

行政院國家科學委員會專題研究計畫 成果報告

應用多核心與格網技術設計同特徵化合物平行化演算法及其 Web Service 之研發 研究成果報告(精簡版)

計畫類別：個別型
計畫編號：NSC 98-2221-E-216-023-
執行期間：98年08月01日至99年07月31日
執行單位：中華大學資訊工程學系

計畫主持人：游坤明
共同主持人：唐傳義
計畫參與人員：碩士班研究生-兼任助理人員：劉建元
碩士班研究生-兼任助理人員：吳書豪
碩士班研究生-兼任助理人員：林宏儒
博士班研究生-兼任助理人員：周嘉奕

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 99 年 10 月 29 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

應用多核心與格網設計技術設計同特徵化合物平行化演算法及其
Web Service 之研發

計畫類別： 個別型計畫 整合型計畫

計畫編號：NSC 98-2221-E-216-023-

執行期間：98 年 8 月 1 日至 99 年 7 月 31 日

執行機構及系所：中華大學資訊工程學系

計畫主持人：游坤明

共同主持人：唐傳義

計畫參與人員：吳書豪、劉建元、林宏儒、周嘉奕、

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

本計畫除繳交成果報告外，另須繳交以下出國心得報告：

赴國外出差或研習心得報告

赴大陸地區出差或研習心得報告

出席國際學術會議心得報告

國際合作研究計畫國外研究報告

處理方式：除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

中 華 民 國 99 年 10 月 25 日

一、摘要

近幾年來由於大量計算需求的科學研究及應用不斷成長，高效能計算平台的需求亦隨之增加，電腦計算平台也由單核心處理器走向多核心處理器、叢集系統、網格系統以及先進的圖形處理器 (GPU) 計算平台，藉由整合大量的運算單元來提供高效能的計算平台。然而現有各研究領域的重要課題大多是對單核心處理器所設計的循序演算法，無法將現有的技術直接運用在此新興的硬體架構之計算平台上，也因此許多傳統的循序計算演算法勢必要重新設計以適應新的平行計算平台，以有效發揮實際的硬體計算能力。在本計畫執行中，我們針對生物資訊領域及電腦輔助製藥領域中兩個重要的問題：最小權值演化樹 (Minimum Ultrametric Tree, MUT) 及化合物推論 (Chemical Compound Inference, CCI)，發展以多核心處理器為基礎之平行演算法，探討多核心處理器的特性及多執行緒實作可能遭遇的問題並且提出解決方案，包括：共同記憶體存取的機制、分時分區記憶體存取機制、動態多執行緒負載平衡機制，以及為上述兩個重要應用問題設計以多核心為架構之平行演算法，並且已將所得之成果整理成論文，於國際研討會上發表。

關鍵詞：多核心處理器、多執行緒、最小權值演化樹、化合物推論

Abstract:

Recently, the computation-intensive research topics and applications grow unceasingly, the demand of high-performance computation platform also increase. The computation platform moves towards multiple-core processor, cluster system, grid system as well as advanced graphics processor (GPU) platform, to provide a high efficiency, high throughput computation platform. However, the existing solutions for the important research topics in various research areas are mostly designed by using the single core processor technologies; they are unable to adapt the new parallel computation platform inevitably. In this project, we take multiple-core and graphics processor as the foundation to design an efficient parallel algorithm, and integrate the grid system to develop a conformity high-performance computation platform. In this project, we carried on four main research topics: First, the development of the multiple-core processor parallel computing techniques. Second, take the advantage of graphics processor to construct high efficient parallel algorithms. Third, integrates multiple-core processors and graphics processors to construct a distributed high-performance computation platform based on grid technology. Fourth, publishes above research results by web services. Moreover, the above parallel algorithms all aims at MUT (Minimum Ultrametric Tree) and CCI (Chemical Compound Inference), two important applications in the bioinformatics and drug research fields. We have organized obtained results and published two international conference papers in the projects.

Keywords: MultiCore, Multi-thread, MUT, CCI

二、 報告內容

1. 前言

近幾年來，由於大量計算需求的科學研究及應用不斷成長，高效能計算平台的需求亦隨之增加，電腦計算平台由單核心處理器 (Single-core processor) 走向多核心處理器 (Multi-core processor)、叢集系統 (Cluster system)、網格系統 (Grid system)、以及先進的圖形處理器 (Graphics Processing Unit, GPU)，藉由整合大量的運算單元來提供高效能的計算平台。由於單核心處理器在時脈上的提升會使得處理器溫度提高、耗電量隨之增加。因此目前處理器的設計不再以追求時脈做為唯一考量，轉而以多核心的封裝提供更強大的性能。從雙核心處理器問市至今，四核心處理器已成為主流的處理器，Intel 公司更表示將在五年內製造出擁有八十核心的處理器，並且也陸續於電腦展展示實驗性質的八十核心處理器。然而現有各領域的重要課題大多是對單核心處理器所設計的循序演算法 (Sequential algorithm)，將現有的技術運用在多核心的環境中並無法有效的使用硬體計算能力，相對的，目前多核心的時脈都較單核心低，甚至可能產生在多核心平台的效能比單核心平台還要低的情形，於是如何設計平行程式便是有效運用多核處理器的關鍵。

2. 研究目的

隨著科學計算的需求日益增加，高效能計算的需求也隨之增加。多核心處理器 (multi-core processor) 已成為目前主流 CPU 的發展方向，也將會是一長期的趨勢，主要有兩個關鍵：(1) 若 CPU 時脈繼續提升造成記憶體時脈無法配合，勢必要增加 cache 的容量以降低等待的時間，然而增加 CPU 的快取成本相對的高；(2) 藉由稍微降低處理器的性能可以有效的減少耗電量(將性能降低至原本的 87% 可以減少一半左右的耗電量)，若將兩個處理器放在一起後，耗電量與原來相當而效能卻能夠提升 73% 左右。這個特性便是目前主流 CPU 朝向多核心發展的最大原因。雖然多核心處理器平台能夠提供更多的計算能力，然而傳統的應用程式及演算法多為單核心處理器所設計，於多核心平台執行現有的程式並未能夠有效的運用多核心平台帶來效能提升的優勢，也迫使現有的課題及演算法需重新審視及設計以適應多核心的架構，然而不同的應用課題有著不同的限制及特性，要開發一種平行程式模型需要龐大的資金及人力的投入。雖然設計多核心程式有相當的困難度，然而多核心處理器確是未來 CPU 發展的趨勢，於是克服多核心處理器的設計挑戰並針對各領域重要的課題研發平行演算法是刻不容緩的項目。在本計畫執行中，我們設計以多核心為架構之演算法用於解決兩個重要的問題：最小權值演化樹 (Minimum Ultrametric Tree, MUT) 及化合物推論 (Chemical Compound Inference, CCI) 問題 除此之外，我們建立相關的 Web Service 提供給生物學家來使用。

3. 文獻探討

最小權值等距演化樹 (Minimum Ultrametric Tree, MUT)

演化樹建立是計算生物學相當重要的研究課題，同時也是分類學家極為重視的課題，分類學家可透過演化樹去觀察現今存在的物種彼間的演化關係及親疏程度，生物學家亦可以透過演化樹的建立及相關的演化知識去推斷其背後的意義。雖然，實際的演化關係未知，而所建立的演化樹皆是根據某個定義所產生的結果，因此，有許多不同的定義被提出，並嘗試去解釋其產生的結果與實際觀察結果間的異同之處。不同的模組被提出之後，相對產生了相當多計算相關的問題，然而這些問

題經長年的探討研究之後，最佳化的演化樹建立問題多屬於NP-hard，亦即在合理的時間內，能建立的演化樹其物種個數相當少。因此，相當多近似演算法(Heuristic Algorithm)被提出，如UPGMA、NJ、PAUP等，不過，對於物種個數不多時，最佳化的演化樹建立仍是有其必要性。

化合物推論(Chemical compound inference, CCI)

由於資訊科學、分子生物學、結構生物學、組合化學、藥物化學等的蓬勃發展，可以藉由電腦輔助藥物設計在短時間內進行大量且複雜的運算後得以縮短新藥研究發展的時間與成本。電腦輔助藥物設計主要是依照Receptor以及Ligand的已知與否可分成：Ligand known、Ligand unknown、Receptor known、Receptor unknown 這四選項進而判斷該使用何種藥物設計策略（可從事於新藥設計、老藥新用、非原廠藥物等設計策略）。化合物推論 (CCI) 技術是一種能產生一系列藥效基團相同但是結構不同的藥物。主要的概念為利用一個映射函式 (Mapping function) 將目標化合物映射至特徵空間 (feature space)，接用採用分支與界定技術找出所有與目標化合物有相同特徵且化學結構不同的化合物。

1. 當Receptor 和Ligand 都是已經有結構的情況下可以採用Docking [1,2] 技術的方式來進行藥物設計，將藥物資料庫 (Ligand database) 的資料在同一個 Receptor 位置進行大量幾何配對與能量計算。
2. 當只有Receptor 結構已知時，可以採用De Novo Ligand Design [3-6]技術的方式，在已知的Receptor 結構之中，置入許多分子片段，再以合理的方式將片段連接起來，形成符合化學原理的全新藥物結構。
3. 當只有Ligand 結構已知的時候，則可以採用3D-QSAR [7-11]或Pharmacophor [12-15] 技術的方式。
4. 當Receptor 和Ligand 的結構都是未知的情況下，目前尚無有效的藥物設計策略。

4. 研究方法

本計畫發展適用於多核心處理器下之平行演算法，針對多核心共享記憶體之架構以及多執行緒為基礎的設計概念為最小權值演化樹 (Minimum Ultrametric Tree, MUT) 及化合物推論 (Chemical Compound Inference, CCI) 設計平行演算法並且使用 web services 包裝研究成果。平行處理指的是處理器可以同時處理多件事情，而一般平行處理可以分為 (1)Task parallelism 以及 (2) Data parallelism。Task parallelism 主要是將多項不同的工作分派給不同的核心執行，這部份的平行化主要由作業系統有效的分派不同的程序 (process) 給不同的核心執行。而 Data parallelism 是本計畫考量的主要課題，然而多核心的程式撰寫以及除錯相當困難，並且容易發生死結 (Deadlock)、資料異變的問題。而多核心處理器有三個主要的挑戰：分割、平行與最佳化，能有一個良好的程式模型的話可較輕易的發展平行應用程式。一般 Data parallelism 的平行程式設計可以分為兩大類：(1) 共享記憶體模型(shared memory model) 以及 (2) 訊息傳遞模型 (message passing model)。

