Edge histogram based sampling with local search for solving permutation problems

Shigeyoshi Tsutsui[a,*], Martin Pelikan[b] and Ashish Ghosh[c]
[a]*Department of Management Information, Hannan University, 5-4-33, Amamihigashi, Matsubara Osaka 580-8502 Japan*
[b]*Department of Math and Computer Science, University of Missouri at St. Louis, 8001 Natural Bridge Rd., St. Louis, MO 63121, USA*
[c]*Machine Intelligence Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata 700 108, India*

Abstract. A basic scheme for solving permutation problems in the framework of probabilistic model-building genetic algorithms (PMBGAs) that uses edge histogram based sampling techniques was reported in [23]. Two sampling algorithms – sampling without template, and the sampling with template were presented. In this paper, we combine local search heuristics with those sampling algorithms to solve the traveling salesman problem (TSP). We tested two types of heuristics; one is a simple heuristic called 2-OPT, and the other is a sophisticated Lin-Kernighan heuristic. The results show that edge histogram based sampling with these heuristics improve the performance significantly, and can solve large problems having thousands of cities fairly well. The algorithm is thus seen to be scalable.

Keywords: Probabilistic model-building genetic algorithms, edge histogram based sampling algorithm, 2-OPT, lin-kernighan heuristic, traveling salesman problem

1. Introduction

GAs should work well for problems that can be decomposed into sub-problems of bounded difficulty [6] and for problems where similar solutions are of similar quality. However, fixed, problem-independent variation operators are often incapable of effective exploitation of the selected population of high-quality solutions and the search for the optimum often becomes intractable [6,18,22]. One of the most promising research directions that focus on eliminating this drawback of fixed, problem-independent variation operators, is to look at the generation of new candidate solutions as a learning problem, and use a probabilistic model of selected solutions to generate the new ones [12,19]. The probabilistic model is expected to reflect the problem structure and, as a result, this approach might provide more effective exploitation of promising solutions than recombination and mutation operators in traditional GAs. The algorithms based on learning and sampling a probabilistic model of promising solutions to generate new candidate solutions are called *probabilistic model-building genetic algorithms* (*PMBGAs*) [19], *estimation of distribution algorithms* (*EDAs*) [15], or *iterated density estimation algorithms* (*IDEAs*) [1].

Most of the work based on this approach focuses on optimization problems where candidate solutions are represented by fixed-length vectors of discrete or continuous variables. However, for many combinatorial problems, permutations provide a much more natural representation for candidate solutions. Despite the great success of PMBGAs in the domain of fixed-length discrete and continuous vectors, few studies are found for permutation and scheduling problems [2,3, 20]. More importantly, these studies take an indirect approach of mapping the permutation problems to fixed-length vectors of discrete or continuous variables,

*Corresponding author. Tel.: +81 72 332 1224; Fax: +81 72 336 2633; E-mail: tsutsui@hannan-u.ac.jp.

which in some cases necessitates the use of repair operators to correct invalid solutions.

Tsutsui [23] introduced a promising approach to use PMBGAs for permutation problems using *edge histogram based sampling*, and showed competitive results on several benchmark instances of the traveling salesman problem (TSP). The approach is named *edge histogram based sampling algorithms* (EHBSAs). Although, the EHBSA was initially tested for solving the TSP, the approach can solve any problem that can be formulated within the domain of fixed-length permutations. As an example, the flow shop scheduling problem was solved by Tsutsui & Miki [25,26].

In this paper, EHBSA has been hybridized with some popular well-known local search heuristics to take advantage of local search. We tested two types of heuristics; one is a simple heuristic for solving TSP, called 2-OPT; and the other is a sophisticated one, known as Lin-Kernighan heuristic [8,13]. The results show that EHBSA added with these heuristics can solve TSPs even with thousand of cities fairly well. Experimental evidences show that this hybrid technique decreases the computational requirements for finding a globally optimal tour in TSP, providing a method capable of solving significantly larger problems of thousands of cities. The method is very robust in the sense of detecting global optima for a wide variety of population sizes.

On the other hand, a combination of the local heuristics with existing recombination based techniques (like OX, PMX, EER) also show advantage, but not so reliable and robust. This is due to the fact that they can detect the global optimum mostly for a bigger population size.

The rest of this article is organized as follows. Section 2 gives a brief review of EHBSA along with experimental results. In Section 3 we describe the proposed algorithms (i.e., EHBSA combined with local search) along with experimental results on a few benchmark problems. Finally, Section 4 summarizes and concludes the report.

2. Edge histogram based sampling algorithms: a brief overview

This section briefs the edge histogram based sampling algorithms (EHBSAs), within the PMBGA framework, and its use for (i) modeling promising solutions, and (ii) generating new solutions by simulating the learned model.

$$\begin{aligned} s^t_0 &= (0,\ 1,\ 2,\ 3,\ 4)\\ s^t_1 &= (1,\ 3,\ 4,\ 2,\ 0)\\ s^t_2 &= (3,\ 4,\ 2,\ 1,\ 0)\\ s^t_3 &= (4,\ 0,\ 3,\ 1,\ 2)\\ s^t_4 &= (2,\ 1,\ 3,\ 4,\ 0) \end{aligned}$$

(a) Population $P(t)$

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4\\ \hline 0 & 0 & 3.1 & 2.1 & 2.1 & 3.1\\ 1 & 3.1 & 0 & 4.1 & 3.1 & 0.1\\ 2 & 2.1 & 4.1 & 0 & 1.1 & 3.1\\ 3 & 2.1 & 3.1 & 1.1 & 0 & 4.1\\ 4 & 3.1 & 0.1 & 3.1 & 4.1 & 0 \end{array}$$

(b) EHM^t of $P(t)$

Fig. 1. An example of symmetric edge histogram matrix for $N = 5$, $L = 5$, $B_{\mathrm{ration}} = 0.04$.

A set of edges that generates a path in a graph (including all nodes) can be used as an alternative representation of a permutation of the nodes in the graph. The first element of the path will determine the first element of the permutation, while the remaining elements of the permutation can be obtained by tracing the remainder of the path until the last element of the path is reached. The basic idea of EHBSA is to collect and exploit information about the presence of edges (frequency of edges) in the entire population of selected high-quality solutions.

