Technical Report
EHU-KZAA-TR-5-2011

Universidad del País Vasco
Euskal Herriko Unibertsitatea

UNIVERSITY OF THE BASQUE COUNTRY
Department of Computer Science and Artificial Intelligence

A review on Estimation of Distribution Algorithms in Permutation-based Combinatorial Optimization Problems

Josu Ceberio, Ekhine Irurozki,
Alexander Mendiburu, Jose A. Lozano

June 2011

San Sebastian, Spain
http://www.ccia-kzaa.ehu.es

Noname manuscript No.
(will be inserted by the editor)

A review on Estimation of Distribution Algorithms in Permutation-based Combinatorial Optimization Problems

Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu and Jose A. Lozano

the date of receipt and acceptance should be inserted later

Abstract Estimation of Distribution Algorithms (EDAs) are a set of algorithms that belong to the field of Evolutionary Computation. Characterized by the use of probabilistic models to represent the solutions and the dependencies between the variables of the problem, these algorithms have been applied to a wide set of academic and real-world optimization problems, achieving competitive results in most scenarios. Nevertheless, there are some optimization problems, whose solutions can be naturally represented as permutations, for which EDAs have not been extensively developed. Although some work has been carried out in this direction, most of the approaches are adaptations of EDAs designed for problems based on integer or real domains, and only a few algorithms have been specifically designed to deal with permutation-based problems. In order to set the basis for a development of EDAs in permutation-based problems similar to that which occurred in other optimization fields (integer and real-value problems), in this paper we carry out a thorough review of state-of-the-art EDAs applied to permutation-based problems. Furthermore, we provide some ideas on probabilistic modeling over permutation spaces that could inspire the researchers of EDAs to design new approaches for these kinds of problems.

1 Introduction

The research work carried out in the field of metaheuristics has provided the community with a large number of tools for solving optimization problems. In this work, we focus on a set of metaheuristics called Estimation of Distribution Algorithms (EDAs) [26, 29, 37, 39] that belong to the field of Evolutionary Algorithms (EAs). The main characteristic of EAs is the use of techniques inspired by the natural evolution of the species. In nature, species change across time; individuals evolve, adapting to the characteristics of the environment. This evolution leads to individuals with better characteristics. The same idea is translated to the world of computation, where an individual represents a particular solution for the problem

Intelligent Systems Group · Computer Science and Artificial Intelligence Department · The University of the Basque Country

1. $D_0 \leftarrow$ Generate M individuals (the initial population) at random
2. **Repeat** for $l = 1, 2, ...$ until a stopping criterion is met
3. $\quad D_l^{Se} \leftarrow$ Select $N \leq M$ individuals from D_{l-1} according to the selection method
4. $\quad p_l(\boldsymbol{x}) = p(x|D_{l-1}^{Se}) \leftarrow$ Estimate the probability distribution of an individual being among the selected individuals
5. $\quad D_l \leftarrow$ Sample M individuals (the new population) from $p_l(\boldsymbol{x})$
6. **End**

Fig. 1: General outline of Estimation of Distribution Algorithms (EDAs).

to be solved, a population comprises several individuals, and different operators such as crossover, mutation and selection techniques are used to make the individuals (solutions) evolve (improve). The most popular reference of these types of algorithms are the Genetic Algorithms (GAs) [14].

As an improvement of GAs, EDAs were introduced in the field of Evolutionary Algorithms in [36], although previous similar approaches can be found in [56]. Unlike GAs, EDAs learn a joint probability distribution associated with the set of most promising individuals at each generation, trying to explicitly express the interrelations between the different variables (characteristics) of the problem. Sampling the probabilistic model generated in the previous generation, a new population of solutions for the problem is created. The algorithm stops iterating and returns the best solution found across the generations when a certain stopping criterion is met, such as a maximum number of generations/evaluations, homogeneous population, or lack of improvement in the solutions. Figure 1 introduces a detailed pseudo-code of EDAs.

Based on this general framework, several EDA approaches have been developed in the last years [26, 29, 38, 39], where each approach learns a specific probabilistic model that conditions the behavior of the EDA from the point of view of complexity and performance. Many works in the literature confirm the good performance of EDAs in the solution of problems from diverse fields. Protein Folding [45], Capacitated Vehicle Routing Problems [52], Calibration of Chemical Applications [35], Finding the Optimal Path in 3D Spaces [53], Software Testing [44], Chemotherapy Treatment Optimization for Cancer [6] or Nuclear Reactor Fuel Management Parameter Optimization [21] are some examples of many real-world problems where EDA-based approaches were applied to find optimal solutions.

In this work, we are interested in the solution of a specific subset of NP-hard optimization problems. Particularly, we refer to those problems whose solutions can be naturally represented as a permutation. Even though the literature provides several EDA approaches for permutation-based problems, most of these approaches are adaptations of EDAs designed initially for the solution of integer or real-value domain problems. We understand integer domain problems as those problems where the search space is defined as

$$\Omega = \{0, \dots, r_1\} \times \dots \times \{0, \dots, r_n\}, \quad \text{where} \quad r_i \in \mathbb{N} \quad i = 1, \dots, n$$

and by real-value-based problems we understand those problems where the search space is an infinite non-numerable subset of $\mathbb{R}^n$.

The EDAs designed for the previous two kinds of problems show several drawbacks when applied to permutation-based problems. The main drawback is that

those EDAs do not learn a probability distribution over a permutation space, but a distribution over an integer or real-values space. Therefore, these models are not summarizing the regularities contained in the permutations.

In order to set the basis for a development of EDAs in permutation-based problems similar to that given for integer and real-value optimization problems, we carry out a thorough review of state-of-the-art EDAs applied to permutation-based problems. Furthermore, we provide some ideas on probabilistic modeling over permutation spaces that could inspire the researchers of EDAs to design new approaches for this kind of problems.

The remainder of this paper is organized as follows. In Section 2 we give a background on permutation-based problems that will be used in Section 3 to base the review of EDA approaches designed for solving permutation-based optimization problems. In Section 4 we carry out a thorough experimental analysis of the existing EDA proposals when applied to classical permutation-based problems. In Section 5 we present several models for the estimation of probability distributions over permutation search spaces giving some advice on their use in EDAs. Finally, Section 6 sums up the main conclusions and raises some ideas for future work.

2 Permutation-based problems

As mentioned previously, many optimization problems find a natural representation of the solution as permutations. In combinatorics, a permutation is understood as a vector $\sigma = (\sigma_1, \ldots, \sigma_n)$ of the indexes $\{1, \ldots, n\}$ such that $\sigma_i \neq \sigma_j$ for all $i \neq j$. We say that index j is in position i in σ when $\sigma_i = j$.

While there exist many combinatorial optimization problems whose solutions are based on permutations, the meaning of these permutations can be different in different problems. This fact is important when solving these problems with EDAs, as the probabilistic model should take into account the semantic information of the permutation. For that reason, in the following paragraphs we introduce some examples of permutation-based problems where, although the codification of the solution is given by permutations, the meaning in each case is different.

2.1 Travelling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) [15] 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

$$\Omega = \{(\sigma_1, \sigma_2, \ldots, \sigma_n) | \sigma_i \in \{1, 2, \ldots, n\}, \sigma_i \neq \sigma_j, \forall i \neq j\}.$$

In a TSP of 4 cities, $\sigma = (3, 2, 4, 1)$ 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, the 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, solution $\sigma' = (1, 3, 2, 4)$ represents the same city-tour that σ 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:

$$F(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sum_{i=2}^{n} d_{\sigma_{i-1}\sigma_i} + d_{\sigma_n,\sigma_1}$$

.

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 indexes in the permutation. The information drawn from the absolute positions of each index is useless, as stated with σ and σ'. Furthermore, no matter which position indexes i and j are in the permutation, if they are adjacent, the contribution to the objective function is the same.

