A GA-based Systematic Reasoning Approach for Solving Traveling Salesman Problems Using an Orthogonal Array-based Crossover

Shinn-Ying Ho (何信瑩)
Department of Information Engineering,
Feng Chia University
syho@fcu.edu.t

Jian-Hung Chen (陳建宏)
Department of Information Engineering,
Feng Chia University
jhchen@plum.iecs.fcu.edu.tw

Abstract

This paper proposes a novel genetic algorithm-based systematic reasoning approach using an orthogonal array-based crossover (OABX) for solving the traveling salesman problem (TSP). OABX makes use of the systematic reasoning ability of orthogonal arrays that can effectively preserve superior sub-paths from parents and guide the soluti on towards better quality. It is shown empirically that OAB outperforms various superior crossovers of canonical approach in both accuracy and speed.

***Keywords**: Algorithms; traveling salesman problem; genetic algorithms; systematic reasoning; orthogonal array-based crossove .*

1. Introduction

The traveling salesman problem (TSP) is one of the most widely discussed combinatorial optimization problems that belong to the class of NP-complete problems [1][8]. The definition of the TSP is that a salesman, starting from his home city and traveling each city exactly once before returning home, is to seek the shortest tour through n cities. The TSP mainly finds the visitation order which minimizes the total traveled distance. Mathematically, given a sequence of cities, c_1, c_2, …, and c_n, and inter-cit distances $d(c_i,c_j)$, find a permutation π of the cities that minimizes the sum of distances C(π) [4][8].

$$C(\pi)=\sum_{i=1}^{n-1}d(C_{\pi(i)},C_{\pi(i+1)})+d(C_{\pi(n)},C_{\pi(1)}) \qquad (1)$$

Due to the complexity of the large TSP, the TSP has received considerable attentions recently in developing efficient approaches for solving the lager and larger TSP [1][8]. Genetic Algorithm (GA), developed b Hollan , has been proved as a powerful tool for solving various optimization problems [3][4]. Several approaches based on GA have been proposed for solving the TSP [1][2][7][12]. Since genetic crossover is the main operator of GA. Therefore, good genetic crossover is essential to make a GA-based search effective and obtain a better solution. However, traditional crossovers use either random combination of parents ' sub-path or greedy mechanism with additional information.

In this paper, we present an intelligent orthogonal array-based crossover (OABX) with the permutation representation which is the most natural representation of a TSP tour. OABX is a systematic reasoning crossover that easily obtains and preserves the superior sub-paths of a chromosome without using any greedy approach. Encouraging computational results demonstrate that OABX outperforms the existing superior crossovers of canonical approach.

The remainder of this work is organized as follows. S ection 2 presents preliminary analysis of GA crossovers for solving the TSP. The use of orthogonal arrays (OAs) to achieve OABX and the GA-based algorithm using OABX are given in Section 3. Computational experiments and statistical analysis of results are presented in Section 4. summary of the stud is made in Section 5.

2. Preliminary Analysis

In this section, we discuss about the role of the GA crossover and examine various GA crossovers for the TSP in literature.

2.1 Role of Crossover

Crossover is the main genetic operator of GA, mainly because crossover tends to perform widespread search for exploiting all solution space and provides exploration in the neighborhood [3][4]. Therefore, a good crossover that leads to better solutions within short computation time is indispensable. Generally speaking, the role of the crossover is to recombine information from good parent solutions into offspring solutions what we hope are even better [3][7]. In summary, the performance of crossovers can be evaluated using the solution qualit and the computation time that crossover spends.

2.2 Various Crossovers for the TSP

During the past decade, in order to solve the TSP effectively, various crossovers have been suggested in literature. These crossovers can be classified into two approaches [4]:

(1) Canonical approach: the essence of the canonical approach is the blind random mechanism. These crossovers need no additional information and do not cost extremely long time, but there is no guarantee that an offspring is better than their parents. Nevertheless, these crossovers are general purposed for solving some ordering problems. These crossovers include PMX, OX, OX2, PBX, CX, IX, UX, UX2, EER, and MX, etc. [4][7][10]. Among the above-mentione crossovers for the TSP, OX, EER and UX2 have demonstrated that their overall performance outperforms other crossovers [7][10].

