Australian Journal of Science and Technology

ISSN Number (2208-6404)
Volume 6; Issue 1; March 2022

Original Article

Parallel algorithm for listing combinatorial of C(n,r)

Nguyen Dinh Lau*

Department of Information Technology, University of Education and Science, University of Danang, Vietnam

ABSTRACT

This paper aims at constructing listing combinatorial Algorithms of C(n,r). C(n,r) is replaced by Cnr. This is the most common and appealing problem in discrete maths. The Cnr listing combinatorial problem is solved by exploiting many different techniques: Using generation and backtracking. The most possible complexity is O(r.Cnr). In, an attempt was made to find the subset of k elements in the set of m elements. The main content in the paper is based on the generation combinatorial algorithm of a smaller set. This will reduce the computation time as compared to the initial set n and r. The complexity is Max{O(int(r/2).C(n.r+int(r/2),int(r/2))),O(r.int(r/2).C(n-int(r/2),r.int(r/2)))}. This paper aims at building parallel algorithms based on the listing combinatorial algorithms in to improve computing performance for listing combinatorial algorithms of Cnr. In addition, to take more advantage of multi-core architecture of the parallel computing system, we build this algorithm on multiple processors. This is a completely new method not being announced in the world. The results of this paper are basically systematized and proven.

Keywords: Algorithm, combinatorial, generation, listing, parallel, subsets

Submitted: 20-12-2021, **Accepted:** 20-02-2022, **Published:** 30-03-2022

INTRODUCTION

Listing combinatorial problems are common problems in discrete maths and have been applied to solve real problems such as: Queens, Course Breakdown and others. However, when the input data is huge, the complexity is extremely large.[1] It is, therefore, essential to propose algorithms with parallel algorithm to improve the complexity.

The author has conducted in-depth studies in combinatorial algorithms and had many articles related to this field published widely nationally and internationally. Permutation and parallelization are discussed in the work named the problem of permutation and parallelization,[2] improved computing performance for listing combinatorial algorithms using multi-processing MPI and Thread library[3] was also presented and published.

In the world, combinatorial algorithms have attracted attention of numerous researchers. The authors in[4-11] discussed and proposed parallel listing combinatorial algorithms that specifically develop a parallel listing combinatorial algorithm of set Cnr.

In paper[1] is underpinned by generation algorithm to build new listing combinatorial algorithms by dividing a set into different subsets to find the combinatorial sequence.

This article has made some new contributions to listing combinatorial Algorithms by constructing listing combinatorial parallel algorithm; giving correct proof; demonstrating complexity and giving illustrative and comprehensive examples. This parallel algorithm develops from the algorithm published in.[1]

LISTING COMBINATORIAL ALGORITHM OF CNR

How to Find the Next Combinatorial

Let $\alpha = \{s_1, s_2, ., s_r\}$, we find the next combinatorial $\beta = \{t_1, t_2, ., t_r\}$.

First of all, it can be observed that the i^{th} element in the combinatorial cannot exceed $n-r+i$ because $n-r+i$ is

Address for correspondence: Nguyen Dinh Lau, Department of Information Technology, University of Education and Science, University of Danang, Vietnam. E-mail: ndlau@ued.udn.vn

considered the maximum value of the i^{th} element. We find m = max $\{i \mid s_i < n - r + i\}$. Then we have

1. $t_i = s_i$ *for* $i = 1, 2., m-1$
2. $t_m = s_m + 1$
3. $t_m + i = s_m + i + 1$ *for* $i = 1, 2., r-m$

Algorithm

- Input: r, n
- Output: a list of combinatorial of Crn with increasing order.
- Steps:

```
Generation (int n, int r)
{
While (s[1]! = n-r+1)
                    {
        printcombinatorial(r);
        cout<<endl;
        int m=0;
        for (int i=r; i>=1; i--)
        if(s[i]<n-r+i)
                    {
                    m=i;
                    break;
                    }
        s[m]=s[m]+1;
        for (int i=m+1; i<=r; i++)
        s[i]= s[i-1] +1;
        }//while
}
```

There is Cnr^{th} loop of while (line 2), and There is r^{th} loop of for (line 7). Thus we have complexity O(r. Cnr)