2.2 Flow Shop Scheduling Problem (FSSP)

The Flow Shop Scheduling Problem [17] consists of scheduling n jobs ($i = 1, \ldots, n$) on m machines ($j = 1, \ldots, m$). A job consists of m operations and the j-th operation of each job must be processed on machine j for a specific time. A job can start on the j-th machine when its $(j-1)$-th operation has finished on machine $(j-1)$, and if 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 codified 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 $c_{\sigma_i,j}$ is the completion time of the job scheduled in the i-th position in the sequence on machine j. $c_{\sigma_i,j}$ is computed as $c_{\sigma_i,j} = p_{\sigma_i,j} + max\{c_{\sigma_i,j-1}, c_{\sigma_{i-1},j}\}$. Therefore, the objective function F is defined as follows:

$$F(\sigma_1, \sigma_2, \ldots, \sigma_n) = c_{\sigma_n,m}$$

As can be seen, the solution of the problem is given by the processing time of the last job σ_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 $\sigma_1, \ldots, \sigma_{n-1}$ 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.

2.3 Linear Ordering Problem (LOP)

In the Linear Ordering Problem (LOP), we are given an $n \times n$ 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

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

$$F(\sigma_1, \ldots, \sigma_n) = \sum_{i=1}^{n} \sum_{j=i}^{n} c_{\sigma_i \sigma_j}$$

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

2.4 Quadratic Assignment Problem(QAP)

The Quadratic Assignment Problem (QAP) [23] is the problem of allocating a set of facilities to a set of locations, with a cost function associated to the distance and flow between the facilities. The objective is to assign each facility to a location such that the total cost is minimized. Specifically, we are given two $n \times n$ input matrices with real elements $\mathbf{H} = [h_{ij}]$ and $\mathbf{D} = [d_{kl}]$, where h_{ij} is the flow between facility i and facility j and d_{kl} is the distance between location k and location l. Given n facilities, the solution of the QAP is codified as a permutation $\sigma = (\sigma_1, \ldots, \sigma_n)$ where each σ_i $(i = 1, \ldots, n)$ represents the facility that is allocated to the i-th location. The fitness of the permutation is given by the following objective function:

$$F(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} * d_{\sigma_i \sigma_j}$$

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 indexes.

As stated in the previous problems, the semantic meaning of the permutation may change completely depending on the problem being dealt with. In order to efficiently solve these problems, it is essential to choose the permutation that allows probabilistic models to discover and preserve relative ordering constraints, absolute ordering constraints or adjacency relations of the indexes in the permutation.

3 EDAs for permutation-based optimization problems

This section is devoted to carrying out a review of the different EDA approaches in literature for permutation-based problems. We classify the existing EDAs for solving permutation-based problems into three groups. A first group is composed of those EDAs designed originally for solving integer domain problems and adapted to simulate permutation individuals at the sampling step. In a second group, we place those approaches designed for solving real-value optimization problems that have been modified to handle permutations. Beyond adaptations of existing approaches, the literature includes a few works where the authors introduce specific designs

of EDAs for permutation-based problems, or general designs that are applied to permutation-based problems to illustrate their usefulness for the first time. We place these EDA approaches in the third group.

In the following sections we explain each group in detail and we elaborate on the weak and strong points of each proposal.

3.1 Adaptations of integer encoding EDA approaches

One lead that EDA researchers have followed to deal with permutations is the use of EDAs designed for integer-based problems [8,10,24,25]. These algorithms learn, departing from a dataset of permutations, a probability distribution over a set $\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_n$ where $\Omega_i = \{1, 2, \ldots, r_i\}$, $r_i \in \mathbb{N}$ $i = 1, \ldots, n$, instead of learning a distribution over a permutation space. Therefore, the sampling of these models may not provide permutation individuals but an individual in Ω.

In order to overcome this deficiency, the authors simulate permutation individuals by modifying the sampling step. The most common method to sample a probabilistic model in EDAs is the Probabilistic Logic Sampling algorithm [18]. In this sampling strategy, variables are instantiated following an ancestral order. To sample the i-th ordered variable, the previous $(i-1)$-th variables have to be instantiated. In order to obtain a permutation, the following changes have to be made to the sampling strategy. A permutation can be obtained if the i-th variable is not allowed to take the values instantiated by the previous variables. To do that, when i-th variable has to be sampled, the probability of the previous sampled values is set to 0 and the local probabilities of the rest of the values are normalized to sum 1. Although this procedure leads to permutations, we note that every time that we modify the probabilities to enable sampling permutation individuals, the information kept by the probabilistic model is denaturalized somehow in the sampled solutions.

Without taking into account the complexity of the probabilistic model learnt by the EDA used (univariate, bivariate or multivariate), many integer-based approaches such as Univariate Marginal Distribution Algorithm (UMDA) in [25], Mutual Information Maximization for Input Clustering (MIMIC) in [3], Dependency-Trees [40] or Estimation of Bayesian Network Algorithm (EBNA) [3] have been adapted to deal with permutation-based problems.

3.2 Adaptations of real encoding EDA approaches

Another way that the research community of EDAs has found to approach permutation-based problems is by means of EDAs designed for solving real-value-based problems. These algorithms are based on a method that allows to decodify a real-valued vector as a permutation. Given a real vector $(x_1, x_2, \ldots, x_n)$ of length n, a permutation individual can be obtained from it by ranking the positions using the values x_i, $(i = 1, \ldots, n)$. Supposing we have the real vector:

$$(2.35, 3.42, 9.35, 0.32, 11.54, 10.42, 5.23, 4.2, 7.8),$$

the permutation obtained when decoding the vector, is $(2, 3, 7, 1, 9, 8, 5, 4, 6)$. Introduced first by Bean [2], this strategy is called the Random Keys algorithm. The

main advantage of random keys is that they always provide feasible solutions, since each real-valued vector represents a permutation. However, as stated by Bosman and Thierens [5], random keys strategy is not effective and introduces large overheads since every time that an individual must be evaluated, an ordering algorithm has to be applied to get the corresponding permutation. The ineffectiveness of the approach is related with the redundancy that the codification involves. One can easily notice that real-valued vectors with different values can lead to the same permutation. The real vector

$$(1.78, 3.90, 7.03, 1.24, 12.56, 9.87, 4.27, 4.10, 0.60)$$

would represent the permutation $(2, 3, 7, 1, 9, 8, 5, 4, 6)$. In both cases, the permutation that codifies the real vector is the same, although the vector is different, therefore the same fitness value is assigned by the objective function. This creates many plateaus in the corresponding real-value optimization problems that the EDA is solving.

This random key strategy has been jointly used with different EDAs for real-valued problems [5, 42]. In [30] the Job Shop Scheduling Problem is approached with UMDA for the continuous domain, MIMIC approach for the continuous domain (MIMIC$_c$) and Estimation of Gaussian Networks Algorithms (EGNAs).

3.3 Permutation-oriented EDA approaches

In addition to the previously introduced EDA approaches, the EDAs research community has tried to go a step forward designing new algorithms that consider the real nature of permutations. In the following sections the outcome of that work is introduced and explained in detail. Although some of these algorithms could be considered in the previous two groups, we have introduced them in this group as they have specific designs for permutations or they have been applied to illustrate their usefulness for the first time over permutation-based problems.

3.3.1 IDEA Induced Chromosome Elements Exchanger (ICE)

Bosman and Thierens [5] introduced a new algorithm called IDEA Induced Chromosome Elements Exchanger (ICE) to deal with permutation-based problems. They proposed a modification of the IDEA approach introduced previously by the same authors in [4].

IDEA follows the general framework defined for real-valued-based EDAs considering that the selected population follows a Gaussian distribution. A specific characteristic of IDEA is to factorize the Gaussian density function (pdf) as a product of marginal distributions. Particularly, the variables are partitioned into several subsets and a marginal pdf is estimated for each group. IDEA can be directly applied to permutation-based problems using the random keys representation. However, Bosman and Thierens [5] rejected this strategy since the joint use of random keys and real-value based EDAs, as previously reported by the authors, does not lead to very effective optimization algorithms. To overcome this problem, the ICE algorithm was proposed in which probabilistic sampling of new solutions is replaced by a specialized crossover operator that takes into account the partition of the variables in the probabilistic model learnt. Given two parents, the new

1. Set the position counter $k \longleftarrow 1$.
2. Obtain first node σ_1 uniformly at random from $\{1, 2, \ldots, n\}$.
3. Sample index k.
   3.1. Set to 0 previously sampled variables in row σ_{k+1} of $\mathbf{E}$.
   3.2. Normalize non-sampled variables in row σ_{k+1} of $\mathbf{E}$.
   3.3. Sample next variable σ_{k+1}.
4. Update the position counter $k \longleftarrow k + 1$.
5. If $k < n$, go to Step 3.

