Missouri Estimation of Distribution Algorithms Laboratory

Node Histogram vs. Edge Histogram: A Comparison of PMBGAs in Permutation Domains

Shigeyoshi Tsutsui, Martin Pelikan, and David E. Goldberg

MEDAL Report No. 2006009

July 2006

Abstract

Previous papers have proposed an algorithm called the edge histogram sampling algorithm (EHBSA) that models the relative relation between two nodes (edge) of permutation strings of a population within the PMBGA framework for permutation domains. This paper proposes another histogram based model we call the node histogram sampling algorithm (NHBSA). The NHBSA models node frequencies at each absolute position in strings of a population. Sampling methods are similar to that of EHBSA. Performance of NHBSA is compared with that of EHBSA using two types of permutation problems: the FSSP and the quadratic assignment problem (QAP). The results showed that the NHBSA works better than the EHBSA on these problems.

Keywords

Probabilistic model-building genetic algorithms (PMBGAs), estimation of distribution algorithms (EDAs), permutation problems, edge histogram, node histogram, traveling salesman problem, flow shop scheduling problem, quadratic assignment problem.

Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis
One University Blvd., St. Louis, MO 63121
E-mail: medal@cs.umsl.edu
WWW: http://medal.cs.umsl.edu/

Node Histogram vs. Edge Histogram: A Comparison of PMBGAs in Permutation Domains

Shigeyoshi Tsutsui
Dept. of Management and Information Science, Hannan University
5-4-33 Amamihigashi, Matsubara, Osaka 580-5802, Japan
tsutsui@hannan-u.ac.jp

Martin Pelikan
Dept. of Math and Computer Science, University of Missouri at St. Louis
8001 Natural Bridge Rd., St. Louis, MO 63121
pelikan@cs.umsl.edu

David E. Goldberg
Illinois Genetic Algorithms Laboratory and Dept. of General Engineering,
University of Illinois at Urbana-Champaign
117 Transportation Building, 104 S. Mathews Ave., Urbana, IL 61801
deg@illigal.ge.uiuc.edu

Abstract

Previous papers have proposed an algorithm called the edge histogram sampling algorithm (EHBSA) that models the relative relation between two nodes (edge) of permutation strings of a population within the PMBGA framework for permutation domains. This paper proposes another histogram based model we call the node histogram sampling algorithm (NHBSA). The NHBSA models node frequencies at each absolute position in strings of a population. Sampling methods are similar to that of EHBSA. Performance of NHBSA is compared with that of EHBSA using two types of permutation problems: the FSSP and the quadratic assignment problem (QAP). The results showed that the NHBSA works better than the EHBSA on these problems.

Key words: Probabilistic model-building genetic algorithms (PMBGAs), estimation of distribution algorithms (EDAs), permutation problems, edge histogram, node histogram, traveling salesman problem, flow shop scheduling problem, quadratic assignment problem

1. Introduction

GAs should work well for problems that can be decomposed into sub-problems of bounded difficulty [7] 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 [7], [12], [21].

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 [8], [12], [13].

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) [13] or estimation of distribution algorithms (EDAs) [9].

Most work on PMBGAs 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 can be found on PMBGAs in permutation domains [1], [14], and even these studies take an indirect approach of mapping permutation problems to fixed-length vectors of discrete or continuous variables, that in some cases necessitate the use of repair operators to correct invalid permutations.

Previous papers [22], [23] introduced a promising approach to learning and sampling probabilistic models for permutation problems using edge histogram models. We called the approach *edge histogram based sampling algorithms (EHBSA)*. EHBSA can be used to directly encode and sample probability distributions over permutations without requiring a transformation of permutations to another domain or a repair operator. EHBSA was applied to several permutation problems including the travel salesman problem (TSP) [22], [23], the flow shop scheduling problem (FSSP) [25] and the capacitated vehicle routing problem (CVRP) [24]. The results indicated that using edge histogram models provides competitive results on the TSP where relative relation between two nodes is related to performance. However, on the FSSP, where absolute position of each node in a string is related to its performance, the performance of the EHBSA was not so competitive when compared with traditional genetic operators.

In this paper, we propose the node histogram model, another histogram based model within the PMBGA framework for permutation domains. In contrast to the EHBSA which models the relative relation between two nodes (edge) in permutation strings, the node histogram model models node frequencies at each absolute position in strings of a population and takes more direct form of UMDA [9]. Sampling methods are similar to that of EHBSA. In this paper, we call the approach *node histogram based sampling algorithm (NHBSA)*. Performance of NHBSA is compared with that of EHBSA using two types of permutation problems; the FSSP and the quadratic assignment problem (QAP). The results showed that the NHBSA works

better than the EHBSA on these problems.

In the remainder of this paper, a review of the EHBSA is described in Section II, NHBSA is described in Section III, experiments on the performance of NHBSA and analysis are given in Section IV, future work related to this paper is discussed in Section V, and finally, Section VI concludes the paper.

2. A Review of EHBSA

This section reviews how the EHBSA can be used to (1) model promising solutions and (2) generate new solutions by simulating the learned model. The review is presented in extra detail because EHBSA is closely related to the NHBSA.

An edge is a link or connection between two nodes and has important information about the permutation string. The basic idea of the 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 is constructed and new solutions are generated by sampling, based on the EHM. New solutions replace some of the old ones and the process is repeated until the termination criteria are met.

2.1 Developing Edge Histogram Matrix (EHM)

In EHBSA, two types of EHM are defined; symmetrical EHM ($EHM_{(S)}$) and asymmetrical EHM ($EHM_{(A)}$).