2.1. Edge histogram matrix

Let the permutation represented by the kth individual in a population $P(t)$ of size N, at generation t, be denoted by $s^t_k = (\pi^t_k(0), \pi^t_k(1), \ldots, \pi^t_k(L{-}1))$, where $(\pi^t_k(0), \pi^t_k(1), \ldots, \pi^t_k(L{-}1))$ define a permutation of $(0, 1, \ldots, L{-}1)$, and L is the length of the permutation. The edge histogram matrix EHM^t $(e^t_{i,j})$ $(i, j = 0, 1, \ldots, L{-}1)$ of the population is symmetrical and consists of L^2 elements. The (i, j)th element of the matrix is obtained by

$$e^t_{i,j} = \begin{cases} \sum_{k=1}^{N} (\delta_{i,j}(s^t_k) + \delta_{j,i}(s^t_k) + \varepsilon & \text{if } i \neq j\\ 0 & \text{if } i = j \end{cases}, \quad (1)$$

where $\delta_{ij}(s^t_k)$ is a delta function defined as

$$\delta_{i,j}(s^t_k) = \begin{cases} 1 \text{ if } \exists h\ [h \in \{0, 1, \cdots L-1\} \\ \quad \wedge \pi^t_k(h) = i \wedge \pi^t_k((h+1) \\ \quad \bmod L) = j] \\ 0 \text{ otherwise} \end{cases}, \quad (2)$$

and $\varepsilon(> 0)$ biases the sampling toward random permutations (which can also be visualized as a form of mutation). To have similar selection pressure for random permutations (for all problems and parameter settings), ε should be proportional to the expected value of $e^t_{i,j}$. Since the average value of $e^t_{i,j}$ for $i \neq j$ in EHM^t is $2LN/(L^2\text{-L}) = 2N/(L{-}1)$, we use

```
1. Set the position counter $p \leftarrow 0$.
2. Obtain first node $c[p]$ randomly from {0, 1,2,□ .L□1}
3. Construct a roulette wheel vector $rw[]$ from $EHM^t$ as $rw[j] \leftarrow e^t_{c[p],j}$ (j=0, 1, .., L□1)
4. Set 0 to previously sampled nodes present in $rw[]$ ($rw[c[i]] \leftarrow 0$ for $i$ =0, .., $p$).
5. Sample next node $c[p+1]$ with probability $rw[x]/\sum_{j=0}^{L-1} rw[j]$ using roulette wheel $rw[]$.
6. Update the position counter $p \leftarrow p+1$.
7. If $p<L$□1, go to Step 3 else stop.
```

Fig. 2. EHBSA/WO.

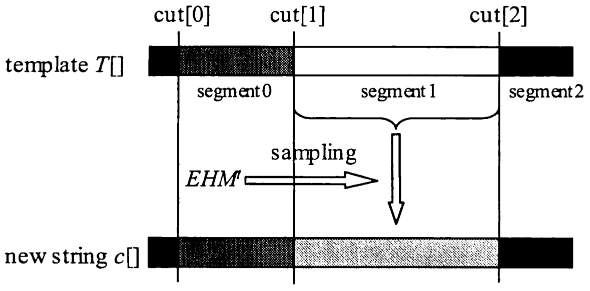

Fig. 3. An example of EHBSA/WT/3.

$$\varepsilon = \frac{2N}{L-1} B_{\text{ratio}} \tag{3}$$

where $B_{\text{ratio}}(> 0)$ or the bias ratio is a constant related to the pressure toward random permutations. A smaller value of B_{ratio} reflects the real distribution of edges in the parent population, whereas a larger value of B_{ratio} will allow infrequent addition of new edges. An example of EHM^t is shown in Fig. 1.

Although the edge histogram matrix defined above is *symmetric* (i.e., $e_{i,j} = e_{j,i}$) and it is, thus, applicable only to problems where the orientation of edges does not matter (e.g., symmetric TSP), an *asymmetric* edge histogram matrix can be defined for problems where the orientation of edges does matter (e.g., asymmetric TSP and flow shop scheduling) [25,26].

2.2. Sampling algorithms

We use two algorithms for sampling the edge histogram matrix EHM^t: (i) the edge histogram based sampling algorithm without template (EHBSA/WO), and (ii) the edge histogram based sampling algorithm with template (EHBSA/WT).

2.2.1. EHBSA/WO

Let us denote the elements of the permutation, to be sampled, by $c[i]$ for $i \in \{0, 1, \ldots, L-1\}$. EHBSA/WO starts by randomly selecting the initial element of it (denoted by $c[0]$). The sampling continues recursively using a variant of the roulette-wheel selection algorithm [5]. Let us assume that the last element generated is $c[i]$ (thus we have generated $i + 1$ elements so far). The new element $c[i + 1]$ is set to j (we restrict potential values of j so that $j \neq c[k]$ for all $k \in \{0, 1, \ldots, i\}$; otherwise, the new edge would create a cycle) with a probability proportional to the element $e^t_{c[i],j}$ of the given edge histogram matrix EHM^t. The sampling continues until the entire permutation has been generated. Figure 2 shows the schematic description of the edge histogram based sampling algorithm without template (EHBSA/WO).

2.2.2. EHBSA/WT

EHM^t gives a marginal edge histogram and has no graphical structure. EHBSA/WT is intended to make up for this disadvantage by using a template in sampling a new individual. In generating each new individual, a template individual is chosen from $P(t)$ (normally, randomly). Using n ($n > 1$) random cut points the template is divided into n segments. Then, we choose one segment randomly and sample nodes for that segment. Nodes in other segments remain unchanged.

We denote this sampling method as EHBSA/WT/n. Since the average length of one segment is L/m, EHBSA/WT/n generates new strings that are different in L/m nodes, on an average, from their templates. Figure 3 shows an example of EHBSA/WT/3. In this example, nodes of the new individual after cut[2] and before cut[1] are the same as those of the template. New nodes are sampled for only segment1 (from cut[1] up to, but not including, cut[2]) based on the EHM^t using a similar algorithm to that of EHBSA/WO. Figure 4 shows the schematic representation of the EHBSA/WT.

