An analysis of Edge Assembly Crossover for the Traveling Salesman Problem

Yuichi Nagata* and Shigenobu Kobayashi*

***Graduate School of Interdisciplinary Science and Engineering, Tokyo Institute of Technology, Yokohama 226-8502, Japan.**
nagata@fe.dis.titech.ac.jp, kobayasi@dis.titech.ac.jp

Abstract

The GA with using Edge Assembly Crossover(EAX) have shown a good performance for Traveling Salesman Problems(TSPs). This paper will examine why EAX brings good performance to GAs.

Many crossovers for TSPs have been proposed so far. We first examine advantages of EAX by comparing it with other crossovers, and confirm some requirements to design a good crossover.

Next, we examine what advantages the EAX has being compared with other representative operators for TSPs.

1 Introduction

Genetic algorithms(GAs) have been applied to various problems so far. However, in representative combinatorial optimization problems most of GAs have not been comparable with local search, simulated annealing and tabu search, etc which use problem-specific neighborhood operators. Then, what possibilities GAs have to surpass these search methods which use problem-specific neighborhood operators. If certain features of problem domain are known in advance, problem-specific crossovers should be designed in consideration of these features to obtain good performance, and several problem-specific crossovers have been proposed so far.

A problem-specific crossover is regarded as an operator to define neighborhood for a couple of solution candidates in the search space. Neighborhood search operators are usually defined to a single solution candidate in the search space and should be designed with using problem-specific knowledges so that the neighbors of a good solution tends to have good solution in it. So it is similarly desirable that the neighborhood of a good pair of parents defined by the crossover tends to have good offsprings in it.

To design a problem-specific crossover, it is intuitionally important that offsprings inherit appropriate *characteristics* from parents. *Characteristics* are substructures of solution candidates and should be selected so as to have the following qualities.

(1) Substructures whose existing probability in a solution candidate tend to be higher and higher with its fitness being higher.

(2) An influence of the substructure on the fitness tend to be independent of other substructures in a solution candidate.

For example, in one-max function on the bit-string representation in which the evaluation of a solution candidate is simply the number of 1-bits in it, the variables of each bits are ideal for *characteristics*. But it would be more simple and effective to use bit-flipping neighborhood than standerd crossovers such as one-point or uniform crossovers, etc. Moreover, it would be difficult to find such *characteristics* in advance at interesting problems.

Traveling salesman problems(TSPs) are one of the most representative combinatorial optimization problems. Let $G = (V, E, w)$ to be a weighted complete graph with n vertices, where V, E and w are a set of vertices, a set of edges and weights of edges respectively. The goal of TSPs is to find an hamilton cycle(tour) H that minimizes the tour length $\sum_{e \in H} w(e)$.

In TSPs, edges constructing a tour are appropriate for *characteristics* because the shorter the length of the tour, the higher the existing probability of the edges included in the tour and the influence on the tour length of an edge e can be determined as its length $w(e)$ independent of other edges in the tour except each edges in the tour have interactions that must form a hamilton cycle. Therefore several crossovers that use edges as *characteristic* have been proposed so far. Howevere more simple neighborhood operators based on edges, k-opt [2] and LK neighborhood [3], etc were proposed and it had seemed to be more effective. Therefore, it had been questionable that the GA using crossovers have any advantages against the local searchs, simulated annealings and tabu search, etc using such neighborhood operators.

Fortunately the GA using crossover EAX shown comparable or better performance to other sophisticated methods in TSPs [1]. These results indidate that the GAs with an appropriate crossover may have great potential as optimization methods for certain problem domain. In the following section, we examine what factors of the EAX contributes the good performance of the optimization for TSPs.

2 A Comparison between Crossovers

2.1 Several crossovers for TSPs

So far, several crossovers, EXX [4], EAX [1] and EX [5], etc in which edges of parents are inherited to offsprings from its parents have been proposed for TSPs.

0-7803-5731-0/99$10.00 ©1999 IEEE

In this section, we examine why EAX shows good performance being compared with other crossovers from the viewpoints of inheritance of edges. The algorithm of EAX is displayed in appendix.

