

# The Effect of Introducing a *Tag Node* in Solving Scheduling Problems Using Edge Histogram Based Sampling Algorithms

**Shigeyoshi Tsutsui**

Department of Management Information  
Hannan University  
5-4-33 Amamihigashi, Matsubara,  
Osaka 580-8502, Japan  
tsutsui@hannan-u.ac.jp

**Tomoyuki Hiroyasu**

Department of Knowledge Engineering and  
Computer Sciences, Doshisha University  
1-3 Tatara, Miyakodani, Kyo-tanabe,  
Kyoto, 610-0321, Japan  
{tomo@is, mmiki@mail}.doshisha.ac.jp

**Mitsunori Miki**

**Abstract-** We have proposed probabilistic model-building genetic algorithms (PMBGAs) for solving sequence problems using edge histogram-based sampling algorithms (EHBSAs) in a previous paper. In this paper, we explore the effect of introducing a tag node in string representations for solving flow shop scheduling problems with EHBSAs. The results showed EHBSAs using strings with a tag worked better than EHBSAs using strings without a tag node.

## 1 Introduction

There has been a growing interest in developing evolutionary algorithms based on probabilistic models [Pelikan 99], [Larranaga 02]. In this scheme, the offspring population is generated according to the estimated probabilistic model of the parent population instead of using traditional recombination and mutation operators. The model is expected to reflect the problem structure, and as a result it is expected that this approach provides a more effective mixing capability than recombination operators in traditional GAs. These algorithms are called probabilistic model-building genetic algorithms (PMBGAs) or estimation of distribution algorithms (EDAs).

Many studies on PMBGAs have been performed in discrete (mainly binary) domains [Baluja 94], [Baluja 97], [Servet 97], [Harik 98], [Harik 99], [Pelikan 00] and there are several attempts to apply PMBGAs in continuous domains [Mühlenbein 96], [Sebag 98], [Bosman 99], [Bosman 00], [Larranaga 99], [Larranaga 00], [Tsutsui 01]. However, few studies on PMBGAs in permutation representation domains are found [Robles 02], [Larranaga 02].

In [Tsutsui 02a], we have proposed an approach to PMBGAs in permutation domains, focusing on solving the Traveling Salesman Problem (TSP) and compared its performance with traditional recombination operators. In this approach, we developed a symmetrical edge histogram matrix from the current population, where an edge is a link between two nodes in a string. We then sampled nodes of a new string according to the edge histogram matrix. We called this method the edge histogram-based sampling algorithm (EHBSA). We proposed two types of EHBSAs, an *edge histogram-based*

*sampling algorithm without template (EHBSA/WO)* and an *edge histogram-based sampling algorithm with template (EHBSA/WT)*. The results showed EHBSA/WT worked fairly well on the test suite.

In [Tsutsui 02b], we applied EHBSAs to flow shop scheduling problems. Since in a scheduling problem, each edge is directional, we used an asymmetrical edge histogram matrix from the current population. The results showed EHBSA/WT worked well also in flow shop scheduling problems.

In this paper, we explore the effect of introducing a *tag node* (TN) in a chromosome representation and show the TN can improve the performance of EHBSA with an asymmetrical edge histogram matrix. Here, the TN is a virtual node and is used to indicate which node is the first node (i.e., first job) in a string representation. Normally, in a sequence problem, the node in the first position in a string is assumed to be the first node (job). In our approach, we assume each position of nodes in a string containing a TN to be relative. Absolute position is obtained as follows: in a string with a TN, the node immediately following the TN is interpreted as the first node, i.e., job in a string. The TN is virtual because it does not correspond to any real nodes or jobs. However, the TN works as if it is a normal node in modeling and sampling of the EHBSA.

In the remainder of this paper, a brief review of EHBSA is described in Section 2, tag nodes and corresponding sampling EHBSAs are discussed in Section 3, and the empirical analysis is given in Section 4. Section 5 concludes the paper.

## 2 EHBSAs

This section reviews how the edge histogram based sampling algorithm (EHBSA) can be used to (1) model promising solutions and (2) generate new solutions by simulating the learned model.

### 2.1 The Basic Description of the Algorithm

An edge is a link or connection between two nodes and has important information about the permutation string. Some crossover operators, such as Edge Recombination (ER) [Whitley 89] and enhanced ER (eER) [Starkweather, 91], which are used in traditional two-parent recombination, use

the edge distribution only in the two parents strings. The basic idea of the edge histogram based sampling algorithm (EHBSA) is to use the edge histogram of the whole population in generating new strings. The algorithm starts by generating a random permutation string for each individual population of candidate solutions. Promising solutions are then selected using any popular selection scheme. An *edge histogram matrix (EHM)* for the selected solutions are constructed and new solutions are generated by sampling, based on the edge histogram matrix. New solutions replace some of the old ones and the process is repeated until the termination criteria are met.

## 2.2 Developing Edge Histogram Matrix

### 2.2.1 A Symmetrical Edge Histogram Matrix

A symmetrical edge histogram matrix is intended to apply to problems such as a symmetrical TSP where each edge has no direction. In this case, we assume edges 1→2 and 2→1 are identical. If there is an edge 1→2, then we assume the edge 2→1 also exists. Here, 1 and 2 are nodes.

