# New EAX Crossover for Large TSP Instances

Yuichi Nagata

Graduate School of Information Sciences,
Japan Advanced Institute of Science and Technology
nagatay@jaist.ac.jp

**Abstract.** We propose an evolutionary algorithm (EA) that applies to the traveling salesman problem (TSP). The EA uses edge assembly crossover (EAX), which is known to be efficient and effective for solving TSPs. Recently, a fast implementation of EAX and an effective technique for preserving population diversity were proposed. This makes it possible to compare the EA with EAX comparable to state-of-the-art TSP heuristics based on Lin-Karnighan heuristics. We further improved the performance of EAs with EAX, especially for large instances of more than 10,000 cities. Our method can find optimal solutions for instances of up to 24978 cities within a day using a single Itanium 2 1.3-GHz processor. Moreover, our EA found three new best tours for unsolved national TSP instances in a reasonable computation time.

## 1 Introduction

The traveling salesman problems (TSPs) are widely cited NP-hard combinatorial optimization problems because they are so intuitive and easy to state. In Johnson and McGeoch's surveys [1][2], the most efficient approximation methods for TSPs were based on Lin-Kernighan local searches (LKLS) [3]. The chained Lin-Kernighan algorithm (CLK) [4] is a more sophisticated LKLS. Helsgaun [5] proposed another type of efficient LKLS (LKH). The tour-merging method [12] has been thought be a very powerful approximation method; the best tour is searched for on a restricted graph constructed of the union of dozens of high-quality solutions obtained using CKL or LKH.

Many evolutionary algorithms (EAs) have been applied to TSPs. Much effort has been devoted to designing effective crossovers suitable for TSPs because the performances of EAs are highly dependent on the design of crossovers. The edge assembly crossover (EAX) proposed by Nagata and Kobayashi [6] is known to be an effective crossover for TSPs.

However, EAs without LKLS have been found to be less effective than state-of-the-art TSP heuristics based on LKLS. Therefore, hybrid algorithms composed of EAX and CLK have been proposed [7]. On the other hand, Nagata [11] proposed a fast implementation of EAX and an effective method of preserving population diversity. This technique significantly improved the performance of EAs using EAX and demonstrated that EAs without LKLS can perform as well as state-of-the-art TSP heuristics based on LKLS.

T.P. Runarsson et al. (Eds.): PPSN IX, LNCS 4193, pp. 372–381, 2006.

In this paper, we further improve EAX to apply EAs to large instances of more than 10,000 cities because we found that the EAX used in [11] is not appropriate for large instances. The remainder of this paper is organized as follows. In Section 2, we look at existing work related to EAX. Our improvement of EAX for large instances is described in Section 3. In Section 4, we discuss our experiments and results. Section 5 is the conclusion.

## 2 Previous Work

In this section, we will introduce work related to this paper. First, we will briefly describe the algorithm of EAX [6] (See Ref. [6] or [11] for details). Then, some strategies for using EAX effectively, proposed in [10] and [11], are also described.

### 2.1 Outline of EAX

The following and Fig. 1 is an outline of EAX.

**Step 1:** Denote a pair of parents as tour-A and tour-B, and define $G_{AB}$ as a graph constructed by merging tour-A and tour-B.
**Step 2:** Divide the edges on $G_{AB}$ into *AB-cycle*s, where an *AB-cycle* is defined as a closed loop on $G_{AB}$ that can be generated by alternately tracing the edges of tour-$A$ and tour-$B$.
**Step 3:** Construct an *E-set* by selecting *AB-cycle*s according to a given rule.
**Step 4:** Generate an intermediate solution by applying the *E-set* to tour-$A$, *i.e.,* by removing tour-$A$'s edges in the *E-set* from tour-$A$ and adding tour-B's edges in the *E-set* to it.
**Step 5:** Modify the intermediate solution to generate a valid tour by connecting its sub-tours. Two sub-tours are connected by deleting one edge from each sub-tour and adding two edges to connect them. Which sub-tours are connected and which edges are deleted are determined heuristically.

