2. Approach

2.1 Motivations

When the crossover is used as a main engine of search, it is required that it may generate children superior to both of parents with high probability. To realize this, in combinatorial optimization problems, it is important to inherit good substructures of parents to their children properly. Several heuristic crossovers have been proposed for the purpose of good inheritance.

However, most of the previous crossovers have been focussing in inheriting substructures of parents to children simply as much as possible ,under the premise that constraints generic to problems should be satisfied. For example in the TSP, edges or sub-paths are chosen naturally as substructures. Crossovers to inherit these substructures as much as possible have been proposed [3][4][5][6]. However, in general, children may include new elements that do not exist on their parents to satisfy the constraints. In most of previous crossovers, such new elements are generated artificially only to satisfy the constraints, but they are inappropriate for the purpose of the optimization.

For example, in Edge Recombination (ER) [4] for STSP, children inherit 95~99% edges from parents tours. However, 1~5% new edges are generated and attached to children to satisfy the constraint. New edges scarcely contribute to the improvement of tour length especially in a case where parents exist near the optimum.

Most of existing crossovers have the above drawbacks. The EXX [11] was proposed to suppress this drawback. In the EXX, valid tours are generated by only edges of parents. However, the EXX has a problem that it can generate only small variety of children from parents, because of the severe condition of the EXX that a new tour must be generated by only edges of parents under the constraints of STSP.

A child generated by the EXX become improvement of parents with relatively high probability, but EXX can not generate enough number of children. On the other hand, the ER can generate enough number of children, but each child become improvement of parents with relatively low probability.

Considering drawbacks of the existing crossovers, the following crossover is presented. We explain the crossover for STSP and ATSP respectively.

1. Intermediate individuals are generated by only edges of parents so that a relaxed constraint of the TSP is satisfied.
2. The intermediate individuals are modified so that the original constraints of the TSP are satisfied.

Here, the relaxed constraints of both STSP and ATSP are the assignment relaxation as follows:

(STSP) Each degree of all vertices is two respectively. Intermediate individuals satisfying this constraint are such invalid tours as shown in Fig.11.

(ATSP) Each indegree and outdegree of all vertices are one respectively. Intermediate individuals are such invalid tours as shown in Fig.6.

These intermediate individuals can be easily modified into valid tours by a method described in section 2.3. This modification generally does not cause a change for the worse , because it can be done for the purpose of improvement, not randomly, but systematically. Moreover, relaxing constraints of the TSP at the phase of utilizing parents information, a number of children can be generated from a pair of parents.

Therefore this crossover enable parents to generate enough number of children, and each generated child becomes improvement of parents with high probability.

2.2 Generation of Intermediate Individuals

The method of generating intermediate individuals for ATSP is more elegant than that for STSP. We first explain its summary for ATSP, next for STSP.

2.2.1 In the case of ATSP

In the proposed crossover, intermediate individuals are generated by assembling edges of two parent tours. Its summary is described in the following. First, consider tour-A and tour-B as parents (Fig.1). We can generate intermediate individuals by removing a several number of edges from tour-A and adding the same number of edges of tour-B to it without loosing the generality. In this method intermediate individuals are generated if and only if

(1) when an directed edge (a,b) of tour-A is removed from tour-A, an edge of tour-B whose initial vertex is vertex a is added to it, and

(2) when an directed edge (c,d) of tour-B is added to tour-A, an edge of tour-A whose terminal vertex is vertex d is removed from it.

Let G' a directed graph obtained by overlapping tour-A and tour-B, which is shown in Fig.2. To explain a set of edges that consists of edges of tour-A removed from tour-A and edges of tour-B added to it so that the above conditions (1) and (2) are satisfied, we define the following term "*AB-cycle*".

[Def.] *AB-cycle*

A cycle generated by tracing edges of tour-A from the initial vertex to the terminal vertex and edges of tour-B from the terminal vertex to the initial vertex alternatively on graph G' is defined as *AB-cycle*.

AB-cycles generated on G' (Fig.2) are shown in Fig.4, where the edges on G' are divided into eight *AB-cycles*.

[Theorem 1]

All edges on the graph G' are divided into *AB-cycles* uniquely.

Fig.3 shows this division into *AB-cycles* on G', in which the *AB-cycles* are those shown in Fig.4 respectively.