Let string of  $k$ -th individual in population  $P(t)$  at generation  $t$  represent  $s_k^t = (\pi_k^t(0), \pi_k^t(1), \dots, \pi_k^t(L-1))$ . ( $\pi_k^t(0), \pi_k^t(1), \dots, \pi_k^t(L-1)$ ) are the permutation of  $(0, 1, \dots, L-1)$ , where  $L$  is the length of the permutation. Then, a symmetrical edge histogram matrix  $EHM_{(s)}^t(e_{ij}^t)$  ( $i, j = 0, 1, \dots, L-1$ ) of population  $P(t)$  consists of  $L^2$  elements as follows:

$$e_{i,j}^t = \begin{cases} \sum_{k=1}^N (\delta_{ij}(s_k^t) + \delta_{ji}(s_k^t)) + \varepsilon & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \quad (1)$$

where  $N$  is the population size,  $\delta_{ij}(s_k^t)$  is a delta function defined as

$$\delta_{i,j}(s_k^t) = \begin{cases} 1 & \text{if } \exists h [h \in \{0, 1, \dots, L-1\} \wedge \\ & \pi_k^t(h) = i \wedge \pi_k^t((h+1) \bmod L) = j] \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

and  $\varepsilon$  ( $\varepsilon > 0$ ) is a bias to control *pressure* in sampling nodes, just like those used for adjusting the selection pressure in the proportional selection in GAs. The average number of edges of element  $e_{i,j}^t$  in  $EHM_{(s)}^t$  is  $2LN/(L^2-L) = 2N/(L-1)$ . Values  $e_{i,j}^t$  will be later be used in a variant of proportionate selection, and the value of  $\varepsilon$  thus influences selection pressure toward edges. To achieve comparable selection pressure for all problems and parameter settings,  $\varepsilon$  should be proportional to the expected value of  $e_{i,j}^t$ . Therefore,

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

where  $B_{\text{ratio}}$  ( $B_{\text{ratio}} > 0$ ), the bias ratio, is a constant related to selection pressure.

### 2.2.2 An Asymmetrical Edge Histogram Matrix

An asymmetrical edge histogram matrix is intended to apply to problems such as asymmetrical TSP and scheduling problems where each edge has direction. For example, in flow shop scheduling problems in this paper, each string represents a sequence of jobs to be processed. A

string  $s = \{1, 2, 3, 0\}$  means that first, job 1 must be processed, then jobs 2, 3, and 0 follow in this sequence. In this case, there are four edges, i.e., 1→2, 2→3, 3→0, and 0→1. Thus in a scheduling problem, each edge is directional and the edge-histogram matrix becomes asymmetrical.

An asymmetric edge-histogram matrix  $EHM_{(A)}^t(e_{ij}^t)$  ( $i, j = 0, 1, \dots, L-1$ ) of population  $P(t)$  consists of  $L^2$  elements as follows:

$$e_{i,j}^t = \begin{cases} \sum_{k=1}^N \delta_{ij}(s_k^t) + \varepsilon & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \quad (4)$$

where  $\delta_{ij}(s_k^t)$  is a delta function defined by Eq. 2. The average number of edges of element  $e_{i,j}^t$  in  $EHM_{(A)}^t$  is  $LN/(L^2-L) = N/(L-1)$ . So the bias  $\varepsilon$  for  $EHM_{(A)}^t$  is defined as in Eq. 3 as follows:

$$\varepsilon = \frac{N}{L-1} B_{\text{ratio}} \quad (5)$$

## 2.3 Sampling Methods

In this subsection, we review how to sample a new string from the edge histogram matrix. There are two types of sampling methods; one is an edge-histogram based sampling algorithm without template (EHBSA/WO), and the other an edge-histogram based sampling algorithm with template (EHBSA/WT).

### 2.3.1 EHBSA/WO

In a symmetrical EHM such as used in the symmetrical TSP, the absolute positions (loci) of a string have no meaning. For example, string  $s_1 = (0, 1, 2, 3, 4)$  and string  $s_2 = (4, 0, 1, 2, 3)$  represent the same tour. However, in an asymmetrical EHM such as for scheduling problems, these two strings represent two completely different solutions. Thus, we must consider how to determine the initial position and what node we assign to the position. In [Tsuetsui 02b], we have proposed two types of EHBSA/WO, EHBSA/WO/T and EHBSA/WO/R.

#### EHBSA/WO/T

Let us represent a new individual permutation by  $c[]$ . In EHBSA/WO/T, the initial sampling position is always the first position, i.e.  $c[0]$ . The value for  $c[0]$  is taken from a *pseudo template individual PT[]* which is taken from the current population  $P(t)$  randomly, and a new individual permutation  $c[]$  is generated straightforwardly as follows:

1. Set the position counter  $p \leftarrow 0$
2. Choose a pseudo template  $PT[]$  from  $P(t)$
3. Obtain first node as  $c[0] \leftarrow PT[0]$
4. Construct a roulette wheel vector  $rw[]$  from  $EHM_{(A)}^t$  as  $rw[j] = e_{c[p],j}^t$  ( $j=0, 1, \dots, L-1$ )
5. Set to 0 previously sampled nodes in  $rw$  ( $rw[c[i]] = 0$  for  $i=0, 1, \dots, p$ )
6. Sample the next node  $c[p+1]$  with probability  $rw[x] / \sum_{j=0}^{L-1} rw[j]$  using roulette wheel  $rw[]$
7. Update the position counter  $p \leftarrow p+1$

