Algorithms for Irregular Data Redistribution Problems

不規則資料重新分配排程問題之演算法設計

Fu Home Liu (劉富翃)、Chang Wu Yu (俞征武)
中華大學資訊工程系
cwyu@chu.edu.tw

摘要—不規則資料重新分配問題在平行處理的環境下是一個基本且重要的問題。此問題之最佳化排程可視為一種二分平面圖(biplanar graphs)之新的邊塗色問題(the chromatic max-edge-coloring problem)。此圖論問題已經被證明其為 NP-complete。我們將為此問題提出一個找出最佳解的演算法及設計一個快速的貪婪演算法(greedy algorithm)。除此之外，有些情況會希望傳輸的步驟(次數)愈少愈好，所以，在最少的傳輸次數下且成本最小的問題我們也會一併討論。我們並且設計一個簡單有效的貪婪法並與最佳解作比較。根據我們的實驗顯示此貪婪法可以有 6 成以上的機率找到最佳解及最差的狀況其成本低於最佳解的 1.64 倍。

關鍵詞—資料重新分配問題、邊塗色問題

一、問題及背景

平行處理已經被廣泛且有效地運用在科學問題的解決上。在有大量資料的情況下，平行處理可以減低每個處理器的負荷量，並提升效能。一般而言，矩陣的資料分配於各處理器中的方法可以分成規則及不規則兩種：規則資料分配通常用 BLOCK-CYCLIC(c) 來做陣列的分解 [14, 15]。而不規則資料分配則沒有特定的方法，所以分解不平均的陣列時，大都用個別的方法處理。例如，High Performance Fortran version 2 (HPF2) 有提供一種陣列資料分配的機制，稱為 Generalized Block Distribution (GEN_BLOCK) [19, 20]，其可以將分成數個不同大小區段的陣列對應到另一個區段分法不同的陣列，如下圖所示：

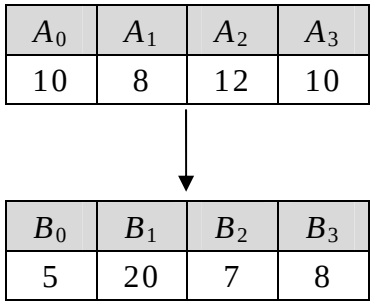

圖 1、陣列資料重新分配

一般而言，系統於適當地時機執行資料之重新分配有助於提昇整體系統的效率。而不規則資料分配，如 GEN_BLOCK 所提供之功能，可根據每台處理器所能負擔的計算能力分配適當的資料量。若我們將處理器看作是點(vertex)，處理器之間的資料分配(傳遞)關係看作是線(edge)，這樣就會成為一個二分圖的問題。圖 2 即為

圖 1 轉成的圖問題的結果。

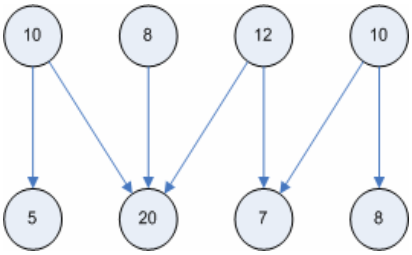

圖 2、資料重新分配用圖表示

圖 2 描述了處理器之間的資料傳輸方法，我們假定在這個分散式系統中，每個處理器一次只能送資料給一台處理器且只能從一台處理器收資料；如此，這個資料傳送之排程問題便成為塗色問題中的塗邊問題(edge coloring problem)。圖 3 為圖 2 問題的一種排程。

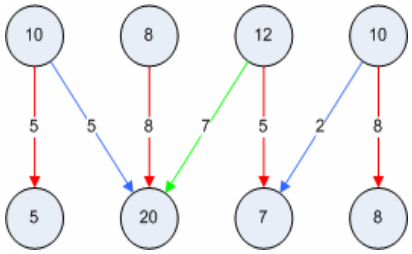

圖 3、用塗色表示資料分配

在圖 3 中，相同顏色的線代表可以同時傳送的資料，傳送的資料量愈大。所花的成本（時間）也愈多。因為我們考慮平行處理系統為同步處理。故每個傳送步驟的成本為所有可同時傳送（顏色相同）的資料成本之最大者。表一列出此排程的整體成本。