由於傳統平行程式的計算平台多為叢集系統或網格系統，於是訊息傳遞模型是平行演算法領域中大家比較熟悉的模型。然而多核心平台的特色是共享同一記憶體空間，此時若採用訊息傳遞模型不僅會造成不必要的資料封裝及傳遞，效能也比不上記憶體直接存取，於是對於多核心平台本計畫主要採用多執行緒 (multi-threaded) 做為程式開發的基礎。

5. 結果與討論

本計畫執行所獲之成果條列如下：

1. 提出並設計於多核心(Multi-core)的平行化演化樹建構演算法；
2. 提出平行化合物列舉演算法並可以列出在 input space 中找到最接近的近似解，可以應用該特性讓生物學家調整在feature space 中的值，並找出與該似接近之化合物；
3. 設計完成的演算法包裝成 web service 元件，使得國內外有興趣的相關團隊利用多核心計算資源快速解決 MUT 及CCI 課題；
4. 整理計畫成果為兩篇論文，並發表於EI等級之國際研討會議中(ICA3PP 2010及 SMC 2010)；
5. 將計畫成果整理成兩篇論文投稿至國際著名期刊尋求發表機會。

三、參考文獻

- [1] L. A.R., S. B.K., and P. C.E, "Prediction of protein–ligand interactions. Docking and scoring: successes and gaps," J. Med. Chem, vol. 49, pp. 5851-5855,2006.
- [2] P. E., W. W.P., and C. P.S, "A detailed comparison of current docking and scoring methods on systems of pharmaceutical relevance," Proteins, vol. 56, pp.235-249, 2004.
- [3] S. A. Khedkar, A. K. Malde, and E. C. Coutinho, "Design of Inhibitors of the MurF Enzyme of Streptococcus pneumoniae Using Docking, 3D-QSAR, and de Novo Design," J. Chem. Inf. Model, vol. 47, pp. 1839-1846, 2007.
- [4] Y. Iwata and M. Arisawa, "Discovery of novel aldose reductase inhibitors using a protein structure-based approach: 3D-database search followed by design and synthesis," J. Med. Chem., vol. 44, pp. 1718-1728, 2001.
- [5] E. Perola, K. Xu, and T. M. Kollmeyer, "Successful virtual screening of a chemical database for farnesyltransferase inhibitor leads," J. Chem. Inf. Model., vol. 43, pp. 401-408, 2000.
- [6] H. Mauser and M. Stahl, "Chemical Fragment Spaces for de novo Design," J.Chem. Inf. Model., vol. 46, pp. 318-324, 2007.
- [7] A. N. Jain, "Optimization of biaryl piperidine and 4-amino-2-biarylurea MCH1 receptor antagonists using QSAR modeling, classification techniques and virtual screening," J Comput Aided Mol Des., vol. 21, pp. 281-306, 2007.
- [8] G. O. Tiziano Tuccinardi, Armando Rossello, Claudiu T. Supuran, and Adriano Martinelli, "Homology Modeling and Receptor-Based 3D-QSAR Study of Carbonic Anhydrase IX," J. Chem. Inf. Model, vol. 47, pp. 2253-2262, 2007.
- [9] M. O. Taha, Y. Bustanji, A. G. Al-Bakri, A.-M. Yousef, W. A. Zalloum, I. M. Al-Masri, and N. Atallah, "Discovery of new potent human protein tyrosine phosphatase inhibitors via pharmacophore and QSAR analysis followed by in silico screening," Journal of Molecular Graphics and Modelling, vol. 25, 2007.
- [10] H. J. Bohm, "The Computer-Program Ludi - A New Method For The Denovo Design Of Enzyme-Inhibitors," J. Comput Aided Mol Des., vol. 6, 1992.
- [11] P. Prathipati and A. K. Saxena, "Evaluation of Binary QSAR Models Derived from LUDI and MOE Scoring Functions for Structure Based Virtual Screening," Journal of Molecular Graphics and Modelling. Chem. Inf. Model., vol. 46, pp.39-51, 2006.

- [12] M. J. McGregor, "A Pharmacophore Map of Small Molecule Protein Kinase Inhibitors," J. Chem. Inf. Model, vol. 47, pp. 2474-2382, 2007.
- [13] M. Rella, C. A. Rushworth, J. L. Guy, A. J. Turner, T. Langer, and R. M. Jackson, "Structure-Based Pharmacophore Design and Virtual Screening for Novel Angiotensin Converting Enzyme 2 Inhibitors," J. Chem. Inf. Model. , vol. 46, pp.708-716, 2006.
- [14] M. Gastreich, M. Lilienthal, H. Briem, and H. Claussen, "Ultrafast de novo docking combining pharmacophores and combinatorics," J Comput Aided Mol Des., vol. 20, pp. 717-734, 2006.
- [15] T. M. Steindl, D. Schuster, G. Wolber, C. Laggner, and T. Langer, "High-throughput structure-based pharmacophore modelling as a basis for successful parallel virtual screening," J Comput Aided Mol Des., vol. 20, pp. 703-715, 2006.

四、計畫成果自評

本計畫除了提出以多核心為架構之平行演算法解決MUT 與CCI問題外，並且將計畫的成果以 web services(如圖一、二) 形態包裝後發佈。此種方式除了可以滿足生物學家及製藥領域的學者使用上的方便，而且以標準 web services 包裝的物件可以直接引用在生物學家現有的工作流程軟體、或者與標準的 web service 串連，使我們的方法成為未來不可或缺的服務。本計畫之研究內容與原計畫相符，並且已達成預期之目標，研究成果不但具有學術價值，同時也具有一定之應用價值。

1. 對於參與之工作人員所獲得之訓練

此計畫的發展以多核心處理器平行演算法解決上述問題，開發過程中所使用及學習的多執行緒技術及 MPI 技術，對於未來參與人員投入軟體產業開發高效能程式有非常大的幫助。參與人員可以學習到平行程式開發的理論、實務、並且學習獨立思考的能力。在參與的過程中，也可以學習研究方法、團隊工作精神、以及論文撰寫經驗。

2. 學術期刊已發表於研討會並將投稿於國際期刊

本計畫的研究成果已在兩個EI國際研討會上發表(ICA3PP 2010及 SMC 2010，請參閱附錄)，並將整理成兩篇論文投稿至國際著名期刊尋求發表機會，並希望能結合國內外從事此方面研究的研發能量，發展更有國際競爭力及高效能的系統。

Home | Services | Portfolio | Forum | Information | Links | About us | Contact us

Menu

- » Home
- » Upload mol
- » Query Result
- » Monitor
- » temp

Add mol File

Upload mol File.

User E-mail :

mol File :

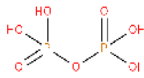
mol Text :

K value :

Submit :

Notice : Hello World. this line can say something about upload....

圖一 web service

Home Services Portfolio Forum Information Links About us Contact us	
Menu	Query Result.
<ul style="list-style-type: none"> » Home » Upload mol » Query Result » Monitor » temp 	<p>Please Input TaskID String.</p> <p>TaskID : <input type="text"/> <input type="button" value="Query"/></p> <p>Query Result as follow.</p> <p>k50539yh03mymqic4sbz6d0r.k03.inferred.00001.mol</p> <p>New mol Download</p> <p>smi : <chem>O(P(=O)(O)O)P(=O)(O)O</chem> k50539yh03mymqic4sbz6d0r.k03.inferred.00001.mol</p>  <p>1</p> <p>Notice : Hello World. this line can say something about query action....</p>

圖二 web service

五、 附錄

- [1]. Jiayi Zhou, Kun-Ming Yu, Chun Yuan Lin, Kuei-Chung Shih¹ and Chuan Yi Tang, “Balanced Multi-process Parallel Algorithm for Chemical Compound Inference with Given Path Frequencies,” The 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010) - Lecture Notes in Computer Science, Vol. 6082, Part II, pp.188-197, Springer-Verlag, May, 2010. (EI) (NSC98-2221-E-216-023 and NSC97-2221-E-216-020)
- [2]. Jiayi Zhou, Kun-Ming Yu* and Bin-Chang Wu, “Parallel Frequent Patterns Mining Algorithm on GPU,” 2010 IEEE International Conference on Systems, Man, and Cybernetics (SMC2010), pp. 435-440, 2010 (Istanbul, Turkey, Oct. 10 - 13, 2010). (EI). (NSC 98-2221-E-216-023), *corresponding author.

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

達成目標

未達成目標（請說明，以 100 字為限）

實驗失敗

因故實驗中斷

其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文： 已發表 未發表之文稿 撰寫中 無

專利： 已獲得 申請中 無

技轉： 已技轉 洽談中 無

其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

本計畫提出並設計於多核心(Multi-core)的平行化演化樹建構演算法，可以列出在 input space 中找到最接近的近似解，可以應用該特性讓生物學家調整在 feature space 中的值，並找出與該似接近之化合物；同時在本計畫得執行過程中，我們亦提出一個 GPU 平行化資料探勘演算法，有效加快資料探勘時間。本計畫亦將設計完成的演算法包裝成 web service 元件，使得國內外有興趣的相關團隊利用多核心計算資源快速解決 MUT 及 CCI 課題。

本計畫執行所獲得的成果將對 MUT、CCI 及資料探勘等問題，或是對於多核心計算領域之研究學者有相當大的助益。

行政院國家科學委員會補助國內專家學者出席國際學術會議報告

98 年 8 月 25 日

報告人姓名	游坤明	服務機構 及職稱	中華大學資工系 教授
時間 會議地點	98/8/17 –98/8/19 Nanchang, China	本會核定 補助文號	
會議 名稱	(中文) 2009 IEEE 粒度計算國際研討會 (英文) 2009 IEEE International Conference on Granular Computing		
發表 論文 題目	(中文) 運用 MPI 與平行分支與界定技術進行化學化合物之推論 (英文) Parallel Branch-and Bound Approach with MPI Technology in Inferring Chemical Compounds with Path Frequency		

一、 參加會議經過

會議的開幕典禮由江西財經大學訊息管理學院院長致簡單的歡迎詞，並由會議的委員會主席說明本屆的投稿與錄取篇數 (投稿 345 篇，錄取 165 篇) 及本屆會議的特色與會議重點後，隨即展開，本屆會議分別於第一天及第二天的會議議程中各安排了三個場次之粒度計算與生物資料處理暨應用領域之最新趨勢之專題報告: (1). Generalized Bags and Their Relations: An Alternative Model for Fuzzy Set Theory and Applications, (2). Formal Theory of Granular Computing and Two Major Applications, (3). High-throughput DNA Sequencing and Bioinformatics: Bottlenecks and Opportunities, (4). Ubiquitous/Pervasive Intelligence: Visions and Challenges, (5). Ubiquitous Personalized Information Processing with Wildcard 以及 (6). Computational Models in Systems Biology，分別由 Sadaaki Miyamoto (University of Tsukuba, Japan)、T. Y. Lin (San Jose State University, USA)、Stephen Kwok-Wing TSUI (The Chinese University of Hong Kong, China) Laurence T. Yang (St Francis Xavier University, Canada)、Xindong Wu (The University of Vermont, USA) 以及 Xue-wen Chen (The University of Kansas, USA) 作精彩的專題報告。正式之論文報告分別於第三個場次的專題報告後進行，本人之論文『Parallel Branch-and Bound Approach with MPI Technology in Inferring Chemical Compounds with Path Frequency』被安排在第一天的 – Session C3 “Intelligent Data