2.1.1 A Symmetrical EHM ($EHM_{(S)}$)

A $EHM_{(S)}$ is intended to apply to problems such as a symmetrical TSP where each edge has no direction. In this case, we assume edges $i \rightarrow j$ and $j \rightarrow i$ are identical. If there is an edge $i \rightarrow j$, then we assume the edge $j \rightarrow i$ also exists. Here, i and j are nodes and $i \neq j$.

Let string of k-th individual in population $P(t)$ at generation t represent $s^t_k = \{\pi^t_k(0), \pi^t_k(1), ..., \pi^t_k(L-1)\}$, where $\{\pi^t_k(0), \pi^t_k(1), ..., \pi^t_k(L-1)\}$ is a permutation of (0, 1, ..., L–1) and L is the length of the permutation. Then, a $EHM_{(s)}{}^t$ ($e^t_{i,j}$) (i, j =0, 1, .., L–1) of population $P(t)$ consists of L^2 elements as follows:

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

where N is the population size, $\delta_{i,j}(s^t_k)$ is a delta function defined

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

and ε (ε>0) biases the sampling toward random permutations and it can thus be seen as a form of mutation. To use a comparable pressure toward random permutations for all problems and parameter settings, ε should be proportional to the expected value of $e^t_{i,j}$. Since the average of elements $e^t_{i,j}$ for $i \neq j$ in $\mathrm{EHM}_{(s)}{}^t$ is $2LN / (L^2–L) = 2N / (L–1)$, we get

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

where B_{ratio} ($B_{\mathrm{ratio}} > 0$) or the bias ratio is a constant related to the pressure toward random permutations.

2.1.2 An Asymmetrical EHM ($\mathrm{EHM}_{(A)}$)

An asymmetrical EHM ($\mathrm{EHM}_{(A)}$) is intended to apply to problems such as an asymmetrical TSP and scheduling problems where each edge has direction. In these cases, edges $i \rightarrow j$ and $j \rightarrow i$ are not the same.

A $\mathrm{EHM}_{(A)}{}^t$ ($e^t_{i,j}$) (i, j =0, 1, ..., L–1) of population $P(t)$ consists of L^2 elements as follows:

$$e^t{}_{i,j} = \begin{cases} \sum_{k=1}^{N} \delta_{i,j}(s^t_k) + \varepsilon & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{4}$$

where $\delta_{i,j}(s^t_k)$ is a delta function defined by Eq. 2. The average number of edges of element $e^t_{i,j}$ in $\mathrm{EHM}_{(A)}{}^t$ is $LN/(L^2–L) = N/(L–1)$. So the bias ε for $\mathrm{EHM}_{(A)}{}^t$ is defined as it is in Eq. 3 as follows:

$$\varepsilon = \frac{N}{L-1} B_{\mathrm{ratio}} \tag{5}$$

2.1.3 Tag Node

Since the $\mathrm{EHM}_{(A)}$ described in Subsection *A*. has no explicit information on the absolute positions of nodes in a string, we cannot apply it to problems such as the flow shop scheduling problem in which the absolute position of each node in a string is related to its performance.

The *tag node (TN)* is an approch to introducing information on the absolute position of each node in a string to these problems [25]. 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 $L_{\mathrm{VS}} = L+1$, where L is the length of real string (RS; string without TN). The TN in the VS works as a tag in a permutation string to indicate the first node, i.e., a node which follows the TN is assumed to be the first node in the solution. The TN is *virtual* 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. 1 shows how to obtain RS from VS. In this case, L = 6. We can use any symbol to represent the TN in a VS. However, for implementation convenience, we use an integer number 6 (in general number L) to represent it in a VS. Then, a VS of length L+1 is represented as a permutation of {0, 1, …, L–1, L} corresponding numbers 0, 1, …, L–1 to real nodes and number L to the TN.

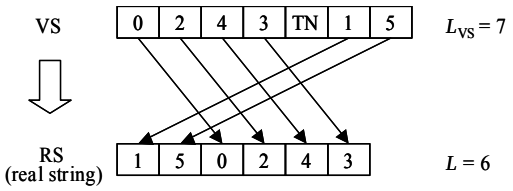

Fig. 1 A Permutation string with the Tag Node

2.2 Sampling Algorithms

There are two sampling algorithms for sampling the EHM^t: (1) the edge histogram based sampling algorithm without template (EHBSA/WO), and (2) 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 the new permutation (denoted by $c[0]$). The sampling continues recursively using a random proportional rule. Let us assume that the last element of the new permutation that we have generated is $c[p]$ (that means that we have generated $p+1$ elements so far). The new element $c[p+1]$ is set to x (we restrict potential values of x so that $x \neq c[i]$ for all $i \in \{0, 1, \ldots, p\}$) with a probability proportional to the element $e^t_{c[i],x}$ of the given edge histogram matrix EHM^t. The sampling continues until the entire permutation has been generated. Fig. 2 shows the schematic description of the EHBSA/WO.

Note that EHBSA/WO is similar to the sampling in Ant Colony Optimization (ACO) [3].

1. Generate a candidate node list $C = \{0, 1, \ldots, L-1\}$.
2. Set the position counter $p \leftarrow 0$.
3. Obtain first node $c[0]$ randomly from $[0, L-1]$ and remove node $c[0]$ from C.
4. Construct a roulette wheel from EHM^t and sample node x with probability $e^t_{c[p],x} / \sum_{j \in C} e^t_{c[p],j}$
5. Set $c[p+1] \leftarrow x$ and remove node x from C.
6. Update the position counter $p \leftarrow p+1$.
7. If $p < L-1$, go to Step 4.
8. Obtain a new individual string $c[]$.

