Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Effect of Using Partial Solutions in Edge Histogram Sampling Algorithms with Different Local Searches

Shigeyoshi Tsutsui
Department of Management Information Science
Hannan University
Matsubara, Japan
tsutsui@hannan-u.ac.jp

***Abstract*—In previous study, we have proposed EHBSA within the EDA framework for permutation domains, and showed better performance than traditional GAs. The important feature of EHBSA is to use partial solutions from previous generations. In this paper, we analyze the effectiveness of using partial solutions using a wide range of problem sizes, without local search, and incorporating two types of local search. One of the most important finding in this paper is that we were able to confirm that using partial solutions is effective for all cases in which we use no local search, 3-OPT local search, and Lin-Kernighan (LK) local search. Future work for this research is also discussed.**

***Index Terms*—Genetic Algorithm, EHBSA, *e*EHBSA, EDA, partial solutions, combining local search, 3-OPT, Lin-Kernighan heuristic, traveling salesman problem**

I. INTRODUCTION

As a new evolutionary computation scheme, there has been a growing interest in developing algorithms based on probabilistic models. 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 more effective mixing capability than recombination operators in traditional GAs.

These algorithms are called *estimation of distribution algorithms (EDAs)* [1]. Many studies on EDAs have been performed in discrete (mainly binary) domains and there are several attempts to apply EDAs in continuous domains. However, a few studies on EDAs in permutation representation domains are found.

In a previous study [2], [3], we proposed an approach of EDAs in a permutation representation domain, and compared its performance with traditional recombination operators. In this approach, we develop an *edge histogram matrix* (EHM) from the current population, where an *edge* is a link between two nodes in a string. We then sample nodes of a new string according to the edge histogram matrix. We call this method the *edge histogram based sampling algorithm* (EHBSA). We tested the algorithm in the Traveling Salesman Problem (TSP), a typical, well-known optimization problem in permutation representation domains using relatively small size instances in TSPLIB [4]. In the study, we did not apply any local search. The results showed EHBS worked fairly well on the test problems compared with traditional GAs with crossover operators such as OX [5], PMX [6] and eER [7]. Further, we found it works well with small population size. The approach was applied to flow shop scheduling in [8], and capacitated vehicle routing problems in [9]. These results also showed better convergence than traditional GAs with OX, PMX and eER crossover operators. For comparisons to traditional GAs, please refer to these references.

The most important feature of EHBSA is the method of creating new solutions. In creating a new solution, we generate its permutation string not only from the current EHM but we also incorporate into it a part of a permutation string from the population of the previous generation, which we call partial solutions. This method is somewhat similar to crossover operations between existing solutions and solutions from EHM. Using partial solutions greatly reduces premature convergence and effectively incorporates previously found partial solutions in new solutions.

An important parameter of EHBSA is the ratio which determines the number of nodes obtained by sampling EHM. In this paper, we use a revised EHBSA, which we refer it to the enhanced EHBSA (*e*EHBSA) here, and analize the effect of using partial solutions on performance using various sizes of TSP instances with different types of local search. For problems of tens cities, we test *e*EHBSA without local search. For problems in the hundreds of cities, we combine *e*EHBSA and 3-OPT local search, a medium strength search. Finally, for problems with cities in the thousands, we combine *e*EHBSA and the Lin-Kernighan heuristic [10], which is considered the most powerful heuristic in solving TSP. Performing these three types of experiments, we confirm the effectiveness of using partial solutions in *e*EHBSA for various types of heuristics.

In the remainder of this paper, first we give, in Section II, a brief review of the estimation of distribution algorithms (EDAs). In Section III, *e*EHBSA is described. Then, In Section IV, an empirical analysis of the effectiveness of using partial solutions in *e*EHBSA is performed. Finally, Section V concludes this paper.

II. A BRIEF OVERVIEW OF EDAS

GAs should work well for problems that can be decomposed into sub-problems of bounded difficulty [11] 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 [12], [11], [13]. 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 [14], [15], [13], [16]. 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 estimation of distribution algorithms (EDAs) [1].

EDAs evolve a population of candidate solutions to the given problem by building and sampling a probabilistic model of promising solutions. EDAs start with a random population of candidate solutions (individuals). Each iteration of EDAs starts by selecting better individuals from the current population. Next, the probability distribution of the selected population of individuals is estimated. New individuals are then generated according to this estimate, forming the population of candidate solutions for the next generation. The process is repeated until the termination conditions are satisfied.