Analysis and Applications”之場次發表。

二、 與會心得

GrC 2009 是一個在粒度計算(Granular computing)研究領域中具有指標性的最重要國際會議，此次會議共有三百四十多篇的論文投稿，但僅錄取了一百六十餘篇優秀的粒度計算資訊系統領域的論文，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，參與此次會議之篇學者可藉著此次會議，在粒度計算資訊系統與生物資訊運用之各研究領域互相作深入的討論，而且可藉此機會認識在此領域中之權威研究學者，對於此後從事此領域之研究有相當之助益。

三、 考察參觀活動(無是項活動者省略)

無。

四、 建議

無。

五、攜回資料名稱及內容

1. 大會議程
2. The Proceeding of 2009 IEEE International Conference on Granular Computing 研討會論文集。

六、 其他

Balanced Multi-process Parallel Algorithm for Chemical Compound Inference with Given Path Frequencies*

Jiayi Zhou¹, Kun-Ming Yu^{2**}, Chun Yuan Lin³, Kuei-Chung Shih¹,
and Chuan Yi Tang¹

¹ Department of Computer Science,
National Tsing Hua University, Hsinchu 300, Taiwan
jyzhou@mx.nthu.edu.tw, shiuh0307@seed.net.tw,
cytang@cs.nthu.edu.tw

² Department of Computer Science and Information Engineering,
Chung Hua University, Hsinchu 300, Taiwan
yu@pdlab.csie.chu.edu.tw

³ Department of Computer Science and Information Engineering,
Chang Gung University, Taoyuan 333, Taiwan
cyulin@mail.cgu.edu.tw

Abstract. To enumerate chemical compounds with given path frequencies is a fundamental procedure in Chemo- and Bio-informatics. The applications include structure determination, novel molecular development, etc. The problem complexity has been proven as NP-hard. Many methods have been proposed to solve this problem. However, most of them are heuristic algorithms. Fujiwara *et al.* propose a sequential branch-and-bound algorithm. Although it reaches all solutions and avoids exhaustive searching, the computation time still increases significantly when the number of atoms increases. Hence, in this paper, a parallel algorithm is presented for solving this problem. The experimental results showed that computation time was reduced even when more processes were launched. Moreover, the speed-up ratio for most of the test cases was satisfactory and, furthermore, it showed potential for use in drug design.

Keywords: Branch-and-bound algorithm, load-balancing, chemical compound inference, drug design.

1 Introduction

The enumeration of chemical compounds that has the same characteristics is one of the fundamental issues in Chemo- and Bio-informatics. Its applications include structure determination using mass-spectrum [1-2], reconstructing molecular structure with given signatures [3-4], classification of compounds [5], etc. In an effort to improve on existing algorithms, many studies have proposed different ways of dealing with the

* This work is partially supported by Nation Science Council. (NSC98-2221-E-216-023 and NSC97-2221-E-216-020).

** Corresponding author.

enumeration of constraints for other purposes, such as the virtual exploration of chemical universe [6-7]. However, none can guarantee that all solutions can be found since they are based on heuristic algorithms and require additional operations to avoid generating isomorphic results.

The *Kernel Methods (KMs)* approach maps the data in the *input space* into a high dimension *feature space*. The data are computed as points in the *feature space* where each coordinate represents one feature of the data. In applying *KMs* to chemical compounds, all compounds are mapped to *feature vectors* in the *feature space*. The definition of feature vector is widely based on *frequencies of labeled paths* [8] or *frequencies of small fragments* [5]. In this study, the *pre-image* problem was focused on by enumerating all chemical compounds with the same path frequency. The desired object was computed as a point in the feature space, and then the point was mapped back to the input space. This point was called the *pre-image* of the point. A given point in the feature space (*frequency path*) was then mapped back to the point in the input space (*chemical compound*). Let ψ be the mapping function, then the *pre-image* is defined as follows: given a point y in the feature space, to find all x in the input space such that $y = \psi(x)$. Therefore, the chemical compound inference (*CCI*) is defined as follows: given a target compound c and computed path frequency $\psi(c)$. The objective is to infer all compounds $c_1 \dots c_n$ such that $\psi(c_i) = \psi(c)$ for $i = 1, \dots, n$. Since the solution for *CCI* can be applied to new chemical compounds with the same path frequencies, it may be useful in drug design.

Akutsu and Fukagawa have proved that to enumerate chemical compounds with given constraints is an NP-hard problem [9]. The branch-and-bound algorithm is generally used to solve a wide variety of NP-hard problems [10], such as *Traveling Salesman*, *Knapsack*, *Vertex Covering*, avoiding exhaustive searches. For *CCI* problem, Fujiwara *et al.* have proposed an efficient branch-and-bound algorithm [11]. The branching and bounding strategies are based on the path frequency and the valence constrains of atoms, respectively, to avoid the generation of an invalid tree, thus reducing the search space. Although this algorithm outperforms the exhaustive search algorithms, the computation time increases significantly when the number of atoms in a compound grows.

A balanced multi-process parallel branch-and-bound algorithm for *CCI* (*BMPBB-CCI*) was designed in this study. In *BMPBB-CCI*, the searching space is dynamic divided into p subspaces with p processors. Since each subspace is independent, it can be processed by a processor and then applied to a sequential branch-and-bound algorithm. Hence, it minimizes the parallel communication cost between processors since only local communication is required. Moreover, two types of queuing, *global queue (GQ)* and *local queue (LQ)*, are used for load-balancing in branching. A hashable and a serial data structure were used to store the path frequencies instead of the *Trie* structure because it can be transferred between processors as a plain string.

The development trend in computers is one of multi-core processors. Adding more cores to a computer makes it faster, but it also leads to difficulties in designing programs. Although OpenMP [12] is a standard and easy to use multi-threading library that gives the performance of multi-core processors, it is not flexible enough for

delicate operations or for designing complicated parallel algorithm. Therefore, *BMPBB-CCI* was implemented using multi-process architecture (*MPA*). Different from multi-thread architecture, for long running programs, *MPA* obtains memory protection and access control benefits of operating systems. In addition, when a process crashes it does not affect the remaining running processes. Moreover, it also could be easily extended to distributed memory multi-processors system, e.g. a multi-node cluster. The experimental results showed that *BMPBB-CCI* found the optimal solution for chemical compounds in a short time. In addition, it also had a satisfactory speed-up ratio.

2 Preliminaries

Being different from a *simple* graph, a graph that allows multiple edges is called a *multigraph*. A multigraph without self-loop and cycle is called a *multitree*. Let Σ denote a label set with each label representing a symbol of an atom, e.g. $\Sigma = \{C, O, H\}$. A Σ -labeled multitree can be denoted as $T = (V, E)$ with a vertex set V and edge set E . A valence function, $val : \Sigma \rightarrow \mathbb{Z}^+$, is introduced to be the maximum number of bonds an atom can hold, e.g. $val(C) = 4$ and $val(O) = 2$. For any vertex v in T , the valence of v is $val(l(v))$, where l is the function returning label $\in \Sigma$ of vertex v . A chemical compound can be treated as a (Σ, val) -labeled multitree, where the number of edges represents the chemical bonds between two vertices (atoms). The path frequencies of the (Σ, val) -labeled multitree are defined. Let $P = (v_0, \dots, v_s)$ be the path in multitree T and $l(P) = (l(v_0), \dots, l(v_s))$ be the labeled sequence of the path. For a labeled sequence t , let $occ(t, T)$ stand for the number of paths of t in multitree T , where the multi-edge is treated as a single edge. Then Σ^k denotes the set of sequences with k labels, and $\Sigma^{\leq k} = \bigcup_{j=1}^k \Sigma^j$. The path frequencies with level K are defined as $f_K(T) = occ(t, T)_{t \in \Sigma^{K+1}}$. Fig. 1 illustrates the path frequencies of un-rooted (Σ, val) -labeled multitree (a chemical compound, $C_2O_2H_4$) with level $K=1$. Where all paths are treated as "directed", thus $occ(OH, T) = occ(HO, T) = 1$ and so on.

CCI is further defined in the following [11]. Given a finite label $K \geq 1$, a target with path frequencies g , and a valence function $val : \Sigma \rightarrow \mathbb{Z}^+$. Find all (Σ, val) -labeled multitree T such that $f_K(T) = g$ and $deg(v) = val(l(v))$ for each vertex v in multitree $T = (V, E)$. However, the main concern with *CCI* is to avoid the generation of isomorphic chemical compounds. Many methods [13-14] propose to solve this problem by choosing a unique vertex or a unique pair of adjacent vertices as a root. Fujiwara *et al.* [11] applied *centroid-rooted* [15] *left-heavy* properties (Theorem 1) to avoid the generation of isomorphic chemical compounds.

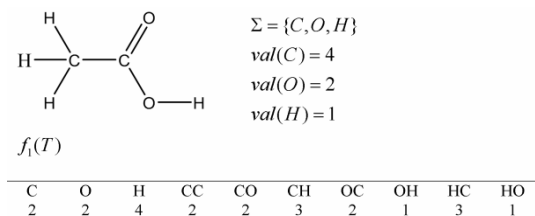


Fig. 1. Example of (Σ, val) -labeled multitree T

Theorem 1. For any tree with n vertices, either there is a unique vertex v^* such that each subtree obtained by removing v^* to contain at most $\lfloor (n-1)/2 \rfloor$ vertices, or there is a unique edge e^* such that both of the subtrees obtained by removing e^* contain exactly $n/2$ vertices.

Vertex v^* is identified as a *unicentroid* of the tree, or a *bicentroid* for e^* . In order to introduce *left-heavy* properties, multitree T is indexed using a depth-first search (DFS) order. In general, the vertex sequence, v_0, v_1, \dots, v_n , is labeled by DFS from the root vertex. The depth function $d(v)$ of vertex v is the number of edges in $P(v)$, where the depth of the root vertex is 0. The *depth label sequence* of T is defined as $DL(T) = (d(v_0), l(v_0), \dots, d(v_{n-1}), l(v_{n-1}))$. $T(v)$ denotes a subtree rooted at vertex v and all of its descendants. The *left-heavy* properties are defined as follows. T is *left-heavy* if $i < j$ implies $DL(T(v_i)) \geq DL(T(v_j))$ for any two siblings v_i and v_j .