A New Approach to Construct Listing Combinatorial Algorithm of Cnr

It can be seen that the bigger the complexity O(r. Cnr) of the listing combinatorial algorithm of Cnr with gets, the more the n elements increase. Therefore, the author suggests constructing new listing combinatorial algorithm by finding every single subset. For each subset, A_i contains fixed first elements of combinatorial while unfixed last elements of combinatorial are in $R_i \subset X_i$ (R_i as set listing combinatorial Cmk), where $m = |X_i|$, m <n and k <r. The A_i and X_i are defined in the following section.[1]

The algorithm listing combinatorial of Cnr

Hence n=7, r=5 then Cnr=21

It is time to divide and allocate elements of combinatorial for A_i and X_i

Analysis 1

1. Let X_1 = {3,4,5,6,7}. Let A_1 with 2 first elements {1,2} ∪ R_1 ($R_1 \subset X_1$, card (R_1)=3). R_1 as set listing combinatorial Cmk (m=5, k=3), Cmk=C53=10 with 10 combinatorial as in Table 1

Table 1: Listing combinatorial of C75 equa to $A_i \cup R_i$

No.	Combinatorial $A_i \cup R_i$	Data of $R_i \subset X_i$
1.	12U345	$R_1 \subset X_1$={3,4,5,6,7}, with m=5, k=3, C53=10
2.	12U346	
3.	12U347	
4.	12U356	
5.	12U357	
6.	12U367	
7.	12U456	
8.	12U457	
9.	12U467	
10.	12U567	
11.	13U456	$R_2 \subset X_2$={4,5,6,7}, with m=4, k=3, C43=4
12.	13U457	
13.	13U467	
14.	13U567	
15.	14U567	$R_3 \subset X_3$={5,6,7}, with m=3, k=3, C33=1
16.	23U456	$R_4 \subset X_4$={4,5,6,7}, with m=4, k=3, C43=4
17.	23U457	
18.	23U467	
19.	23U567	
20.	24U567	$R_5 \subset X_5$={5,6,7}, with m=3, k=3, C33=1
21.	34U567	$R_6 \subset X_6$={5,6,7}, with m=3, k=3, C33=1

2. Let X_2 = (4,5,6,7}. Let A_2 with 2 first elements {1,3} ∪ R_2 ($R_2 \subset X_2$, card (R_2)=3). R_2 as set listing combinatorial Cmk (m=4, k=3), Cmk=C43=4 with 4 combinatorial as in Table 1
3. Let X_3 = (5,6,7}. Let A_3 with 2 first elements {1,4} ∪ R_3 ($R_3 \subset X_3$, card (R_3)=3). R_3 as set listing combinatorial Cmk (m=3, k=3), Cmk=C33=1 with 1 combinatorial as in Table 1
4. Let X_4 = (4,5,6,7}. Let A_4 with 2 first elements {2,3} ∪ R_4 ($R_4 \subset X_4$, card (R_4)=3). R_4 as set listing combinatorial Cmk (m=4, k=3), Cmk=C43=4 with 4 combinatorial as in Table:
5. Let X_5 = (5,6,7}. Let A_5 with 2 first elements {2,4} ∪ R_5 ($R_5 \subset X_5$, card (R_5)=3). R_5 as set listing combinatorial Cmk (m=3, k=3), Cmk=C33=1 with 1 combinatorial as in Table 1
6. Let X_6 = (5,6,7}. Let A_6 with 2 first elements {3,4} ∪ R_6 (R_6
7. X_6, card (R_6)=3). R_6 as set listing combinatorial Cmk (m=3, k=3), Cmk=C33=1 with 1 combinatorial as in Table 1.

Analysis 2
Subset A_i (i=1, 2,…, 6) ⊂ B= {1,2,3,4}. $|A_i|$=2

Analysis 3
Subset general A can be identified as follows:
Let n, r. A_i (i=1,2,…,C(n-r+int(r/2),int(r/2))) ⊂ B={1,2,…,n-r+int(r/2). With $|A_i|$=int(r/2), |B|=n-r+int(r/2).