8. If  $p < L-1$ , go to Step 4
9. Obtain a new individual string  $c[]$

### EHBSA/WO/R

In EHBSA/WO/T, the initial sampling position is fixed to 0. On the other hand, in EHBSA/WO/R, the initial sampling position is chosen from  $[0, L-1]$  randomly as follows:

1. Obtain random initial sampling position  $p_{\text{initial}}$  from  $[0, L-1]$
2. Choose a pseudo template  $PT[]$  from  $P(t)$
3. Obtain first node as  $c[p_{\text{initial}}] \leftarrow PT[p_{\text{initial}}]$
4. Set the position counter  $p \leftarrow p_{\text{initial}}$
5. Construct a roulette wheel vector  $rw[]$  from  $EHM_{(A)'}'$  as  $rw[j] = e^{e'_{c(p),j}}$  ( $j=0, 1, \dots, L-1$ )
6. Set to 0 previously sampled nodes in  $rw$  ( $rw[c[i]] = 0$  for  $i = p_{\text{initial}}, (p_{\text{initial}} + 1) \bmod L, \dots, p$ )
7. Sample the next node  $c[(p+1) \bmod L]$  with probability  $rw[x] / \sum_{j=0}^{L-1} rw[j]$  using roulette wheel  $rw[]$
8. Update the position counter  $p \leftarrow (p+1) \bmod L$
9. If  $(p+1) \bmod L \neq p_{\text{initial}}$ , go to Step 5
10. Obtain a new individual string  $c[]$

Here, note both EHBSA/WO/T and WO/R are the same for problems which have a symmetrical EHM.

### 2.3.2 EHBSA/WT

EHM is 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 string. In generating each new individual, a template individual is chosen from  $P(t)$  (normally, randomly). The  $n$  ( $n > 1$ ) cut points are applied to the template randomly. When  $n$  cut points are obtained for the template, the template should be divided into  $n$  segments. Then, we choose one segment randomly and sample nodes for the segment. Nodes in other  $n-1$  segments remain unchanged. We denote this sampling method as EHBSA/WT/ $n$ . Since the average length of one segment is  $L/n$ , EHBSA/WT/ $n$  generates new strings which are different  $L/n$  nodes on average from their templates. Fig. 2 shows an example of EHBSA/WT/3. In this example, nodes of new string from after cut[2] and before cut[1] are the same as the nodes of the template. New nodes are sampled from cut[1] up to, but not including, cut[2] based on the  $EHM'$ .

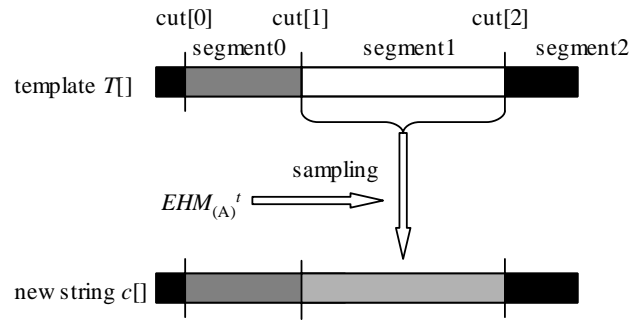


Fig. 1. An example of EHBSA/WT/3

The sampling method for EHBSA/WT/ $n$  is as follows:

1. Choose a template  $T[]$  from  $P(t)$ .
2. Obtain a sorted cut point array  $cut[0], cut[1], \dots, cut[n-1]$  randomly.
3. Choose a cut point  $cut[l]$  by generating random number  $l \in [0, n-1]$ .
4. Copy nodes in  $T[]$  to  $c[]$  from after  $cut[(l+1) \bmod n]$  and before  $cut[l]$
5. Set the position counter  $p \leftarrow cut[l]-1$
6. Construct a roulette wheel vector  $rw[]$  from  $EHM_{(A)'}'$  as  $rw[j] = e^{e'_{c(p),j}}$  ( $j=0, 1, \dots, L-1$ )
7. Set to 0 copied and previously sampled nodes in  $rw$  ( $rw[c[i]] = 0$  for  $i = cut[(l+1) \bmod n], \dots, p$ )
8. Sample the next node  $c[(p+1) \bmod L]$  with probability  $rw[x] / \sum_{j=0}^{L-1} rw[j]$  using roulette wheel  $rw[]$
9. Update the position counter  $p \leftarrow (p+1) \bmod L$
10. If  $(p+1) \bmod L = cut[(l+1) \bmod n]$ , go to Step 6
11. Obtain a new individual string  $c[]$

Let  $f(x)$  be the probability density function of the length of a segment to be sampled in EHBSA/WT/ $n$ . Then,  $f(x)$  is obtained as [Tsutsumi 03]

$$f(x) = \frac{(n-1)}{L} \left(1 - \frac{x}{L}\right)^{n-2}. \quad (6)$$

Fig. 2 shows the probability density function  $f(x)$ . For  $n = 2$ ,  $f(x)$  is uniformly distributed on  $[0, L]$ . Thus, the length of a segment to be sampled in EHBSA/WT/2 is uniformly distributed on  $[0, L]$ . When the length of the segment is small, the EHBSA/WT/2 samples a small number of nodes, performing a kind of local search improvement over the template 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 improvement over the template individual or produces a new string. Thus, we can expect the EHBSA/WT/2 to work by balancing global and local improvements. For  $n > 2$ , short segments are more likely to occur.