The *AB-cycle* is a minimum unit of the set of edges that satisfies the above condition (1) and (2). By removing edges of tour-A included in the *AB-cycle* from tour-A and adding the edges of tour-B included in the *AB-cycle* to it, the intermediate individual is generated. We call this procedure the application of the *AB-cycle* to the tour-A. If a *AB-cycle* that consists of two same directed edges as shown in Fig.4 (④~⑧) is applied to the tour-A, this application is not change the tour-A. Therefor we call this kind of *AB-cycle* an ineffective *AB-cycle*. The *AB-cycle* that is not ineffective is called an effective *AB-cycle*. Applying some arbitrary *AB-cycles* to tour-A, the intermediate individual can be generated too. In general, the sets of edges that satisfy above conditions (1) and (2) can be constructed of some arbitrary *AB-cycles*. We call such sets of edges Exchangeable Edge Sets (*E-sets*). For example, by applying each *E-set* shown in Fig.5 (a)~(c) to tour-A, intermediate individuals are generated as shown in Fig.6 (a)~(c) respectively.

If edges on G' are divided into k effective *AB-cycles*, 2^k *E-sets* can be constructed and 2^k intermediate individuals can be generated too.

[Theorem 2]

The intermediate individuals constructed by assembling edges of tour-A and tour-B correspond to the ways of choosing effective *AB-cycles* one to one.

Constructing the *E-set* by choosing effective *AB-cycles* divided on G' and applying it to tour-A, intermediate individuals that is constructed of only edges of tour-A and tour-B can be generated.

2.2.2 In the case of STSP

The generation of intermediate individuals for STSP is similar to that for ATSP. In this case intermediate individuals are generated if and only if

(1) when an undirected edge (a,b) of tour-A is removed from tour-A, undirected edges of tour-B that are incident to vertex a and b are added to it respectively, and

(2) when an undirected edge (c,d) of tour-B is added to tour-A, undirected edges of tour-A that are incident to vertex c and d are removed from it respectively.

As well as the case of ATSP, Fig.7 shows tour-A and tour-B as parents, and Fig.8 shows an undirected graph G'. An *AB-cycle* for STSP is defined as follows.

[Def.] ***AB-cycle***

A cycle generated by tracing edges of tour-A and edges of tour-B alternatively on graph G' is defined as *AB-cycle*.

[Theorem 3]

All edges on G' are divided into *AB-cycles*. Though this division into *AB-cycles* is not determined uniquely, there is no way of division that exist some edges that dose not belong to any *AB-cycle*.

Fig.9 shows examples of this division on G'.

2.3 Modification operation

The intermediate individual is modified into a valid tour by merging its subtours. This procedure is practically the same between ATSP and STSP.

Two subtours are merged by removing one edge from each subtour respectively and reconnecting it to obtain one subtour. Consider an intermediate individual that consists of k subtours. Let U_i be the i-th subtour and we define

$$w_{ij} = \min\{-w(v_1,v_2)-w(v_3,v_4)+w(v_1,v_4)+w(v_3,v_2) \mid (v_1,v_2)\in U_i,\ (v_3,v_4)\in U_j\}.$$

The value w_{ij} is the cost required to merge two subtours U_i and U_j. In the case of STSP, not only an edge (v_3,v_4) but also an edge (v_4,v_3) must be considered in the above formula. By finding the minimum spanning tree on a graph in which a weight between each pair among k vertices is defined as w_{ij}, we can modify an intermediate individual into a valid tour effectively. Especially, when the weights of edges are defined by the Euclidean distance, this modification operation is very effective. Fig.12 shows a concept of this modification for ATSP.

2.4 Outline of EAX

The generation of a new tour from tour-A and tour-B by the above method is performed as follows.

(1) Generate *AB-cycles* from parents (tour-A and tour-B). (Fig.3 or Fig.9)

(2) Construct a *E-set* by choosing *AB-cycles* on the basis of some criterion. (Fig.5 or Fig.10)

(3) Generate an intermediate individual by applying the *E-set* to tour-A. (Fig.6 or Fig.11).

(4) Modify the intermediate individual into a valid one by modification. (Fig.12)

The above procedure is called generation of children by the Edge Assembly Crossover (EAX).