The following are comments that are helpful in Section 3.

- The union of all *AB-cycle*s generated in step (2) is equal to $G_{AB}$.
- In practice, *AB-cycle*s constructed of duplicated edges are neglected in step (3) because they have no effect on step (4). These *AB-cycle*s are called *ineffective AB-cycle*s. The other are called *effective AB-cycle*s.

In step (3), the *E-set* can be constructed from any combination of *AB-cycle*s. The following two methods were proposed in previous reports [6][10].

**EAX-Rand:** The *E-set* is constructed by randomly selecting *AB-cycle*s with provability 0.5. The intermediate solution tends to equally include edges of tour-A and tour-B.
**EAX-1AB:** The *E-set* is constructed from a single *AB-cycle*. The intermediate solution tends to be similar to tour-A; i.e., children are generated by removing a small number of edges from tour-A and adding the same number of edges to it.

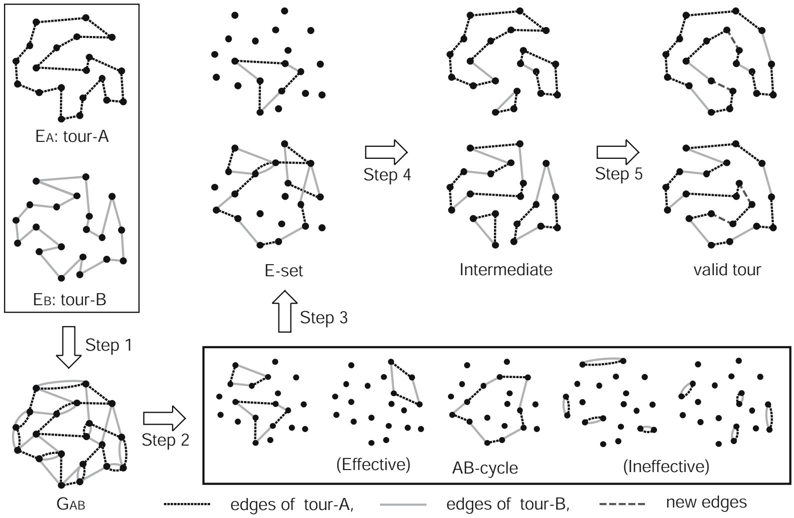


**Fig. 1.** Outline of EAX

### 2.2 Some Strategies for EAX

In previous work [10,11], these two methods were bifurcated according to the quality of the solutions in the population.

**Stage I:** EAX-1AB is used until no improvements in the shortest tour length in the population are observed over a period of time.
**Stage II:** EAX-Rand is used after stage I is finished, i.e., when EAX-1AB can no longer improve individuals in the population.

The reasons for using stage I are (i) the efficiency of the computational cost of EAX-1AB and (ii) the capability of preserving the population diversity. When EAX-1AB is used, changes of edges in the EAX algorithm are localized, and calculation is sped up. An especially efficient implementation of EAX-1AB was proposed by Nagata [11]. Moreover, EAX-1AB can prevent the population from converging wastefully by eliminating changes of edges that do not shorten the tour length [10].

Stage II is useful because EAX-Rand can produce wider varieties of children than can EAX-1AB. Stage II can actually improve individuals in the population even when EAX-1AB can no longer improve them [10,11].

## 3 Proposed Method

In this section, we propose a new EAX used in stage II instead of EAX-Rand.

First, we define some notation. The *size of an E-set* and the *size of an AB-cycle* are defined as the number of tour-A edges included in the *E-set* and the *AB-cycle*, respectively. $Gain_{Modi}$ is an improvements of tour length from an intermediate solution to a valid tour which is defined by $Gain_{Modi} = \sum_{e \in E_{remove}} w(e) - \sum_{e \in E_{add}} w(e)$, where $E_{add}$ and $E_{remove}$ are sets of edges that are added and removed, respectively, in step (5) of the EAX algorithm. $w(e)$ is a weight of an edge $e$.

