Multi-armed Bandit Based Evolutionary Model Adaptation for Permutation Problems

Chu-Yu Hsu r01921096@ntu.edu.tw
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Tian-Li Yu tianliyu@ntu.edu.tw
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Jia-Hui Wu r02921074@ntu.edu.tw
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Abstract

1 Introdution

2 Permutation Problems

Many optimization problems find a natural representation of the solution as permutations. In combinatorics, a permutation can be understood as a vector $\pi = (\pi_1, \pi_2, ..., \pi_n)$ of indices $(1, 2, ..., n)$ such that $\pi_i \neq \pi_j$ for all $i \neq j$. We say that the index j is in position i in π when $\pi_i = j$.

Although the solution of many combinatorial optimization problems can be encoded as permutations, the way that information is represented by the indices in the permutation differ from one problem to another. It is importation when solving these problems with EDAs. Therefore, the permutation models EDAs using should take the semantics of the permutation into account. Since the permutation model aims to learn the semantics of the set of selected solutions, the performance of the algorithm is closely related to the encoding of the individuals and to the ability of the permutation model to exploit the information given by those individuals.

In some problems, solution quality depends mainly on the relative ordering of pairs of indices in the permutation and the task is to find a permutation to fulfil a set of relative-ordering constraints. For instance, an example of this type of problem is job-shop scheduling. Sometimes solution quality depends mainly on the pairs of indices that are located next to each other, which is a typical feature of the traveling salesman problem. In some permutation problems, the quality of solution mainly depends on the absolute positions of indices in the permutation. For example, the quadratic assignment problem is this type of problem.

As the semantics of the permutation problems differs, the understanding of the permutation problem help understand for what the permutation model is designed for and the difficulty that EDAs for permutation problem are facing. In the following section, we introduce some typical examples of permutation problems with different semantics.

2.1 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) (Goldberg and Lingle, 1985) consists of looking for the shortest path, in terms of time, distance, or any similar criterion, to go over n different cities visiting each city only once and returning to the city of departure. A solution is usually given by a sequence of cities which is represented as a permutation. The search space is denoted as

$$\Pi = \{(\pi_1, \pi_2, ..., \pi_n) | \pi_i \in \{1, 2, ..., n\}, \pi_i \neq \pi_j, \forall i \neq j\}.$$

In a TSP instance of 4 cities, $\pi = (3, 2, 4, 1)$ would be a possible solution, indicating that the initial city is 3, then 2, 4, 1, finally coming back to 3. As we assume that the first city of the path is not fixed, TSP is a problem with cyclic solutions, and each solution can be represented by $2n$ different permutations for symmetric instances and n for asymmetric instances. For instance, the solution $\pi' = (1, 3, 2, 4)$ represents the same city-tour that π does, while the permutations are different.

The objective function F, is defined as the sum of the distances of going from city $i-1$ to i, denoted as d_{ij}, through all cities in the order specified by the permutation:

$$F(\pi_1, \pi_2, ..., \pi_n) = \sum_{i=2}^{n} d_{\pi_{i-1}\pi_i} + d_{\pi_n \pi_1}.$$

In TSP we note that the relevant information for the calculation of the fitness function of a solution is given by the relative ordering of the indices in the permutation. The information drawn from the absolute positions of each index is useless, as stated with π and pi'. Furthermore, no matter which position indices i and j are in the permutation, if they are adjacent, the contribution to the objective function is the same.

2.2 Capacitated Vehicle Routing Problem

The capacitated vehicle routing problem (CVRP) (Toth and Vigo, 2001) is a most generalized version of TSP. CVRP involves a fleet of vehicles which supply customers with parcels. The vehicles are based at a depot, or hub, from where they must begin and end their route. Customers are located at given points on a map. Each vehicle has a given capacity and each customer has a given demand. The delivery must be accomplished by finding optimal routes which result in a minimal total distance.

Let C be delivery capacity of each vehicle, L_i be the length of the route of vehicle i, W_i be the total weight of the parcels assigned to vehicle i, and N_v be the number of vehicles. Then the capacitated vehicle routing problem can be formulated as follows:

$$\text{Minimize } L = \sum_{i=1}^{N_v} L_i$$
$$\text{Subject to } W_i \leq C \text{for each} i = 1, 2, ..., N_v$$

The encoding method for CVRP basically follows the method used in (Tsutsui and Wilson, 2004). The vehicles and the customers are encoded into a fixed size permutation. In the permutation, the node numbers $0, 1, 2..., N_c - 1$ are assigned for each customer, and node numbers $N_c, N_c + 1, N_c + 2, ..., N_c + N_v - 2$ are assigned for route dividing, where N_c is the number of customers. Hence, $N_v - 1$ different node numbers are used as virtual nodes for route dividing.

Figure 1 shows an example of this permutation representation. In this case, there are four vehicles ($N_v = 4$), and the number of customers is eight ($N_c = 8$). Thus, the

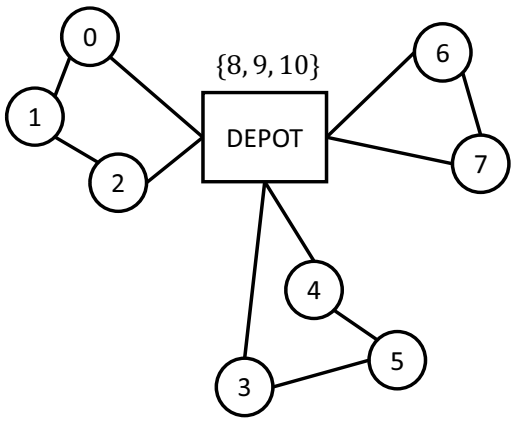


Figure 1: Permutation Representation of CVRP, $N_c = 8$, $N_v = 4$

node numbers $0, 1, 2, ..., 7$ are for customers and numbers $8, 9$ and 10 are for the depot. The permutation $\pi = \{0, 1, 2, 9, 5, 4, 3, 8, 10, 7, 6\}$ is interpreted as the three vehicle routes. In this sample, nodes 8 and 10 are adjacent and being virtual nodes, there is no distance between them and so no additional vehicle is assigned. Thus, three vehicles are used in this string.

The objective function F, is defined as the sum of the distances traveled by each vehicle:

$$F(\pi_1, \pi_2, ..., \pi_n) = \sum_{i=1}^{N_v} L_i + K \cdot \sum_{i=1}^{N_v} \max(0, L_i \frac{W_i - C}{C}),$$

where the first term of this equation is the total length of the routes of N_v vehicle and the second term is the penally for exceeding the weight capacity. This term is normalized by the capacity C so that the penalty is applied consistently among variations in the absolute value. K is a constant value.

In CVRP, the calculation of the objective function can be broken into calculation of the sum of each route distance of the vehicle. Then, the distance of each route is given by the relative ordering of the indices in the route. Dividing customers into different routes for vehicle increases difficulty of this problem than that in TSP. Since the objective function partially account on the relative ordering of the indices in the permutation, the information drawn from the relative ordering of each index is more significant.

Vehicle Routing Problem With Time Window

The vehicle routing problem with Time Window (VRPTW)(Toth and Vigo, 2001) is the variation of CVRP. VRPTW has one more constraint than CVRP besides the capacity of the vehicle. The delivery locations have time windows within which the deliveries (or visits) must be made, or the routing scheme is failed. In the original VRPTW, the number of vehicle is also a variable of this optimization problem. Whereas, to only prove of the concept that MABMA can handle with more complicated problem, we reduce original VRPTW to a fixed vehicle VRPTW (fvVRPTW). That is, the optimal number of vehicle is given as CVRP does.

The objective function of the fvVRPTW is like the one defined in Section 2.2 but

the delay penalty is added:

$$F(\pi_1, \pi_2, ..., \pi_n) = \sum_{i=1}^{N_v} L_i + K \cdot \sum_{i=1}^{N_v} \max\left(0, L_i \frac{W_i - C}{C} + D_i\right),$$

where D_i is the delay penalty of vehicle i. The delay penalty D_i defined as the difference of the arrival time and the location closing time. For more specific definition, please refer to (Toth and Vigo, 2001).

2.3 Permutation Flow-Shop Scheduling Problem

In the permutation flow-shop scheduling problem (PFSP) (Gupta and Stafford Jr, 2006; French, 1982; Allahverdi et al., 2008), n jobs $(i = 1, 2, ..., n)$ have to be scheduled on m machines $(j = 1, 2, ..., m)$ in such a way that a given criterion is minimized. A job consists of m operations and the j-th operation of each job must be processed on machine j for a given specific processing time without interruption. The processing times are fixed, non-negative values and every job is available at time zero. At a given time, a job can start on the j-th machine when its $(j-1)$-th operation has finished on machine $(j-1)$, and machine j is free.

