

Applying parallel computing for enumerating orthogonal arrays

Hien¹ H.T. Phan, Man¹ V.M. Nguyen and Anh¹ T. Nguyen

Faculty of Computer Science, HoChiMinh University of Technology, Vietnam

Abstract. Orthogonal array is an interesting notation in Mathematic and Design of Experiment. Of which, enumerating orthogonal arrays of strength t is also an important issue. So far some methods have been proposed for enumerating orthogonal arrays of strength t , however parallelize methods for this purpose still haven't been investigated. For this purpose, this paper will present an efficient parallel method for enumerating orthogonal arrays of strength t based on a serial backtrack search method. This method was implemented utilizing MPI and executed on the Supernode II cluster of HoChiMinh City University of Technology (HCMUT), Vietnam. Initial experiment results of the parallelization yields good speedups, efficiency ratios.

Keywords: DOE, parallel computing, discrete mathematics, industrial statistics, combinatorial search.

1 Introduction

Enumerating of orthogonal arrays has been studied for a long time. According to the recent research, N. Sloane [1] at AT&T published in 2005 a large amount of orthogonal array of strength 2 with run size $N \leq 100$. Also in this year, Man V.M. Nguyen reported many orthogonal arrays of strength 3 with run size at most 100 [2]. Some new methods have been proposed for enumerating orthogonal arrays of strength t where t is a positive integer at least 3 [2]. Among them, an important algorithm is the algorithm named *Finding lexicographically-least orthogonal array* with using sub-algorithm named *Backtrack search extend a column*. This algorithm is a serial, exhausted backtrack search method. In some cases, it takes a long time for enumerating orthogonal arrays with specific parameters.

To improve the performance and reduce the time of enumerating orthogonal arrays of strength t , a new parallel method was proposed and investigated in this paper to parallelize that original serial algorithm. The motivation of this method is that we can use the Lex-least matrices of the smaller parameter set to extend a new column and enumerate all Lex-least matrices of consecutive parameter set. This parallel method has been implemented using the MPI and executed on the Supernode II system of HCMUT, Vietnam. Some results about the speedup and efficiency ratio proved the usefulness of the new parallel method in comparison with the original serial method.

In this report, first, the definition of orthogonal array and the concept of enumerating orthogonal of strength t are introduced. Second, the serial algorithm named *Finding*

lexicographically-least orthogonal array and the sub-algorithm named *Backtrack search extend a column* are presented for enumerating orthogonal of strength t . Third, a new parallel method named *Reusing-Lex-Least-Matrixes* algorithm is investigated to parallelize the original serial algorithm. Finally, some results about the speedup and efficient ratio of this parallel method are presented.

2 Orthogonal array

2.1 Notation of orthogonal array

Let r_1, r_2, \dots, r_d be a list of natural numbers, and for each i , let Q_i be a set of size r_i and we call Q_i *factor*. The elements of a factor are called its *levels*. The (full) *factorial design* with respect to these factors is the Cartesian product $D = Q_1 \times \dots \times Q_d$. A fractional design or fraction F of D is a subset consisting of elements of D (possibly with multiplicities). It can be considered that F is symmetric if $r_1 = r_2 = \dots = r_d$, otherwise F is assumedly mixed. Let $s_1 > s_2 > \dots > s_m$ be the distinct factor sizes of F , and suppose that F has exactly a_i factors with s_i levels. The partition

$$r_1 r_2 \dots r_d = s_1^{a_1} s_2^{a_2} \dots s_m^{a_m}$$

is called the *design type* of F . The r_i is usually taken in non-increasing order, so that they are related to the s_k by

$$s_1 = r_1 = \dots = r_{a_1}, s_2 = r_{a_1+1} = \dots = r_{a_1+a_2}, \dots,$$

$$s_m = r_{a_1+a_2+\dots+a_{m-1}+1} = \dots = r_{a_1+a_2+\dots+a_m} = r_d$$

For instance: $F = \{(0, 0, 0, 0), (0, 1, 0, 1), (0, 0, 1, 1), (0, 1, 1, 0), (1, 0, 0, 0), (1, 1, 0, 1), (1, 0, 1, 1), (1, 1, 1, 0), (2, 1, 1, 1), (2, 0, 1, 0), (2, 1, 0, 0), (2, 0, 0, 1), (3, 1, 0, 0), (3, 1, 0, 1), (3, 1, 1, 0), (3, 1, 1, 1)\}$ is a $(4, 2^3)$ mixed fractional design.

