Modern Applied Science; Vol. 7, No. 8; 2013
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

# New Permutation Generation Under Exchange Strategy (PGuES)

Sharmila Karim[1], Haslinda Ibrahim[1], Zurni Omar[1] & Nazihah Ahmad[1]

[1] School of Quantitative Sciences, College of Arts and Sciences, Universiti Utara Malaysia, Sintok, Kedah, Malaysia

Correspondence: Sharmila Karim, School of Quantitative Sciences, College of Arts and Sciences, Universiti Utara Malaysia, Sintok 06010, Kedah, Malaysia. Tel: 60-4-928-6972. E-mail: mila@uum.edu.my

Received: June 2, 2013 Accepted: June 20, 2013 Online Published: July 5, 2013

doi:10.5539/mas.v7n8p15 URL: http://dx.doi.org/10.5539/mas.v7n8p15

**Abstract**

An exchange based technique for generating permutation is presented in this paper. This strategy involved exchanging two consecutive elements, to generate the starter sets. For list all permutations, the exchange and reversing operations will be employed on these starter sets. The numerical result shows that new method is better than other existing methods.

**Keywords:** starter sets, permutation, exchanging two elements, exchange and reversing operations

## 1. Introduction

An arrangement of finite distinct element in particular order is called a permutation. There are various method have been published such as interchange (Heap, 1963; Fike, 1975), lexicographic order (Ord-Smith, 1970), transposition (Lispki & Warsaw, 1979), ranking and unranking (Zaks, 1984), random permutation (Gries & Xue, 1988), nested cycling (Iyer, 1995), shift cursor, and level (Gao & Wang, 2003), partial reversion (Shin, 2002; Thongchiew, 2007), Viktorov (2007), and Ibrahim et al. (2010).

Generating starter sets and exploited them for listing all permutation proposed by Ibrahim et al. (2010) in which they introduced a new permutation technique based on distinct starter sets by employing circular and reversing circular operation. The crucial task of Ibrahim et al. (2010) method was generating the distinct starter sets and eliminating the equivalence starter sets. Karim et al. (2011a, 2011b) proposed new operation strategies for generating distinct starter sets without generating the equivalence starter sets based on circular operation. Because the strategy of generating starter sets that is not unique, a new operation strategy namely exchange two (Karim et al., 2011b) proposed. At the end, all $n!$ distinct permutations is listed also under circular and reversing circular operation. In this paper, we generate the starter sets under exchange operation and for listing down all permutation, we employ exchange and reversing exchange operation.

## 2. Some Definitions

The following definitions will be used throughout this study.

**Definition 2.1** *Starter set, S is a basis to enumerate other permutations.*

**Definition 2.2** *An equivalence starter set is a set that can produce the same permutation from other starter set.*

**Definition 2.3** *A Latin square of order n is an n×n array in which n distinct symbols are arranged where each element occurs once in each row and column.*

**Definition 2.4** *The circular permutation of order n (CP) is a Latin square of order n which is obtained by employing the circular permutation operation over all elements.*

**Definition 2.5** *The reverse of circular permutation (RoCP) is also a Latin square of order n which is obtained by reversing arrangement element in each row of circular permutation.*

**Definition 2.6** *The exchange operation is a permutation that interchanges two integers k and l, $k \neq l$ , but leaves all other integers fixed.*

**Definition 2.7** *The reverse set is a set that is produced by reversing the order of permutation set.*

**Definition 2.8** *The exchange permutation of order n (EP) is an n×n array in which element in first position only are exchange to the right.*

**Definition 2.9** *The reverse of exchange permutation (RoEP) is an n×n array which is obtained by reversing arrangement element in each row of exchange permutation.*

## 3. Algorithm Development for Permutation

The development of new algorithm is divided into two stages. The first stage is starter sets generation while the second stage is permutation generation by exploiting the results in the first stage. The methods to generate ($n$-1)!/2 starter sets are based on exchange operations. For listing all $n!$ distinct permutations, the EP and RoEP operations will be employed. Thus it would be worthwhile to see the advantages of this operation in term of time computation.