### 3.1 Limitations of EAX-1AB and EAX-Rand

Let *POP* be a population that EAX-1AB can no longer improve. Such a population is usually highly refined. Therefore, each individual in *POP* is trapped in a deep local optima. To further improve individuals in *POP*, intermediate solutions should be formed so as to satisfy the following two conditions.

- (C-I) Intermediate solutions should be formed by changing tour-A extensively. In other words, the size of the *E-set* should be large to overcome deep local optima.
- (C-II) The number of sub-tours in an intermediate solution should be as small as possible.

The reason for C-II is a limitation of step (5) of the EAX algorithm. If an intermediate solution consists of $k$ sub-tours, they must be connected into a valid tour by $k-1$ operations like 2-opt moves. 2-opt move is a transition from one tour to another by exchanging two edges. Indeed, step (5) of the EAX algorithm usually increases the tour length of a resulting valid tour ($Gain_{Modi} < 0$) when tour-A is highly refined because 2-opt moves are the most restricted method of connecting two sub-tours. Thus, the number of sub-tours in an intermediate solution should be restricted to increase $Gain_{Modi}$.

However, C-I and C-II usually conflict because the number of sub-tours in an intermediate solution tends to increase as the size of *E-set* increases. Considering C-I and C-II, the drawbacks of EAX-1AB and EAX-Rand can be summarized as the following three hypotheses. Typical examples of intermediate solutions for each case are illustrated in Fig. 2. We will verify these hypotheses in Section 3.3.

- (i) If an *E-set* is constructed from a single small-sized *AB-cycle*, C-I is not satisfied.
- (ii) If an *E-set* is constructed from a single large-sized *AB-cycle*, C-II is not satisfied.
- (iii) If an *E-set* is constructed by randomly selecting multiple *AB-cycle*s (EAX-Rand), C-II is not satisfied. However, this method can produce improved tours from *POP* at least in principle because a wide variety of *E-set*s can be constructed. However, the likelihood is very low.

### 3.2 EAX-Block

In this subsection, we propose a method of selecting *AB-cycle*s for constructing an *E-set* that can produce an intermediate solution satisfying C-I and C-II. We call EAX using this method EAX-Block.

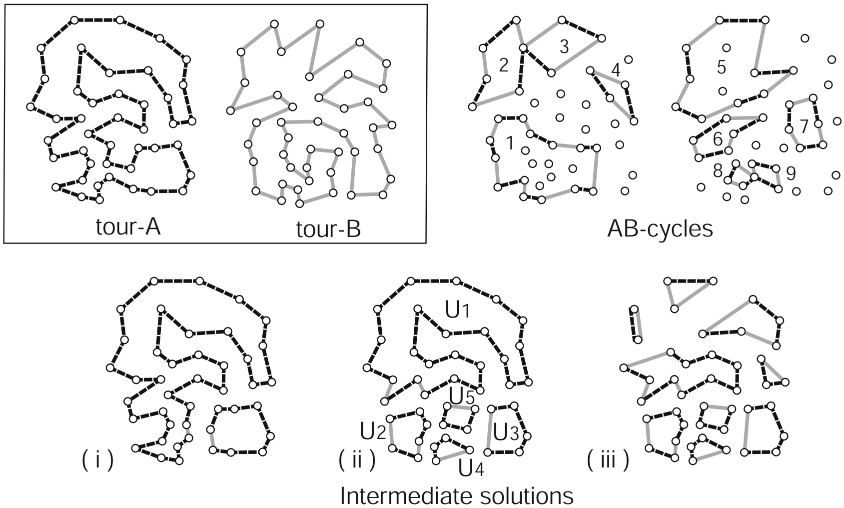


**Fig. 2.** Typical examples of intermediate solutions generated by *E-set*s constructed of (i) a single small-sized *AB-cycle* (*AB-cycle* 9), (ii) a single large-sized *AB-cycle* (*AB-cycle* 1), and (iii) randomly selected multiple *AB-cycle*s (*AB-cycle* 1, 2, 3, 4). Tour-A, tour-B and *AB-cycle*s are also illustrated ( Ineffective *AB-cycle*s are omitted).