表 1、圖三排程的成本計算

次數	1 (red)	2 (blue)	3 (green)	Total
成本	8	5	7	20

對於不規則資料重新分配的排程問題，前人設計出一些 heuristic 演算法，但都不是最佳解。在[28]，Yu 將此問題定義成*二分平面圖*(biplanar graphs)上的一個新的圖論問題，即 *the max-edge-coloring* problem (MECP)，此問題目的在於為上述的有權重(weighted graph)的通訊圖(即一個二分平面圖)$G=(V, E)$找到一個邊塗色(edge colorings)$\{E_1, E_2, E_3, \ldots, E_z\}$使得 $\sum_{i=1}^{z} \max \{w(e_k) | e_k \in E_i\}$ 值最小，這裡的 $w(e_k)$是 e_k 的權重。在[28]中，Yu 並提出一個低於最佳解 2 倍的逼近演算法。

本篇論文是希望利用較有效率的暴力法來找到最少成本的解。除此之外，有些情況會希望傳輸的步驟(次數)愈少愈好，所以，在最少的傳輸次數下且成本最小的問題我們也會一併討論。我們並且設計一個簡單有效的貪婪法並與最佳解作比較。根據我們的實驗顯示我們的貪婪法可以有 6 成以上的機率找到最佳解及最差的狀況其成本低於最佳解的 1.64 倍。

二、前人成果

規則陣列重新分配技術可以分成兩部分：Communication Sets Identification 和 Communication Optimization。前者包括 PITFALLS [17] 與 ScaLAPACK [16] 的索引集合產生方法、Park et al. [14] 想出的演算法用在處理器之間 BLOCK-CYCLIC 資料重新分配之時。Dongarra et al. [15] 提出 BLOCK-CYCLIC 分解的重新分配演算法。Zapata *et al.* [1] 用 CRS 結構為 BLOCK-CYCLIC 資料重新分配設計平行稀疏重新分配碼。同樣地，Generalized Basic-Cycle Calculation 在[3]有描述。Communication Optimization 技術提供不同方面的方法來減少多餘的溝通在重新

分配的運作上，限制傳輸多餘資料的處理器對應技術[10, 12, 4]就是其例子，減少訊息起始成本的多方重新分配策略[11]、避免節點問題的溝通排程方法[2, 7, 13, 21]，以及重疊溝通與多餘計算的Strip Mining Approach [18]。

在不規則陣列重新分配方面，之前的文獻著重在索引和訊息的產生或溝通效率。Guo et al. [9] 提出一個溝通集合產生和減少溝通成本的符號分析方法。針對減少通訊成本來說，Lee et al.[12]提出四個邏輯處理器描繪演算法在不規則陣列重新分配上。

Guo *et al.*[19, 20]用一個各個擊破的演算法來解不規則陣列重新分配。Neighbor Message Set (NMS) 演算法將相同發送端（或接收端）的溝通訊息分成幾個群組，在根據特定狀態合併 NMS 之後再排出最後的答案。

在[21]裡，Yook 和 Park 提出一個改變位置的演算法，這個演算法由兩個階段所組成，分別為列表排程階段與改變位置階段，第一個階段以降冪排序全域訊息然後配置它們到溝通排程中，當有衝突發生時，改變位置階段就會進行一系列的重新排程運作，但是這個階段會導致多餘的計算並降低演算法的效能。

三、暴力法之設計

首先設計一個有效率的演算法來找到最佳解。此法找出最佳解後，並可以和將來所設計的貪婪演算法做比較。

因為每個處理器在同一步中只能傳送或接收一個來自其它處理器的資料，所以排程所需最小步數（最少所需顏色）為圖的 max degree。假設有 n 條線，m 個顏色，總共就有 m^n 種顏色組合，但是共點的線須不同色，所以存在共點的線相同顏色的顏色組合就不是合理的排程方法，也沒有計算的必要。為了避開上述的顏色組合並產生其餘所有顏色組合以找出最佳解，我們設計以下演算法。