(2) Heuristic approach: the essence of the heuristic approach is that these crossovers incorporate heuristic information or the greedy mechanism. Although this approa ch intends to generate an improved offspring, it needs additional information or costs extremely long time to make the improvement. Among the conventional heuristics for the TSP, there are two basic construction methods the nearest neighbor and the best insertion heuristics [4]. These crossovers includes EAX [13], heuristic crossover [4]. These crossovers are special purposed for solving TSP.

OABX makes use of the systematic reasoning ability of orthogonal arrays that can effectively analyze the impact factor of each sub -path in parents and efficiently guide the solution towards better quality without using any greedy mechanism or additional information. Therefore, the proposed OABX, which belongs to neither canonical approach nor heuristic approach, can be categorized into the third approach: a novel systematic reasoning approach. Moreover, OABX can be incorporated with other GA-based algorithms with heuristics, such as best insertion or k nearest neighbors heuristics. It follow that OABX combines the advantages of two traditional approaches in both accuracy and speed In this study, OABX is compared with the superior crossovers of canonical approach, OX, EER and UX2 [7][10].

3. Genetic Algorithm using OABX

The principle of OABX relies on OAs which are described in Section 3.1. The use of OAs to achieve an intelligent OABX with an illustration for solving the TSP are described in Section 3.2 . The simple GA we applied is provided in Section 3.3.

3.1 Orthogonal Array and Factor Analysis

Orthogonal Array (OA) and factor analysis, which are representative methods of quality control [11], also work to improve the crossover efficiently. The superiority of O in obtaining better results for large parameter optimization problems has bee demonstrated [5][6]. The definition of OA is as follows. Let there be N factors of two levels. The number of total combinations is 2^N. Columns of two factors are orthogonal when 4 pairs, (1,1), (1,2), (2,1), and (2,2), occur equally in all experiments. When any two factors in an experimental set are orthogonal, the set is called an OA. To establish an OA for use of N factors of two levels, we obtain an integer $n = 2^{\lceil \log(N+1) \rceil}$, build an orthogonal arra $L_n(2^{n-1})$ with n rows and (n-1) columns, and select N columns.

Factor analysis can evaluate the effects of factors on the evaluation function, rank the most effective factors, and determine the best level for each factor such that the evaluation function is optimized. Orthogonal experiment design can reduce the number of experiments for the factor analysis. The number of OA experiments for single factor analysis is only n. For instance, Table 1 illustrates an orthogonal array $L_8(2^7)$.

Table 1. Orthogonal arra $L_8(2^7)$

Exp. no.	Factors							Function Evaluation value
	1	2	3	4	5	6	7	
1	1	1	1	1	1	1	1	y_1
2	1	1	1	2	2	2	2	y_2
3	1	2	2	1	1	2	2	y_3
4	1	2	2	2	2	1	1	y_4
5	2	1	2	1	2	1	2	y_5
6	2	1	2	2	1	2	1	y_6
7	2	2	1	1	2	2	1	y_7
8	2	2	1	2	1	1	2	y_8

Let y_t be the positive function evaluation value of experiment no. t. Define the main effect of factor j with level k S_{jk},

$$S_{jk} = \sum_{t=1}^{n} Y_t^2 \times [\textit{the level of Exp no t of factor j is k}] \quad (2)$$

where

$$[\textit{condition}] = \begin{cases} 1 & \textit{if the condition is true} \\ 0 & \textit{oterwise} \end{cases},$$

and

$$Y_t = \begin{cases} y_t & \text{if the function is to be maximized} \\ 1/y_t & \text{if the function is to be minimized} \end{cases}$$

Note that the main effect reveals the individual effect of a factor. The most effective factor j has the largest main effect difference (MED) $|S_{j1} - S_{j2}|$. If $S_{j1} > S_{j2}$, the level 1 of factor j is better than the level 2 on the contribution for the optimization function. Otherwise, level 2 is better

3.2 Orthogonal Array-Based Crossover

The representation of a chromosome for OABX approach adopts the permutation representation. That two parents breed two children using OABX consists of the following two procedures: OA procedure and repairing procedure.

3.2.1 OA procedure

The OA procedure of OABX is as follows

Step 1: Select the first N columns of OA $L_n(2^{n-1})$ where $n = 2^{\lceil \log(N+1) \rceil}$. Note that let the chromosome be uniformly separated into N sub-paths and each sub-path of a chromosome be regarded as a factor in OA.