It is usually considered that a fractional design is a matrix whose rows correspond to the elements of the multiset, in any order, and whose columns correspond to the factors. Therefore the example above becomes

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}^T,$$

where T denotes transpose. The rows of F are referred as *runs*, so the number of rows is the *run size*. For our purposes, the factor sets have no internal structure; we usually take $Q_i = Z_{r_i} = \{0, 1, \dots, r_i - 1\}$

A *subfraction* of F is obtained by choosing a subset of the factors (columns), and removing the other factors. A fraction is called *trivial* if it is a multiple of a full design, i.e, it contains every possible row with the same multiplicity.

Let t be a natural number. A fraction F is called *t-balanced* if, for each choice of t factors, the corresponding subfraction is trivial. In other words, every possible

combination of coordinate values from a set of t factors occurs equally often. The example above has strength 3 but not strength 4. A t -balanced fraction F is also called an orthogonal array of strength t . If F has N rows, we write

$$F = OA(N; s_1^{a_1} . s_2^{a_2} \dots s_m^{a_m}; t)$$

2.2 Enumerating orthogonal array

Certainly, the number of OAs with a given parameter set and run size is very big. To enumerate how many OAs with given parameter set (levels) and run size, a special method must be used, that is the concept of *isomorphic class*. For instance, the $OA(16;4.2^3;3)$ will be investigated. If two columns or two rows or two symbols of one column are exchanged, some new OAs will be created. These OAs are equivalent. These OAs and the original OA are in a group or in an isomorphic class. For each class, it is necessary to choose only one arbitrary member to represent for this class, and so, the problem "Enumerating OAs" becomes: counting how many isomorphic classes. In Figure 1, some $OA(16;4.2^3;3)$ belong to same isomorphic class, and certainly they are equivalent

