DOI:10.1587/transfun.2024EAL2025

Publicized:2024/06/24

LETTER

Edge Assembly Crossover Incorporating Tabu Search for the Traveling Salesman Problem

Maaki SAKAI[†], **Kanon HOKAZONO**[††], ***and*** **Yoshiko HANADA**[††a)], ***Nonmembers***

SUMMARY In this letter, we propose a method to introduce tabu search into Edge Assembly Crossover (EAX), which is an effective crossover method in solving the traveling salesman problem (TSP) using genetic algorithms. The proposed method, called EAX-tabu, archives the edges that have been exchanged over the past few generations into the tabu list for each individual and excludes them from the candidate edges to be exchanged when generating offspring by the crossover, thereby increasing the diversity of edges in the offspring. The effectiveness of the proposed method is demonstrated through numerical experiments on medium-sized instances of TSPLIB and VLSI TSP.

key words: *traveling salesman problem, genetic algorithm, tabu search*

1. Introduction

For tackling combinatorial optimization problems using a Genetic Algorithm (GA) [1], it is important to design a crossover method that effectively passes on good partial solutions (traits) to offspring. In the past, a number of crossover methods that focus on the inheritance of edge traits have been proposed for solving Traveling Salesman Problem (TSP) [2]–[6]. Edge Assembly Crossover (EAX) has proven to be the most successful crossover method among them [3], [6].

EAX maintains preferable traits by generating offspring with only edges of both parents to the extent possible while obtaining new traits by limited recombination of the edges between the two. EAX can search the solution space with high accuracy by increasing the diversity of individuals in the population, i.e., the diversity of edges included in the population. In [6], Nagata et al. have focused on the diversity of trait inheritance and introduced a survival selection strategy for offspring that form the next generation in order to ease the loss of edge diversity in the population. This strategy has high search performance for very large instances of TSP, outperforming other promising heuristic methods [6]. In addition to the trait inheritance, the diversity of trait acquisition is also important. Increasing the diversity of offspring generated throughout the search is expected to improve the search performance. However, a direct approach that enhances the diversity of offspring in EAX has not yet been proposed.

In this letter, to improve the diversity of exchanged edges, i.e., the diversity of offspring, in EAX, we propose *EAX-tabu* [7], which incorporates a tabu search algorithm [8] into the EAX process. In the proposed method, the edges exchanged by EAX over the previous several generations are recursively recorded for each individual as tabu edges. The proposed method prohibits to exchange the tabu edges when generating offspring, which prevents bias in the exchanged edges while ensuring the acquisition of diverse traits throughout the search. In this letter, we verify the effectiveness of EAX-tabu for medium-scale instances of TSPLIB [9] and VLSI TSP [10]. Through a series of experiments, we show that incorporating the tabu search algorithm into EAX improves the search performance.

[†]MicroAd, Inc.
[††]Kansai University
a) E-mail: hanada@kansai-u.ac.jp

2. Traveling Salesman Problem

TSP can be described as follows: Given a set of n cities (vertices) $V = \{v_1, \cdots, v_n\}$ and distances between cities $w(v_i, v_j)$ $(1 \leq i, j \leq n)$, we need to find the shortest tour forming Hamilton cycle $H = (v_{(1)}, \cdots, v_{(n)})$, where $v(i)$ denotes the ith visited city starting from an arbitrary city in V. If we let $\{v_i, v_j\}$ be the edge between v_i and v_j, then a tour is represented as a set $E = \{\{v_{(i)}, v_{(i+1)}\} | 1 \leq i \leq n,\ v_{(n+1)} = v_{(1)}\}$ of edges. The objective function value is defined by the tour length calculated by $\sum_{i=1}^{n} w(v_{(i)}, v_{(i+1)})$.

3. Edge Assembly Crossover

3.1 Flow of EAX

By incorporating edges of one parent into the tour of another parent, EAX acquires new traits without destroying favorable traits of both the parents with relatively limited change. Let E_A and E_B be sets of edges constituting the tours of parent individuals p_A and p_B, respectively. The flow of EAX that generates offspring c_A and c_B from p_A and p_B is described as follows.