The goal of the optimization is to minimize the processing time of all the jobs, or in other words, to minimize the processing time of the last job. The solution is encoded as a permutation of length n that represents the ordering in which the jobs are going to be processed. This means that for each machine the order of the jobs is the same and it is given as a permutation. For instance, in a problem of 4 jobs and 3 machines, the solution $(1, 2, 3, 4)$ represents that job 1 is processed first, next job 2 and so on.

Let $p_{i,j}$ denote the processing time for job i on machine j,and $c_{i,j}$ denote the completion time of job i on machine j. Then $c_{\pi_i,j}$ is the completion time of the job scheduled in the i-th position in the sequence on machine j. $c_{\pi_i,j}$ is computed as $c_{\pi_i,j} = p_{\pi_i,j} + \max(c_{\pi_i,j-1}, c_{\pi_{i-1},j})$. Therefore, the objective function F is defined as follows:

$$F(\pi_1, \pi_2, ..., \pi_n) = c_{\pi_n,m}.$$

As can be seen, the quality of a solution of the problem is given by the processing time of the last job π_n in the permutation, since this job finishes the last. Even though the objective function is given by the time of this last job, the completion time of this last job depends on the ordering of the previous $\pi_1, \pi_2, ..., \pi_{n-1}$ jobs. Furthermore, in this problem, the value of the objective function can not be decomposed and depends on the position of each index in the permutation as well as on the whole order of the jobs.

2.4 Linear Ordering Problem

The linear ordering problem (LOP) (Martí and Reinelt, 2011) arises in a large number of applications in a number of fields such as economy, sociology, graph theory, archaeology, and task scheduling (Schiavinotto and Stützle, 2004). Two well known examples of LOP are the triangularization of inputoutput matrices of an economy, where the optimal ordering allows economists to extract some information about the stability of the economy, and the stratification problem in archaeology, where LOP is used to find the most probable chronological order

In LOP, for an given $n \times n$ matrix $C = [c_{ij}]$, and the goal is to determine a simultaneous permutation of the rows and columns of C such that the sum of the superdiagonal entries is as large as possible (or equivalently, the sum of the subdiagonal entries is as

small as possible). The solution of LOP is encoded as permutation of length n where each index π_i $(i = 1, ..., n)$ means that the values of the π_i-th row and column of the matrix are reallocated to the i-th position. The objective function is defined as follows:

$$F(\pi_1, \pi_2, ..., \pi_n) = \sum_{i=1}^{n} \sum_{j=i}^{n} c_{\pi_i \pi_j}.$$

In this problem we can see that the contribution of an index π_i to the objective function depends on the previous and posterior indices to it. However it does not depend on the order of these previous and posterior indices.

2.5 Quadratic Assignment Problem

The quadratic assignment problem (QAP) was introduced in (Koopmans and Beckmann, 1957) as a mathematical model for the location of a set of indivisible economical activities. Consider the problem of allocating a set of facilities to a set of locations, with the cost being a function of the distance and flow between the facilities, plus costs associated with a facility being placed at a certain location. The objective is to assign each facility to a location such that the total cost is minimized.

Specifically, for the two given $n \times n$ input matrices with real elements $H = [h_{ij}]$ and $D = [d_{kl}]$, where h_{ij} is the flow between facilities i and j, and d_{kl} is the distance between locations k and l. Given n facilities, the solution of QAP is encoded as a permutation $\pi = (\pi_1, \pi_2, ..., \pi_n)$ where each π_i $(i = 1, ..., n)$ represents the facility that is allocated to the i-th location. The fitness of the permutation is given by the following objective function:

$$F(\pi_1, \pi_2, ..., \pi_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} d_{\pi_i \pi_j}.$$

In the solution of QAP, the absolute position of a facility (node) effects the cost between The quality of the solution is determined by the absolute position of each index (facility) in the permutation as regards the absolute position of the remaining indices.

3 Existing Semantic Models For Permutation Problems

This chapter give brief introductions of the existing semantic models for permutation problems. The semantic models are probabilistic models for solving permutation problems. The semantic models can describe the semantics of the specific permutation problems. Many semantic models, including the edge histogram matrix (EHM), the node histogram matrix (NHM), the Plackett-Luce (PL) model and mallows model (Ceberio et al., 2012; Bosman and Thierens, 2001; Tsutsui, 2002; Tsutsui et al., 2006; Ceberio et al., 2011, 2013; Pelikan et al., 2007), have been proposed. We introduce three of these models as representative models. The first model is EHM proposed in (Tsutsui, 2002) which learns the adjacency of the indices. The second model is NHM (Tsutsui et al., 2006) which learns the absolute position of the indices. The third model is the PL model introduced to EDA literature in (Ceberio et al., 2013) which learns the probability model of the ordering of the permutation sequence.

3.1 The Edge Histogram Matrix

In (Tsutsui, 2002) a new type of EDA for permutation problems called the edge histogram based sampling algorithm (EHBSA) is introduced. The algorithm estimates a

probabilistic model that learns the adjacency of the indices in the selected individuals at each generation, which is named EHM. For an L-dimensional problem, the model is given by a matrix $E = [e_{ij}]$ where $e_{ij} = P(\pi_{k+1} = j|\pi_k = i)$ and $i, j \in \{1, 2, ..., L\}$ and $k \in \{1, 2, ..., L-1\}$. Each e_{ij} is added a ϵ value in order to control the pressure in sampling and avoid individuals with probability Θ or 1. ϵ is denoted as

$$\epsilon = \frac{2N}{L-1} B_{ratio},$$

where N is the size of the set of the selected individuals and B_{ratio} ($B_{ratio} > 0$) is a constant defined by the authors.

In order to sample the probabilistic model, the authors use an algorithm that samples the positions of the permutation ordered, starting with position 1. Once position i-th has been sampled, position $(i+1)$-th is sampled using the row of matrix E corresponding to the index sampled at position i-th. This row is modified by setting to 0 those values which previously appeared and normalizing the rest of the values. A pseudo-code for the sampling algorithm can be seen in Algorithm **??**.

In addition to this sampling, the authors propose another sampling strategy that extends the one introduced by using an individual of the previous generation to sample a new individual. The new sampling strategy consists of the following steps. A parent individual is selected from the previous generation at random and $c > 2$ crossover points in the individual are selected uniformly at random, dividing the parent into c segments of variable length. Randomly selected $c-1$ segments of the parent are copied to the new individual and the remaining non-sampled segment in the individual is simulated by sampling the probabilistic model with the previously introduced strategy. This sampling procedure leads to new individuals that differ from their parents on average on the positions of L/c indices.

According to the authors, the introduced sampling strategies are called sampling without template (EHBSAWO) and sampling with template (EHBSAWT) respectively.

3.2 The Node Histogram Matrix

In (Tsutsui et al., 2006) the node histogram based sampling algorithm (NHBSA) is introduced. NHBSA builds a first order marginals matrix that represents the distribution of the indices across the (absolute) positions of the individuals in the set of the selected individuals. The model of a L-dimensional problem is given by a matrix $H = [h_{ij}]$ where $h_{ij} = P(\pi_i = j)$ and $i, j \in \{1, 2, ..., L\}$. Hence, h_{ij} represents the probability of the index j to be in the i-th position in the selected individuals.

As in EHBSA, a ϵ is added to each h_{ij} in order to control the pressure in sampling, where N represents the size of the set of the selected individuals and B_{ratio} is a positive constant ratio set by the authors. ϵ is denoted as

$$\epsilon = \frac{N}{L} B_{ratio},$$

The design of NHBSA focuses particularly on those problems where the main contribution to the objective function is given by the absolute position of the indices in the permutation. As regards the sampling method, two strategies are proposed to simulate new individuals. A first proposal introduced by the authors uses a sampling strategy that samples the positions of the permutation randomly. Similarly to EHBSAWO, at each step, the sampling algorithm sets to 0 the probabilities in H of the variables sampled in the individual and normalizes the probabilities of the remaining variables to sum 1.

The second sampling algorithm uses a parent individual from the previous generation to create the new individual. A random individual is picked up from the previous generation and c random single positions are copied to the new individual. The remaining empty positions are filled by sampling the probabilistic model. The authors denote as NHBSAWT and NHBSAWO, NHBSA that use the sampling with template and sampling without template respectively.