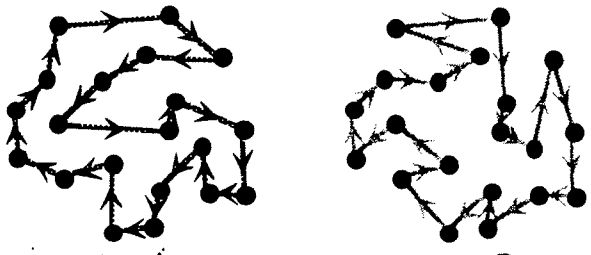
3. The EAX algorithm

In this section, details and features of the EAX are described.

3.1. The algorithms of generating *AB-cycles*

3.1.1 In the case of ATSP

In the case of ATSP, considering the definition of *AB-cycle* and [Theorem 1], the algorithm of dividing directed edges on graph G' into *AB-cycles* is easy. It can be performed as follows.



tour-A tour-B

Fig.1: Two parents tour for ATSP.

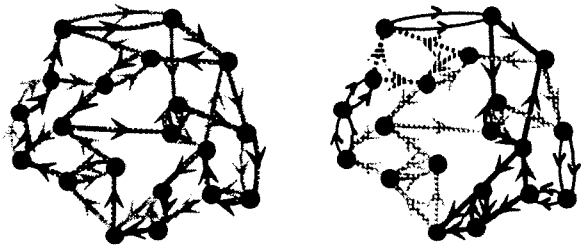


Fig.2: A directed graph G'.

Fig.3: A division of directed edges into *AB-cycles* on G'.

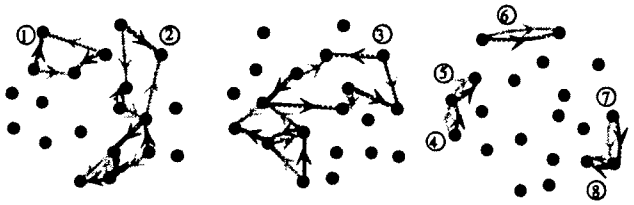
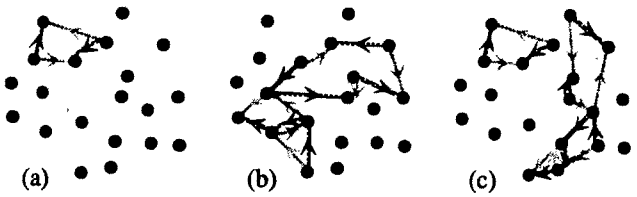
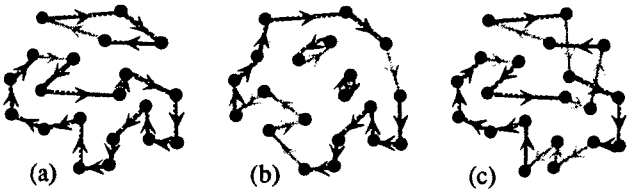


Fig.4: *AB-cycles* generated on G'.



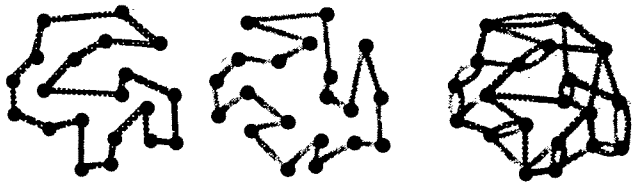
(a) (b) (c)

Fig.5: Examples of *E-set* generated on G'.



(a) (b) (c)

Fig.6: Intermediate individuals obtained by applying *E-sets* shown in Fig.5 to tour-A respectively.



tour-A tour-B

Fig.7: Two patents tour for STSP.

Fig.8: An undirected graph G'.

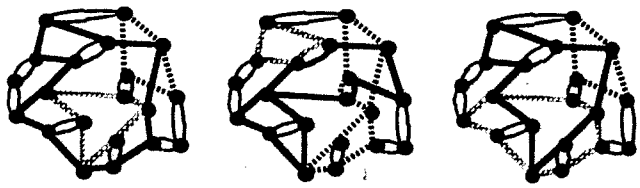
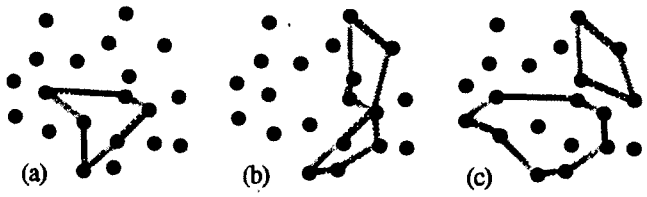
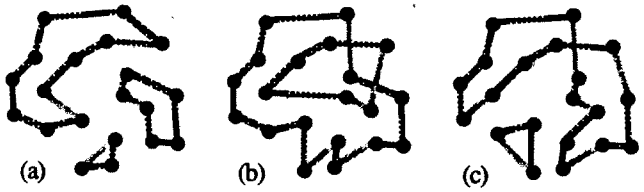