<table border="1"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>D</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>2</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>3</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>3</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>0</td><td>1</td></tr> </tbody> </table>	A	B	C	D	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	1	1	1	0	2	1	1	1	2	0	1	0	2	1	0	0	2	0	0	1	3	1	1	1	3	0	1	0	3	1	0	0	3	0	0	1	<table border="1"> <thead> <tr><th>A</th><th>C</th><th>B</th><th>D</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>2</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>3</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>3</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>0</td><td>1</td></tr> </tbody> </table>	A	C	B	D	0	0	0	0	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	0	1	0	1	1	1	1	0	1	1	1	1	0	2	1	1	1	2	1	0	0	2	0	1	0	2	0	0	1	3	1	1	1	3	1	0	0	3	0	1	0	3	0	0	1	<table border="1"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>D</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>2</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>3</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>3</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>0</td><td>1</td></tr> </tbody> </table>	A	B	C	D	1	1	1	0	0	1	0	1	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	2	1	1	1	2	0	1	0	2	1	0	0	2	0	0	1	3	1	1	1	3	0	1	0	3	1	0	0	3	0	0	1	<table border="1"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>D</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>3</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>3</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>2</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> </tbody> </table>	A	B	C	D	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	0	3	0	0	0	3	1	0	1	3	0	1	1	3	1	1	0	2	1	1	1	2	0	1	0	2	1	0	0	2	0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	1	0	0	1
A	B	C	D																																																																																																																																																																																																																																																																																
0	0	0	0																																																																																																																																																																																																																																																																																
0	1	0	1																																																																																																																																																																																																																																																																																
0	0	1	1																																																																																																																																																																																																																																																																																
0	1	1	0																																																																																																																																																																																																																																																																																
1	0	0	0																																																																																																																																																																																																																																																																																
1	1	0	1																																																																																																																																																																																																																																																																																
1	0	1	1																																																																																																																																																																																																																																																																																
1	1	1	0																																																																																																																																																																																																																																																																																
2	1	1	1																																																																																																																																																																																																																																																																																
2	0	1	0																																																																																																																																																																																																																																																																																
2	1	0	0																																																																																																																																																																																																																																																																																
2	0	0	1																																																																																																																																																																																																																																																																																
3	1	1	1																																																																																																																																																																																																																																																																																
3	0	1	0																																																																																																																																																																																																																																																																																
3	1	0	0																																																																																																																																																																																																																																																																																
3	0	0	1																																																																																																																																																																																																																																																																																
A	C	B	D																																																																																																																																																																																																																																																																																
0	0	0	0																																																																																																																																																																																																																																																																																
0	0	1	1																																																																																																																																																																																																																																																																																
0	1	0	1																																																																																																																																																																																																																																																																																
0	1	1	0																																																																																																																																																																																																																																																																																
1	0	0	0																																																																																																																																																																																																																																																																																
1	0	1	1																																																																																																																																																																																																																																																																																
1	1	0	1																																																																																																																																																																																																																																																																																
1	1	1	0																																																																																																																																																																																																																																																																																
2	1	1	1																																																																																																																																																																																																																																																																																
2	1	0	0																																																																																																																																																																																																																																																																																
2	0	1	0																																																																																																																																																																																																																																																																																
2	0	0	1																																																																																																																																																																																																																																																																																
3	1	1	1																																																																																																																																																																																																																																																																																
3	1	0	0																																																																																																																																																																																																																																																																																
3	0	1	0																																																																																																																																																																																																																																																																																
3	0	0	1																																																																																																																																																																																																																																																																																
A	B	C	D																																																																																																																																																																																																																																																																																
1	1	1	0																																																																																																																																																																																																																																																																																
0	1	0	1																																																																																																																																																																																																																																																																																
0	0	1	1																																																																																																																																																																																																																																																																																
0	1	1	0																																																																																																																																																																																																																																																																																
1	0	0	0																																																																																																																																																																																																																																																																																
1	1	0	1																																																																																																																																																																																																																																																																																
1	0	1	1																																																																																																																																																																																																																																																																																
0	0	0	0																																																																																																																																																																																																																																																																																
2	1	1	1																																																																																																																																																																																																																																																																																
2	0	1	0																																																																																																																																																																																																																																																																																
2	1	0	0																																																																																																																																																																																																																																																																																
2	0	0	1																																																																																																																																																																																																																																																																																
3	1	1	1																																																																																																																																																																																																																																																																																
3	0	1	0																																																																																																																																																																																																																																																																																
3	1	0	0																																																																																																																																																																																																																																																																																
3	0	0	1																																																																																																																																																																																																																																																																																
A	B	C	D																																																																																																																																																																																																																																																																																
0	0	0	0																																																																																																																																																																																																																																																																																
0	1	0	1																																																																																																																																																																																																																																																																																
0	0	1	1																																																																																																																																																																																																																																																																																
0	1	1	0																																																																																																																																																																																																																																																																																
3	0	0	0																																																																																																																																																																																																																																																																																
3	1	0	1																																																																																																																																																																																																																																																																																
3	0	1	1																																																																																																																																																																																																																																																																																
3	1	1	0																																																																																																																																																																																																																																																																																
2	1	1	1																																																																																																																																																																																																																																																																																
2	0	1	0																																																																																																																																																																																																																																																																																
2	1	0	0																																																																																																																																																																																																																																																																																
2	0	0	1																																																																																																																																																																																																																																																																																
1	1	1	1																																																																																																																																																																																																																																																																																
1	0	1	0																																																																																																																																																																																																																																																																																
1	1	0	0																																																																																																																																																																																																																																																																																
1	0	0	1																																																																																																																																																																																																																																																																																
Original orthogonal array	Exchange two columns B, C	Exchange two rows (1 and 8)	Exchange two symbols 1 and 3 in first column																																																																																																																																																																																																																																																																																

Fig 1. Some equivalent orthogonal arrays of $OA(16;4.2^3;3)$

3 A lex-least algorithm for enumerating orthogonal array

3.1 Lex-least matrix

In this part, we will present a comparison metric of two arbitrary matrices. This comparison is called lexicographically less comparison [2].

For two vectors u and v of length N , we say u is lexicographically less than v , written $u < v$, if there exists an index $j \in \{1, \dots, N - 1\}$ such that $u[i] = v[i]$ for all $1 \leq i \leq j$ and $u[j + 1] < v[j + 1]$.