Fig. 2 Schematic description of EHBSA/WO

2.2.2 EHBSA/WT

EHM^t described in *A.* does not consider interactions between edges, which can make the sampling rather

ineffective in practice. EHBSA/WT attempts to improve the sampling by using a template. EHBSA/WT starts by selecting a template individual from the selected population $P(t)$. We use a simple strategy for selecting the template individual and set the template to a random individual from $P(t)$ (Other strategies could be used for selecting templates; for example, the best individual in the current population could be used as a template).

A crucial question when using a template to sample new permutations is how to determine the positions of permutation elements or nodes that are to be copied from the template and the positions of nodes that are going to be generated anew. To ensure robustness across a wide spectrum of problems, it should be advantageous to introduce variation both in the positions of permutation elements that are to be generated anew as well as in the number of these elements.

To choose positions to generate anew, we use a simple method inspired by n-point crossover of simple GAs. In EHBSA/WT/n, the template permutation is first mapped onto a circle so that the last position is followed by the first one. To generate each new individual, $n \geq 2$ positions in the template are selected at random, dividing the template into n segments of variable length. The segment the elements of which are to be generated anew is chosen randomly out of these n segments; all remaining elements are copied directly from the template.

Randomness in generating cut positions and choosing one of these segments introduces variation in segment positions and lengths of elements that are going to be generated anew. The probability distribution of segment lengths can be controlled by n; for n=2, segment lengths are distributed uniformly, but as n grows, shorter segment lengths become dominant. Again, other methods can be used to select positions that are to be generated anew.

Fig. 3 shows an example of EHBSA/WT/3 where 3 cut points are used. In this example, three segments, segment0, segment1, and segment2 are generated by cut points cut[0], cut[1] and cut[2], and segment1 is chosen for sampling nodes. Nodes of the new string in segment0 and segment2 (before cut[1] and after cut[2]) are the same as the nodes 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 in EHBSA/WO.

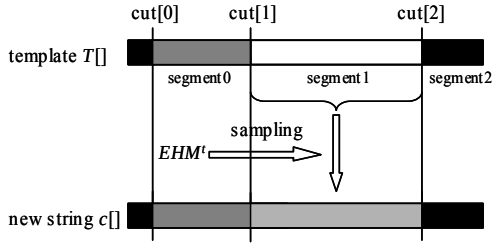

Fig. 3. An example of EHBSA/WT/3

3. Node Histogram Based Sampling Algorithm

In the EHBSA described in Section 2, primary attention is paid to the adjacency relationship between nodes. In contrast to this, in the NHBSA described in this section, primary attention is paid to how each node distributes across string positions in a population $P(t)$. Thus, node distribution in string positions is modeled. Basically, sampling techniques similar to those used for EHBSA can be applied to NHBSA.

3.1 Developing Node Histogram Matrix (NHM)

Let string of k-th individual in population P(t) at generation t represent $s^t_k = \{\pi^t_k(0), \pi^t_k(1), ..., \pi^t_k(L-1)\}$ as in Section II. Then, a node histogram matrix NHM^t $(n^t_{i,j})$ $(i, j = 0, 1, .., L-1)$ of population $P(t)$ consists of L^2 elements as follows:

$$n^t_{i,j} = \sum_{k=1}^{N} \delta_{i,j}(s^t_k) + \varepsilon \tag{6}$$

where N is the population size, $\delta_{i,j}(s^t_k)$ is a delta function defined as

$$\delta_{i,j}(s^t_k) = \begin{cases} 1 & \text{if } \pi^t_k(i) = j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Explicitly saying, integer part of $n^t_{i,j}$ of NHM indicates the total number of node j at (absolute) position i of strings in $P(t)$, i.e., it represents how nodes are distributed in the string positions in a population.

The average number of nodes of element $n^t_{i,j}$ in NHM^t is $LN/L^2 = N/L$. So the bias ε for NHM is defined as in Eqs. 3 and 5 of EHBSA as follows:

$$\varepsilon = N/L \, B_{\mathrm{ratio}} \tag{8}$$

An example of NHM is show with $\mathrm{EHM}_{(\mathrm{A})}$ for the same population $P(t)$ in Fig. 4. Please note that values in diagonal positions of EHM $e^t_{i,i}$ (i = 1, 2, …, L–1) are not used in the sampling algorithm though they are defined 0 in Eqs. 1 and 4 for convenience. In contrast to them, values in diagonal positions of NHM $n^t_{i,i}$ (i = 1, 2, …, L–1) show frequencies of node i at i-th positions in the population and are used in the sampling algorithm without any special treatments.

3.2 Sampling Algorithm

As in EHBSA, we consider two types of sampling algorithms for sampling the node histogram matrix NHM^t: (1) the node histogram based sampling algorithm without template (NHBSA/WO), and (2) the node histogram based sampling algorithm with template (NHBSA/WT).

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

$$\begin{matrix} & \text{node } j \\ \text{position } i & \begin{pmatrix} 1.2 & 2.2 & 0.2 & 1.2 & 1.2 \\ 1.2 & 1.2 & 0.2 & 1.2 & 2.2 \\ 0.2 & 0.2 & 3.2 & 1.2 & 1.2 \\ 0.2 & 2.2 & 1.2 & 2.2 & 0.2 \\ 3.2 & 0.2 & 1.2 & 0.2 & 1.2 \end{pmatrix} \end{matrix} \qquad \begin{matrix} & \text{node } j \\ \text{node } i & \begin{pmatrix} 0 & 3.25 & 0.25 & 2.25 & 0.25 \\ 1.25 & 0 & 2.25 & 0.25 & 1.25 \\ 1.25 & 1.25 & 0 & 2.25 & 1.25 \\ 1.25 & 1.25 & 0.25 & 0 & 3.25 \\ 2.25 & 0.25 & 3.25 & 0.25 & 0 \end{pmatrix} \end{matrix}$$

(1) NHM^t (2) $\mathrm{EHM}^t{}_{(\mathrm{A})}$

Fig. 4 An example of NHM and EHM for P(t) with N = 5, L = 5, and B_{ratio} = 0.2

3.2.1 NHBSA/WO

Let us denote the elements of the permutation to be sampled by $c[i]$ for $i \in \{0, 1, \ldots, L-1\}$ as in EHBSA. NHBSA/WO samples nodes of each position $c[i]$ with random position sequence. More specifically, first we generate random permutation index $r[]$ of $\{0, 1, \ldots, L-1\}$. Each node of $c[r[p]]$ is sampled for p = 0, 1, …, L–1 (with random sequence of $r[p]$). Fig. 5 shows the schematic description of the NHBSA/WO. Please note here that this sampling scheme is very similar to EHBSA/WO in Section II.