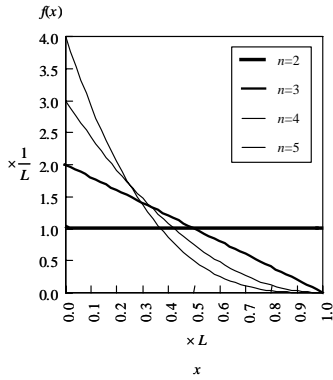


Fig. 2. Probability density function  $f(x)$

### 3 Introducing the Tag Node (TN) to EHBSAs

#### 3.1 Tag Node and Virtual String

In a scheduling problem where a solution is represented by a permutation string, the performance of each string is tightly linked not only to the relative sequence of nodes (jobs) but also to the absolute position of nodes in the string. Since EHM, described in Section 2, has no explicit information on the absolute positions of nodes in each string, it may be useful to introduce some additional information on the absolute position of each node in a string.

The *tag node* (TN) proposed in this paper is an approach to introduce information on the absolute position of each node in a string. In addition to normal nodes, we add a TN to each permutation string. We call a string with a TN a *virtual string* (VS). String length of a VS is  $L_{VS} = L+1$ , where  $L$  is the length of real string (RS; string without TN). The TN in VS works as a tag in a permutation string to indicate the first node (job), i.e., the next node following the TN is assumed to be the first node (job) in the solution. The TN is a *virtual node* because it does not correspond to any real nodes, or jobs. However, the TN in a VS is treated as if it is a normal node in modeling and sampling of the EHBSA.

Fig 3 shows how to obtain RS from VS. In this case,  $L = 6$ .

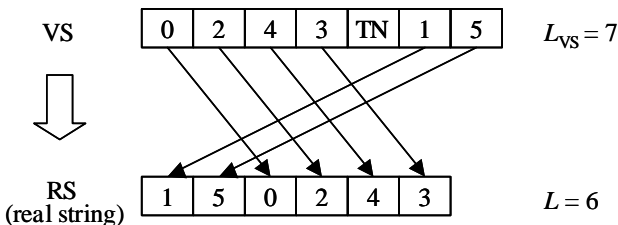


Fig. 3 An example of a virtual string

#### 3.2 Sampling Methods

Sampling methods for VS are the same as sampling in strings without TN, referred to in Section 2. Only a small difference exists in EHBSA/WO/T. In EHBSA/WO/T, sampling starts with fixing the node in position 0. In VS, this may correspond to sampling by fixing VS in position 0.

Sampling for VS in EHBSA/WO/R and EHBSA/WT is completely the same as sampling for strings without TN. Here, note that the TN is sampled in precisely the same manner as normal nodes and sampling is performed from a random position in a VS.

#### 3.3 Representation of the TN in a VS

We can use any symbol to represent the TN in a VS. However, for implementation convenience, we use an integer number to represent it in a VS. Let us consider the string in Fig. 3, for example. In this example, string length  $L = 6$  and the length of VS  $L_{VS} = 7$ . Nodes 0, 1, ..., 5 represent real nodes. Then, we assign number 6 to the TN (see Fig. 4). Thus, a VS of length  $L+1$  is represented as a permutation  $\{0, 1, \dots, L-1, L\}$  corresponding numbers 0, 1, ...,  $L-1$  to real nodes and number  $L$  to the TN. With this representation, we do not need any special modification in basic EHBSAs

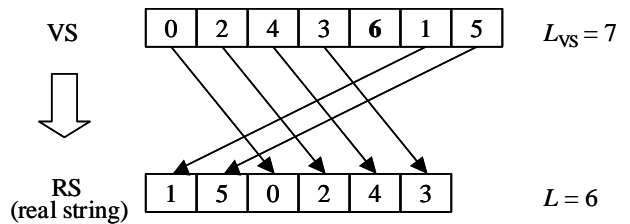


Fig. 4 Representation of the TN in a VS

### 4 Empirical Study

In this section, we explore the effectiveness of introducing the TN node in solving flow shop scheduling problems using EHBSAs.

#### 4.1 Experimental Methodology

##### 4.1.1 Evolutionary models

The evolutionary model is the same as the model used for symmetrical  $EHM_{(A)'} in [Tsutsui 02b] as follows:$

##### Evolutionary model for EHBSA/WT

Let the population size be  $N$ , and let it, at time  $t$ , be represented by  $P(t)$ . The population  $P(t+1)$  is produced as follows (Fig. 5):

1. Edge distribution matrix  $EHM_{(A)}'$  described in Subsection 2.2 b) is developed from  $P(t)$
2. A template individual  $T[]$  is selected from  $P(t)$  randomly

3. EHBSA/WT described in Subsection 2.3 b) is performed using  $EHM_{(A)'}$  and  $T[]$ , and generates a new individual  $c[]$
4. The new  $c[]$  individual is rearranged, removing the VN (see Figs xx and xx) and then evaluated
5. If  $c[]$  is better than  $T[]$ , then  $T[]$  is replaced with  $c[]$ , otherwise  $T[]$  remains, forming  $P(t+1)$