3.3 The Plackett-Luce Model

The Plackett-Luce (PL) model (Marden, 1996) is the probabilistic model used in the EDA proposed in (Ceberio et al., 2013), which is called Plackett-Luce EDA. It is belongs to the family of Thurstone order statistics models. It takes its name from the combination of two works which are proposed by Plackett (Plackett, 1975) and Luce (Luce, 2005)(Plackett, 1975).

Formally, the PL model is given by

$$P(\pi|W) = \prod_{i=1}^{n-1} \frac{w_{\pi_i}}{\sum_{j=i}^{n} w_{\pi_j}}, \tag{1}$$

where $W = \{w_1, w_2, ..., w_n\}$ is a parameter vector. Comparing the parameters w_i and w_j,i, the larger parameter w_i has the higher probability that item i appears in the front of item j on the permutation. The typical way to fit a PL model is by maximum likelihood estimation (MLE) of the parameters w. Hunter (Hunter, 2004) describes a way to do this using a Minorization-Maximization (MM) algorithm.

Once the parameter vector is known, we can sample new individual as following procedure. First, we sample the first position of the individual with the probability $P(\pi(1) = j) = w_j$. The following positions are sampled by sampling the remaining w which are normalized to sum 1. Repeating the process until all positions are sampled.

4 Essence of Semantic Model Adaptation

This chapter show the essence of semantic model adaptation. Due to the different semantics of permutation problems, it is non-trivial to determine which model. Besides, the semantics of some permutation problem needs more than one model to describe. PFSP is an example of such problem that needs models which can handle connection of nodes and absolution position of nodes (Tsutsui et al., 2006).

The semantics go beyond the capability of existing models. As a result, existing models can not capture the semantics precisely. For example, LOP, of which objective function is mainly contributed by the order of indices, can not be interpreted precisely by either NHM or EHM. When it comes the real-world problems, the semantics in the permutation may even complicated (Ceberio et al., 2012). For this reason, the capability of solving the more complicated permutation problems is crucial.

5 Essence of Semantic Model Adaptation

This chapter shows the essence of semantic model adaptation. Due to the different semantics of permutation problems, it is non-trivial to determine which model. Besides, the semantics of some permutation problem needs more than one model to describe. PFSP is an example of such problem that needs models which can handle connection of nodes and absolution position of nodes (Tsutsui et al., 2006).

The semantics go beyond the capability of existing models. As a result, existing models can not capture the semantics precisely. For example, LOP, of which objective

	NHBSA	EHBSA
bayg29	1.34×10^{-1}	0.00
dantzig42	4.69×10^{-1}	1.43×10^{-3}
eil51	5.59×10^{-1}	0.00
berlin52	5.55×10^{-1}	0.00
eil76	9.72×10^{-1}	1.86×10^{-4}

(a) TSP

	NHBSA	EHBSA
nug18	3.11×10^{-4}	8.50×10^{-3}
nug17	0.00	1.15×10^{-3}
nug21	6.56×10^{-4}	1.05×10^{-2}
bur26b	1.82×10^{-4}	1.36×10^{-3}
bur26a	2.58×10^{-4}	1.51×10^{-3}

(b) QAP

Table 1: Average error for each instance

function is mainly contributed by the order of indices, can not be interpreted precisely by either NHM or EHM. When it comes the real-world problems, the semantics in the permutation may even complicated (Ceberio et al., 2012). For this reason, the capability of solving the more complicated permutation problems is crucial.

5.1 Effects of Different Models

As a semantic model can only express specific permutation problems, model-choosing becomes critical. For illustration, the following experiment uses TSP and QAP as example. In TSP, the relative ordering of each indices in permutations contributes the fitness. However, the information drawn from the absolute positions of each indices are useless. In contrast, on QAP, the quality of the solution is determined by the absolute position of each index in the permutation. Hence, the relative ordering of the indices in permutation does not matter for QAP.

For investigating the capability of different semantic models, we test EHBSA and EHBSA on QAP and TSP with the population size $N = 2 \times L$ and the NFE limit $E_{max} = L \times 40000$, where L is the problem size. For every instance, we perform 10 independent runs.

The experiment settings are as follows:

Table 1 shows the average error for each instance of problems. The average error is calculated as the normalized difference between the best objective value obtained by the algorithm and the best known solution. The lower the values are, the better the performance. As can been seen, the EHBSA outperforms the NHBSA in all instances of TSP. In contrast, the NHBSA outperforms the EHBSA in all instances of QAP.

This result confirms that model-choosing is crucial for EDAs for permutation problems. This performance difference between these two models starts from the design. In other word, EHM captures the adjacency of indices which matches the semantics of TSP, while NHM captures the absolute positions of indices which is more fitted on QAP. These differences makes the nature that before solving any permutation problem, a semantic model which is able to describe the semantics of the problem should be chosen.

Although model-choosing significantly affects the performance, the semantics of permutation problem is not always so clear for the literature (Ceberio et al., 2012). For instance, the objective function of the permutation flow-shop scheduling problem (PFSP) is given by the completion time of the last job that depends on the ordering of the jobs. Intuitively, EHM is the best choice for PFSP, but the empirical result shows that NHM ties with EHM. For this reason, model-choosing rises the difficulty for solving permutation problems.

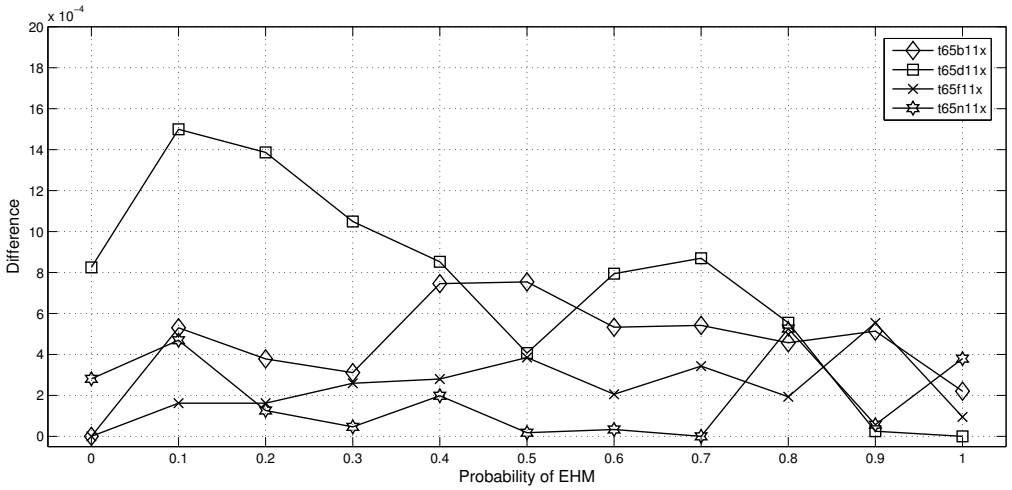


Figure 2: Difference of average of each probability p on LOP

5.2 Verification by sweeping

Since some problems exist, of which the semantics is not exactly belong the index adjacency or absolute position, incorporating different models is a way to achieve higher fitness. In order to investigate the potential for improvement, we hybridizes EHM and NHM in the model-building stage in EDAs.

In our experiments,we performs a trial of hybridization of EHM and NHM. The hybrid algorithm generally follows Tsutsui's evolutionary model, but with some modification. Different from Tsutsui's evolutionary model (Tsutsui, 2006), the hybrid algorithm picks one model to sample new individuals with the given probability and updates both models at the same time. The probability for the EHM p starts from 0.0 to 1.0 and increases 0.1 by each iteration. Then, we perform the hybrid algorithm with the population size $N = 2 \times L$ and the NFE limit $E_{max} = L \times 40000$ independent 10 runs on each iteration. After each trial, we records its the best solution and its objective function value. LOP and PFSP are tested in our experiments, and four instances of each problem are picked: $t65b11x$, $t65d11x$, $t65f11x$ and $t65n11x$ from LOLIB[1] and $ta021$, $ta041$, $ta051$ and $ta071$ from Taillard's instance library [2]. In the end, the best mixing ratio of models is revealed.

Figures 2 shows the difference between average of each probability. The values on the y-axis is the normalized difference between the average the best objective values in each trial and the worst average of object values among all trials. The higher the values is, the better the performance. The values on the x-axis are the probability p of to sampling EHM. In other word, the value $p = 1.0$ indicates that the hybrid algorithm is identical to EHBSA which is by pure EHM, and the value $p = 0.0$ indicates that the hybrid algorithm is identical to NHBSA which is by pure NHM.