1. Generate a random position index permutation $r[]$ of [0, 1, …, L–1]
2. Generate a candidate node list C = {0, 1, …, L–1}
3. Set the position counter $p \leftarrow 0$
4. Sample node x with probability $n^t_{r[p],x} / \sum_{j \in C} n^t_{r[p],j}$.
5. Set $c[r[p]] \leftarrow x$ and remove node x from C.
6. Update the position counter $p \leftarrow p+1$.
7. If $p<L$, go to Step 4, otherwise obtain a new individual string $c[]$.

Fig. 5 Schematic description of NHBSA/WO

3.2.2 NHBSA/WT

NHBSA/WT attempts to improve the sampling by using a template as in EHBSA/WT in Section II. EHBSA/WT starts by selecting a template individual from the selected population $P(t)$. We use the same strategy for selecting the template individual with NHBSA/WT and set the template to a random individual from $P(t)$.

A new individual is created by copying a part of the template directly and generating the remaining positions according to the NHM learned from the population of promising solutions. A big difference between NHBSA/WT and EHBSA/WT is in how they select positions of nodes which are copied from template and

how to sample nodes according to the EHM or NHM. In EHBSA/WT, nodes in continuous positions in segments are copied and then nodes in a segment are sampled according to the EHM (see Fig 3). To obtain the sampling segment, we used n cut points.

In NHBSA/WT, we generate strings, some positions copied and others sampled, through a random sampling sequence. The number of nodes to be sampled l_{WT} is generated by using an n cut point approach similar to EHBSA, as follows. First we apply n cut points to the template string $T[]$ and choose one segment from n segments randomly, from which we obtain l_{WT} as the number of nodes in that segment. Note here we generate segments only to obtain value of l_{WT} (we can obtain the value of by more easy method as discuss in Section V.B.). Then we copy the nodes of $L–l_{WT}$ from template string $T[]$ and sample nodes of l_{WT} according to the NHM. The remaining nodes are then directly sampled, forming a new string.

More specifically, first we generate a random permutation of position index $r[]$ of $\{0, 1, \ldots, L–1\}$ as in NHBSA/WO. Each node of $c[r[p]]$ is copied from template string $T[r[p]]$ for $p = 0, 1, \ldots, L–l_{WT}–1$. Then, each node of $c[r[p]]$ is sampled for $p = L–l_{WT}, L–l_{WT}+1, \ldots, L–1$. The schematic description of the NHBSA/WT is show in Fig. 6. When we use n cut points for NHBSA/WT, we represent it as NHBSA/WT/n as in EHBSA/WT.

1. Generate a random position index permutation $r[]$ of $[0, 1, \ldots, L–1]$
2. Generate a candidate node list $C = \{0, 1, \ldots, L–1\}$
3. Set the position counter $p \leftarrow 0$
4. Sample node x with probability $n^t_{r[p],x} / \sum_{j \in C} n^t_{r[p],j}$.
5. Set $c[r[p]] \leftarrow x$ and remove node x from C.
6. Update the position counter $p \leftarrow p+1$.
7. If $p<L$, go to Step 4, otherwise obtain a new individual string $c[]$.

Fig. 6 Schematic description of NHBSA/WT

4. Experiments

In this section, we explore the performance of NHBSA comparing them with that of EHBSA using two types of problems, flow shop scheduling problem (FSSP) and quadratic assignment problem (QAP). These problems are *NP*-hard and have been approached with many methods, such as simulated annealing [2], taboo search [18], [19], [20], hybrid genetic search [4], ACO [16], [17]. Since our main attention is to compare NHBSA with EHBSA, in this paper we do not compare our algorithm with those above algorithms.

4.1 Test Problems

(1) Flow shop scheduling problem (FSSP)

General assumptions of the FSSP 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 FSSP 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 $\{0, 1, ..., L-1\}$.

(2) Quadratic assignment problem

The QAP is the problem of assigning a set of facilities to a set of locations and can be stated as a problem to find permutations ϕ which minimize

$$f(\phi) = \sum_{0}^{L-1}\sum_{0}^{L-1} a_{ij} b_{\phi(i)\phi(j)}, \tag{9}$$

where $A = (a_{ij})$ and $B = (b_{ij})$ are two $L \times L$ matrices and ϕ is a permutation of $\{0, 1, \ldots, L-1\}$. Matrix A is a distance matrix between locations i and j, and B is the flow between facilities r and s. Thus, the goal of the QAP is to place the facilities on locations in such a way that the sum of the products between flows and distances are minimized. To date instances $L \geq 20$ can generally not be solved to optimality and one has to apply heuristic algorithms which find high quality solutions in a relatively short computation time [16].

4.2 Evolutionary models

This section describes evolutionary models for all methods included in our comparison. All models are based on the steady-state scheme.