### *3.1 Starter Sets Generation Under Exchange Based*

Let $S$ be the set of $n$ elements such that $S = [1,2,\ldots,n]$. Cases $n = 2$ and 3 are trivial. In this section, we illustrate how the algorithm works for cases $n = 4$ and $n = 5$ where it was generated recursively. The step by step of starter sets derivation is demonstrated for $n = 4$ and $n = 5$ as follows.

Step 1: Let [1,2,3,4] be an initial permutation and without loss of generality, the first element is fixed.

Step 2: Select element in the ($n$-2)th position i.e. element '2'. This element is exchange until it occupies the $n$th position. The three distinct starter sets are produces as follows:

| |
|---|
| 1 2 3 4 |
| 1 3 2 4 |
| 1 3 4 2 |

Figure 1. List of Starter Sets for $n = 4$

Next, we shall illustrate the second strategy for $n = 5$.

Step 1: Set [1,2,3,4,5] be an initial permutation and without loss of generality, the first element is fixed.

Step 2: Select the element in the (n-2)th position i.e. element '3'. This element is exchange until it occupies the $n$th position. The three distinct starter sets are produces as follows:

| |
|---|
| 1 2 3 4 5 |
| 1 2 4 3 5 |
| 1 2 4 5 3 |

Figure 2. Starter sets from the exchange of the ($n$-2)th element

Step 3: Select the element in the ($n$-3)th position i.e. '2' in each starter sets from Step 2. This element is exchange until it occupies the *nth* position. The other 12 distinct starter sets are produces as follows:

| |
|---|
| 1 2 3 4 5 |
| 1 3 2 4 5 |
| 1 3 4 2 5 |
| 1 3 4 5 2 |
| 1 2 4 3 5 |
| 1 4 2 3 5 |
| 1 4 3 2 5 |
| 1 4 3 5 2 |
| 12 4 5 3 |
| 1 4 2 5 3 |
| 1 4 5 2 3 |
| 1 4 5 3 2 |

Figure 3. All starter set for $n = 5$

*3.2 Permutation Generation Under Exchange and Reversing Operation*

In this section, we demonstrate how *n!* distinct permutations are listed by performing of EP (column A) and RoEP (column B) to each starter sets (see Figures 4 and 5 ).

Case $n = 4$.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **1** | **2** | **3** | **4** | 4 | 3 | 2 | 1 |
| 2 | 1 | 3 | 4 | 4 | 3 | 1 | 2 |
| 2 | 3 | 1 | 4 | 4 | 1 | 3 | 2 |
| 2 | 3 | 4 | 1 | 1 | 4 | 3 | 2 |
| **1** | **3** | **2** | **4** | 4 | 2 | 3 | 1 |
| 3 | 1 | 2 | 4 | 4 | 2 | 1 | 3 |
| 3 | 2 | 1 | 4 | 4 | 1 | 2 | 3 |
| 3 | 2 | 4 | 1 | 1 | 4 | 2 | 3 |
| **1** | **3** | **4** | **2** | 2 | 4 | 3 | 1 |
| 3 | 1 | 4 | 2 | 2 | 4 | 1 | 3 |
| 3 | 4 | 1 | 2 | 2 | 1 | 4 | 3 |
| 3 | 4 | 2 | 1 | 1 | 2 | 4 | 3 |