Step 2: Let level 1 and level 2 of factor j represent the j^th sub-path of a chromosome coming from the parent 1 and parent 2, respectively.

Step 3: Evaluate the function values y_t for experiment no. t where t = 1, 2, …, n. Note that y_t is the tour length and is equal to the C(π) in equation (1).

Step 4: Compute the main effect S_{jk} where j = 1 2, …, N and k = 1, 2.

Step 5: Rank the most effective factors from rank 1 to rank N by comparing S_{jk}

Step 6: The better factors, from rank 1 to rank $\lceil N/2 \rceil$, are preserved in their correspondin positions to form the chromosome of children.

Notably, pre-computed function evaluation value of each sub -path can be recorded to accelerate the computation time in Step 3. Hereafter, the function values y_t can be fast obtained by summarizing the evaluation values of the pre-computed sub-paths and the distances betwee all neighboring sub-path pairs using a table-looking method. Moreover, the size of OA $L_n(2^{n-1})$ can be adjusted according to the problem size to analyze different factors. The larger N used, the smaller sub-path representing a factor in a chromosome will be analyzed. Generally, considering the computation time of one generation and the effect of OAs, let the number of N varies from 3 to 31 depending on the problem size.

OA procedure preserves the more contributive sub-path (factor) in the parents' chromosomes rather than the random preservation of the parents' sub-paths, such as OX. After OA procedure, two proto-childre are generated, however, they are not feasible representation of a tour. It follows that a repairing procedure is essential to embed in this crossover, in order to obtain a feasible representation of a tour. In this study, we propose a general repairing procedure for repairing the offspring.

3.2.2 General Repairing Procedure

The general repairing procedure is used for general case without using additional heuristic information. The repairing procedure is described as the following steps:

Step 1: (deletion step) Delete the cities which are already in the proto -child from the second parent. The resulted sequence of cities contains the cities that proto-child needs.

Step 2: (fill step) Place the cities into the unfixed positions of the proto-child from left to right according to the order of the sequence to produce an offspring.

Finally, a good feasible offspring can be obtained through the repairing procedure based on the systematic analysis of OAs.

3.2.3 Illustration by an Example

In this section, OABX is illustrated by a conci se example, as shown in Table 2 and Table 3. Let P1 and P2 be parents, C1' and C2' be the proto-children after OA procedure, and C1 and C2 be the offspring after the entire OABX operation. Let the problem size be 28 and N = 7. Therefore, the length of sub-path is 28/7 = 4, and the preserved sub-paths by OA procedure will be from rank 1 to rank $\lceil N/2 \rceil$ (=4). Let

P1: (2,11,12,5,1,15,10,13,8,3,19,20,0,16,9,25,18,23, 17,4,7,6,22,24,26,14,27,21).

P2: (8,17,26,19,13,6,21,20,23,18,10,25,0,27,2,14, 15, 5,11,7,22,1,24,4,16,12,9,3).

Table 2. Results of various procedure. Notably, –1 denotes the position which is left unfixed.

Corresponding sub-paths	1	2	3	4	5	6	7
P1	2,11,12,5	1,15,10,13	8,3,19,20	0,16,9,25	18,23,17,4	7,6,22,24	26,14,27,21
P2	8,17,26,19	13,6,21,20	23,18,10,25	0,27,2,14	15,5,11,7	22,1,24,4	16,12,9,3
C1'	-1,-1,-1,-1	1,15,10,13	8,3,19,20	-1,-1,-1,-1	18,23,17,4	7,6,22,24	-1,-1,-1,-1
C2'	8,17,26,19	-1,-1,-1,-1	-1,-1,-1,-1	0,27,2,14	15,5,11,7	-1,-1,-1,-1	16,12,9,3
C1	26,21,25,0	1,15,10,13	8,3,19,20	27,2,14,5	18,23,17,4	7,6,22,24	11,16,12,9
C2	8,17,26,19	1,10,13,20	25,18,23,4	0,27,2,14	15,5,11,7	6,22,24,21	16,12,9,3

Table 3. OA $L_8(2^7)$ and factor analysis.