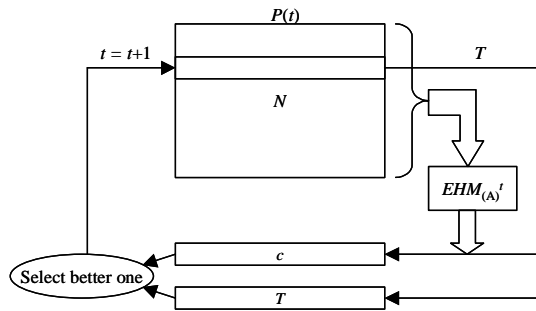


Fig. 5. Evolutionary model for EHBSA/WT

**The evolutionary model for EHBSA/WT and EHBSA/WR**

The evolutionary model for EHBSA/WR is basically the same as the model for EHBSA/WT, except EHBSA/WR uses a pseudo template  $PT[]$ .

**4.1.2 Flow shop problems and performance measures**

General assumptions of flow shop scheduling problems can be described as follows: Jobs are to be processed on multiple machines sequentially. There is one machine at each stage. Machines are available continuously. A job is processed on one machine at a time without preemption, and a machine processes no more than one job at a time. In this paper, we assume that  $L$  jobs are processed in the same order on  $m$  machines. This means that our flow shop scheduling is the  $L$ -job and  $m$ -machine sequence problem. The purpose of this problem is to determine the sequence of  $L$  jobs which minimizes the *makespan* (i.e., the completion time of all jobs). This sequence is denoted by a permutation string of  $\{0, 1, \dots, L-1\}$ .

As test problems, we generated two flow shop scheduling problems, [20-job 10-machine], [40-job 10-machine] problems. In designing each problem, we specified the processing time of each job at each machine as an random integer in the interval [1, 99].

50 runs were performed. Each run continued until the population was converged, or evaluations reached  $E_{max}$ .  $E_{max}$  was set to 200,000. The performance is measured by the minimum makespan (*best*), mean of the minimum makespan (*mean*) in 50 runs and standard deviation of the minimum makespan (*std*). Population sizes of 60, 120, 240 were used for EHBSAs. As to the bias ratio  $B_{ratio}$  in Eq. 5, a  $B_{ratio}$  values of 0.001 was used. As to the cut points,  $n$  for EHBSA/WT,  $n = 2, 3,$  and  $4$  were tested.

**4.1.3 Blind search**

In solving scheduling problems using GAs, mutation operators play an important role. Several types of mutation

operators are proposed. Also, it is well known that combining GAs with local optimization methods or heuristics greatly improve the performance of the algorithms. However, in this experiment, we use no mutation and no heuristic to see the pure effect of applying proposed algorithms. Thus, the algorithm is a *blind search*.

**4.2 Empirical analysis of results**

**4.2.1 Results on 20-job and 10-machine flow shop problem**

Results on the 20-job and 10-machine problem are shown in Table 1. As was found in [Tsutsui 02a] with TSP, we can confirm clearly that EHBSA/WT shows much better performance, i.e. smaller makespan, than EHBSA/WR in all results in the table. Both EHBSA/WR/T and EHBSA/WR/R showed obviously worse performance than EHBSA/WT/ $n$  ( $n = 2, 3, 4$ ).

In EHBSA/WR, we can see that there is meaningful difference between EHBSA/WR/T and EHBSA/WR/R. EHBSA/WR/R, which samples nodes from a randomly chosen position of the string, shows better performance than EHBSA/WR/T, which fixes the initial sampling position to the first position of the string.

Next, let's explore the effect of introducing the tag node (TN) to EHBSA/WT. The best value of the *mean* in methods without TN was observed in EHBSA/WT/2 with population size = 60 and the value = 1525.8. However, with TN, the value decreases to 1522.9. Although the difference is not so remarkable, the important observation is that in all experiments with EHBSA/WT, values of the *mean* with TN always showed better than those without TN. On average, the value of the *mean* with TN is smaller than those without TN by 0.31% .

Table 1 shows only results at evaluations = 200000. Fig. 6 shows the convergence processes of EHBSA/WT with and without TN for (a) EHBSA/WT/2, (b) EHBSA/WT/3, and (c) EHBSA/WT/4, with population size = 60. From this figure, we can see that values of mean with TN always converge faster than those without TN, taking significant smaller values.