Fig.9: Examples of the division of undirected edges into *AB-cycles* on G'. Ineffective *AB-cycles* are shown by thin lines.



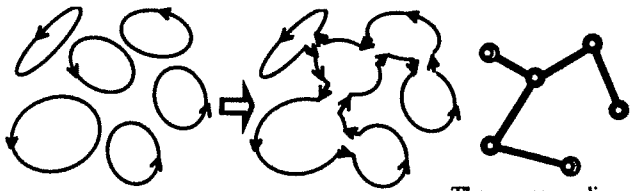
(a) (b) (c)

Fig.10: Examples of *E-set* generated on G'.



(a) (b) (c)

Fig.11: Intermediate individuals obtained by applying *E-sets* of Fig.10 to tour-A respectively.



An intermediate individual. A valid tour The corresponding minimum spanning tree.

Fig.12: The modification of an intermediate individual into a valid tour and the corresponding minimum spanning tree.

[Algorithm: Generating *AB-cycles* (ATSP)]

(1) Let R be a graph that consists of the directed edges of both tour-A and tour-B.

(2) Select an edge of tour-A randomly on R and perform a picture drawn with a single stroke of the brush on R by tracing edges of tour-A from the initial vertex to the terminal vertex and edges of tour-B from the terminal vertex to the initial vertex alternatively. This tracing is determined uniquely. As the result, a *AB-cycle* is necessarily generated. Go to (3).

(3) Record the *AB-cycle* generated, and remove the edges contained in this *AB-cycle* from R. If R has no edge, then terminate this procedure, else go to (2).

3.1.2 In the case of STSP

Since the division of undirected edges on G' into *AB-cycles* is not determined uniquely in STSP (see Theorem 3), several methods of the division are conceivable. In this paper we use a simple method described as follows.

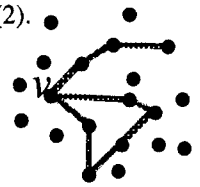
[Algorithm: Generating *AB-cycles* (STSP)]

(1) Let R be a graph that consists of the undirected edges of both tour-A and tour-B.

(2) Select an edge (**) randomly on R and perform a picture drawn with a single stroke of the brush on R by tracing edges of tour-A and edges of tour-B alternatively. If there exist two candidates that can be traced next, select one of the edges randomly. If a *AB-cycle* is included on the way (see below figure), go to (3).

(3) Record the *AB-cycle* included, and remove the edges contained in this *AB-cycle* from R. If R has no edge, then terminate this procedure, else go to (2).

(**)If the *AB-cycle* generated just before is a case shown in right figure, an edge that is incident to vertex v is selected.



3.2 Choosing *AB-cycles* for constructing a *E-set*

Constructing a *E-set* by choosing several effective *AB-cycles* and applying it to tour-A, an intermediate individual is generated. Effective *AB-cycles* can be chosen independently for constructing *E-sets*. Therefore, if the edges on G' is divided into k effective *AB-cycles*, 2^k intermediate individuals can be generated respectively.

The following two methods for choosing effective AB-cycles are proposed.

[Method 1: Random Selection]

Each effective *AB-cycles* is chosen randomly with probability 0.5. The EAX that adopts this method is called EAX(rand).

[Method 2: Heuristic Selection]

We propose a heuristic method that considers a balance between exploitation and exploration.

We use the following notations.

tour-A: one parent tour with shorter tour length.

tour-B: the other parent tour with longer tour length.

E_A: an set of edges that constructs tour-A.

E_B: an set of edges that constructs tour-B.

AB_cycle_i: i-th effective *AB-cycle*(i=1,2,...,k).

n_i: the number of edges on $cycle_i$.

e: an edge.

$w(e)$: a weight of e.