以下用一個例子說明我們提出的改進後的暴力演算法。

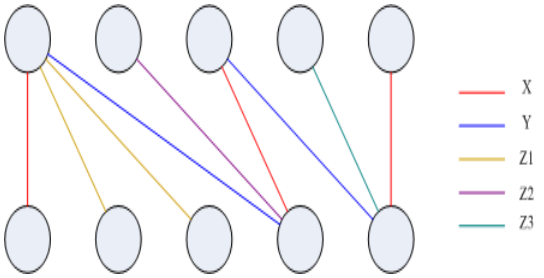

圖 4、找出 X, Y, Z 集合

如圖四，首先找出 X, Y 及所有 Z 集合，其次初始每條線的顏色。X 集合中的線不受限制，可任意選擇顏色；Y 集合的線因 X 已經選擇藍色，必須選其它顏色；而各個 Z 集合的線必須考量到 X, Y 與在同一個 Z_i 中的線的顏色再選擇顏色，如下圖所示：

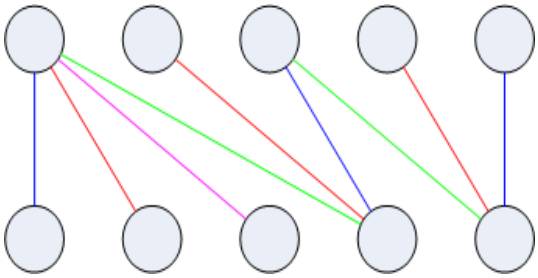

圖 5、顏色的初始

要換到下一組顏色組合時，先換各個 Z 集合，若 Z 集合都換完，就換 Y 集合，若 Y 集合換完，再換 X 集合，若連 X 集合都換完，就結束演算法。

因為此圖 max degree 為四，最少需用四個顏色，而 Z 集合受限於 X 及 Y，所以

只能用兩個顏色，因此，Z_1, Z_2 及 Z_3 分別有兩種顏色組合，整個 Z 就有 2×2×2=8 種顏色組合。若與 Y 集合的線共點的 X 集合的線的顏色不同，則此 Y 集合的線有兩種顏色組合；若相同，則有三種顏色組合。X 集合則是有 $4^3 = 64$ 種顏色組合。

演算法 1、暴力法

定義 E 為圖(graph)中的所有線(edge)的集合
1 將 E 中的所有線照圖中的位置由左自右排序
2 找出 E 中的骨幹 B (backbone)線集合。即 B={所有兩個端點 degree 皆大於一的邊}∪{最左與最右的兩條邊}。
3 為 B 中的線編號 $e_0, e_1, e_2, \ldots e_k$。
4 令線集合 X 為{$e_i \in B | i \in$ 偶數}。
5 令線集合 Y 為{$e_i \in$ B$| i \in$ 奇數}。
6 令線集合 F 為 E-B。
7 令線集合 Z_i 為 F 中第 i 個連通元件(connected component)的線集合。
8 令整數 *minCost* 的值為無限大。
9 初始 X 中的線的顏色。
10 初始 Y 中的線的顏色，使得圖不會產生抵觸(conflict)。
11 對於所有的 Z_i，利用相鄰 X 或 Y 的線所沒有用到的顏色來做初始。
12 對於所有的 Z_i。
  12.1 當 Z_i 能夠且換到下一組線的顏色組合時，跳至步驟 21。
13 當 Y 沒有下一組線的顏色組合時，跳至步驟 17。
14 將 Y 換到下一組線的顏色組合。
15 對於所有的 Z_i，利用相鄰 X 或 Y 的線所沒有用到的顏色來做初始（同步驟 11）。
16 跳至步驟 21。
17 當 X 沒有下一組線的顏色組合時，跳至步驟 24。
18 將 X 換到下一組線的顏色組合。
19 初始 Y 中的線的顏色，使得圖不會產生抵觸（同步驟 10）。
20 對於所有的 Z_i，利用相鄰 X 或 Y 的線所沒有用到的顏色來做初始（同步驟 11）。
21 令整數 *cost* 為現在的圖所算出的成本。
22 讓 *minCost* 為 min{*cost*, *minCost*}
23 跳至步驟 12。
24 演算法結束，*minCost* 為最小的成本。

使用這個演算法可大幅降低尋找最佳解所需時間。對於一個上下各七個點的圖且 max degree 為四，使用將所有顏色組合都計算過的方法來尋找最佳解需要一分鐘，但在同一台電腦上執行此演算法只需不到一秒即可找到最佳解。