Step 1 Construct a multigraph $G_{A,B} = (V, E_A \cup E_B)$.
Step 2 Apply *AB-cycle decomposition* (discussed in Section 3.2) to $G_{A,B}$ and obtain a set D of *AB-cycles*.
Step 3 Generate a subset E_{set} of D according to some criterion. Let $g_{AB}^{(i)}$ and $E^{(i)}$ $(1 \leq i \leq |E_{set}|)$ be ith *AB-cycle* constituting E_{set} and the set of edges of each $g_{AB}^{(i)}$, respectively.
Step 4 Generate *intermediate individuals* I_A and I_B by applying $E_A \leftarrow E_A \oplus E^{(i)}$ and $E_B \leftarrow E_B \oplus E^{(i)}$ $(1 \leq i \leq |E_{set}|)$ to the tours of p_A and p_B, respectively.
Step 5 Apply *Repairing* (discussed in Section 3.3) to I_A and I_B and obtain offspring c_A and c_B.

3.2 AB-cycle Decomposition

AB-cycle, represented by g_{AB} in the flow of EAX, is a closed path consisting of an even number of edges, which can be obtained by alternately tracing edges in E_A and E_B on the multigraph $G_{A,B}$. *AB-cycle* is the most important element of EAX, which is used in extracting candidate edges to be exchanged between parents. The procedure of the *AB-cycle decomposition* in Step 2 of the flow is shown in Algorithm 1. In this algorithm, $d(v)$ denotes the degree of each city v in $G_{A,B}$; all initial values are 4. Lines 4 to 12 are the operations to extract one *AB-cycle* from $G_{A,B}$. Here, each *AB-cycle* is treated as a graph (V', E'), where V' and E' are subsets of cities V and edges $E_A \cup E_B$ in $G_{A,B}$, respectively. All cities and edges in $G_{A,B}$ are decomposed so that they belong to one of the *AB-cycles*, yielding a set D of *AB-cycles*.

Each *AB-cycle* constituting D represents a set of candidate edges to be exchanged. In actuality, EAX generates E_{set} which is a subset of D. Thus, the exchanged edges between parents correspond to edges of the *AB-cycles* included in E_{set}. There are two simple ways to construct E_{set}: *EAX-RAND* [3] and k-multiple strategy [6]. EAX-RAND incorporates *AB-cycles* included in D into E_{set} with a probability of 0.5. The k-multiple strategy selects a small number of *AB-cycles* randomly from D to generate offspring locally. EAX-RAND can generate a wide variety of offspring. However, the size of E_{set} follows a binomial distribution, so that an extremely small E_{set} is hard to construct.

We here consider a method to construct E_{set} of arbitrary size with equal probability to enhance the local search property while keeping the diversity of offspring. The size of E_{set} is uniformly, randomly selected from among $[1, |D|]$ for each pair of parents. Then *AB-cycles* are randomly chosen from D according to the selected number to make up E_{set}. Hereafter, this method is referred to as *EAX-UNIFORM*.

3.3 Intermediate Individual and Repairing

In Step 4 of the flow, for all edges included in E_{set}, EAX deletes edges belonging to E_A from the tour of p_A and adds edges belonging to E_B, thereby generating a set of multiple decomposed subtours, or partial cycles. The set thus obtained is called *intermediate individual*. Let intermediate individual generated from p_A by this operation be I_A. This operation of exclusive superposition of edges in E_{set} into the tour of one parent is also used to generate the intermediate individual I_B from p_B.

The *Repairing* is then applied to these generated intermediate individuals in Step 5. The procedure of *Repairing* is shown in Algorithm 2, where intermediate individual I is represented as a set of subtours $\{S_1, \cdots S_m\}$ and the distance of edge $e_{ij} = \{v_i, v_j\}$ is abbreviated as $w(e_{ij})$. This operation iteratively connects subtours of an intermediate individual so that a single feasible tour is formed as an offspring. Each iteration generates edges that connect two close subtours with the shortest distance between them.