$f(e)$: a ratio that e exists among the population. At the beginning of each generation, for all edges(${}_nC_2$ in n-city STSP or ${}_nP_2$ in ATSP), we calculate the ratio that each edge exists among the population. For example, $f(e)$=0.4 means that 40% individuals have the edge e among the population.

By using the above notations, the following is calculated.

$$gain_i = \sum_{e \in AB\_cycle_i \cap E_A} w(e) - \sum_{e \in AB\_cycle_i \cap E_B} w(e) \quad (1)$$

$$f_{A_i} = \sum_{e \in AB\_cycle_i \cap E_A} f(e), \quad f_{B_i} = \sum_{e \in AB\_cycle_i \cap E_B} f(e) \quad (2)$$

$$\overline{f_A} = \frac{\sum_i \sum_{e \in AB\_cycle_i \cap E_A} f(e)}{\sum_i n_i / 2}, \quad \overline{f_B} = \frac{\sum_i \sum_{e \in AB\_cycle_i \cap E_B} f(e)}{\sum_i n_i / 2} \quad (3)$$

$$div_i = (f_{A_i} - \overline{f_A} \times n_i / 2) - (f_{B_i} - \overline{f_B} \times n_i / 2) \quad (4)$$

$$sum\_gain\_plus = \sum_{i \ s.t. \ gain_i \geq 0} gain_i \quad (5)$$

$$sum\_div\_plus = \sum_{i \ s.t. \ div_i \geq 0} div_i \quad (6)$$

$$GAIN_i = \frac{gain_i}{sum\_gain\_plus} \quad (7)$$

$$DIV_i = \frac{div_i}{sum\_div\_plus} \quad (8)$$

$$F_i = GAIN_i + \alpha \times DIV_i \quad (9)$$

The value of $gain_i$ by Eq.(1) represents a gain of the tour length from tour-A to an intermediate individual obtained by applying AB_cycle_i to tour-A. The value of div_i is one of measures for representing contribution of AB_cycle_i to the diversity among the population. Applying AB_cycle_i to tour-A, if edges removed from tour-A relatively more exist among the population or edges added to tour-A relatively less, div_i takes a positive value. In a case of the reverse, it takes a negative value. Therefore, if a AB_cycle_i with a positive $gain_i$ is chosen, a gain from tour-A to an intermediate individual is obtained. And if a AB_cycle_i with a positive div_i is chosen, the diversity among the population is maintained. The above two criteria do not consider the effects of the modification from intermediate individuals to valid tours. These criteria are normalized by Eq.(7) and (8). F_i by Eq.(9) is an criterion that represents the trade-off between the normalized gain $GAIN_i$ and the normalized diversity DIV_i. By choosing all AB_cycle_i with positive F_i, a *E-set* is constructed. We can change the trade-off between the gain and the diversity, or exploitation and exploration by adjusting the parameter α. This method is called EAX(α=・). We use EAX(α=1) in the following experiments.

3.3. The Algorithm of Modification.

The role of the modification is to merge subtours of the intermediate individual into the valid tour.

[Algorithm: Modification]

(1) Set an intermediate individual to a current intermediate individual T.

(2) Let U_i(i=1,2,...) be subtours constructing T. Choose a subtour U_r whose number of edges is the fewest in the subtours U_i(i=1,2,...).

(3) Find a pair of edges (v_1,v_2) and (v_3,v_4) so that it maximizes

$w(v_1,v_2)+w(v_3,v_4)-w(v_1,v_4)-w(v_3,v_2)$,

$((v_1,v_2)\in U_r, (v_3,v_4)\in U_i (i\neq r))$.

Let U_s be the subtour that includes the edge (v_3,v_4) determined by the above formula. Remove the edges (v_1,v_2) and (v_3,v_4) from T, and add new edges (v_1,v_4) and (v_3,v_2) to it. As a result, U_r and U_s are merged into one subtour.

(4) If T consists of one subtour, then the procedure is terminated, else T is updated and go to (1).

Here, at step (3), for each edge (v_1,v_2), it is sufficient to search an edge (v_3,v_4) "near" to (v_1,v_2). In practice, for each edge (v_1,v_2), as candidates of edge (v_3,v_4) we investigate only edges incident to vertices within i-th near from v_1 and v_2 respectively. When weights w are given by the Euclidean distance, the value of i may be sufficiently small.