{ij}|$ denotes the transmission message size through the redistribution.

***Definition* 3**: Every message transmission link in irregular data redistribution is not overlapped. Hence, the total number of message transmission link E is $P \le E \le 2 \times P - 1$.

***Definition* 4**: Each processor has more than one e_{ij} to send data to destination processors or receive data from other source processors. The number D of e_{ij} owned by one processor is denoted as *D-degree* and the maximum *D-degree* of all processors is denoted as *Max-degree*. We denote that the processors have the *Max-degree* number of messages as P_{max}.

***Definition* 5**: If SPi sends messages to $DPj-1$ and $DPj+1$, the transmission between SPi and DPj must exist, where $1 \le i, j \le P$. This result was mentioned as the consecutive communication property [12].

Fig.1 shows an example of redistributing two GEN_BLOCK distributions on $A[SPi]$ and $A[DPi]$. Table 1(a) shows mapped communication message size to source processors and destination processors, respectively. The communications between source and destination processor sets are depicted in Fig 1. There are 13 transmission messages, e_{11}, e_{21}, e_{22}, …e_{77} among the processors involved in the redistribution. Due to the considerable influence of node contention, a processor can only send at most one message to another processor in each communication step and the same is true for the receiving message. The messages, which cannot be scheduled in the same communication step, are called conflict tuple. For instance, $\{e_{11}, e_{21}\}$ is a conflict tuple since they have a common destination processor $DP1$; $\{e_{21}, e_{22}\}$ is also a conflict tuple because of the common source processor $SP2$. Table 1(b) shows a simple schedule result for this example.

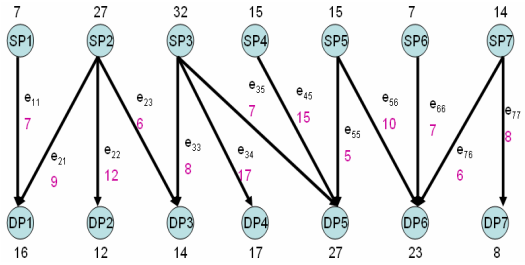


Figure 1. An example of data redistribution

Table 1(a). The total message size of redistribution data for each processor in Fig. 1.

SP1	SP2	SP3	SP4	SP5	SP6	SP7
7	27	32	15	15	7	14
DP1	DP2	DP3	DP4	DP5	DP6	DP7
16	12	14	17	27	23	8

Table 1(b). A simple schedule

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	e_{56}, e_{23}, e_{35}
Step3:	e_{21}, e_{33}, e_{55}, e_{76}

3. Proposed Algorithm

The performance of a data redistribution procedure is determined by four costs: index computational cost T_i, schedule computational cost T_s, message packing/unpacking cost T_p, and data transfer cost. The data transfer cost for each communication step consists of start-up cost T_{su} and transmission cost T_t. Let the unit transmission time τ denote the cost of transferring a message of unit length. In general, the message startup cost is directly proportional to the number of communication steps. The total number of communication steps is denoted by N. The total redistribution time equals $T_i+T_s+\sum_{i=1}^{N}(T_p + T_{su} + m_i\tau)$, where $m_i = \text{Max}\{e_1, e_2, e_3, .., e_{k-1}\}$ and e_j represents the size of message scheduled in the i^{th} communication step for $j = 1$ to k. In irregular redistribution, messages of varying sizes are scheduled in the same communication step. Therefore, the largest size of message in the same communication step dominates the data transfer time required for this communication step.

The main idea of the Degree-Reduction-and-Coloring (DRC) algorithm is to diminish the degree of P_{max} repeatedly by scheduling the message in the first step of data redistribution process until *Max-degree* is equal to 2. The remaining messages are then scheduled into the communication steps by utilizing the concept of bipartite graph coloring mechanism. The details of the steps will be described in the following subsections.

3.1 Degree-Reduction Step

The goal in this step is to reduce *Max-degree* repeatedly in each iteration, until *Max-degree* is equal to 2. An example of 4-degree communication redistribution has taken as shown in Fig 2. In the first phase (phase-1) of degree-reduction step, the messages are sorted by the non-increasing order of $|E_{ij}|$, and the result is shown in Table 2. Then, DRC selects the messages into step1 of the schedule according to non-increasing order of message size except those messages causing the conflict. After phase-1, the *Max-degree* will be decreased by 1 (from 4 to 3). Fig 3(a) and Table 3(b) show this scenario. DRC repeat the procedure until the *Max-degree* reaches 2, which is depicted in Fig 4.

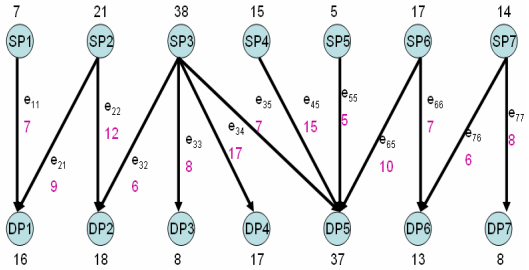
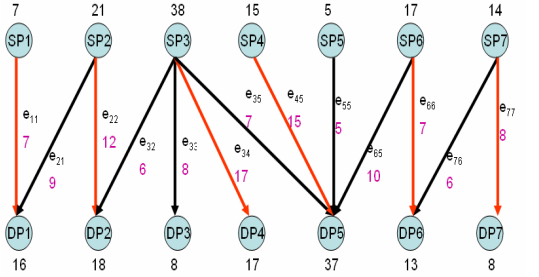


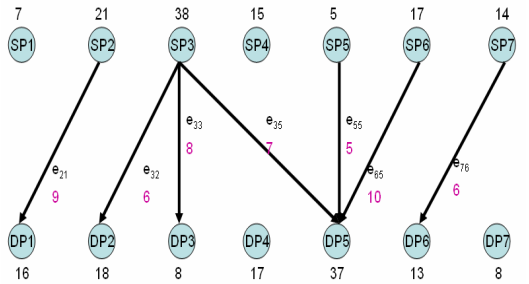
Figure 2. A data redistribution example with *Max-degree* = 4

Table 2. The messages are sorted by non-increasing order of message size

Msg no.	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Msg size	17	15	12	10	9	8	8	7	7	7	6	6	5



(a)



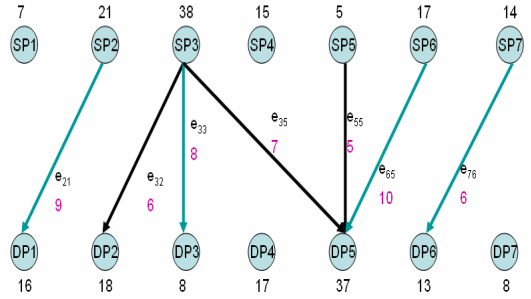
(b)

Figure 3. The messages communication (a) before phase-1 of the degree-reduction step; (b) after phase-1 of the degree-reduction step.

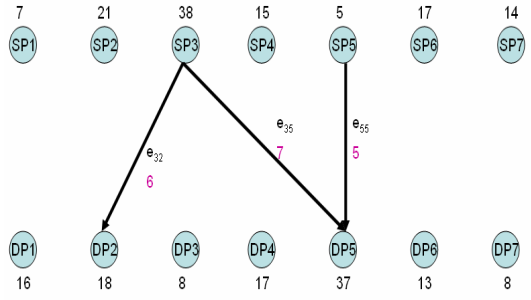
Table 4. The schedule after phase-1

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	
Step3:	
Step4:	

Message number	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Message size	17	15	12	10	9	8	8	7	7	7	6	6	5



(a)



(b)

Figure 4. The messages communication (a) before phase-2 of the degree-reduction step; (b) after phase-2 of the degree-reduction step.

Table 5. The schedule after the procedure of phase-2

Message no.	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Message size	17	15	12	10	9	8	8	7	7	7	6	6	5

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	e_{65}, e_{21}, e_{33}, e_{76}
Step3:	
Step4:	

3.2 Message-Coloring Step

After completing the degree-reduction step, we can obtain a redistribution graph with *Max-degree* of 2 and the resulting redistribution graph is *2-edge colorable* [2], since it is a bipartite graph and its maximum degree is equal to 2. In the Message-Coloring Step, DRC schedules the left messages into the same step in a non-increasing order to accomplish an optimal scheduling unless a conflict occurs. Figure 5 shows the outcome of message-coloring and Table 6 shows the final schedule obtained from DRC algorithm.

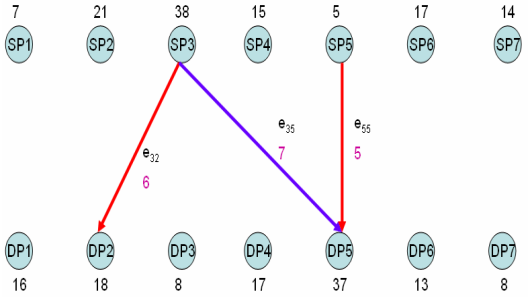


Figure 5. The outcome of redistribution graph after applying the message coloring mechanism

Table 6. The final schedule obtained from DRC

Msg no.	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Msg size	17	15	12	10	9	8	8	7	7	7	6	6	5

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	e_{65}, e_{21}, e_{33}, e_{76}
Step3:	e_{35}
Step4:	e_{32}, e_{55}

The algorithm of the Degree-Reduction-Coloring is given as follows.

==========

Algorithm DRC
generating messages;
// generate messages from *AS*[*Pi*] to *AD*[*Pi*]
step = **maximum degree;**
sort_msgSize();
// sorting in decreasing order by message size
while (*step* **> 2)**
{
choose_msg(*step***);**
// selecting message without conflict tuple, set into *(maximal degree - step + 1)* schedule step
step--
} // degree-reduction iteration
while (remaining_messages != null)
{
selecting_msg(*maximal degree*-1**);**
// selecting message set into *maximal degree-1* schedule step
check_msg_continue_set();
// check remaining message set
coloring_maximal_msg(*maximal degree***);**
// color the maximal message with degree *maximal degree -1* and the neighbor message with *maximal degree*
} // message coloring mechanism
end of DRCM

==========

4. Performance Evaluation

To evaluate the performance of the proposed methods, we have implemented the DRC along with the Divide-and-Conquer algorithm [23]. The performance simulation is discussed in two categories, even GEN_BLOCK and uneven GEN_BLOCK distributions. In even GEN_BLOCK distribution, each processor owns similar size of data. In contrast to even distribution, few processors might be allocated by grand volumes of data with uneven distribution. Since data elements could be centralized to some specific processors, it is also possible for those processors to have the maximum degree of communications.

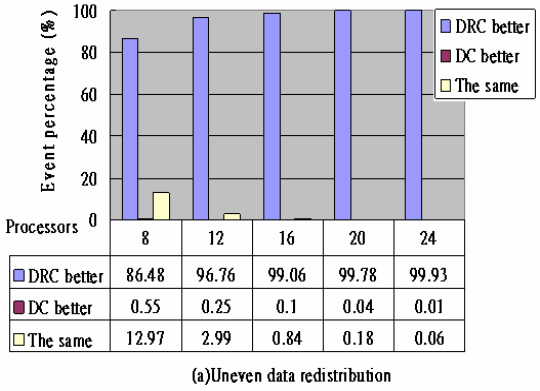
The simulation program generates a set of random integer number and the size of message as *A*[*SPi*] and *A*[*DPi*]. Moreover, the total message size sending from SPi equals to the total size receiving to DPi keeping the balance between source processors and destination processors.