**EAX-Block:**

**1.** Select a large-sized *AB-cycle*. Let it be a center *AB-cycle*. Note that the top $N_{ch}$ largest-sized *AB-cycle*s are selected as center ones when $N_{ch}$ children are generated from a pair of parents.

**2.** Apply the center *AB-cycle* to tour-A and form an intermediate solution. Let $U_i$ $(i = 1, \ldots, k)$ be the i-th sub-tour, where $k$ is the number of sub-tours. Let $U_1$ be the largest sub-tour, i.e., including the largest number of edges.

**3.** Select *AB-cycle*s that satisfy the following conditions.
-(c1): They have connections to vertices in $U_i$ $(i = 2, \ldots, k)$.
-(c2): Their sizes are smaller than that of the center *AB-cycle*.

**4.** Construct an *E-set* from the center *AB-cycle* and the *AB-cycle*s selected in step 3.

Fig. 2 and Fig. 3 illustrate an example of EAX-Block. In step (1), *AB-cycle* 1 illustrated in Fig. 2 is selected as a center *AB-cycle*. Therefore, an intermediate solution (ii) in Fig. 2 is produced in step (2). In step (3) and (4), an *E-set* is constructed of *AB-cycle*s 1, 6, 7, 8 and 9 as shown in Fig. 3, where ineffective *AB-cycle*s satisfying (c1) are also included in the *E-set* for the sake of simplicity of an explanation described below. A resulting intermediate solution produced by the *E-set* is illustrated in Fig. 3.

Now, properties of EAX-Block are described. For the sake of simplicity, ineffective *AB-cycle*s can be selected in step (3), and condition (c2) is not considered here. First, we define the following terms.

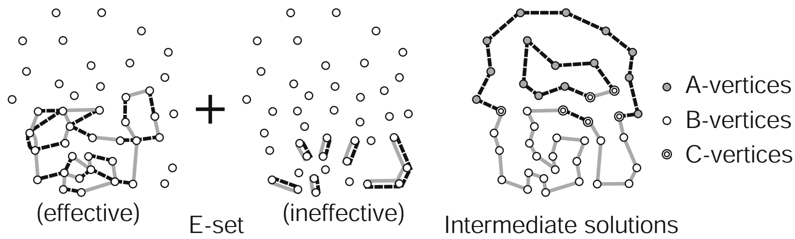


**Fig. 3.** Typical example of an *E-set* and an intermediate solution generated by EAX-Block. The *E-set* is constructed of *AB-cycle*s 1, 6, 7, 8 and 9 illustrated in Fig. 2 and ineffective *AB-cycle*s adjacent to $U_2, \ldots, U_5$.

**A-vertex:** A vertex that is connected to no tour-A (tour-B) edge in the *E-set*. It is connected to two tour-A edges in intermediate solutions.

**B-vertex:** A vertex that is connected to two tour-A (tour-B) edges in the *E-set*. It is connected to two tour-B edges in intermediate solutions.

**C-vertex:** A vertex that is connected to one tour-A (tour-B) edge in the *E-set*. It is connected to one tour-A edge and one tour-B edge in intermediate solutions.

All vertices in $U_2, \ldots, U_k$ are B-vertices because of condition (c1) in step (3) (Remember the comments mentioned in Section 2.1.). Vertices in $U_1$ that are geographically far from the other sub-tours tend to be A-vertices if the sizes of *AB-cycle*s selected in step (3) are small. Other vertices are C-vertices, which are located between A-vertices and B-vertices. In Fig. 3, vertices in the intermediate solution are divided into A-, B- and C-vertices. When the number of C-vertices increase, the number of sub-tours tend to increase as shown in Fig. 2.

