Edge Assembly Crossover with Tabu for Traveling Salesman Problem

1st Maaki Sakai
Graduate School of Science and Engineering, Kansai University
Osaka, Japan
k697938@kansai-u.ac.jp

2nd Yoshiko Hanada
Faculty of Engineering Science
Kansai University
Osaka, Japan
hanada@kansai-u.ac.jp

3rd Yukiko Orito
Department of Economics
Hiroshima University
Hiroshima, Japan
orito@hiroshima-u.ac.jp

***Abstract*—This paper proposes a new tabu technique to enhance the search performance of edge assembly crossover (EAX). EAX is known as the most promising recombination operator for the traveling salesman problem (TSP) among population-based evolutional approaches such as genetic algorithm (GA). It succeeds in generating a sophisticated combination of parents' traits and makes GA comparable to state-of-the-art heuristics for the TSP. EAX can find the optimal in very large instances, however, it sometimes does not work well on instances involving a lattice pattern. This is because, in the instance including cities that are arranged in a lattice pattern, it is difficult to find the best tour due to the existence of many alternative candidate tours. In instances that include such patterns, the diversity of edges in the population is decreased because the EAX concentrates and repeats exchanging edges in this local pattern, and pays less attention to combinations of edge globally. In this paper, we improve EAX itself by introducing a tabu scheme to inhibit repeating the exchange of the same edges in order to reduce the bias of edge to be exchanged between parents and enhance the diversity in offspring. Numerical experiments show the search performance of our new method in the instances selected from TSPLIB and VLSI TSP.**

***Index Terms*—traveling salesman problem, genetic algorithm, crossover**

I. Introduction

Traveling salesman problem (TSP) is the problem to find the shortest Hamiltonian cycle, under given definitions of a list of cities and the distances between each pair of cities. TSP is one of the famous NP-hard combinatorial optimization problems, and various heuristics or meta-heuristics such as evolutionary computations have been developed to find or approximate the optimum in large instances of TSP.

In optimizing TSP by genetic algorithm (GA), crossover operators have been considered to play a central role since its design strongly impacts on the search performance of GA. To solve TSP effectively, it is important to design the crossover operator that can handle problem-specific structures and traits (building blocks), and various crossover operators focusing on the inheritance of parents' preferable traits have been developed [1]–[8]. Among crossovers, edge assembly crossover (EAX) [5] is the most effective crossover that has succeeded in generating offspring which inherit edges from two parents with keeping parents' traits. In EAX, to combine edges of two parents' tours, first, EAX extracts some closed cycles referred to *AB-cycle* that is constructed by tracing edges of the two tours alternately. By applying *Eset* which consists of a number of *AB-cycles* to each tour respectively, the tour is then decomposed to a set of closed cycles (sub-tours). Finally, all sub-tours are connected repeatedly with short edges one by one until become one complete closed cycle. The closed cycle constructed in this manner above is the offspring of EAX. By using a small *Eset* to limit the number of edges exchanged between parents in EAX, it has been shown the local search performance improves [11]. In the optimization of TSP using genetic algorithms, it is particularly important to consider the exchange of various edges and to enhance the diversity of edges in the population in order to improve the search performance [9], [10]. Therefore, survival selection of offspring for the next generation has been discussed to improve the search performance of EAX [12], [13]. The latest EAX [13] that introduces an entropy related to edge rarity in the survival selection is one of the most successful EAXs. There are edges which are difficult to be generated again, once they lost from the population. In the latest EAX, it is possible to select and to make such edges survive in the next population. By this extensions, EAX has made GA comparable to or outperform state-of-the-art heuristics for the TSP such as Lin-Kernighan (LK) based algorithms [14]–[16].

EAXs have succeeded in finding the optimum in very large TSP instances, however, a drawback exists in EAX that does not work very well on instances involving a lattice pattern of cities. In instance where cities are even partially arranged in a lattice pattern, it is difficult to find the best tour due to the existence of many alternative optimal or near-optimal candidates. In such a pattern, the EAX may decrease the search performance because it concentrates and repeats the edge exchange in the lattice part and does not pay less attention to exchange edges in other parts of the lattice pattern. This issue is expected to be eased if we can avoid the edge exchange that is concentrated in specific points in EAX. In this paper, we improve the search performance of EAX in the lattice pattern by introducing a tabu scheme into EAX to inhibit repeating the exchange of the same edges based on an archive of *AB-cycles* generated in the past generation. Here, as a preliminary study, we implement our new method into the original EAX to evaluate the effectiveness of our scheme.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

Through the numerical experiments, we show the improvement of the search performance in relatively small instances from TSPLIB [17] and VLSI TSP [18].