We assume that the data computation (communication) time in the simulation is represented by the transmission size $|E_{ij}|$. In the following figures, the percentage of events is plotted as a function of the message size and the number of processors. Also, in the figures, “DRC Better” represents the percentage of the number of events that the DRC algorithm has lower total computation (communication) time than the Divide-and-Conquer algorithm, while “DC Better” gives the reverse situation. If both algorithms have the same total computation (communication) time, “The Same Results” represents the number of that event.

In the uneven distribution, the size of message’s up-bound is set to be B*1.7 and that of low-bound is set to be B*0.3, where B is equal to the sum of total transmission message size / total number of processors. In the even distribution, the size of message’s up-bound is set to be B*1.3 and that of low-bound is set to be B*0.7. The total message-size is 10M.

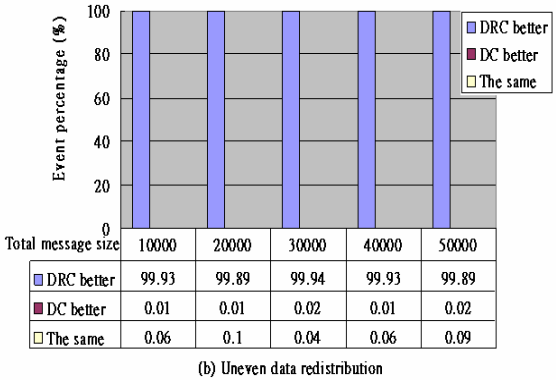
Fig 6(a) and 6(b) show the simulation results of both the DRC and the Divide-and-Conquer algorithm

with different number of processors and total message size. The number of processors is from 8 to 24. We can observe that the DRC algorithm has better performance in the uneven data redistribution compared with Divide-and-Conquer algorithm. Since the data is concentrated in the even case, from Fig 7(a) and 7(b), we can observe that DRC has better performance compared with the uneven case. In both even and uneven cases, DRC performs better than the Divide-and-Conquer algorithm.



Processors	8	12	16	20	24
DRC better	86.48	96.76	99.06	99.78	99.93
DC better	0.55	0.25	0.1	0.04	0.01
The same	12.97	2.99	0.84	0.18	0.06

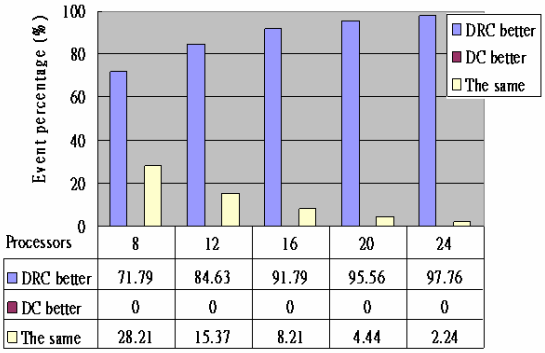
(a)Uneven data redistribution



Total message size	10000	20000	30000	40000	50000
DRC better	99.93	99.89	99.94	99.93	99.89
DC better	0.01	0.01	0.02	0.01	0.02
The same	0.06	0.1	0.04	0.06	0.09

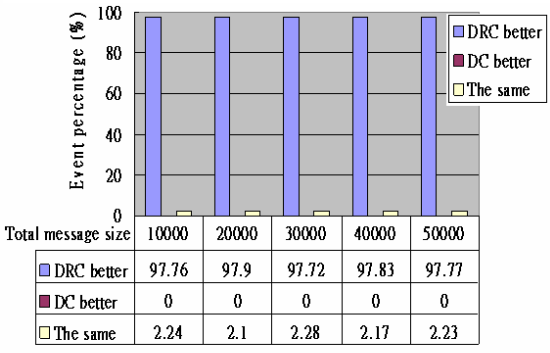
(b) Uneven data redistribution

Figure 6. The events percentage of computing time is plotted (a) with different number of processors and (b) with different number of total message sizes in 24 processors, on the uneven data set.



Processors	8	12	16	20	24
DRC better	71.79	84.63	91.79	95.56	97.76
DC better	0	0	0	0	0
The same	28.21	15.37	8.21	4.44	2.24

(a) Even data redistribution



Total message size	10000	20000	30000	40000	50000
DRC better	97.76	97.9	97.72	97.83	97.77
DC better	0	0	0	0	0
The same	2.24	2.1	2.28	2.17	2.23

(b) Even date redistribution

Figure 7. The events percentage of computing time is plotted (a) with different number of processors and (b) with different number of total message sizes in 24 processors, on the even data set.

5.Conclusion

In this paper, we have presented a Degree-Reduction-Coloring (DRC) scheduling algorithm to efficiently perform HPF2 irregular array redistribution on a distributed memory multi-computer. The DRC algorithm is a simple method with low algorithmic complexity to perform GEN_BLOCK array redistribution. The DRC algorithm is an optimal algorithm in terms of minimal number of steps. In the same time, DRC algorithm is also a near optimal algorithm satisfying 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 DRC as well as the Divide-and-Conquer redistribution algorithm. The experimental results show improvement in communication costs and high practicability on different processor hierarchies. Also, the experimental results indicate that both of them have good performance on GEN_BLOCK redistribution. In many situations, DRC is better than the Divide-and-Conquer redistribution algorithm.

Reference

[1] G. Bandera and E.L. Zapata, “Sparse Matrix Block-Cyclic Redistribution,” Proceeding of IEEE Int'l. Parallel Processing Symposium (IPPS'99), San Juan, Puerto Rico, 355 - 359 ,April 1999

[2] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, Macmillan, London, 1976.

[3] Frederic Desprez, Jack Dongarra and Antoine Petitet, “Scheduling Block-Cyclic Data redistribution,” IEEE Trans. on PDS, vol. 9, no. 2, pp. 192-205, Feb. 1998.

[4] Minyi Guo, “Communication Generation for Irregular Codes,” The Journal of Supercomputing, vol. 25, no. 3, pp. 199-214, 2003.

[5] Minyi Guo and I. Nakata, “A Framework for Efficient Array Redistribution on Distributed

Memory Multicomputers," The Journal of Supercomputing, vol. 20, no. 3, pp. 243-265, 2001.

[6] Minyi Guo, I. Nakata and Y. Yamashita, "Contention-Free Communication Scheduling for Array Redistribution," Parallel Computing, vol. 26, no.8, pp. 1325-1343, 2000.

[7] Minyi Guo, I. Nakata and Y. Yamashita, "An Efficient Data Distribution Technique for Distributed Memory Parallel Computers," Joint Symp. on Parallel Processing (JSPP'97), pp.189-196, 1997.

[8] 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.

[9] Edgar T. Kalns, and Lionel M. Ni, "Processor Mapping Technique Toward Efficient Data Redistribution," IEEE Trans. on PDS, vol. 6, no. 12, pp. 1234-1247, December 1995.

[10] S. D. Kaushik, C. H. Huang, J. Ramanujam and P. Sadayappan, "Multiphase data redistribution: Modeling and evaluation," International Parallel Processing Symposium (IPPS'95), pp. 441-445, 1995.

[11] Peizong Lee, Academia Sinica, and Zvi Meir Kedem, "Automatic Data and Computation Decomposition on Distributed Memory Parallel Computers," ACM Transactions on Programming Languages and systems, Vol 24, No. 1, pp. 1-50, January 2002.

[12] 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.

[13] Y. W. Lim, Prashanth B. Bhat and Viktor and K. Prasanna, "Efficient Algorithms for Block-Cyclic Redistribution of Arrays," Algorithmica, vol. 24, no. 3-4, pp. 298-330, 1999.

[14] C.-H Hsu, S.-W Bai, Y.-C Chung and C.-S Yang, "A Generalized Basic-Cycle Calculation Method for Efficient Array Redistribution," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 12, pp. 1201-1216, Dec. 2000.

[15] Ching-Hsien Hsu, Kun-Ming Yu, "An Optimal Processor Replacement Scheme for Efficient Communication of Runtime Data Realignment," Parallel and Distributed and Processing and Applications, - Lecture Notes in Computer Science, Vol. 3358, pp. 268-273, 2004.

[16] 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.

[17] Antoine P. Petitet and Jack J. Dongarra, "Algorithmic Redistribution Methods for Block-Cyclic Decompositions," IEEE Transactions on Parallel and Distributed Systems, vol. 10, no. 12, pp. 1201-1216, Dec. 1999

[18] Neungsoo Park, Viktor K. Prasanna and Cauligi S. Raghavendra, "Efficient Algorithms for Block-Cyclic Data redistribution Between Processor Sets," IEEE Transactions on Parallel and Distributed Systems, vol. 10, No. 12, pp.1217-1240, Dec. 1999.

[19] .L. Prylli and B. Touranchean, "Fast runtime block cyclic data redistribution on multiprocessors," Journal of Parallel and Distributed Computing, vol. 45, pp. 63-72, Aug. 1997.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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。

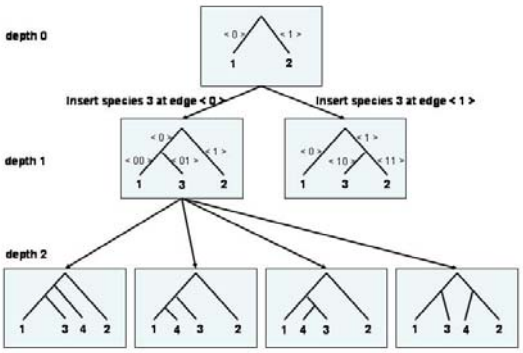


圖1. 建立分支界限樹 (BBT) [3]