As can be seen, in some instances of PFSP, the best objective value is obtained in the trials that both models have chance to sample new individual. In all of the instances of LOP, the experiments with only one model have lower object value than the one with hybridization of two models. These experiments indicate that mixing different models

[1] http://www.optsicom.es/lolib/
[2] http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html

can be more efficient when solving permutation problems. Thus, we make the three attempt at model adaptation in the follow chapters.

6 The Enumeration Method

At the first attempt, we use enumeration method to choose model. This chapter shows the algorithm of enumeration method. Besides, we test enumeration method on different permutation problems, and this experiments shows advantages and disadvantages of enumeration method.

For the highest fitness solution, every type of problems needs a most expressive semantic model to express. Although the literature has deep knowledge about some of permutation problems, new problems are still formatted into permutation problems consistently, to which the existing models cannot precisely capture. When the semantics is beyond the capability of existing models, another new expressive semantic model for the specific problem from new problems is needed. Unfortunately, designing new semantic models is not a trivial work. In the second experiment of the previous section, the result shows the potential of mixing different models. The experiment also shows that with limited semantic models, through incorporating different semantic models, the hybridization of EHM and NHM solve the specific problems without the expressive semantic model more efficiently than algorithms with single model. Although we can find the best parameter setting through sweeping each mixing ratio thoroughly, it is impractical when the number of used semantic models grows.

Algorithm 1: The Enumeration Model

Input: The initial population P,
The semantic model set $\mathbb{M}$
Output: The offspring population O

build every semantic model for P;
randomly chose $p \in P$ as template;
initialize an individual set C ;
for $m \in \mathbb{M}$ **do**
  generate a new individual c by model m;
  $C \leftarrow C \cup \{c\}$
$c \leftarrow thebestsolution \in C$;
$O \leftarrow$P $\cup\{c\} - \{p\}$;
return O;

In the enumeration method, multiple semantic models generate new solution candidates, and the best one is chosen. This chapter shows that the strategy of generating new individuals from multiple models at once and keeping the best descendant in population outperform in some problems.

The enumeration method acts almost similarly to EHBSA or NHBSA except for multiple models. In each generation, the enumeration method randomly picks one individual from the population as template and then generates new individuals from each models with the picked template if need. Next, the population keeps the best individual among the newly generated individuals and the template, and discards the rest. Finally, the enumeration updates each semantic models with the new population, and continues the evolution until the terminal criterion is met. The pseudo-code is in Algorithm 1.

	EHBSA	Enum2	NHBSA		EHBSA	Enum2	NHBSA
LOP				**TSP**			
t65b11xx	1	2	3	**burma14**	2	2	2
t65d11xx	3	1	2	**bayg29**	2	3	1
t65f11xx	2	1	3	**dantzig42**	2	3	1
t65n11xx	2	1	3	**eil51**	1	3	2
t65w11xx	3	1	2	**berlin52**	2	3	1
t69r11xx	2	1	3	**eil76**	1	3	2
t75i11xx	3	2	1	**eil101**	1	3	2
PFSP				**QAP**			
ta001	1	2	3	**nug17**	3	1	2
ta021	3	1	2	**nug18**	3	2	1
ta031	1.5	1.5	3	**nug20**	3	2	1
ta041	2	1	3	**nug21**	3	2	1
ta051	3	1	2	**tai10a**	2	3	1
ta061	1.5	1.5	3	**bur26a**	3	2	1
ta071	2	1	3	**bur26b**	3	2	1

Table 2: Rank of algorithms, comparison for the 2-model enumeration method

In our experiments, we perform enumeration method with the population size $N = 2 \times L$ and the NFE limit $E_{max} = L \times 40000$. And template technique is used, NHM and EHM are used semantic models. The cut-point number is 3 for template.

The results are in Table 2. Four problems are tested in the experiment. In our experiments, we use the fractional ranking system as the index of the performance. Then rank is sorted by the average fitness of each algorithm. The lower the rank is, the better the performance. Items that compare equal receive the same ranking number, which is the mean of what they would have under ordinal rankings. Equivalently, the ranking number of 1 plus the number of items ranked above it plus half the number of items equal to it. This strategy has the property that the sum of the ranking numbers is the same as under ordinal ranking. Table 2 shows that the enumeration outperformed the other algorithms on LOP and PFSP. On the contrary, the enumeration method underperformed on TSP and QAP. In this experiments, the enumeration method performs better on the problems without a existing expressive semantic model for the specific problem from the problems.

In spite of the higher fitness solution on the problems without a existing expressive semantic model for the specific problem from the problems, the enumeration method shows worse performance on problems which have the expressive semantic models for the problems. Furthermore, adding new models into the system makes the situation even worse. Since the enumeration method iterates all of models in system, it wastes lots of resources on unexpressive models, instead of only focusing on the models which can generating competitive solutions. Thus, we attempt to develop another method in the follow chapter.

7 The Bayesian Method

At the second attempt, we use Bayesian probability to determine which semantic model is chosen. This chapter shows the introduction of Bayesian probability, the method of model adaptation. Besides, we investigate the issue of this method.

To avoid wasting lots of resources on unexpressive models, we need a more advanced model adaptation technique. That is to say, the model adaptation should invest the limited computing resources in the promising model more often and explore different models when the most expressive model is uncertain. For this purpose, the probability of choosing the model is measured in terms of the Bayesian posterior probability.

Although this method can work theoretically, it doesn't work as expected in our experiments. We will describe this phenomenon in the follow section.

7.1 The Bayesian Posterior Probability

In Bayesian inferencing, the posterior probability (Lee, 2012) of a random event is the conditional probability that is assigned after the relevant evidence is taken into consideration. Similarly, the posterior probability distribution is the probability distribution of an unknown quantity, treated as a random variable, conditional on the evidence got from an experiment.

In our case, we want to measure the posterior probability of a semantic model given the permutation problem P: $Pr(M|P)$. Assuming all semantic models are equally probable, the posterior is proportional to its likelihood $Pr(P|M)$, which can be easily calculated as $Pr(P|M) = \prod_{\forall i \in P} Pr(i|M)$, where P is the population containing individuals and $Pr(i|M)$ is the conditional probability of a specific individual in the population given the semantic model.

7.2 Model Adaptation

Algorithm 2: The Bayesian Method

Input: The initial population P,
The semantic model set $\mathbb{M}$
Output: The offspring population O

```
build each semantic model m ∈ 𝕄 for P;
measure the posterior of each model m ∈ 𝕄 according to P;
offspring population O := {};
foreach p ∈ P do
    choose a model m according the probability distribution P(𝕄);
    let p be the template, generate a new individual c by model m;
    if c is better than p then
        O := O ∪ {c};
    else
        O := O ∪ {p};
return O;
```

Since we have the method to measure the posterior of the semantic models, this method can be applied to the model adaptation. For the preliminary study, the posterior of the semantic model is directly used as the probability of choosing the model to generate new individuals.

Formally, for each model $m \in M$, where M is the set of the models the system has, $Pr(m)$ is the probability that the model m is chosen as the most expressive model to

generate new individuals among the model set M. The probability is defined as

$$Pr(m) = \frac{q_m}{\Sigma_{j \in M} q_j},$$

where q_j is the posterior of the semantic model j.

For model adaptation, the Bayesian method chooses a model before the sampling phase. Because the population reflect the semantics of the permutations as the population evolves, the posterior of the expressive models for the specific permutation problem increases. Hence, these models for the specific permutation problem have more chance to generate new individuals, while the other candidate models still have little chance.

The pseudo code is in Algorithm 2. The model adaptation stage between model building and sampling. Let the population size be N. In every generation, the Bayesian method iterates the whole population to make each individual template for generating new individual, and then keeps the better one between new generated offspring and the template. This difference accelerates the population reflecting the problem semantics, due to the population moves toward the high fitness solutions.

7.3 Empirical Study

In this subsection, the performance of the Bayesian method is investigated. In practice, the conditional probability of the population given the model will be expressed in logarithm for computation convenience, since the value is too small for the float-point number in computer.

For the experiment purpose, TSP and the flat problem are tested. The objective value of the flat problem is a constant, no matter the permutation is. Hence, the posterior of the solution on the flat problem is independent from the semantic models.