Most work on EDAs focuses on parameter optimization for problems defined over fixed-length vectors of discrete or real-valued variables [13], [15], but only few studies can be found that focus on optimization of problems with solutions represented by permutations. Bosman and Thierens [17], [18] use random keys to represent permutations together with a marginal product factorization to estimate the continuous distribution of random keys. Robles, et al. also use random keys to enable the use of EDAs from continuous parameter optimization in solving permutation problems [19].

III. Enhanced EHBSA (eEHBSA)

Here, we describe eEHBSA and indicate when the method diverges from the previous model. These modifications were implemented to make the parameter setting related to using partial solutions easier and increase the efficiency of the generational cycle.

A. Edge Histogram Matrix

An $edge$ is a link or connection between two nodes in a permutation string. The basic idea of the edge histogram based sampling algorithm (EHBSA) is to use the edge distribution 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 can be then selected using any popular selection scheme.

An *edge histogram matrix* (EHM) for the selected solutions is constructed and new solutions are generated by EHM

$$\begin{aligned} s^t_1 &= (\,0,\ 1,\ 2,\ 3,\ 4\,)\\ s^t_2 &= (\,1,\ 3,\ 4,\ 2,\ 0\,)\\ s^t_3 &= (\,3,\ 4,\ 2,\ 1,\ 0\,)\\ s^t_4 &= (\,4,\ 0,\ 3,\ 1,\ 2\,)\\ s^t_5 &= (\,2,\ 1,\ 3,\ 4,\ 0\,)\end{aligned} \qquad \begin{pmatrix} 0 & 3.1 & 2.1 & 2.1 & 3.1\\ 3.1 & 0 & 4.1 & 3.1 & 0.1\\ 2.1 & 4.1 & 0 & 1.1 & 3.1\\ 2.1 & 3.1 & 1.1 & 0 & 4.1\\ 3.1 & 0.1 & 3.1 & 4.1 & 0 \end{pmatrix}$$

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

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

sampling. New solutions replace some of the old ones and the process is repeated until the termination criteria are met.

An EHM is $L \times L$ matrix, where L is the problem size. An example of EHM at t, $EHM^t = (e^t_{i,j})$, is shown in Fig. 1. The integer value of each $(e^t_{i,j})$ represents number of edges from node i to node j ($i \rightarrow j$) in the population. Note here, in a symmetrical permutation problem such as symmetrical TSP, an edge $i \rightarrow j$ and an edge $j \rightarrow i$ are considered as the same in EHM. In an asymmetrical permutation problem such as a scheduling problem, both have different positions in EHM. The fractional value (ε) represents minimum value to give a bias to control pressure in sampling nodes and is given as [2]

$$\varepsilon = \begin{cases} \frac{2N}{L-1} B_{ratio} \text{ for symmetric} \\ \frac{N}{L-1} B_{ratio} \text{ for asymmteric} \end{cases} \tag{1}$$

where B_{ratio} ($B_{ratio} > 0$) 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 bigger value of B_{ratio} will allow infrequent addition of new edges.

B. Revised Sampling Method

In eEHBSA (EHBSA), we call an individual, from which a partial solution is obtained for generating a new string, a template. As shown in Fig. 2, a new individual c[] is generated by sampling a number of nodes in positions (loci) within the template string while nodes in other positions remain unchanged.

A crucial question when we create a new solution c[] is how to determine which part of the partial solution the c[] will borrow from the template. To ensure robustness across a wide spectrum of problems, it is advantageous to introduce variation both in the portion and the number of nodes of the partial solution that is borrowed from the template. First we choose the starting node position of the partial solution randomly. Thereafter, the number of nodes of the partial solution must be determined.

Let us represent the number of nodes that are constructed based on EHM by l_s. Then, l_c, the number of nodes of the partial solution, which c[] borrows from the template, is $l_c = L–l_s$. Here, let us introduce a control parameter γ which can define $E(l_s)$ (the average of l_s) by $E(l_s) = L \times \gamma$. To determine the sampling portion in a string, in original EHBSA we used the n cut-point approach. However with this approach, $E(l_s)$ is L/2 L/3, $\cdots$ for n= 2, 3, 4, $\cdots$, and, γ corresponds to

$1/n$, i.e., γ can take only the values of 0.5, 0.333, 0.25, $\cdots$. corresponding to n= 2, 3, 4, $cdots$. In the current research in this paper, we extend this elementary method to a more flexible technique which allows for γ taking values in the rage [0.0, 1.0].