In general, an quality of offsprintg generated by a crossover is higher with the ratio of edges inherited from parents to offsptings being higher, but the variety of offsprings is narrower. It is desirable for neighborgoods that a probability of improvement par an arbitrary neighber is high and the number of neighbor is large. Consider a local search for example, by using such an neighborhood, current solution candidate tend not to be trapped in locally optimal solutions and can move improbed one by few evaluation. Thus, the trade-off between the ratio of edges inherited from parents to its offsprings and the variety of offsprings is important. In the following, we will compare the crossover EXX, EAX and EX from this point of view.

Offsprings generated by EXX are completely composed of edges of parents. Though EAX generates offsprings with few edges that are not included in both parents, these new edges are selected as short as possible in modification procedure(See Appendix). So this kind of edges generated by EX are long and have bad influence on a tour length of offsprings, we used 2-opt local search after applying EX to modify these long edges. We call this procedure EX-2opt. Since improvement in a quality of tour length was hardly found by using 2-opt local search after applying EXX or EAX on condition that parents are refined better than 2-opt locally optimal solution , 2-opt local search was not used for EXX and EAX.

2.2 The comparison of performance

First, we compare the performance of above crossovers. The experiments were performed under the following conditions.

(1) Set $t = 0$. Generate N_{pop}(popluration size) solution candidates using 2-opt local search. Let these $\Omega(t)$.

(2) Make $N_{pop}/2$ pairs of parents by random sampling from $\Omega(t)$ without replacement.

(3) For each pair of parents, applying crossover by N_{cross} times and put best two between offsprings and parents to $\Omega(t+1)$.

(4) If no improvement of shortest tour length in population is not observed over 40 generation, then stop, else let $t = t + 1$ and go to (2).

So EXX generates offsprings systematically, all offsprings included in the neighborhood of EXX were generated at step (3).

We used instances, att532(532 city problem) and rat575(575 city problem) [6] in this experiments. N_{pop} was set to be 250 and 300 for att532 and rat575 respectively, and 10, 100 and 500 were tested as N_{cross} for both instances. 30 runs were performed for each parameters.

Table 1: The performance of GA using each crossovers

att532(optimun:27686)				
	N_{cross}	Best	Average	Gen
EX-2opt	10	27693[0]	27704.5(0.067%)	159.0
	100	27693[0]	27702.0(0.058%)	47.8
	500	27693[0]	27696.6(0.038%)	26.6
	1000	27693[0]	27694.2(0.027%)	22.7
EAX	10	27686[3]	27696.7(0.039%)	25.0
	100	27686[12]	27691.4(0.020%)	15.2
	500	27686[15]	27689.9(0.014%)	12.9
EXX	—	27715[0]	27767.2(0.293%)	214.8

rat575(optimun:6773)				
	N_{cross}	Best	Average	Gen
EX-2opt	10	6775[0]	6780.3(0.108%)	327.1
	100	6774[0]	6776.2(0.047%)	86.1
	500	6773[3]	6774.5(0.022%)	46.3
EAX	10	6773[1]	6775.4(0.035%)	26.9
	100	6773[8]	6774.2(0.017%)	16.8
	500	6773[10]	6774.0(0.015%)	14.1
EXX	—	6787[0]	6804.0(0.457%)	259.0

The results of these experiments are shown in Table ??. In this table, a best tour length and the number of trials leading to an optimal solution over 30 runs are shown in a column *Best*. The average tour length over 30 runs and its percentage of excess from the optimum are displayed in a column *Average*. *Gen* displays the average number of generation until the best tour is found by a single run.

The results shows that the performance of EAX is relatively good being compared with other crossovers.

2.3 Empirical comparison of the trade-off

In this subsection, these crossovers will be compared from the viewpoints of the trade-off between the ratio of inheritance of edges and the variety of offsprings. For each crossover, the following experiments were performed. We used rat575 as an instance.