Randomness in generating cut points and choosing one of the segments introduces variation in segment positions and lengths of elements that are going to be generated. Let $f(x)$ be the probability density function of the length of a segment to be sampled in EHBSA/WT/n. Then, $f(x)$ can be obtained as follows [24]:

```
1.  Choose a template $T[]$ from $P(t)$.
2.  Obtain sorted cut points array cut[0], cut[1], □, c ut[$n$□1]randomly.
3.  Choose a cut point cut[$l$] by generating a random number $l \in [0, n$□$1]$
4.  Copy nodes in $T[]$ to $c[]$ after cut[($l$+1) mod $n$] and before cut[$l$].
5.  Set the position counter $p \leftarrow$ (cut[$l$]□1+$L$) mod $L$.
6.  Construct a roulette wheel vector $rw[]$ from $EHM^t$ as $rw[j] \leftarrow e^t_{c[p],j}$ ($j$=0, 1, .., $L$□1)
7.  Set 0 to previously sampled node in in $rw[]$ ($rw[c[i]] \leftarrow 0$ for $i$ = cut[($l$+1) mod $n$], □ ,
    $p$).
8.  Sample next node $c[(p+1) \bmod L]$ with probability $rw[x]/\Sigma_{j=0}^{L-1} rw[j]$ using roulette
    wheel $rw[]$.
9.  Update the position counter $p \leftarrow (p+1) \bmod L$.
10. If $(p+1) \bmod L \neq$ cut[($l$+1) mod $n$], go to Step 6 else stop.
```

Fig. 4. EHBSA/WT.

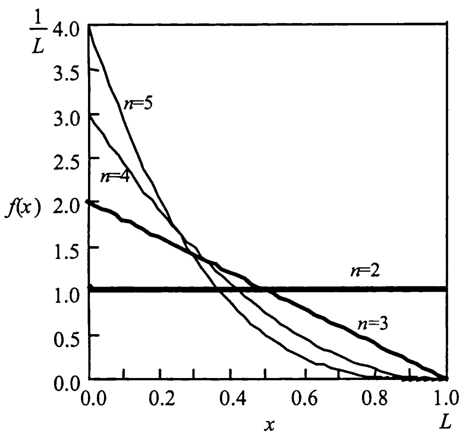

Fig. 5. Probability density function $f(x)$.

$$f(x) = \frac{(n-1)}{L}\left(1 - \frac{x}{L}\right)^{n-2}. \tag{4}$$

Figure 5 shows the probability density function $f(x)$. For $n = 2$, $f(x)$ is uniformly distributed in $[0, L]$. Thus, the length of a segment to be sampled in EHBSA/WT/2 is uniformly distributed in [0, L]. When the length of the segment is small, the EHBSA/WT/2 samples a small number of nodes; thus performing a kind of local search and improves the quality of the individual. On the other hand, when the length is large, the EHBSA/WT/2 samples a large number of nodes, performing a kind of global search. This can either improve the template individual or produce a new individual. Thus, we can expect EHBSA/WT/2 to work having a balancing of global and local improvements. For $n > 2$, short segments are more likely to occur.

2.3. Models of evolution

All evolutionary models are based on the steady-state scheme. Models used are as follows.

Model for EHBSA/WT: Let the population size be N, and let it, at time t, be denoted by $P(t)$. Note that individuals in the initial population $P(0)$ are generated randomly. The population $P(t+1)$ is produced as follows (see Fig. 6):

1. Edge histogram matrix EHM^t (see Section 2.1) is computed from $P(t)$.
2. A template individual $T[]$ is selected from $P(t)$ randomly.
3. EHBSA/WT (see Section 2.2.2) is executed using EHM^t and $T[]$ to generate a new individual $c[]$.
4. The new individual $c[]$ is evaluated. If $c[]$ is better than $T[]$, then $T[]$ is replaced by $c[]$, else do nothing.
5. $P(t+1)$ is generated performing steps 1–4, N times.

Model for EHBSA/WO: The evolutionary model for EHBSA/WO is basically the same as the model for EHBSA/WT, with the exception that EHBSA/WO uses a pseudo template $PT[]$.

Model for two-parent recombination operators: To compare the performance of the proposed methods with that of the traditional two-parent recombination operators, we used the same steady-state framework. Two parents are selected from $P(t)$ randomly and recombination operator is applied on them to produce an offspring. If the offspring is better than the worst parent, we incorporate it into the population. This scheme was previously used in the GENITOR algorithm [27].

Table 1
Results of Berlin52

Pop size		60				120				240			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
EHBSA	WO	0/20	-	-	0.0533	2/20	88071.0	8436.0	0.0315	4/20	287587.0	23920.9	0.0221
	WT/2	14/20	87270.3	40038.5	0.0008	15/20	170383.7	86103.6	0.0008	18/20	268762.6	45087.0	0.0003
	WT/3	16/20	79686.1	19976.6	0.0020	18/20	148990.4	29788.0	0.0003	20/20	275306.7	37282.2	0
	WT/4	20/20	102421.0	39764.6	0	20/20	179610.1	24327.0	0	20/20	325404.7	40214.8	0
	WT/5	20/20	153894.1	39368.5	0	20/20	240952.7	44367.0	0	17/20	389686.5	52160.4	0.0020
Pop size		240				480				960			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
OX		0	-	-	0.0889	2/20	138016.0	23839.0	0.0553	2/20	204510.5	11278.5	0.0558
PMX		0	-	-	0.5164	0/20	-	-	0.3698	0/20	-	-	0.2983
EER		0	-	-	0.0420	8/20	66100.0	8527.1	0.0034	18/20	119112.5	6897.5	0.0013

