数理解析研究所講究録 1335 巻 2003 年 204-210

ParGAPによるアソシエーションスキームの並列バックトラック計算

宮本　泉

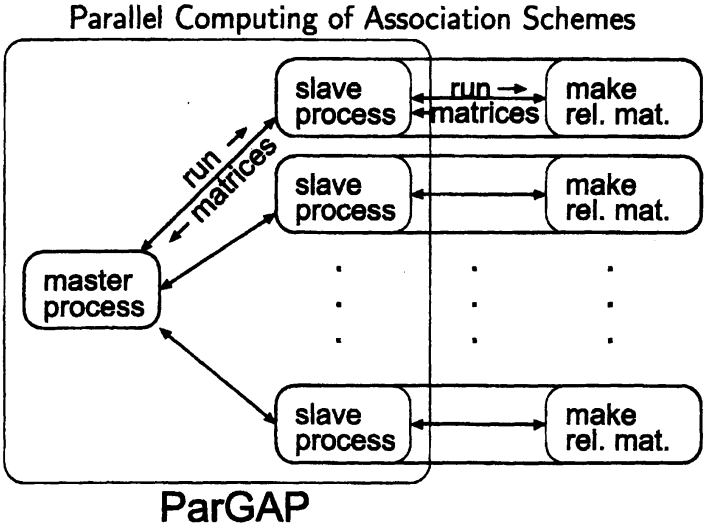
IZUMI MIYAMOTO

山梨大学　コンピュータ・メディア工学科

DEPARTMENT OF COMPUTER SCIENCE UNIVERSITY OF YAMANASHI*

1 Introduction

並列計算は、群を中心とする代数的数式処理ソフトウェア GAP のシェアパッケージ ParGAP をインタフェースとして使用して、アソシエーションスキームを構成する C-プログラムを並列に動かして行います。



アソシエーションスキームを構成する最新プログラムは花木さんの作成で、本実験ではそれを改造して使用しています。使用する計算機は、約７０台、PentiumIII、800MHz、メモリー 256MB です。User にはメモリー 100MB の使用制限があります。ネットワークは 100base、AnyLAN です。

2 Association schemes

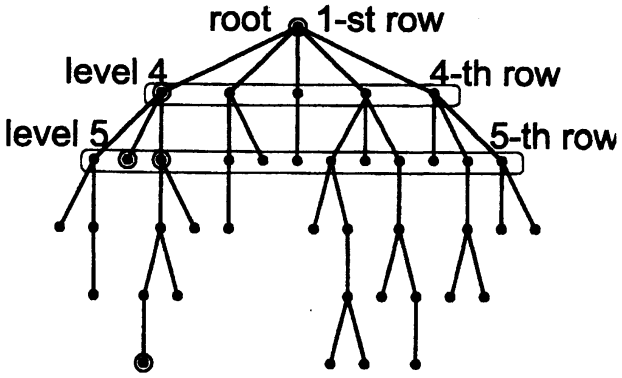
アソシエーションスキームの定義 : A set of $(d+1)$ matrices $A_0, A_1, \cdots, A_d$ of size $n \times n$ is an *association scheme* of order n if itsatisfies the following.

*izumi@esi.yamanashi.ac.jp

1. A_i {0,1}-matrix, A_0 =*identity matrix*
2. $A_0 + A_1 + A_2 + \cdots + A_d = J$(*all one matrix*)
3. for any i, ${}^tA_i = A_{i^*}$ holds for some $i*$
4. $A_iA_j = \Sigma_{k=0}^{d} p_{i,j,k}A_k$

The relation matrix A of the association scheme is defined by $A = 0A_0 + 1A_1 + 2A_2 + \cdots + dA_d$.

A searching tree of a backtrack method



relation matrix A を使って計算する。

例：input [[1, 1, 1, 2, 2], [1, 3, 2, 4, 5]]

the root

$$A = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 3 & & 0 & \cdot & \cdot & \cdots & & \\ 2 & & & 0 & & & & \\ 4 & & & & 0 & & & \\ 4 & & & & & 0 & & \\ 5 & & & & & & 0 & \\ 5 & & & & & & & 0 \end{pmatrix},$$

a node at the deepest level=an association scheme

$$A = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & 3 & 2 & 4 & 4 & 5 & 5 \\ 3 & 2 & 0 & 1 & 5 & 5 & 4 & 4 \\ 2 & 3 & 1 & 0 & 5 & 5 & 4 & 4 \\ 4 & 4 & 5 & 5 & 0 & 1 & 2 & 3 \\ 4 & 4 & 5 & 5 & 1 & 0 & 3 & 2 \\ 5 & 5 & 4 & 4 & 3 & 2 & 0 & 1 \\ 5 & 5 & 4 & 4 & 2 & 3 & 1 & 0 \end{pmatrix}$$

a node at level 4 (4行目まで完成した relation matirx)

$$A = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & 3 & 2 & 4 & 4 & 5 & 5 \\ 3 & 2 & 0 & 1 & 5 & 5 & 4 & 4 \\ 2 & 3 & 1 & 0 & 5 & 5 & 4 & 4 \\ 4 & 4 & 5 & 5 & 0 & & & \\ 4 & 4 & 5 & 5 & & 0 & & \\ 5 & 5 & 4 & 4 & & & 0 & \\ 5 & 5 & 4 & 4 & & & & 0 \end{pmatrix}$$