Analysis 4
Suppose that A_i[int(r/2)] is the value of the last element of A_i. When n=7 and r = 5, we have:
A_1[int(r/2)] = A_1[2] = 2
A_2[int(r/2)] = A_2[2] = 3
A_3[int(r/2)] = A_3[2] = 4
A_4[int(r/2)] = A_4[2] = 3
A_5[int(r/2)] = A_5[2] = 4
A_6[int(r/2)] = A_6[2] = 4

Analysis 5:
X_i can be identified from A_i as follows:
X_i={A_i[int(r/2)]+1, A_i[int(r/2)]+2,…, n}

Theorem 1: $|A_i \cup R_i|$=r, with $|R_i|$= r-int(r/2), $R_i \subset X_i$[1]

Analysis 6:
The example shows that the number of combinatorial is the times to get three elements in X_i (i = 1, 2,…, 6). It means that the number of combinatorial = 6
C53+C43+C33+C43+C33+C33=10+4+1+4+1+1=21=C75.

In general, we have the number of combinatorial as follows:
[C(n-int(r/2),r-int(r/2))+2.C(n-int(r/2)-1,r-int(r/2))+3.C(n-int(r/2)-2),r-int(r/2))+…+r-int(r/2)C(n-int(r/2)-n+r),r-int(r/2))]+[C(n-int(r/2)-1,r-int(r/2))+2C(n-int(r/2)-2),r-int(r/2))+…+r-int(r/2)C(n-int(r/2-n+r),r-int(r/2))] +…+[C(n-int(r/2)-n+r),r-int(r/2))]

Similarly, the number of new Approach listing combinatorial equa to Cnr.
Hence there are two main steps to construct new listing combinatorial algorithm:

1. Initialize subset A_i. $|A_i|$ =int(r/2), $A_i \subset B$ = {1, 2,…, n-r+int(r/2), |B|=n-r+int(r/2).
   i=1, 2, 3,…, C(n-r+int(r/2), int(r/2)).
2. For A_i: Print $A_i \cup R_i$ with $|R_i|$= r-int(r/2), $R_i \subset X_i$

Constructing new listing combinatorial algorithm
Input: n, r
Output: Print all listing combinatorial of Cnr
1. Find A_i: Let A_i listing combinatorial C(n-r+int(r/2), int(r/2))
2. Find X_i:
   for A_i (i=1,2,…, C(n-r+int(r/2), int(r/2)).
   Initialize: X_i={A_i[int(r/2)]+1, A_i[int(r/2)]+2,…, n}
3. for A_i (i=1,2,3,…, C(n-r+int(r/2), int(r/2)).
   Print all listing combinatorial $A_i \cup R_i$ with $|R_i|$= r-int(r/2), $R_i \subset X_i$ (R_i as set listing combinatorial Cmk, m=$|X_i|$, k=r-$|A_i|$)

Theorem 2: Max($|X_i|$) =$|X_1|$
Proof: See[1]

Theorem 3: The complexity
Max{O(int(r/2).C(n-r+int(r/2),int(r/2))),O(r-int(r/2).C(n- int(r/2), r-int(r/2)))} <O(r.C (n,r))
The complexity of the new listing combinatorial algorithm is smaller than the complexity of the previous combinatorial algorithm.[1]

PARALLEL ALGORITHM FOR LISTING COMBINATORIAL OF CNR

Ideas of Parallel Algorithm
It can be seen that the bigger the complexity *Max{O(int(r/2).C(n-r+int(r/2),int(r/2))),O(r-int(r/2).C(n-int(r/2),r-int(r/2)))} <O(r.C (n,r))* of new approach the listing combinatorial algorithm of Cnr with gets, the more the n elements increase.

Sequential algorithms might take a long time to process if *n* length is large. Therefore, it is necessary to build parallel algorithms to improve computing performance for the algorithms.

In sequential algorithms, the author suggests constructing new listing combinatorial algorithm by finding every single subset. For each subset, A_i contains fixed first elements of combinatorial while unfixed last elements of combinatorial are in $R_i \subset X_i$ (R_i as set listing combinatorial Cmk), where m = $|X_i|$, m <n and k <r. The A_i and X_i are defined in the section 2.3.