The probability density function of l_s with the c cut-point appraoch is [3]:

$$f_s(l) = \frac{(c-1)}{n}\left(1-\frac{l}{n}\right)^{c-2}, \ 0 < l < n, \ c \geq 2. \quad (2)$$

Here, we extend c so that it can take a continuous value (c $\geq$2). Then, we can obtain a generalized $f_s(l)$ by setting c = 1/γ in Eq. 2 The probability density function for l_s in eEHBSA is obtained:

$$f(l_s) = \begin{cases} \frac{1-\gamma}{L\gamma}\left(1-\frac{l_s}{L}\right)^{\frac{1-2\gamma}{\gamma}} & \text{for } \gamma \in (0,\ 0.5] \\ \frac{\gamma}{L(1-\gamma)}\left(\frac{l_s}{L}\right)^{\frac{2\gamma-1}{1-\gamma}} & \text{for } \gamma \in (0,\ 1] \end{cases} \quad (3)$$

as in our previous study on the cunning Ant System (cAS) [20]. As seen in Fig. 2, for a given γ value, l_s is generated according to Eq. (3), and c_{top}, the first position of partial solution for c[] to borrow from the template, is sampled randomly. Then, the partial solution of length $l_c = L - l_s$, and which starts from c_{top} is copied to c[] from the template. Then the remaining sequence of nodes of length l_s in c[] is sampled according to EHM probabilistically as follows. Let i be a node in a position of a new string c[], then node j for next position is determined by selecting edge $i \rightarrow j$ according to the following probability $p^k_{ij}(t)$:

$$p^k_{i,j}(t) = \begin{cases} \frac{e^t_{i,j}}{\sum\limits_{s \in F_k(i)} e^t_{i,s}} & \text{if } j \in F_k(i) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $F_k(i)$ is the set of nodes that remain to be selected in string c[] connected node i to make the solution feasible. To save sampling time, eEHBSA uses the textitcandidate list [21] with candidate size (C_{size}) of 20. Using the candidate list, the computation complex for getting one solution according to Eq. (4) should nearly be $O(n)$.

Please note here that if we use $\gamma = 1$, then it means we generate solutions only from current EHM without using template. In EHBSA, we called EHBSA with $\gamma = 1$ EHBSA/WO (EHBSA without template) and EHBSA with $0 < \gamma < 1$ EHBSA/WT (EHBSA with template).

C. Revised Evolutionary Mode

In the evolutionary model of original EHBSA, we used a generational model in which only one new solution is generated in one generation. In eEHBSA, we generate N solutions in one generation to efficient generational iteration as shown in Fig. 3. Here, N represents population size. Let $P(t)$ represent population at generation t. The population $P(t+1)$ is produced as follows: For each individual I_i in population $P(t)(i = 1, 2, \ldots, N)$, we generate new solutions I'_i using I_i as their templates. Then, we compare each pair (I_i, I'_i). If I'_i is better than I_i, I_i is replaced with I'_i, otherwise I_i remains in the population, and thus the next population $P(t+1)$ is formed.

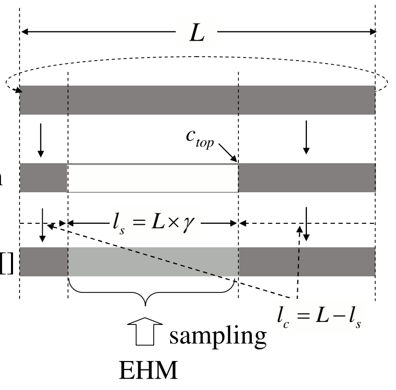

Fig. 2. Generation a new solution using partial solution

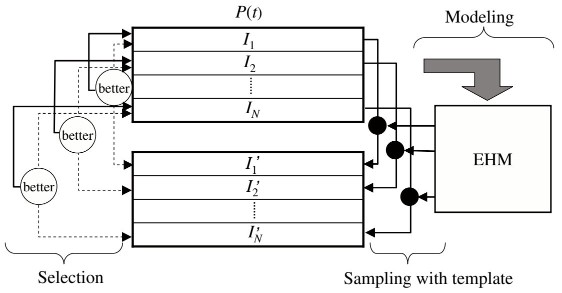

Fig. 3. Revised Evolutionary Model

D. Combining eEHBSA and Local Searches

For problems comprising hundreds of cities, we applied 3-OPT local search. Before we evaluate each individual, we improve it by 3-OPT local search.