II. Edge Assembly Crossover

EAX is a powerful crossover proposed by Nagata, for TSP. It has been designed to generate offspring by combining edges from two parents and adding relatively few new short edges. The procedure of EAX is briefly described as following.

Step 1 Let p_A and p_B be an individual respectively selected from the population. EAX first creates a multigraph denoted by G_{AB} consisting of all edges of p_A and p_B. In G_{AB}, the two paths are considered to be different paths even if they have the same source and the same destination.

Step 2 Find a closed cycle by tracing the edges of p_A and p_B alternately from G_{AB}. This closed cycle is called *AB-cycle*, and the edges of generated *AB-cycle* are removed from G_{AB}. By repeating this operation, all the edges of G_{AB} can be divided into several *AB-cycles*.

Step 3 Construct *Eset* which consist of several *AB-cycles* selected randomly from all *AB-cycles* extracted in the step 2. In this selection, however, *AB-cycles* that have only two edges are excluded.

Step 4 Edges of p_A included in the *Eset* are deleted from p_A. Instead of deleting p_A's edges, edges of p_B included in *Eset* are added to p_A. By these operations, p_A becomes a set of closed cycles (sub-tours) which partially includes edges from p_B.

Step 5 All sub-tours that constitute p_A are connected repeatedly with short edges one by one as shown Fig. 1, until one complete closed cycle is reassembled as an offspring.

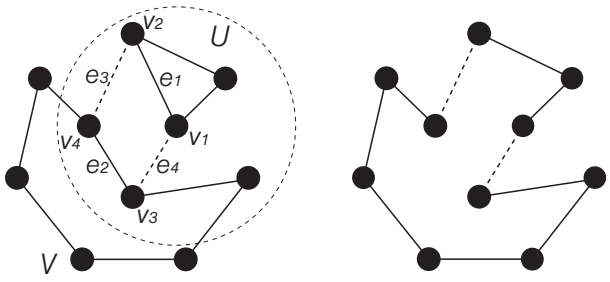

Fig. 1. Connecting two closed cycles: Find the edge combination $\{e_1, e_2, e_3, e_4\}$ where the value of $\{-w(e_1)-w(e_2)+w(e_3)+w(e_4)\}$ is the smallest, where $w(\cdot)$ means the path length. Connect the cycle U and the cycle V by deleting e_1, e_2 and adding $e3, e4$. Here U and v_1 are the smallest closed cycle and one of its cities, respectively. The city v_3 is v_1's neighborhood city that not included in the cycle U, and V is the closed cycle includes v_3. The cities v_2 and v_4 are adjacent cities of v_1 and v_3, respectively.

The complete closed cycle obtained in the manner above is an offspring based on p_A. By applying these operations to p_B, in a similar way to p_A, an offspring is generated based on p_B. EAX can generate a wide variety of offspring that keep both parents' traits, by constructing various *Esets* from two parents. Offspring which have shortest tour length are selected as the next populations, so that preferable traits are inherited to the next generation.

At step 3 of EAX, there are several manner to construct *Eset*. In this paper, as a preliminary study, we adopt the original EAX (EAX-Rand) [11] that all generated *AB-cycles* are selected with the probability of 1/2 to constitude the *Eset*.

III. Proposal of EAX-tabu

A. Edge Assembly Crossover using tabu scheme

One of structures which make difficult for GA to find the optimal solution is a lattice pattern. In cities arrangement of this pattern, there are many alternative tours that have the same or almost equivalent path lengths, so that GA cannot identify the best easily even if the lattice pattern partially exists. The same applies to GA with EAX. When an instance partially includes the lattice pattern, there are many paths, i.e., many variations of edge pattern, in the pattern compared to other parts where cities do not arrange in the lattice pattern. Thus the diversity of edges is an imbalance in whole cities. This causes that many edge candidates inside lattice pattern have possible to occupy *AB-cycles*, which EAX concentrates on and repeats exchanging edges in this lattice pattern while pays less attention to combinations of edges in other parts.