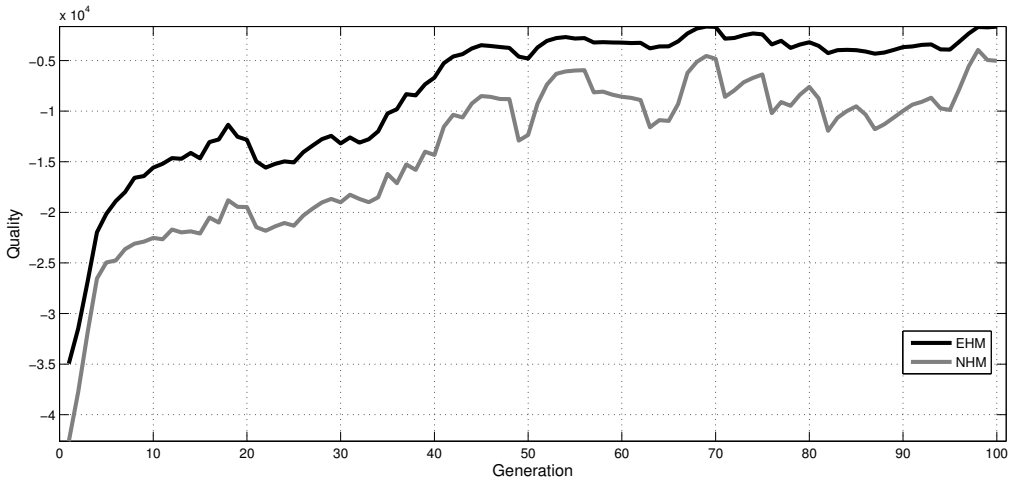
In the experiment of the Bayesian method, EHM and NHM are used. The population size N is 1000 for TSP and N is 100 for the flat problem. We perform independent 10 runs on each instance. The problem size of the flat problem is 50. The instance eil51 of TSP has been used in this empirical study [3].

Figure 3 shows the posterior of the semantic models in term of the logarithm the conditional probability of the population given the semantic model. The higher the posterior, the better the model describes the problem. As can be seen, at the beginning of the trials on the flat problem and TSP, the posterior of NHM is lower than that of EHM. After 5 generations, the posterior of NHM becomes higher than that of EHM on the flat problem and remains higher until the trial ends.

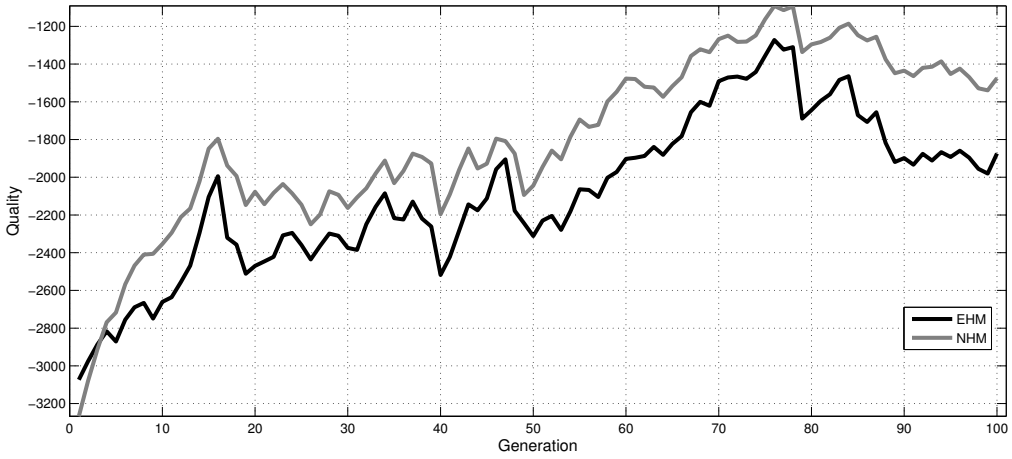
Although the posterior of EHM in TSP is higher than that of NHM as we expected, the posterior of NHM on TSP grows faster than EHM as the population evolves. To demonstrate the phenomenon, we test the flat problem in our experiments. The posterior of the both models on the flat problem is not identical as we expected. This result shows that NHM is more sensitive than EHM to population changes.

Of course, these effects of the phenomenon can be eliminated by preproduce. However, introducing new techniques to the Bayesian method for balancing different models also increase the cost of adding new models into this system, so we attempt to develop another method in the follow chapter.

[3] **TSPLIB** http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/



(a) TSP, eil51, $L = 51$, $N = 1000$



(b) Flat, $L = 50$, $N = 100$

Figure 3: (Temp)The posterior of the semantic model, tournament selection $S = 2$

8 Multi-armed Bandit Based Model Adaptation

At the last attempt, we use the UCB algorithm to choose model. The UCB algorithm is commonly used to solve multi-armed bandit problem. This chapter shows the introduction of UCB and the model adaptation framework.

Ideally, we choose one of the most expressive semantic models for the problem in each generation to reduce NFE. However, without given the semantics of the problem in prior, we can only explore several different semantic models information before fully exploiting the most promising model. Therefore, we use the techniques from the multi-armed bandit problem to balance exploration and exploitation.

8.1 Multi-armed Bandit Problem

A multi-armed bandit (MAB) is like a traditional slot machine but with K levers. When pulled, each lever provides a reward drawn from the distribution associated to that specific lever. The gambler has no prior knowledge of reward distribution of the levers, but through repeated trials, he can focus on the most profitable lever (Mohri, 2005; Kuleshov and Precup, 2000).

The classic MAB problem consists of random variables $X_{i,n}$ for $1 \leq i \leq K$ and $n \geq 1$, where each i stands for the index of a gambling machine (*i.e.*, the arm of a bandit). Successive plays of machine i yield rewards $X_{i,1}, X_{i,2}, ...$ which are independent and identically distributed by unknown law and with unknown expectation.

For a formal definition, the random variables are independent across the arms and are associated to probability distributions $(D_1, D_2, ..., D_k)$ with expectations $(\mu_1, \mu_2, ..., \mu_k)$ and variances $(\sigma_1, \sigma_2, ..., \sigma_k)$. Throughout the playing, the distributions are unknown for gambler. The goal for gambler is to collect as much rewards as possible by pulling the armed over many times. At each turn, $t = 1, 2, ...$, the gambler selects an arm, with index $j(t)$ and the times played n, and receives a reward $X_{j(t),n}$. The MAB algorithms specify a strategy or a polity by which the gambler should choose an arm i at each turn.

The most popular performance measure for bandit algorithms is the total *expected regret*, defined for any fixed turn T as:

$$R_T = T\mu^* - \sum_{t=1}^{T} \mu_{j(t)},$$

where $\mu^* = \max_{i=1,...,k} \mu_i$ is the expected reward from the best arm. Thus, the regret is the loss caused by the policy not always playing the best machine. For a large class of pay-off distributions, there is no policy whose regret would grow slower than $O(\ln n)$ (Lai and Robbins, 1985). For such pay-off distributions, a policy is said to resolve the exploration-exploitation trade-off if its regret growth rate is within a constant factor of the best possible regret rate.

Upper Confidence Bound Algorithms

The UCB algorithms have been proposed by (Auer et al., 2002; Auer, 2003) which are a family of widely used in the MAB problem. The UCB family was designed as a simpler, more elegant implementation of the idea of optimism in the face of uncertainty, proposed by Lai & Robbins (Lai and Robbins, 1985). An extension of UCB-style algorithms to sequential, tree-based planning was developed in (Kocsis and Szepesvári, 2006), and it has been proved very successful in Go playing programs (Kocsis and Szepesvári, 2006; Gelly and Silver, 2007; Gelly et al., 2008).