The advantage of EAX-Block over EAX-1AB and EAX-Rand is that intermediate solutions can be generated by assembling a block of tour-A edges and a block of tour-B edges. If the sizes of all *AB-cycle*s selected in step (3) are small, the number of C-vertices tends to be small. In this case, EAX-Block has an ideal property and can satisfy the conditions (C-I) and (C-II).

### 3.3 Behaviors

Now, we demonstrate the behaviors of EAX-1AB, EAX-Rand, and EAX-Block.

The distribution frequency of sizes of *AB-cycle*s that are obtained by applying EAX to a pair of parents is shown in Fig. 4 (a). These data are averaged over 150 pairs of parents selected from *POP* by random sampling without replacement, where *POP* was generated by stage I with EAX-1AB as described in Section 4.1. The instance usa13509 [9] was used for these experiments.

On average, 124.6 *AB-cycle*s are generated from a pair of parents[1]. As shown in Fig. 4 (a), while most *AB-cycle*s have sizes smaller than 10, relatively large

[1] 95% edges are removed as ineffective*AB-cycle*s because individuals in *POP* are similar to each other.

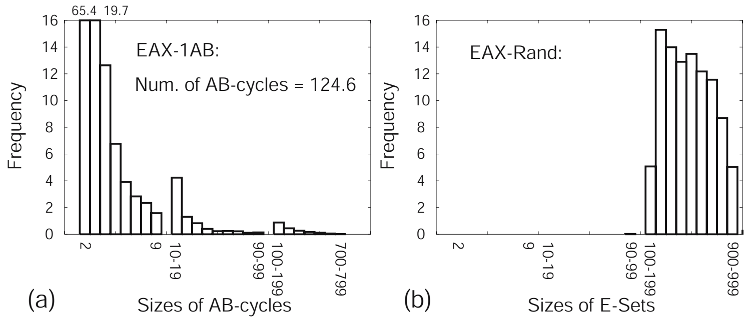


**Fig. 4.** (a) Distribution frequency of size of *AB-cycle*s. (b) Distribution frequency of size of *E-set*s. Note that scales of x-axes are different.

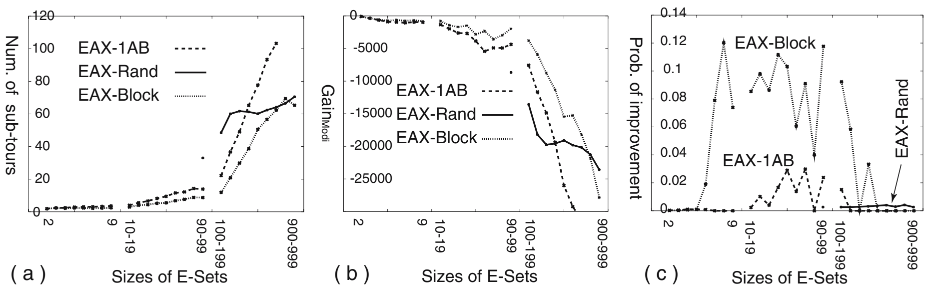


**Fig. 5.** Behaviors of EAX-1AB, EAX-Rand, and EAX-Block. Note that scales of x-axes are different.

*AB-cycle*s having sizes of larger than 100 are usually contained in this example. Figure 4 (b) shows the distribution frequency of the size of *E-set*s that were obtained by applying EAX-Rand to the same population, where 100 *E-set*s are generated form a pair of parents. Obviously, the sizes of the *E-set*s generated by EAX-Rand are larger than those generated by EAX-1AB.