For any two depth label sequences $DL(T_1) = (d_{1,0}, l_{1,0}, d_{1,1}, l_{1,1}, \dots, d_{1,n_1}, l_{1,n_1})$ and $DL(T_2) = (d_{2,0}, l_{2,0}, d_{2,1}, l_{2,1}, \dots, d_{2,n_2}, l_{2,n_2})$, $DL(T_1) > DL(T_2)$ means that there is a $i \in [1, \min(n_1 - 1, n_2 - 1)]$ such that $d_{1,j} = d_{2,j}$ and $l_{1,j} = l_{2,j}$ for $j = 0, \dots, i - 1$. In addition to either (1) $d_{1,i} > d_{2,i}$, or (2) $d_{1,i} = d_{2,i}$ and $l_{1,i} > l_{2,i}$.

Fujiwara *et al.* [11] propose a branch-and-bound algorithm to enumerate all tree-like chemical compounds with given path frequencies. It starts from an empty multitree, then iteratively creates a multi-tree rooted in atom v , where $l(v) \in \Sigma$. After that, new children multitree offspring can be obtained by inserting vertex v' where $l(v') \in \Sigma$ at the right-most path. If T violates (1) *centroid-rooted* constraints, (2) $f_K(T) \leq g$, and (3) $\deg(v) \leq val(l(v))$ for each $v \in T$ then candidate T is bounded immediately.

3 BMPBB-CCI

BMPBB-CCI was designed on shared memory multi-core computers. However, the built-in shared memory facilities of the operating system (OS) somewhat restricted

portability to another OS. In addition, this facility allows fewer types of data structures. Therefore, a socket-based manager process was implemented which held various types of data structures. In addition, the manager also could be extended to distributed memory computing architecture, such as cluster systems. Fig. 2 shows the framework of the *BMPBB-CCI*, where the *manager module* of the *manager process* implemented a socket-based communication object which supported the data structure, such as *list*, *double-ended queue (dequeue)* and *dictionary*. A *Global Queue (GQ)* was also implemented on the manager process as well as a *Local Queue (LQ)* on each *computing process* to balance the workload among processors and to reduce inter-process communication.

There are three different processes, *main process (MP)*, *computing process (CP)* and *manager process (MgP)*. Since the valence of the H atom is 1 and always attached in the leaf node, the H atom can be removed during branching operations. The path frequency with the H atom is computed in step 2. After that, in steps 3 and 4, the *MgP* is created and started, and the required shared objects are also allocated in this step. In steps 5 and 6, the *CP* is created and launched at each computing core. Moreover, the obtained Id of created shared objects is passed to each *CP*. Finally, the *MP* joins all launched *CPs* until it is terminated and then *MP* writes the results to disk.

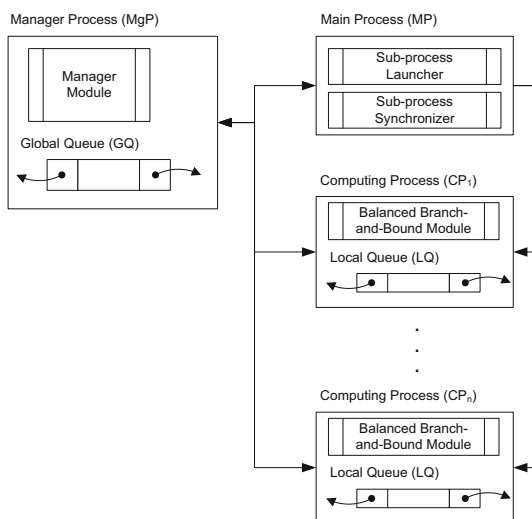


Fig. 2. Multi-process framework of BMPBB-CCI

The branch and bound operations are done in *CP*. Moreover, *CP* also uses the shared object to balance the workload. An overview of the algorithm of *CP*, show that it is an infinite loop. It loops the algorithm over until all solutions are found. From steps 2 to 6, the compound in *LQ* is chosen first. If the *LQ* is empty, then the compound in the *GQ* is chosen if that is not empty. Otherwise, the new atom in *insertAtomQueue* is selected as a new root of the candidate compound in step 5. The

bounding operations are applied in steps 7 to 13. H atoms are inserted back to *comp* to check the feature vector constrains and valence constrains in steps 8 and 9. The *centroid-rooted* and *left-heavy* properties are verified in step 10 to avoid generation of isomorphic chemical compounds. If the *comp* passes all verifications, it is inserted into *resultQueue* in step 11. The *comp* is dropped immediately if its path frequency $f_K(comp)$ greater than g_{noH} (step 13). Steps 14 to 17 are branching operations. In step 16, the potentially new pair of atoms are verified in the g_{noH} . If the pair is presented in g_{noH} then the new candidate compound *newComps* is generated. Moreover, borrow a single bond transformation as proposed by Fujiwara *et al.* [11]. The number of bonds of the attached new vertex is limited by the maximum number of bonds between pairs of atoms in the target compound. Therefore, the searching spaces can be significantly reduced to save on computation time. In order to balance the workload, when there are too few compounds in the *GQ*, the *newComps* are appended to *GQ*, otherwise the *newComps* are appended to *LQ* (step 17). Since the *GQ* is filled during branching operations, the *CP* can immediately acquire the candidate compounds from *GQ* without waiting for other *CPs* to transfer candidates.

MgP creates and obtains a shared object via socket-based connections. The benefits of the shared object facilities are (1) they deal with various types of data structures, (2) they have built-in synchronization facility, and (3) they share objects from different computers. The algorithm for *BMPBB-CCI* is given below.

Algorithm. BMPBB-CCI

Input: Target compound and valence function

Output: Chemical compound which confirms to path frequencies of given compound

Environment: Multi-core architecture computers

Main Process (MP)

- Step 1: Remove H atoms from given target compound and compute its path frequency g_{noH} .
- Step 2: Insert the H atoms back to the given target compound and compute its path frequency g_H .
- Step 3: Create and start *MgP* and create a global queue on *MgP*.
- Step 4: Allocate shared object form *MgP*, *resultQueue* and *insertAtomQueue*.
- Step 5: Create *CP* on each computing core in processor.
- Step 6: Start *CP* with following parameters: g_{noH} , g_H , *resultQueue*, *insertAtomQueue*.
- Step 7: Wait all started *CPs* terminated.
- Step 8: Write the results to disk.

Computing Process (CP)

- Step 1: **while True:**
- Step 2: **if** local queue **is not** empty:
 comp = pop last item of local queue
- Step 3: **else:**
 if global queue **is not** empty:
 comp = pop last item of global queue
 else:
-

```

Step 4:      if resultQueue is not empty:           break
Step 5:      else if insertAtomQueue is not empty:
              comp = pop a atom from insertAtomQueue and create new compound
Step 6:      else:           break
Step 7:      if num of atom of comp = target compound w/o H:
Step 8:      Add H atom to comp
Step 9:      if  $f_K(comp) \neq g_H$  :           continue
Step 10:     Check the centroid-rooted and left-heavy properties
Step 11:     If pass the checks, add the comp to resultQueue
Step 12:     else: Check if  $f_K(comp) \leq g_{noH}$ 
Step 13:     for  $v$  in right most path of comp:
Step 14:     if  $v$  is the last atom on right most path:
              insertAtom =  $\Sigma$ 
              else: insertAtom =  $s$  for all  $s < l(\text{next vertex of } v)$ 
Step 15:     for  $s$  in insertAtom:
              if  $(s, v)$  in  $g_H$  :
                  Insert  $s$  on  $v$  to generate new candidate  $newComp_0, \dots, newComp_q$  such that bond
                   $\leq$  the maximum number of bond of pair .. in target compound
Step 16:     if  $\text{len}(GQ) \leq (\text{numberOfProcessors})^2 / 2$ 
              Insert  $newComp_i$  to  $GQ$  for  $i = 0, \dots, q$ 
              else: Insert . to  $LQ$  for  $i = 0, \dots, q$ 

```

Manager Process (MgP)

```

Step 1:      Create a socket and waiting for the request.
Step 2:      if incoming request is shared object creation:
              Create a shared object of request data type
              return shared object Id
Step 3:      if incoming request is shared object read:
              Read the corresponding shared object of given Id
              return the read value
Step 4:      if incoming request is shared object write:
              Acquire a lock object
              Write the given data to shared object of given Id
              Release a lock object

```

4 Experimental Results

BMPBB-CCI was implemented in Python language 2.6 on *IBM System x 3650 T* consisting of 2 Intel Xeon 3.20GHz CPUs (8 computing cores, 4 cores per CPU). Two data sets were used to verify *BMPBC-CCI*. One was KEGG LIGAND database [16], the other was a set of 22 compounds for neuraminidase (NA) inhibitors of the influenza A virus from Zhang *et al.* [17].

The chemical compounds in KEGG LIGAND's database were used to verify the performance of *BMPBB-CCI*, and path frequencies were computed for levels 1 and 2. Due to page limitation, selected compounds and their properties are shown in Table 1, where (1) C00064, C00073, and C00077 are the entries for L-Glutamine, L-Methionine and L-Ornithine in the KEGG LIGAND database, (2) n_1 is the number of atoms of an entry and n_2 is the number of atoms which removed H atoms, (3) fs is the number of feasible solutions found. Fig. 3 (a)-(c) shows the computation time of

BMPBB-CCI with a different number of processes. It was found that a large number of processes (*computing process, CP*) reduced computation time. Moreover, smaller K values had fewer constraints and more feasible solutions were found (see Table 1). Therefore, there were more candidate compounds in branch-and-bound operations, leading to larger solution spaces to be traversed. The computation time was long when level K was small. Fig. 3 (d) illustrates the speed-up ratio of *BMPBB-CCI* from (a)-(c). The speed-up ratio increased when the number of processes increased. Moreover, the speed-up ratios were satisfactory even for 8 processes. This result showed that *BMPBB-CCI* was scalable.