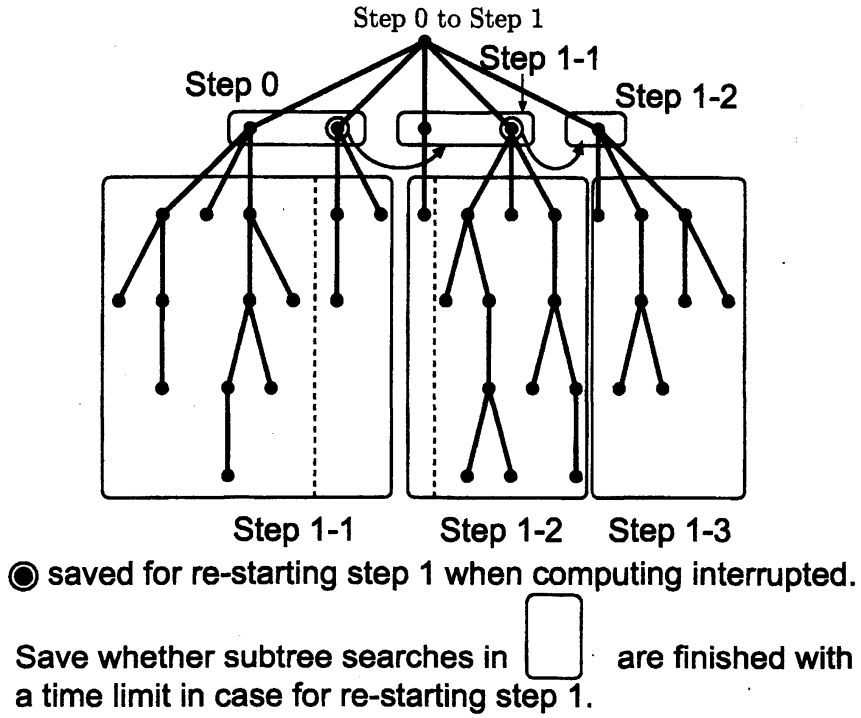
nodes at level 5 (children of the above node)

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & 3 & 2 & 4 & 4 & 5 & 5 \\ 3 & 2 & 0 & 1 & 5 & 5 & 4 & 4 \\ 2 & 3 & 1 & 0 & 5 & 5 & 4 & 4 \\ 4 & 4 & 5 & 5 & 0 & 1 & 2 & 3 \\ 4 & 4 & 5 & 5 & 1 & 0 & & \\ 5 & 5 & 4 & 4 & 3 & & 0 & \\ 5 & 5 & 4 & 4 & 2 & & & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & 3 & 2 & 4 & 4 & 5 & 5 \\ 3 & 2 & 0 & 1 & 5 & 5 & 4 & 4 \\ 2 & 3 & 1 & 0 & 5 & 5 & 4 & 4 \\ 4 & 4 & 5 & 5 & 0 & 1 & 3 & 2 \\ 4 & 4 & 5 & 5 & 1 & 0 & & \\ 5 & 5 & 4 & 4 & 2 & & 0 & \\ 5 & 5 & 4 & 4 & 3 & & & 0 \end{pmatrix}$$

５行目の最後の３成分（と、それによって定まる５列目の最後の３成分）のみが異なる。

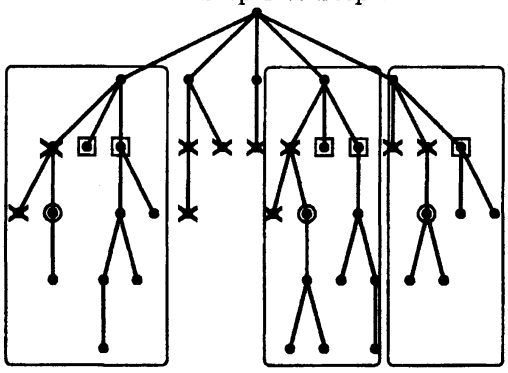
3 並列計算

Our parallel computing:(Step 0) → Step 1 → Step 2 → ···



図の様に、例えば、relation matrix の５行目まで完成したもののうちの一部分を求める。以下、図に示す手順での５行目まで完成したの relation matrix を全て求める一方で、得られた relation matrix のそれぞれに対して、例えば、30 秒の時間制限を設けて、その先の可能な relation matrix を並列に計算する。この間に、完成したと未完成のものが得られる。未完成のものが次の step への input になる。