This study builds up parallel algorithms to improve computing performance for listing combinatorial algorithms. To parallel execution, author divide the data set input and allocate them to the processors. The article focuses on (i) the analysis of the input data structure to divide data for the sub processors, and (ii) the construction of parallel algorithms - proof of correctness and analysis of computing complexity. Then the comparison of the results of the parallel algorithm with the sequential algorithm and the comparison of the execution time on different sub processors is discussed.

Parallel Algorithm
This newly-built parallel algorithms use *h* processors (P_1, P_2,…,P_h). The processors P_i receives the output value which is the input value of A_i and X_i as in Table 2.

Parallel algorithm finding combinatorial of n elements
Input: n, r, *h* processors (P_1, P_2,…,P_i)
Out: Listing combinatorial
Steps:

Table 2: Listing combinatorial of C75 equa to $A_i \cup R_i \subset X_i$ and divide the data set input to the processors P_i

No.	Data set input to P_i	Combinatorial $A_i \cup R_i$	Data of $R_i \subset X_i$
1.	P_1: receive input is A_1 and X_1	12∪345	$R_1 \subset X_1$={3,4,5,6,7}, with m=5, k=3, C53=10
2.		12∪346	
3.		12∪347	
4.		12∪356	
5.		12∪357	
6.		12∪367	
7.		12∪456	
8.		12∪457	
9.		12∪467	
10.		12∪567	
11.	P_2: receive input is A_2 and X_2	13∪456	$R_2 \subset X_2$={4,5,6,7}, with m=4, k=3, C43=4
12.		13∪457	
13.		13∪467	
14.		13∪567	
15.	P_3: receive input is A_3 and X_3	14∪567	$R_3 \subset X_3$={5,6,7}, with m=3, k=3, C33=1
16.	P_4: receive input is A_4 and X_4	23∪456	$R_4 \subset X_4$={4,5,6,7}, with m=4, k=3, C43=4
17.		23∪457	
18.		23∪467	
19.		23∪567	
20.	P_5: receive input is A_5 and X_5	24∪567	$R_5 \subset X_5$={5,6,7}, with m=3, k=3, C33=1
21.	P_6: receive input is A_6 and X_6	34∪567	$R_6 \subset X_6$={5,6,7}, with m=3, k=3, C33=1

1. Let A_i listing combinatorial C(n-r+int(r/2), int(r/2))
2. for A_i (i=1,2,3,…, C(n-r+int(r/2), int(r/2)).
   Initialize: X_i={A_i[int(r/2)]+1, A_i[int(r/2)]+2,…, n}
3. Divide the data set and allocate them to h processors (P_1, P_2,…,P_h)
   Processor P_1 receive A_1 and X_1
   Processor P_2 receive A_2 and X_2
   and processor P_i receive A_i and X_i.
4. Processors (P_1, P_2,…,P_i) perform concurrently
   - P_i Find listing combinatorial R_i with $|R_i|$= r-int(r/2), $R_i \subset X_i$ (R_i as set listing combinatorial Cmk, m=$|X_i|$, k=r-$|A_i|$)
   - P_i Print all listing combinatorial $A_i \cup R_i$.

Theorem 4: The parallel algorithm is true.
Proof: This parallel algorithm develops from the algorithm published in.[1] In theorem 1, theorem 2, theorem 3, the author proved the sequential algorithm is true. The perform of the parallel algorithm is to divide the data sets A_i and X_i for P_i. Processors P_i receive A_i and X_i and find listing combinatorial R_i next print all listing conbinatorial $A_i \cup R_i$. This is consistent with the perform of the sequential algorithm.

Theorem 5: The complexity of parallel algorithm is O(int(r/2).C(n-r+int(r/2),int(r/2))) OR O(r-int(r/2).C(n-int(r/2),r-int(r/2))) perform in processor P_1
Proof: The complexity of the sequential algorithm Max{O(int(r/2).C(n-r+int(r/2),int(r/2))),O(r-int(r/2).C(n- int(r/2), r-int(r/2)))}

Indeed, Initialize A_i including combinatorial C(n-r+int(r/2), int(r/2)) then the complexity is O(int(r/2).C(n-r+int(r/2), int(r/2)))

According to theorem 2, Max($|X_i|$) =$|X_1|$ hence the combinatorial r-int(r/2) from X_1 is the maximum. We also have X_1= {A_1[int(r/2)]+1, A_1[int(r/2)]+2,…,n}={int(r/2)+1, int(r/2)+2,…,n} then the number of elements of X_1 is $|X_1|$=n–(int(r/2)+1)+1=n- int(r/2). It can be deduced that the complexity to identify combinatorial r-int(r/2) from X_1 is O(r-int(r/2).C(n- int(r/2), r-int(r/2))).