Fig. 2: General outline of the Edge Histogram Based Sampling Algorithm.

individual is constructed by randomly picking the values of the variables of a block from a parent. For each block, one of the parents is chosen uniformly at random. Note that in ICE the probabilistic model is not explicitly used to sample new individuals, but only the information related with the partition of the variables is used.

3.3.2 Edge histogram models

In [46, 50] a new type of EDA for permutation-based problems called Edge Histogram Based Sampling Algorithm (EHBSA) is introduced. The algorithm estimates a probabilistic model that learns the adjacency of the indexes in the selected individuals at each generation. For an n-dimensional problem, the model is given by a matrix $\mathbf{E} = [e_{ij}]$ where $e_{ij} = P(\sigma_{k+1} = j | \sigma_k = i)$ and $i, j \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, \ldots, n-1\}$. Each e_{ij} is added a ε value in order to control the pressure in sampling and avoid individuals with probability 0 or 1. ε is denoted as

$$\varepsilon = \frac{2N}{n-1} B_{ratio},$$

where N is the size of of the set of the selected individuals and B_{ratio} ($\mathrm{B}_{ratio} > 0$) 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 $\mathbf{E}$ 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 pseudocode for the sampling algorithm can be seen in Figure 2.

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 n/c indexes.

According to the authors, the introduced sampling strategies are called *sampling without template* (EHBSA$_{WO}$) and *sampling with template* (EHBSA$_{WT}$) respectively.

In [49] the author extends EHBSA to solve the FSSP, designing an asymmetrical edge histogram model. In [48] a revised EHBSA is proposed, referred to as enhanced EHBSA (eEHBSA). This approach presents a more flexible sampling procedure (cut-point selection) and modifies the way the new generation is created.

3.3.3 Node histogram models

In [51] the Node Histogram Based Sampling Algorithm (NHBSA) is introduced. The NHBSA builds a first order marginals matrix that represents the distribution of the indexes across the (absolute) positions of the individuals in the set of the selected individuals. The model of a n-dimensional problem is given by a matrix $\mathbf{H} = [h_{ij}]$ where $h_{ij} = P(\sigma_i = j)$ and $i, j \in \{1, 2, \ldots, n\}$. 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

$$\varepsilon = \frac{N}{n} B_{ratio}$$

The design of the NHBSA focuses particularly on those problems where the main contribution to the objective function is given by the absolute position of the indexes 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 EHBSA$_{WO}$, at each step, the sampling algorithm sets to 0 the probabilities in $\mathbf{H}$ of the variables sampled in the individual and normalizes the probabilities of the remaining variables to sum 1. A pseudocode for the sampling algorithm can be seen in Figure 3.

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 NHBSA$_{WT}$ and NHBSA$_{WO}$, the NHBSA that use the *sampling with template* and *sampling without template* respectively.

In [47] several variations of sampling methods for NHBSA are proposed, such as replacing the random sampling sequence used in the algorithm with the sequential sampling sequence like EHBSA. Another approach changes the number of sampling nodes randomly instead of using a fixed number. Using probability density functions to determine the number c crossover points is also introduced in [48].

1. Generate a random permutation $pos[]$ of $1, \ldots, n$.
2. Generate a candidate node list $C = 1, \ldots, n$.
3. Set the position counter $k \longleftarrow 1$.
4. Sample index $pos[k]$.
   4.1. Set to 0 those probabilities of variables already sampled in $pos[k]$ in $\mathbf{H}$.
   4.2. Normalize remaining probabilities to sum 1.
   4.3. Sample node x.
5. Set $\sigma_{pos[k]} \longleftarrow x$ and remove node x from C.
6. Update the position counter $k \longleftarrow k + 1$.
7. If $k < n$, go to Step 4.

Fig. 3: General outline of the Node Histogram Based Sampling Algorithm.

3.3.4 Recursive EDA

Romero et al. [43] proposed a new class of EDAs called Recursive EDA (REDA). The REDA is an optimization strategy based on EDAs that consists of k optimization stages (see figure 4). In an initial stage an EDA is applied to the problem and a solution is obtained. In a second stage, the variables of the problem are divided in two groups of similar size (when possible). Next, an EDA is executed over the variables that belong to the first group, while the variables of the second group remain fixed to the values given by the optimal solution in the previous stage. This process is repeated, fixing the variables in the second group and optimizing over the first group. This completes the second stage. The remaining stages follow the same procedure recursively. For instance, in the third stage, each group of the second stage is divided in two groups, and each group of variables is optimized separately. The algorithm stops when the number of variables in a group reaches a minimum threshold.

The motivation behind this proposal is to reduce the computational cost of learning the model (which in [34] is identified as the most expensive step of an EDA) by solving smaller problems at each stage.

Although every EDA approach could be used for optimization at each stage, due to the recursive nature of the strategy, the authors suggest using EDAs such as UMDA or MIMIC that permit keeping the computational cost feasible, since the EDA is executed repeatedly.

Even though this strategy is a general scheme and could be applied to any optimization problem, the authors proposed this algorithm for the optimization of the triangulation of bayesian networks, and therefore we classify it as a specific EDA for permutation-based problems.

Regarding the codification scheme, REDAs use the previously introduced random keys encoding in the continuous approaches. However, for discrete domain, they refuse to use straight forward individual codification as do the approaches introduced in Section 3.1. Instead of that, they propose a new codification that allows to learn probability distributions over permutations. In order to do that, they set a bijection between the numbers $\{1, \ldots, n!\}$ to the set of permutations of order n. This bijection is based on the decomposition in prime factors of $n!$ that is given such that $n! = p_1^{n_1} \cdot \ldots \cdot p_r^{n_r}$. An individual is then represented as a vector of length r, corresponding to the number of prime factors. Position i-th

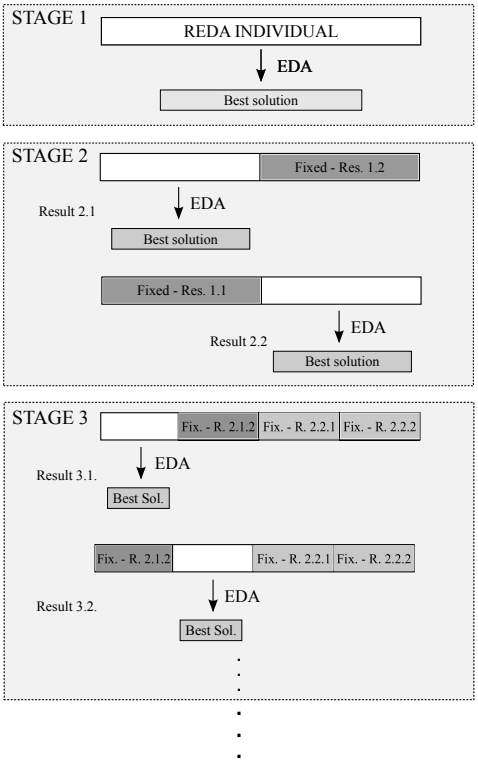

Fig. 4: Recursive EDA Strategy

Function *FromIndividualToPermutation*(input individual, output perm)
1. Initialize a vector l = (1,2,. . . ,n)
2. **For** i=1 to n-1
3. perm(i) = l(individual(i))
4. update l, removing the individual(i) position
5. **End**
6. perm(n) = l(1)

Fig. 5: Function to obtain the permutation *perm* codified by the individual *ord*.

in the individual can take $n_i + 1$ values $\{0, 1, \ldots, n_i\}$, representing the possible exponent of the i-th prime factor. Therefore, given an individual we obtain an integer, and from it the permutation is obtained. Given a particular individual, the corresponding permutation is obtained by the procedure defined in Figure 5. Table 1 shows an example for this procedure, supposing a problem of size 4, and given individual (2,2,1).

Although this codification allows to learn a probability distribution over permutation spaces, the decodification process denaturalizes the relation between the variables and permutations.