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]。

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

表 1. GlobusToolkit 所提供的服務

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。

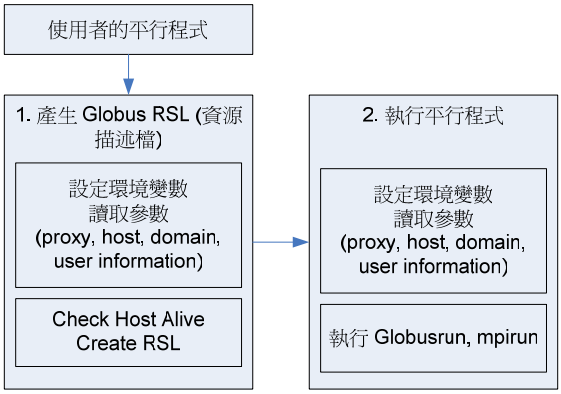


圖 2. UniGrid 上的程式流程

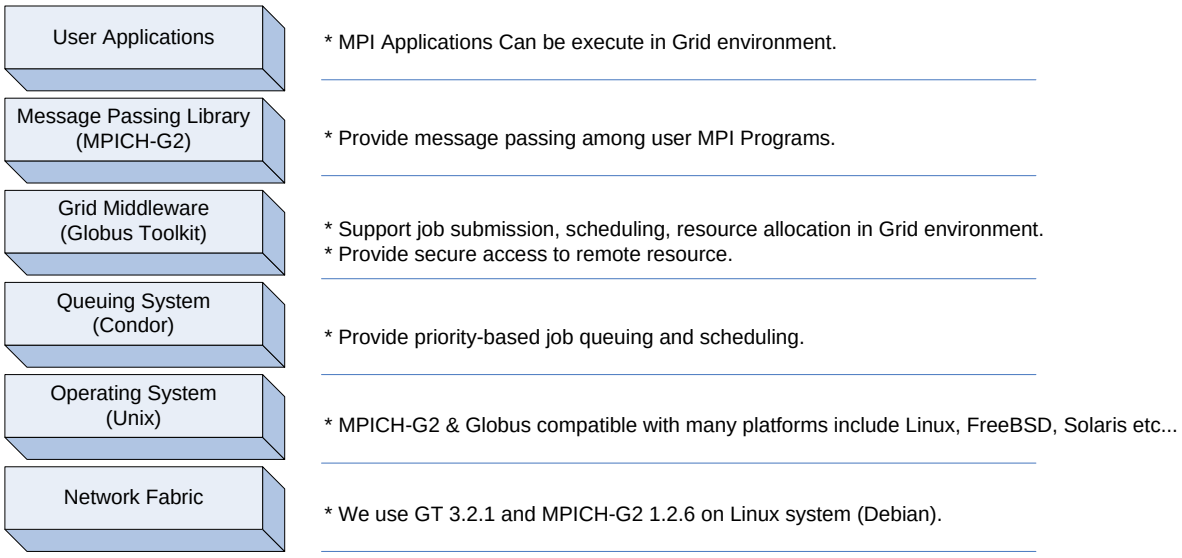


圖 3 .UniGrid 的架構圖[2]

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送的程式觀念。

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

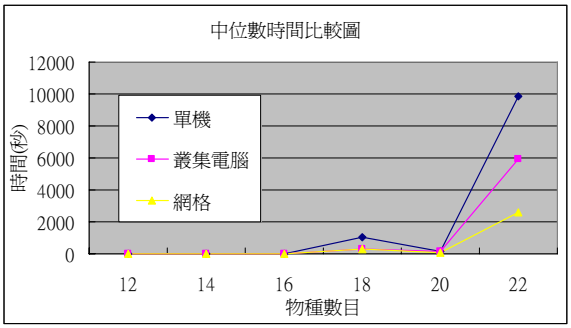


圖 4. 中位數時間比較圖

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

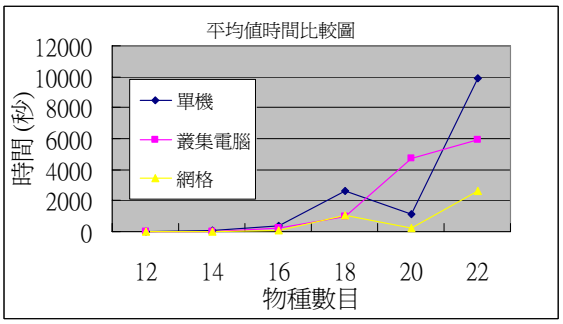


圖 5. 平均數時間比較圖

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

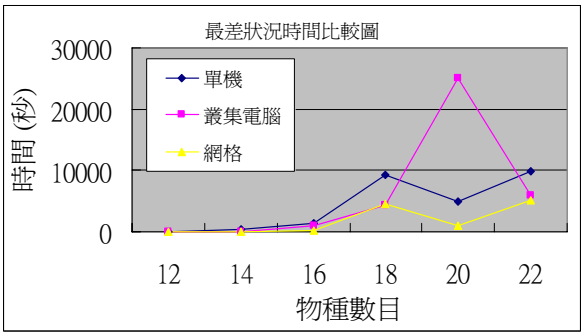


圖 6. 最差狀況時間比較圖

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 組物種數目 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

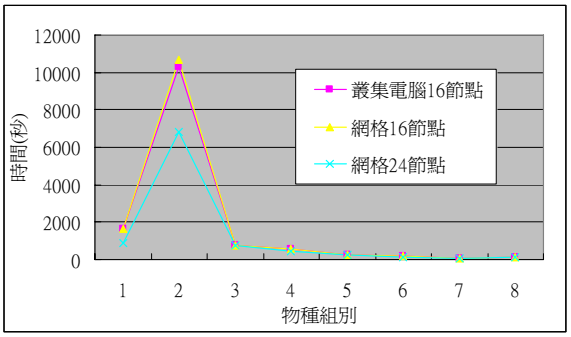


圖 7. 叢集電腦與網格使用不同節點數比較圖

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/)

17

UTCE: Ultrametric Tree Construction and Evaluation platform

Kun-Ming Yu, Jayi Zhou, Chun-Yuan Lin, Chuan Yi Tang

Introduction

UTCE is a platform with tools for ultrametric tree construction and tree evaluation. Phylogenetic trees can be used by biologists to describe the evolutionary relationship among of living species. Many models or tools have been proposed to construct phylogenetic trees. One of the popular models is an ultrametric tree with the assumption of a constant rate. UPGMA is one of well-known ultrametric tree building algorithms.

Although UPGMA often fails to reconstruct an evolutionary tree when a molecular clock does not apply, it is suitable for some cases of clocklike data. Moreover, the assumption of a constant rate can be used by biologists to estimate or disprove initial observations or hypotheses for their research work.

However, UPGMA is a heuristic algorithm and can not guarantee the constructed phylogenetic tree with minimum size. UTCE provides an efficient minimum ultrametric tree construction tool, **PBBU**, with a parallel branch-and-bound algorithm. The input of PBBU is a metric distance matrix or original/pre-aligned multiple DNA/protein sequences of species.

It should be noted that the constructed phylogenetic tree by PBBU could not be used to reject other phylogenetic trees by UPGMA and other tools. The goal of PBBU is to give another viewpoint for biologists to observe the evolution relationship of species under the assumption of a molecular clock hypothesis and the minimum evolution principle. In addition, in UTCE, two logical methods, 3PR and 4PR, are designed to evaluate a phylogenetic tree and/or the corresponding metric distance matrix.

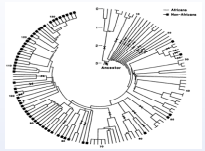
Acknowledgement

The work is partially supported by National Science Council. NSC 94-2213-E-216 -028

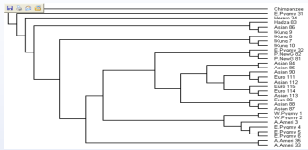
UTCE is freely available at:
http://pdcluster1.csie.chu.edu.tw/tree2

Parallel Branch-and-Bound

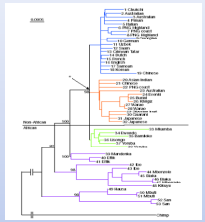
It is a parallel branch-and-bound algorithm to construct an minimum ultrametric tree from a distance matrix with the number of species and time constraints. User can specify the number of processors for computing and time constrain. The results will send by e-mail within given time.



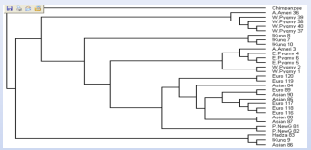
Science, 1991, mtDNA in D-loop, using PAUP* 3.0

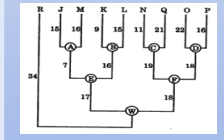


The results of PBBU. Here we can observe the Chimpanzee already separated with others.

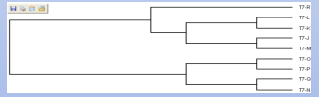


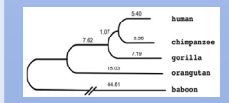
Nature, 2000, mtDNA not in D-loop, using PAUP*4.0





Science, 1992, T7 bacteriophage, NJ, UPGMA also can construct this tree.



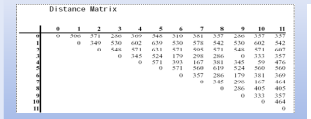


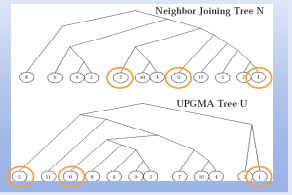
PNAS, 2006, NJ, non-coding region EnM001

We also obtain this tree by using UTCE with the same distance matrix from paper.

3-Point Relationship (3PR)

3PR is a logical method to check the relation for any triple of species (a, b, c) in distance matrix, which is preserved or not in the constructed phylogenetic trees. Moreover, it can illustrate the consistent between distance matrix and evolution for various tree construction methods.



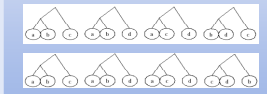


d[0,1] = 506 > d[0,7] = 381, however in the NJ tree, 0 and 1 are closer to each other than 0 and 7

4-Point Relationship (4PR)

4PR is another logical method to find contradictive species in distance matrix by checking the LCA relations in any set of 4 species.

For any set of 4 species (a, b, c, d), it has 4 triplets of species (a, b, c), (a, b, d), (a, c, d) and (b, c, d) and each triplet has its LCA relation. For any set of 4 species (a, b, c, d), no phylogenetic tree can conform to all of four LCA relations when they are ((a,b),c);, ((a,b),d);, ((a,c),d);, ((b,d),c); or ((a,b),c);, ((a,b),d);, ((a,c),d);, ((c,d),b); and this set is contradictive.



```
----- contradiction sets -----
283
W.Pygmy_1   A.American !Kung_7     European_8
W.Pygmy_1   A.American !Kung_10    European_8
W.Pygmy_1   E.Pygmy_6  !Kung_7     European_8
W.Pygmy_1   E.Pygmy_6  !Kung_10    European_8
W.Pygmy_2   W.Pygmy_1  E.Pygmy_4   A.American
W.Pygmy_2   W.Pygmy_1  E.Pygmy_5   A.American
W.Pygmy_2   W.Pygmy_1  E.Pygmy_6   A.American
W.Pygmy_2   European_8 !Kung_8     P.NewG_80
A.American  European_8 !Kung_8     P.NewG_80
.
.
.
```

The sample output contradiction set.

Sequence Alignment

A simple dynamic programming algorithm is used to compute the edit distance amount any two species (a,b) with DNA or protein sequences.