Algorithm UCB1, whose finite-time regret is studied in details by (Auer et al., 2002), is a simplest algorithm that succeeds in resolving the exploration-exploitation trade-off. In the UCB, rewards are either 0 or 1, and the empirical quality $\overline{x}$ lies in $[0, 1]$. It keeps track the average rewards $X_i, T_i(t-1)$ for all the arms and chooses the arm with the best upper confidence bound:

$$I_t = \underset{i \in \{1,...,K\}}{\operatorname{argmax}} \overline{X}_{i,T_i(t-1)} + c_{t-1,T_i(t-1)}, \tag{2}$$

where $c_{t,s}$ is a bias sequence chosen to be

$$C_{t,s} = \sqrt{\frac{2 \ln t}{s}}. \tag{3}$$

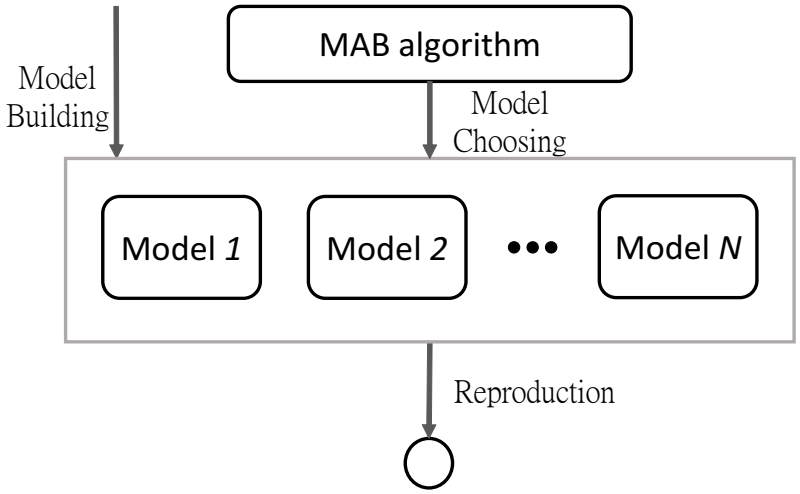


Figure 4: The Model Adaptation Flow

The bias sequence is such that if X_{it} were independently and identically distributed then the inequalities

$$\begin{cases} \mathbb{P}(\overline{X} \geq \mu_i + c_{t,s}) \leq t^{-4}, \\ \mathbb{P}(\overline{X} \leq \mu_i - c_{t,s}) \leq t^{-4} \end{cases} \tag{4}$$

were satisfied. This follows from Hoeffding's inequality.

The authors also propose another algorithm, UCB1-Tuned, which they claim performs better in practice but comes without theoretical guarantees. The main feature of UCB1-Tuned is that it takes into account the variance of each arm and not only its empirical mean. More specifically, at turn $t = 1, 2, ...$ the algorithm picks an arm $j(t)$ as

$$j(t) = \underset{i \in \{1,\ldots,K\}}{\operatorname{argmax}} \left(\mu_i + \sqrt{\frac{\ln t}{n_i} \min\left(\frac{1}{4}, V_i(n_i)\right)} \right), \tag{5}$$

where

$$V_i(t) = \sigma_i^2(t) + \sqrt{\frac{2 \ln t}{n_i(t)}}. \tag{6}$$

The estimate of the variance $\sigma_i^2(t)$ can be computed as usual by maintaining the empirical sum of squares of the reward, in addition to the empirical mean. Audibert *et al.* (Audibert et al., 2009) provide expected regret bounds and regret concentration results for variance-based UCB algorithms similar to UCB1-Tuned.

8.2 The Proposed Framework

In this paper, the model adaptation framework incorporates with the UCB algorithms for a preliminary study. Although many other MAB techniques exist and show good performances (Kuleshov and Precup, 2000), the UCB family is commonly used recent years, and is easy to be implemented. Figure 4 shows the flow of the model adaptation framework. Along with the original EDA framework, the model adaptation adds a new phase "Model Choosing" between phase "Model Building" and "Sampling". As can be seen, instead of generating a population of new individual with one model in

the original version of sampling procedure, in our framework, new individuals are sampled one by one with different models.

Algorithm 3: Multi-armed Bandit Based Model Adaptation

Input: The initial population P,
The semantic model set $\mathbb{M}$
Output: The offspring population O

```
1  generate one individuals from each model;
2  update each model m ∈ 𝕄;
3  randomly chose p ∈ P;
4  choose a semantic model m_UCB with highest UCB value;
5  let p be the template, generate a new individual c by model m_UCB;
6  if c is better than p then
7      O ← P ∪ {c} − {p};
8      reward model m_j with value 1.0;
9  else
10     O ← P;
       reward model m_j with value 0.0;
11 return O;
```

Algorithm 3 introduces a details pseudo-code of the proposed multi-armed bandit based model adaptation (MABMA). Lines 10 and 11 replace p with the newly generated individual c. To improve performance, we will introduce a new methodology to choose replaced individual in next chapter. As we treat the semantic models as arms in the MAB problem, at the initialization stage of the algorithm, the UCB algorithms needs initial value of the semantic models. Thus, each model m_j have to generate one new individual by a randomly picked template for the initialization of UCB.

Properly rewarding the selected arm is a crucial element that should be carefully designed to ensure success (Fialho et al., 2010). The choice of the reward must encourage exploitation of the best arm while still allowing exploration of the other arms. Most methods use as raw reward the fitness improvement of the newborn offspring compared to a base individual, that might be its parent (Barbosa and Sá, 2000; Fialho et al., 2008, 2009; Lobo and Goldberg, 1997; Tuson and Ross, 1998), the current best in the population (Davis, 1989), or the median individual of the current population (Julstrom, 1995). The concept used in our case is borrowed from an assumption made in original UCB (Auer et al., 2002) – the reward are with support in $[0, 1]$. The reward given to the bandit is binary. An arm is assigned a reward of 1 if the fitness of the newborn individual is better than the template, and 0 otherwise.

This reward policy shares similar arguments with the policy used in (De Rainville et al., 2013). The direct fitness allocation suffers when the bandit is updated with a new fitness causing a loss – if this fitness dominates any other arm causing a gain, the arm receives a higher credit for the loss. Since the template plays a critical role in the fitness of the new born individual, the fitness cannot reflect the appropriateness of a semantic model. Furthermore, the scale of the fitness value dramatically differs from problem to problem, or even from instance to instance. The fitness can not reflect the difference between two permutations. For instance, two permutation could have huge difference in the fitness because merely two nodes swapped its position in the permutation.

	EHBSA	MABMA-UCB1	MABMA-UCBT	NHBSA	PLEDA
A-n32-k5	1	5	3	2	4
A-n33-k5	1.5	4	1.5	3	5
A-n33-k6	1.5	1.5	3	4	5
A-n34-k5	1	2	3	4	5
A-n36-k5	1	4	2	3	5
A-n63-k9	2	3	1	4	5
A-n64-k9	1	2	3	4	5
A-n65-k9	1	2	3	4	5
A-n69-k9	1	3	2	4	5
A-n80-k9	1	3	2	4	5

Table 3: The rank on CVRP

	EHBSA	MABMA-UCB1	MABMA-UCBT	NHBSA	PLEDA
C101_0025	2	2	2	4	5
C103_0025	2	2	2	4	5
RC102_0025	1	2	3	5	4
RC105_0050	3	2	1	4	5
C101_0100	4	1	2	3	5
C101_0025	3	1.5	1.5	4	5
C103_0025	3	1.5	1.5	4	5
RC102_0025	1	2	3	5	4
RC105_0050	3	2	1	4	5
C101_0100	4	1	2	3	5

Table 4: The rank of the fvVRPTW

8.3 Empirical Study

In this subsection, the performance of MABMA is investigated. Several permutation problems are tested. For comparison, MABMA with UCB1 and UCB1-Tuned are be tested. In the experiment of Multi-armed Bandit Based Model Adaptation, EHM, NHM and the PL model are used. For each instance, we perform independent 10 runs with the population size $N = 2 \times L$ and the NFE limit $E_{max} = L \times 40000$. In this section, CVRP and the variation of CVRP – the vehicle routing problem with Time Window (VRPTW) (Toth and Vigo, 2001) – are tested.

The results are shown in Table 4. MABMA-UCB1 is for MABMA with the UCB1 algorithm; MABMA-UCBT is for MABMA with the UCB1-Tuned algorithm. The values listed in the tables are the average of the best fitness of the algorithm obtained in 10 trials. Because CVRP aims to minimize the distance vehicles traveled, the fitness is the smaller, the better the performance.

In CVRP instances, EHBSA outperforms the other algorithms. As can be seen, the average fitnesses of the MABMA algorithms are most close to the ones of optimal model, EHM. As the problem size becomes larger, the difference between EHBSA and MABMA decreases. In the fvVRPTW, MABMAs outperform the other algorithms on large problem instances. On the easier problem instance, MABMAs and EHBSA has identically performance, because they achieve the optimal solution.

Remarkably, on CVRP, the performance of MABMA-UCBT is more closer to that of

EHBSA than that of MABMA-UCB1, while MABMA-UCB1 achieve better performance in the fvVRPTW. This fact may be caused by the variance factor in the UCB1-Tuned algorithm can help the MABMA stick to the most appropriate model better. However, when the most appropriate model is not existed, the UCB1 algorithm can start to explore in the earlier stage.