**column A** **column B**

Figure 4. List of 4! permutations

Case n = 5

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **4** | **5** | **3** | 3 | 5 | 4 | 2 | 1 |
| 2 | 1 | 4 | 5 | 3 | 3 | 5 | 4 | 1 | 2 |
| 2 | 4 | 1 | 5 | 3 | 3 | 5 | 1 | 4 | 2 |
| 2 | 4 | 5 | 1 | 3 | 3 | 1 | 5 | 4 | 2 |
| 2 | 4 | 5 | 3 | 1 | 1 | 3 | 5 | 4 | 2 |
| **1** | **4** | **2** | **5** | **3** | 3 | 5 | 2 | 4 | 1 |
| 4 | 1 | 2 | 5 | 3 | 3 | 5 | 2 | 1 | 4 |
| 4 | 2 | 1 | 5 | 3 | 3 | 5 | 1 | 2 | 4 |
| 4 | 2 | 5 | 1 | 3 | 3 | 1 | 5 | 2 | 4 |
| 4 | 2 | 5 | 3 | 1 | 1 | 3 | 5 | 2 | 4 |
| **1** | **4** | **5** | **2** | **3** | 3 | 2 | 5 | 4 | 1 |
| 4 | 1 | 5 | 2 | 3 | 3 | 2 | 5 | 1 | 4 |
| 4 | 5 | 1 | 2 | 3 | 3 | 2 | 1 | 5 | 4 |
| 4 | 5 | 2 | 1 | 3 | 3 | 1 | 2 | 5 | 4 |
| 4 | 5 | 2 | 3 | 1 | 1 | 3 | 2 | 5 | 4 |
| **1** | **4** | **5** | **3** | **2** | 2 | 3 | 5 | 4 | 1 |
| 4 | 1 | 5 | 3 | 2 | 2 | 3 | 5 | 1 | 4 |
| 4 | 5 | 1 | 3 | 2 | 2 | 3 | 1 | 5 | 4 |
| 4 | 5 | 3 | 1 | 2 | 2 | 1 | 3 | 5 | 4 |
| 4 | 5 | 3 | 2 | 1 | 1 | 2 | 3 | 5 | 4 |
| **1** | **2** | **3** | **4** | **5** | 5 | 4 | 3 | 2 | 1 |
| 2 | 1 | 3 | 4 | 5 | 5 | 4 | 3 | 1 | 2 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 1 | 4 | 5 | | 5 | 4 | 1 | 3 | 2 |
| 2 | 3 | 4 | 1 | 5 | | 5 | 1 | 4 | 3 | 2 |
| 2 | 3 | 4 | 5 | 1 | | 1 | 5 | 4 | 3 | 2 |
| **1** | **3** | **2** | **4** | **5** | | 5 | 4 | 2 | 3 | 1 |
| 3 | 1 | 2 | 4 | 5 | | 5 | 4 | 2 | 1 | 3 |
| 3 | 2 | 1 | 4 | 5 | | 5 | 4 | 1 | 2 | 3 |
| 3 | 2 | 4 | 1 | 5 | | 5 | 1 | 4 | 2 | 3 |
| 3 | 2 | 4 | 5 | 1 | | 1 | 5 | 4 | 2 | 3 |
| column A | | | | | | column B | | | | |

Figure 5. List of 5! permutations

As can be observed from Figures 4 and 5, there no redundancy permutation occurs when the starter sets are exploited for generating all permutations by employing EP and RoEP operations.

**Remark 3.1** *The **bold** permutation in **Column A** represents the starter sets.*

*3.3 Exchange Algorithm*

Let define the algorithm for exchange operation strategy as a PGuES which is a recursion algorithm with only one recursive call. The integer $n$ be an initial inputs, and temp = $n$-1 is an initial rank of the algorithm PGuES. At each stage of recursion, a rank for procedure PGuES is reduced from $n$-1 to 2 where each recursion call updates the entries in the storage. The general algorithm for permutation generation by employing EP and RoEP operation on starter sets which was generated from the exchange operation is as follows:

Algorithm 1 PGuES

```
PGuES(temp)
if temp = 2 then
for i = 1 to n do
        performing EP and RoEP operations for all element
        end for
return
end if
temp = temp + 1
for i = temp to n do
performing exchanges operation to the element at temp-th position call PGuES(temp)
end for
```

The comparison among our methods for permutation generation is presented in the following section.

*3.4 Differences Among Our Methods*

The two algorithms were published namely PERMUT1 and PERMUT2. The PERMUT1 and PERMUT2 are an algorithm where starter sets was generated by circular operation and exchange operation respectively. Then for listing all permutation, both algorithms employed CP and RoCP operations on their starter sets. Given the example for $n$ = 4.