(1) For $t = 0, 1, 2, ...$, the following procedure (2)-(4) is performed.

(2) Prepare $\Omega(t)$ obtained by the previous subsection with using EAX. And generates 150 pairs of parents by random sampling from $\Omega(t)$ without replacement.

(3-a) For each pair of parents, applying a crossover and take following data of offsprings.

- e_{share}:the ratio of edges included in both of parents.
- e_{more}:the ratio of edges included in one parent which is larger value.
- e_{less}:the ratio of edges included in one parent which is fewer value.

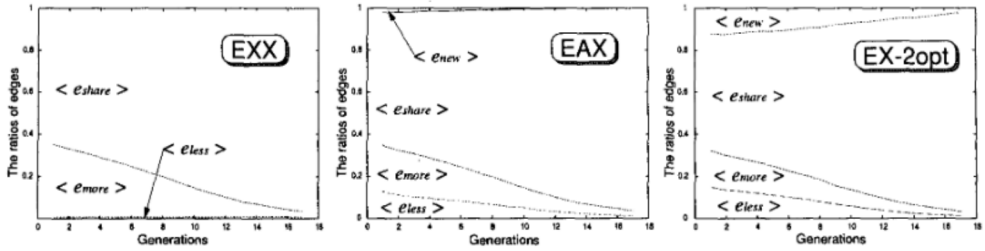

Figure 1: $< e_{share} >, < e_{more} >, < e_{less} >$ *and* $< e_{new} >$ are shown.

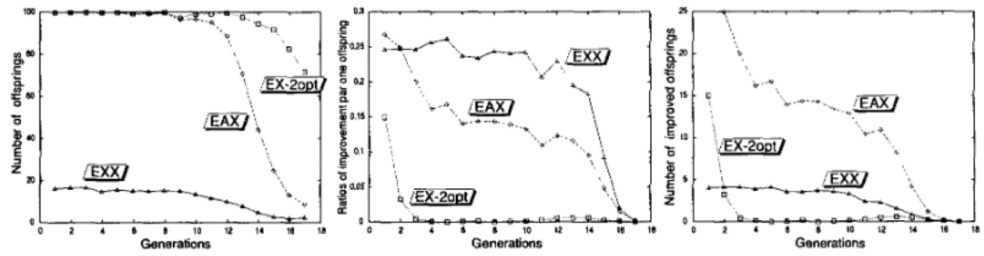

Figure 2: $< Num_{ch} >$(left), $< Raito_{imp} >$(middle) and $< Num_{imp} >$(right) are shown.

- e_{new}:the ratio of edges included in none of parents.

(3-b) For each pair of parents, applying crossovers by 100 times and take following data of offsprings.

- Num_{ch}:the number of offsprings which are different each other and not identical to parents.
- Num_{imp}:the number of improved offspring which are different each other.
- $Ratio_{imp}$:the ratio of improvements per one offspring, that is Num_{imp}/Num_{ch}.

(4) For each generation t, take averages for each data. We denote the averages of data X as $< X >$.

* Similarly to previous subsection, EXX generated all possible offsprings from a pair of parents.

In Figure 1, $< e_{share} > + < e_{more} > + < e_{less} >$ displays the averaged ratios of edges inherited to offsprings from parents. These ratios are higher and higher in EXX, EAX and EX-2opt order. Moreover, Figure 2(middle) shows that ratios of improvement per one offspring are higher and higher in the same order. These results indicate that a crossover superior in the inheritance of edges tends to generate better offsprings par a single offspring. However Figure 2(left) shows that the variety of the offsprings generated by each crossovers are fewer in the same order. Thus there exists the trade-off between the ratios of inheritance of edges and the variety of the offsprings. Consequently, Figure 2(right) shows that EAX can generate largest number of improved offsprings than the other crossover by the 100 crossovers.

EAX can generate relatively large number of improved offsprings due to the appropriate trade-off between the ratio of inheritance and the variety of the offsprings on condition that the number of crossover is limited to 100.