The behaviors of the three EAXs are shown in Fig. 5. The data are averaged over each size of an *E-set*. Figure (a) shows the number of sub-tours in intermediate solutions generated by the three types of EAXs. As shown, the number of sub-tours tends to increase as the size of the *E-set* increases. On the other hand, Fig. (b) shows that $Gain_{Modi}$ tend to decrease as the size of the *E-set* increases. We can see that the graphs in Figs. (a) and (b) are symmetric with respect to the x-axis. Thus, the number of sub-tours and $Gain_{Modi}$ have a negative correlation. Based on the condition (C-II), EAX-Block is the best method among the three types of EAXs because the number of sub-tours ($Gain_{Modi}$) of EAX-Block is the smallest (largest) among them. Consequently, EAX-Block can improve tour-A more frequently than can EAX-1AB and EAX-Rand, as shown in Fig. (c). Fig. (c) shows the probabilities of obtaining improved individuals from tour-A using *E-set*s constructed by the three types of EAXs.

# 4 Experiments

## 4.1 Experimental Setting

We compared EAX-1AB, EAX-Rand, and EAX-Block on several TSP benchmarks. Experiment setting is the same as the Nagata's works [11] where the edge entropy measure was used to maintain population diversity, and the fast implementation of EAX was used. This experiments are implemented in C++ and executed using Itanium 2 1.3-GHz single processor with 126 GB of RAM.

**Stage I:** EAX-1AB was applied to TSP benchmarks using selection model I [11], where the population size ($N_p$) was set to 300, and an initial population was generated by the 2-opt local search. The number of children generated from a pair of parents ($N_{ch}$) was set to 30. If the shortest tour length in the population stagnated over 150 generations, then the run was terminated. Ten trials were executed for each instance. The resulting population is denoted as $POP_i$ ($i = 1, \ldots, 10$) for each run.

**Stage II:** For ($i = 1, \ldots, 10$), EAX-Rand or EAX-Block was applied to TSP benchmarks using $POP_i$ as the initial population. Selection model I was used. Although $N_p$ was necessarily 300, $N_{ch}$ was set to 100 in this case to enhance the searches. The termination conditions were the stagnation of 100 generations for EAX-Rand or 50 for EAX-Block.

## 4.2 Results

In these experiments, eight large instances were chosen from TSPLIB [9] and twelve large instances from the national TSPs [13]. The results of EAX-1AB, EAX-Rand, and EAX-Block are listed in Table 1. As shown, EAX-Rand improved the qualities of the solutions obtained by EAX-1AB with a few exceptions. Although EAX-Rand can usually find optimal (best known) solutions for the instances with up to 10,000 cities, it fails larger instances. In contrast, EAX-Block can find optimal (best known) solutions for instances of up to 24978 cities. Moreover, the CPU times needed to terminate runs of EAX-Block were about 10 times faster than those of EAX-Rand.The reasons are that (i) EAX-Block can improve populations more rapidly than EAX-Rand and that (ii) EAX-Block can generate individuals faster than EAX-Rand.

In this experiment, EAX-Block found three new best solutions to the national TSPs benchmarks. This is the first improvement in several years. The new best tours (tour lengths) are pa8079 (114855), ho14473 (177092), and bm33708 (959291).

We compare EAX-Block with other state-of-the-art TSP heuristic algorithms. Our proposed approach is categorized as approximation methods for TSPs that consume relatively large time but aim at finding very near-optimal solution. So, we chose HeSEA [7] and tour-merging technique [12] that are categorized as the same class. HeSEA is a hybrid algorithm composed of EAX and CLK [4]. Tour-merging method look for a best tour on a restricted graph consisting of the union of set of tours obtained by LKH[5].

**Table 1.** Comparisons of performances of EAs using three types of EAX. "Opt." column indicates number of trials that reached optimal solutions in ten trials. "Err." indicates average length by which best tour exceeded optimal tour in each trial. "Gen." indicates average generation required to reach best individual in each trial. "Time" means average CPU time in seconds required for one trial. For unsolved instances, a number of trials that reach best tour known today is listed in "Opt.", and "–" is filled in "Err.". If new best tour is found, "Improve (number of these trials)" is filled in "Opt.".