(1) Model for NHBSA/WT and EHBSA/WT

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

1. NHM^t (EHM^t) is computed from $P(t)$.
2. A template individual T[] is selected from $P(t)$ randomly.
3. EHBSA/WT (NHBSA/WT) is executed using NHM^t (EHM^t) and T[] to generate a new individual c[].
3. The new individual c[] is evaluated.
4. If c[] is better than T[], then T[] is replaced with c[], otherwise the population remains unchanged, forming $P(t+1)$.

(2) Evolutionary model for NHBSA/WO and EHBSA/WO

The evolutionary model for NHBSA/WO and EHBSA/WO is similar to the model for NHBSA/WT and NHBSA/WT, except that NHBSA/WO and NHBSA/WO do not use a template T[]. The new string c[] is compared to a randomly selected individual i[] in $P(t)$, and if c[] is better than i[], i[] is replaced with c[]; otherwise, the population remains unchanged.

(3) Evolutionary model for two-parent recombination operators

Although more powerful state-of-art operators, such as EAX by Nagata [10], exist, to compare the performance of the proposed methods with the performance of traditional two-parent recombination operators we use OX, PMX, and EER, because they are not bound to specific applications. We use the same steady-state scheme for two-parent recombination operators as well, and after selecting and recombining two parents, one of the two new offspring is incorporated into the original population. In our generational model, two parents are selected from $P(t)$ randomly. No bias is used in this selection. Then we apply a recombination operator to produce one child. This child is compared with its parents. If the child is better than the worst parent, then the parent is replaced with the child.

4.3 Results

4.3.1 TSP

NHBSA aims to solve permutation problems where the absolute position of each node in a string is related to performance. So, it is clear that NHBSA does not work well on the TSP. To make this clear, we ran it on a TSP instance eil51, 51-city TSP. The following control parameters values were used: population size $N = L \times 2$ (L is number of cities), $B_{\text{ration}} = 0.0002$, and the maximum number of evaluations $E_{\max} = L \times 40{,}000$. For WT, cut-point numbers of 2, 3, 4, 5, and 6 were tested. The results are shown in Table 1. Here, *EXCESS* indicates the average

excess from the best known solution over 20 independent runs and *#OPT* indicates the number of runs which found the best know solution.

As expected, NHBSA did not work well for TSP where a relative sequence of nodes in a string is related to the performance. EHBSA/WT worked well in both *EXCESS* and *#OPT* and much better than other EDA approaches in permutation domains such as [14] as reported in [22], [23].

Table 1 Results of TSP (eil51)

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	25.72%	0/20	1.17%	0/20	EER	4.84%	0/20
WT/2	35.29%	0/21	**0.00%**	20/20	PMX	53.76%	0/20
WT/3	31.29%	0/22	0.01%	19/20	OX	8.18%	0/20
WT/4	29.98%	0/20	**0.00%**	20/20			
WT/5	27.08%	0/20	0.02%	18/20			
WT/6	27.01%	0/20	0.12%	10/20			

Optimal value = 426

4.3.2 FSSP

For test problems of FSSP, we use ta031 (50-job×5-machine), ta041 (50×10), ta061 (100×5), and ta071 (100×10) from http://ina2.eivd.ch/collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html.

The same values of control parameters as in *1)* were used: population size $N = L \times 2$ (L is number of jobs), $B_{ration} = 0.0002$, and the maximum number of evaluations $E_{max} = L \times 40{,}000$. For WT, cut-point numbers of 2, 3, 4, 5, and 6 were tested. Results are summarized in Tables 2-5.

Table 2 Results of ta031 (50×5)

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	0.07%	12/20	0.85%	0/20	EER	0.75%	0/20
WT/2	0.01%	19/20	0.06%	13/20	PMX	0.08%	12/20
WT/3	**0.00%**	20/20	0.05%	15/20	OX	0.19%	6/20
WT/4	**0.00%**	20/20	0.07%	12/20			
WT/5	**0.00%**	20/20	0.04%	16/20			
WT/6	**0.00%**	20/20	0.03%	17/20			

Optimal value = 2724

Table 3 Results of ta041 (50×10)

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	3.95%	0/20	5.43%	0/20	EER	8.17%	0/20
WT/2	3.26%	0/20	2.91%	0/20	PMX	4.61%	0/20
WT/3	3.01%	0/20	2.93%	0/20	OX	5.12%	0/20
WT/4	3.15%	0/20	**2.83%**	0/20			
WT/5	3.01%	0/20	3.01%	0/20			
WT/6	2.97%	0/20	3.03%	0/20			

Optimal value = 2991

Table 4 Results of ta061 (100×5)

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	0.03%	5/20	0.81%	0/20	EER	0.63%	0/20
WT/2	0.03%	3/20	0.03%	3/20	PMX	0.03%	6/20
WT/3	0.03%	5/20	0.03%	1/20	OX	0.02%	14/20
WT/4	**0.02%**	7/20	0.03%	4/20			
WT/5	**0.02%**	9/20	0.03%	6/20			
WT/6	**0.02%**	7/20	0.03%	4/20			

Optimal value = 5493

Table 5 Results of ta071 (100×10)

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	1.51%	0/20	4.93%	0/20	EER	7.09%	0/20
WT/2	1.36%	0/20	1.94%	0/20	PMX	1.19%	0/20
WT/3	1.26%	0/20	1.89%	0/20	OX	2.05%	0/20
WT/4	1.23%	0/20	1.92%	0/20			
WT/5	**1.16%**	0/20	1.86%	0/20			
WT/6	1.17%	0/20	1.95%	0/20			

Optimal value = 5770

On ta041 (50×10), the best result was obtained with EHBSA. However the best results were obtained with NHBSA for ta031 (50×5), ta061 (100×5), and ta071 (100×10). For example, *EXCESSes* of ta061 (100×5) with NHBSA are smaller than those with EHBSA for WO, WT/4, WT/5, and WT/6. The *#OPT*s of NHBSA are also greater than those of EHBSA. On ta071, no runs could find the best solution. However, NHBSA always outperformed EHBSA in values of *EXCESS*. Among the traditional crossover operators, PMX showed relatively better performance.