Table 1. Properties of selected compounds

Entry Formula	n_1	n_2	K	fs
C00064	20	10	1	274
$C_5H_{10}N_2O_3$			2	3
C00073	21	9	1	339
$C_5H_{11}NO_2S$			2	2
C00077	21	9	1	236
$C_5H_{12}N_2O_2$			2	3

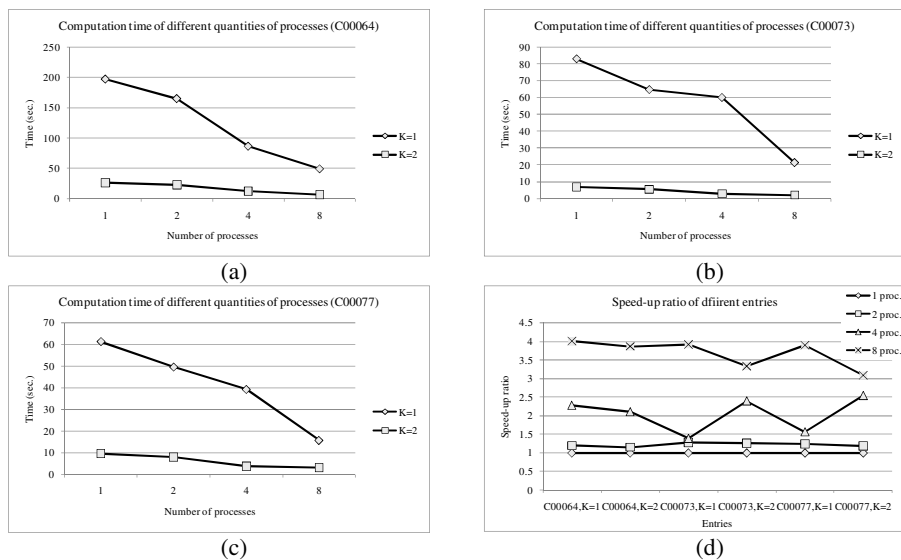


Fig. 3. Computation time and speed-up ratio

BMPBB-CCI was used to infer novel chemical compounds for the second data set to show that it had potential for drug design. A pharmacophore model consists of a 3D arrangement of a collection of features necessary for the biological activity of ligands (compounds). At first, the pharmacophore model (by Accelrys DiscoveryStudio) was

constructed for NA of an influenza A virus from Zhang *et al.* [17]. Then the model was tested with Tamiflu and Zanamivir. The results (Table 2) showed that it was reliable since the predicted IC_{50} and actual IC_{50} were on the same quantity level. Finally, the model was used to test new chemical compounds inferred by *BMPBB-CCI*. Due to page limitation, only the compound Cpd was used as an example (Table 2). The new compound Cpd-Reb had a better predicted IC_{50} and Fit value than Cpd-Ori. DiscoveryStudio CDOCKER's docking program was also used to compute the interactions between compound and protein. The best CDOCKER interaction energy for Cpd-Ori was 42.565. The best CDOCKER interaction energy for Cpd-Reb was 45.324. The docked poses are given in Fig. 4. These results showed that the novel compound Cpd-Reb may be a candidate for NA inhibiting of the influenza A virus.

Table 2. Results for test compounds in the pharmacophore model

Compound	Actual IC_{50} (nM)	Estimated IC_{50} (nM)	Fit value	Mapped feature				
				HD1*	HD2*	HY*	NI*	PI*
Tamiflu	1	8.502	10.95		+	+	+	+
Zanamivir	1.3	4.848	11.194	+	+		+	+
#Cpd-Ori	6300	5639.67	8.129		+		+	+
#Cpd-Reb	NA	6.577	11.062	+	+	+		+

*HD1: hydrogen-bond donor 1; HD2: hydrogen-bond donor 2; HY: hydrophobic group;
NI: negative ionizable group; PI: positive ionizable group.

#Cpd-Ori: original compound Cpd; Cpd-Reb: novel compound inferred by *BMPBB-CCI*.

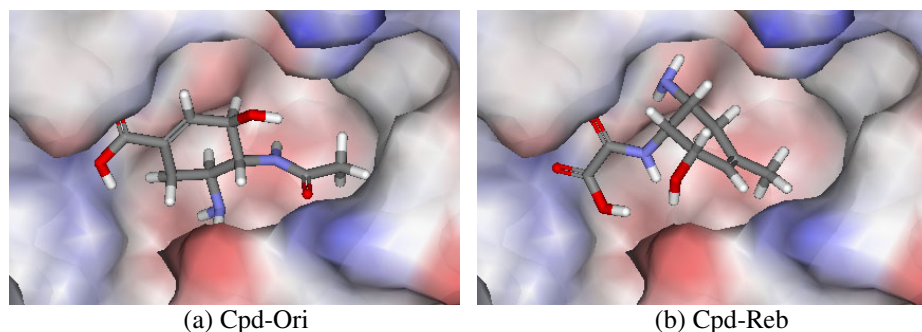


Fig. 4. The Cpd-Ori and Cpd-Reb docked pose into NA protein (PDB code: 2hu4)

5 Conclusions

An algorithm (*BMPBB-CCI*) was designed and verified on multi-core computers. From the experimental results, it was observed that *BMPBB-CCI* reduced computation time with more processors in the case of the KEGG LIGAND database. It also achieved a satisfactory speed-up ratio for most of the test cases. Moreover, it showed potential for drug design.

Acknowledgement

We are grateful to the National Center for High-performance Computing for computer time and facilities.

References

1. Buchanan, B., Feigenbaum, E.: DENDRAL and Meta-DENDRAL, Their Applications Dimension. *Artif. Intell.* 11, 5–24 (1978)
2. Funatsu, K., Sasaki, S.: Recent advances in the automated structure elucidation system, chemics. Utilization of two-dimensional nmr spectral information and development of peripheral functions for examination of candidates. *J. Chem. Inf. Comput. Sci.* 36(2), 190–204 (1996)
3. Faulon, J., Churchwell, C., Visco Jr., D.: The signature molecular descriptor. 2. Enumerating molecules from their extended valence sequences. *J. Chem. Inf. Comput. Sci.* 43(3), 721–734 (2003)
4. Hall, L., Dailey, R., Kier, L.: Design of molecules from quantitative structure-activity relationship models. 3. Role of higher order path counts: path 3. *J. Chem. Inf. Comput. Sci.* 33(4), 598–603 (1993)
5. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowl. Data Eng.* 17(8), 1036–1050 (2005)
6. Fink, T., Reymond, J.: Virtual Exploration of the Chemical Universe up to 11 Atoms of C, N, O, F: Assembly of 26.4 Million Structures (110.9 Million Stereoisomers) and Analysis for New Ring Systems, Stereochemistry. *J. Chem Inf. Model.* 47(2), 342–353 (2007)
7. Mauser, H., Stahl, M.: Chemical fragment spaces for de novo design. *J. Chem. Inf. Model.* 47(2), 318–324 (2007)
8. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: *ICML*, pp. 321–328 (2003)
9. Akutsu, T., Fukagawa, D.: Inferring a Graph from Path Frequency. In: Apostolico, A., Crochemore, M., Park, K. (eds.) *LOPSTR 2004*. LNCS, vol. 3573, pp. 371–382. Springer, Heidelberg (2005)
10. Yu, C., Wah, B.: Efficient branch-and-bound algorithms on a two-level memory system. *IEEE Trans. Softw. Eng.* 14(9), 1342–1356 (1988)
11. Fujiwara, H., Wang, J., Zhao, L., Nagamochi, H., Akutsu, T.: Enumerating Treelike Chemical Graphs with Given Path Frequency. *J. Chem. Inf. Model.* 48(7), 1345–1357 (2008)
12. OpenMP, <http://openmp.org/>
13. Nakano, S., Uno, T.: Generating colored trees. In: Kratsch, D. (ed.) *WG 2005*. LNCS, vol. 3787, pp. 249–260. Springer, Heidelberg (2005)
14. Wright, R., Richmond, B., Odlyzko, A., McKay, B.: Constant time generation of free trees. *SIAM J. Comput.* 15, 540 (1986)
15. Jordan, C.: Sur les assemblages de lignes. *J. Reine Angew. Math.* 70(185), 81 (1869)
16. KEGG Ligand database, <http://www.genome.jp/kegg/ligand.html>
17. Zhang, J., Yu, K., Zhu, W., Jiang, H.: Neuraminidase pharmacophore model derived from diverse classes of inhibitors. *Bioorg. Med. Chem. Lett.* 16, 3009–3014 (2006)

Parallel Frequent Patterns Mining Algorithm on GPU

Jiayi Zhou

Department of Computer Science
National Tsing Hua University
Hsinchu 300, Taiwan
jyzhou@mx.nthu.edu.tw

Kun-Ming Yu[†]

Department of Computer Science
and Information Engineering
Chung Hua University
Hsinchu 300, Taiwan
yu@chu.edu.tw

Bin-Chang Wu

Department of Computer Science
and Information Engineering
Chung Hua University
Hsinchu 300, Taiwan
aquavit@pdlab.csie.chu.edu.tw

Abstract—Extraction of frequent patterns from a transactional database is a fundamental task in data mining. Its applications include association rules, time series, etc. The Apriori approach is a commonly used generate-and-test approach to obtain frequent patterns from a database with a given threshold. Many parallel and distributed methods have been proposed for frequent pattern mining (FPM) to reduce computation time. However, most of them require a Cluster system or Grid system. In this study, a graphic processing unit (GPU) was used to perform FPM with a GPU-FPM to speed-up the process. Because of GPU hardware delimitations, a compact data structure was designed to store an entire database on GPU. In addition, MemPack and CLProgram template classes were also designed. Two datasets with different conditions were used to verify the performance of GPU-FPM. The experimental results showed that the speed-up ratio of GPU-FPM can achieve 14.857 with 16 times of threads.

Keywords—frequent pattern mining, parallel processing, graphic processing unit (GPU), OpenCL

I. INTRODUCTION

Both in business and scientific research, there has been a tremendous growth of data that needs to be processed. Extracting information from large amount of data is necessary in making correct and effective decisions. Different methods have been developed to determine the characteristics and interrelationships of data. Association rule learning, classification, clustering, and regression commonly need to mine data. Extraction of frequent patterns from a transaction-oriented database is one of the most important of these. Frequent patterns represent the number of times an itemset appears in a given database. Therefore, if an itemset is frequent it means there are strong relationship between items.

Most frequent patterns mining (FPM) either uses the generate-and-test (*Apriori*-like) [1] or the pattern growth approach (*FP-growth*) [2]. The core concept of the Apriori-like approach is that if the length k pattern is not frequent in the database, then the super-pattern (length $k+1$) will not be frequent. It uses a bottom-up approach, extending frequent subsets one item at a time. Since the generated candidates are independent, this approach is suitable for parallelization.

Although many Apriori-like methods have been proposed [3-4], the computation time increases significantly when the database contains a large number of transactions. Some studies [5-7] apply parallel and distributed techniques to speed-up the mining processes. However, most of them require a high performance computing system, e.g., a Cluster or Grid System.

In recent years, the graphic processing unit (GPU) has developed from a 3D rendering device for games to a general-purpose computing device [8]. While the GPU can only execute some simple instructions and functions, compared with CPU, it has a large number of computing units. Moreover, using GPU as a high-performance computing device which does not only reduce the deployment cost but also saves on maintenance. GPU programming strategies can be classified according to either graphic APIs or GPU programming language. It is difficult for developers to use the graphic APIs since they need understand the graphic hardware and encode their data to graphic vectors. The most serious problem is the use of previously designed parallel algorithms. Therefore, GPU programming language is currently being used to develop GPU-enabled programs. NVIDIA and ATI have been proposed as GPU programming language by CUDA [9] and Stream [10] respectively. Programs can be written in C programming language (C99) to use the power of GPU. However, CUDA can only be used on NVIDIA's GPU and vice versa. Therefore, OpenCL [11] was proposed in 2009 to deal with this situation. The program design with OpenCL not only can be executed on different brand GPU devices, but also on multi-core CPUs.