In this study, to ease the imbalance in exchanging edges in EAX, we newly introduce a tabu scheme into EAX to inhibit repeating the exchange of the same edges based on an archive of *AB-cycles*. In our tabu scheme, each individual in the population has an archive that keeps a history of exchanged edges. At t-th generation, a current individual is an offspring generated by applying an *Eset*, a set of *AB-cycles*, of two parents in (t-1)-th generation. Thus the edges newly added to the current individual have been recorded by the *Eset*. The archive of exchanged edges for each individual is constructed as a union set of *Eset* up to past *tenure* generations. Here, we denote such an archive by *pre-Eset*. The parameter *tenure* is the period for keeping the *Eset*. The procedure of EAX with a tabu scheme (EAX-tabu for short) is described as following. As shown in the procedure, here, we incorporate the tabu scheme to the original EAX of which *Eset* is constructed by *AB-cycles* selected with the probability of 1/2. Figure 2 illustrates the example of EAX-tabu.

Step 1 Let p_A and p_B be an individual respectively selected from the population. Creates a multigraph G_{AB} consisting of all edges of p_A and p_B.

Step 2 Decompose G_{AB} entirely to a set of *AB-cycles*.

Step 3 Select edges randomly from *pre-Eset* of p_A and call these edges *tabu-edges*.

Step 4 Exclude *AB-cycles* which include *tabu-edges* from the set of *AB-cycles* obtained at step 2.

Step 5 Construct *Eset* which consist of several *AB-cycles* selected randomly from available *AB-cycles* that consists of more than four edges.

Step 6 Edges of p_A included in the *Eset* are deleted from p_A. Instead of deleting p_A's edges, edges of p_B included in *Eset* are added to p_A. By these operations, p_A becomes a

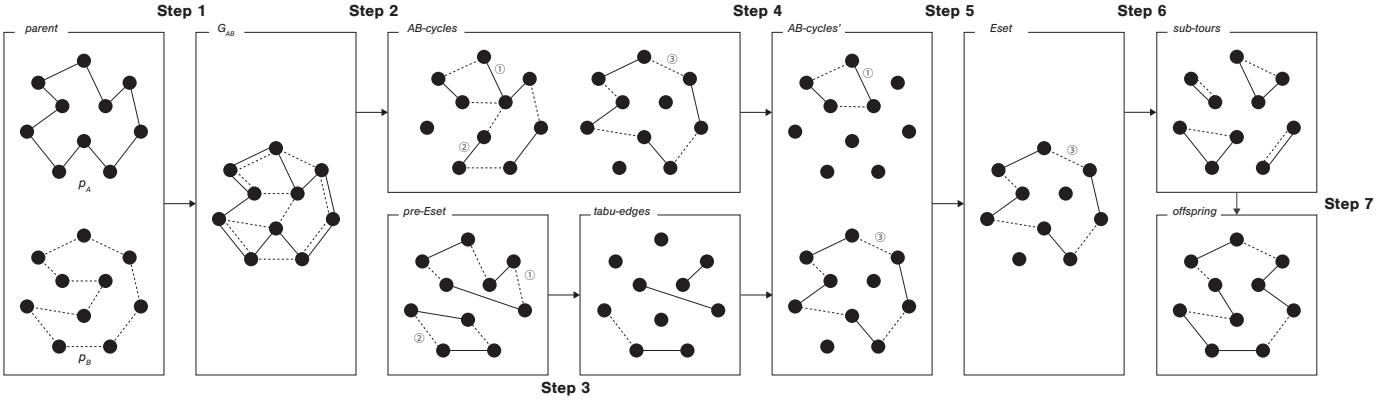

Fig. 2. Example of EAX-tabu: G_{AB} is decomposed to three *AB-cycles*. Four *tabu-edges* are selected from *pre-Eset*, and *AB-cycle* (2) involving *tabu-edges* is excluded. Construct *Eset* from two available *AB-cycles*. Then apply *Eset* to p_A to obtain a set of sub-tours. Finally connect them with short paths to generate an offspring.

set of closed cycles (sub-tours) which partially includes edges from p_B.