| | EAX-1AB (Stage I) | | | | EAX-Block (Stage II) | | | | EAX-Rand (Stage II) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instances | Opt. | Err. (%) | Gen. | Time (sec) | Opt. | Err. (%) | Gen. | Time (sec) | Opt. | Err. (%) | Gen. | Time (sec) |
| fnl4461 | 0 | 0.0014 | 848 | 1512 | 10 | 0.0000 | 3 | 42 | 10 | 0.0000 | 30 | 185 |
| rl5915 | 10 | 0.0000 | 319 | 992 | 10 | 0.0000 | 0 | 36 | 10 | 0.0000 | 0 | 125 |
| r11849 | 0 | 0.0041 | 1252 | 7646 | 10 | 0.0000 | 15 | 298 | 9 | 0.0000 | 61 | 2533 |
| usa13509 | 0 | 0.0126 | 2486 | 13249 | 6 | 0.0001 | 43 | 496 | 0 | 0.0019 | 173 | 7164 |
| brd14051 | 0 | 0.0129 | 2729 | 15550 | 10 | 0.0000 | 31 | 712 | 0 | 0.0033 | 216 | 10193 |
| d15112 | 0 | 0.0181 | 3076 | 21244 | 4 | 0.0001 | 107 | 1662 | 0 | 0.0081 | 165 | 40060 |
| d18512 | 0 | 0.0186 | 3496 | 25392 | 8 | 0.0000 | 77 | 1526 | 0 | 0.0047 | 253 | 14037 |
| pla33810 | 0 | 0.0182 | 5014 | 38424 | 0 | 0.0076 | 79 | 1892 | 0 | 0.0124 | 456 | 18930 |
| pm8079 | Improve (5) | | 800 | 1596 | Improve (9) | | 1 | 128 | Improve (9) | | 6 | 830 |
| ei8246 | 0 | -- | 1476 | 4556 | 9 | -- | 1 | 439 | 5 | -- | 53 | 4511 |
| ar9152 | 0 | -- | 1226 | 7038 | 9 | -- | 2 | 135 | 9 | -- | 10 | 217 |
| ja9847 | 0 | 0.0057 | 1664 | 4705 | 3 | 0.0017 | 6 | 275 | 3 | 0.0028 | 31 | 1574 |
| kz9976 | 0 | -- | 1700 | 5967 | 9 | -- | 21 | 532 | 1 | -- | 119 | 5685 |
| fi10639 | 0 | -- | 1891 | 6955 | 5 | -- | 34 | 808 | 0 | -- | 169 | 9276 |
| mo14185 | 0 | -- | 2379 | 10481 | 9 | -- | 44 | 1089 | 0 | -- | 0 | 10756 |
| ho14473 | 0 | -- | 1218 | 3365 | Improve (10) | | 57 | 359 | Improve (7) | | 67 | 5445 |
| it16862 | 0 | 0.0173 | 3094 | 14778 | 2 | 0.0007 | 109 | 892 | 0 | 0.0173 | 0 | 19778 |
| vm22775 | 0 | 0.0089 | 3814 | 21346 | 1 | 0.0007 | 45 | 3457 | 0 | 0.0089 | 0 | 20686 |
| sw24978 | 0 | 0.0209 | 4522 | 36946 | 3 | 0.0010 | 129 | 2500 | 0 | 0.0209 | 0 | 26542 |
| bm33708 | 0 | -- | 6339 | 56305 | Improve (1) | | 168 | 5094 | 0 | -- | 0 | 36087 |

Table 2 show the results. As compared with the results of HeSEA and Tour-merging, EAX-Block could find optimal solutions in some large instances with smaller CPU times even where other method could not find them.

## 5 Conclusion

We improved the edge assembly crossover (EAX) to apply EAs using EAX to large TSP instances having more than 10,000 cities. Our results demonstrated that EAX-Block is suitable for large TSP instances.

We observed that the following two conditions are needed to improve highly refined near-optimal solutions using EAX for large instances. (C-I) The *E-set* should be large enough to overcome deep local optima. (C-II) The number of sub-tours in an intermediate solution should be as small as possible. We proposed EAX-Block to satisfy these conditions. The key idea of EAX-Block is assembling blocks of tour-A edges and blocks of tour-B edges to generate intermediate solutions. We demonstrated that EAX-Block is better than EAX-1AB and EAX-Rand in terms of the above conditions.