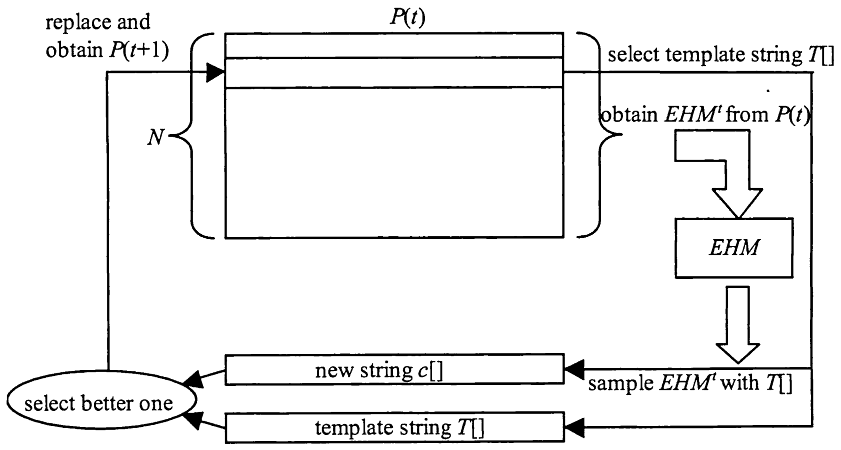
replace and obtain $P(t+1)$
$P(t)$
select template string $T[]$
obtain EHM^t from $P(t)$
N
EHM
new string $c[]$
sample EHM^t with $T[]$
select better one
template string $T[]$

Fig. 6. Evolutionary model for EHBSA/WT.

2.4. *Results and analysis*

Here we show results of 20 simulations for each TSP instance and each parameter setting. Each run was terminated either when the optimal tour was found, or when the population converged, or when the number of evaluations reached $E_{\max}$ (supplied by the user). We used the Berlin52 and pr76 TSP instances available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. Values of $E_{\max}$ were 500000 and 1000000 for Berlin52 and pr76, respectively. Population size 60, 120, 240 were used for EHBSA; for other algorithms, population size 240, 480 and 960 were used. The bias ratio B_{ratio} of Eq. (3) is set to 0.005 for all experiments.

We evaluated the performance of different algorithms by measuring the following quantities: *#OPT* (the number of runs in which the algorithm succeeded in finding the optimal tour), *MNE* (the mean number of evaluations to find the global optimum in those runs where it did find the optimum, and *Error* ((the average length of the best solution over the 20 runs – optimal tour length)/optimal tour length).

Results on Berlin52 are shown in Table 1. EHBSA/WO could not find the optimal tour with $N = 60$ and could find the optimal in 2 and 4 runs with $N = 120$ and 240, respectively. On the other hand, EHBSA/WT/n found the optimum tour more frequently and its *Error* values were much smaller than those of EHBSA/WO. Specially, EHBSA/WT/4, 5 with $N = 60$ found the optimal tour in all 20 runs and showed small values of MNEs. From other operators, Extended ER (EER) [21] showed the best performance. The EER with $N =$ 480 and 960 found the optimal tour 8 and 18 times, respectively, although the values are poorer than those of EHBSA/WT. OX [17] showed worse performance than EER. PMX [5] performed the worst.

Table 2
Results of pr76

Pop size		60				120				240			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
EHBSA	WO	0/20	-	-	0.0560	0/20	-	-	0.0288	0/20	-	-	0.0764
	WT/2	13/20	339947.6	60683.5	0.0021	19/20	615354.5	88813.6	0.0001	19/20	1083794.9	154414.8	0.0004
	WT/3	14/20	395428.8	166260.6	0.0009	19/20	712166.7	195965.6	0.0001	20/20	1117316.7	131584.2	0
	WT/4	14/20	503356.7	154634.0	0.0017	18/20	866578.9	197105.3	0.0008	18/20	1595085.9	215993.2	0.0001
	WT/5	18/20	707941.4	200621.3	0.0006	19/20	1375098.1	237743.3	0.0001	6/20	1812298.3	201695.2	0.0016

Pop size	240				480				960			
Operator	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
OX	0/20	-	-	0.1575	0/20	-	-	0.1112	0/20	-	-	0.0766
PMX	0/20	-	-	1.2795	0/20	-	-	0.9500	0/20	-	-	0.6934
EER	0/20	-	-	0.0725	1/20	109242.0	0.0	0.0231	2/20	261935.0	6349.0	0.0076

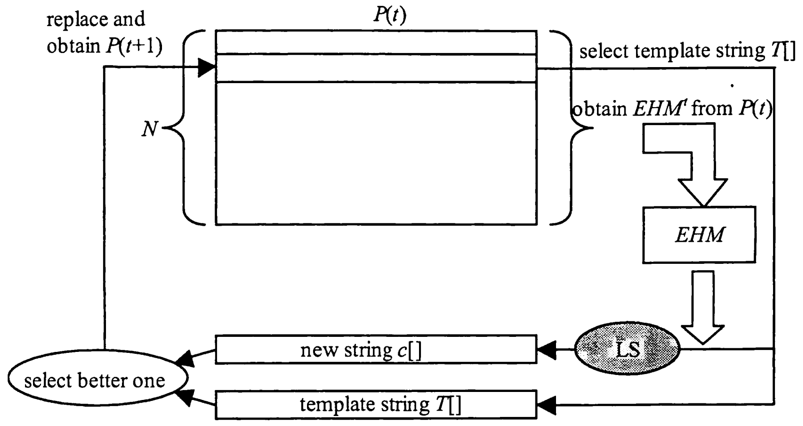
replace and obtain $P(t+1)$
$P(t)$
select template string $T[]$
N
obtain EHM^t from $P(t)$
EHM
LS
new string $c[]$
template string $T[]$
select better one

Fig. 7. Combining EHBSA with local search.

Comparing the performance of EHBSA/WT with that of other operators, EHBSA/WT was better than EER and it was significantly better than OX and PMX. One big difference between EHBSA/WT and EER is that EHBSA/WT requires a smaller population.

Results on pr76 are shown in Table 2. EHBSA/WO could not find the optimum tour in pr76, which confirms bad scalability of this variant of EHBSA. On the other hand, EHBSA/WT/n found the optimal tour in almost all cases. With $N = 60$, EHBSA/WT/2, 3, 4, and 5 found the optimal tour 13, 14, 14, and 18 times, respectively. With $N = 120$, EHBSA/WT/2, 3, 4, and 5 found the optimal tour 19, 19, 18, and 19 times, respectively. With $N = 240$, EHBSA/WT had similar values of #OPT, showing larger value of MNE. Thus, we can see that the performance of EHBSA/WT is much better than EHBSA/WO in this experiment. Among the other operators, only EER was able to find the optimal tour in one run with $N = 480$ and in two runs for $N = 960$. OX and PMX could not find the optimal tour at all.