In this study, the GPU-FPM algorithm was used to speed-up the mining processes for FPM when using GPU. Although GPU is a powerful computing device, there are limitations: like memory size, memory latency, etc. Therefore, the data structure has to be re-designed for the FPM algorithm on GPU. Since the verification time dominates computation time, the main goal of GPU-FPM is to use GPU to verify generated candidates in order to speed-up the FPM processes. Compact Data Structure, MemPack, and CLProgram class were used to achieve this. For verifying the GPU-FPM performance, it was implemented on Microsoft Windows with OpenCL 1.0, in addition, data generated by an IBM Quest Data Generator [12] was used. The proposed algorithm was tested under different conditions, including different transaction lengths, threads,

--
* This work is partially supported by National Science Council. (NSC 98-2221-E-216-023)

† The corresponding author.

block sizes, and thresholds. The experimental results showed that GPU-FPM significantly reduced the computation time with increasing threads. The speed-up ratio achieved 14.857 with 16 times of threads (in case of the T40110D100K threshold being 1900, and block size 10). Moreover, even in the worst case, GPU used 89.942% of the execution time. This means that GPU-FPM efficiently used the GPU computing power.

The rest of the paper is organized as follows: In section 2, the FPM, GPU, and OpenCL are described. The proposed GPU-FPM is introduced in section 3 and the experimental results are illustrated in section 4. Finally, the conclusions discussed in section 5.

II. PRELIMINARIES

A. Frequent Pattern Mining (FPM)

The main concept of FPM is to find the number of times a given pattern appears in a database. *FPM* is defined as follows:

Let D be a transactional database consisting of a set of transactions T_1, T_2, \dots, T_n : $D = \{T_1, T_2, \dots, T_n\}$. Let I be a set of items i_1, i_2, \dots, i_m , a set $X = \{i_1, i_2, \dots, i_k\} \subseteq I$ called an itemset or a k -itemset if it consists of k items. The support of an itemset X is the number of transactions containing X .

$$\text{support}(X, D) = |\{i \mid X \subseteq T_i, X \subseteq I\}| \text{ for } i = 1 \dots n$$

An itemset is called frequent if the support is greater than or equal to the given absolute minimal threshold ξ . FPM is given a set of items I , a database D , and a minimal threshold ξ , then find $\text{FP}(D, \xi)$.

$$\text{FP}(D, \xi) = \{X \subseteq I \mid \text{support}(X, D) \geq \xi\}$$

B. Graphic Processing Unit (GPU)

GPU is a parallel-oriented computing device. It always consists of massive processing units to perform mathematical computing. It used to be used as a co-processor CPU for games and 3D design applications. The DirectX 9 proposed in 2005, has taken graphics cards to the next generation because of vertex and pixel shaders being integrated in general-purpose processing units—introducing the universal shader. The mainstream GPU has hundreds to thousands computing units. Each unit can be regarded as a simplified CPU. Compared with the multicore CPU, the number of processing units has also increased. Consequently, GPU also has a whole new application—general-purpose computing on graphics processing units (GPGPU).

C. OpenCL

The GPU programming language can be classified as graphic APIs (DirectX, OpenGL, etc.), GPU programming language (NVIDIA CUDA [9], ATI Stream [10], OpenCL [11], etc). Previously, GPU programming required developers with in-depth knowledge of graphics programming and hardware. In order to utilize the computation resources on GPU, developers had to encode data to a graphic vector, and then use the DirectX or OpenGL functions to perform rendering. After that,

the rendered data had to be decoded. This procedure not only required graphic programming knowledge, but also depended on different GPUs. Recently, CUDA and Stream have been proposed by NVIDIA and ATI. Both of them provide C interface and allow developers to adapt the hardware, e.g., number of processing units, size of local and global memory. However, previous frameworks could only be used with the respective GPUs, e.g., CUDA could only be executed on NVIDIA's GPUs.

In order to solve this situation, the Khronos Group and many industry-leading companies created the OpenCL. OpenCL is an open and cross-platform parallel heterogeneous programming system. It provides a uniform programming environment for developers to write efficient and portable codes using a diverse mix of multi-core CPUs, GPUs, and other processors.

III. GPU-FPM

The goal of GPU-FPM was using massive processing units on GPU to speed-up the FPM procedures. However, each processing unit on GPU can only perform simple instructions. Another important issue is memory size and access latency. Therefore, the algorithm and data structure had to be re-designed for GPUs to fully utilize its computation resources. **Figure 1** illustrates the architecture of GPU-FPM. GPU-FPM has the following features: (1) data handling between CPU and GPU, (2) compact data structure, and (3) highly parallel.

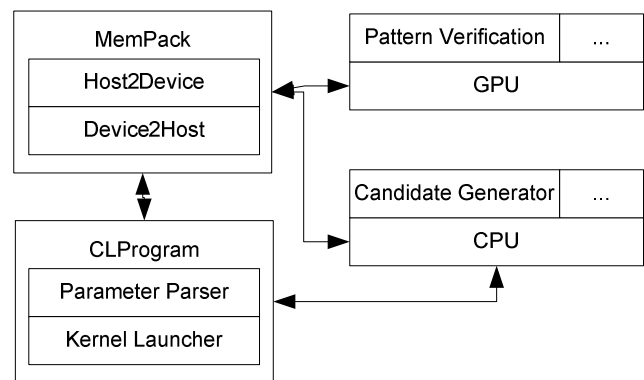


Figure 1. Architecture of GPU-FPM

A. Compact Data Structure

The memory access latency on GPU is very high, and it limits the computational speed-up ratio. Therefore, reducing the number of fetchings of memory improves the performance. It fetches the memory many times if used directly on a transaction-oriented database. This is because the entire transaction needs to be scanned for verifying each single itemset. Consequently, a transaction identification set (*Tidset*) was used to directly select transactions instead of scanning whole databases. *Tid* and *Tidset* were defined as follows:

$$\text{Tid}(i_j) = \{i_j \cap T_k \neq \emptyset\} \text{ for } k = 1 \dots n$$

$$\text{Tidset} = \{\text{Tid}(i_j)\} \text{ for } j = 1 \dots m$$

For example, if transactions 1 and 3 contain item i_1 , $\text{Tid}(i_1) = \{1, 3\}$, then a whole transaction-oriented database is represented by Tidset. In order to store the Tidset to memory on GPU, TidValue and TidIndex arrays were used to represent Tidset. **Figure 2** is an example of TidValue and TidIndex arrays. The TidValue array stored the Tid of each item, e.g., $\text{Tid}(i_1) = \{1, 3\}$, $\text{Tid}(i_2) = \{1, 2, 5\}$, $\text{Tid}(i_3) = \{2\}$, etc. (**Figure 2** (a)) The boundary of each item on the TidValue array was determined by the TidIndex array. The TidIndex stored each items start and end position, e.g., item i_4 ranging from 6 to 10 means six cells were used for i_4 in TidValue array and values were stored from $\text{TidValue}[6]$ to $\text{TidValue}[11]$. Therefore, the information required for mining was transformed from database to two arrays.

1	3	1	2	5	2	4	5	7	10	15	18	...
Tid(i_1)		Tid(i_2)			Tid(i_3)	Tid(i_4)						

(a) TidValue

0	1	2	4	5	5	6	11	...
i_1		i_2		i_3		i_4		

(b) TidIndex

Figure 2. Example of TidValue and TidIndex

B. GPU-FPM

Compared with CPU, GPU is special hardware with massive processing units. GPU processing is in single instruction, multiple data (SIMD) and there is no support recursion on it. Therefore, a compact data structure was designed and implemented to store necessary data for mining on GPU. The FPM could be roughly summarized to the following steps: load database, generate candidate itemset, and verify the candidate itemset frequently or not. Candidate itemset verification usually dominates computing time. Therefore, in this study, GPU was used to reduce candidate verification time.

GPU-FPM was an Apriori-based mining algorithm and it generated and verified the itemset to produce frequent patterns. Since memory access between CPU and GPU is a common operation, MemPack was designed to lower GPU programming complexities. MemPack is C++ class template that provided abilities to store different types of data, e.g., int, float, customized structure, class, etc. Two transfer functions: *Host2Device* and *Device2Host* and two memory control functions: *ReleaseHost* and *ReleaseDevice* were also provided. Moreover, the *CLProgram* class was also designed to have the

following abilities: allow arbitrary number of parameters, bind arbitrary of MemPack, launch with arbitrary number of threads, and launch with CPU. The GPU-FPM algorithm follows:

Algorithm GPU-FPM

Input: a transaction database D and a given minimum threshold ξ .

Output: a complete set of frequent patterns $\text{FP}(D, \xi)$.

1. Load D from disk.
 2. Generate Tidset via scanning the D and store it on hash table.
 3. Transform hash table to compact array structure—TidValue and TidIndex.
 4. Create MemPacks $mpTidValue$ and $mpTidIndex$ to store TidValue and TidIndex.
 5. Perform Host2Device to copy $mpTidValue$ and $mpTidIndex$ to GPU.
 6. Use prefix tree data structure to generated candidate itemset.
 7. Create MemPack $mpCandIS$ to store generated candidates.
 8. Perform Host2Device to copy $mpCandIS$ to GPU.
 9. Create MemPack $mpResults$ for storing results.
 10. Create CLProgram $clProg$ to store related parameters and bind the $mpTidValue$, $mpTidIndex$, $mpCandIS$, and $mpResults$.
 11. Perform launch kernel of $clProg$ (on GPU)
 - a. Each processing unit (PU) allocated a set of candidate itemsets (CIs)
 - b. for each CI in CIs
 - i. PU compute the support of CI according to $mpTidValue$ and $mpTidIndex$
 - ii. If support of CI greater than or equal to given threshold ξ then set it is frequent on $mpResults$, else is not frequent.
 12. Wait until kernel code executed.
 13. Perform Device2Host of $mpResults$ to store the results.
 14. Perform Step 6 until all candidates generated and verified.
-

IV. EXPERIMENTAL RESULTS

In order to evaluate the performance of the proposed algorithm, GPU-FPM was implemented along with OpenCL library and Visual C++ on Microsoft Windows. Synthesized datasets generated by IBM's Quest Synthetic Data Generator were used to verify the algorithm. The hardware and software configurations are given in **Table 1**. The algorithm evaluated with different transaction lengths, different threads, different block sizes, and different thresholds. **Table 2** gives the details of the dataset testing.