	Factor j							
Exp. No.	1	2	3	4	5	6	7	y_t
1	1	1	1	1	1	1	1	2702
2	1	1	1	2	2	2	2	2568
3	1	2	2	1	1	2	2	2748
4	1	2	2	2	2	1	1	2953
5	2	1	2	1	2	1	2	2498
6	2	1	2	2	1	2	1	2868
7	2	2	1	1	2	2	1	2953
8	2	2	1	2	1	1	2	2501
S_{j1} $(*10^{-8})$	53.571	$\boxed{57.044}$	$\boxed{56.316}$	54.433	$\boxed{55.083}$	$\boxed{57.178}$	48.790	
Rank	6	2	3	5	4	1	7	
S_{j2} $(*10^{-8})$	$\boxed{55.638}$	52.165	52.893	$\boxed{54.776}$	$\boxed{54.126}$	52.031	$\boxed{60.419}$	
Rank	2	6	5	3	4	7	1	

First, OA $L_8(2^7)$ is used. The values of all variables for factor analysis are shown in Table 3. For instance, in the 2nd experiment, the function evaluation value y_2 can be obtained from the cities in the 1st, 2nd and 3rd sub-paths of P1, and the 4th, 5th, 6th and 7th sub-paths of P2. The main effect S_{21} can be obtained from eqn. (2), $S_{21} = y_1^{-2} + y_2^{-2} + y_5^{-2} + y_6^{-2}$. The child C1' can be derived from preserving the 2nd, 3rd, 5th and 6th sub-paths of P1, and the child C2' can be derived from preserving the 1st, 4th, 5th and 7th sub-paths of P2. Second, perform the repairing procedure. For instance, in the deletion step, the resulted sequence of cities that C1 needs is (P2 – C1'), (26,21,25,0,27,2,14,5,11,16,12,9). In the fill step, the resulted sequence of cities are filled in the unfixed positions of C1' in the same order. The results for various procedures are shown in Table 2 in corresponding sub-path order.

3.3. Simple Genetic Algorithm

The simple genetic algorithm (SGA) [3] applied in the performance comparison for various superior crossovers is as follows

Step 1: Initialize a population of chromosomes.

Step 2: Evaluate each chromosome in the population.

Step 3: Generate offspring by selecting parents and applying crossover and inverse mutation.

Step 4: Evaluate the new chromosomes and insert them into the population.

Step 5: If stopping criteria are met, return the best. Otherwise, go to Step 3.

4. Comparisons of Performance Evaluations

In order to demonstrate the superiority of OA X, we compare OABX's performance with those of the most outstanding crossovers, EER, O and UX2, which had already been examined by literature [7][10]. Wit regard to examine the performance of those crossovers in justice, we implement them by the following rules:

(1) Implement all the crossovers using SGA;

(2) In order to observer various crossove 's ability and efficiency, we use the population size 10 to eliminate the influence of population size [3]. The parameters of SGA are $P_s = 0.3$, $P_c = 0.7$ and $P_m = 0.3$; and

(3) Examine all crossovers in solving the large TS for the purpose of observing their ability and efficiency.

All the tested benchmarks are obtained from

TSPLIB [9] and the optimum solution s have already been presented in literature. The comparison results by averaging crossover computation time are obtained using Pentium 166 CPU computer with instance pr2392 which consists of 2392 cities, as shown in Table 5. All the simulation results using t wenty independent runs for each instance are summarized in Table 4. Fig. 1 illustrates the comparisons of convergence speed and accuracy where OABX(num) means that OABX with an OA $L_{num}\left(2^{num-1}\right)$ is used. From the experimental results, it can be obviously seen that

(1) The quality of solutions obtained by OABX is superior to that of other outstanding crossovers, especially when the problem sizes become larger and larger;

(2) OABX outperforms the investigated crossovers in convergence speed and accuracy, especially when the heuristic method is used, as shown in Fig. 1; and

(3) The computation time of OABX is within reasonable time.

5. Conclusions

In this paper, we have proposed a novel genetic algorithm-based systematic reasoning approach using an orthogonal arra -based crossover (OABX) for solving the traveling salesman probl em (TSP). And it can be easily seen that OABX proceeds almost the same as OX, however, OABX preserves the better sub-paths rather than OX preserves sub -paths randomly. The encouraging results demonstrate that OABX combines the two traditional approaches' advantages, listed as follows

(1) OABX makes use of the systematic reasoning ability of orthogonal arrays can effectively preserve superior order information from parents and guide the solution towards better quality within reasonable computation time.