3. Improving performance of EHBSA with local search

Recombination-based optimizers sometimes use local search (LS) for improving solutions locally as the search progresses. In this section, we examine such a hybrid approach, where EHBSA is combined with local search heuristics for improving solutions. We tested two types of heuristics; one is a simple heuristic for solving TSP called 2-OPT, and the other is a sophisticated Lin-Kernighan [8,13] heuristic. A local search is applied on newly generated strings at every generation as shown in Fig. 7.

3.1. Combining EHBSA with 2-OPT

In this section we describe a combination of EHBSA with standard 2-OPT heuristic. Let us first describe the 2-OPT heuristic.

Table 3
EHBSA with 2-OPT on Berlin52 and pr76

TSP instance		EHBSA/WT/2	EHBSA/WT/2+2-OPT
berlin52	MNE	87270.3	90
	#OPT	14/20	20/20
pr76	MNE	339947.6	375.1
	#OPT	13/20	20/20

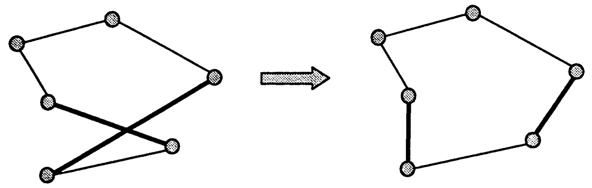
Fig. 8. An example of one step of 2-OPT local search.

3.1.1. 2-OPT Heuristic

2-OPT proceeds by checking pairs of nonadjacent edges in a given tour, and computing the improvement in the tour length after rearranging these pairs of edges by exchanging the terminal nodes of the two edges in each pair as shown in Fig. 8. If no pair of edges can be rearranged to improve the current tour, the algorithm is terminated. Otherwise, the pair of edges that improves the performance maximum (minimum tour length) is rearranged, and the algorithm is executed again.

Since there are L edges for a tour of length L, checking all pairs of edges will need O(L^2) steps. Determining an upper bound for the number of passes showing improvement is very difficult. A conservative bound would assume that all pairs of edges can be rearranged; but in practice no more than L improvements should be expected, yielding a bound O(L^3). In EHBSA we use 2-OPT to check if any improvement can be made in the newly generated solution, thereby increasing the computational cost of generating a new solution to O(L^3) from O(L^2).

3.1.2. Results

Table 3 shows the results of the combination of EHBSA/WT/2 with 2-OPT heuristic for berlin52 and pr76 problems (Section 2). The population size was chosen as 60. From this table, we can see that the EHBSA/WT with 2-OPT heuristic solves the problems much faster than without heuristic. In the following experiments, we evaluate the effect of the combination of EHBSA with 2-OPT heuristic using larger problems (TSP instances with a larger number of cities). For these experiments, the maximum number of evaluations is set to 100,000 for all tested TSP instances. The remaining settings are the same as those for the experiments described in Section 2.

The results for a 226-city problem pr226 are shown in Table 4. In this case, EHBSA/WO shows better performance than EHBSA/WT. For example, for a population of $N = 15$, #OPT of EHBSA/WO and EHBSA/WT ranges in 16–17, no explicit difference exists between them. However, MNE of EHBSA/WO is 277.7, which is about one third to half of those of EHBSA/WT (for different population sizes). For $N = 60$, #OPT of EHBSA/WO and EHBSA/WT are both 20. However, the MNE of EHBSA/WO is 990.3, which is again about one third to half of those of EHBSA/WT. Thus, we can say that for this problem EHBSA/WO is better than EHBSA/WT.

Since EHBSA/WO does not use a template, it can produce more unique new tours than EHBSA/WT. As a result 2-OPT efficiently improves the tour. Thus, we can see that the combination of EHBSA/WO and 2-OPT shows good performance for this problem. Note that the number of evaluations for solving the 226-city problem with a hybrid EHBSA+2-OPT is much lower than that for solving the 52-city problem with pure EHBSA. Thus, as expected, use of local search improves the performance and strengthens the robustness of EHBSA.

On the other hand, other recombination operators perform rather poorly. OX succeeded in 14 out of the 20 runs with $N = 240$ with MNE = 3658.5. With $N = 960$, it succeeded in 20 runs, but MNE was much higher. EER and PMX showed similar performance to OX. This confirms that both EHBSA/WO and EHBSA/WT are better than the standard algorithms when combined with a local search heuristic.

Table 5 shows the results on a larger problem, the 318-city problem lin318. EHBSA/WO again showed a lesser value of MNE than EHBSA/WT. However, its success rate is lower than EHBSA/WT. For this problem, EHBSA/WT/3, population size $N = 30$ showed #OPT = 20 and MNE = 11928.4. Although #OPT increased with n of EHBSA/WT/n, values of MNE also increased. The number of evaluations increased compared to the 226-city problem, but it is still smaller than that for the 52-city problem without the local search. This reconfirms the advantages of using local search in combination with EHBSA (this time EHBSA/WT is better than EHBSA/WO). The performance of OX and PMX decreased compared to the 226-city TSP, but for some settings these operators yielded comparable performance to EHBSA. However, it seems that this TSP instance is relatively easy for OX and PMX. EER performed the worst.