Table 1: Example of the function *FromIndividualToPermutation*

Step	*individual*	List l	*perm*	Updated list l
1	(2,2,1)	(1,2,3,4)	()	-
For, $i = 1$	(**2**,2,1)	(1,**2**,3,4)	(2)	(1,3,4)
For, $i = 2$	(2,**2**,1)	(1,**3**,4)	(2 3)	(1,4)
For, $i = 3$	(2,2,**1**)	(**1**,4)	(2 3 1)	(4)
6	(2,2,1)	(**4**)	(2 3 1 4)	()

3.4 Hybrid EDAs

In addition to the previous algorithms, several hybrid EDAs have also been proposed for permutation-based problems. These algorithms generally combine standard EDA approaches with other techniques such as local search [20, 54, 55] or Particle Swarm Optimization (PSO) [28]. In [54, 55] an operator called Guided Mutation is introduced which combines a conventional mutation operator with a probabilistic model learnt at each step. In [7], Chen et al. propose a hybrid EDA for solving single machine scheduling problems that combines classic univariate and bivariate probabilistic models with crossover and mutation genetic operators.

Due to the complexity of studying hybrid approaches, we decided not to include these approaches in the experiments.

4 Experiments

In the following sections we introduce the setup of the experiments and the analysis of the results.

4.1 Experiments setup

We carried out an empirical evaluation of the most representative EDA approaches reviewed in this paper. In order to do that, we considered it interesting to analyze their behavior using a benchmark of classical test problems. Particularly, we selected the following sets of 24 instances for each problem type:

- TSP: *bays29*, *berlin52*, *burma14*, *ch130*, *dantzig42*, *eil51*, *eil76*, *eil101*, *fri26*, *gr17*, *gr24*, *gr48*, *gr96*, *gr137*, *hk48*, *pr76*, *pr107*, *pr124*, *pr136*, *rat99*, *st70*, *swiss42*, *ulysses16* and *ulysses22*[1].
- QAP: *bur26a*, *bur26b*, *bur26c*, *bur26d*, *nug17*, *nug18*, *nug20*, *nug21*, *tai10a*, *tai10b*, *tai12a*, *tai12b*, *tai15a*, *tai15b*, *tai20a*, *tai20b*, *tai25a*, *tai25b*, *tai30a*, *tai30b*, *tai35a*, *tai35b*, *tai40a* and *tai40b*[2].
- LOP: *t75i11xx*, *t65f11xx*, *t65b11xx*, *t65d11xx*, *t65i11xx*, *t65l11xx*, *t65n11xx*, *t65w11xx*, *t69r11xx*, *t70b11xx*, *t70d11xx*, *t70d11xxb*, *be75eec*, *be75np*, *be75oi*, *be75tot*, *tiw56n54*, *tiw56n58*, *tiw56n62*, *tiw56n66*, *tiw56n67*, *stabu70*, *stabu74*, *stabu75* and *usa70*[3].

[1] TSPLIB. http://www2.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp
[2] Éric Taillard's web page. http://mistic.heig-vd.ch/taillard/problemes.dir/qap.dir/qap.html
[3] Optsicom Project. http://heur.uv.es/optsicom/LOLIB/#instances

– FSSP: *tai20×5, tai20×10, tai50×10* and *tai100×20*[4] (The first six instances from each file).

Regarding the set of selected algorithms, the choice has been made according to the classification of EDAs presented in Section 3. From the set of integer-based EDA approaches, we have chosen UMDA, MIMIC, EBNA$_{BIC}$ and TREE. From the group of EDAs belonging to the continuous domain, we have chosen UMDA$_c$ and EGNA$_{ee}$. In addition, all the EDAs specifically designed for solving permutation optimization problems have been selected for the comparison: IDEA-ICE, EHBSA$_{WT}$, EHBSA$_{WO}$, NHBSA$_{WO}$, NHBSA$_{WT}$ and REDA. For comparison purposes we have included a very well known GA, the Ordering Messy Genetic Algorithm (OmeGA) [22].

As previously mentioned, there are hybrid EDAs that have been applied to several permutation-based problems. In these algorithms, it is quite complex to measure what the contribution of the probabilistic model to the optimization process is. Due to this fact, we have limited the experiments to 'pure' EDAs since we aim to analyze the capacity of the different probabilistic models used for solving permutation codification problems.

For each algorithm and problem instance 10 runs have been completed. Table 2 shows the values for the execution parameters of all EDAs, being n the size (number of variables) of the instance. Regarding specific-EDA parameters, the values suggested by their respective authors have been used. Romero et al. [43] suggest executing REDA with fast execution EDAs since they will be run repeatedly, thus we use UMDA and MIMIC, as the authors do in their experiments. On the other hand, Tsutsui [51] suggests setting the B_{ratio} constant to 0.0002 for EHBSA and NHBSA.

Table 2: Execution parameters set of the algorithms.

Parameter	Value
Population size	$10n$
Selection size	$10n/2$
Offspring size	$10n - 1$
Selection type	Ranking selection method
Elitism selection method	The best individual of the previous generation is guaranteed to survive
Stopping criterion	A maximum number of generations: $100n$

4.2 Results

Table 3 shows the average error and standard deviation for each type of problem[5]. This average error is calculated as the normalized difference between the best

[4] Éric Taillard's web page. http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html

[5] Average and standard deviation of the fitness value results obtained for all the instances tested (4 problem types × 24 instances) by the 14 algorithms can be found in http://www.sc.ehu.es/ccwbayes/members/jceberio/home/index.html

Table 3: Average error and standard deviation for each type of problem. Results in bold indicate the best average result found.

EDA		TSP	QAP	LOP	FSSP
UMDA	avg.	0.5077	0.0298	0.1511	0.0538
	dev.	0.3315	0.0153	0.0354	0.0308
MIMIC	avg.	0.6762	0.0390	0.1495	0.0351
	dev.	0.4371	0.0211	0.0351	0.0117
EBNA$_{BIC}$	avg.	0.5051	0.0310	0.1508	0.0545
	dev.	0.3438	0.0153	0.0358	0.0326
TREE	avg.	1.2554	0.0526	0.1761	0.0601
	dev.	0.8637	0.0318	0.0369	0.0223
UMDA$_c$	avg.	1.2792	0.2118	0.3303	0.1535
	dev.	0.9408	0.1420	0.0384	0.0322
EGNA$_{ee}$	avg.	1.1830	0.1655	0.3118	0.1424
	dev.	0.8886	0.1006	0.0481	0.0335
IDEA-ICE	avg.	1.2090	0.0801	0.1743	0.0734
	dev.	0.7610	0.0320	0.0322	0.0253
EHBSA$_{WT}$	avg.	**0.0037**	0.0256	0.1371	**0.0276**
	dev.	**0.0059**	0.0189	0.0328	**0.0232**
EHBSA$_{WO}$	avg.	0.1251	0.0653	0.2239	0.0626
	dev.	0.1544	0.0395	0.0400	0.0453
NHBSA$_{WT}$	avg.	1.0680	**0.0112**	**0.1366**	0.0277
	dev.	0.8659	**0.0130**	**0.0328**	0.0215
NHBSA$_{WO}$	avg.	0.3385	0.0222	0.1375	0.0326
	dev.	0.2443	0.0144	0.0326	0.0226
REDA$_{UMDA}$	avg.	2.0550	0.1426	0.2131	0.0986
	dev.	1.1909	0.0811	0.0467	0.0541
REDA$_{MIMIC}$	avg.	1.7410	0.1727	0.2794	0.1242
	dev.	1.3057	0.0963	0.0750	0.0443
OmeGA	avg.	1.2860	0.1347	0.3336	0.1281
	dev.	0.8513	0.0684	0.0750	0.0734

objective value obtained by the algorithm and the best known solution. Note that each entry in the table is the average of 240 values (24 instances $\times$ 10 runs). The lower the values are, the better the performance. Looking at these results, it can be seen that Tsutsui's EHBSA$_{WT}$ and NHBSA$_{WT}$ are by far the algorithms that provide the best results on average for every problem type. These results show the high influence of the templates (WT approaches) when sampling new individuals. At the opposite end, the results show that continuous codification EDAs, REDA approaches and OmeGA are, without doubt, the algorithms that perform the worst.