Here we note that both NHBSA/WT and EHBSA/WT always showed much better performance than NHBSA/WO and EHBSA/WO, respectively. These observations are consistent with EHBSA/WT on TSP in 1) and in [22] and [23]. The effectiveness of using the template will be discussed in the following section.

4.3.3 QAP

For test problems of QAP, we used problem instances available at http://www.seas.upenn.edu/qaplib/. According to [20], test instances can be classified into i) randomly generated instances, ii) grid-based distance matrix, iii) real-life instances, and iv) real-life-like instances. In this experiment, we used tai020b, tai25b, tai30b, tai35b, and tai40b which are classified as real-life-like instances.

Except for population size, the same values of control parameters as in FSSP were used. For population size, $N = L \times 10$ (L is the number of locations and is equal to the number of facilities used in Eq. 9), the maximum number of evaluations $E_{max} = L \times 200{,}000$. Results are summarized in Tables 6-9.

From these results, we can confirm that the NHBSA/WT has very good performance on the test problems. It could solve the test problem with the *EXCESS* values of 0.01 ~ 0.16%. These values are smaller than that of EHBSA/WT by more than 10 times. Traditional crossover operators performed similarly or even worse than EHBSA/WT. Again, NHBSA/WO and EHBSA/WO did not show good performance.

Table 6 Results of tai25b

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	0.88%	1/20	1.07%	0/20	EER	3.11%	0/20
WT/2	0.07%	4/20	0.27%	0/20	PMX	0.79%	0/20
WT/3	0.02%	14/20	0.41%	0/20	OX	3.84%	0/20
WT/4	**0.01%**	16/20	0.58%	0/20			
WT/5	0.04%	11/20	0.76%	0/20			
WT/6	0.03%	15/20	0.94%	0/20			

Optimal value = 344355646

Table 7 Results of tai30b

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	10.36%	0/20	2.59%	0/20	EER	1.52%	0/20
WT/2	0.25%	1/20	0.56%	0/20	PMX	1.96%	0/20
WT/3	0.21%	0/20	0.58%	0/20	OX	4.44%	0/20
WT/4	**0.13%**	0/20	0.60%	0/20			
WT/5	0.15%	0/20	0.69%	0/20			
WT/6	0.17%	2/20	0.77%	0/20			

Optimal value = 637117113

Table 8 Results of tai35b

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	5.95%	0/20	3.06%	0/20	EER	3.67%	0/20
WT/2	0.30%	0/20	1.70%	0/20	PMX	2.26%	0/20
WT/3	0.32%	0/20	1.91%	0/20	OX	4.00%	0/20
WT/4	0.25%	2/20	1.68%	0/20			
WT/5	**0.23%**	2/20	2.29%	0/20			
WT/6	0.24%	2/20	2.68%	0/20			

Optimal value = 283315445

Table 9 Results of tai40b

	NHBSA		EHBSA			Crossover	
	EXCESS	*#OPT*	*EXCESS*	#OPT		*EXCESS*	*#OPT*
WO	4.40%	0/20	2.12%	0/20	EER	2.06%	0/20
WT/2	0.32%	0/20	1.99%	0/20	PMX	3.68%	0/20
WT/3	0.25%	0/20	2.75%	0/20	OX	7.99%	0/20
WT/4	0.18%	0/20	3.51%	0/20			
WT/5	**0.16%**	1/20	3.53%	0/20			
WT/6	0.19%	0/20	4.32%	0/20			

Optimal value = 637250948

The performance difference between NHBSA and EHBSA on the FSSP was not as extreme as on the QAP. In regard to this, we can say that the makespan of the FSSP depends on not only absolute position of nodes (jobs) but also the partially relative relation of nodes in a string, i.e., edges. As a result, the EHBSA, which uses the edge histogram model of the population, did work on the FSSP. In contrast to this, performance of the QAP may depend mainly on absolute position of nodes in a string. Accordingly, the NHBSA, which uses the node histogram model of the population works well on the QAP.

4.4 Effectiveness of Using Template

As shown in subsection *C*., NHBSA with template (NHBSA/WT) worked much better than without template (NHBSA/WO) as with EHBSA on TSP. Here, we consider the effect of using the template in NHBSA.

In NHBSA/WT, a new individual is created by copying a part of the template directly and generating the remaining positions according to the NHM learned from the population of promising solutions. Here note that all individuals in $P(t)$ have gone through the process of selection, so their quality can be assumed to be relatively high. As a result, a new individual has a chance to receive a good combination of nodes from the current population and nodes newly sampled from the NHM.

Another important feature of using template is the fact that the number of new nodes in a new string of NHBSA/WT/*n* is only *L*/*n*. This acts to reduce the speed of change of strings in a population preventing premature convergence.

Fig. 7 shows the convergence processes of NHBSA on tai30b. The results were taken as average over 20 runs. In the early stage evolution, NHBSA/WO finds better solutions faster than NHBSA/WT. However, the evolution stops at round 60,000 evaluations due to premature convergence. On the other hand, NHBSA/WT continues evolution. Fig. 8 shows the convergence processes of EHBSA on the same instance. A similar process to NHBSA is observed.