**Table 1 Results on 20-job and 10-machineflowshop problem (makespan)**

EHBSA	TN	Performance Measure	Population Size		
			60	120	240
EHBSA/WO/T	without	best	1537.0	1545.0	1559.0
		mean	<b>1569.3</b>	<b>1570.1</b>	<b>1581.0</b>
		std	15.9	11.5	10.1
	with	best	1545.0	1536.0	1528.0
		mean	<b>1576.2</b>	<b>1560.5</b>	<b>1549.7</b>
		std	18.7	18.8	12.7
EHBSA/WO/R	without	best	1525.0	1525.0	1545.0
		mean	<b>1543.5</b>	<b>1544.8</b>	<b>1566.1</b>
		std	7.6	10.5	10.0
	with	best	1522.0	1519.0	1516.0
		mean	<b>1544.8</b>	<b>1533.1</b>	<b>1528.6</b>
		std	14.5	6.8	5.4
EHBSA/WT/2	without	best	1520.0	1520.0	1525.0
		mean	<b>1525.8</b>	<b>1527.8</b>	<b>1533.7</b>
		std	3.5	5.3	4.1
	with	best	1516.0	1516.0	1520.0
		mean	<b>1522.9</b>	<b>1521.8</b>	<b>1525.9</b>
		std	3.8	3.2	4.0
EHBSA/WT/3	without	best	1516.0	1520.0	1526.0
		mean	<b>1526.6</b>	<b>1529.2</b>	<b>1534.7</b>
		std	4.6	4.4	4.5
	with	best	1516.0	1516.0	1520.0
		mean	<b>1522.1</b>	<b>1525.2</b>	<b>1528.7</b>
		std	2.3	4.3	4.0
EHBSA/WT/4	without	best	1520.0	1516.0	1525.0
		mean	<b>1528.2</b>	<b>1529.6</b>	<b>1535.9</b>
		std	4.6	5.1	5.1
	with	best	1519.0	1519.0	1521.0
		mean	<b>1525.4</b>	<b>1525.7</b>	<b>1531.2</b>
		std	4.1	3.4	4.4

### 4.2.2 Results on 40-job and 10 machine flow shop problem

Results on the 40-job and 10-machine problem are shown in Table 2. Again, we can confirm clearly that EHBSA/WT shows much better performance, i.e. smaller makespan, than EHBSA/WO in all results in the table. In EHBSA/WO, we can see EHBSA/WO/R shows better performance than EHBSA/WO/T, as in the case of the 20-job and 10-machine problem.

The effectiveness of introducing the TN for EHBSA/WT is also clearly observed in this problem, too. The best value of the *mean* without TN was observed in EHBSA/WT/2 with population size = 120, the value being 2593.6. However, with TN, the value decreases to 2591.9. As was observed in the 20-job and 10-machine problem, the difference is again not so remarkable. However, the important observation is that in all experiences of EHBSA/WT, values of the *mean* with TN always showed better than those without TN. On average, the *mean* with TN is smaller than without TN by 0.34%. This effect can be observed for EHBSA/WO, too.

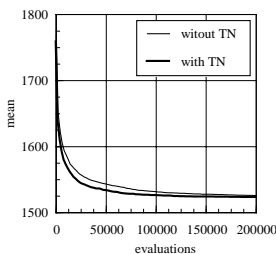
Fig. 7 shows the convergence processes of EHBSA/WT with and without TN for (a) EHBSA/WT/2, (b) EHBSA/WT/3, and (c) EHBSA/WT/4, with population size = 60. From this figure, we can see the values of *mean* with TN always converge faster than those without TN, resulting in significantly smaller values as was observed in Fig 6 on the 20-jobs and 10-machine problem.

### 4.2.3 Considerations

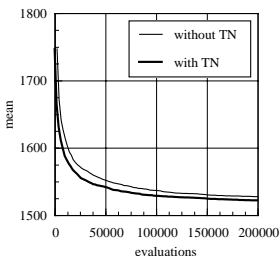
As shown in subsection 4.2.1 and 4.2.2, introducing the TN with EHBSA consistently improved the performance of the EHBSA on flow shop scheduling problems. Now let us confirm the statistical difference of mean values between the models with the TN and without the TN using *t*-test. Values of *t* for EHBSA/WT in Tables 1 and 2 are in the range of [3.2, 9.5] except for EHBSA/WT/2 where  $N = 120$  on the 20-job and 10-machine problem (in this case,  $t = 0.629$ ). Since the value of  $df = 99$ , we can set the level of significance below 0.002.

Since EHM has no explicit information on the absolute position of each node, the TN in EHBSA works as a tag that indicates the initial position in a string, thus as we intended, it improves performance of a problem where in addition to the relative position of each node, the absolute position of each node effects the performance.

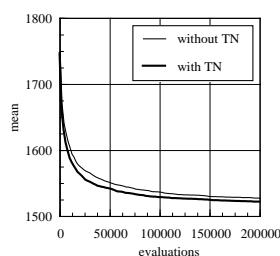
Here, we must note that without using the TN, EHBSA can somehow maintain information about absolute position of each node in a string of the population. That is because, for example in EHBSA/WT, nodes in a new string that are not sampled, inherit the same absolute positions from its template string. Introducing the TN helps the managing of information on absolute position more, and thus increases the performance.



(a) EHBSA/WT/2



(b) EHBSA/WT/3



(c) EHBSA/WT/4

Fig. 6 Convergence process of EHBSA/WT on 20-jobs and 10-machine

Table 2 Results on 40-job and 10-machine flowshop problem (makespan)