Table 1. Hardware and Software configuration

Item	Description
CPU	AMD Phenom II X4 965 3.4 GHz
Memory	8G DDR3 memory
GPU	ATI Radeon HD 5850 with 1440 stream processing units and 1G DDR5 memory
OS	Microsoft Windows 7
Compiler	Microsoft Visual C++ 2008 w/ SP1
SDK	ATI Stream SDK 2.0 w/ OpenCL 1.0 support

Table 2. Statistical Characteristic of Datasets

Dataset	Avg Trans Len	Avg Len of Max Pattern	No of Trans
T10I4D100K	10	4	100
T40I10D100K	40	10	100

A. Various Thread Numbers Quantities

In this section, two datasets with different threads and thresholds were used to verify the performance of GPU-FPM. **Figure 3** and **Figure 4** illustrate the computation time of various thresholds and threads. The computation time of the same threads was affected by the threshold. A smaller threshold refers to a smaller degree of support becoming a frequent pattern. There were 385 and 13,253 frequent itemsets with $\xi = 1000$ and $\xi = 200$, respectively. The speed-up ratio is depicted in **Figure 5** and **Figure 6**. For 16 times of threads, the best speed-up ratio was 13.066. Even in the worst case, it was 11.037. The average speed-up ratio was 12.576.

B. Various Block Sizes

In this section, GPU-FPM used different block sizes to verify the performance. The block size is the number of candidates that each processing unit on GPU deals with at each kernel launch. A small block size implied that the kernel had to be launched more times. **Figure 7** and **Figure 8** show the computation time of various block sizes. The *Var* stands for the block size changing with the number of threads, and the block size (*blockSize*) and number of threads (*noOfThread*) had the following relationship.

$$blockSize * noOfThread = 1024$$

Although a larger block size saved a bit on computation time (block size from 2 to 10, saving 0.826 second in case of T10I4D100K with 1024 threads), it only had a small influence on computation time. In some cases, larger block size even caused more computation time. The speed-up ratio is shown in **Figure 9** and **Figure 10**. The trend of the speed-up ratio could not be observed from the results. According to the experimental results, the block size had no significant effect on the computation time.

C. Computation Time used by CPU and GPU

Finally, the computation time used by CPU and GPU is depicted in **Figure 11** and **Figure 12**. GPU occupied most of the computation time in all cases. This means that the GPU-FPM

algorithm on GPU had the biggest workload. It also pointed out that pattern verification required much computing resources than pattern generation. For 1,024 threads, GPU occupied 93.837% computation time on average.

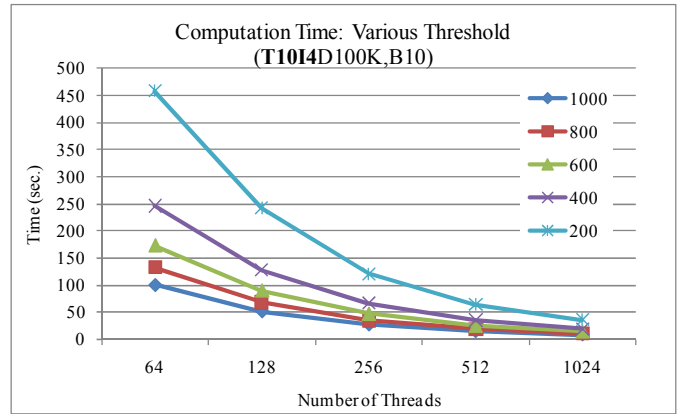


Figure 3. Computation Time of Various Thresholds (T10I4D100K, B10)

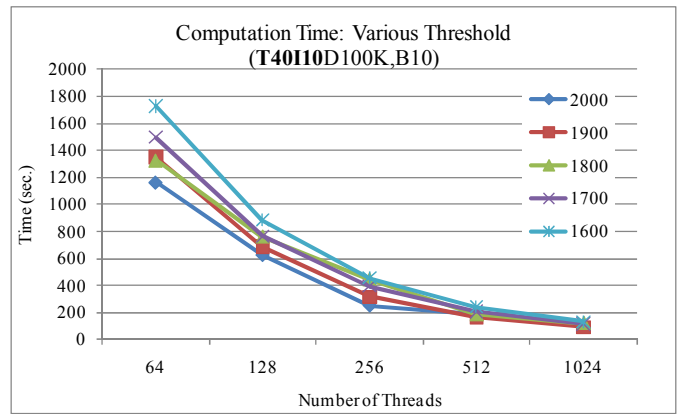


Figure 4. Computation Time of Various Thresholds (T40I10D100K, B10)

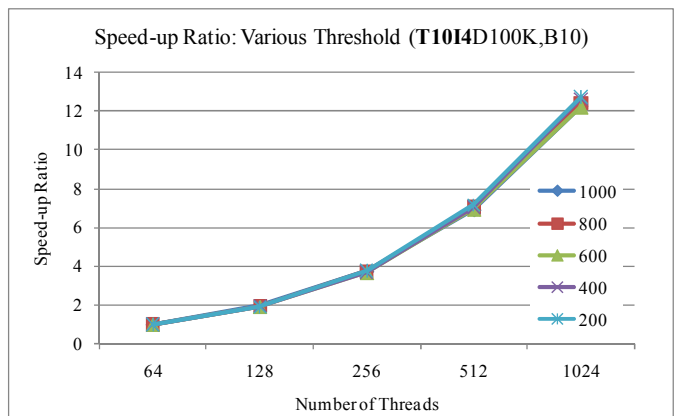


Figure 5. Speed-up Ratio of Various Thresholds (T10I4D100K, B10)

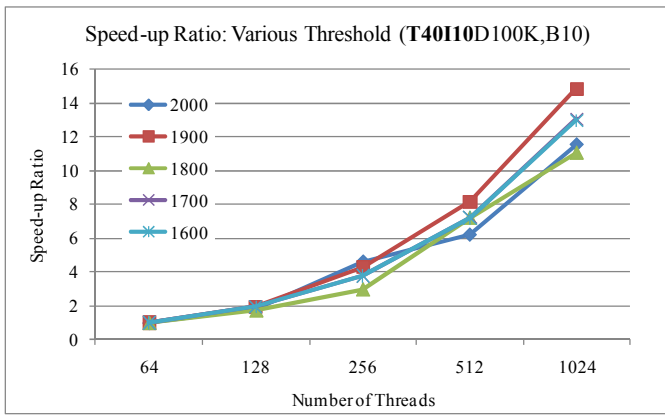


Figure 6. Speed-up Ratio of Various Thresholds (T40I10D100K, B10)

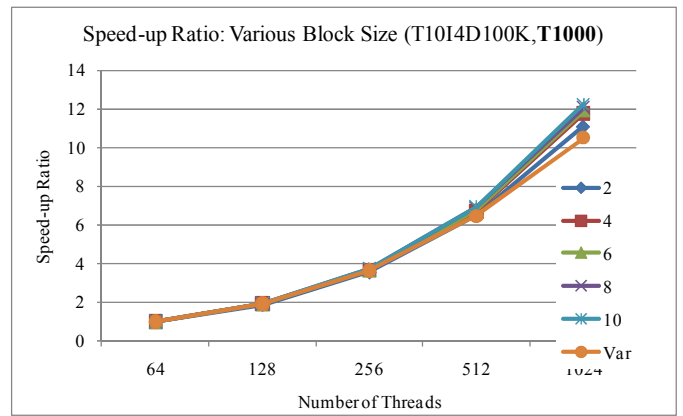


Figure 9. Speed-up Ratio of Various Block Sizes (T10I4D100K, T1000)

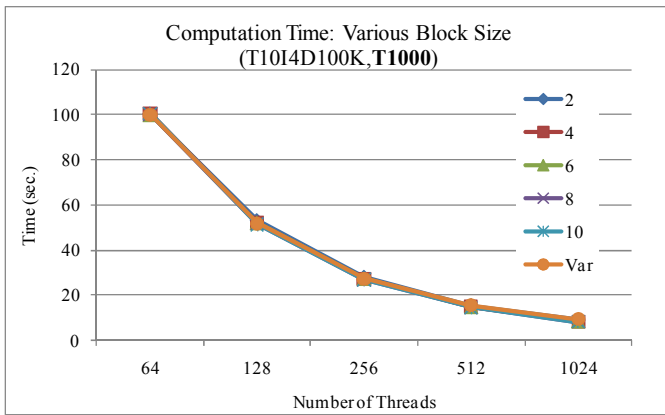


Figure 7. Computation Time of Various Block Sizes (T10I4D100K, T1000)

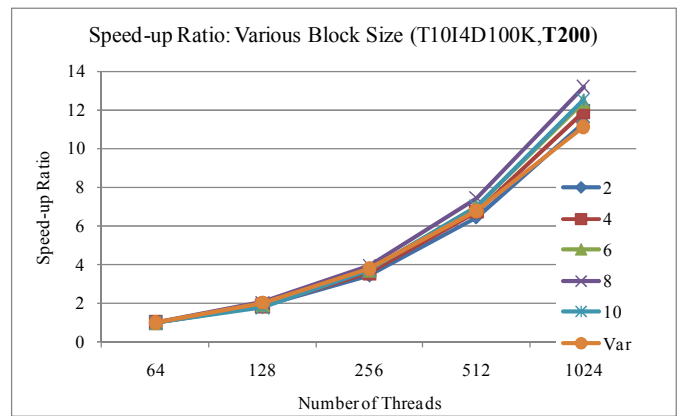


Figure 10. Speed-up Ratio of Various Block Sizes (T10I4D100K, T200)

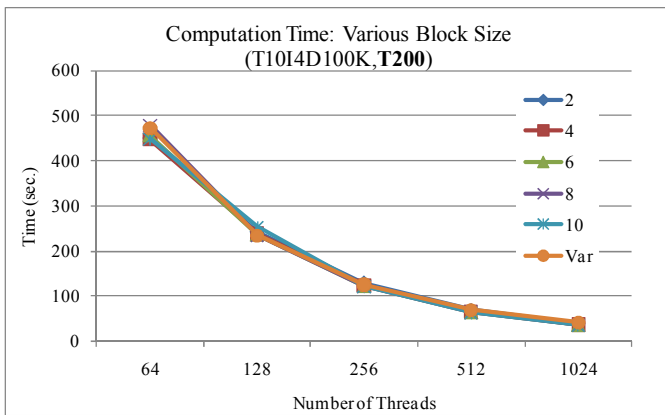


Figure 8. Computation Time of Various Block Sizes (T10I4D100K, T200)