附錄二：出席國際學術會議心得報告

第一屆工程與電腦科學國際研討會

International MultiConference of Engineers and Computer Scientists 2006 (IMECS 2006)

一、前言

第一屆工程與電腦科學國際研討會(IMECS 2006)於西元 2006 年 6 月 20 日至 6 月 22 日在香港舉行。本次會議共有十四個 Workshop 一起舉行，分別為 WAIA'06、IWB'06、IWCS'06、IWDMA'06、IWEE'06、IWFE'06、IWIE'06、IWINDE'06、IWICWS'06、IWOR'06、IWSCCS'06、IWSE'06、IWWN'06 以及 ICMHA'06，研討會的規模相當大。筆者參加之 Workshop 為 IWCS'06，IWCS'06 錄取 24 篇資訊科學理論研究領域之優秀論文，分為四個場次進行討論，筆者並被邀請擔任”IWCS Session I”之 Session Chair。工程與電腦科學國際研討會雖然是第一屆舉辦，但是研討議題之廣泛與參加之專家學者人數之多，比其他具有舉辦歷史之國際研討會更具規模，台灣大約有二十來位之學者參加此盛會。

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，本次會議於第一天的會議議程中安排了二個場次之專題報告，正式之論文報告分別於三天三十二個場次的專題報告後進行，筆者之論文『An Efficient Scheduling Algorithm for Irregular Data Redistribution 』被安排在第一天的” IWCS Session I”之場次發表，筆者並且擔任此場次之會議主席，同時筆者之論文並獲得優秀論文獎(The Certificate of Merit)。

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資訊理論與技術的論文，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. IMECS 2006 研討會論文集
3. 優秀論文獎(The Certificate of Merit)獎狀

附錄二：出席國際學術會議心得報告

第一屆工程與電腦科學國際研討會

International MultiConference of Engineers and Computer Scientists 2006 (IMECS 2006)

一、前言

第一屆工程與電腦科學國際研討會(IMECS 2006)於西元 2006 年 6 月 20 日至 6 月 22 日在香港舉行。本次會議共有十四個 Workshop 一起舉行，分別為 WAIA'06、IWB'06、IWCS'06、IWDMA'06、IWEE'06、IWFE'06、IWIE'06、IWINDE'06、IWICWS'06、IWOR'06、IWSCCS'06、IWSE'06、IWWN'06 以及 ICMHA'06，研討會的規模相當大。筆者參加之 Workshop 為 IWCS'06，IWCS'06 錄取 24 篇資訊科學理論研究領域之優秀論文，分為四個場次進行討論，筆者並被邀請擔任”IWCS Session I”之 Session Chair。工程與電腦科學國際研討會雖然是第一屆舉辦，但是研討議題之廣泛與參加之專家學者人數之多，比其他具有舉辦歷史之國際研討會更具規模，台灣大約有二十來位之學者參加此盛會。

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，本次會議於第一天的會議議程中安排了二個場次之專題報告，正式之論文報告分別於三天三十二個場次的專題報告後進行，筆者之論文『An Efficient Scheduling Algorithm for Irregular Data Redistribution 』被安排在第一天的” IWCS Session I”之場次發表，筆者並且擔任此場次之會議主席，同時筆者之論文並獲得優秀論文獎(The Certificate of Merit)。

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資訊理論與技術的論文，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. IMECS 2006 研討會論文集
3. 優秀論文獎(The Certificate of Merit)獎狀

An Efficient Parallel Algorithm for Ultrametric Tree Construction Based on 3PR*

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

[1] Department of of Computer Science and Information Engineering, Chung Hua University
[2] Institute of Engineering Science, Chung Hua University
[3] Institute of Molecular and Cellular Biology, National Tsing Hua University
[4] Department of Computer Science, National Tsing Hua University
300, Hsinchu, Taiwan, R.O.C
[1] yu@chu.edu.tw, [2] jyzhou@pdlab.csie.chu.edu.tw,
[3] cyulin@mx.nthu.edu.tw, [4] cytang@cs.nthu.edu.tw

Abstract. In the computational biology and taxonomy, to construct phylogenetic tree is an important problem. A phylogenetic tree can represent the relationship and histories for a set of species and helpful for biologists to observe existent species. One of popular model is ultrametric tree, and it assumed the evolution rate is constant. UPGMA is one of well-known ultrametric tree algorithm. However, UPGMA is a heuristic algorithm, and it can not guarantee the constructed tree is minimum size. To construct minimum ultrametric tree (MUT) has been shown to be an NP-hard problem. In this paper, we propose an efficient parallel branch-and-bound algorithm with 3-Point Relationship (3PR) to reduce the construction time dramatically. 3PR is a relationship between a distance matrix and the constructed phylogenetic tree. The main concept is for any two species closed to each other in a distance matrix should be also closed to each other in the constructed phylogenetic tree. We use this property to mark the branching path with lower priority or higher, then we move the lower ranked branching path to delay bound pool instead of remove it to ensure the optimal solution can be found. The experimental results show that our parallel algorithm can save the computing time and it also shows that parallel algorithm with 3PR can save about 25% of computing time in average.

Keywords: phylogenetic tree, minimum ultrametric tree, parallel branch-and-bound algorithm, 3-point relationship, 4-point relationship.

1 Introduction

To construct phylogenetic trees is an important problem in the computational biology and in taxonomy, the phylogenetic tree can represent the histories for a set of species and helpful for biologists to observe existent species or evaluate the relationship of them. However, the real evolutionary histories are unknown in practice. Therefore, many methods had been proposed and tried to construct a meaningful phylogenetic tree, which is closing to the real one.

* The work is partially supported by National Science Council. (NSC 94-2213-E-216 -028).
** The corresponding author.

G. Min et al. (Eds.): ISPA 2006 Ws, LNCS 4331, pp. 215 – 220, 2006.

In the input of distance matrix, a phylogenetic tree is constructed according to the distance matrix [10,11]. In general, these values are edit distances between two sequences of any two species. There are many different models and motivated algorithmic problems were proposed [1,9]. However, most of optimization problems for phylogenetic tree construction have been show to be NP-hard [2-4,6,7]. An important and commonly used model is assumed that the rate of evolution is constant. Based on this assumption, the phylogenetic tree will be an ultrametric tree (*UT*), which is rooted, leaf labeled, and edge weighted binary tree. Because many of these problems are intractable and NP-hard, biologists usually construct the trees by using heuristic algorithm. The Unweighted Pair Group Method with Arithmetic mean (*UPGMA*, [1]) is one of the popular heuristic algorithms to construct *UTs*.

Although construct *MUTs* is an NP-hard problem, it is still worthy to construct for middle-size of species. Thus, it seems possible to find an optimal tree using exhaustive search. Nevertheless, for n species, the number of rooted and leaf label tree is, it grows very rapidly. For example, $A(10) > 10^7$, $A(20) > 10^{21}$, $A(30) > 10^{37}$. Hence, it is impossible to exhaustively search for all possible trees even n are middle-size. Wu *et al*. [13] proposed a branch-and-bound algorithm for constructing *MUTs* to avoid exhaustive search. The branch-and-bound strategy is a general technique to solve combinatorial search problems.

In this paper, 3-Point Relationship (3PR) is used to construct *MUTs* more efficiently. 3PR is the relationship between a distance matrix and the constructed phylogenetic tree. The concept is that in triplet of species (a, b, c), any of two species which is closed to each other in the distance matrix should aslo be closed to each other in the constructed phylogenetic tree in a distance matrix. The experimental results show that *PBBU* with 3PR can reduce about 25% computation time both in sequential and parallel algorithms.

The paper is organized as follows. In section 2, some preliminaries for sequential branch-and-bound algorithm and 3PR are given. Parallel algorithm is described in section 3. Section 4 shows our experimental results, and final section is our conclusions.

2 Preliminaries

In this paper, we present *PBBU* with 3PR for construct minimum ultrametric tree. In the following, we denote an unweighted graph G=(V,E,w) with a vertex set V, an edge set E, and an edge weight function w. Some definitions are given as follows:

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

***Definition* 2:** Let $T = (V, E, w)$ 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 $w(T) = \sum_{e \in E} w(e)$.

***Definition* 3:** For any M (not necessarily a metric), *MUT* for M is T with minimum $w(T)$ 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.

***Definition* 4:** 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 $x \to y$ if and only if x is an ancestor of y.

***Definition* 5:** The distance between distance matrix and rooted topology of phylogenetic 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 $1 \le i, j, k \le n$. Otherwise is contradictory.

2.1 Sequential Branch-and-Bound Algorithm for MUTs

In the *MUT* construction problem, the branch-and-bound is a tree search algorithm and repeatedly searches the branch-and-bound tree (*BBT*) [8,14] to find a better solution until optimal one is found. The *BBT* is a tree which can represent a topology of *UTs*. Assume that the root of *BBT* has depth 0, hence each node with depth i in BBT represents a topology with a leaf set $\{1,...,i+2\}$.

2.2 3-Point Relationship (3PR)

3PR is a logical method to check the *LCA* relation for any triplet of species (a, b, c) in a distance matrix, which is preserved or not in the constructed phylogenetic trees. For any two species (a, b), $LCA(a, b)$ denotes the least common ancestor of (a, b). If (x, y) are two nodes in a phylogenetic tree, $x \to y$ is written if x is an ancestor of y. For a triplet of species (a, b, c) in the distance matrix M, if the distance $M[a, b]$ of species a and b is less than $M[a, c]$ and $M[b, c]$, $LCA(a, c)=LCA(b, c) \to LCA(a, b)$ (as ((a, b), c); in Newick tree format). For a triplet of species (a, b, c), it is contradictive if the least common ancestor relation in a distance matrix is not preserved in the constructed phylogenetic tree. *3PR* can be used to evaluate the qualities of constructed phylogenetic trees. A phylogenetic tree is considered unreliable if the number of contradictive triplets is large. The evaluated result may be useful for biologists to choose a feasible phylogenetic tree construction tool.

3 Parallel Branch-and-Bound Algorithm with 3PR

Parallel Branch-and-Bound Algorithm with *3PR* (*PBBU* with *3PR*) is designed on distributed memory multiprocessors and the master-slave architecture. The *PBBU* uses a branch-and-bound technique to avoid exhaustive search of possible trees. For load-balance purpose, the master processor (*MP*) contains a *Global Pool* and each slave processor (SP) has Local Pool, moreover we use new data structure instead of the link list to store *BBT*.

In [5], *3PR* is applied as a tree evaluation method. We use this property to put lower rank branching path to Delay Bound Pool (*DBP*) when selecting branch path in the branch-and-bound algorithm. For example, Table 1 is the distance matrix and Figure 1 shows two candidates when inserting the third species c. In *PBBU* without *3PR*, both (a) and (b) candidates need to be added to the pool when branching. However, topology of (b) is closing to distance matrix, it obtained higher rank, and (a) has lower rank. In *PBBU* with *3PR*, only (b) (with higher rank) candidate will be

selected due to the distance of *a* and *c* is greater than the distance of *b* and *c*. This result is based on the conception that in a triplet of species (*a*, *b*, *c*), any of two species which is closed to each other in the distance matrix should also be closed to each other in the corresponding phylogenetic tree in a distance matrix. However, it cannot be directly used to bound another branching path, and *PBBU* with *3PR* put others candidates to the *DBP* to ensure the optimal solution can be found.

Table 1. Distance matrix

	a	*b*	*c*
a	0	25	20
b	25	0	15
c	20	15	0

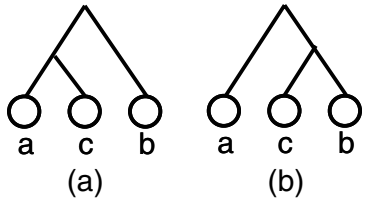


Fig. 1. Candidate *BBT*

4 Experimental Results

In the experimental results, we implement *PBBU* and *PBBU* with *3PR* on a Linux based PC cluster. Each computing node is an AMD Athlon PC with a clock rate of 2.0 GHz and 1GB memory. Each node is connected with each other by 100Mbps network. There are two data sets used to test our algorithms. One is a random data set, which is generated randomly. The distance matrix in the random data set is metric and the range of distances is between 1 and 100. Another is a data set composed of 136 Human Mitochondrial DNAs (*HMDNA*), which is obtained from [12]. Its distance matrix is metric and the range of distances is between 1 and 200. In order to eliminate the problems of data dependence, for each testing data, we run 10 instances. Then we compare the average, median, and worst cases.