For problems comprising thousands of cities, we applied Lin-Kernighan algorithm (LK) [22]. The implementation of LK is complex compared with 3-OPT heuristics. There are many variant implementations for LK. An important, and widely adopted scheme is the repeated use of the basic LK algorithm. The scheme is referred to as Chained Lin-Kernighan [22], or Iterated Lin-Kernighan [23]. In addition to the basic LK, Chained LK repeatedly utilizes a method for perturbing a given tour which is called $kick$.

We used Chained LK called *Concorde TSP solver* (Concorde) developed by D. Applegate et. al, which is available for research purposes at [24]. Current eEHBSA is written in JAVA and Concorde is written in C, so we combined it with eEHBSA using Java Native Interface (JNI). For each solution, we applied n iterations of Chained LK with *random-walk kicks*, which is reported to have the best performing $kick$ [22].

TABLE I
COMPARISON BETWEEN eEHBSA AND CROSSOVER OPERATORS

Instance	Operator	Population Size $N = k*n$							
		k=2		k=10		k=20		k=30	
		#OPT	T_{avg} (sec)	#OPT	T_{avg} (sec)	#OPT	T_{avg} (sec)	#OPT	T_{avg} (sec)
oliver30	eEHBSA	**20**	**0.08**						
	OX	1	0.02	11	0.06	18	0.14	19	0.18
	PMX	0	-	0	-	0	-	2	0.10
	eER	0	-	15	0.29	20	0.55	20	0.84
gr48	eEHBSA	**20**	**0.60**						
	OX	0	-	0	-	1	0.80	4	1.08
	PMX	0	-	0	-	0	-	0	-
	eER	0	-	8	3.05	14	5.68	19	8.48
berlin52	eEHBSA	**20**	**0.57**						
	OX	0	-	2	0.58	6	1.19	6	1.75
	PMX		-	0	-	0	-	0	-
	eER	0	-	17	3.94	20	7.46	20	10.70
pr76	eEHBSA	**20**	**3.71**						
	OX	0	-	0	-	0	-	0	-
	PMX	0	-	0	-	0	-	0	-
	eER	0		1	20.95	8	40.69	10	61.64

IV. EXPERIMENTS

A. Experimental Conditions

The machine we used was Intel Intel® Core^{TM} i7 965 (3.2 GHz) Processor, 6GB main memory, and 32-bit WindowsXP. The following instances in TSPLIB [4], which range in tens and thousands of cities, were used:
(1) Class I instances (without local search): The instances are oliver30, gr48, berlin52, and pr76. Population size is set to problem size $n \times 2$. Executions were terminated when the number of solution constructions reached 20000$\times n$ or their optimal solutions were found.

(2) Class II instances (with 3-OPT heuristic): The instances are lin318, pcb442, att532, and rat783. Population size is set to problem size $n/15$. Executions were terminated when the number of solution constructions reached 1000$\times n$ or their optimal solutions were found.
(3) Class III instances (with LK heuristic): The instances are fl3795 and rl5934. Population size is set to 4. Executions were terminated when the number of solution constructions reached $n/10$ or their optimal solutions were found.

For all experiments, we set B_{ratio} to 0.005 and 20 runs were performed. We evaluated the algorithms by measuring their $\#OPT$ (number of runs in which the algorithm succeeded in finding the global optimum), T_{avg} (mean time in second to find the global optimum in those runs where it did find the optimum).

B. Basic Performace of eEHBSA

TABLE I compares the eEHBSA having $\gamma = 0.3$ with traditional crossover operators, i.e., OX, PMX, and eER. For fair comparison, we designed a genera-tonal model for two-parent recombination operators as similar as possible to that of the eEHBSA. From this table, we can confirm that eEHBSA shows smaller values of T_{avg} than the traditional crossover operators with small size of population.

C. Effect of Using Partial Solutions

Now, we will see the effect of using partial solutions. Recall that γ specifies the proportion of nodes to be generated using EHM (the remaining nodes are taken from the template). To investigate the effect of using partial solutions on the performance of eEHBSA, we examined the performance of eEHBSA for different values of $\gamma \in [0.1, 1]$. γ values were varied from 0.1 to 1 with step 0.1.