Since the population lose its diversity as the generations progress, identical offsprintgs tend to be generated at the last stage of searchs. So EAX could no longer increase Num_{ch} effectively by increasing Num_{cross} from 100 to 500 at the last stage of searchs, GA that used EAX with setting N_{cross} to 500 hardly improve the quality of tour length against those with 100. On the other hand, EX-2opt could increase Num_{ch} by increasing N_{cross} from 100 to 500 due to the wide vriety of offsprings, and GA using EX-2opt with setting N_{cross} to 500 superior to those with 100 in a qualitiy of tour length. So Num_{ch} of EXX was too few, the GA using EXX was inferior to the GA using other crossovers.

3 A Comparison between EAX and Neighborhood operators for TSPs

In this section, we examine what advantages the EAX has being compared with the representative neighborhood operators for TSPs. Neighborhood operators usually define the neighborhood for one solution candidate, while crossovers are regarded to define it for two solution candidates. We adopted k-opt neighborhood [2] and LK neighborhood [3] as the ordinary neighborhood operators.

The k-opt neighborhood is defined by deleting k

edges from a current tour and then reconnecting to valid tour in the other possible way. The larger k is, the larger the number of improved solution candidates and the smaller the ratio of improvement per one arbitrary solution candidate in the neighborhood. In other words, when k is large, current solution candidate tends not to be trapped in locally optimal solution, but the number of neighbor explode exponentially. The LK neighborhood employs alternation of k-opt neighborhood with variable k, which realize large k without explosion of neighbors using depth-first search in edges based on effective heuristics. In relation to k-opt neighborhood, if we restrict k-opt neighborhood to secqential k-opt neighorhood[3], there exist effective condition that reduce the sub-space not including any improved neighbors [3]. In the following, when k-opt neighborhood is used we used this secuential k-opt neighborhood to save the CPU time. While the sequential k-opt neighborhood misses improved solution candidates in non-sequential k-opt neighborhood, we confirmed that local searchs using both neighborhoods reached locally optimal solution with almost equal tour length.

Though cossovers for TSPs basically generate offsprings by assembling edges of parents into tours, it is regarded to be depth-first search in edges to one parent based on other parent. Let k to be the depth at the depth-first search in edges, in other words the exchange of edges for one parent is performed using the information of other parent with maximal number of edges exchanged being limited up to k. In the following, we call the former of parents a acceptor and the latter of parents a donor. This number k is upperbound of $(e_{new} + e_{less}) * N_{city}$ using the previous notations, where N_{city} is the number of city.

If the crossover can restrict neighborhood of one parent(acceptor) to effective search space using information of other parent(donor) with large number k, it may have advantages against representative neighborhoods for TSPs. To examine this, we performed the following experiments.

Let EAX-k to be the alteration of EAX in which the number of edges deferent between a acceptor and offsprings is restricted not to exceed a certain number s given in advance. If k is small enough, EAX-k would generate offsprings which resemble to acceptor. The EAX-$N_{city}/2$ corresponts to the normal EAX. EAX-k is regarded to restricted k-opt neighborhood of one parent, namely EAX-k define restricted k-opt neighborhood for one of parents(acceptor) with using the other parent(donor) information. We could not control the number k explicitly except for some bias to decrease k, we neglected the offsprings with k being larger than k. As the bias to control k, we selected a single *ABcycle* in the algorithm of EAX(step 2)(See Appendix).

To compare the EAX-k neighborhood with k-opt neighborhood and LK neighborhood, the following two methods were performed. The rat575 was used as an instance and 30 runs were performed for each methods.

- **Multi-start k-opt(or LK)**

(1) Perform the procedure (2)-(3) by 300 times.

(2) Generate initial solution candidate randomly. Set this as a current candidate.

(3) For the current candidate, search its neighbors using k-opt(or LK) neighborhood until improved one is found or all neighbors are evaluated. If improved one is found, then replace the current candidate with it and repeat this procedure (3), else return the current candidate and finish this procedure (3).

(4) Return the best solution candidate obtained so far.

