A Novel Parallel Algorithm for Enumerating Combinations*

Zhou B. B., Brent R. P. and Qu X.

Computer Sciences Laboratory
The Australian National University
Canberra, ACT 0200, Australia

Liang W. F

Department of Computer Science
The Australian National University
Canberra, ACT 0200, Australia

Abstract *In this paper we propose a new algorithm for parallel enumeration of combinations. This algorithm uses N processing elements (or PEs). We prove that, if N and M are relatively prime, each PE will do the same operations and generate the same number of distinct combinations so that the computational load is well balanced. The algorithm has an important application in solving the problem of fault tolerance in replicated file systems.*

1 Introduction

A number of parallel algorithms for generating combinations and permutations has been introduced in literature (e.g., those in [1, 2, 3, 4]). Those algorithms may be classified into two types. The first type of algorithms, for enumerating combinations (or permutations) of M out of N elements, uses M processing elements (or PEs). These PEs work cooperatively to generate one combination at a time, that is, the i^{th} PE only generates the i^{th} elements of each subset (assuming PEs are numbered).

Using K PEs for K a positive integer, the second type of algorithms can generate combinations in lexicographic order and each PE may produce an interval of $\frac{1}{K}\binom{N}{M}$ subsets. The best algorithm for this type is described in [1]. In that algorithm each combination is associated with a unique integer. By using those integers, a PE can easily determine the first combination in the interval. After the first combination is generated, the rest combinations in that interval can easily be obtained.

In this paper we present a new parallel algorithm. This algorithm uses N PEs, each of which generates $\frac{1}{N}\binom{N}{M}$ distinct combinations. Assume that there are N locations which are indexed, say, from 0 to $N-1$ and that each location is equipped with a PE. A special feature of this algorithm is that regular communication patterns can be obtained if each location requires information from different locations associated with the elements in a generated combination. This requirement may be found in the problem of fault tolerance in replicated file systems [5].

2 The Algorithm

Assume that N PEs are in different locations which are numbered from 0 to $N-1$. The basic idea of our algorithm is that at each step a primitive pattern of M integers (out of N consecutive integers starting from zero) is first chosen as

$$P = \{T_0, T_1, \cdots, T_l, \cdots, T_{M-1}\} \tag{1}$$

where $T_0 = 0$, $T_l < N$ and $T_i < T_j$ if $i < j$. Location i then generates a combination P_i of size M according to this primitive pattern, that is,

$$\begin{aligned} P_i \quad = \quad & \{(i+T_0) mod N, (i+T_1) mod N, \\ & \cdots, (i+T_l) mod N, \cdots, (i+T_{M-1}) mod N\} \end{aligned} \tag{2}$$

where $0 \leq i \leq N-1$. If a set of primitive patterns is chosen properly, all $\binom{N}{M}$ combinations can be generated in parallel at N locations.

To obtain those proper primitive patterns, we must solve the following two problems. Consider an example of $N = 8$ and $M = 4$. It is easy to see that combination {0, 3, 4, 7} will be generated at locations 0 and 4 and that combination {1, 4, 5, 0} be generated at locations 1 and 5 when {0, 3, 4, 7} is used as a primitive pattern. Thus the first problem is how to obtain a primitive pattern which generates only distinct combinations.

Primitive patterns are defined as *dependent primitive patterns* if a combination can be generated by either of those patterns. Otherwise they are called *independent primitive patterns*. The second problem to

*Appeared in *Proc. ICPP*, 1996, Vol. II, 70–73.

be solved is how to avoid using dependent primitive patterns. In the following we prove that, if N and M are relatively prime, (i.e., $(N, M) = 1$) the combinations generated by the same primitive pattern are all distinct and dependent primitive patterns can only generate the same set of combinations. If N and M are chosen to be relatively prime, therefore, we can find a fixed number of independent primitive patterns for generating all distinct combinations exactly only once. With these independent patterns each location will produce the same number of distinct combinations. The computational load is thus well balanced.

The following two lemmas show that the combinations generated by the same primitive pattern are all distinct if $(N, M) = 1$.