9 Replacement Strategy

This chapter focuses on replacement strategy and give a brief introduction to the restricted tournament replacement (RTR). This chapter also shows the effects of RTR, the essence of distance-measure adaptation and the method of distance-measure adaptation. To reduce NFE, we use RTR to decide which individual should be replaced or be preserved. Thus, we define three types of distance measures for RTR on the permutation problems.

9.1 Restricted Tournament Replacement

The restricted tournament replacement (RTR) is a commonly used niching technique in the field of EDAs. RTR is also known as restricted tournament selection (RTS) which is first proposed in (Harik, 1995). RTR uses the tournament strategy to decide the part which get to move on to the next generation. We present the process of RTR in algorithm 4, where the $Distance$ function is usually the Hamming distance in binary problems. However, Hamming distance is not suitable for the permutation problems. Thus, we define new distance for the permutation problems by characteristics of the permutation problems. In our work, the window size w is suggested to set to $L/3$.

Algorithm 4: The algorithm of RTR

Input: The original population $X = \{x_1, x_2, ..., x_{n-1}\}$;
The offspring population Y;
The window size w;
Output: The population X of the next generation;
for $y \in Y$ **do**
  Choose a random subset $S = \{s_0, s_1, ..., s_{w-1}\}(0 \leq s_i < n)$;
  Find the x_{s_i} in $\text{argmin}_{s_i \in S}\, Distance(x_{s_i}, y)$;
  if $Fitness(y) > Fitness(x_{s_i})$ **then**
    $x_{s_i} \leftarrow y$;
return X;

9.2 Distance Metric

This subsection shows three types of distance measures for RTR on the permutation problems: edge distance, node distance and order distance. We define three distance measures by considering semantics of permutation problems.

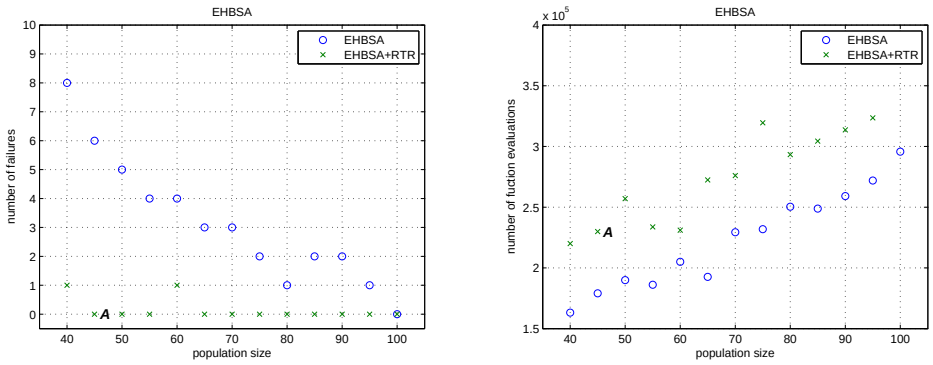
Edge Distance

An edge is connection between two nodes. The edge distance is calculated by different edge between two individuals. Let $D_{edge}(i, j)$ denote the edge distance between the two individuals i and j, and

$$ES_i = \{\forall k(\pi_{(k \bmod L),i}, \pi_{(k+1 \bmod L),i})\} \text{ is an edge set of individual } i.$$

Thus, $D_{edge}(i,j)$ is calculated as follows:

$$D_{edge}(i,j) = L - |ES_i \cap ES_j|.$$



(a) PopulationSize-Failures (b) PopulationSize-N_{fe}

Figure 5: (Temp)Results of EHBSA on benchmark $eil51$ with template technique and $B_{ratio} = 0.00$.

We test EHBSA and EHBSA+RTR which uses the edge distance with different population size 20 times on benchmark $eil51$ respectively. The results are shown in Figure 5. The number of function evaluations increases and number of failures decreases as population size growing up. Comparing the performance of EHBSA and EHBSA+RTR, number of function evaluations of EHBSA+RTR is larger than EHBSA, but number of failures of EHBSA+RTR is the smaller than EHBSA. The point A represent the best performance of EHBSA+RTR, and it is smallest number of function evaluations among all lowest failure rate points in Figure 5. We get the conclusion that EHBSA+RTR with minimum required population size outperforms EHBSA. Edge distance is useful on TSP.

Node Distance

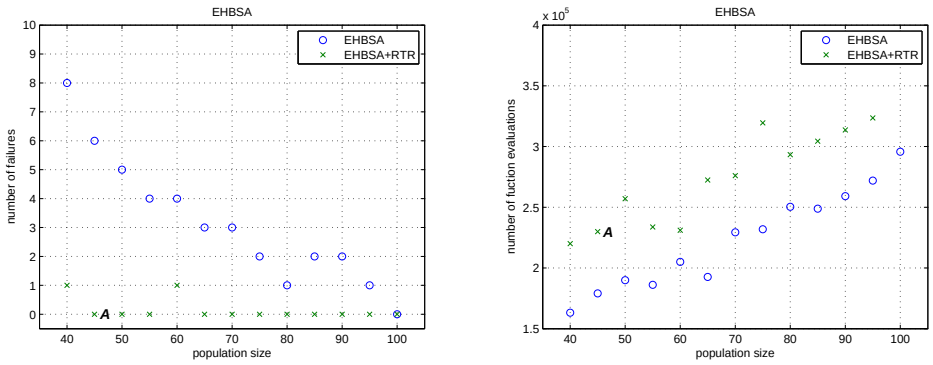
The node distance is calculated by each node at different position between two individuals. Let $D_{node}(i,j)$ denote the node distance between the two individuals i and j. $D_{node}(i,j)$ is calculated as follows:

$$D_{node}(i,j) = \sum_{k=0}^{ell-1} r_{i,j}(k),$$

where $r_{i,j}(k)$ is a function defined as

$$r_{i,j}(k) = \begin{cases} 1, & \text{if } \pi_{k,i} \neq \pi_{k,j} \\ 0, & \text{otherwise} \end{cases}.$$

We test NHBSA and NHBSA+RTR which uses the node distance with different population size 20 times on FSSP respectively. The results are shown in Figure 6. This experiment indicates that NHBSA+RTR with minimum required population size outperforms NHBSA, and node distance is useful on FSSP.



(a) PopulationSize-Failures (b) PopulationSize-N_{fe}

Figure 6: (Temp)Results of NHBSA on benchmark $ta031$ with template technique and $B_{ratio} = 0.00$.

Order Distance

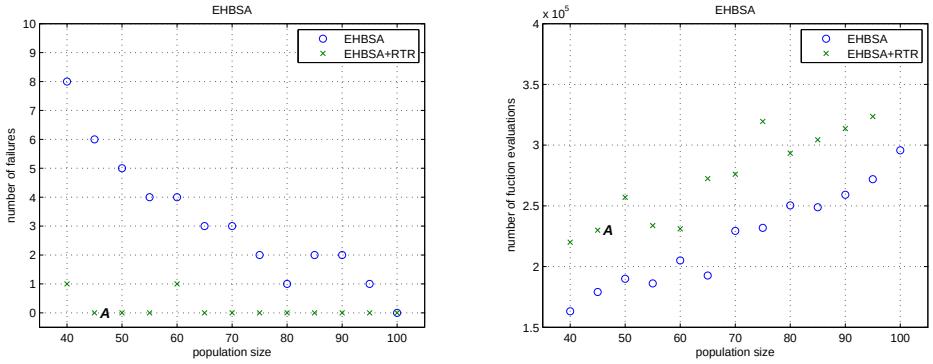
The order distance is calculated by different orders of every node pair between two individuals. Let $D_{order}(i, j)$ denote the order distance between the two individuals i and j. $D_{order}(i, j)$ is calculated as follows:

$$D_{order}(i, j) = \sum_{u} \sum_{v} \delta_{i,j}(u, v),$$

where $\delta_{i,j}(u, v)$ is a function defined as

$$\delta_{i,j}(u, v) = \begin{cases} 0, & \text{if } H_{\pi_u}, i > H_{\pi_v}, i \text{ and } H_{\pi_u}, j > H_{\pi_v}, j \\ 1, & \text{otherwise.} \end{cases}$$

In the above, H_{π_x}, i denotes π_x of individual i.