Algorithm 1 AB-cycle Decomposition
```
1: $V \leftarrow \{v_1, \cdots, v_n\},\ D \leftarrow \emptyset,\ {}^\forall v \in V; d(v) \leftarrow 4.$
2: while $|V| > 0$ do
3:     $V' \leftarrow \emptyset,\ E' \leftarrow \emptyset.$
4:     Select one vertex $v \in V$ at random and set $v_s \leftarrow v$.
5:     repeat
6:         Select at random one edge $e_A = \{v, v'\} \in E_A$ with $v$ as an
           endpoint. $d(v) \leftarrow d(v)-1,\ d(v') \leftarrow d(v')-1.$
7:         $E' \leftarrow E' \cup \{e_A\},\ V' \leftarrow V' \cup \{v'\}.$
8:         $E_A \leftarrow E_A \setminus \{e_A\}$ and $v \leftarrow v'$.
9:         Select at random one edge $e_B = \{v, v'\} \in E_B$ with $v$ as an
           endpoint. $d(v) \leftarrow d(v)-1,\ d(v') \leftarrow d(v')-1.$
10:        $E' \leftarrow E' \cup \{e_B\},\ V' \leftarrow V' \cup \{v'\}.$
11:        $E_B \leftarrow E_B \setminus \{e_B\}$ and $v \leftarrow v'$.
12:    until $v_s \neq v$
13:    $D \leftarrow D \cup (V', E').$
14:    Remove all $v$ from $V$ such that $d(v) = 0$.
15: end while
```

Algorithm 2 Repairing Intermediate Individual
```
1: while $|I| > 1$ do
2:     Find the partial subtour $S^* = (V^*, E^*) \in I$ where $|V^*|$ is minimum
       and $I \leftarrow I \setminus S^*$.
3:     Solve $min_{S^{**} \in I}\{-w(e) - w(e') + w(e'') + w(e''')\}$ and find the
       subtour $S^{**} = (V^{**}, E^{**}) \in I$, where edges $e$ and $e'$ indicate
       $\{v_1, v_2\} \in E^*$ and $\{v_3, v_4\} \in E^{**}$, then $e''$ and $e'''$ indicate edges
       $\{v_1, v_3\}$ and $\{v_2, v_4\}$.
4:     $V^{**} \leftarrow V^{**} \cup V^*,\ E^{**} \leftarrow (E^{**} \cup E^*) \setminus \{e, e'\} \cup \{e'', e'''\}.$
5: end while
```

4. Proposed Method

As the population moves toward convergence, the exchange edges in EAX become biased and the ability to generate new traits decreases. To ease the bias, we propose EAX-tabu by incorporating a tabu search scheme into EAX [7]. Tabu search is a local search that works effectively on combinatorial optimization problems, including TSP [8]. EAX is highly compatible with tabu search since it does not significantly perturb the tours of parents, and changes in edges are almost exclusively limited to the edges possessed by the parents. Tabu search focuses on the change of a single search point. When generating neighborhood solutions, it refers to a tabu list that records changes over a certain period in the past called *tabu tenure* in order to prohibit the repetition of the same change. To incorporate the concept of tabu search into GA using the multipoint search in this letter, past changes for each individual are archived in the tabu list.

Edge changes in individuals in EAX are primarily caused by the acquisition and loss of edges due to the application of *AB-cycles* to the tours. Therefore, by archiving the set of *AB-cycles*, i.e., E_{set}, applied in the past for each individual, EAX can refer to the past E_{set} as a tabu list. The flow of the proposed EAX-tabu is as follows, where the archives of parents p_A and p_B are denoted as E^A_{pre} and E^B_{pre}. The archive of each individual in the initial population is initialized with $\emptyset$. Since each offspring obtains more edges from one of its parents, it inherits the archive of that parent.