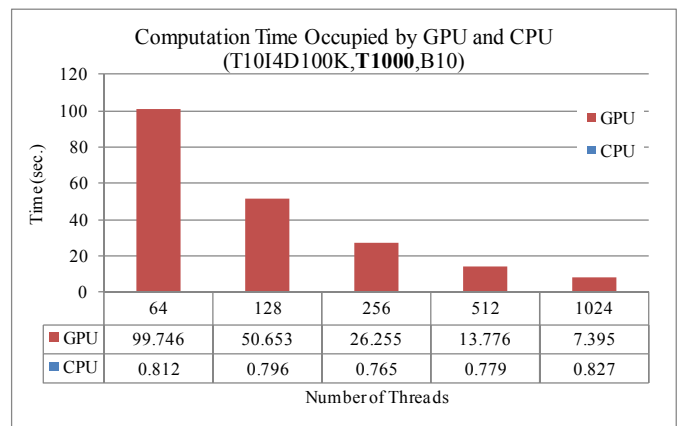


Figure 11. Computation Time Occupied by GPU and CPU (T10I4D100K, T1000, B10)

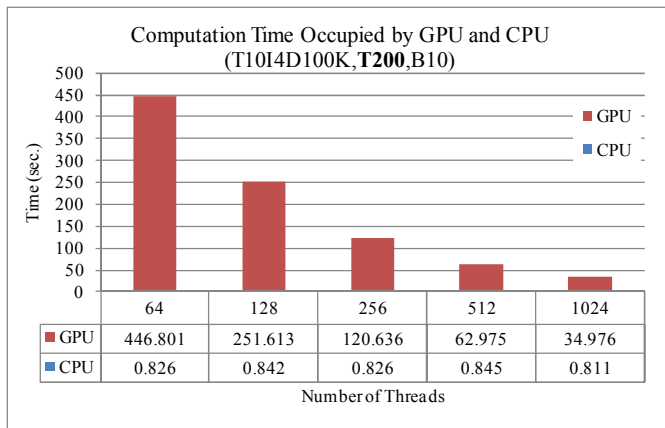


Figure 12. Computation Time Occupied by GPU and CPU (T10I4D100K, T200, B10)

V. CONCLUSIONS

Frequent pattern mining (FPM) is important and fundamental in data mining. Most FPM methods can be classified as Apriori-like or FP-growth-like. However, the computation time increased significantly when the number of transactions grew. In this study, a GPU based parallel algorithm—GPU-FPM was used to speed-up the mining processes. In order to conform to GPU hardware delimitation, a compact data structure was used to store entire database in GPU. Moreover, two template classes, MemPack and CLProgram were also used. Two datasets with different conditions were used to verify the performance of GPU-FPM. The speed-up ratio was 12.57 and 7.11 for 16 and 8 times of threads on average. In addition, most computation time was occupied by GPU because of all pattern verification processes being performed by it.

REFERENCES

[1] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," in International Conference on Very Large Data Bases, 1994, pp. 487-499.

[2] J. Han, J. Pei, Y. Yin *et al.*, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53-87, 2004.

[3] E. Lazcorreta, F. Botella, and A. Fernández-Caballero, "Towards personalized recommendation by two-step modified Apriori data mining algorithm," *Expert Systems with Applications*, vol. 35, no. 3, pp. 1422-1429, 2008.

[4] J. Park, M. Chen, and P. Yu, "An effective hash-based algorithm for mining association rules," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 175-186, 1995.

[5] A. Javed, and A. Khokhar, "Frequent pattern mining on message passing multiprocessor systems," *Distributed and Parallel Databases*, vol. 16, no. 3, pp. 321-334, 2004.

[6] K.-M. Yu, J. Zhou, T.-P. Hong *et al.*, "A Load-Balanced Distributed Parallel Mining Algorithm," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2486-2494, 2009.

[7] M. Chen, C. Huang, K. Chen *et al.*, "Aggregation of orders in distribution centers using data mining," *Expert Systems with Applications*, vol. 28, no. 3, pp. 453-460, 2005.

[8] D. Luebke, M. Harris, N. Govindaraju *et al.*, "GPGPU: general-purpose computation on graphics hardware." p. 208.

[9] NVIDIA. "Compute Unified Device Architecture (CUDA)," http://www.nvidia.com/object/cuda_home_new.html.

[10] ATI. "Stream SDK," <http://developer.amd.com/gpu/ATIStreamSDK/>.

[11] OpenCL. "OpenCL," <http://www.khronos.org/opencl/>.

[12] R. Agrawal, and R. Srikant, "Quest Synthetic Data Generator. IBM Almaden Research Center, San Jose, California," 2009.

行政院國家科學委員會補助國內專家學者出席國際學術會議報告

98 年 8 月 25 日

報告人姓名	游坤明	服務機構 及職稱	中華大學 資工系副教授
時間 會議地點	98/8/17 –98/8/19 Nanchang, China	本會核定 補助文號	
會議 名稱	(中文) 2009 IEEE 粒度計算國際研討會 (英文) 2009 IEEE International Conference on Granular Computing		
發表 論文 題目	(中文) 運用 MPI 與平行分支與界定技術進行化學化合物之推論 (英文) Parallel Branch-and Bound Approach with MPI Technology in Inferring Chemical Compounds with Path Frequency		

一、 參加會議經過

會議的開幕典禮由江西財經大學訊息管理學院院長致簡單的歡迎詞，並由會議的委員會主席說明本屆的投稿與錄取篇數 (投稿 345 篇，錄取 165 篇)及本屆會議的特色與會議重點後，隨即展開，本屆會議分別於第一天及第二天的會議議程中各安排了三個場次之粒度計算與生物資料處理暨應用領域之最新趨勢之專題報告: (1). Generalized Bags and Their Relations: An Alternative Model for Fuzzy Set Theory and Applications, (2). Formal Theory of Granular Computing and Two Major Applications, (3). High-throughput DNA Sequencing and Bioinformatics: Bottlenecks and Opportunities, (4). Ubiquitous/Pervasive Intelligence: Visions and Challenges, (5). Ubiquitous Personalized Information Processing with Wildcard 以及 (6). Computational Models in Systems Biology，分別由 Sadaaki Miyamoto (University of Tsukuba, Japan)、T. Y. Lin (San Jose State University, USA)、Stephen Kwok-Wing TSUI (The Chinese University of Hong Kong,China) Laurence T. Yang(St Francis Xavier University, Canada)、Xindong Wu (The University of Vermont, USA) 以及 Xue-wen Chen (The University of Kansas,USA) 作精彩的專題報告。正式之論文報告分別於第三個場次的專題報告後進行，本人之論文『Parallel Branch-and Bound Approach with MPI Technology in Inferring Chemical Compounds with Path Frequency』被安排在第一天的 – Session C3 “Intelligent Data Analysis and Applications”之場次發表。

二、 與會心得

GrC 2009 是一個在粒度計算(Granular computing)研究領域中具有指標性的最重要國際會議，此次會議共有三百四十多篇的論文投稿，但僅錄取了一百六十餘篇優秀的粒度計算資訊系統領域的論文，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，參與此次會議之篇學者可藉著此次會議，在粒度計算資訊系統與生物資訊運用之各研究領域互相作深入的討論，而且可藉此機會認識在此領域中之權威研究學者，對於此後從事此領域之研究有相當之助益。

三、 考察參觀活動(無是項活動者省略)

無。

四、 建議

無。

五、攜回資料名稱及內容

1. 大會議程
2. The Proceeding of 2009 IEEE International Conference on Granular Computing 研討會論文集。

六、 其他

無衍生研發成果推廣資料

98 年度專題研究計畫研究成果彙整表

計畫主持人： 游坤明		計畫編號： 98-2221-E-216-023-					
計畫名稱： 應用多核心與格網技術設計同特徵化合物平行化演算法及其 Web Service 之研發							
成果項目		量化			單位	備註(質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等)	
		實際已達成數(被接受或已發表)	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比			
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力 (本國籍)	碩士生	3	3	100%	人次	
		博士生	1	1	100%		
博士後研究員		0	0	100%			
專任助理		1	1	100%			
國外	論文著作	期刊論文	0	1	100%	篇	
		研究報告/技術報告	0	0	100%		

						<p>1. Jiayi Zhou, Kun-Ming Yu, Chun Yuan Lin, Kuei-Chung Shih1 and Chuan Yi Tang, 'Balanced Multi-process Parallel Algorithm for Chemical Compound Inference with Given Path Frequencies,' The 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010) - Lecture Notes in Computer Science, Vol. 6082, Part II, pp. 188-197, Springer-Verlag, May, 2010. (Busan, Korea, 5/21-23, 2010). (EI) (NSC98-2221-E-216-023 and NSC97-2221-E-216-020)</p> <p>2. Jiayi Zhou, Kun-Ming Yu* and Bin-Chang Wu, 'Parallel Frequent Patterns Mining Algorithm on GPU,' 2010 IEEE International Conference on Systems, Man, and Cybernetics (SMC2010), pp. 435-440, 2010 (Istanbul, Turkey, Oct. 10 - 13, 2010). (EI). (NSC 98-2221-E-216-023), *corresponding author.</p>
		研討會論文	2	1	100%	
		專書	0	0	100%	章/本
專利		申請中件數	0	0	100%	件
		已獲得件數	0	0	100%	
技術移轉		件數	0	0	100%	件
		權利金	0	0	100%	千元

參與計畫人力 (外國籍)	碩士生	0	0	100%	人次	
	博士生	0	0	100%		
	博士後研究員	0	0	100%		
	專任助理	0	0	100%		

其他成果 (無法以量化表達之 成果如辦理學術活 動、獲得獎項、重要 國際合作、研究成果 國際影響力及其他 協助產業技術發展 之具體效益事項 等，請以文字敘述填 列。)	無					
----------------------------------------------------------------------------------------------------------------	---	--	--	--	--	--

	成果項目	量化	名稱或內容性質簡述
科 教 處 計 畫 加 填 項 目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與(閱聽)人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

達成目標

未達成目標（請說明，以 100 字為限）

實驗失敗

因故實驗中斷

其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文： 已發表 未發表之文稿 撰寫中 無

專利： 已獲得 申請中 無

技轉： 已技轉 洽談中 無

其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

本計畫提出並設計於多核心(Multi-core)的平行化演化樹建構演算法，可以列出在 input space 中找到最接近的近似解，可以應用該特性讓生物學家調整在 feature space 中的值，並找出與該似接近之化合物；同時在本計畫得執行過程中，我們亦提出一個 GPU 平行化資料探勘演算法，有效加快資料探勘時間。本計畫亦將設計完成的演算法包裝成 web service 元件，使得國內外有興趣的相關團隊利用多核心計算資源快速解決 MUT 及 CCI 課題。

本計畫執行所獲得的成果將對 MUT、CCI 及資料探勘等問題，或是對於多核心計算領域之研究學者有相當大的助益。