1) Results of Class I Instances: The results of $\#OPT$ and T_{ave} for oliver30, gr48, berlin52, and pr76 are shown in Fig. 4. When we run eEHBSA without local seach for this class of instances, the results showed $\#OPT = 19 or 20$ for $\gamma \in$ [0.2 0.5] showing smaller T_{avg} than other values of γ.

On oliver30, $\#OPT = 20$ for $\gamma \in [0.1, 0.6]$. The best T_{avg} was obtained for γ=0.3 with $T_{avg} = 0.08$ seconds. We can see that $\#OPT \ll 20$ and values of T_{avg} are much bigger for $\gamma > 0.6$ than those for $\gamma \leq 0.6$. On gr48, berlin52 and pr76, results are similar to that of oliver30. Promising γ value range in [0.2, 0.5], [0.2, 0.6], and [0.2, 0.5] for gr48, berlin52, and pr76, respectively. On all of these class instances, minimum T_{avg} is obtained for $\gamma = 0.3$.

2) Results of Class II Instances: When we combined EHBSA with 3-OPT for this class of instances, the results showed $\#OPT = 20$ for many cases of γ values. Thus, we will see the effect of γ by $\#OPT$ and T_{ave}. The results of $\#OPT$ and T_{ave} for lin318, pcb442, att532, and rat783 are shown in Fig. 5. From these results, we can see a clear effect of variations of γ in cases where we combine eEHBSA with 3-OPT heuristic. We find that even when combined with 3-OPT heuristic, γ remains to be an important parameter of the eEHBSA.

On lin318, the optimal solution was found for all γ values. The best T_{avg} was obtained for γ=0.4 with $T_{avg} = 2.0$ seconds and the worst T_{avg} was for $\gamma = 1$ with $T_{avg} = 11.2$ seconds. We can see that T_{avg} is much bigger for $\gamma \geq 0.8$ than for $\gamma \leq 0.7$. On pr439, att532, and rat783, $\#OPT = 20$ was not obtained for $\gamma = 0.8 and 1$. On these three instances, minimum T_{avg} is obtained for $\gamma = 0.2$.

3) Results of Class III Instances: For Class III instances, in which a stronger local search than Class II was used, we can see that the effect of γ by $\#OPT$ and T_{ave} is similar to the effect that it had on Class II instances. The results of $\#OPT$ and T_{ave} for fl3795 and rl5934 are shown in Figs. 9 and 10, respectively. On fl3795, the best T_{avg} was obtained for γ=0.4 with $T_{avg} = 23.1$ seconds. We can see that T_{avg} is much bigger for $\gamma \geq 0.8$ than for $\gamma \leq 0.6$. On rl5934, minimum T_{avg} is obtained for $\gamma = 0.2$ with $T_{avg} = 682.0$ seconds.

V. CONCLUSIONS

In our previous papers, we have proposed EHBSA within the EDA framework for permutation domains, and showed better performance than traditional GA with typical crossover operators. The important feature of EHBSA is to use partial