EHBSA	TN	Performance Measure	Population Size		
			60	120	240
EHBSA/WO/T	without	best	2696.0	2711.0	2731.0
		mean	<b>2760.8</b>	<b>2773.5</b>	<b>2781.8</b>
		std	23.1	18.8	17.9
	with	best	2600.0	2580.0	2629.0
		mean	<b>2659.8</b>	<b>2670.8</b>	<b>2701.5</b>
		std	34.6	36.6	28.4
EHBSA/WO/R	without	best	2684.0	2690.0	2730.0
		mean	<b>2751.6</b>	<b>2761.6</b>	<b>2773.8</b>
		std	23.0	20.9	17.1
	with	best	2574.0	2590.0	2609.0
		mean	<b>2636.5</b>	<b>2640.5</b>	<b>2658.7</b>
		std	32.4	29.5	20.9
EHBSA/WT/2	without	best	2582.0	2565.0	2553.0
		mean	<b>2610.1</b>	<b>2593.6</b>	<b>2598.6</b>
		std	13.4	12.9	12.9
	with	best	2571.0	2570.0	2568.0
		mean	<b>2600.1</b>	<b>2591.9</b>	<b>2589.2</b>
		std	16.8	14.2	11.9
EHBSA/WT/3	without	best	2590.0	2582.0	2579.0
		mean	<b>2620.0</b>	<b>2607.2</b>	<b>2609.4</b>
		std	13.3	11.7	12.2
	with	best	2578.0	2577.0	2577.0
		mean	<b>2608.5</b>	<b>2599.2</b>	<b>2601.7</b>
		std	12.3	11.5	11.1
EHBSA/WT/4	without	best	2603.0	2594.0	2607.0
		mean	<b>2643.8</b>	<b>2631.9</b>	<b>2633.4</b>
		std	13.1	13.2	13.2
	with	best	2599.0	2599.0	2594.0
		mean	<b>2630.4</b>	<b>2623.5</b>	<b>2624.0</b>
		std	12.9	9.2	12.3

From above discussion, we may expect that the TN works well for traditional two-parent recombination operators. Fig. 8 shows the convergence processes with and without TN in the original order crossover *OX* [Oliver 87] and the enhanced edge recombination operator *eER* [Starkweather 91]. The evolutionary model used was similar to that described in Fig 5 and population size = 480 was used.

From this figure, we can see that introducing the TN is effective for traditional recombination operators, too. Especially, *eER* showed a big difference in performance.

Since the TN is treated as just a normal node in algorithms, there is no special processing necessary in introducing the TN. The difference of computational complexity with and without TN is influenced only by the string length; without TN the length is  $L$ , and with TN the length is  $L+1$ .

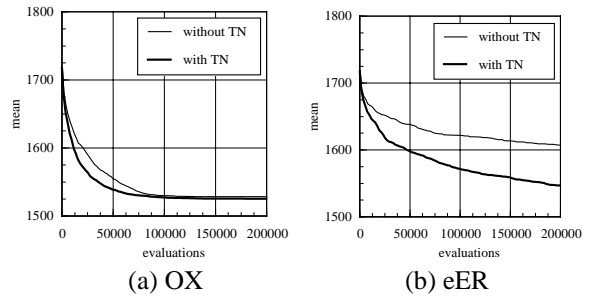


Fig. 8 Convergence processes of (a) OX and (b) eER on 20-jobs and 10-machine

### 5 Conclusions

EHBSAs (edge histogram based sampling algorithms) have been proposed for permutation representation problems based on probabilistic model-building genetic algorithm (PMBGAs) schemes. There are two types of EHBSAs, EHBSA/WT and WHBSA/WO. EHBSA/WT works fairly well on TSPs, symmetrical sequencing problems.

In a previous study, we applied EHBSAs to flow shop scheduling problems. Since in a scheduling problem, each edge is directional, we used an asymmetrical edge histogram matrix from the current population. The results showed EHBSA/WT worked fairly well in flow shop scheduling problems.

In this paper, we explored the effect of introducing a tag node (TN) with EHBSA using 20-jobs and 10-machine, and 40-jobs and 10-machine flow shop problems. The results showed that introducing the TN could improve the performance of EHBSA/WT. We were also able to show that introducing the TN is effective for traditional two-parent recombination operators.

Since introducing the TN does not increase computational complexity, the approach can be applicable

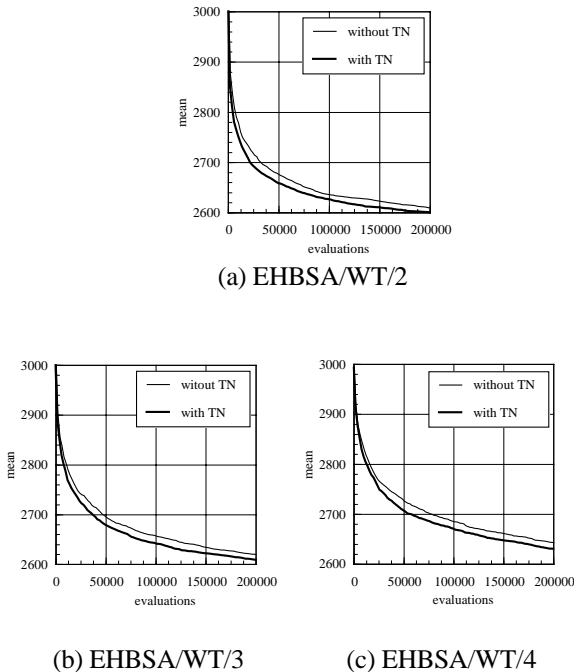


Fig. 7 Convergence process of EHBSA/WT on 40-jobs and 10-machine

to not only EHBSA but also to other permutation-based recombination operators in solving scheduling problems such as flow shop problems. But a detailed study of this remains for future work.