Table 6 shows that increasing the number of cities from 318 to 439 necessitated a slight increase in the

Table 4
Results of pr226

Pop size		15				30				60			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
EHBSA	WO	16/20	277.7	64.8	0.0004	17/20	534.6	164.1	0.0000	20/20	990.3	144.3	0
	WT/2	16/20	519.2	94.0	0.0000	20/20	964.8	165.7	0	20/20	1639.9	413.1	0
	WT/3	15/20	758.2	226.6	0.0003	20/20	1295.7	355.4	0	20/20	1860.8	281.9	0
	WT/4	15/20	779.9	164.7	0.0000	20/20	1369.0	310.7	0	20/20	2171.1	436.9	0
	WT/5	17/20	986.9	233.3	0.0000	20/20	1626.9	297.4	0	20/20	2702.1	632.5	0
Pop size		240				480				960			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
OX		14/20	3658.5	686.0	0.0001	19/20	6072.1	1113.8	0.0000	20/20	11363.9	1438.3	0
PMX		17/20	3363.9	551.8	0.0000	19/20	5996.7	1401.7	0.0000	20/20	11076.0	1149.5	0
EER		12/20	3644.8	743.8	0.0000	19/20	6450.7	1446.5	0.0000	19/20	10905.3	3718.0	0.0000

Table 5
Results for lin318

Pop size		15				30				60			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
EHBSA	WO	2/20	2305.5	108.5	0.0019	12/20	5580.3	976.4	0.0005	18/20	11664.1	1613.2	0.0002
	WT/2	10/20	7830.7	9183.7	0.0003	18/20	7850.9	2097.9	0.0001	20/20	15220.1	2706.8	0
	WT/3	16/20	8993.6	6306.7	0.0003	20/20	11928.4	6790.4	0	20/20	19997.8	6466.6	0
	WT/4	20/20	15970.7	10235.3	0	20/20	20865.1	9506.9	0	20/20	30609.1	12317.4	0
	WT/5	20/20	22430.1	19460.7	0	20/20	25825.0	13329.0	0	20/20	35579.2	16460.6	0
Pop size		240				480				960			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
OX		5/20	8660.6	1815.8	0.0009	15/20	16291.3	3390.2	0.0003	20/20	26704.1	3297.2	0
PMX		6/20	7413.5	1675.6	0.0008	13/20	14219.2	2521.2	0.0003	20/20	29290.9	3683.7	0
EER		3/20	10815.3	372.0	0.0006	12/20	21156.8	1375.3	0.0002	17/20	42207.2	5354.5	0.0001

population size used by EHBSA. For many settings, the number of evaluations for the 439-city problem decreased compared to the smaller problem of 318-cities presented above, which indicates good scalability of EHBSA. However, other recombination operators show a significant decrease in performance compared to the 318-city problem (Table 5). EER is not capable of achieving reliable convergence to the optimum even with a population size $N = 960$, whereas convergence of PMX and OX became reliable only with $N = 960$, showing very inferior performance compared to EHBSA/WT with almost all settings.

To summarize the results for hybrid algorithms, we can observe that EHBSA provides robust performance (over all population sizes) even for large TSP instances. The other recombination based methods (combined with 2-OPT heuristic) for TSP provided performance that is much inferior with respect to reliability and computational efficiency. #OPT found by EHBSA is always more than those of other operators (even with a larger population size). Correspondingly, MNE is much less for EHBSA than those of other operators. Comparing EHBSA/WO and EHBSA/WT we can say that EHBSA/WO works better than EHBSA/WT for smaller problems, whereas EHBSA/WT mostly shows superior performance for larger problems.

3.2. *Combining EHBSA with lin-kernighan heuristic*

In this section we describe a combination of EHBSA with Lin-Kernighan heuristic. Let us first describe the heuristic.

3.2.1. *Lin-kernighan heuristic*

Lin-Kernighan (LK) [8,13] local search is based on 2-OPT moves. The current solution is viewed as an *anchored* Hamiltonian path P rather than as a Hamiltonian circuit. The *anchor* of the path is a fixed end-point

Table 6
Results for pr439

Pop size		15				30				60			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
EHBSA	WO	0/20	NaN	NaN	0.0008	0/20	NaN	NaN	0.0006	7/20	9234.7	1827.4	0.0002
	WT/2	6/20	2681.8	917.6	0.0003	13/20	5659.3	2214.6	0.0001	18/20	10284.9	2107.9	0.0000
	WT/3	6/20	11649.8	13536.9	0.0002	15/20	9084.7	3920.3	0.0001	19/20	14321.3	3727.9	0
	WT/4	14/20	8812.1	5820.7	0.0001	19/20	13021.1	6705.4	0.0000	20/20	17126.9	3509.9	0
	WT/5	11/20	19993.1	18286.6	0.0001	19/20	24371.4	17549.8	0.0000	20/20	25006.0	5944.2	0
Pop size		240				480				960			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
OX		1/20	49358.0	0.0	0.0007	4/20	15203.5	999.0	0.0004	10/20	29125.3	3199.9	0.0001
PMX		2/20	8532.0	425.0	0.0008	5/20	17100.0	2470.6	0.0002	10/20	31571.7	4176.3	0.0002
EER		1/20	10815.3	0.0	0.0003	4/20	25714.8	1980.0	0.0001	8/20	65775.5	20199.8	0.0001

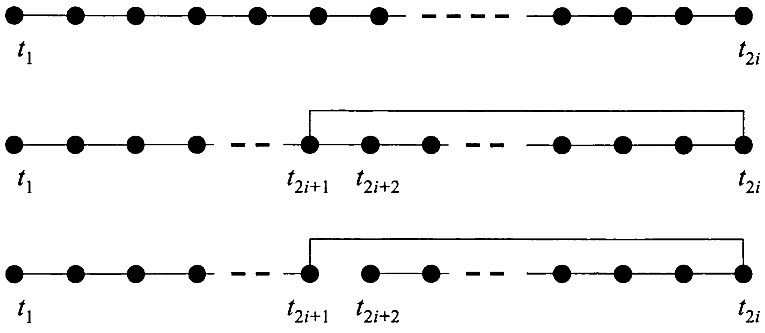

Fig. 9. An example of one step of LK local search.

city t_1, as illustrated in the first part of Fig. 9. Let t_{2i} denote the other endpoint of the path P_i that exists at the beginning of step i of the LK search. The tour corresponding to path P_i can then be obtained by adding the edge $\{t_{2i}, t_1\}$. At each step we only consider 2-OPT moves that flip some suffix of the path, i.e. one of the tour edges being broken is $\{t_1, t_{2i}\}$. Furthermore, the new neighbor t_{2i+1} of t_{2i} must be such that the length of the *one-tree* (spanning tree plus one edge) obtained by adding the edge $\{t_{2i}, t_{2i+1}\}$ to Pi (middle part of Fig. 9) is less than the length of the best tour seen so far. This restriction is a generalization of the criterion in 2-OPT that $d(t_2, t_3)$ be less than $d(t_1, t_2)$. The general one-tree restriction can be implemented using neighbor lists similarly as for 2-OPT. As in the neighbor-list implementations of 2-OPT, using neighbor-lists of length $k$imposes an additional constraint that only the k nearest cities to t_{2i} can be considered as candidates for t_{2i+1}, even if additional cities would satisfy the one-tree restriction.