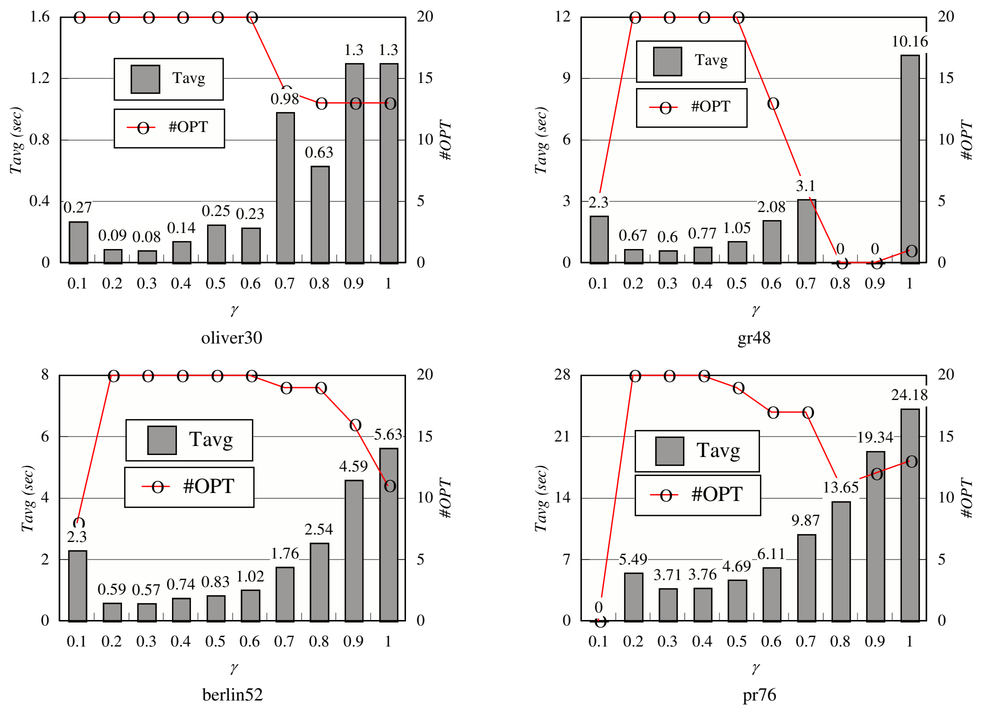

Fig. 4. Results of Class I instances (without local search)

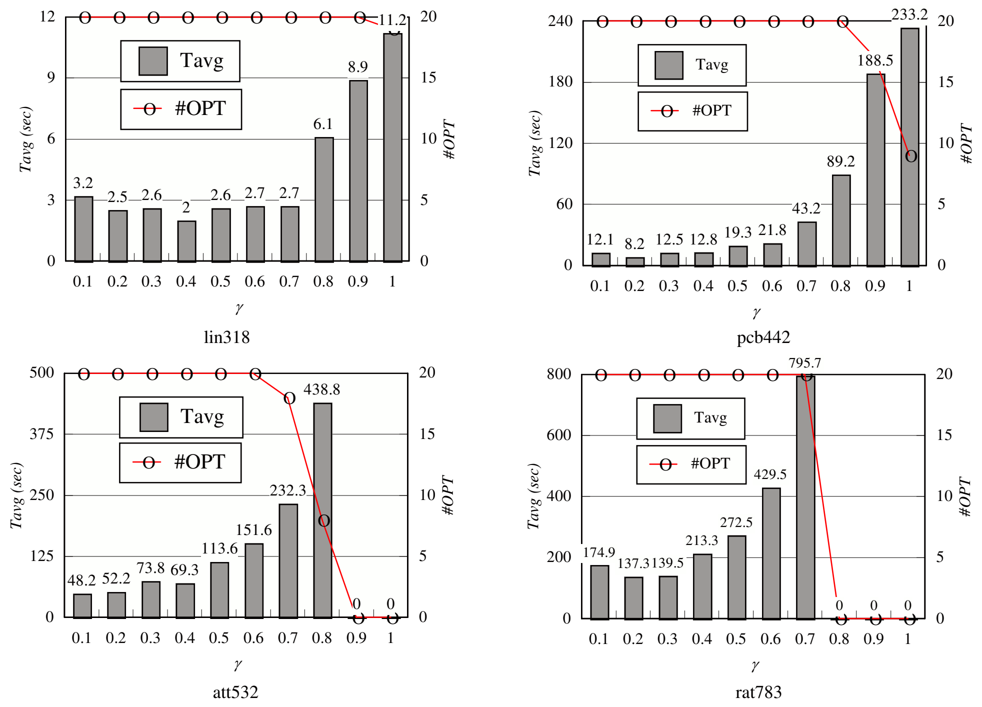

Fig. 5. Results of Class II instances (with 3-OPT)

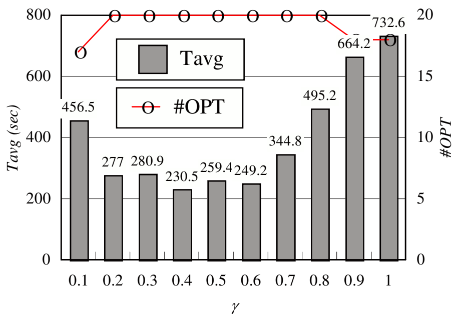

Fig. 6. Results on fl3795 with LK

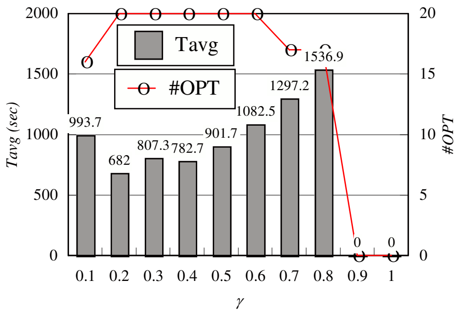

Fig. 7. Results on rl5934 with LK

solutions from previous generations. In this paper, we analyze the effectiveness of using partial solutions using a wide range of problem sizes, without local search, and incorporating two types of local search.