The results confirm the classification of the problem types given in Section 2 in relation with the contribution to the objective function of the indexes in the permutation. For instance, in TSP the relevant information for the calculation of the fitness function is given by the relative ordering of the indexes. The results in Table 3 show that, for the TSP, the algorithm that learns the adjacency of the indexes is that which better results obtains. On the other hand, NHBSA$_{WT}$ performs the best for the QAP as the probabilistic model is focused on estimating the probability distribution of the indexes in the absolute positions of the permutation. In problems for which the contribution of the index is mixed, such as LOP and FSSP, NHBSA$_{WT}$ and EHBSA$_{WT}$ have similar behavior.

In order to carry out a statistical analysis of the results obtained in the experiments, and following the suggestions given in [13], we decided to use non-parametric tests. The authors state that, for multiple-problem analysis –as is our case–, due to the dissimilarities in the results and the small size of the sample to be analyzed, a parametric test may result in erroneous conclusions. The descriptions given in Section 2 state that the semantic meaning of the permutations may change depending on the problem type we deal with, and thus we presume different performances of the EDAs for the TSP, QAP, LOP and FSSP. Due to this fact, we carried out individual statistical tests of the EDAs for each problem type.

The statistical analysis will be conducted in two steps. First, we will check if significant differences exist among the results obtained. For this purpose, Friedman's test will be used. This test ranks the algorithms for each problem, providing also an average rank value for each algorithm. These ranks can be consulted in Table 4.

Table 4: Average rankings of the algorithms. The lower the rank is, the better the performance.

EDA	TSP	QAP	LOP	FSSP
UMDA	5.87	4.16	5.12	6.04
MIMIC	7.70	5.41	4.91	4.37
EBNA$_{BIC}$	5.83	4.41	5.04	5.89
TREE	9.87	7.25	7.91	7.12
UMDA$_c$	9.97	13.66	12.95	13.12
EGNA$_{ee}$	8.64	11.41	12.12	12.08
IDEA-ICE	9.70	8.66	7.79	8.49
EHBSA$_{WT}$	1.27	3.79	1.68	1.91
EHBSA$_{WO}$	2.12	7.83	10.16	6.54
NHBSA$_{WT}$	7.52	1.5	1.56	2.04
NHBSA$_{WO}$	3.79	2.45	2.75	3.20
REDA$_{UMDA}$	11.91	10.54	8.87	10.52
REDA$_{MIMIC}$	10.37	12.62	11.08	11.58
OmeGA	10.37	11.25	13.00	12.04

The p-values resulting from applying Friedman's test are lower than 0.0001, which is below the level of significance considered ($\alpha = 0.05$). This means that there exist significant differences among the observed results. Once the rejection of the null hypothesis has been proved, a post-hoc method will be used to carry out all pairwise comparisons. Particularly, Shaffer's static procedure will be used, as suggested for such cases in [12]. Again, the significance level has been fixed to $\alpha = 0.05$. Results obtained from this procedure are represented in Figures 6, 7, 8 and 9 by means of critical difference diagrams. These diagrams draw the ranking of the algorithms and link with a horizontal line those groups of algorithms for which no significant differences were found (p-values higher than $\alpha = 0.05$).

The statistical analysis confirms the good performance of NHBSA and EHBSA, and particularly those algorithms that use the template strategy. Even if UMDA, MIMIC and EBNA$_{BIC}$ are not designed specifically to deal with permutation-based problems, being the closest to Tsutsui's algorithms, they show an acceptable performance.

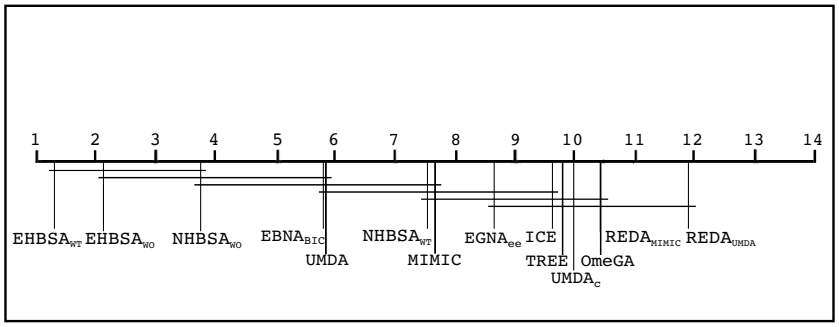

Fig. 6: Critical difference ranking diagram of TSP results.

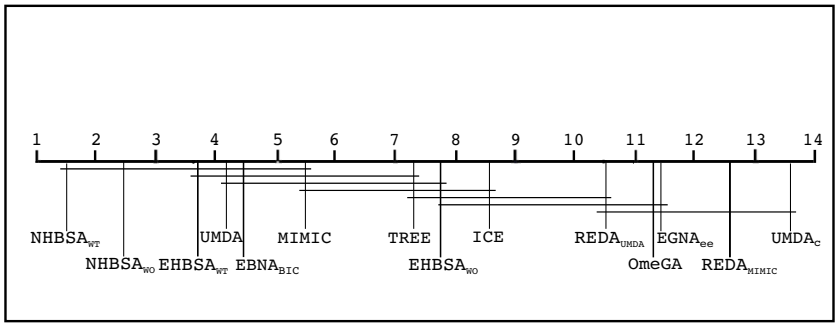

Fig. 7: Critical difference ranking diagram of QAP results.

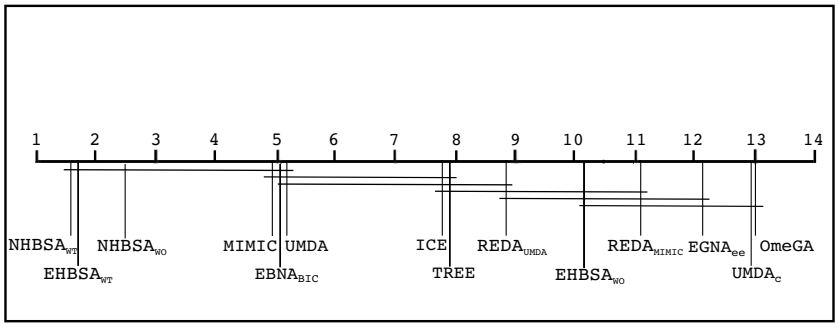

Fig. 8: Critical difference ranking diagram of LOP results.

Surprisingly, the results achieved by EBNA$_{BIC}$ and TREE do not outperform those achieved by UMDA. As stated in the literature, EBNA$_{BIC}$ and TREE algorithms are supposed to be more powerful than univariate algorithms, such as UMDA, since the first two learn (in)dependencies between variables and UMDA, instead, assumes independence between variables. In order to understand this behavior, we studied the probabilistic models learnt by EBNA$_{BIC}$ at each step. We realized that the learnt structure was an empty structure at all the times. And

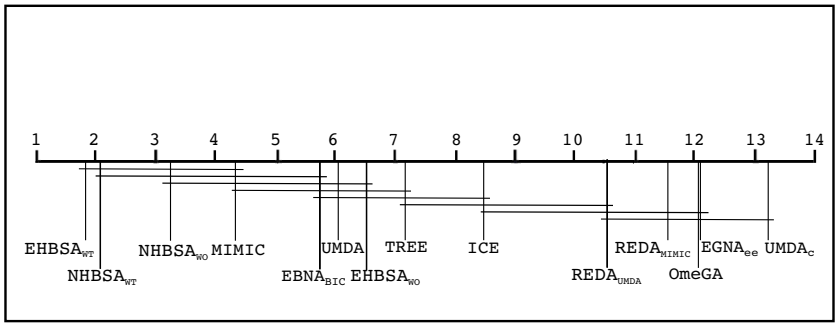

Fig. 9: Critical difference ranking diagram of FSSP results.

thus, the behavior of EBNA$_{BIC}$ results similar to that of UMDA. The reason why the learning algorithm does not learn any structure is as follows:

When we analyze the performance of these algorithms, it is important to note that they work in the probability space of size n^n (being n the size of the problem). This means that when we introduce an arc $X_i \rightarrow X_j$ in the probabilistic model, the number of parameters for codifying the local probability distribution of X_i is multiplied by n. EBNA$_{BIC}$ includes the arcs that improve the BIC score the most. This score is based on the maximum likelihood between nodes and a penalty term related to the complexity of structure. Taking into account the population size used in the experiments, when we try to add an arc, the increase in the likelihood is always smaller than the increase in the complexity, and therefore no arc is added. In the case of TREE, due to its design, the structure learning algorithm is forced to add those arcs that have the highest mutual information. However, as the population size does not provide enough information, the learnt tree turns out to be an over-fitted model.