Finally we have the complexity Max{O(int(r/2).C(n-r+int(r/2),int(r/2))),O(r-int(r/2).C(n- int(r/2), r-int(r/2)))}
Let *h* processors (P_1, P_2,…,P_i).

Case 1

If O(int(r/2).C(n-r+int(r/2),int(r/2)))>O(r-int(r/2).C(n- int(r/2), r-int(r/2)))
Then the complexity of the listing combinatorial sequential algorithm is:
O(int(r/2).C(n-r+int(r/2), int(r/2))). Therefore, the complexity of the listing combinatorial parallel algorithm is:
O(int(r/2).C(n-r+int(r/2), int(r/2))) perform in processor P_1.

Case 2

If O(int(r/2).C(n-r+int(r/2),int(r/2)))<O(r-int(r/2).C(n- int(r/2), r-int(r/2)))
Then the complexity of the listing combinatorial sequential algorithm is:
O(r-int(r/2).C (n- int(r/2), r-int(r/2))). Therefore, the complexity of the listing combinatorial parallel algorithm is: O(r-int(r/2).C (n- int(r/2), r-int(r/2))) perform in processor P_1.

In both cases the complexity of the listing combinatorial parallel algorithm is smaller than the complexity of the previous combinatorial algorithm is O(n. Cnr).

CONCLUSION

In this paper, the author has built a listing combinatorial parallel algorithm of Cnr based on the listing combinatorial sequential algorithms in. In particular, the author also analyzed

tthe algorithm logically and also proved 2 theorems (theorem 4 and theorem 5) related to the parallel algorithm.

Last but not least, the paper proves that the complexity of the listing combinatorial parallel algorithm is smaller than that of existing combinatorial sequential algorithm in both cases and perform in processor P_1.

REFERENCES

1. Lau ND. A new approach to listing combinatorial algorithm of CNR. Aust J Sci Technol 2021;5:539-43.
2. Lau ND. Parallel Algorithm List Permutations. Quy Nhon, Binh Dinh, Vietnam: Science and Technics publisher; 2017. p. 348-53.
3. Lau ND. Improved computing performance for listing combinatorial algorithms using multi-processing MPI and thread library. Int J Comput Sci Inform Technol 2018;10:16.
4. Stojmenovic I. Listing combinatorial objects in parallel. Int J Parallel Emerg Distrib Syst 2006;21:127-46.
5. Akl SG, Gries D, Stojmenovic I. An optimal parallel algorithm for generating combinations. Inform Proc Lett 1989;33:135-9.
6. Akl SG, Stojmenovic I. Parallel algorithms for generating integer partitions and compositions. J Comb Math Comb Comput 1983;13:107-20.
7. Chen GH, Chern MS. Parallel generation of permutations and combinations. BIT Numer Math 1986;26:277-83.
8. Djokic B, Miyakawa M, Sekiguchi S, Semba I, Stojmenovic I. Parallel algorithms for generating subsets and set partitions. In: Asano T, Ibaraki T, Imai H, Nishizeki T, editors. Vol. 450. Proceedings of SIGAL International Symposium on Algorithms. Tokyo, Japan: Lecture Notes in Computer Science; 1990. p. 76-85.
9. Elhage H, Stojmenovic I. Systolic generation of combinations from arbitrary elements. Parallel Proc Lett 1992;2:241-8.
10. Kapralski A. New methods for the generation of permutations, combinations, and other combinatorial objects in parallel. J Parallel Distrib Comput 1993;17:315-26.
11. Torres M, Barrera J. Aparallel algorithm for finding small sets of genes that are enough to distinguish two biological states. Genet Mol Biol 2004;27:686-90.