Lemma 1 *Assume that the greatest common divisor of b and d is e, that is, $(b, d) = e$ for $0 < d \leq b$ and $e \geq 1$. If the elements in a given primitive pattern satisfy the equations*

$$T_{b+i} = T_b + T_i \tag{3}$$

and

$$T_{d+j} = T_d + T_j \tag{4}$$

where $0 \leq i \leq M-1-b$, $0 \leq j \leq b-1$, then T_e divides both T_b and T_d, or written as $T_e \mid T_b$ and $T_e \mid T_d$.

Proof. Setting $b = d * q^{(1)} + r^{(1)}$ for $q^{(1)} > 0$ and $0 < r^{(1)} \leq d-1$ and applying it to (3), we have

$$T_{d*q^{(1)}+r^{(1)}+i} = T_{d*q^{(1)}+r^{(1)}} + T_i,$$

or

$$T_{d+d*(q^{(1)}-1)+r^{(1)}+i} = T_{d+d*(q^{(1)}-1)+r^{(1)}} + T_i. \tag{5}$$

If $0 \leq i \leq d-1$, then $d*(q^{(1)}-1) + r^{(1)} + i \leq b-1$. Applying (4) to (5), we have

$$T_d + T_{d*(q^{(1)}-1)+r^{(1)}+i} = T_d + T_{d*(q^{(1)}-1)+r^{(1)}} + T_i,$$

or

$$T_{d*(q^{(1)}-1)+r^{(1)}+i} = T_{d*(q^{(1)}-1)+r^{(1)}} + T_i.$$

Continuing the above process, we can obtain

$$T_{r^{(1)}+i} = T_{r^{(1)}} + T_i \tag{6}$$

where $0 \leq i \leq d-1$.

Let $d = r^{(1)}q^{(2)} + r^{(2)}$ for $q^{(2)} > 0$ and $0 < r^{(2)} \leq r^{(1)} - 1$. Using (4) and (6), for the same reason we may have

$$T_{r^{(2)}+i} = T_{r^{(2)}} + T_i \tag{7}$$

where $0 \leq i \leq r^{(1)} - 1$. By continuously using the *Euclidean algorithm* (for finding the greatest common divisor of b and d) and applying the same procedure as above, we eventually obtain

$$r^{(n-3)} = r^{(n-2)}q^{(n-3)} + e,$$
$$r^{(n-2)} = q^{(n-2)} * e \tag{8}$$

and

$$T_{e+i} = T_e + T_i \tag{9}$$

where $0 \leq i \leq r^{(n-2)} - 1$.

Now the process goes backward, that is, we first calculate $T_{r^{(n-2)}}$. From (8), we have

$$\begin{aligned} T_{r^{(n-2)}} &= T_{q^{(n-2)}e} \\ &= T_{e+(q^{(n-2)}-1)e}. \end{aligned} \tag{10}$$

Since $e \geq 1$, then

$$\begin{aligned} (q^{(n-2)} - 1)e &= q^{(n-2)}e - e \\ &= r^{(n-2)} - e \\ &\leq r^{(n-2)} - 1. \end{aligned}$$

Applying (9) to (10), we thus obtain

$$T_{r^{(n-2)}} = T_e + T_{(q^{(n-2)}-1)e}$$

and further

$$T_{r^{(n-2)}} = q^{(n-2)}T_e.$$

Similarly, we can have

$$\begin{aligned} T_{r^{(n-3)}} &= T_{q^{(n-3)}r^{(n-2)}+e} \\ &= q^{(n-3)}T_{r^{(n-2)}} + T_e \\ &= (q^{(n-3)}q^{(n-2)} + 1)T_e. \end{aligned}$$

Thus T_e also divides $T_{r^{(n-3)}}$. Continuing to trace back, we can finally obtain that T_e divides both T_d and T_b. If $b = f * e$ and $d = g * e$ for $f > 0$ and $g > 0$, in particular, we have $T_b = f * T_e$ and $T_d = g * T_e$. The proof for this is easy, but tedious and thus omitted. □

Lemma 2 *If $(N, M) = 1$, all the combinations generated by the same primitive pattern at different locations will be distinct.*

Proof. We prove this lemma by showing that different locations may generate the same combination by the same primitive pattern only if N and M have a common divisor greater than one.