## Acknowledgments

This research is partially supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under Grant-in-Aid for Scientific Research number 13680469, and a grant to RCAST at Doshisha University from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

## References

- [Baluja 94] Baluja, S.: Population-based incremental learning: A method for interacting genetic search based function optimization and coemptive learning, *Tech. Rep. No. CMU-CS-94-163*, Carnegie Mellon University (1994).
- [Baluja 97] Baluja, S. and Davies: Using optimum dependency-trees for combinatorial optimization: learning the structure of the search space, *Tech. Rep. No. CMU-CS-97-107*, Carnegie Mellon University (1997)
- [Bosman 99] Bosman, P. and Thierens, D.: An algorithmic framework for density estimation based evolutionary algorithms, *Tech. Rep. No. UU-CS-1999-46*, Utrecht University (1999).
- [Bosman 00] Bosman, P. and Thierens, D.: Continuous iterated density estimation evolutionary algorithms within the IDEA framework, *Proc. of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference GECCO-2000*, pp.197-200 (2000).
- [Harik 98] Harik, G., Lobo, F. G., and Goldberg, D. E.: The compact genetic algorithm, *Proc. of the Int. Conf. Evolutionary Computation 1998 (ICEC 98)*, pp. 523-528 (1998).
- [Harik 99] Harik, G: Linkage learning via probabilistic modeling in the ECGA, *Technical Report IlliGALReport 99010*, University of Illinois at Urbana-Champaign, Urbana (1999).
- [Larranaga 99] Larranaga, P., Etxeberria, R., Lozano, J.A., and Pena, J.M.: Optimization by learning and simulation of Bayesian and gaussian networks, *University of the Basque Country Technical Report EHU-KZAAIK -4/99* (1999).
- [Larranaga 00] Larranaga, P., Etxeberria, R., Lozano, J. A., and Pena, J. M.: Optimization in continuous domains by learning and simulation of Gaussian networks, *Proc. of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pp. 201-204 (2000).
- [Larranaga 02] Larranaga, P. and Lozano, J. A. (eds): *Estimation of distribution algorithms*, Kluwer Academic Publishers (2002).
- [Mühlenbein 96] Mühlenbein, H and Paa, G.: From recombination of genes to the estimation of distribution I. Binary parameters, *Proc. of the Parallel Problem Solving from Nature - PPSN IV*, pp. 178-187 (1996).
- [Oliver 87] Oliver, I, Smith, D., and Holland, J.: A study of permutation crossover operators on the travel salesman problem, *Proc. 2nd Int. Conf. on Genetic Algorithms*, pp. 224-230 (1987).
- [Pelikan 99] Pelikan, M., Goldberg, D. E., and Lobo, F. G.: A survey of optimization by building and using probabilistic models, *Technical Report IlliGAL Report 99018*, University of Illinois at Urbana-Champaign (1999).
- [Pelikan 00] Pelikan, M., Goldberg, D. E., and Cantu-Paz, E.: Linkage problems, distribution estimate, and Bayesian network, *Evolutionary Computation*, Vol. 8, No. 3, pp. 311-340 (2000).
- [Robles 02] Robles, V., Miguel, P. D., and Larranaga, P.: Solving the traveling salesman problem with EDAs, *Estimation of Distribution Algorithms*, Larranaga, P. and Lozano, J. A. (eds), Kluwer Academic Publishers, Chapter 10, pp. 211-229 (2002).
- [Sebag 98] Sebag, M. and Ducoulombier, A.: Extending population-based incremental learning to continuous search spaces, *Proc. of the Parallel Problem Solving from Nature - PPSN V*, pp. 418-427 (1998).
- [Servet 97] Servet, I. L., Trave-Massuyes, L., and Stern, D.: Telephone network traffic overloading diagnosis and evolutionary computation techniques, *Proc. of the Third European Conference on Artificial Evolution (AE 97)*, pp. 137-144 (1997).
- [Starkweather, 91] Starkweather, T., McDaniel, S., Mathias, K, Whitley, D, and Whitley, C.: A comparison of genetic sequence operators, *Proc of the 4th Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, pp. 69-76 (1991).
- [Tsutsui 01] Tsutsui, S., Pelikan, M., and Goldberg, D. E.: Evolutionary Algorithm using Marginal Histogram Models in Continuous Domain, *Proc. of the 2001 Genetic and Evolutionary Computation Conference Workshop Program*, pp. 230-233 (2001).
- [Tsutsui 02a] Tsutsui, S.: Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram, *Proc. of the 7th Parallel Problem Solving from Nature - PPSN VII*, pp. 224-233 (2002).
- [Tsutsui 02b] Tsutsui, S. and Miki, M.: Solving Flow Shop Scheduling Problems with Probabilistic Model-Building Genetic Algorithms using Edge Histograms, *Proc. of the 4th Asia-Pacific Conference on Simulated Evolution And Learning -SEAL02* (2002).
- [Tsutsui 03] Tsutsui, S., Pelikan, M., and Goldberg, D. E: Using Edge Histogram Models to Solve Permutation Problems with Probabilistic Model-Building Genetic Algorithms, *IlliGAL Report No. 2003022*, University of Illinois at Urbana-Champaign (2003).
- [Whitley 89] Whitley, D., Starkweather, T., and Fuquay, D.: Scheduling problems and traveling salesman problem: The genetic edge recombination operator, *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, Morgan Kaufmann (1989).