Figure 2 and 3 show that *PBBU* with *3PR* and delay bound technique can find the optimal solution and save about 25% of computation time than *PBBU* without *3PR*. Because *3PR* technique move lower ranking candidates which disaccording to *3PR* to delay bound pool, after that, the better bounding value can be found early. Afterward it can bound more candidates to decreasing computation time.

Figure 4 is the speed-up ratio of *HMDNA* data set. We observed that the speed-up ratio of *3PR* is better than it without *3PR*. Furthermore, the difference between *3PR* and without *3PR* is larger when the number of processors increasing. Because of the tighter bounding value can be found quickly with more processors. It also shows that our algorithm is scalable in large number of computing resources. Figure 5 shows the computation time of 16 processors of *PBBU* with *3PR* for different number of species. We can observe that the computation time grow rapidly when the number of species increasing. Moreover, the reduced proportion between *PBBU* and *PBBU* with *3PR* is increasing with larger number of species. We consider that large number of species contains more candidates that a tighter bounding value which can be obtained from *3PR* technique can also bound grater number of candidates; it can decreasing the computation time.

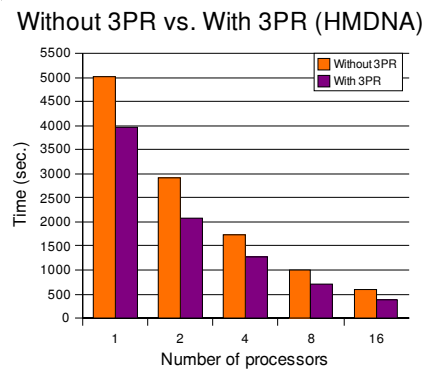


Fig. 2. 3PR vs. Without 3PR (HMDNA)

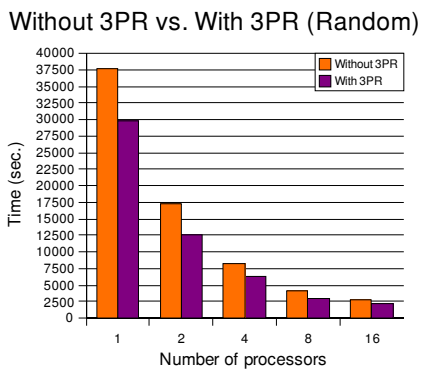


Fig. 3. 3PR vs. Without 3PR (Random)

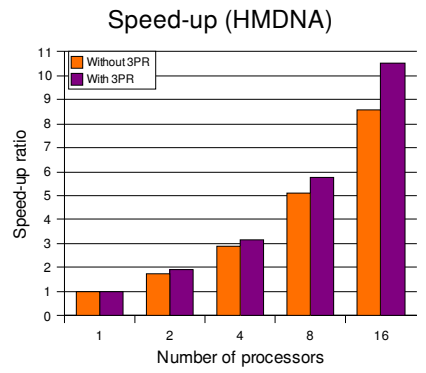


Fig. 4. Speed-up ratio (HMDNA)

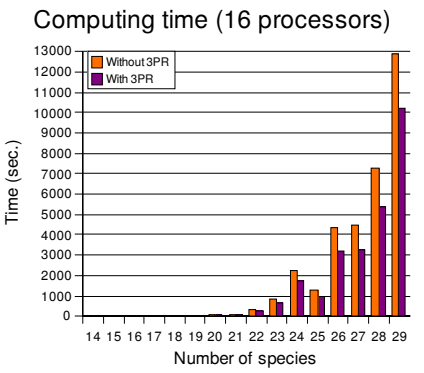


Fig. 5. Computing time (16 processors)

5 Conclusions

In this paper, we have designed *PBBU* with *3PR* for constructing *MUTs* problem. The *3PR* is the relationship between distance matrix and constructed evolutionary tree. It moves candidates which do not fit *3PR* to delay bound pool in branch-and-bound algorithm. After that, we can obtain the tighter bounding value quickly and uses it to bound more candidates. In order to evaluate the performance of our proposed algorithm, a random data set and a practical data set of *HMDNA* are used. The experimental results show that *PBBU* with *3PR* can find optimal solution for 36 species within a reasonable time on 16 PCs. Furthermore, the speed-up ratio shows the performance of our algorithm is good in our PC cluster environment. Moreover, the results also show that *PBBU* with *3PR* can save about 25% in average of computing time than *PBBU* without *3PR*, and it assured the results are optimal with the delay bound technique.

References

1. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein “Introduction to Algorithm,” MIT Press, 1990.
2. W.H.E. Day “Computationally difficult parsimony problems in phylogenetic systematics,” J. Theoretic Biol., 103, 1983, pp.429-438.
3. W.H.E. Day “Computational complexity of inferring phylogenies from dissimilarity matrices,” Bulletin of Math. Biol., 49, 1987, pp.461-467.
4. W.H.E. Day, D.S. Johnson, and D. Sankoff “The computational complexity of inferring rooted phylogenies by parsimony,” Math. Biosci., 81, 1986, pp.33-42.
5. C.T. Fan, “The evaluation model of evolutionary tree,” Master Thesis, Nationa Tsing Hua University, 2000.
6. L.R. Foulds “Maximum savings in the Steiner problemin phylogeny,” J. Theoretic Biol., 107, 1984, pp.471-474.
7. D. Gusfield “Algorithms on Strings, Trees, and Sequences, computer science and computational biology,” Cambridge University Press, 1997.
8. M.D. Hendy and D. Penny “Branch and bound algorithms to determine minimal evolutionary trees,” Math. Biol., 59, 1982, pp.277-290.
9. S. Kumer, K. Tamura, M. Nei “MEGA: Molecular Evolutionary Genetics Analysis software for miceocomputers,” Comput. Appl. Biosci., 10, 1994, pp.189-191.
10. W.H. Li “Molecular Evolution,” Sinauer Associates, 1997.
11. R.D.M. Page “TreeView: An application to display phylogenetic trees on personal computers,” Comput. Appl. Biosci., 12, 1996, pp.357-358.
12. L. Vigilant, M. Stoneking, H. Harpending, K. Hawkes and A.C. Wilson “African Populations and the Evolution of Human Mitochondrial DNA,” Science, 253, 1991, pp.1503-1507.
13. B.Y. Wu, K.M. Chao, and C.Y. Tang “Approximation and Exact Algorithms for Constructing Minimum Ultrametric Trees from Distance Matrices,” J. Combinatorial Optimization, 3, 1999, pp.199-211.
14. C.F. Yu and B.W Wah “Efficient Branch-and-Bound Algorithms on a Two-Level Memory System,” IEEE Trans. Parallel and Distributed Systems, 14, 1988, pp.1342-1356.

An Efficient Scheduling Algorithm for Irregular Data Redistribution

Kun-Ming Yu and Yi-Lin Tsai

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

Abstract. Dynamic data redistribution is used to enhance the performance of an algorithm and to achieve data locality in parallel programs on distributed memory multi-computers. Therefore, the data redistribution problem has been extensively studied. Previous results focus on reducing index computational cost, schedule computational cost, and message packing/unpacking cost. However, irregular data redistribution is more flexible than regular data redistribution; it can distribute different sizes of data segments of each processor to those processors according to their own computation capability. High Performance Fortran 2 (HPF-2), the current version of HPF, provides an irregular distribution functionality, such as GEN_BLOCK which addresses some requirements of irregular applications for the distribution of data in an irregular manner and explicit control of load balancing. In this paper, we present a degree-reduction-and-coloring (DRC) algorithm for scheduling HPF2 irregular array redistribution. We devoted to obtain the minimal number of transmission steps as well as to reduce the overall redistribution time. The proposed algorithm intends to reduce the number of maximum transmission messages in the first phase and then applies graph-coloring mechanism to obtain the final schedule. The proposed method not only avoids node contention, but also shortens the overall redistribution time. To evaluate the performance of DRC algorithm, we have implemented DRC algorithms along with the Divide-and-Conquer algorithm. The simulation results show that DRC algorithm has significant improvement on communication costs compared with the Divide-and-Conquer algorithm.

Keywords: Irregular redistribution, communication scheduling, GEN_BLOCK, degree-reduction

1. Introduction

Parallel computing systems have been widely adopted to solve complex scientific and engineering problems. To efficiently execute a data-parallel program on distributed memory multi-computers, an appropriate data distribution is critical to the performance. Appropriate distribution can balance the computational load, increase data locality, and reduce inter-processor communication. Array redistribution is crucial for system performance because a specific array distribution may be appropriate for the current phase, but incompatible for the subsequent one. Many parallel programming languages thus support run-time primitives for rearranging the array distribution of a program. The data redistribution problem has been widely studied in the literature. In general, data redistribution can be classified into two categories: the regular data redistribution [1,5,6,7,9,11,13,15,18] and the irregular data redistribution [4,8,22-24]. The regular data redistribution decomposes data of equal sizes into processors. There are three types of this data redistribution, called BLOCK, CYCLIC, and BLOCK-CYCLIC(n). The irregular data distribution employs user-defined functions to specify data distribution unevenly. High Performance FORTRAN 2 (HPF-2) provides GEN_BLOCK functionality and makes it possible to handle different processors dealing with appropriate data size according to their computation capability. Previous works emphasized the minimal steps of data redistribution and scheduled the ordering of messages with minimal total transmission size. In the regular array redistribution, [15] proposed an Optimal Processor Mapping (OPM) scheme to minimize the data transmission cost for general BLOCK-CYCLIC regular data realignment. Optimal Processor Mapping (OPM) utilized the maximum matching of realignment logical processors to achieve the maximum data hits for reducing the amount of data exchange transmission cost. In the irregular array redistribution problem, [22, 23] proposed a greedy algorithm to utilize the Divide-and-Conquer technique to obtain near optimal scheduling while attempting to minimize the size of total communication messages as well as the number of steps.

In this paper, we bring up the Degree-Reduction-and-Coloring (DRC) algorithm to efficiently perform GEN_BLOCK array redistribution. In section 2, we define the data communication model of irregular data redistribution and give an example of GEN_BLOCK data redistribution as the preliminary. Section 3 describes the DRC algorithm for the irregular redistribution problem. The performance analysis, simulation results and practical transmission with MPI on SMP/Linux cluster are presented in section 4. Finally, the conclusions are given in section 5.

2. Data communication models

In this section, we present some properties of irregular data redistribution with GEN_BLOCK functionality. There are no repetitive communication

patterns in the irregular GEN_BLOCK array redistribution. A data redistribution can be represented by a bipartite graph, called a *redistribution graph*. To simplify the presentation, notations and terminologies used in this paper are defined in the following.