Without loss of generality, we assume that P_0 and P_a for $a > 0$ are the same combination and have the forms

$$P_0 = \{T_0, T_1, \cdots, T_b, \cdots, T_{M-1}\}$$

and

$$P_a = \{(a+T_0) mod N, (a+T_1) mod N, \cdots, (a+T_b) mod N, \cdots, (a+T_{M-1}) mod N\}.$$

Assume $(a+T_0) mod N = T_b$, or $a = T_b$ for $0 < b \leq M-1$. For $0 \leq l \leq M-1$ we have

$$T_{(b+l) mod M} = (a+T_l) mod N.$$

If $l \leq M-1-b$, then

$$T_{(b+l) mod M} = T_{b+l} \leq T_{M-1}.$$

We know that $(a+T_l) mod N$ must increase to reach T_{M-1} before it becomes T_0 as l increases. For $l \leq M-1-b$, then

$$(a+T_l) mod N = a+T_l = T_b+T_l.$$

Thus we obtain

$$T_{b+l} = T_b + T_l \tag{11}$$

where $0 \leq l \leq M-1-b$.

Let $M = hb+d$ for $h > 0$ and $0 < d \leq b$. If $l = (h-1)b+d-1 = M-1-b$, from (11) we have

$$T_b + T_{(h-1)b+d-1} = T_{M-1}.$$

The next immediate element in P_a must be equal to $T_0 = 0$. Otherwise, the two combinations will not be the same. Thus we have

$$(T_b + T_{(h-1)b+d}) mod N = T_0 = 0,$$

or

$$T_b + T_{(h-1)b+d} = N. \tag{12}$$

For $0 \leq i \leq b-1$, then

$$(T_b + T_{(h-1)b+d+i}) mod N = T_i,$$

or

$$T_b + T_{(h-1)b+d+i} = N + T_i \tag{13}$$

¿From equations in (12) and (13), we obtain

$$T_{(h-1)b+d+i} = T_{(h-1)b+d} + T_i \tag{14}$$

where $0 \leq i \leq b-1$.

From (11) we can have

$$\begin{aligned} T_{kb+i} &= T_{b+(k-1)b+i} \\ &= T_b + T_{(k-1)b+i} \\ &\vdots \\ &= kT_b + T_i \end{aligned}$$

for $kb+i \leq M-1$.

Since $(h-1)b+d+i \leq M-1$ for $i \leq b-1$, the equation in (14) can then be rewritten as

$$(h-1)T_b + T_{d+i} = (h-1)T_b + T_d + T_i,$$

or

$$T_{d+i} = T_d + T_i \tag{15}$$

for $0 \leq i \leq b-1$.

We see from the above discussion that T_i (for $0 \leq i \leq M-1$) must satisfy the two equations in (11) and (15) if $P_0 = P_a$. Let the greatest common divisor of b and d be $(b,d) = e$, or $b = f*e$ and $d = g*e$ for $e \geq 1$, $f > 0$ and $g > 0$. ¿From Lemma 1 we have $T_b = fT_e$ and $T_d = gT_e$. Therefore, from (12) we obtain

$$\begin{aligned} N &= T_b + T_{(h-1)b+d} \\ &= hT_b + T_d \\ &= h*fT_e + gT_e \\ &= (h*f+g)T_e \\ &= cT_e \end{aligned}$$

where $c = h*f+g$. We also have

$$\begin{aligned} M &= h*b+d \\ &= h*f*e+g*e \\ &= (h*f+g)e \\ &= c*e. \end{aligned}$$

Since h, f and g are all greater than zero, then $c = h*f+g > 1$. Therefore, N and M must have a common divisor greater than one. □

We now prove that dependent primitive patterns can only generate the same set of combinations.