The experimental results show that the EA with EAX-Block can find optimal solutions for large instances of up to 24978 cities in a reasonable CPU time. Moreover, three new best tours were found for unsolved national TSP instances.

**Table 2.** Performances of other state-of-the-art TSP heuristics. These Data are copied from the original papers ( Data are not available in Blank cells). HeSEA and Tour-merging were executed twenty and one trials for each instance, respectively. CPU time of HeSEA and Tour-merging are based on Pentium IV 1.2-GHz and EV6 Compaq Alpha 500-MHz processors, respectively.

| | HeSEA | | | Tour-merging | | |
|---|---|---|---|---|---|---|
| Instances | Opt. | Err. (%) | Time (sec) | Opt. | Err. (%) | Time (sec) |
| fnl4461 | 16/20 | 0.0005 | 2,349 | | | |
| rl5915 | 19/20 | 0.0001 | 2,773 | 1/1 | 0.0000 | 63,954 |
| r11849 | | | | 1/1 | 0.0000 | 646,483 |
| usa13509 | 0/20 | 0.0074 | 34,948 | 0/1 | 0.0001 | 968,473 |
| brd14051 | | | | 0/1 | 0.0030 | 1,676,314 |
| d15112 | | | | 1/1 | 0.0000 | 1976,174 |
| d18512 | | | | 0/1 | 0.0071 | 3,704,852 |
| pla33810 | | | | 0/0 | 0.0998 | 43,632,379 |

## References


1. D. S. Johnson: Local Optimization and the Traveling Salesman Problem, Automata, Languages and Programming, Lecture note in Computer Science 442, pringer, Heidelberg, pp. 446–461.
2. 8-th DIMACS Implementation Challenge: The Traveling Salesman Problem, http://www.research.att.com/ dsj/chtsp.
3. S. Lin and B. Kernighan, Effective heuristic algorithms for the traveling salesman problem, Oper. Res., vol. 21, pp. 498–516, 1973.
4. D. Applegate, R. Bixby, V. Chvatal, and W. Cook, "Finding tours in the TSP. Technical Report 99885, Forschungsinstitut fur Diskrete Mathematik, Universitat Bonn, 1999.
5. K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic, "Eur. J. Oper. Res., vol. 126, no.1, pp. 106–130, 2000.
6. Y. Nagata and S. Kobayashi, Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem, Proc. of the 7th Int. Conference on Genetic Algorithms, pp. 450-457, 1997.
7. H. K. Tsai, J. M. Yang, Y. F. Tsai, and C. Y. Kao, An Evolutionary Algorithm for Large Traveling Salesman Problem, IEEE Transaction on SMC-part B, vol. 34, no. 4, pp. 1718- 1729, 2004.
8. K. Maekawa, N. Mori, H. Kita, and H.Nishikawa, A Genetic Solution for the Traveling Salesman Problem by Means of a Thermodynamical Selection Rule, Proc. 1996 IEEE Int. Conference on Evolutionary Computation, pp. 529-534, 1996.
9. TSPLIB95, http://www.iwr.uni-heidelberg.de/iwr/compt/soft/TSPLIB95
10. Y. Nagata, The EAX algorithm considering diversity loss, Proc. of the 8th Int. Conference on Parallel Problem Solving from Nature, pp. 332–341, 2004.
11. Y. Nagata, Fast EAX algorithm Considering Population Diversity for Traveling Salesman Problems, Proc. of the 6th Int. Conference on EvoCOP2006, pp. 171–182, 2006.
12. W. Cook and P. Seymour, Tour Merging via Branch-Decomposition, INFORMS Journal on Computing, vol. 15, no. 3, pp. 233–248, 2003.
13. National Traveling Salesman Problems.http://www.tsp.gatech.edu/world/ countries.html.