***Definition* 1**: Given an irregular GEN_BLOCK redistribution on array $A[SPi]$ and $A[DPi]$ over P processors, the *source processors* of array data elements $A[SPi]$ are denoted as SPi; the *destination processors* of array elements $A[DPi]$ are denoted as DPi where $1 \le i \le P$.

***Definition* 2**: A bipartite graph $G = (V, E)$ is used to represent the communications of an irregular data redistribution between source and destination processors. Vertices of G are used to represent the processors. Edge e_{ij} in G denotes the message sent from SPi to DPj, where $e_{ij} \in E$. $|E_{ij}|$ denotes the transmission message size through the redistribution.

***Definition* 3**: Every message transmission link in irregular data redistribution is not overlapped. Hence, the total number of message transmission link E is $P \le E \le 2 \times P - 1$.

***Definition* 4**: Each processor has more than one e_{ij} to send data to destination processors or receive data from other source processors. The number D of e_{ij} owned by one processor is denoted as *D-degree* and the maximum *D-degree* of all processors is denoted as *Max-degree*. We denote that the processors have the *Max-degree* number of messages as P_{max}.

***Definition* 5**: If SPi sends messages to $DPj-1$ and $DPj+1$, the transmission between SPi and DPj must exist, where $1 \le i, j \le P$. This result was mentioned as the consecutive communication property [12].

Fig.1 shows an example of redistributing two GEN_BLOCK distributions on $A[SPi]$ and $A[DPi]$. Table 1(a) shows mapped communication message size to source processors and destination processors, respectively. The communications between source and destination processor sets are depicted in Fig 1. There are 13 transmission messages, e_{11}, e_{21}, e_{22}, …e_{77} among the processors involved in the redistribution. Due to the considerable influence of node contention, a processor can only send at most one message to another processor in each communication step and the same is true for the receiving message. The messages, which cannot be scheduled in the same communication step, are called conflict tuple. For instance, $\{e_{11}, e_{21}\}$ is a conflict tuple since they have a common destination processor $DP1$; $\{e_{21}, e_{22}\}$ is also a conflict tuple because of the common source processor $SP2$. Table 1(b) shows a simple schedule result for this example.

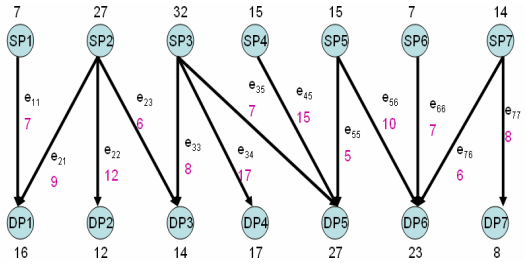


Figure 1. An example of data redistribution

Table 1(a). The total message size of redistribution data for each processor in Fig. 1.

SP1	SP2	SP3	SP4	SP5	SP6	SP7
7	27	32	15	15	7	14
DP1	DP2	DP3	DP4	DP5	DP6	DP7
16	12	14	17	27	23	8

Table 1(b). A simple schedule

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	e_{56}, e_{23}, e_{35}
Step3:	e_{21}, e_{33}, e_{55}, e_{76}

3. Proposed Algorithm

The performance of a data redistribution procedure is determined by four costs: index computational cost T_i, schedule computational cost T_s, message packing/unpacking cost T_p, and data transfer cost. The data transfer cost for each communication step consists of start-up cost T_{su} and transmission cost T_t. Let the unit transmission time τ denote the cost of transferring a message of unit length. In general, the message startup cost is directly proportional to the number of communication steps. The total number of communication steps is denoted by N. The total redistribution time equals $T_i+T_s+\sum_{i=1}^{N}(T_p + T_{su} + m_i\tau)$, where $m_i = \text{Max}\{e_1, e_2, e_3, .., e_{k-1}\}$ and e_j represents the size of message scheduled in the i^{th} communication step for $j = 1$ to k. In irregular redistribution, messages of varying sizes are scheduled in the same communication step. Therefore, the largest size of message in the same communication step dominates the data transfer time required for this communication step.

The main idea of the Degree-Reduction-and-Coloring (DRC) algorithm is to diminish the degree of P_{max} repeatedly by scheduling the message in the first step of data redistribution process until *Max-degree* is equal to 2. The remaining messages are then scheduled into the communication steps by utilizing the concept of bipartite graph coloring mechanism. The details of the steps will be described in the following subsections.

3.1 Degree-Reduction Step

The goal in this step is to reduce *Max-degree* repeatedly in each iteration, until *Max-degree* is equal to 2. An example of 4-degree communication redistribution has taken as shown in Fig 2. In the first phase (phase-1) of degree-reduction step, the messages are sorted by the non-increasing order of $|E_{ij}|$, and the result is shown in Table 2. Then, DRC selects the messages into step1 of the schedule according to non-increasing order of message size except those messages causing the conflict. After phase-1, the *Max-degree* will be decreased by 1 (from 4 to 3). Fig 3(a) and Table 3(b) show this scenario. DRC repeat the procedure until the *Max-degree* reaches 2, which is depicted in Fig 4.

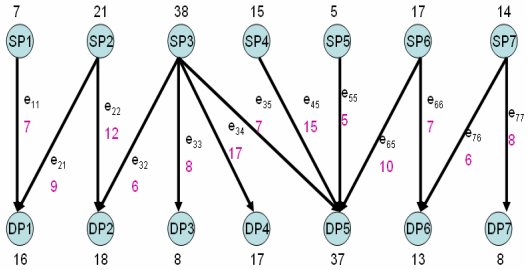
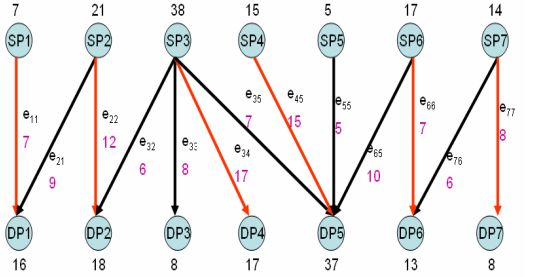


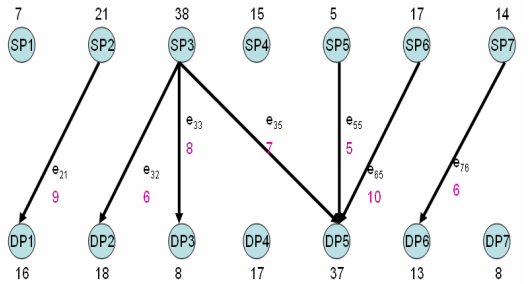
Figure 2. A data redistribution example with *Max-degree* = 4

Table 2. The messages are sorted by non-increasing order of message size

Msg no.	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Msg size	17	15	12	10	9	8	8	7	7	7	6	6	5



(a)



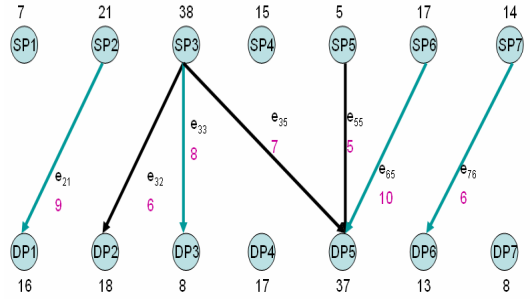
(b)

Figure 3. The messages communication (a) before phase-1 of the degree-reduction step; (b) after phase-1 of the degree-reduction step.

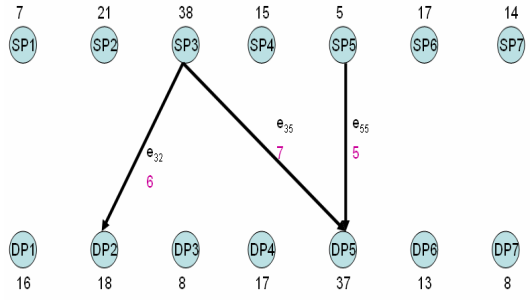
Table 4. The schedule after phase-1

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	
Step3:	
Step4:	

Message number	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Message size	17	15	12	10	9	8	8	7	7	7	6	6	5



(a)



(b)

Figure 4. The messages communication (a) before phase-2 of the degree-reduction step; (b) after phase-2 of the degree-reduction step.

Table 5. The schedule after the procedure of phase-2

Message no.	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Message size	17	15	12	10	9	8	8	7	7	7	6	6	5

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	e_{65}, e_{21}, e_{33}, e_{76}
Step3:	
Step4:	

3.2 Message-Coloring Step

After completing the degree-reduction step, we can obtain a redistribution graph with *Max-degree* of 2 and the resulting redistribution graph is *2-edge colorable* [2], since it is a bipartite graph and its maximum degree is equal to 2. In the Message-Coloring Step, DRC schedules the left messages into the same step in a non-increasing order to accomplish an optimal scheduling unless a conflict occurs. Figure 5 shows the outcome of message-coloring and Table 6 shows the final schedule obtained from DRC algorithm.

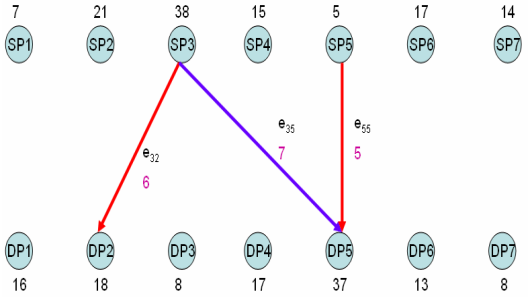


Figure 5. The outcome of redistribution graph after applying the message coloring mechanism

Table 6. The final schedule obtained from DRC

Msg no.	e_{34}	e_{45}	e_{22}	e_{65}	e_{21}	e_{33}	e_{77}	e_{11}	e_{35}	e_{66}	e_{32}	e_{76}	e_{55}
Msg size	17	15	12	10	9	8	8	7	7	7	6	6	5

Schedule Table	
Step1:	e_{34}, e_{45}, e_{22}, e_{77}, e_{11}, e_{66}
Step2:	e_{65}, e_{21}, e_{33}, e_{76}
Step3:	e_{35}
Step4:	e_{32}, e_{55}

The algorithm of the Degree-Reduction-Coloring is given as follows.

==========

Algorithm DRC
generating messages;
// generate messages from *AS*[*Pi*] to *AD*[*Pi*]
step = **maximum degree;**
sort_msgSize();
// sorting in decreasing order by message size
while (*step* **> 2)**
{
choose_msg(*step***);**
// selecting message without conflict tuple, set into *(maximal degree - step + 1)* schedule step
step--
} // degree-reduction iteration
while (remaining_messages != null)
{
selecting_msg(*maximal degree*-1**);**
// selecting message set into *maximal degree-1* schedule step
check_msg_continue_set();
// check remaining message set
coloring_maximal_msg(*maximal degree***);**
// color the maximal message with degree *maximal degree -1* and the neighbor message with *maximal degree*
} // message coloring mechanism
end of DRCM

==========

4. Performance Evaluation

To evaluate the performance of the proposed methods, we have implemented the DRC along with the Divide-and-Conquer algorithm [23]. The performance simulation is discussed in two categories, even GEN_BLOCK and uneven GEN_BLOCK distributions. In even GEN_BLOCK distribution, each processor owns similar size of data. In contrast to even distribution, few processors might be allocated by grand volumes of data with uneven distribution. Since data elements could be centralized to some specific processors, it is also possible for those processors to have the maximum degree of communications.