In this paper, we analized the effect of using partial solutions on performance of enhaced EHBSA (*e*EHBSA) using various sizes of TSP instances with different types of local search. One of the most important finding in this paper is that we were able to confirm that using small γ value ([0.2, 0.4]) is effective for all cases in which we use no local search, 3-OPT local search, and LK local search. Here, smaller value of γ means that when a new solution is created we use a larger section of the partial solution and a smaller sampling of the NHM.

Although we could confirm the effectiveness of using partial solutions through experimental analysis, extensive theoretical analysis on the effectiveness of using partial solutions is must be done. One of the approaches for theoretical analysis is to perform entropy analysis of the population and EHM. But this study is remains for future work. Applying analysis such as this to other permutation problems, such as scheduling problems, also remains for future work.

Acknowledgment

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 19500199.

References

[1] H. Mühlenbein and G. Paass, "From recombination of genes to the estimation of distributions I. Binary parameters," *Proceedings of the 1996 Parallel Problem Solving from Nature*, pp. 178–187, 1996.

[2] S. Tsutsui, "Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram," *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII)*, pp. 224–233, 2002.

[3] S. Tsutsui, M. Pelikan, and D. Goldberg, "Using edge histogram models to solve permutation problems with probabilistic model-building genetic algorithms," *IlliGAL Report No. 2003022, University of Illinois*, 2003.

[4] "TSPLIB," 2008, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

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

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

[7] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A comparison of genetic sequence operators," *Proceeding of the 4th International Conference on Genetic Algorithms*, pp. 69–76, 1991.

[8] S. Tsutsui and M. Miki, "Using edge histogram models to solve flow shop scheduling problems with probabilistic model-building genetic algorithms," *Recent Advances in Simulated Evolution and Learning, Kay Chen Tan, Meng Hiot Lim, Xin Yao, and Lipo Wang Eds, Advances in Natural Computation Series, Chapter 13*, pp. 230–249, 2004.

[9] S. Tsutsui and G. Wilson, "Solving capacitated vehicle routing problems using edge histogram based sampling algorithms," *Proceedings of the 2004 Congress on Evolutionary Computation (CEC' 04)*, pp. 1150–1157, 2004.

[10] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol. 21, pp. 498–516, 1973.

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

[12] D. Thierens, "Analysis and design of genetic algorithms, Doctoral dissertation, katholieke universiteit leuven," 1995.

[13] M. Pelikan, "Bayesian optimization algorithm: From single level to hierarchy. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL," 2002.

[14] M. Pelikan, D. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[15] P. Larrañaga and J. Lozano, "Estimation of distribution algorithms: A new tool for evolutionary computation," 2002.

[16] M. Pelikan, K. Sastryand, and D. E. Goldberg, "iBOA: The incremental bayesian optimization algorithm," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2008)*, pp. 455–462, 2008.

[17] P. Bosman and D. Thierens, "Crossing the road to efficient IDEAs for permutation problems," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 219–226, 2001.

[18] ——, "Permutation optimization by iterated estimation of random keys marginal product factorizations," *Proceedings of the 2002 Parallel Problem Solving From Nature*, pp. 331–340, 2002.

[19] V. Robles, P. Miguel, and P. Larrañaga, "Solving the traveling salesman problem with EDAs," *Estimation of Distribution Algorithms*, pp. 211–229, 2002.

[20] S. Tsutsui, "*c*AS: Ant colony optimization with cunning ants," *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)*, pp. 162–171, 2006.

[21] J. L. Bentley, "Fast algorithms for geometric traveling salesman problems," *ORSA Journal on Computing*, vol. 4, pp. 387–411, 1992.

[22] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems," *(INFORMS*.

[23] D. S. Johnson, "Experimental analysis of heuristics for the STSP," *G. Gutin and A. P. Punnen (ed.), The Traveling Salesman Problem and Variations*, vol. 9, pp. 369–443, 2002.

[24] "ANSI C Code as gzipped tar file, Concorde TSP Solver," 2006, http://www.tsp.gatech.edu/concorde.html.