(a) PopulationSize-Failures (b) PopulationSize-N_{fe}

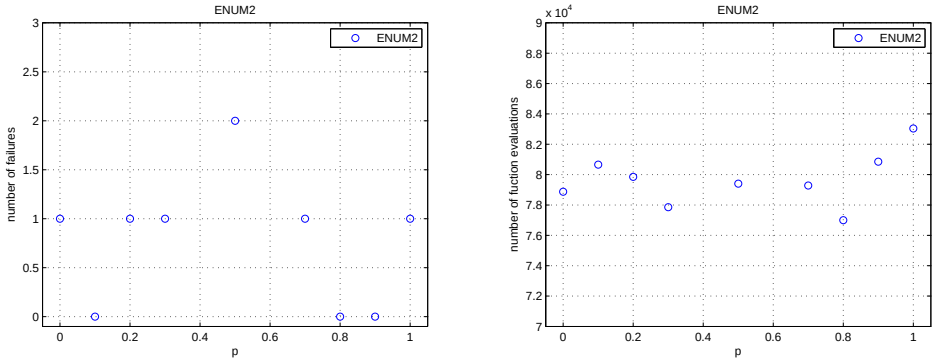
Figure 7: (Temp)Results of Enum2(EHBSA, NHBSA) on benchmark $t65i11xx$ with template technique and $B_{ratio} = 0.000$.

We test Enum2 and Enum2+RTR which uses the order distance with different population size 20 times on LOP respectively. The results are shown in Figure 6. This experiment indicates that Enum2+RTR can solve LOP more effective than Enum2, and order distance is useful on LOP.

9.3 Essence of Distance-measure Adaptation

This section shows the essence of distance-measure adaptation. For the same reason of essence of model adaptation(chapter 5), the mechanism of distance-measure adaptation is necessary. Due to the different semantics of the permutation problems , the best fit distance measure is non-trivial.

In our experiments, we choose edge distance with the probability p and choose node distance with the probability $1 - p$ in each generation. The probability p starts from 0.0 to 1.0 and increases 0.1 by each iteration. the instances of LOP are used : $t65b11x, t65d11x, t65f11x$ and $t65n11x$ from LOLIB. We perform independent 20 runs with the the NFE limit $E_{max} = L \times 40000$ for each iteration.



(a) p-Failures (b) p-N_{fe}

Figure 8: (Temp)Results of enumeration method on LOP with RTR by using the edge distance and node distance.

Figure 8 shows NFE and the number of failures of each probability p. We can find that the best performance is not at two ends of range of p. Therefore, best distance measure is not fixed on node distance or edge distance. This experiment indicates that mixing different distance measure can solve the permutation problems more effectively.

9.4 Model-associated Method

The model-associated method is a method of distance-measure adaptation. In each RTR, the model-associated method chooses the corresponding distance measure with selected model on model choosing phase. For example, let $MS = \langle m_1, m_2, ..., m_{N_M} \rangle$ be an ordered set of models and $DS = \langle d_1, d_2, ..., d_{N_M} \rangle$ be an ordered set of distance measures, where N_M is the number of models. Besides, m_i represents model i, and d_i denotes the corresponding distance measure of model i. the model-associated method chooses the distance measure corresponding to the selected model in the model choosing phase. Table 5 shows the models and their corresponding distance measures.

To investigate the performance of the model-associated method, we test Enum2+RTR by using the edge distance, the node distance and the model-associated method on LOP with the NFE limit $E_{max} = L \times 40000$.

Semantic Models	Corresponding Distance
NHM	node
EHM	edge
the PL model	order

Table 5: Corresponding distance measures of models

9.5 Empirical Study

This section shows the performance of MABMA+RTR on the different permutation problems. For comparison, MABMA-UCB, MABMA-UCBT, MABMA-UCB+RTR, MABMA-UCBT+RTR are tested on LOP,TSP,QAP and VRPTW. In the experiment of the model-associated method, EHM, NHM and the PL model are used. For each instance, we perform independent 20 runs with the NFE limit $E_{max} = L \times 40000$.

10 Conclusion

References

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985 – 1032.

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902.

Auer, P. (2003). Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Barbosa, H. J. and Sá, A. M. (2000). On adaptive operator probabilities in real coded genetic algorithms. In *Workshop on Advances and Trends in Artificial Intelligence for Problem Solving (SCCC'00)*.

Bosman, P. A. and Thierens, D. (2001). Crossing the road to efficient ideas for permutation problems. In *Proc. of the Genetic and Evolutionary Computation Conf.–GECCO*, pages 219–226.

Ceberio, J., Irurozki, E., Mendiburu, A., and Lozano, J. A. (2012). A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence*, 1(1):103–117.

Ceberio, J., Mendiburu, A., and Lozano, J. A. (2011). Introducing the mallows model on estimation of distribution algorithms. In *Neural Information Processing*, pages 461–470. Springer.

Ceberio, J., Mendiburu, A., and Lozano, J. A. (2013). The plackett-luce ranking model on permutation-based optimization problems. In *2013 IEEE Congress on Evolutionary Computation (CEC)*, pages 494–501. IEEE.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

De Rainville, F.-M., Sebag, M., Gagné, C., Schoenauer, M., and Laurendeau, D. (2013). Sustainable cooperative coevolution with a multi-armed bandit. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1517–1524.

Fialho, Á., Da Costa, L., Schoenauer, M., and Sebag, M. (2008). Extreme value based adaptive operator selection. In *Parallel Problem Solving from Nature–PPSN X*, pages 175–184. Springer.

Fialho, Á., Da Costa, L., Schoenauer, M., and Sebag, M. (2010). Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1-2):25–64.

Fialho, Á., Schoenauer, M., and Sebag, M. (2009). Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 779–786. ACM.

French, S. (1982). *Sequencing and scheduling: an introduction to the mathematics of the job-shop*, volume 683. Ellis Horwood Chichester.

Gelly, S., Hoock, J.-B., Rimmel, A., Teytaud, O., Kalemkarian, Y., et al. (2008). On the parallelization of monte-carlo planning. In *ICINCO*.

Gelly, S. and Silver, D. (2007). Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM.

Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, volume 154. Lawrence Erlbaum, Hillsdale, NJ.

Gupta, J. N. and Stafford Jr, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711.

Harik, G. R. (1995). Finding multimodal solutions using restricted tournament seletion. pages 24–31.

Hunter, D. R. (2004). Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, pages 384–406.

Julstrom, B. A. (1995). What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 81–87, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kocsis, L. and Szepesvári, C. (2006). *Bandit based monte-carlo planning*. Springer, New York, NY.

Koopmans, T. C. and Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76.

Kuleshov, V. and Precup, D. (2000). Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Lee, P. M. (2012). *Bayesian statistics: an introduction*. John Wiley & Sons, New York, NY.

Lobo, F. G. and Goldberg, D. E. (1997). Decision making in a hybrid genetic algorithm. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 121–125. IEEE.

Luce, R. D. (2005). *Individual choice behavior: A theoretical analysis*. Courier Dover Publications, Mineola, New York.

Marden, J. I. (1996). *Analyzing and modeling rank data*. CRC Press, Boca Raton, Florida.

Martí, R. and Reinelt, G. (2011). *The linear ordering problem: exact and heuristic methods in combinatorial optimization*, volume 175. Springer, New York, NY.

Mohri, M. (2005). Multi-armed Bandit Algorithms and Empirical Evaluation. pages 437–448.

Pelikan, M., Tsutsui, S., and Kalapala, R. (2007). Dependency trees, permutations, and quadratic assignment problem. In *Genetic And Evolutionary Computation Conference: Proceedings of the 9 th annual conference on Genetic and evolutionary computation*, volume 7, pages 629–629.

Plackett, R. L. (1975). The analysis of permutations. *Applied Statistics*, pages 193–202.

Schiavinotto, T. and Stützle, T. (2004). The linear ordering problem: Instances, search space analysis and algorithms. *Journal of Mathematical Modelling and Algorithms*, 3(4):367–402.

Toth, P. and Vigo, D. (2001). *The vehicle routing problem*. Siam, Philadelphia, PA.

Tsutsui, S. (2002). Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In *Parallel Problem Solving from Nature?PPSN VII*, pages 224–233. Springer.

Tsutsui, S. (2006). A comparative study of sampling methods in node histogram models with probabilistic model-building genetic algorithms. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 4, pages 3132–3137. IEEE.

Tsutsui, S., Pelikan, M., and Goldberg, D. E. (2006). Node histogram vs. edge histogram: A comparison of pmbgas in permutation domains. *MEDAL Report*, (2006009).

Tsutsui, S. and Wilson, G. (2004). Solving capacitated vehicle routing problems using edge histogram based sampling algorithms. In *IEEE Congress on Evolutionary Computation(CEC), 2004.*, volume 1, pages 1150–1157. IEEE.

Tuson, A. and Ross, P. (1998). Adapting operator settings in genetic algorithms. *Evolutionary computation*, 6(2):161–184.