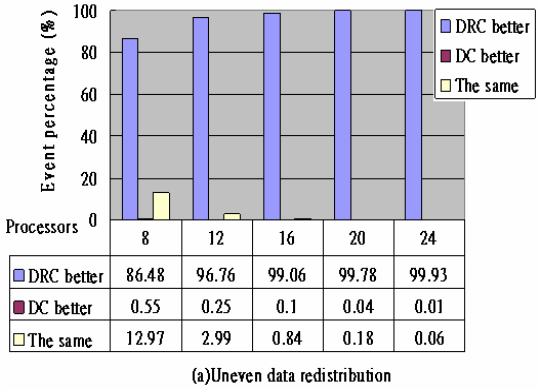
The simulation program generates a set of random integer number and the size of message as *A*[*SPi*] and *A*[*DPi*]. Moreover, the total message size sending from SPi equals to the total size receiving to DPi keeping the balance between source processors and destination processors.

We assume that the data computation (communication) time in the simulation is represented by the transmission size $|E_{ij}|$. In the following figures, the percentage of events is plotted as a function of the message size and the number of processors. Also, in the figures, “DRC Better” represents the percentage of the number of events that the DRC algorithm has lower total computation (communication) time than the Divide-and-Conquer algorithm, while “DC Better” gives the reverse situation. If both algorithms have the same total computation (communication) time, “The Same Results” represents the number of that event.

In the uneven distribution, the size of message’s up-bound is set to be B*1.7 and that of low-bound is set to be B*0.3, where B is equal to the sum of total transmission message size / total number of processors. In the even distribution, the size of message’s up-bound is set to be B*1.3 and that of low-bound is set to be B*0.7. The total message-size is 10M.

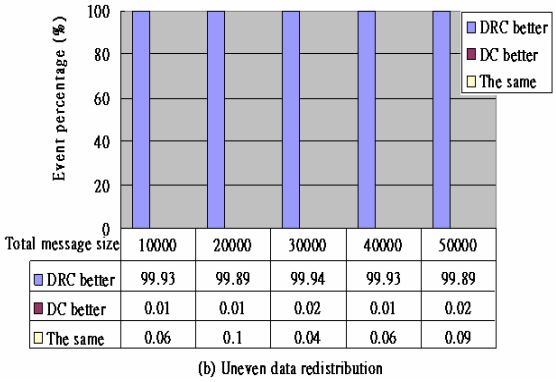
Fig 6(a) and 6(b) show the simulation results of both the DRC and the Divide-and-Conquer algorithm

with different number of processors and total message size. The number of processors is from 8 to 24. We can observe that the DRC algorithm has better performance in the uneven data redistribution compared with Divide-and-Conquer algorithm. Since the data is concentrated in the even case, from Fig 7(a) and 7(b), we can observe that DRC has better performance compared with the uneven case. In both even and uneven cases, DRC performs better than the Divide-and-Conquer algorithm.



Processors	8	12	16	20	24
DRC better	86.48	96.76	99.06	99.78	99.93
DC better	0.55	0.25	0.1	0.04	0.01
The same	12.97	2.99	0.84	0.18	0.06

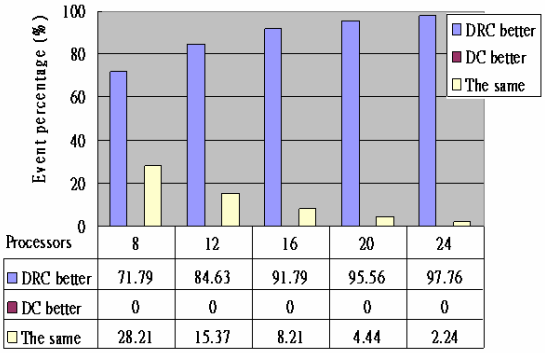
(a)Uneven data redistribution



Total message size	10000	20000	30000	40000	50000
DRC better	99.93	99.89	99.94	99.93	99.89
DC better	0.01	0.01	0.02	0.01	0.02
The same	0.06	0.1	0.04	0.06	0.09

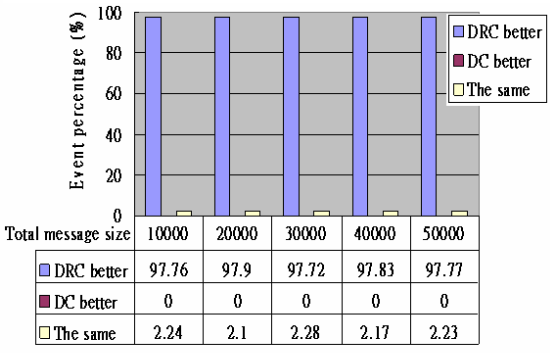
(b) Uneven data redistribution

Figure 6. The events percentage of computing time is plotted (a) with different number of processors and (b) with different number of total message sizes in 24 processors, on the uneven data set.



Processors	8	12	16	20	24
DRC better	71.79	84.63	91.79	95.56	97.76
DC better	0	0	0	0	0
The same	28.21	15.37	8.21	4.44	2.24

(a) Even data redistribution



Total message size	10000	20000	30000	40000	50000
DRC better	97.76	97.9	97.72	97.83	97.77
DC better	0	0	0	0	0
The same	2.24	2.1	2.28	2.17	2.23

(b) Even date redistribution

Figure 7. The events percentage of computing time is plotted (a) with different number of processors and (b) with different number of total message sizes in 24 processors, on the even data set.

5.Conclusion

In this paper, we have presented a Degree-Reduction-Coloring (DRC) scheduling algorithm to efficiently perform HPF2 irregular array redistribution on a distributed memory multi-computer. The DRC algorithm is a simple method with low algorithmic complexity to perform GEN_BLOCK array redistribution. The DRC algorithm is an optimal algorithm in terms of minimal number of steps. In the same time, DRC algorithm is also a near optimal algorithm satisfying 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 DRC as well as the Divide-and-Conquer redistribution algorithm. The experimental results show improvement in communication costs and high practicability on different processor hierarchies. Also, the experimental results indicate that both of them have good performance on GEN_BLOCK redistribution. In many situations, DRC is better than the Divide-and-Conquer redistribution algorithm.

Reference

[1] G. Bandera and E.L. Zapata, “Sparse Matrix Block-Cyclic Redistribution,” Proceeding of IEEE Int'l. Parallel Processing Symposium (IPPS'99), San Juan, Puerto Rico, 355 - 359 ,April 1999

[2] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, Macmillan, London, 1976.

[3] Frederic Desprez, Jack Dongarra and Antoine Petitet, “Scheduling Block-Cyclic Data redistribution,” IEEE Trans. on PDS, vol. 9, no. 2, pp. 192-205, Feb. 1998.

[4] Minyi Guo, “Communication Generation for Irregular Codes,” The Journal of Supercomputing, vol. 25, no. 3, pp. 199-214, 2003.

[5] Minyi Guo and I. Nakata, “A Framework for Efficient Array Redistribution on Distributed

Memory Multicomputers," The Journal of Supercomputing, vol. 20, no. 3, pp. 243-265, 2001.

[6] Minyi Guo, I. Nakata and Y. Yamashita, "Contention-Free Communication Scheduling for Array Redistribution," Parallel Computing, vol. 26, no.8, pp. 1325-1343, 2000.

[7] Minyi Guo, I. Nakata and Y. Yamashita, "An Efficient Data Distribution Technique for Distributed Memory Parallel Computers," Joint Symp. on Parallel Processing (JSPP'97), pp.189-196, 1997.

[8] 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.

[9] Edgar T. Kalns, and Lionel M. Ni, "Processor Mapping Technique Toward Efficient Data Redistribution," IEEE Trans. on PDS, vol. 6, no. 12, pp. 1234-1247, December 1995.

[10] S. D. Kaushik, C. H. Huang, J. Ramanujam and P. Sadayappan, "Multiphase data redistribution: Modeling and evaluation," International Parallel Processing Symposium (IPPS'95), pp. 441-445, 1995.

[11] Peizong Lee, Academia Sinica, and Zvi Meir Kedem, "Automatic Data and Computation Decomposition on Distributed Memory Parallel Computers," ACM Transactions on Programming Languages and systems, Vol 24, No. 1, pp. 1-50, January 2002.

[12] 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.

[13] Y. W. Lim, Prashanth B. Bhat and Viktor and K. Prasanna, "Efficient Algorithms for Block-Cyclic Redistribution of Arrays," Algorithmica, vol. 24, no. 3-4, pp. 298-330, 1999.

[14] C.-H Hsu, S.-W Bai, Y.-C Chung and C.-S Yang, "A Generalized Basic-Cycle Calculation Method for Efficient Array Redistribution," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 12, pp. 1201-1216, Dec. 2000.

[15] Ching-Hsien Hsu, Kun-Ming Yu, "An Optimal Processor Replacement Scheme for Efficient Communication of Runtime Data Realignment," Parallel and Distributed and Processing and Applications, - Lecture Notes in Computer Science, Vol. 3358, pp. 268-273, 2004.

[16] 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.

[17] Antoine P. Petitet and Jack J. Dongarra, "Algorithmic Redistribution Methods for Block-Cyclic Decompositions," IEEE Transactions on Parallel and Distributed Systems, vol. 10, no. 12, pp. 1201-1216, Dec. 1999

[18] Neungsoo Park, Viktor K. Prasanna and Cauligi S. Raghavendra, "Efficient Algorithms for Block-Cyclic Data redistribution Between Processor Sets," IEEE Transactions on Parallel and Distributed Systems, vol. 10, No. 12, pp.1217-1240, Dec. 1999.