Step 1-2 Same processes as Step 1-2 of EAX (section 3.1)
Step 3 $E_{pre} \leftarrow E^A_{pre} \cup E^B_{pre}$.
Step 4 Add edges included in E_{pre} as tabu edges to the tabu list E_T according to some criterion.
Step 5 Remove *AB-cycles* whose edges are contained in E_T from D.
Step 6-8 Same processes as Step 3-5 of EAX
Step 9 $E^A_{pre} \leftarrow E^A_{pre} \cup E_{set}$, $E^B_{pre} \leftarrow E^B_{pre} \cup E_{set}$, and set E^A_{pre} and E^B_{pre} as the archives of c_A and c_B, respectively.
Step 10 Delete *AB-cycles* that have expired after a given retention period (i.e., tabu tenure) from E^A_{pre} and E^B_{pre}.

In Step 5, all edges contained in E_{pre} are selected as tabu edges with a probability of 0.5 to constitute a tabu list E_T. Since E_{pre} keeps each *AB-cycle* as a graph, some edges could be redundantly archived in the tabu list. This means that edges that exist in the archive more frequently, i.e., edges that have been exchanged more frequently in the past, are more likely to be selected as tabu edges.

5. Numerical Experiments

We evaluated the search performance of EAX-tabu by comparing it with that of EAX-UNIFORM. In EAX-tabu, after removing *AB-cycles* from D according to the tabu list E_T, E_{set} is constructed in the same way as EAX-UNIFORM. In the experiments, we used 10 medium-scale instances from TSPLIB and VLSI TSP, respectively. For both EAXs, the generation alternation model in the GA was the same as that used in [4], [6], where the population size was set to 300 and the number of offspring generated by one pair of parents in the crossover was set to 200. In the case where the best solution in the population was not updated for 30 generations, the search was terminated as having achieved convergence.

5.1 Performance of EAX-tabu

Table 1 shows the number of trials that produced the optimal solution and the average number of generations required for convergence in EAX-tabu and EAX-UNIFORM ("EAX" in the table) for each of the 10 instances. These results are from 30 trials. For EAX-tabu, a tabu tenure of 5 was employed across the experiment.

From Table 1, we observe that the proposed method finds optimal solutions with a high probability compared to EAX-UNIFORM in most instances. In addition, the proposed method increases the number of generations required for the convergence of the search, which means the ability of the population to generate new solutions is maintained longer by the tabu scheme.

Among the instances, no improvement in performance was observed in fnl4461 (TSPLIB). Figure 1 shows the transition of the number of *AB-cycles* generated in one pair of parents in solving fnl4461 by EAX-tabu. For comparison, it shows the transition in bgb4335 (VLSI TSP), which has the same scale with respect to the number of cities as fnl4461 and

Table 1 Comparisons of the number of trials producing the optimum and the number of generations for convergence

TSPLIB			VLSI TSP		
Instance	EAX	EAX-tabu	Instance	EAX	EAX-tabu
rat575	22 (49.8)	27 (71.0)	icw1483	29 (50.1)	30 (79.5)
vm1084	25 (48.2)	28 (65.6)	djc1785	20 (57.6)	30 (106.2)
d1291	23 (44.7)	28 (61.7)	xpr2308	1 (60.0)	21 (126.3)
u1432	14 (52.9)	17 (109.9)	mlt2597	15 (59.4)	30 (104.9)
vm1748	7 (56.0)	29 (91.0)	lsm2854	10 (66.2)	27 (134.6)
u2152	7 (56.6)	22 (110.7)	fdp3256	0 (68.2)	28 (144.2)
pr2392	22 (61.5)	29 (120.7)	ltb3729	2 (71.7)	25 (151.2)
pcb3038	0 (73.8)	12 (159.9)	bgb4355	8 (76.4)	28 (186.8)
fnl4461	0(102.0)	0 (212.1)	xqd4966	1 (79.5)	18 (166.9)
rl5915	2 (66.0)	4 (136.9)	fea5557	8 (81.1)	27 (205.4)

trials (generations)