- **GA(EAX-k)**

(1) Set t=0. Generate 300 solution candidates using 2-opt local search. Let these $\Omega(t)$.

(2) Make 150 pairs of parents by random sampling from $\Omega(t)$ without replacement.

(3) For each parents, generate 20 offspring using EAX-k on condition that one is acceptor and the other is donor. If improved one is found, then put it to $\Omega(t+1)$, else put the acceptor to $\Omega(t+1)$. The same procedure is performed with roles of acceptor and donor being exchanged.

(4) If no improvement of shortest tour length in population over 40 generation, then stop, else let $t = t+1$ and go to (2).

The GA(EAX-k) is regarded as an alternation of multi-start k-pot using restricted k-opt neighborhood. Namely, each solution candidate(acceptor) in population moves to other improved neighbor in its k-opt neighborhood restricted by its mate(donor). Because the EAX-k neighborhood is completely included in the k-opt neighborhood, GA(EAX-k) could not get better solution candidate in a quality than the multi-start k-opt on condition that the population size of GA(EAX-k) is equal to the number of iteration in a multi-start k-opt. But being compared with multi-start k-opt, GA(EAX-k) could reduce the number of evaluation until locally optimal solutions of each method is obtained.

Table 2 shows the results of the multi-start k-opt varying k from 2 to 9. Table 3 shows the results of the GA(EAX-k) varying k from 2 to 30. In the table, *Best* and *Average* are the same term at Table 1, and *No.ofevaluations* denotes the number of evaluation of neighbors required for one run.

It is natural that the multi-start k-opt is superior to the GA(EAX-k) in a quality at the same number k. However, GA(EAX-k) reduces the number of evaluations drastically to obtain almost equal solution quality obtained by multi-start k-opt search. But the number of evaluation may not be a suitable measure because the computational efforts par one evaluation requires different CPU time for each neighborhoods. Then,

Table 2: The performance of Multi-Start k-opt(or LK)

rat575(optimum:6773)				
	Best	Average	No. of evaluations	time
2-opt	6992(3.23%)	7034.5(3.86%)	1.71×10^{7}	14
3-opt	6835(0.91%)	6852.4(1.17%)	3.24×10^{7}	297
4-opt	6788(0.22%)	6813.9(0.60%)	2.81×10^{8}	630
5-opt	6786(0.19%)	6799.0(0.39%)	9.20×10^{8}	3451
6-opt	6786(0.19%)	6793.2(0.30%)	3.23×10^{9}	6102
7-opt	6777(0.06%)	6785.2(0.18%)	1.13×10^{10}	24690
8-opt	6775(0.03%)	6781.3(0.12%)	4.29×10^{10}	92524
9-opt	6776(0.04%)	6779.6(0.10%)	2.22×10^{11}	518292
LK	6794(0.31%)	6810.8(0.55%)	8.17×10^{6}	119

Table 3: The performance of GA(EAX-k)

rat575(optimum:6773)								
k	Best	Average	No. of evaluations	**k**	Best	Average	No. of evaluations
1				**12**	6774[0]	0.046%	3.14×10^{5}
2	6992[0]	3.863%	2.97×10^{4}	**14**	6774[0]	0.022%	2.90×10^{5}
3	6915[0]	2.720%	1.24×10^{5}	**16**	6773[6]	0.012%	2.82×10^{5}
4	6848[0]	1.614%	1.56×10^{5}	**18**	6773[9]	0.010%	2.79×10^{5}
5	6825[0]	1.170%	2.71×10^{5}	**20**	6773[15]	0.007%	2.68×10^{5}
6	6810[0]	0.802%	2.87×10^{5}	**22**	6773[21]	0.004%	2.57×10^{5}
7	6788[0]	0.472%	3.29×10^{5}	**24**	6773[18]	0.006%	2.64×10^{5}
8	6777[0]	0.282%	3.54×10^{5}	**26**	6773[22]	0.004%	2.61×10^{5}
9	6779[0]	0.200%	3.28×10^{5}	**28**	6773[21]	0.004%	2.53×10^{5}
10	6775[0]	0.122%	3.27×10^{5}	**30**	6773[25]	0.002%	2.62×10^{5}

Table2 include the column *time* that denotes averaged CPU time required for a single run. GA(EAX-k) required 326 seconds on average for a single run that is almost independent of the number k. These runs were implemented by GNU C on Soralis2.5.1 with PC of Pentium Processor 200MHz.