There are many variant implementations for LK search (e.g., [10,16]). An important, and widely adopted, part of Lin and Kernighan's overall tour-finding scheme is the repeated use of the basic LK algorithm. The scheme is referred to as *Iterated Lin-Kernighan* [11]. Here we used an iterated implementation of Concord TSP solver, where we used "walk kick" on it. For more details, please see http://www.tsp.gatech.edu/concorde.html.

3.2.2. Results

Results for a 3795-city problem fl3795 using EHBSA sampling combined with LK local search is given in Table 7, and those for a 5934-city problem rl5934 in Table 8. For these experiments, the maximum number of evaluations is set to 1,000 and 3,000 for fl3795 and rl5934, respectively. The remaining settings are the same as those for the experiments described in Section 2.

It is evident from the results given in Tables 7 and 8, that incorporation of LK search heuristics with either the recombination based standard algorithms or EHBSA/WT improves (in terms of both MNE and #OPT) the performance. They can solve bigger prob-

Table 7
Results for fl3795

Pop size		5				10				15			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
EHBSA	WT/2	20/20	55.6	24.1	0	20/20	81.1	35.3	0	20/20	78.5	40.7	0
	WT/3	20/20	70.4	69.5	0	20/20	76.7	35.9	0	20/20	60.4	32.7	0
	WT/4	20/20	94.3	64.9	0	20/20	67.0	36.0	0	20/20	65.4	34.7	0
	WT/5	20/20	81.6	79.4	0	20/20	79.1	52.1	0	20/20	86.0	40.6	0
Pop size		10				15				30			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
OX		17/20	43.4	14.4	0.0003	19/20	49.8	23.9	0.0001	20/20	77.9	31.4	0
PMX		18/20	50.7	22.5	0.0008	19/20	60.1	30.2	0.0001	20/20	84.8	27.8	0
EER		20/20	149.1	70.5	0.0000	20/20	167.7	77.6	0	20/20	235.7	117.9	0

Table 8
Results for rl5934

Pop size		5				10				15			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
EHBSA	WT/2	20/20	292.3	75.4	0	20/20	520.9	146.294	0	20/20	575.9	220.192	0
	WT/3	20/20	288.2	191.1	0	20/20	352.5	130.982	0	20/20	540.9	165.039	0
	WT/4	20/20	270.1	130.0	0	20/20	372.8	126.72	0	20/20	496.8	135.91	0
	WT/5	20/20	278.6	180.9	0	20/20	403.3	193.322	0	20/20	523.7	194.093	0
Pop size		15				30				60			
Operator		#OPT	MNE	STD	Error	#OPT	MNE	STD	Error	#OPT	MNE	STD	Error
OX		11/20	167.5	43.0779	0.0003	19/20	285.2	75.0882	0.0001	20/20	412.6	93.6421	0
PMX		15/20	159.5	34.0996	0.0000	17/20	276.9	62.3335	0.0000	20/20	516.4	99.2514	0
EER		17/20	220.6	47.4676	0.0000	20/20	408.7	84.4797	0	20/20	800.8	74.8484	0

lems even with 5934 cities quite well. This establishes the utility of LK search in combination with evolutionary algorithms.

It is seen from the Tables that for all population sizes (even for a small population size of 5) EHBSA/WT combined with a LK heuristic succeeded in finding out the optimum solution for all runs. However, other operators (OX, PMX, EER) combined with LK heuristics could not find the optimum in all runs for smaller population sizes. Let us take the case of population size 15 and rl5934 problem. The EHBSA/WT could detect the optimum 100% times; whereas the OX detected the optimum 11 times, PMX, 15 and EER, 17 times out of 20. They could succeed in all runs only when the population size was as big as 60. Other measures like MNE, STD are comparable for all techniques. Similar are the results even for the 3795 cities problem.

To summarize the results for hybrid algorithms, we can observe that EHBSA provides robust performance (over all population sizes) even for large TSP instances. The other recombination based methods (combined with LK heuristic) for TSP provided performance that is inferior with respect to reliability and computational efficiency. #OPT found by EHBSA is always 100%; whereas the other operators needed larger population size for achieving this result. Thus, we can say that hybrid EHBSA works well for smaller population sizes, and thus are robust and incur less computational cost.

An experiment was also conducted to check if a particular template (e.g., the present best individual) would work better than a random template, but the results did not show much improvement.

4. Summary and conclusions

A concept of edge histogram based sampling (EHBSA) for solving permutation problems was introduced by Tsutsui [23]. Its applicability to solve TSP and flow shop-scheduling problems was also demonstrated [25,26].

In this paper EHBSA has been hybridized with some popular well-known local search heuristics (namely the

2-OPT and the Lin-Kernighan heuristics) to take advantage of local search. Experimental evidences show that this hybrid technique decreases the computational requirements for finding a globally optimal tour in TSP, providing a method capable of solving significantly larger problems of thousands of cities. The method is very robust in the sense of detecting global optima for a wide variety of population sizes. The effectiveness of EHBSA, over other recombination based techniques, is more clear when we use a weaker heuristic (like 2-OPT), whereas the effectiveness of EHBSA is less if we use a sophisticated heuristic like the Lin-Kernighan heuristic. However, we can always observe some advantage of EHBSA over the classical recombination based techniques.

Note that EHBSAs do not use any specific domain knowledge of TSP such as the presence of all cities in the permutation or the distance metric for estimating the distances between the cities. That is why EHBSA can be used to solve other permutation problem like the flow shop-scheduling problem [25,26]. In the future, more in-depth empirical analysis is required to confirm that EHBSA combined with domain specific heuristic performs better in other permutation problems also.