Let $F = [c_1, \dots, c_d]$, $F' = [c'_1, \dots, c'_d]$ be any pair of fractions where c_i, c'_i are columns. We say F is column-lexicographically less than F' , written $F < F'$, if and only if there exists an index $j \in \{1, \dots, d - 1\}$ such that $c_i = c'_i$ for all $1 \leq i \leq j$ and $c_{j+1} < c'_{j+1}$ lexicographically. The smallest matrix of an isomorphic class will be called *Lex-least matrix* and it will be the only representative of this isomorphic class.

3.2 Backtrack search extend a column

This backtrack search method extends a column by putting step-by-step each symbol on each cell of a new column. If at one cell, we have more than one value for which, we will continue to extend for a specific value after storing the all other values on a Stack. The algorithm will check the trivial feature of each vector t -element to ensure that all new matrices are orthogonal arrays of strength t . If at any time, we can't search for the next cell, we will push the value on the top of Stack and come back to extend for a new value. If we reach a new orthogonal array, we will check if this matrix is Lex-least matrix. If not, we will remove it.

Therefore the output of this algorithm is all Lex-least matrices of the new orthogonal arrays with a new column attached. If we have n Lex-least matrices, we will have n isomorphic classes because every Lex-least matrix is the only representative of one isomorphic class.

The following picture illustrates the idea of this algorithm:

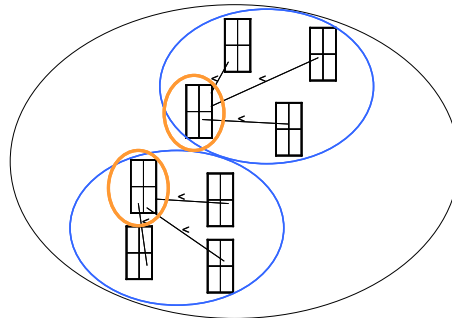


Fig 2. The idea of backtrack search extend a column algorithm

In above figure, the blue rings are two isomorphic classes. In each class, there are many equivalent matrices, but just only one Lex-least matrix (orange matrix), which is the smallest matrix, is chosen for representing every isomorphic class.

4 Reusing-Lex-Least-Matrices parallel algorithm

The motivation for this method stem from the fact that we found that all Lex-least matrices of $OA(16;3.2^4;3)$ can be used as all inputs to extend a new column for enumerating all Lex-least matrices of $OA(16;3.2^5;3)$. This reasoning follows from the lemma below:

Lemma 1: *It is assumed that one lex-least matrix of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$ is generated. If the last column of this matrix is removed, a lex-least matrix of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m}; t)$ will be created*

Proof:

This lemma is proved by contradiction.

If the result of the deletion is not a Lex-least matrix of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$, we can permute all column having the s_m level of this result matrix to get a new Lex-least of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$. If we add the deleted column to the new permuted matrix, we will have a new matrix of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$, and this new matrix will be smaller than the original matrix. This is unreasonable because the original matrix is a Lex-least matrix of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$. This contradiction proves the result.

The important impact of this lemma is that all Lex-least matrices of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$ can be used as all inputs to extend a new column for enumerating all lex-least matrices of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$.

The *Reusing-Lex-Least-Matrices* parallel algorithm is now specified with the model [9] as follow:

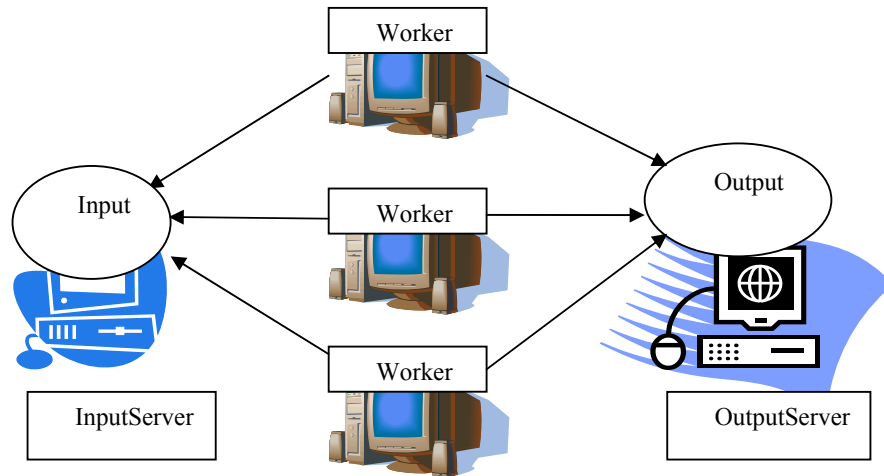


Fig. 3. Model of the Reusing-Lex-Least-Matrices parallel algorithm

In this algorithm, the *Server-Client* approach is used. A process named InputServer will read all inputs are the lex-least matrices of $OA(N; s_1^{a_1} .s_2^{a_2} \dots s_m^{a_m+1}; t)$ and distribute each input for each worker. Each worker after receiving one input matrix from InputServer will extend the input matrix for one column and send every result matrix generated to the OutputServer. If the worker finish enumerating for one matrix input, it will send request to InputServer and get a new matrix input from InputServer. This work repeats until InputServer has no input.

This method use *Data parallel* scheme. Each worker just requests one input from InputServer at a time, generates all matrix output, send every output to OutputServer (and go on). So we don't have any communication between workers, the communication takes place in InputServer and Worker (InputServer receiving request from any worker and transfer one matrix input to that worker, this work repeat until it has no input) and Worker-OutputServer (just receiving all Lex-least matrix outputs from every worker). In other words, the computing work mainly takes place in workers, the communication will takes place mainly between InputServer-Worker and between Worker-OutputServer.

5 Speedup and Efficiency

The *Reusing-Lex-Least-Matrices* parallel algorithm has been implemented using the Message Passing Interface. This algorithm has been executed on the Supernode II cluster of HCM University of Technology, Vietnam. The cluster has 64 dual-Xeon nodes (128 processors) with gigabit ethernet interconnected network. We allocated two processes for each node. To simplify the evaluation, 89 Lex-least-matrices of

OA(72;3.2⁴;3) have been chosen as inputs and we distribute inputs for each worker process for enumerating all non-isomorphism class of OA(72;3.2⁵;3). Finally, some results about the time execution, speedup and efficiency have been recorded as below:

Table 1. Execution time of the *Reusing-Lex-Least-Matrices* parallel algorithm.

Number of processes	Execution Time (minutes)
1	705' (sequence time)
10	109'
20	68'
30	56'

Table 2. Speedup and efficiency ratio of the *Reusing-Lex-Least-Matrices* parallel algorithm.

Number of processes	Speedup	Efficiency
10	6.46	0.646
20	10.37	0.519
30	12.6	0.42

With these results, we can express the trend of the speedup and efficiency by using the charts below:

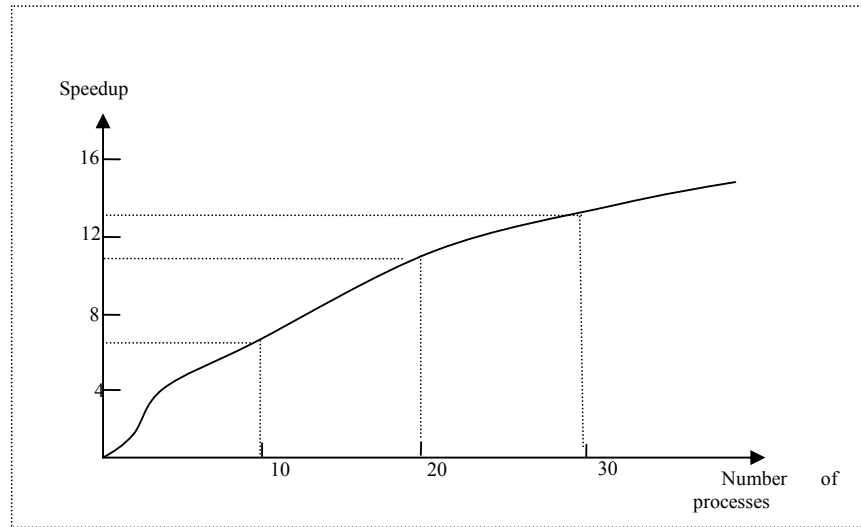


Fig. 4. Speedup chart.