(2) OABX obtains better solutions without using any greedy mechanism or additional information.

(3) OABX is an efficient and general-purposed crossover, and OABX can easily incorporate with other heuristic methods such as best insertion or k nearest neighborhood.

It has been shown empirically that OABX outperforms various crossovers in both accuracy and convergence speed.

References

[1] Aarts, E. A., Lenstra, J. K., *Local Search in Combinatorial Optimization,* Wiley and Sons, New York, 1997.

[2] Chatterjee, S., Carrera, C., Lynch, L. A., Genetic Algorithms and Traveling Salesman Problems, *European Journal of Operational Research 93*, pp. 490-510, 1996.

[3] Davis, L., *Handbook of Genetic Algorith* , Van Nostrand Reinhold, New York, 1991.

[4] Gen, M., Cheng, R *Genetic Algorithms and Engineering Design,* Wiley and Sons, New York, 1997.

[5] Ho, Shinn-Ying., Shu, L -Sun., Chen, Hung-Ming., Intelligent Genetic Algorithm with a New Intelligent Crossover Using Orthogonal Arrays, *GECCO-99: Proc. of the Genetic and Evolutionary Computation Conf.,* Orlando, Florida, USA, vol 1, pp. 289-296 , July 14-17, 1999.

[6] Ho, Shinn-Ying., Chen, Hung-Ming, Shu, Li-Sun., Solving Large Knowledge Base Partitioning Problems Using an Intelligent Genetic Algorithm, *GECCO-99: Proc. of the Genetic and Evolutionary Computation C onf.,* Orlando, Florida, USA, vol 1, pp. 871-878, July 14-17, 1999.

[7] Poon, W. P., Carter J. N., Genetic Algorithm Crossover Operators for Ordering Applications , *Computers and Operations Research,* vol. 22, no. 1, pp. 135-147, 1995.

[8] Reinelt, G., *The Traveling Salesman.* Springer-Verlag, 1994.

[9] Reinelt, G., TSPLIB WebPage, http://www.iwr.uniheidelberg.de/iwr/comopt/soft/TSPLIB98/TSPLIB.html, 1998.

[10] Starkweather, T., McDaniel, S., Mathias, K., Whitley, D., A Comparison of Genetic Sequencing Operators, *Proc. Fourth Int. Conf. Genetic Algorithms and their Applications* , pp. 69-76, 1991.

[11] Taguch, G., Konishi S., *Orthogonal Arrays and Linear graphs,* Dearbon, MI: American Supplier Institute, 1987.

[12] Voudouris, C, Tsang, E., Guided Local Search and its Application to the Tra veling Salesman Problem, *European Journal of Operational Research* 113, pp. 469-499, 1999.

[13] Nagata, Y. and Kobayashi, S., Edge Assembly Crossover:A High-Power Gebetic Algorithm for the Traveling Salesman Problem, *Proc. of ICGA 97,* pp. 450-457, 1997.

Table 4. Simulation results of various crossovers

Data Sets	Crossover	Gen.	Average	Best/Worst	Rank
Lin105 (optimum =14379)	OABX(15)	10000	15463.266	14683/16372	1
	EER	10000	15995.667	14885/16971	2
	OX	10000	16029.400	15136/17171	3
	UX2	10000	16212.800	15462/16923	4
Lin318 (optimum =42029)	OABX(15)	30000	56718.233	54617/59726	1
	EER	30000	59428.168	56368/64023	3
	OX	30000	58943.332	55890/62068	2
	UX2	30000	61411.435	57828/64613	4
Pr439 (optimum =107217)	OABX(15)	30000	186311.127	171573/196527	1
	EER	30000	197833.563	187646/213718	3
	OX	30000	194584.297	186008/204981	2
	UX2	30000	203932.297	193375/221766	4
Pr1002 (optimum =259045)	OABX(15)	30000	913648.712	876632/945239	1
	EER	30000	924192.813	890737/963171	2
	OX	30000	936139.313	911860/967911	3
	UX2	30000	990777.438	955897/1019295	4
Pr2392 (optimum =378032)	OABX(15)	30000	3381711.450	3287341/3488939	1
	EER	30000	3543006.500	3382992/3640556	3
	OX	30000	3449067.000	3372940/3548791	2
	UX2	30000	3690463.000	3602666/3786763	4