四、貪婪法之設計

貪婪法在解決一些問題時有不錯的表現，有時可以獲得最佳解或近似解，我們試著在這個問題上設計一個貪婪法的演算法。

此排程問題之成本計算為將每個步驟的成本相加所得，每個步驟的成本為同步驟（相同顏色的線）中資料傳輸量最大者，因此總成本的大小只會與這些步驟中傳輸量最大者有關。這些傳輸量最大者可以看成是在「保護」同步驟中其餘的資料傳輸，使得它們不會被突顯出來而影響到總成本的大小。在所有的資料傳輸中，會有一個（或多個）最大的傳輸量，這些資

料傳輸不可必免的要保護其它資料傳輸，所以盡可能讓所有資料傳輸都被最大資料傳輸所保護以達到降低成本的目標是很直覺的方法，因此我們設計以下演算法。下圖是一個用圖表示處理器與資料傳輸關係的例子，我們準備利用上述貪婪法排程。

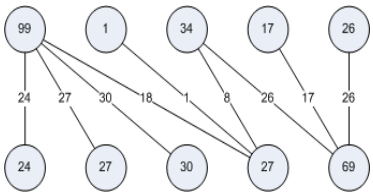

圖 6、準備排程的圖

首先以線的傳輸量由大到小排序，結果會是 30, 27, 26, 26, 24, 18, 17, 8, 1，再從最大的資料傳輸開始加入第一個步驟，如果發生抵獨（共點）就不要將線加入步驟，並繼續判斷剩下的線。圖七右邊為在第一步會做傳輸的線，左邊為還未排程的線。

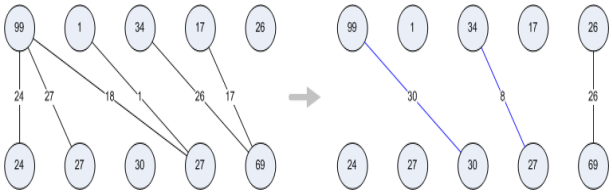

圖 7、排程的第一步驟

接著在剩下的線中繼續選擇第二步驟的排程。

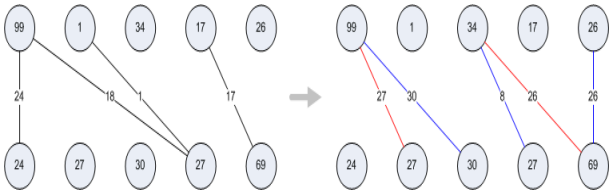

圖 8、排程的第二步驟

當左邊的圖的 max degree 為二時，就要進行另一個處理。此時有兩個連通元件。找出連通元件中的兩個獨立集合，以第一個連通元件為例就是 {24, 1} 及 {18}，從傳輸量較大者開始尋找可以加入的步驟，結果 {24, 1} 被分到第三步、{18} 被分到第四步，之後再對下個連通元件做相同的處理。最後的結果如下圖所示，總成本是九十九，與最佳解相同。

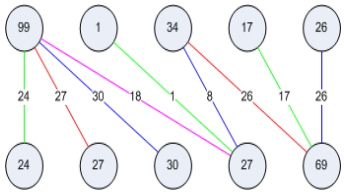

圖 9、貪婪法排程的結果

演算法 2、貪婪法

定義 E 為圖(graph)中所有線(edge)的集合
1 令整數 s 為 0。
2 對 E 中的線依照重量(weight)由大至小排序。
3 對於 E 中每一條線 $e \in E$（重量由大至少）。
  3.1 如果圖中有一個點(vertex)的度數(degree)大於二，跳至步驟 3.4。
  3.2 令 C_i 為只剩 E 的圖的第 i 個連通元件(connected component)。
  3.3 對於每個 C_i。
    3.3.1找出 C_i 中的兩個獨立集合(independent set) p_0, p_1。
    3.3.2如果 p_0 的成本小於 p_1 的成本，交換 p_0 與 p_1。
    3.3.3將 p_0 加入依成本由大到小排好的第一個步驟中，使得圖不會產生抵觸(conflict)。
    3.3.4將 p_1 加入依成本由大到小排好的第一個步驟中，使得圖不

會產生抵觸。

3.3.5結束演算法。

3.4 將 e 放入第 s 個步驟中。

3.5 如果發生抵觸，取出 e 並將之放回 E 中。

4 $s=s+1$。

5 跳至步驟 3。

五、貪婪法的效能分析

下表為貪婪法在不同點數、最大傳輸量、比較次數、正確率與最大倍率的統計結果。點數 5 代表圖上下各有 5 點；最大傳輸量為亂數產生圖時傳輸量的亂數上限。

表 2、貪婪法的效能

點數	最大傳輸量	比較次數	正確率	最大倍率
5	30	1000	65.4%	1.55
6	30	1000	63.9%	1.6
5	100	1000	66.5%	1.59
6	100	1000	62.6%	1.64