Lemma 3 *If two combinations generated by different primitive patterns are the same, then any combination generated by one of these primitive patterns can also be generated by the other.*

Proof. Let P and P' be two distinct primitive patterns

$$P = \{T_0, T_1, \cdots, T_l, \cdots, T_{M-1}\}$$

and

$$P' = \{{T_0}', {T_1}', \cdots, {T_l}', \cdots, {T'}_{M-1}\}.$$

Without loss of generality, we assume that P_b and ${P_a}'$ are the same combination generated by P and P' respectively for $a-b=e$ and $e>0$. Then

$$a = b+e = b+T_c$$

	0	1	2	3	4	5	6	7	8
group 1	0	1	2	3	4				
group 2	0	1	2	3		5			
	0	1	2	3			6		
	0	1	2	3				7	
group 3	0	1	2		4	5			
	0	1	2		4		6		
	0	1	2		4			7	
	0	1	2			5	6		
	0	1	2			5		7	
	0	1	2				6	7	
group 4	0	1		3	4		6		
	0	1		3	4			7	
	0	1		3		5		7	
	0	1			4	5		7	

Figure 1: A number of $\frac{1}{N}\binom{N}{M} = 14$ independent primitive patterns for $N = 9$, $M = 5$.

where $0 \leq c \leq M-1$. We thus have

$$(b + T_{(c+l) mod M}) mod N = (a + T_l{}') mod N,$$

or

$$T_{(c+l) mod M} = (e + T_l{}') mod N$$

and

$$\begin{aligned}&(k + T_{(c+l) mod M}) mod N \\ &\quad = (k + (e + T_l{}') mod N) mod N \\ &\quad = (k + e + T_l{}') mod N \\ &\quad = ((k+e) mod N + T_l{}') mod N\end{aligned}$$

where $0 \leq l \leq M-1$ and $0 \leq k \leq N-1$.

Since modular arithmetic is applied in our computation, $(k+e) mod N$ for $0 \leq k \leq N-1$ has N distinct values which correspond to the N locations and similarly $(c+l) mod M$ for $0 \leq l \leq M-1$ has M distinct values which associate with the M indices of T_l. It is easy to see from the above equation that every combination generated by P can also be generated by P' and *vice versa*. We thus conclude that dependent primitive patterns only generate the same set of combinations. □

3 Discussions

In the previous section we have proved that, if N and M are chosen relatively prime, there exists a set of independent primitive patterns. Using these patterns, all $\binom{N}{M}$ combinations can be generated in parallel at N locations. Now the problem is how to construct these independent primitive patterns in a reasonable and systematic way. We have a very simple method to do that. Here we only present an example of $N = 9$ and $M = 5$, as shown in Fig. 1. (For details see [6].) Because of the simplicity and regularity, the method can easily be implemented.

Our algorithm can also be extended to the case when N and M are not relatively prime. Because Lemma 2 cannot be applied, however, some special care has to be taken into consideration. It is easy to prove that N may not divide $\binom{N}{M}$ if N and M are not relatively prime. The computational load may then not be well balanced. This imbalance of computational load occurs only in certain steps in which each of the given primitive patterns generates the same combination at different locations. When our method for generating independent primitive patterns is applied, those patterns can easily be identified and a simple technique may be applied to ensure that each combination is generated exactly once only.

In this paper we only discussed how to use N PEs to enumerate $\binom{N}{M}$ combinations. Another interesting problem is how to generate those combinations by using only P PEs for $P \leq N$. One solution to this problem can be found in [6].

References

[1] S. G. Akl, "Adaptive and optimal parallel algorithms for enumerating permutations and combinations", *The Computer Journal*, Vol. 30, No. 5, 1987, pp. 433-436.

[2] G. H. Chen and M. S. Chern, "Parallel generation of permutations and combinations", *BIT*, Vol. 26, 1986, pp. 277-283.

[3] C. J. Lin and J. C. Tsay, "A systolic generation of combinations", *BIT*, Vol. 29, 1989, pp. 23-36.

[4] I. Stojmenovic, "An optimal algorithm for generating equivalence relations on a linear array of processors", *BIT*, Vol. 30, 1990, pp. 424-436.

[5] B. B. Zhou, R. P. Brent, X. Qu and W. F. Liang, "A method for solving the problem of fault tolerance in replicated file systems", in preparation.

[6] B. B. Zhou, R. P. Brent, X. Qu and W. F. Liang, "A New Method for Parallel Generation of Combinations", in preparation.