Table 5. The average computation time of various crossovers

Crossover	OX	UX2	EER	OABX(3)	OABX(7)	OABX(15)
Computation time (sec)	0.0101	0.0125	0.0507	0.0110	0.0125	0.0159

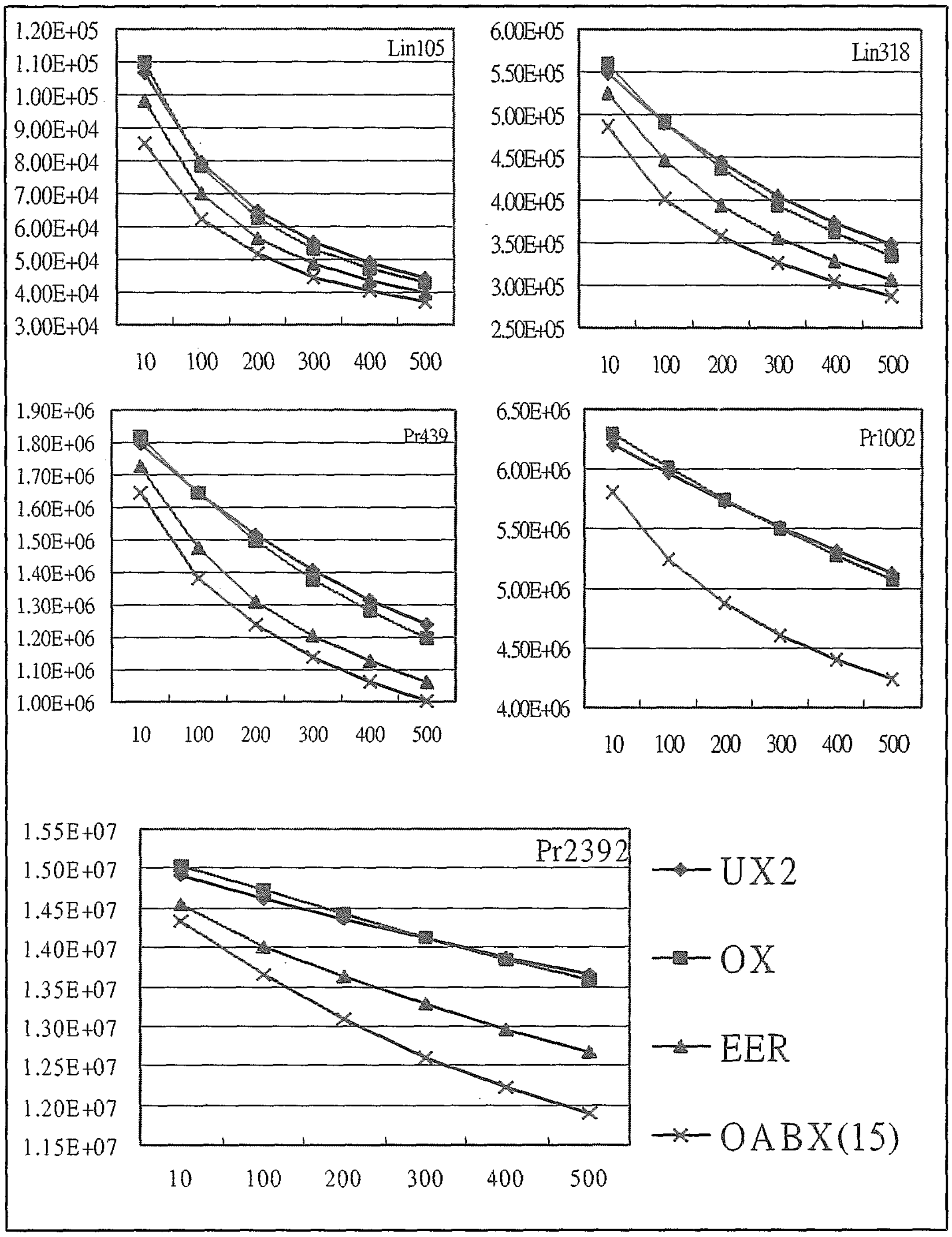

Fig. 1. The comparisons of convergence speed and accuracy for various crossovers in different instances. X axis is the GA generations and Y axis is the distance C(π).