Step 7 All sub-tours that constitute p_A are connected repeatedly with short edges one by one until one complete closed cycle is reassembled as an offspring.

The *Eset* generated at step 5 is newly added to the *pre-Eset* of p_A, and instead the oldest *Eset* are deleted. The value of *tenure* is important. If it is set to small, that causes a circulation, while the large value of *tenure* decreases the number of available of *AB-cycles*.

B. Generation Alternation Model

In numerical experiments, we use the generation alternation model used in the conventional EAX [11] described as follows.

Step 1 /initialization/ Generate N_{pop} random tours (individuals) and apply 2-opt to all individuals to compose the initial population$\{p_1, \ldots, p_{N_{pop}}\}$.

Step 2 /selection for reproduction/ Reset indexes {1, 2, $\cdots$, N_{pop}} to each individual randomly. Then select N_{pop} pairs of parents $\{p_i,\ p_{i+1}\}$ $(1 \leq i \leq N_{pop})$ from the population, where $p_{N_{pop}+1} = p_1$.

Step 3 /crossover/ Apply EAX-tabu to each pair of parents $\{p_i,\ p_{i+1}\}$ N_{off} times and generate offspring $\{c_1, \ldots, c_{N_{off}}\}$.

Step 4 /selection for survival/ For each pair of parents, select the best offspring, c_{best}, from $\{c_1, \ldots, c_{N_{off}}, p_i\}$ and replace p_i with c_{best}. Add the *Eset* used in the crossover to the *pre-Eset* of p_i. If the size of the *pre-Eset* is larger than *tenure*, remove the oldest *E-set* from the *pre-Eset*.

Step 5 /terminal criterion/ Go to 2 until some terminal condition is satisfied, e.g., generations and/or the number of evaluations.

IV. Numerical Experiments

Here we evaluate the search performance of EAX-tabu. In experiments, we used instances 14 instances up to 5915 cities from TSPLIB [17] and 10 instances up to 4355 cities from VLSI TSP [18]. The former benchmarks have various kinds of city arrangement, while the most instances in the latter benchmarks include the lattice pattern found in the VLSI design.

A. Performance of EAX-tabu

To evaluate the effectiveness of tabu in EAX, we compare EAX-tabu with the original EAX (EAX-Rand) and verify whether more optimal solutions can be obtained by increasing the number of offspring generated in the crossover.

The population size N_{pop} was set to 300, and the number of offspring N_{off} was set to 50, 100 and 200. In the proposed EAX-tabu, the parameter *tenure*, the period for keeping *Eset*, was set to 1. The number of trials was set to 30. Each run was terminated when all individuals in the population were converged to the same solution, or no update of the best fitness in the population is found through 30 generations.

The experimental results are shown in Table I and Table II. In these results, opt. indicates the optimal value of the instance, and #opt indicates the number of runs that reached the optimum. The best performance among the settings of offspring size is highlighted by bold.

From Table I and Table II, we can see that EAX-tabu outperforms EAX-Rand and confirm that EAX-tabu improves the search performance in most instances by enlarging the number of offspring generated in the crossover. We also find that a significant improvement of EAX by using the tabu scheme on VLSI TSP instances where the most cities are arranged in the lattice pattern. In the instance fnl4461 of TSPLIB, both EAX cannot find optimal solution. In this instance, it has been shown that small *Eset* to limit the number of exchanged edges performs well compared to EAX-Rand [11].