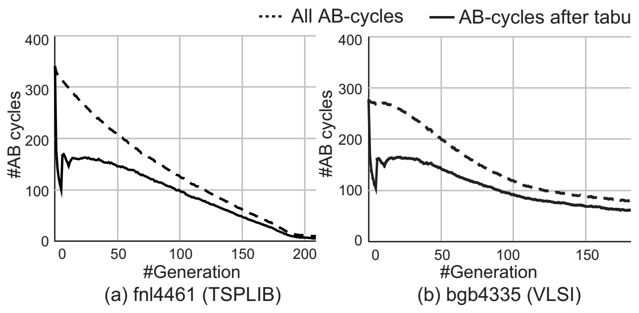

Fig. 1 Transition of the number of AB-cycles

where the tabu scheme worked well. In the figure, the dashed line represents the average number of all the *AB-cycles* generated by the *AB-cycle* decomposition process, and the solid line represents the average number of *AB-cycles* after being removed by the tabu list. Both are the results of a typical trial. Figure 1 shows that although the total number of generated *AB-cycles* decreases with convergence in bgb4335, the number of *AB-cycles* is kept high even after removing some AB-cycles that contain tabu edges. In contrast, in fnl4461, the number of *AB-cycles* decreases as the search progresses, and is finally close to 0. Thus, in such a case, the tabu scheme does not work well.

5.2 Effectiveness of Tabu

In general, it is known that restricting the number of *AB-cycles* applied to offspring improves search performance in EAX [6]. To demonstrate the superiority of the proposed method over a straightforward approach that just limits *AB-cycles* selection, we conducted a comparative analysis between EAX-tabu and EAX reducing the selection probability of *AB-cycles* named *EAX-limit*. In the 20 instances used in the experiments, we observed that *AB-cycles* are removed in each pair of parents in EAX-tabu at an average rate of 0.3 to 0.5 (TSPLIB) and 0.3 to 0.4 (VLSI TSP) throughout all generations. Moreover, they have been removed at a maximum rate of 0.6 to 0.7. From these trends, EAX-limit randomly selected *AB-cycles* in the proportions of 0.3, 0.5, and 0.7 to the entire set and then some of those are randomly incorporated into E_{set} as in EAX-UNIFORM.

Table 2 Performance Comparison of EAX-tabu (tabu) and EAX-limit

TSPLIB	EAX-limit			tabu	VLSI TSP	EAX-limit			tabu
	0.3	0.5	0.7			0.3	0.5	0.7	
rat575	24	23	20	27	icw1483	30	30	30	30
vm1084	27	23	25	28	djc1785	29	26	23	30
d1291	28	30	30	28	xpr2308	5	4	9	21
u1432	13	14	12	17	mlt2597	18	15	20	30
vm1748	10	11	13	29	lsm2854	19	15	18	27
u2152	1	1	5	22	fdp3256	1	0	0	28
pr2392	22	19	16	29	ltb3729	6	9	4	25
pcb3038	1	0	1	12	bgb4355	3	3	4	28
fnl4461	0	0	0	0	xqd4966	3	0	0	18
rl5915	4	4	3	4	fea5557	17	16	9	27

Table 2 shows the results of comparing the number of trials that produced the optimal solution. These results are from 30 trials. From the results of EAX-limit in Table 2 and those of EAX-UNIFORM (“1.0” under EAX-limit) shown in Table 1, we can confirm that restricting the number of *AB-cycles* improves the performance of EAX in some instances. For vm1748, u2152, and pcb3038 in TSPLIB and most instances of VLSI TSP, the simple suppression of *AB-cycles* did not show a significant improvement in the search performance, and EAX-tabu showed superior performance to EAX-limit. These instances have city layouts of a lattice pattern or a straight line, which have a strong edge dependence among neighborhood cities, and the diversity of exchange edges has a significant impact on the performance of the crossover. In such instances, the tabu scheme is deemed effective, as it restricts the exchange edges while preserving diversity across successive generations.

5.3 Influence of Tabu Tenure

Here, we examine the effect of tabu tenure on search performance. Figure 2 shows the proportion of trials that found optimal solutions in 8 instances of TSPLIB and VLSI TSP. These results are from 30 trials; tabu tenure was set to 1, 3, 5, 7, and 10. The other search conditions are the same as in the previous sections. Note that tenture=0 corresponds to EAX-UNIFORM. For comparison, we refer to the results of the most recent EAX that uses edge entropy-based survival selection for improving the diversity of trait inheritance (labeled with *EAX-ent*). The results of EAX-ent in instances of TSPLIB are from [6]; we identify these instances by attaching an asterisk ‘*’ to their labels.