The EAX-k restrict neighborhood of one of parent to effective range with using the other parent information in the exhaustive k-opt neighborhood, which realize large number k that is impractical for exhaustive k-opt neighborhood. Moreover, EAX restrict search space more effective than the heuristic depth-first search rules such as LK neighborhood. Therefore, GA using EAX shows good performance compared with multi-start local searchs using representative neighborhood.

Table 3 shows that the frequency of optimal solution obtained by GA(EAX-k) increases monotonously as increasing k until k is equal to about 22. Since the instance used here, rat575 has several semi-optimal solutions of length 6774 whose number of different edges from optimum are 4, 15, 16, 24, and 25, the above tendency is natural. This tendency suggests that the number k larger than a certain number, about 22 in this instance, is not essentially important for the ability of EAX. Moreover, the GA(EAX-k) with suitable k obtain the optimal solution with higher probability than the GA using normal EAX experimented in 2.2. This tendency was usually observed in other instances. We suppose that GA(EAX-k) is superior to GA using normal EAX in maintaining the diversity of the population because GA(EAX-k) leaves two offsprings from a pair of parents in which one is similar to one parent and the other is similar to the other parent. The suitable number k will due to instances, but we don't know suitable number k in advance. In our other experiments, we used the alternation of EAX to realize approximately suitable k that select a single *ABcycle* in the algorithm of EAX(step 2)(See Appendix).

4 Conclusion

In this paper, we discussed why GAs with using EAX realize good performance for TSPs from two viewpoints.

first, several crossovers for TSPs were compared each other, and we indicated that the appropriate trade-off between the ratio of edges inherited to offsprings from parents and the variety of offsprings is important. And we confirmed that EAX realized the appropriate trade-off and then be able to generate relatively large number of improved offsprings from a pair of parents.

Next, some advantages of EAX against representative neighborhood operators for TSPs were examined. We conclude that the EAX restrict neighborhood of one of parent to effective range based on the other parent information, which give more effective bias to search space than some heuristics given in advance.

REFERENCES

[1] Y. Nagata and S.Kobayashi, Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem, Proc. of the 7th Int. Conf. on Genetic Algorithms, pp.450-457, 1997.

[2] G. Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications, Vol.840 of Lecture Notes in Computer Science, Springer-Verlag, 1994.

[3] S. Lin and B. Kernighan, An Effective Heuristic Algorithms for the Traveling Salesman Problem, Operations Research, Vol.21, pp.498-516, 1973.

[4] 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. Conf. on Evolutionary Computation, pp. 529-534, 1996.

[5] D. Whitley, T. Starkweather and D. Fuquay, Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator, Proc. of the 3ed Int. Conf. on Genetic Algorithms, pp.133-140, 1989.

[6] TSPLIB95, http://www.iwr.uni-heidelberg.de/iwr/compt/soft/TSPLIB95/TSPLIB.html.

A Appendix

[Notations]

- $G = (V, E, w)$: a weighted complete graph corresponding to an instance.
- E_A, E_B: a set of edges included in parents respectively. (See Figure 3)
- $G_{AB} = (V, E_A \cup E_B, w)$. (See Figure 3)
- *ABcycle*: a closed loop on G_{AB},
  $C = \{e_{l_1}, e_{l_2}, \ldots, e_{l_{2n}}\}$ which satisfies
  $\forall i = (1, 2, \ldots, n)$,
  $e_{l_{2i-1}} \in E_A \quad e_{l_{2i}} \in E_B$,
  e_{l_i} and $e_{l_{i+1}}$ are adjacent to each other.
  (See Figure 4)

[The algorithm of EAX]

(1) Generate *ABcycle* from G_{AB} by **procedure** [Generate *ABcycle*]. Let C_i to be *ABcycle* generated.