B. Effect of tabu period

Here, we verify the effect of the setting of parameter *tenure* on the search performance of EAX-tabu. Here we use TSPLIB instances to compare with one of the latest implements of EAX. The parameter *tenure* was set to 1, 3, 5, 7 and 10. The population size N_{pop} was set to 300, and the number of

TABLE I
TSPLIB RESULT

instance (#cities)	opt.	#opt. $N_{off}=50$ EAX-Rand	$N_{off}=50$ EAX-Tabu	$N_{off}=100$ EAX-Rand	$N_{off}=100$ EAX-Tabu	$N_{off}=200$ EAX-Rand	$N_{off}=200$ EAX-Tabu
rat575 (575)	6773	15	17	17	21	**22**	**22**
u724 (724)	41910	20	20	21	23	22	**29**
vm1084 (1084)	239297	23	23	21	24	**25**	**25**
pcb1173 (1173)	56892	23	23	**27**	**27**	23	26
d1291 (1291)	50801	23	24	23	**27**	23	**27**
u1432 (1432)	152970	10	17	10	**21**	14	20
d1655 (1655)	62128	0	2	**8**	4	4	4
vm1748 (1748)	336556	4	20	7	**22**	7	17
u1817 (1817)	57201	3	5	1	5	0	10
u2152 (2152)	64253	4	10	6	**22**	7	**22**
pr2392 (2392)	378032	18	**23**	14	22	22	22
pcb3038 (3038)	137694	0	**1**	0	0	0	**1**
fnl4461 (4461)	182566	0	0	0	0	0	0
rl5915 (5915)	565530	0	4	1	3	2	**5**

TABLE II
VLSI RESULT

instance (#cities)	opt.	#opt. $N_{off}=50$ EAX-Rand	$N_{off}=50$ EAX-Tabu	$N_{off}=100$ EAX-Rand	$N_{off}=100$ EAX-Tabu	$N_{off}=200$ EAX-Rand	$N_{off}=200$ EAX-Tabu
xit1083 (1083)	3558	**30**	**30**	**30**	**30**	**30**	**30**
icw1483 (1483)	4416	29	**30**	**30**	**30**	29	**30**
djc1785 (1785)	6115	13	16	20	21	20	**25**
dcb2086 (2086)	6600	15	23	24	25	24	**27**
xpr2308 (2308)	7219	0	11	3	10	1	**14**
mlt2597 (2597)	8071	14	21	14	20	15	**23**
pia3056 (3056)	8258	0	**1**	0	0	0	**1**
fdp3256 (3256)	10008	0	0	0	2	0	**3**
ltb3729 (3729)	11821	1	1	2	4	2	**5**
bgb4355 (4355)	12723	0	1	0	7	**8**	**8**

offspring N_{off} was set to 200. The number of trials was set to 30, and terminal criteria were the same as the previous section.

The experimental results are shown in Table III. The results of the latest version of EAX (EAX-Ent for short, in the result) are also listed for reference [12]. EAX-Ent has introduced an entropy relevant to edge occurrence into the selection for survival for the next generation. Note that the condition of the experiment is different in the results. The number of trials was 20 in EAX-Ent and different from that of EAX-tabu, so that we compare the ratio of runs that reach the optimal value. In the results, %opt. indicates the ratio of success runs. Ave. and Gene. are the averaged value of the best solution and the averaged converegd generation out from 30 trials. The best performance across the methods is highlighted by the bold, and the best performance in our method is highlighted by the underline.

From the result we can see that a larger value of *tenure* improves the search performance of EAX-tabu. The appropriate setting depends on the instance, however, a remarkable improvement between when *tenure*=1 and *tenure*=3. A circulation where the the gain and loss of the same edges repeat has a possibility to occur when *tenure* is set to 1. It is supposed that EAX-tabu suppresses the occurrence such a circulation by enlarging the value of *tenure*.

V. CONCLUSION

In this paper, we proposed a new tabu scheme for EAX in order to solve instances involving the lattice pattern. Our tabu scheme inhibited repeating the exchange of the same edges in order to reduce the bias of edge to be exchanged between parents and enhance the diversity in offspring. We implement the tabu scheme to the original EAX and evaluated the effectiveness and the search performance of our method on TSPLIB instances and VLSI TSP. Through the experiments, we could find a significant improvement of EAX on VLSI TSP instances where the most cities are arranged in the lattice pattern. By enlarging the value of *tenure*, the search performance improved, however its appropriate setting depends on the instance. Further verification of the search performance and behavior analysis of the method is required. In addition, the effectiveness in large datasets including more than 10,000