Table 1. Starter sets generation for $n = 4$

| Algorithm | Strategy | Starter Sets |
|---|---|---|
| PERMUT1 | Circular | [1,2,3,4], [1,3,4,2], [1,4,2,3] |
| PERMUT2 | Exchange | [1,2,3,4], [1,3,2,4], [1,3,4,2] |
| PGuES | Exchange | [1,2,3,4], [1,3,2,4], [1,3,4,2] |

Next is example of listing 4! permutation for PERMUT1 and PERMUT2 algorithms.

| **Column A** | | | | **Column B** | | | |
|---|---|---|---|---|---|---|---|
| **1** | **2** | **3** | **4** | 4 | 3 | 2 | 1 |
| 2 | 3 | 4 | 1 | 1 | 4 | 3 | 2 |
| 3 | 4 | 1 | 2 | 2 | 1 | 4 | 3 |
| 4 | 1 | 2 | 3 | 3 | 2 | 1 | 4 |
| **1** | **3** | **4** | **2** | 2 | 4 | 3 | 1 |
| 3 | 4 | 2 | 1 | 1 | 2 | 4 | 3 |
| 4 | 2 | 1 | 3 | 3 | 1 | 2 | 4 |
| 2 | 1 | 3 | 4 | 4 | 3 | 1 | 2 |
| **1** | **4** | **2** | **3** | 3 | 2 | 4 | 1 |
| 4 | 2 | 3 | 1 | 1 | 3 | 2 | 4 |
| 2 | 3 | 1 | 4 | 4 | 1 | 3 | 2 |
| 3 | 1 | 4 | 2 | 2 | 4 | 1 | 3 |

Figure 6. List of 4! permutations for PERMUT1

| **Column A** | | | | **Column B** | | | |
|---|---|---|---|---|---|---|---|
| **1** | **2** | **3** | **4** | 4 | 3 | 2 | 1 |
| 2 | 3 | 4 | 1 | 1 | 4 | 3 | 2 |
| 3 | 4 | 1 | 2 | 2 | 1 | 4 | 3 |
| 4 | 1 | 2 | 3 | 3 | 2 | 1 | 4 |
| **1** | **3** | **2** | **4** | 4 | 2 | 3 | 1 |
| 3 | 2 | 4 | 1 | 1 | 4 | 2 | 3 |
| 2 | 4 | 1 | 3 | 3 | 1 | 4 | 2 |
| 4 | 1 | 3 | 2 | 2 | 3 | 1 | 4 |
| **1** | **3** | **4** | **2** | 2 | 4 | 3 | 1 |
| 3 | 4 | 2 | 1 | 1 | 2 | 4 | 3 |
| 4 | 2 | 1 | 3 | 3 | 1 | 2 | 4 |
| 2 | 1 | 3 | 4 | 4 | 3 | 1 | 2 |

Figure 7. List of 4! permutations for PERMUT2

From Table 1, the starter sets for PERMUT2 and PGuESS are similar. However as can be observed from Figures 4 and 7, for listing all permutation, PERMUT2 employed CP and RoCP meanwhile PGuES employed EP and RoEP.

## 4. Numerical Results

For permutation generation algorithm performance, firstly our algorithm is compared to our previous PERMUT1 and PERMUT2 algorithms. Then it will be compared to other permutation generation program namely Langdon

(1967), and Thongchiew (2007). All sequential algorithms are implemented in C language and tested on the HP Computer with Intel Xeon E5504 2.0 GHz processor and 4.00 GB Random Access Memory (RAM). The result is an execution time without printing statements.

Table 2. The computation time between new algorithm based on starter sets (in seconds)

| *n* | PGuES | PERMUT1 | PERMUT2 |
|---|---|---|---|
| 6 | 0.000007 | 0.000015 | 0.000014 |
| 7 | 0.000040 | 0.000133 | 0.000104 |
| 8 | 0.000299 | 0.000933 | 0.000892 |
| 9 | 0.002555 | 0.009631 | 0.008799 |
| 10 | 0.033888 | 0.128964 | 0.126090 |
| 11 | 0.279070 | 1.048040 | 0.912727 |
| 12 | 3.211444 | 12.599008 | 11.716310 |
| 13 | 40.501254 | 172.164892 | 163.756450 |
| 14 | 560.528134 | 2555.933766 | 2448.900501 |

Table 2 indicates that our new algorithm, PGuES is better than our previous PERMUT1 and PERMUT2. The following table shows the results of run times among new algorithm for permutation generation.