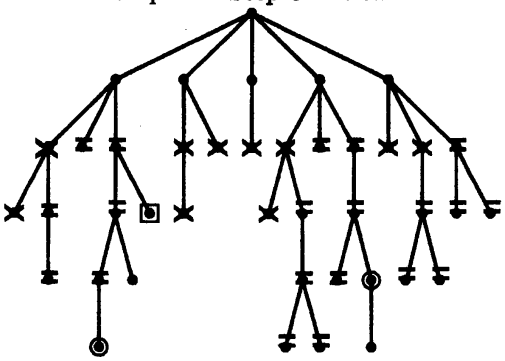
Step 1 to Step 2



searched in step 1
remained and saved in step 1
→distributed to generate ◙ in step 2
Distribute ◉ & ◙ for subtree searchs in step 2
within a ceratin time limit.
Save whether subtree searchs in ▢ are finished
with a time limit in case for re-starting step 2.

入力から、例えば 6 行目まで完成した relation matrix で、まだ生成されていないものを全て求めて、得られた行列に対して step 1 と同様に、時間制限付で、それから先の可能な relation matrix を計算する。得られた結果の処理は step 1 と同様にする。

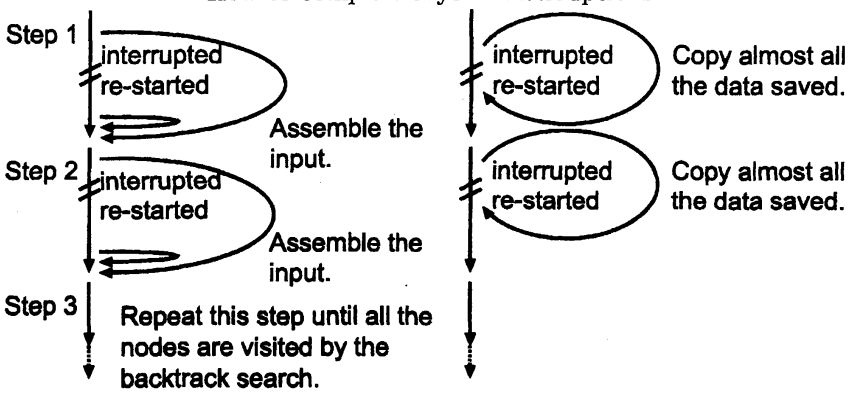
Step 2 to Step 3 and ···



searched in step 2
◉ remained in step 2
→distributed to generate ◙ in step 3
Distribute ◉ & ◙ for subtree searchs in step 3
within a time limit.

Step 2 と同様な計算を未完成な relation matrix がでてこなくなるまで繰返す。

How to compute beyond interruptions



これまでは、計算が中断されても、それぞれの step で保存しておいた data を使って再 start できるが、次の step に続けては進めなかった。むしろ、中断されるのが普通だったので、次の step に続けては進むことは考えていなかった。今回、完成した relation matrix 以外の data を再 start 時に新出力ファイルにコピーして、それまでの結果に形式上は追加するかたちにプログラムを改造して、自動的に、次の step に継続して進めるようにした。その結果、実験データも容易に取れるようになった。

4 計算実験

まず、並列計算のデータ保存と通信にかかる時間を次の表で推定した。計算時間の単位は断らない限り秒です。

Basic data for input $[[11,11],[2,1]]$
Comparing with parallel computing with one slave

Ordinary computing		30 sec time limit		unlimited	
CPU time	wall clock time	master=slave	master≠slave	master=slave	master≠slave
2250720 milli-sec	2251	2288	2284	2265	2262

Accelerating data for input $[[11,11],[2,1]]$ with the time limit of 30 seconds (4 trials)

Number of slaves	1-st	2-nd	3-rd	4-th	unlimited
5	663	633	471	493	471
10	342	316	253	259	306
20	184	172	139	141	218
30	145	124	101	100	-
40	109	103	95	94	-
50	116	94	84	88	-
60	108	84	73	76	-
70	86	73	72	70	-

上は、一つの入力で slave の数を変えて時間を計った。時間制限を設けない方法では、最大の subtree の search に 215sec かかるので、30　slaves 以上は省略した。

Some fast results on input $[[11,11],[2,1]]$

Number of slaves	30 sec	90% busy	12 sec	90% busy	unlimited	90% busy
5	471	458	472	457	471	447
10	249	223	246	230	306	282
20	140	113	133	115	218	100
30	99	73	98	75	-	
40	93	59	76	57	-	
50	84	41	69	43	-	
60	73	29	63	37	-	
70	70	18	58	30	-	