IDEA-ICE shows a moderate efficiency, while REDA, OmeGA and the classical approaches for continuous domains, EGNA$_{ee}$ and UMDA$_c$ have obtained the worst results (this last performance may be due to the highly redundant encoding domain, as stated in several works).

In addition to these results, we consider it interesting to provide supplementary information about the number of times that the algorithms are able to get the best known solutions, and average number of iterations (generations) needed by them. Results in Table 5 show again that the highest success rates belong to the EHBSA and NHBSA approaches, taking note of the high influence of employing the template strategy. Surprisingly, there is not any EDA able to achieve optimal solutions for the LOP instances. In general, such low rates demonstrate the weakness of the compared EDAs to achieve optimum solutions in permutation-based problems.

In order to analyze the behavior of EDAs in relation to the iterations needed by each algorithm to obtain its best solution, Table 6 introduces the average and standard deviation of the number of generations needed to find those best solutions. Note that the number of iterations in crossover-based algorithms such as ICE and OmeGA is dramatically low comparing to the rest of EDAs. Another remarkable fact is the high deviation of the number of iterations.

Table 5: Success rate of the algorithms achieving optimal results.

%	TSP	QAP	LOP	FSSP
UMDA	1.6	0.8	0.0	0.0
MIMIC	0.4	2.1	0.0	0.0
EBNA$_{BIC}$	2.1	0.8	0.0	0.8
TREE	0.4	1.2	0.0	0.0
UMDA$_c$	4.2	0.0	0.0	0.0
EGNA$_{ee}$	3.7	0.0	0.0	0.0
IDEA-ICE	0.0	0.0	0.0	0.0
EHBSA$_{WT}$	74.2	9.6	0.0	11.2
EHBSA$_{WO}$	40.0	5.8	0.0	0.0
NHBSA$_{WT}$	10.4	31.7	0.0	5.4
NHBSA$_{WO}$	2.9	7.9	0.0	2.5
REDA$_{UMDA}$	0.4	0.0	0.0	0.0
REDA$_{MIMIC}$	1.2	0.0	0.0	0.0
OmeGA	0.0	0.0	0.0	0.0

Table 6: Average and standard deviation of the iterations required to find the best solution. REDA algorithms are not included in this analysis due to the recursive strategy that follows the EDA.

EDA		TSP	QAP	LOP	FSSP
UMDA	avg.	5362.78	698.85	4472.73	2973.22
	dev.	3965.94	506.18	884.72	2536.14
MIMIC	avg.	1993.53	531.11	2430.76	775.17
	dev.	1328.35	365.32	521.73	689.54
EBNA$_{BIC}$	avg.	5386.81	717.50	4509.50	3105.74
	dev.	4123.52	529.23	827.77	2626.11
TREE	avg.	5896.24	1487.35	4613.75	2959.04
	dev.	3522.10	840.35	703.43	2246.44
UMDA$_c$	avg.	2440.82	591.67	563.35	447.93
	dev.	2077.21	485.06	722.65	675.64
EGNA$_{ee}$	avg.	3353.38	1123.40	1036.17	1848.13
	dev.	2163.48	499.00	1036.17	1523.86
IDEA-ICE	avg.	258.35	39.32	144.25	289.24
	dev.	224.63	21.21	37.86	341.13
EHBSA$_{WT}$	avg.	3526.13	1702.01	4426.07	3972.13
	dev.	3501.54	817.43	1040.97	2772.07
EHBSA$_{WO}$	avg.	4298.46	1571.58	3724.33	3432.13
	dev.	3719.40	588.09	486.51	2146.82
NHBSA$_{WT}$	avg.	6266.87	1645.08	4254.60	3882.62
	dev.	3592.48	903.79	1099.40	2719.95
NHBSA$_{WO}$	avg.	3979.47	536.72	2571.42	2938.68
	dev.	3610.47	390.46	1020.63	2767.95
OmeGA	avg.	31.70	21.66	32.38	29.58
	dev.	5.17	12.68	1.92	10.26

As a general conclusion, it must be highlighted that those approaches designed to handle the space of permutations are those that obtain the best results. Moreover, NHBSA and EHBSA use only 1-order and index adjacency probabilistic models, which theoretically are too simple to efficiently comprise the underlying probability distribution. These results should encourage the research community to follow this direction, trying to design more effective probability models over the

space of permutations. In the next section, we discuss some ideas that could be useful for this purpose.

5 Discussion

As commented in the previous sections, in order to deal with permutation-based problems the proposed EDA approaches are (i) adaptations of algorithms designed for integer-based problems, (ii) transforming a permutation problem into a continuous optimization problem and then using EDAs designed for continuous domains or (iii) ad hoc approaches using first-order statistics. Contrary to integer problems, where the community has used most of the mechanisms provided for the researchers working in machine learning, and statistics such as graphical models, kernels, etc, this has not been the case for permutations. In this section we give a brief review of the most common probabilistic models to deal with permutation spaces. We also point out some ideas on the use of those models in EDAs, particularly we briefly analyze the learning and sampling algorithms of the models.

Since the most common application of permutations is that of ranking, we will use these two words, permutations and ranking, interchangeably throughout this section.

5.1 Models based on marginals

When working with samples of permutations, the trivial approach consists of maintaining the information relative to the first order marginals, which express the probability of item i being at position j. This information can be stored in $O(n^2)$ space by using an nxn matrix. A natural extension consists of storing higher order marginals. Such marginals correspond to the probability of a specific set of items $(i_1, ..., i_k)$ being at specific positions $(j_1, ..., j_k)$. One may also be interested in maintaining the probability of an item being at the position right after another item without specifying a particular position. In fact, this is the kind of information used in the edge histogram model, [46, 50], while the node histogram model, [51], maintains the first order marginals.

This representation is not only compact, but it is also easy to learn. By using the first order marginals, statistics such as the mode can be computed. However, when it comes to sampling - a necessary step in EDAs- further information about the distribution is required. In [51] a distribution with the given marginals is sampled. However, the actual distribution is unknown. There does not seem to be a closed form for it and it is not clear which are the properties of such a distribution. Actually, there can be infinitely many probability distributions that have a given first order marginal probability matrix. Therefore, among all those distributions how can one select the 'correct' one? A common approach in statistics and machine learning is to consider the maximum entropy distribution, i.e. the distribution that, by having those marginal probabilities, has the highest uncertainty. This is the procedure followed by [1] which showed that such a distribution happens to have the simple expression given by

$$P(\sigma) = exp(\sum_{i=1}^{n} Y_{i,\sigma(i)} - 1).$$

where $Y \in \mathbb{R}^{nxn}$. Unfortunately, it is also shown in [1] that obtaining the Y matrix is #P-hard. Nevertheless, they also give an approximation algorithm which runs in polynomial time for computing the Y parameter.

5.2 Plackett-Luce model

The Plackett-Luce distribution takes its name from the combination of the independent work carried out by Plackett [41] and Luce [31]. Luce's model describes a sequential ranking generator method in which the items are sampled from the first to the last position (i.e., from the most to the least preferred item). The parameter space consists of n positive weights $(w_1, ..., w_n)$ that sum 1. These probabilities are used to sample the first position of the rank, with $P(\sigma(1) = j) = w_j$. The following positions, $\{2, ..., n\}$, are sampled without replacement until a complete ranking is obtained. Note that, in order to sample position i of a permutation by using Luce's model, the probability of selecting item j_1 over j_2 does not depend on the weights of the rest of the items in the set.