TABLE III
COMPATISION OF EACH $tenure$ PARMETER (TSPLIB INSTANCES)

instance (#cities)	opt.	$tenure$	%opt.	Ave.	Gene.
rat575 (575)	6773	0*	0.73	6773.3	48.6
		1	0.73	6773.3	57.1
		3	0.87	6773.2	63.9
		5	0.90	6773.1	70.2
		7	0.90	6773.1	76.8
		10	**0.97**	6773.0	86.1
		EAX-Ent	-	-	-
u724 (724)	41910	0	0.73	41913.0	48.2
		1	0.97	41910.2	58.0
		3	0.87	41911.0	63.8
		5	0.97	41910.2	70.2
		7	0.97	41910.2	78.1
		10	**1.00**	41910.0	84.7
		EAX-Ent	-	-	-
vm1084 (1084)	239297	0	0.83	239303.0	48.1
		1	0.83	239305.0	56.5
		3	0.97	239298.0	62.3
		5	0.93	239299.0	67.1
		7	**1.00**	239297.0	72.4
		10	**1.00**	239297.0	80.8
		EAX-Ent	**1.00**	239297.0	109.0
pcb1173 (1173)	56892	0	0.77	56893.5	51.7
		1	0.87	239305.0	56.5
		3	1.00	56892.0	73.2
		5	0.97	56892.1	87.6
		7	0.90	56892.2	97.8
		10	0.80	56892.2	116.3
		EAX-Ent	**1.00**	56892.0	137.0
d1291 (1291)	50801	0	0.77	50805.2	51.8
		1	0.90	50803.8	52.9
		3	0.93	50803.7	58.4
		5	0.93	50803.4	64.7
		7	0.97	50802.4	69.7
		10	**0.97**	50803.3	77.3
		EAX-Ent	-	-	-
u1432 (1432)	152970	0	0.47	152977.8	70.6
		1	0.67	152976.2	76.1
		3	**0.80**	152973.6	87.1
		5	0.57	152977.8	109.9
		7	0.73	152974.8	132.2
		10	0.63	152988.7	178.3
		EAX-Ent	0.70	153505.4	132.0
d1655 (1655)	62128	0	0.13	62140.0	68.2
		1	0.13	62142.8	72.8
		3	**0.43**	62134.1	83.9
		5	0.37	62134.2	103.2
		7	0.37	62134.6	116.0
		10	0.40	62132.9	143.0
		EAX-Ent	-	-	-
vm1748 (1748)	336556	0	0.23	336654.8	56.8
		1	0.57	336592.2	75.0
		3	0.97	336556.9	80.3
		5	0.97	336556.1	91.3
		7	**1.00**	336556.0	99.3
		10	**1.00**	336556.0	115.5
		EAX-Ent	**1.00**	336556.0	164.0
u1817 (1817)	57201	0	0.00	57218.8	75.0
		1	0.33	57211.1	87.2
		3	0.57	57206.3	94.8
		5	**0.87**	57202.3	125.1
		7	0.80	57202.0	150.1
		10	0.63	57207.4	166.2
		EAX-Ent	-	-	-
u2152 (2152)	64253	0	0.23	64263.3	72.4
		1	0.73	64258.4	81.0
		3	0.77	64254.7	89.5
		5	0.73	64254.7	109.9
		7	0.73	64254.6	131.6
		10	**0.80**	64254.1	168.7
		EAX-Ent	-	-	-
pr2392 (2392)	378032	0	0.73	378042.1	60.8
		1	0.73	378043.0	82.3
		3	0.83	378037.0	99.0
		5	0.97	378033.1	119.3
		7	0.93	378034.0	137.5
		10	0.97	378033.1	162.5
		EAX-Ent	**1.00**	378032.0	263.0
pcb3038 (3038)	137694	0	0.00	137710.9	72.8
		1	0.03	137705.4	107.6
		3	0.20	137702.2	127.9
		5	0.40	137696.6	161.0
		7	0.47	137697.4	191.7
		10	0.43	137697.4	234.7
		EAX-Ent	**1.00**	137694.0	394.0
fnl4461 (4461)	182566	0	0.00	182642.0	101.4
		1	0.00	182603.7	143.6
		3	0.00	182591.5	177.4
		5	0.00	182582.7	217.9
		7	0.00	182577.9	264.7
		10	0.00	182577.1	324.2
		EAX-Ent	**0.90**	182584.3	825.0
rl5915 (5915)	565530	0	0.07	565590.2	66.7
		1	0.17	565562.1	92.8
		3	0.20	565558.1	111.6
		5	0.13	565554.9	138.2
		7	0.07	565556.8	157.2
		10	0.13	565551.7	195.5
		EAX-Ent	**0.75**	565925.9	319.0