雖然正確率不高，但貪婪法所計算出的解與最佳解的差距不大，整體而言是一個不差的演算法。

六、結論

不規則資料重新分配問題之最佳化排程可視為一種二分平面圖之新的邊塗色問題，我們為此問題提出一個找出最佳解的演算法及設計一個快速的貪婪演算法，並觀察到此貪婪法所得到的解與最佳解的差距不大。未來可試著找出此貪婪法的解與最佳解的最大倍率上限，並證明它是一個近似解，或提升貪婪法的正確率，以提升其效率，並將其成果運用於平行系統中不規則資料重新分配之排程設計上。

參考文獻

[1] G. Bandera and E.L. Zapata, "Sparse Matrix Block-Cyclic Redistribution." *Proceding of IEEE Int'l. Parallel Processing Symposium (IPPS99*), San Juan, Puerto Rico, April 1999.

[2] 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.

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

[4] 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.

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

[6] 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.

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

[8] Minyi Guo, I. Nakata, and Y. Yamashita, "An Efficient Data Distribution Technique for Distributed Memory Parallel Computers," *JSPP97*, pp. 189-196, 1997.

[9] 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.

[10] Edgar T. Kalns and Lionel M. Ni, "Processor Mapping Technique Toward Efficient Data Redistribution," *IEEE Trans. on PADS*, vol. 6, no. 12, December 1995.

[11] S. D. Kaushik, C. H. Huang, J. Ramanujam, and P. Sadayappan, "Multiphase data redistribution; Modeling and evaluation." *Proceeding of IPPS"95*, pp. 441-445, 1995.

[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*, 2001.

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

[14] Neungsoo Park, Viktor K. Prassanno, 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.

[15] Antoine P. Petitet and Jack J. Dongerra, " Algorithmic Redistribution Methods for Block -Cyclic Decompositions," *IEEE Trans. on PDS*, vol. 10, no. 12, pp. 1201-1216, Dec. 1999.

[16] 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.

[17] 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.

[18] Akiyoshi Wakatani and Michael Wolfe, "Optimization of Data redistribution for Distributed Memory Multicomputers." Parallel Computing, vol. 21, no.9, pp. 1485-1490,

September 1995.

[19] Hui Wang, Minyi Guo, and Daming Wei, "Divide-andconquer Algorithm for Irregular Redistribution in Parallelizing Compilers", *The Journal of Supercomputing*, vol. 29, no.2, 2004.

[20] Hui Wang, Minyi Guo, and Wenxi Chen, "An Efficient Algorithm for Irregular Redistribution in Parallelizing Compilers," *Proceddings of 2003 International Symposium on Parallel and Distributed Processing with Applications*, LNCS 2745, 2003.

[21] H.-G. Yook and Myung-Soon Park, "Scheduling GEN_CLOCK array Redistribution," *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems*, November, 1999.

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

[23] R. Cole and J. Hoperoft. "On edge-coloring bipartie graphs," *SIAM J. Comput.* vol. 11, pp. 540-546, 1982.

[24] C. W. Yu and G. H. Chen, "Efficient parallel algorithms for doubly convex-bipartite graphs." *Theoretical Computer Science*, vol. 147, pp. 249-265, 1995.

[25] P. Eades, B. D. McKay, and N. C. Wormald. "On an edge crossing problems," *Proc. 9th Australian Computer Science Conference*, Australian National University, 1986, pp. 327-334.

[26] N. Tomil, Y. Kambayashi, and Y. Shuzo. "On planarization algorithms of 2-level graphs," *Papers of tech. group on electronic computers*, IECEJ, EC77-38, pp. 1-12, 1977.

[27] C. W. Yu, “On the complexity of the maximum biplanar subgraph problem,” *Information Science*, vol. 129, pp. 239-250, 2000.

[28] C. W. Yu, ‘On the Complexity of the Max-Edge-Coloring Problem with Its Variants,’ *Lecture Notes in Computer Science*, vol. 4614, pp. 362-374, 2007.