The above model induces a distribution over all possible rankings. It was first used by Plackett and can be written as follows:

$$P(\sigma) = \prod_{i=1}^{n-1} \frac{w_{\sigma(i)}}{\sum_{j=i}^{n} w_{\sigma(j)}}$$

Due to the Markovian nature of the model, it is not easy to make inference over sets of items such as $P(\sigma(n) = i)$. Regarding the learning process of the n parameters of a Plackett-Luce distribution, one can find in the literature methods based on maximum likelihood estimation [19] and Power EP (expectation propagation) [16]. Once the distribution parameters are known, the sampling procedure consists of following Luce's model.

5.3 Mallows model

The Mallows model [32] is a distance based exponential model. The most commonly used metric is the Kendall tau distance, which, given two permutations σ_1 and σ_2, counts the total number of pairwise disagreements between both of them i.e., the minimum number of adjacent swaps to convert σ_1 into σ_2. Formally, it can be written as

$$\tau(\sigma_1, \sigma_2) = |\{(i,j) : i < j, (\sigma_1(i) < \sigma_1(j) \wedge \sigma_2(i) > \sigma_2(j)) \\ \vee (\sigma_2(i) < \sigma_2(j) \wedge \sigma_1(i) > \sigma_1(j))\}|.$$

The above metric can be equivalently written as

$$\tau(\sigma_1, \sigma_2) = \sum_{j=1}^{n-1} V_j(\sigma_1, \sigma_2)$$

where $V_j(\sigma_1, \sigma_2)$ is the minimum number of adjacent swaps to set in the j-th position of σ_1, $\sigma_1(j)$, the value $\sigma_2(j)$.

The Mallows model makes use of this metric to define an exponential probability model for permutations which can be defined by two parameters: The central permutation, σ_0, and the spread parameter, θ. It can be written as

$$P(\sigma) \propto exp(-\theta\tau(\sigma, \sigma_0)).$$

When the spread parameter is $\theta > 0$, the central permutation, σ_0, is the one with the highest probability value and the probability of the other $n! - 1$ permutations is inversely proportional to their distance to the central permutation and the spread parameter θ. Because of these two properties, the Mallows distribution is considered analogous to the Gaussian distribution on the space of permutations.

Among its many extensions, the *generalized Mallows* (GM) model [11] is that which has received an special attention by the community. This extension makes use of n parameters: The central permutation, σ_0, and $n-1$ spread parameters, $\theta_1, ..., \theta_{n-1}$. The probability distribution over each distinct ranking is as follows:

$$P(\sigma) \propto exp(-\sum_{j=1}^{n-1} \theta_j V_j(\sigma, \sigma_0))$$

Note that when the $n-1$ parameters θ_j are constrained to be equal, the generalized Mallows reduces to the Mallows model.

The typical way to learn the parameters of the distribution of a given sample of permutations is to maximize the likelihood of these parameters. Let $\{\sigma_1, ..., \sigma_N\}$ be the given sample. Then, its log-likelihood is given by

$$log\ l(\sigma_1, ..., \sigma_N | \sigma_0, \boldsymbol{\theta}) = -N \sum_{j=1}^{n-1} (\theta_j \bar{V}_j + \log \psi_j(\theta_j))$$

where $\bar{V}_j = \sum_{i=1}^{N} V_j(\sigma_i, \sigma_0)/N$, i.e. $\bar{V}_j$ denotes the observed mean for V_j and $\psi_j(\theta_j)$ refers to the normalization constant. Note that by setting equal values for the spread parameters θ_i for every $i = \{1, ..., n-1\}$, we obtain the expression of the maximum likelihood estimator for the Mallows model. For both Mallows and Generalized Mallows models the problem of finding the MLE for σ_0 and θ is NP-hard. Particularly, the problem of finding the central permutation or consensus ranking is called rank aggregation and is equivalent to finding the MLE estimator of σ_0. One can find several methods for solving this problem, both exact [9] and heuristic [33]. Therefore, if any of these models is applied to EDAs, at each step a NP-hard problem must be solved. However, this is also the case for integer-based problems, where at each step a Bayesian network is learned. Although a NP-complete problem is solved at each generation, EDAs have been successfully applied to integer-based problems. Note that EDAs do not need to solve the Bayesian network learning problem to optimality. On the contrary, the sampling process can be easier for the Mallows model than for the Generalized Mallows model. Note that while the former assigns equal probability values for permutations at equal distance to the central permutation, the latter does not. An obvious way to sample a Mallows model is by using a Markov Chain Monte Carlo method such us a Gibbs sampler.

However, the main drawback of these two models is that their unimodal nature makes the representation of distributions of multimodal optimization problems impossible.

5.4 Non-parametric models

As we have already stated, the unimodality Mallows and Generalized Mallows can be a drawback to deal with multimodal optimization problems. However, they can be used to build a multimodal distribution. In [27] a multimodal non-parametric estimator is built by placing Mallows kernels on the top of the elements of a given sample of permutations. The non-parametric estimator of such a distribution is given by the following equation

$$\hat{p}(\sigma) \propto \sum_{i=1}^{m} exp(-c\, d(\sigma, \sigma_i))$$

where c is a spread parameter.

The proposed estimator is consistent. Moreover, by exploiting the underlying combinatorial properties of permutations the estimator can be efficiently computed. It has been successfully applied to partial ranking problems. On the other hand, the use of a single spread parameter c, which has to be manually set, can limit the quality of the resulting estimator.

6 Conclusions and Future work

In this paper we have reviewed the existing EDA approaches for solving permutation-based optimization problems. We have stated by means of examples of permutation-based problems that, although all solutions are encoded as permutations, their meanings change from one problem to another. We classified the existing EDA approaches for solving permutation-based problems in three groups: (i) adaptations of algorithms designed for integer-based problems, (ii) transforming a permutation-based problem into a continuous optimization problem and then using EDAs designed for continuous domains and (iii) ad hoc approaches using different strategies. The experimental analysis carried out showed that the best results are given by those EDA approaches that implement ad hoc designs. On the contrary, continuous and REDA approaches are those which perform the worst. The experimental analysis also stated that the integer-based EDAs yield good solutions. In fact, those algorithms that find best solutions, EHBSA and NHBSA, are the only EDAs that learn probabilistic models taking into account the characteristics of permutations. This fact suggests that the future work that the research community of EDAs should follow is the use of the probabilistic model over permutation spaces. Following this idea we have introduced several models that could be used with EDAs in order to solve permutation-based problems.

7 Acknowledgments

We gratefully acknowledge the generous assistance and support of Prof. S. Tsutsui and Prof. P. Bosman in this work. This work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2008-06815-C02-01, TIN2010-14931 and Consolider Ingenio 2010 - CSD 2007 - 00018 projects (Spanish Ministry of Science and Innovation) and

COMBIOMED network in computational biomedicine (Carlos III Health Institute). Josu Ceberio holds a grant from Basque Goverment.