この例では、制限時間を 12sec にした方が良いように思われる。この例は、3 steps かかっているが、その3回の計算時間のバランスが 12sec のときに良いことが理由と思える。
いずれにしても、70 slaves は、上の例では多すぎる様である。次の例では、並列計算がうまくいっているようである。

Date for input $[[13,13],[2,1]]$ under the time limit of 720 seconds

num. of slaves	73	number of steps	4
timing	91905	90% busy	89797
interruption	once in step2, three times in step 3		

次の表は、いろいろな level での node 数の求めている。( ) 内の数は、並列計算で次の step への入力として残った node 数です。最後の例では、node 数の分布が変則的で、最後に残った 4 node から、level 17 で非常に多くの children が出てくることが、この次の表で示される。

Number of nodes in the searching trees

Level	$[[11,11],[2,1]]$	$[[13,13],[2,1]]$	$[[1,2,2,2,2,2,2,2,8,8],[1,2,3,4,5,6,7,8,10,9]]$	
5	164	1067 (425)	10	
6	601	15794 (2497)		
7	3769	428728 (4)	18	
9			114	(39)
11			1003	(263)
13			9893	(1228)
15			58011	(4)
16			17387	
$n-1$	104	4508	767	

Data of the 4 nodes at level 15 of
$[[1,2,2,2,2,2,2,2,8,8],[1,2,3,4,5,6,7,8,10,9]]$

node number	number of children	timing
1	4948	3312
2	1137	1098
3	1392	1320
4	4748	3116

それで、下のように改良しないと、並列計算が続行できなかった。

Data 転送量の節約：

Slave も input が何かを知っている。relation matrix の部分行列を送って、input から全行列を復元して計算する。

例：input [[1,1,1,2,2],[1,3,2,4,5]]

Send a node at level 4 (説明図より deeper level の場合もあり):

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & 3 & 2 & 4 & 4 & 5 & 5 \\ 3 & 2 & 0 & 1 & 5 & 5 & 4 & 4 \\ 2 & 3 & 1 & 0 & 5 & 5 & 4 & 4 \end{pmatrix}$$

Receive the children at level 5:(５行目のみを受取る)

$$\begin{pmatrix} 4 & 4 & 5 & 5 & 0 & 1 & 2 & 3 \end{pmatrix}, \begin{pmatrix} 4 & 4 & 5 & 5 & 0 & 1 & 3 & 2 \end{pmatrix}$$

Distribute the nodes at level 5: (最初の４行に受取った５行目を付加えて送る。)

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & 3 & 2 & 4 & 4 & 5 & 5 \\ 3 & 2 & 0 & 1 & 5 & 5 & 4 & 4 \\ 2 & 3 & 1 & 0 & 5 & 5 & 4 & 4 \\ 4 & 4 & 5 & 5 & 0 & 1 & 2 & 3 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 4 & 5 & 5 \\ 1 & 0 & 3 & 2 & 4 & 4 & 5 & 5 \\ 3 & 2 & 0 & 1 & 5 & 5 & 4 & 4 \\ 2 & 3 & 1 & 0 & 5 & 5 & 4 & 4 \\ 4 & 4 & 5 & 5 & 0 & 1 & 3 & 2 \end{pmatrix}$$

Receive the results ($\supseteq$ the inputs to the next step).

上のようにして、前頁最後の例では level 15 の node を slave に送って level 17 の子として relation matrix の１６と１７行目のみを受取るようにして並列計算を続行することができた。

参 考 文 献

[1] G. Cooperman, http://www.ccs.neu.edu/home/gene/pargap.html

[2] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version4*, Lehrstuhl D f Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany and School of Mathematical and Computational Sciences, Univ. St. Andrews, Scotland, 1997.

[3] A. Hanaki. Data of association schemes, published at WWW (1999。チ): http://kissme.shinshu-u.ac.jp/as/.

[4] A. Hanaki and I. Miyamoto, *Classification of association schemes with* 16 *and* 17 *vertices*, Kyushu J. Math. **52** (1998) 383-395. *Classification of association schemes with* 18 *and* 19 *vertices*, Korean J. Comp. App. Math. **5** (1998) 543-551. *Classification of association schemes of order up to* 22, Kyushu J. Math. **54** (2000) 81-86.

[5] I. Miyamoto, *Parallel Backtrack Computing of Association Schemes Using Classroom PC's*, MATHEMATICAL SOFTWARE, Proceedings of the First International Congress of Mathematical Software, A. M. Cohen, X-S. Gao and N. Takayama Ed. World Scintific Publishing Co. Pte. Ltd. (2002) 341-349.

[6] 宮本 泉, GAPパッケージ ParallelGAP の計算実験, Computer Algebra - Algorithms, Implementations and Applications, RIMS kokyuroku **1295** (2002) 29-34.