[19] .L. Prylli and B. Touranchean, "Fast runtime block cyclic data redistribution on multiprocessors," Journal of Parallel and Distributed Computing, vol. 45, pp. 63-72, Aug. 1997.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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。

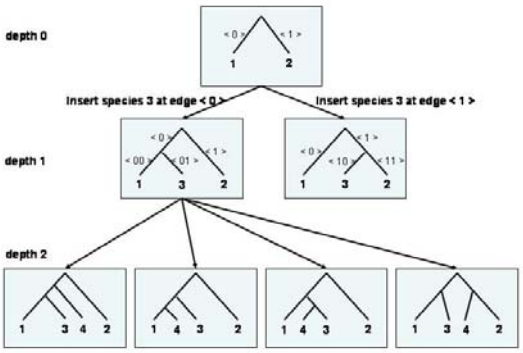


圖1. 建立分支界限樹 (BBT) [3]

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]。

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

表 1. GlobusToolkit 所提供的服務

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。

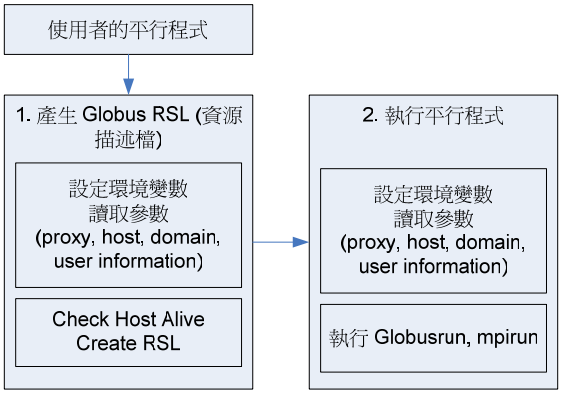


圖 2. UniGrid 上的程式流程

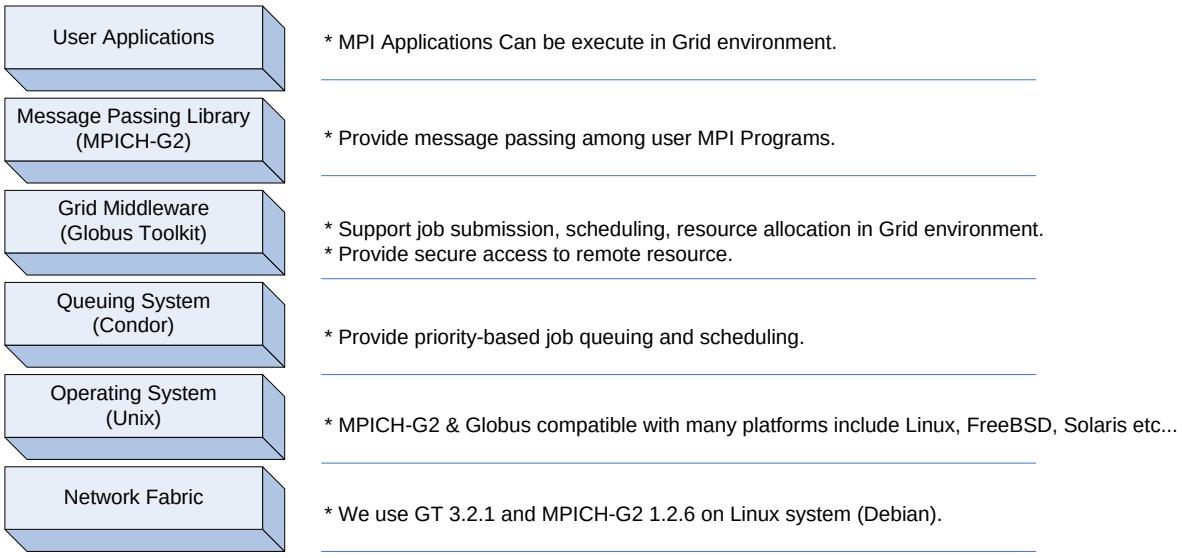


圖 3 .UniGrid 的架構圖[2]

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送的程式觀念。

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

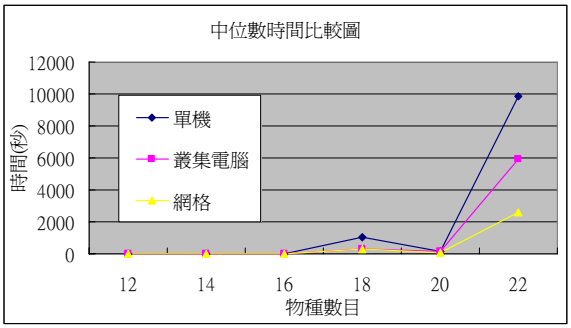


圖 4. 中位數時間比較圖

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

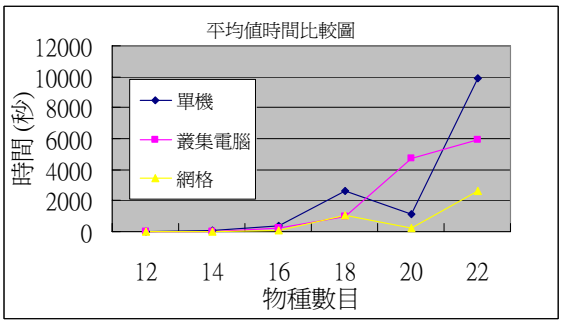


圖 5. 平均數時間比較圖

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

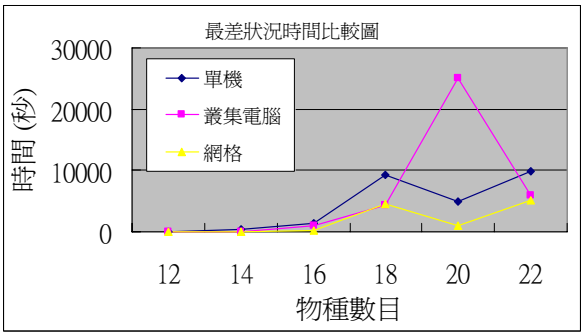


圖 6. 最差狀況時間比較圖

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 組物種數目 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

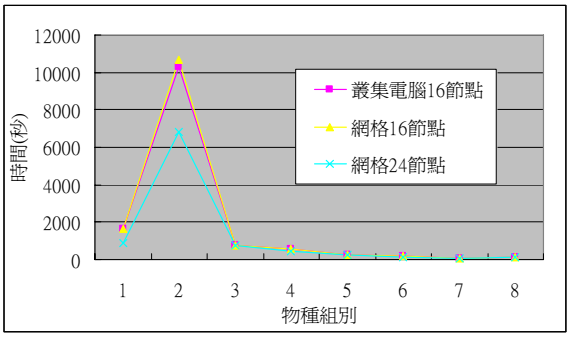


圖 7. 叢集電腦與網格使用不同節點數比較圖

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/)

17

UTCE: Ultrametric Tree Construction and Evaluation platform

Kun-Ming Yu, Jayi Zhou, Chun-Yuan Lin, Chuan Yi Tang

Introduction

UTCE is a platform with tools for ultrametric tree construction and tree evaluation. Phylogenetic trees can be used by biologists to describe the evolutionary relationship among of living species. Many models or tools have been proposed to construct phylogenetic trees. One of the popular models is an ultrametric tree with the assumption of a constant rate. UPGMA is one of well-known ultrametric tree building algorithms.

Although UPGMA often fails to reconstruct an evolutionary tree when a molecular clock does not apply, it is suitable for some cases of clocklike data. Moreover, the assumption of a constant rate can be used by biologists to estimate or disprove initial observations or hypotheses for their research work.

However, UPGMA is a heuristic algorithm and can not guarantee the constructed phylogenetic tree with minimum size. UTCE provides an efficient minimum ultrametric tree construction tool, **PBBU**, with a parallel branch-and-bound algorithm. The input of PBBU is a metric distance matrix or original/pre-aligned multiple DNA/protein sequences of species.

It should be noted that the constructed phylogenetic tree by PBBU could not be used to reject other phylogenetic trees by UPGMA and other tools. The goal of PBBU is to give another viewpoint for biologists to observe the evolution relationship of species under the assumption of a molecular clock hypothesis and the minimum evolution principle. In addition, in UTCE, two logical methods, 3PR and 4PR, are designed to evaluate a phylogenetic tree and/or the corresponding metric distance matrix.

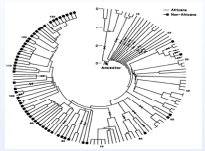
Acknowledgement

The work is partially supported by National Science Council. NSC 94-2213-E-216 -028

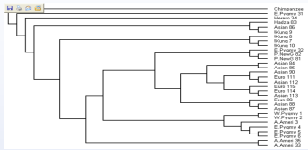
UTCE is freely available at:
http://pdcluster1.csie.chu.edu.tw/tree2

Parallel Branch-and-Bound

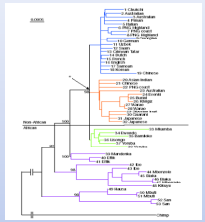
It is a parallel branch-and-bound algorithm to construct an minimum ultrametric tree from a distance matrix with the number of species and time constraints. User can specify the number of processors for computing and time constrain. The results will send by e-mail within given time.



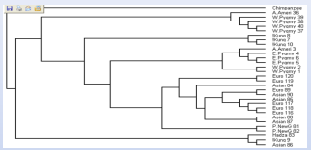
Science, 1991, mtDNA in D-loop, using PAUP* 3.0

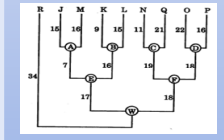


The results of PBBU. Here we can observe the Chimpanzee already separated with others.

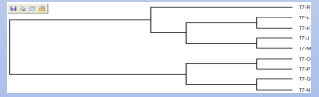


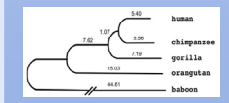
Nature, 2000, mtDNA not in D-loop, using PAUP*4.0





Science, 1992, T7 bacteriophage, NJ, UPGMA also can construct this tree.



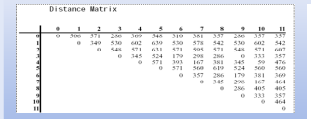


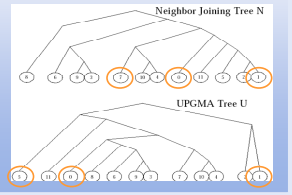
PNAS, 2006, NJ, non-coding region EnM001

We also obtain this tree by using UTCE with the same distance matrix from paper.

3-Point Relationship (3PR)

3PR is a logical method to check the relation for any triple of species (a, b, c) in distance matrix, which is preserved or not in the constructed phylogenetic trees. Moreover, it can illustrate the consistent between distance matrix and evolution for various tree construction methods.



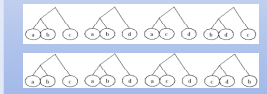


d[0,1] = 506 > d[0,7] = 381, however in the NJ tree, 0 and 1 are closer to each other than 0 and 7

4-Point Relationship (4PR)

4PR is another logical method to find contradictive species in distance matrix by checking the LCA relations in any set of 4 species.

For any set of 4 species (a, b, c, d), it has 4 triplets of species (a, b, c), (a, b, d), (a, c, d) and (b, c, d) and each triplet has its LCA relation. For any set of 4 species (a, b, c, d), no phylogenetic tree can conform to all of four LCA relations when they are ((a,b),c);, ((a,b),d);, ((a,c),d);, ((b,d),c); or ((a,b),c);, ((a,b),d);, ((a,c),d);, ((c,d),b); and this set is contradictive.



```
----- contradiction sets -----
283
W.Pygmy_1   A.American !Kung_7     European_8
W.Pygmy_1   A.American !Kung_10    European_8
W.Pygmy_1   E.Pygmy_6  !Kung_7     European_8
W.Pygmy_1   E.Pygmy_6  !Kung_10    European_8
W.Pygmy_2   W.Pygmy_1  E.Pygmy_4   A.American
W.Pygmy_2   W.Pygmy_1  E.Pygmy_5   A.American
W.Pygmy_2   W.Pygmy_1  E.Pygmy_6   A.American
W.Pygmy_2   European_8 !Kung_8     P.NewG_80
A.American  European_8 !Kung_8     P.NewG_80
.
.
.
```

The sample output contradiction set.

Sequence Alignment

A simple dynamic programming algorithm is used to compute the edit distance amount any two species (a,b) with DNA or protein sequences.