References

1. S. Agrawal, Z. Wang, and Y. Ye. Parimutuel Betting on Permutations. In *Internet and Network Economics*, volume 5385 of *Lecture Notes in Computer Science*, pages 126–137. Springer Berlin / Heidelberg, 2008.
2. J. C. Bean. Genetic Algorithms and Random Keys for Sequencing and Optimization. *INFORMS Journal on Computing*, 6(2):154–160, 1994.
3. E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, and C. Boeres. Inexact graph matching by means of estimation of distribution algorithms. *Pattern Recognition*, 35(12):2867–2880, 2002.
4. P. A. N. Bosman and D. Thierens. Expanding from Discrete to Continuous Estimation of Distribution Algorithms: The IDEA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M. Guervós, and H. P. Schwefel, editors, *PPSN*, volume 1917 of *Lecture Notes in Computer Science*, pages 767–776. Springer, 2000.
5. P. A. N. Bosman and D. Thierens. Crossing the road to efficient IDEAs for permutation problems. In L. S. et al. et al., editor, *Genetic and Evolutionary Computation Conference, GECCO 2001, Proceedings, San Francisco, California, USA, 2001*, pages 219–226. Morgan Kaufmann, 2001.
6. A. E. I. Brownlee, M. Pelikan, J. A. W. McCall, and A. Petrovski. An application of a multivariate estimation of distribution algorithm to cancer chemotherapy. In C. Ryan and M. Keijzer, editors, *GECCO*, pages 463–464. ACM, 2008.
7. S. Chen and M. Chen. Bi-Variate Artificial Chromosomes with Genetic Algorithm for Single Machine Scheduling Problems with Sequence-Dependent Setup Times. In *Proceedings of the Congress on Evolutionary Computation*, 2011.
8. C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, May 1968.
9. W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. In *Proceedings of the 1997 conference on Advances in neural information processing systems 10*, NIPS '97, pages 451–457, Cambridge, MA, USA, 1998. MIT Press.
10. J. S. De Bonet, C. L. Isbell, and P. Viola. MIMIC: Finding Optima by Estimating Probability Densities. In *Advances in Neural Information Processing Systems*, volume 9. M. Mozer, M. Jordan and Th. Petsche eds., 1997.
11. M. A. Fligner and J. S. Verducci. Distance based ranking Models. *Journal of the Royal Statistical Society*, 48(3):359–369, 1986.
12. S. Garcia and F. Herrera. An Extension on "Statistical Comparisons of Classifiers over Multiple Data Set" for all Pairwise Comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.
13. S. Garcia, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 15(6):617–644, 2009.
14. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison/Wesley, Reading MA, 1989.
15. D. E. Goldberg and R. L. Jr. AllelesLociand the Traveling Salesman Problem. In *ICGA*, pages 154–159, 1985.
16. J. Guiver and E. Snelson. Bayesian inference for Plackett-Luce ranking models. In *International Conference on Machine Learning (ICML 2009)*, ICML '09, pages 377–384. ACM, 2009.
17. J. Gupta and J. E. Stafford. Flow shop scheduling research after five decades. *European Journal of Operational Research*, (169):699–711, 2006.
18. M. Henrion. Propagating uncertainty in Bayesian networks by Probabilistic Logic Sampling. In J. F. Lemmer and L. N. Kanal, editors, *UAI*, pages 149–164. Elsevier, 1986.
19. D. R. Hunter. MM Algorithms for Generalized Bradley-Terry Models. *The Annals of Statistics*, 32(1):384–406, 2004.
20. B. Jarboui, M. Eddaly, and P. Siarry. An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. *Computers & OR*, 36(9):2638–2646, 2009.

21. S. Jiang, A. Ziver, J. Carter, C. Pain, A. Goddard, S. Franklin, and H. Phillips. Estimation of Distribution Algorithms for nuclear reactor fuel management optimisation. *Annals of Nuclear Energy*, 33(11-12):1039–1057, 2006.
22. D. Knjazew and D. E. Goldberg. Omega - ordering messy ga: Solving permutation problems with the fast genetic algorithm and random keys. In *GECCO*, pages 181–188, 2000.
23. T. C. Koopmans and M. J. Beckmann. Assignment Problems and the Location of Economic Activities. Cowles Foundation Discussion Papers 4, Cowles Foundation for Research in Economics, Yale University, http://ideas.repec.org/p/cwl/cwldpp/4.html, 1955.
24. P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Combinatorial optimization by learning and simulation of Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI 2000*, pages 343–352, Stanford, CA, USA, 2000.
25. P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In *Proceedings of the Workshop in Optimization by Building and using Probabilistic Models. A Workshop within the 2000 Genetic and Evolutionary Computation Conference, GECCO 2000*, pages 201–204, Las Vegas, Nevada, USA, 2000.
26. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.
27. G. Lebanon and Y. Mao. Non-Parametric Modeling of Partially Ranked Data. *Journal of Machine Learning Research (JMLR)*, 9:2401–2429, 2008.
28. H. Liu, L. Gao, and Q. Pan. A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. *Expert Syst. Appl.*, 38:4348–4360, April 2011.
29. J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
30. J. A. Lozano and A. Mendiburu. Estimation of Distribution Algorithms applied to the job schedulling problem. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.
31. D. Luce R. *Individual Choice Behavior*. Wiley, New York, 1959.
32. C. L. Mallows. Non-null ranking models. *Biometrika*, 44(1-2):114–130, 1957.
33. B. Mandhani and M. Meila. Tractable search for learning exponential models of rankings. In *Artificial Intelligence and Statistics (AISTATS)*, April 2009.
34. A. Mendiburu, J. A. Lozano, and J. Miguel-Alonso. Parallel Implementation of EDAs Based on Probabilistic Graphical Models. *IEEE Transactions on Evolutionary Computation*, 9(4):406–423, 2005.
35. A. Mendiburu, J. Miguel-Alonso, J. A. Lozano, M. Ostra, and C. Ubide. Parallel EDAs to create multivariate calibration models for quantitative chemical applications. *J. Parallel Distrib. Comput.*, 66(8):1002–1013, 2006.
36. H. Mühlenbein and G. Paaß. From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, 1996.
37. M. Pelikan and D. E. Goldberg. Hierarchical problem solving and the Bayesian optimization algorithm. In D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 267–274, San Francisco, CA, 2000. Morgan Kaufmann Publishers.
38. M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.
39. M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
40. M. Pelikan, S. Tsutsui, and R. Kalapala. Dependency Trees, Permutations, and Quadratic Assignment Problem. Technical report, Medal Report No. 2007003, 2007.
41. R. L. Plackett. The Analysis of Permutations. *Journal of the Royal Statistical Society*, 24(10):193–202, 1975.
42. V. Robles, P. de Miguel, and P. Larrañaga. Solving the Traveling Salesman Problem with EDAs. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

43. T. Romero and P. Larrañaga. Triangulation of Bayesian networks with recursive Estimation of Distribution Algorithms. *Int. J. Approx. Reasoning*, 50(3):472–484, 2009.
44. R. Sagarna and J. A. Lozano. Scatter Search in software testing, comparison and collaboration with Estimation of Distribution Algorithms. *European Journal of Operational Research*, 169(2):392–412, 2006.
45. R. Santana, P. Larrañaga, and J. A. Lozano. Protein folding in simplified models with Estimation of Distribution Algorithms. *IEEE Transactions On Evolutionary Computation*, 12(4):418–438, 2008.
46. S. Tsutsui. Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram. In *PPSN*, pages 224–233, 2002.
47. S. Tsutsui. A Comparative Study of Sampling Methods in Node Histogram Models with Probabilistic Model-Building Genetic Algorithms. In *IEEE International Conference on Systems, Man, and Cybernetics. October 8-11, 2006, Taipei, Taiwan*, volume 4, pages 3132–3137, 2006.
48. S. Tsutsui. Effect of Using Partial Solutions in Edge Histogram Sampling Algorithms with Different Local Searches. In *SMC*, pages 2137–2142, 2009.
49. S. Tsutsui and M. Miki. Solving Flow Shop Scheduling Problems with Probabilistic Model-Building Genetic Algorithms using Edge Histograms. In *4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL 02)*, pages 776–780, 2002.
50. S. Tsutsui, M. Pelikan, and D. E. Goldberg. Using Edge Histogram Models to Solve Permutation Problems with Probabilistic Model-Building Genetic Algorithms. Technical report, IlliGAL Report No. 2003022, 2003.
51. S. Tsutsui, M. Pelikan, and D. E. Goldberg. Node Histogram vs. Edge Histogram: A Comparison of PMBGAs in Permutation Domains. Technical report, Medal, 2006.
52. S. Tsutsui and G. Wilson. Solving Capacitated Vehicle Routing Problems Using Edge Histogram Based Sampling Algorithms. In *Proceedings of the IEEE Conference on Evolutionary Computation, Portland, Oregon (USA)*, pages 1150–1157, 2004.
53. B. Yuan, M. E. Orlowska, and S. W. Sadiq. Finding the optimal path in 3d spaces using EDAs - the wireless sensor networks scenario. In *ICANNGA (1)*, pages 536–545, 2007.
54. Q. Zhang, J. Sun, E. Tsang, and J. Ford. Combination of Guided Local Search and Estimation of Distribution Algorithm for Solving Quadratic Assignment Problem. In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 42–48, 2004.
55. Q. Zhang, J. Sun, E. Tsang, and J. Ford. Estimation of Distribution Algorithm with 2-opt Local Search for the Quadratic Assignment Problem. *Studies in Fuzziness and Soft Computing*, 192/2006:281–292, 2006.
56. A. A. Zhigljavsky. *Theory of Global Random Search*. Kluwer Academic Publishers, 1991.