Figure 2 illustrates that augmenting the tabu tenure value in EAX-tabu can enhance its search performance, albeit with variation depending on the instance. The performance enhancement diminishes with larger tabu tenure values in VLSI TSP instances where diversity of exchange edges is required. This is attributed to a reduction in available *AB-cycles* within the crossover due to the expansion of the tabu list. It is also worth noting the exceptional performance of EAX-ent for the fnl4461 instance, which indicates that, in addition to the crossover, maintaining a diversity of trait inheritance in the selection is important.

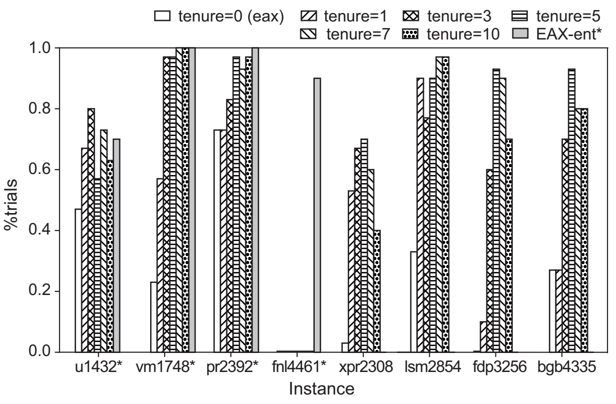

Fig. 2 Influence of tabu tenure setting on search performance

6. Conclusion

In this study, we proposed EAX-tabu that introduces a tabu search scheme into EAX. Through numerical experiments conducted on medium-scale instances of TSPLIB and VLSI TSP, we showed that the tabu scheme can significantly improve the performance of EAX, regardless of the tabu tenure setting. Further preliminary experiments indicated that the tabu scheme also improves the performance of EAX-RAND and EAX-5AB which is a part of the k-multiplex strategy. Moreover, Nagata’s edge entropy-based survival selection [6], emphasizing trait inheritance diversity, has proved highly effective and can be seamlessly integrated with the tabu scheme proposed in our study. However, the evaluation of EAX-tabu’s search performance on larger instances and its compatibility with the edge entropy-based survival selection method are left to future work.

References

[1] J. H. Holland, “Adaptation in Natural and Artiffcial Systems”, University of Michigan Press, Ann Arbor, MI., 1975.
[2] P. Merz, and B. Freisleben, “Genetic local search for the TSP: New results”. Proc. IEEE Internat. Conf. Evolutionary Comput., pp. 159–164, 1997.
[3] Y. Nagata, and S. Kobayashi, “Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem”, Proc. 7th Internat. Conf. on Genetic Algorithms, pp. 450–457, 1997
[4] K. Ikeda, and S. Kobayashi, “Deterministic Multi-step Crossover Fusion: A Handy Crossover for GAs”, Proc. Parallel Problem Solving from Nature VII, pp. 162–171, 2002.
[5] D. Whitley, et al., “A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover”, Proc. Parallel Problem Solving from Nature XI, pp. 566–575, 2010.
[6] Y. Nagata, and S. Kobayashi, “A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem”, INFORMS Journal on Computing, 25(2), pp. 346–363, 2013.
[7] M. Sakai, Y. Hanada, and Y. Orito, “Edge Assembly Crossover with Tabu for Traveling Salesman Problem”, Proc. IEEE Internat. Conf. Evolutionary Comput., No. 19931394, pp. 1–6, 2020.
[8] F. Glover, “Tabu Search - Part 1”. ORSA Journal on Computing. 1 (2), pp. 190–206, 1989.
[9] TSPLIB, `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/` (Last accessed on 2024/03/14)
[10] VLSI TSP, `https://www.math.uwaterloo.ca/tsp/vlsi/` Last accessed on 2024/03/14)