*$tenure = 0$ means EAX-Rand.

cities should be evaluated. There are powerful techniques introduced the latest EAX such as the block strategy in combining *AB-cycles* and the entropy-based selection. More improvement of search performance is expected by combining of these techniques with the tabu scheme. These tasks are left as a future goal.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 17K00352 and 18K11469.

REFERENCES

[1] Mathias, K. and Whitley, D.: Genetic operators, the fitness landscape and the traveling salesman problem. Proc. 2nd Internat. Conf. Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 866, pp. 219–228 (1992).

[2] Maekawa, K., Mori, N., Tamaki,H., Kita, H. and Nishikawa, H.: A Genetic Solution for the Traveling Salesman Problem by Means of a Thermodynamical Selection Rule. Proc. 1996 IEEE International Conference on Evolutionary Computation, pp. 529–534 (1996).
[3] Yamamura, M., Ono, I. and Kobayashi, S.: Emergent Search on Double Circle TSPs using Subtour Exchange Crossvoer, Proc. of 1996 IEEE International Conference on Evolutionary Computation, pp. 535–540 (1996).
[4] Merz, P. and Freisleben, B.: Genetic local search for the TSP: New results. Proc. 1997 IEEE Internat. Conf. Evolutionary Comput., pp. 159–164 (1997).
[5] Nagata, Y and Kobayashi, S.:Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem. Proceedings of 7th International Conference on Genetic Algorithms, Vol. 14, No. 5, pp. 450–457 (1997)
[6] Ikeda, K. and Kobayashi, S.: Deterministic Multi-step Crossover Fusion: A Handy Crossover for GAs, Proc. of Parallel Problem Solving fron Nature, PPSN VII , pp. 162–171 (2002).
[7] Hanada, Y., Hiroyasu, T. and Miki, M.: Genetic Multi-step Search in Interpolation and Extrapolation domain, Proc. The Genetic and Evolutionary Computation Conference 2007, pp. 1242–1249 (2007).
[8] Whitley, D., Hains, D. and Howe, A.: A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover. Proc. 11th Internat. Conf. Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 6323, pp. 566–575 (2010)
[9] Ting, CK.: Improving Edge Recombination through Alternate Inheritance and Greedy Manner, Evolutionary Computation in Combinatorial Optimization, Vol. 3004, pp. 210-219 (2004)
[10] Segura, C., Betello Rionda, S., Hernández Aguirre, A., Valdez Peña, S. I.: A Novel Diversity-based Evolutionary Algorithm for the Traveling Salesman Problem, Proc. of the 2015 Annual Conf. on Genetic and Evolutionary Computation, pp. 489-496 (2015)
[11] Nagata, Y.: New EAX crossover for large TSP instances, Proc. of Parallel Problem Solving fron Nature, PPSN IX, pp. 372–381 (2006).
[12] Nagata, Y.: Fast EAX algorithm considering population diversity for traveling salesman problems. Proc. 6th Internat. Conf. Evolutionary Comput. Combinatorial Optim., Lecture Notes in Computer Science, Vol. 3906, pp. 171–182 (2006).
[13] Nagata, Y and Kobayashi, S.: A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem, INFORMS Journal on Computing, 25(2), pp. 346–363, (2013).
[14] Lin, S. and Kernighan, B. W.: An effective heuristic algorithm for the traveling salesman problem. Oper. Res. 21(2), pp. 498–516 (1973).
[15] Applegate, D., Cook, W. and Rohe, A.: Chained Lin-Kernighan for large traveling salesman problems. INFORMS J. Comput. 15(1), pp.82–92 (2003).
[16] Helsgaun, K.: General k-opt submoves for the Lin-Kernighan TSP heuristic. Math. Programming Comput. 1(2), pp. 119–163 (2009).
[17] TSPLIB 95 : http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
[18] VLSI TSP : http://www.tsp.gatech.edu/world/countries.html