(2) Select *ABcycle* randomly with probability 0.5. Let D to be a union of selected *ABcycles*.

(3) Generate intermediate solution by
$E_A \rightarrow (E_A \cap \bar{D}) \cup (E_B \cap D)$,
(For example, selecting *ABcycle* shown Fig. 4(c) as D and then resulting intermediate solution is shown in Fig. 5(b)).

(4) Modify intermediate solution to valid tour by **procedure** [Modification]. (For example, intermediate solutions shown in Fig. 5(*) is modified into valid tours shown in Fig. 6(*) respectively.)

procedure [Genetate *ABcycle*]
  $i := 0$; $s := 0$;
  $R_A := E_A$; $R_B := E_B$;
  repeat
    select $v_e \in V$ s.t. $\delta_A(v_e) \neq 0$ randomly.
    repeat
      if $P[s] \in E_B$ or $s = 0$ **then**
        select $e \in R_A$ incident to v_e randomly ;
        $R_A := R_A - \{e\}$;
      else
        select $e \in R_B$ incident to v_e randomly ;
        $R_B := R_B - \{e\}$;
      end if
      Let v' to be a vertex of e which is not v_e.
      $v_e := v'$
      $s := s + 1$; $P[s] := e$;
      if P includes *ABcycle* **then**
        $i := i + 1$; Put the *ABcycle* to C_i;
        output(C_i);, Delete C_i from P ;
        $s := s - |C_i|$;
      end if
    until $s = 0$
  until $\forall v \in V$, $\delta_A(v) = 0$
end procedure
where, $\delta_A(v)$ denotes the number of edges incident to vertex included in R_A.

procedure [Modification]
  Let k to be the number of subtours in a intermediate solution;,$s := k$;
  Let $U_i (i = 1, 2, \ldots, k)$ to be each subtours;
  repeat
    $i^* := \arg \min_{i=(1,2,\ldots,s)} |U_i|$;
    $j^* := \arg \min_{j(\neq i^*)} w_{i^*j}$;
    $\{e, e'\} := \arg \min_{e \in U_{i^*}, e' \in U_{j^*}} w_{i^*j^*}$;
    $U_{i^*} := (U_{i^*} \cup U_{j^*} - \{e, e'\}) \cup \{e'', e'''\}$;
    $U_{j^*} := U_s$;
    $s := s - 1$;
  until $s = 1$
  output(U_1);
end procedure
where, $|U_i|$ is the number of edges included in U_i,
$w_{ij} = \min_{e \in U_i, e' \in U_j} -w(e) - w(e') + w(e'') + w(e''')$,
For $(e = uv, \quad e' = u'v')$, $(e'', \quad e''')$ are defined by
$(e'' = uv', \quad e''' = u'v)$ or $(e'' = u'v, \quad e''' = uv')$

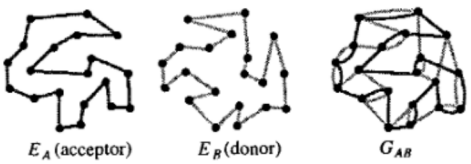

Figure 3: A pair of parent tours.

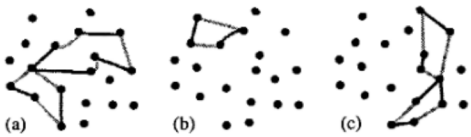

Figure 4: Examples of *ABcycle*.

Figure 5: Examples of intermediate solution. A intermediate solution consists of several subtours.

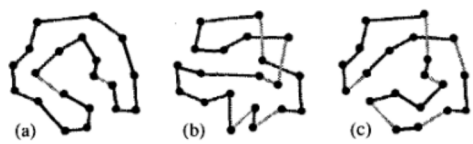

Figure 6: Examples of offsprings modified from above intermediate solution