Acknowledgments

The authors gratefully acknowledge the reviewers' valuable suggestions and constructive criticism that improved the quality of the article. This research was partially supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under Grant-in-Aid for Scientific Research number 13680469. A part of this work was done when Ashish Ghosh visited Hannan University, Osaka with a fellowship from the Japan Society for the Promotion of Science. Martin Pelikan was also partially supported by a research award from the Research Board at the University of Missouri.

References

[1] P.A.N. Bosman and D. Thierens, *Continuous iterated density estimation evolutionary algorithms within the IDEA framework*, Workshop Proc. of the Genetic and Evolutionary Computation Conf, 2000 (GECCO-2000), 2000, 197–200.

[2] P.A.N. Bosman and D. Thierens, *Crossing the Road to Efficient IDEAs for Permutation Problems*, Proc. of the Genetic and Evolutionary Computation Conf. – GECCO-2001, 2001, 219–226.

[3] P.A.N. Bosman and D. Thierens, *Permutation Optimization by Iterated Estimation of Random Keys Marginal Product Factorizations*, Proc. of the 7th International Conf. on Parallel Problem Solving From Nature, 2002, 331–340.

[4] M. Dorigo, V. Maniezzo and A. Colorni, *The Ant System: Optimization by a Colony of Cooperating Agents*, IEEE Trans. on Systems, Man, and Cybernetics-Part B **26**(1) (1996), 29–41.

[5] D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Reading, MA: Addison-Wesley, 1989.

[6] D.E. Goldberg, *The design of innovation: Lessons from and for competent genetic algorithms*, Kluwer, 2002.

[7] G. Harik, *Finding multimodal solutions using restricted tournament selection*, Proc. of the 6th International Conf. on Genetic Algorithms, Morgan Kaumann, 1995, 28–31.

[8] K. Helsgaun, An effective implementation of the Lin-Kernighan traveling salesman heuristic, *European Journal of Operational Research* **126** (2000), 106–130.

[9] J.H. Holland, *Adaptation in natural and artificial systems*, Ann Arbor, MI: The University of Michigan Press, 1975.

[10] D.S. Johnson and L.A. McGeoch, The traveling salesman problem: a case study in local optimization, in: *Local Search in Combinatorial Optimization*, E.H.L. Aarts and J.K. Lenstra, eds, Wiley, 1997, pp. 215–310.

[11] D.S. Johnson, Experimental Analysis of Heuristics for the STSP, in: *The Traveling Salesman Problem and Variations*, (Vol. 9), G. Gutin and A.P. Punnen, eds, Kluwer Academic Publishers, 2002, pp. 369–443.

[12] P. Larrañaga and J.A. Lozano, eds, *Estimation of Distribution Algorithms*, Boston, MA: Kluwer Academic Publishers, 2002.

[13] S. Lin and B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, *Operations Research* **21** (1973), 498–516.

[14] S.W. Mahfoud, *Crowding and preselection revisited*, Proc. of the International Conf. on Parallel problem solving from nature 2, 1992, 27–36.

[15] H. Muehlenbein and G. Paass, *From recombination of genes to the estimation of distributions I. Binary parameters*, Proc. of the 4th International Conf. on Parallel Problem Solving from Nature, 1996, 178–187.

[16] D. Neto, *Efficient Cluster Compensation for Lin-Kernighan Heuristics*, Ph.D. Thesis, Department of Computer Science, University of Toronto, 1999.

[17] I. Oliver, D. Smith and J. Holland, *A study of permutation crossover operators on the travel salesman problem*, Proc. of the 2nd International Conf. on Genetic Algorithms, 1987, 224–230.

[18] M. Pelikan, Bayesian optimization algorithm: From single level to hierarchy, Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. Also IlliGAL Report No. 2002023, 2002.

[19] M. Pelikan, D.E. Goldberg and F. Lobo, A survey of optimization by building and using probabilistic models, *Computational Optimization and Applications* **21**(1) (2002), 5–20, Also IlliGAL Report No. 99018.

[20] V. Robles, P.D. Miguel and P. Larrañaga, Solving the traveling salesman problem with EDAs, in: *Estimation of Distribution Algorithms*, P. Larranaga and J.A. Lozano, eds, Kluwer Academic Publishers, 2002, pp. 211–229.

[21] T. Starkweather, S. McDaniel, K. Mathias D. Whitley and C. Whitley, *A comparison of genetic sequence operators*, Proc. of the 4th International Conf. on Genetic Algorithms, Morgan Kaufmann, 1999, 69–76.

[22] D. Thierens, *Analysis and design of genetic algorithms*, Doctoral dissertation, Katholieke Universiteit Leuven, Belgium, 1995.

[23] S. Tsutsui, *Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram*, Proc. of the Seventh International Conf. on Parallel Problem Solving from Nature (PPSN VII), Springer-Velag, 2002, 224–233.

[24] S. Tsutsui, M. Pelikan and D.E. Goldberg, Using Edge Histogram Models to Solve Permutation Problems with Probabilistic Model-Building Genetic Algorithms, Tech. Rep. IlliGAL Report No. 2003022, University of Illinois, 2003.

[25] S. Tsutsui and M. Miki, *Solving Flow Shop Scheduling Problems with Probabilistic Model-Building Genetic Algorithms using Edge Histograms*, Proc. of the 4th Asia-Pacific Conf. on Simulated Evolution And Learning (SEAL02), 2002, 465–471.

[26] S. Tsutsui and M. Miki, Using Edge Histogram Models to Solve Flow Shop Scheduling Problems with Probabilistic Model-Building Genetic Algorithms, in: *Recent Advances in Simulated Evolution and Learning*, (Vol. 13), K.C. Tan, M.H. Lim, X. Yao and L. Wang, eds, Advances in Natural Computation Series, World Scientific, 2004, pp. 230–249.

[27] D. Whitley, *The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best*, Proc. of the 3rd International Conf. on Genetic Algorithms, Morgan Kaufmann, 1989, 116–121.