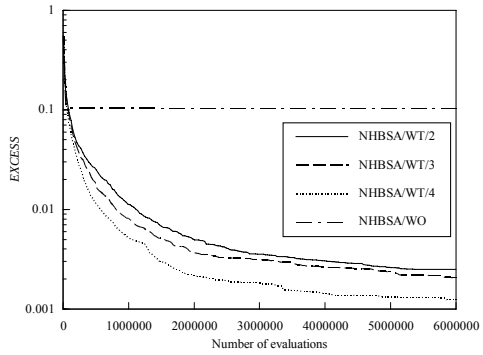

Fig. 7 An example of convergence process of NHBSA

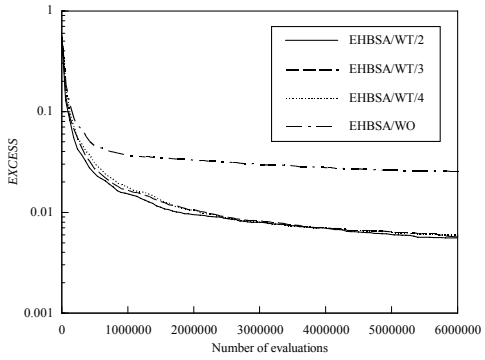

Fig. 8 An example of convergence process of EHBSA

5. Future work

In this section, we discuss major issues to be investigated in relation to this paper.

5.1 Combining the algorithms with local search

Similar to optimization problems and recombination-based optimizers, using local search for improving

solutions locally as the search progresses can significantly improve the performance of the proposed algorithms. So, it is recommended to compare NHBSA and EHBSA with local heuristics.

Table 10 shows a preliminary result on the QAP test problem tai80b, in which we combine NHBSA and EHBSA with robust taboo search (Ro-TS) [19]. Results of Table 10 are obtained as follows: population size N = 40, B_{ration} = 0.0002, and the maximum number of evaluations $E_{\max}$ = 5,000 are used. For each evaluation of a new string, we applied short Ro-TS of length 100 iterations. Although these parameter values are not fine-tuned, NHBSA/WT/2 found the optimal solution 2 times.

A more intensive evaluation of the proposed algorithms with local search remains for future work.

Table 10 Results on tai80b with NHBSA and EHBSA with Ro-TS

	NHBSA		EHBSA	
	EXCESS	*#OPT*	*EXCESS*	*#OPT*
WO	0.75%	1	0.70%	0
WT/2	0.05%	2	0.08%	0

5.2 The distribution of length l_{WT} in EHBSA/WT and NHBSA/WT

In Sections 2 and 3, we have mentioned that the distribution of segment lengths to be generated anew depends on the number of cut points n ($n \geq 2$). The string length L and each cut point are integer values. The length l_{WT} is obtained by randomly applying cut points; the distribution is obtained as below [23].

For simplicity, here we assume L and n to take continuous values and L = 1.0. So, the segment length is in the range of [0, 1] and its density function is obtained as

$$p(l) = (n-1)(1-l)^{n-2}, \quad 0 \leq l \leq 1. \tag{10}$$

The distribution of $f(l)$ is show in Fig. 9. As can be seen from the figure, the probability distribution of l_{WT} can be controlled by n; for n=2, segment lengths are distributed uniformly, but as n grows, shorter lengths of l_{WT} become dominant.

Then, l_{WT} is obtained as $L \times x$ where x is a random number from Eq. 10. Thus, we can obtain l_{WT} from Eq. 10 directly and more effectively. For a given n, the average of l is obtained as

$$\bar{l} = \int_0^1 l \cdot p(l) \cdot dl = 1/n. \tag{11}$$

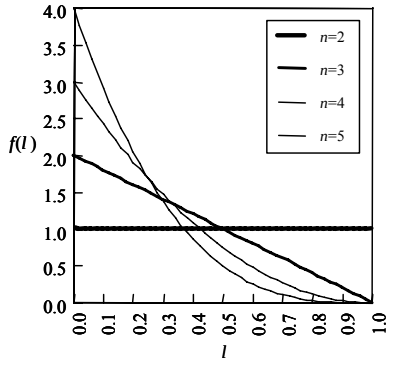

Fig. 9. Probability density function $p(l)$

Let P_T be the rate of number of nodes to be sampled to the string length L. Then, P_T equals $1/n$ for $n \geq 2$. Here, P_T is restricted to values of 0.5, 0.333.., 0.25, and so on. However, if we extend Eq. 10 so that we can assign real number values for n ($n \geq 2$), then we can generate l_{WT} more flexibly. This can be carried out by simply replacing n with $1/P_T$ as

$$p(l) == \left(\frac{1}{P_T} - 1\right)(1-l)^{\frac{1-2P_T}{P_T}}, \quad 0 < P_T \leq 0.5. \tag{12}$$

With Eq. 12, we can generate l_{WT} with any value of P_T between [0, 0.5]. For $0.5 < P_T < 1$, we extend the above logic as follows. First we consider distribution of $l' = 1-l$ and $P'_T = 1-P_T$ in Eq. 12. Then we can obtain the following equation for $0.5 < P_T < 1$.

$$f_T(l) = \frac{P_T}{1-P_T}(l)^{\frac{2P_T-1}{1-P_T}}, \quad 0.5 < P_T < 1. \tag{13}$$

Fig. 4 shows the density distribution function $f(l)$ for P_T values of 0.2, 0.3, 0.5, 0.7, and 0.8.