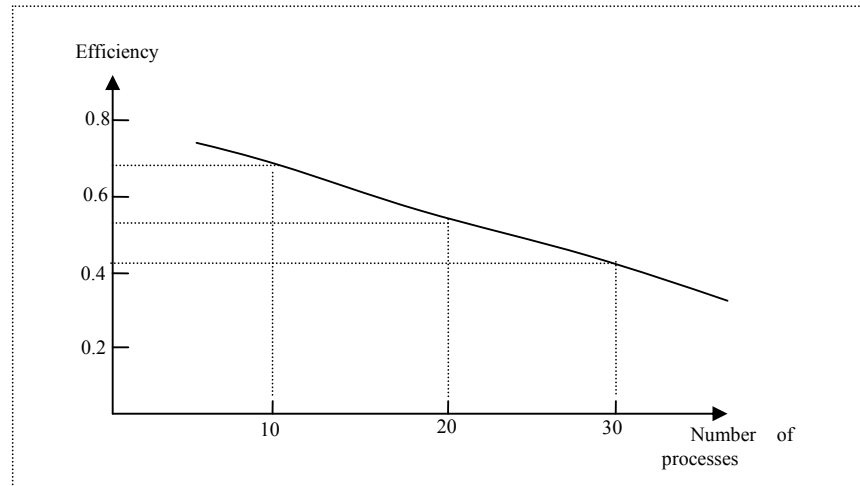


Fig. 5. Efficiency chart.

With the trend increase of speedup, we can see that this parallel algorithm is a useful method, but when we increase the number of processes, the efficiency ratio decrease. At this time, we think the reason of the decrease of efficiency when increase the number of processes is the bottleneck of output server because of receiving a lot of result matrices at a time.

6 Closing remark

This paper has proposed a new parallel method for enumerating orthogonal array of strength t . Currently, our method has been used for enumerating orthogonal arrays strength t where $t=3,4$ only. In particular, with $t=4$, the new ones are: $OA(96;3.2^6;4)$, $OA(96;3.2^7;4)$. We hope that in the near future, some new orthogonal arrays will be enumerated by this method.

References

- [1] N.J.A. Sloane. research.att.com/~njas/hadamard/, 2005
- [2] Nguyen, V. M. Man (2005), Computer-Algebraic Methods for the Construction of Designs of Experiments, Ph.D. thesis, <http://alexandria.tue.nl/extra2/200512795.pdf>, Technische Universiteit Eindhoven.
- [3] Nguyen, V. M. Man (2005), An online service for computing Hadamard matrices and strength 3 orthogonal arrays, www.mathdox.org/nguyen, Technische Universiteit Eindhoven.
- [4] Nguyen, V.M. Man (Jan. 2007), Some new constructions of strength 3 mixed orthogonal arrays, *Journal of Statistical Planning and Inference*, 138, pp. 220-233.

- [5] A. E. Brouwer, A. M. Cohen, and M. V. M Nguyen (2005), Orthogonal arrays of strength 3 and small run sizes. *Journal of Statistical Planning and Inference*, 136, pp. 3268-3280.
- [6] A. S. Hedayat, N. J. A. Sloane, and J. Stufken (1999), Orthogonal arrays. *Springer-Verlag, New York*.
- [7] Madhav, S. P. (2004), Design Of Experiment For Software Testing, www.isixsigma.com/library/content/c030106a.asp.
- [8] Nguyen, V. M. Man and Murray H. Scott (2006), Enumeration of strength t orthogonal arrays, preprint, *Ho-Chi-Minh City's University of Technology, Vietnam*
- [9] Yanjun Zhang (1989), Parallel Algorithms for Combinatorial Search Problems, PhD thesis, *University of Berkeley at California*
- [10] T.A. Nguyen, Marcelo Pasin, Pierre Kuonen (2006), Parallel Object Programming C++ User and Installation Manual, *Grid and Ubiquitous Computing Group*,
- [11] T.A. Nguyen, An Object-oriented Model for Adaptive High Performance Computing on the Computational Grid (2004). PhD thesis, *Swiss Federal Institute of Technology-Lausann*.
- [12] S.R. Dalai and al, Factor-covering designs for Testing Software, *Technometrics* 40(3) (1998) , 234-243, *American Statistical Association and the American Society for Quality*
- [13] M. F. Fecko and al, Combinatorial designs in Multiple faults localization for Battlefield networks (2001), *IEEE Military Communications Conf., Vienna*
- [14] Sudhir Gupta, Balanced Factorial Designs for cDNA Microarray Experiments (2006), *Communications in Statistics: Theory and Methods, Volume 35, Number 8, pp. 1469-1476*