Table 3. The computation time between new algorithm, Langdon (1967) and Thongchiew (2007) (in seconds)

| *n* | PGuES | Langdon | Thongchiew |
|---|---|---|---|
| 6 | 0.000007 | 0.000031 | 0.000141 |
| 7 | 0.000040 | 0.000230 | 0.000968 |
| 8 | 0.000299 | 0.001969 | 0.007806 |
| 9 | 0.002555 | 0.019642 | 0.070213 |
| 10 | 0.033888 | 0.201305 | 0.698074 |
| 11 | 0.279070 | 2.040842 | 7.681903 |
| 12 | 3.211444 | 26.715958 | 92.246500 |
| 13 | 40.501254 | 367.243559 | 1199.657668 |
| 14 | 560.528134 | 2555.933766 | 1953251.000000 |

From Table 3, we observe that PGuES is faster than Langdon (1967) and Thongchiew (2007). From Figures 4, 5, 6 and 7, an advantage of our method is we able to list $n!/2$ permutations (Column B) by reversing of the first $n!/2$ permutations (Column A).

## 5. Conclusion

The central idea of our work for listing permutation is a starter sets generation. The a new strategy to generate the starter sets is presented under exchange operation. Then exchange strategy is exploited to generate all $n!$ permutations using EP and RoEP operations.

**Acknowledgements**

This research is supported by University grant (Code S/O: 12260) from Universiti Utara Malaysia.

**References**

Fike, C. T. (1975). A permutation generation method. *The Computer Journal, 18*(1), 21-22. http://dx.doi.org/10.1093/comjnl/18.1.21

Gao, J., & Wang, D. (2003). *Permutation generation: Two new permutation algorithms*. Retrieved from http://arxiv.org/abs/cs/0306025

Gries, D., & Xue, J. (1988). Generating a random cyclic permutation. *BIT*, 569-572.

Heap, B. R. (1963). Permutations by interchanges. *Computer J., 6*, 293-294. http://dx.doi.org/10.1093/comjnl/6.3.293

Ibrahim, H., Omar, Z., & Rohni, A. M. (2010). New algorithm for listing all permutations. *Modern Applied Science, 4*(2), 89-94.

Iyer, M. (1995). *Permutation generation using matrices*. Retrieved from http://www.ddj.com/184409671

Karim, S., Ibrahim, H., Omar, Z., & Othman, K. I. (2011a). Parallel Strategy for Starter Sets to List All Permutation Based on Cycling Restriction. *Proceeding of The Third International Conference Engineering and Technology*, 251-256.

Karim, S., Omar, Z., & Ibrahim, H. (2011b). Integrated strategy for generating permutation. *International Journal of Contemporary Mathematical Sciences, 6*(24), 1167 - 1174.

Langdon, G. (1967). An Algorithm for generating permutations. *Communication of ACM*, 298-299. http://dx.doi.org/10.1145/363282.363315

Lipski, W., & Warsaw, J. (1979). More on permutation generation method. *Computing*, 357-365. http://dx.doi.org/10.1007/BF02254864

Ord-Smith, J. (1970). Generation of permutation sequences Part 1. *Computer Journal*, 152-155.

Shin, D. (2002). The permutation algorithm for non-sparse matrix determinant in symbolic computation. Proceeding on the 15th CISL winter workshop, pp. 42-54. Japan.

Thongchiew, K. (2007). A computerize algorithm for. *Proc.of World Academy of Science, Engineering and Technology*, pp. 178-183.

Viktorov, O. (2007). Permutation generation algorithm. *Asian Journal of Information Technology*, 956-957.

Zaks, S. (1984). A new algorithm for generation of permutations. *BIT*, 196-204.