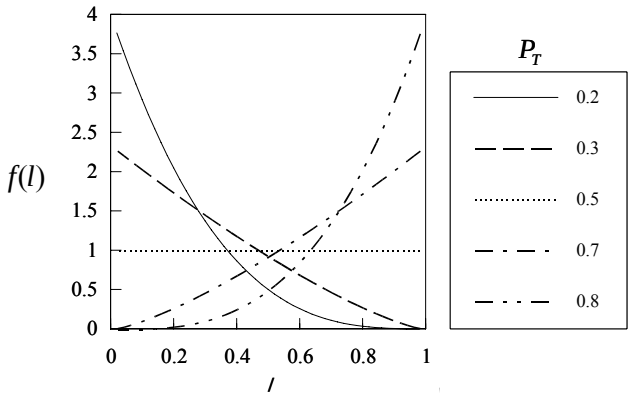

Fig. 4 Probability density function $f_T(l)$ for P_T = 0.2, 0.3, 0.5, 0.7, and 0.8

We can also use other distributions for number of nodes to be sampled, such as the binomial distribution. To test with these distributions remains for future work.

5.3 Applying linkage learning

NHBSA proposed in this paper is based on a marginal distribution model that considers every position in a permutation independently; NHBSA can thus be seen a variant of UMDA algorithm [XXX] for permutation problems. However, challenging permutation problems can be expected to contain interactions between different variables and for such problems considering each position independently may lead to a highly ineffective exploration of the search space. Examples of such difficult problems are deceptive ordering problems [XXX]. Extending NHBSA to incorporate interactions between different permutation elements thus represents an important challenge for future research in this area.

6. Conclusion

In this paper, NHBSA, which uses the node histogram model of a population, was proposed. The NHBSA was compared with EHBSA, which uses the edge histogram model of a population, using the TSP, FSSP, and QAP.

The results showed that performance of the algorithms depends on problem types. For the problems where a relative sequence of nodes in a string is related to the performance, NHBSA did not work well and the advantage of using EHBSA is clear. On the contrary to this, for problems where an absolute position of each node in a string is related to the performance, NHBSA worked better than EHBSA and the advantage of using NHBSA is clear.

Future work for the proposed algorithm was also discussed.

References

[1] P. A. N. Bosman and D. Thierens, "Permutation Optimization by Iterated Estimation of Random Keys Marginal Product Factorizations", *Proc of the Parallel Problem Solving From Nature*, 2002, pp. 331-340.

[2] D. T. Connolly, An improved annealing scheme for the QAP", *European Journal of Operations Research*, 1990, Vol. 46, pp. 93–100.

[3] 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*, 1996, Vol. 26, No. 1, pp. 29-41.

[4] C. Fleurent and J. A. Ferland, “Genetic hybrids for the quadratic assignment problem”, in *Quadratic assignment and related problems* , P.M. Pardalos and H.Wolkowicz (Eds), Vol. 16 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, 1994, American Mathematical Society, pp. 173-187.

[5] L.-M. Gambardella, É D. Taillard, and M. Dorigo, “Ant Colonies for the Quadratic Assignment Problems”, *Journal of the Operational Research Society*, 1999, Vol. 50, pp. 167-176.

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

[7] D. E. Goldberg, *The design of innovation: Lessons from and for competent genetic algorithms*, Vol. 7 of Genetic and Evolutionary Computation, Kluwer, 2002.

[8] P. Larrañaga and J. A. Lozano (Eds.), *Estimation of distribution algorithms*, 2002, Boston, Kluwer.

[9] H. Mühlenbein and G. Paaß, “From recombination of genes to the estimation of distributions I. Binary parameters”, *Proc of the Parallel Problem Solving from Nature*, 1996, pp. 178-187.

[10] Y. Nagata and S. Kobayashi., “Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem”, *Proc. of the 7th Int. Conf. on Genetic Algorithms*, 1997, pp. 450–457.

[11] I. Oliver, D. Smith, J. Holland, “A study of permutation crossover operators on the travel salesman problem”, *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, 1987, pp. 224-230.

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

[13] M. Pelikan, D. E. Goldberg, and F. Lobo, “A survey of optimization by building and using probabilistic models”, *Computational Optimization and Applications*, 2002, Vol. 21, No. 1, pp. 5-20. Also IlliGAL Report No. 99018.

[14] V. Robles, P. D. Miguel, and P. Larrañaga, “Solving the traveling salesman problem with EDAs”, in P. Larranaga and J. A. Lozano (Eds.), *Estimation of Distribution Algorithms*, 2002, pp. 211-229, Kluwer.

[15] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, “A comparison of genetic sequence operators”, *Proc. of the 4th Int. Conf. on Genetic Algorithms*, 1991, pp.

69-76, Morgan Kaufmann.

[16]T. Stutzle and H. Hoos, “Max-Min Ant System, Future Generation Computer Systems”, 2000, Vol.16, No. 9, pp.889-914.

[17]T. Stutzle, “An ant approach to the flow shop problem”, *Proc. of the 6th European Congress on Intelligent Techniques and Soft Computing* (EUFIT’98), 1998, Vol. 3, pp. 1560-1564.

[18]É D. Taillard, “Some efficient heuristic methods for the flow shop sequencing problem”, *European Journal of Operational Research*, 1990, Vol. 47, No. 1, pp. 65-74.

[19]É D. Taillard, “Robust taboo search for the quadratic assignment problem”, *Parallel Computing*, 1991, Vol. 17, pp. 443–455.

[20]É D. Taillard, “Comparison of iterative searches for the quadratic assignment problem”, Location Science, 1995, Vol. 3, pp. 87–105.

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

[22]S. Tsutsui, “Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram”, *Proc. of the 7th Int. Conf. on Parallel Problem Solving from Nature (PPSN VII)*, 2002, pp. 224-233, Springer-Velag.

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

[24]S. Tsutsui and G. Wilson, “Solving Capacitated Vehicle Routing Problems Using Edge Histogram Based Sampling Algorithms”, *Proc. of the 2004 Congress on Evolutionary Computation*, pp. 1150-1157.

[25] 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*, K. C. Tan, M. H. Lim, X. Yao, and L. Wang (Eds), Advances in Natural Computation Series, Chapter 13, 2004, World Scientific.