A Preliminary Study on EDAs for Permutation Problems Based on Marginal-based Models

Josu Ceberio
Intelligent Systems Group
University of the Basque Country EHU/UPV
Donostia, Gipuzkoa, España
jceberio001@ikasle.ehu.es

Alexander Mendiburu
Intelligent Systems Group
University of the Basque Country EHU/UPV
Donostia, Gipuzkoa, España
alexander.mendiburu@ehu.es

Jose A. Lozano
Intelligent Systems Group
University of the Basque Country EHU/UPV
Donostia, Gipuzkoa, España
ja.lozano@ehu.es

ABSTRACT

Estimation of Distribution Algorithms are a class of evolutionary algorithms characterized by the use of probabilistic models. These algorithms have been applied successfully to a wide set of artificial and real-world problems, achieving competitive results in most scenarios. Nevertheless, there are some problems whose solutions can be naturally represented as a permutation, for which EDAs have not been extensively developed. Although some work has been done in this area, 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 this paper, we present an EDA that learns probability distributions over permutations. Particularly, our approach is based on the use of k-order marginals. In addition, we carry out some preliminary experiments over classical permutation-based problems in order to study the performance of the proposed k-order marginals EDA.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Probabilistic algorithms; I.2.8 [**Artificial Intelligence**]: Metrics—*Problem Solving, Control Methods, and Search-Heuristic methods*

General Terms

Algorithm, Performance, Theory

Keywords

Problabilistic Models, Permutations, Estimation Distribution Algorithms, Marginals, Optimization

1. INTRODUCTION

Estimation of Distribution Algorithms (EDAs) [17, 12, 18] are a set of methods that belong to the field of Evolutionary Computation. Similarly to Genetic Algorithms [7], EDAs initialize a random set of individuals and, applying different operators, these individuals are modified to obtain optimal solutions. However, instead of using crossover and mutation operators to generate new individuals, EDAs use probabilistic models, taking advantage of the interrelations between the different variables that represent the individuals. Introduced in [17], they have attracted the interest of the Evolutionary Algorithms research community, which has developed a wide set of different algorithms and techniques in the last decade. EDA approaches have been able to successfully solve different artificial and real-world problems, both in the discrete and continuous domains [13, 14, 19, 20, 25].

In this work, we are interested in the solution of a specific subset of NP-hard optimization problems. We refer to those problems whose solutions can be naturally represented as a permutation. Examples of this kind of problems are: Traveling Salesman Problem (TSP), Quadratic Assignment Problem (QAP), Linear Ordering Problem (LOP), Flow Shop Scheduling Problem (FSSP), etc. In order to apply EDAs, a new challenge has to be dealt with: the codification of a probability distribution over the space of permutations. Unfortunately, representing and learning a probability distribution over a set of permutations is much more difficult than doing the same over integer or real-valued spaces.

Different approaches have been given in the literature to deal with permutation problems by means of EDAs. Most of the approaches propose modifying EDAs designed originally for integer spaces to deal with permutations [4, 23, 24, 16, 21]. In order to guarantee feasible solutions, that is, each value appears once and only once in the solution, the sampling procedure is modified accordingly. This change is necessary because these EDAs do not estimate the probabilistic model over the permutation space but over the integer space. Other EDAs carry out a search in a continuous space [5, 23, 16]. Each n-dimensional real vector is decoded as a permutation. These proposals also present several drawbacks, the main one being the difficulty to translate the relations between the variables in the permutation space to the continuous space and vice versa.

A couple of approaches which take into account the real nature of permutations by setting a probabilistic model over the permutation space are those of Tsutsui et al. [28, 29, 30]. Basically the authors consider 1-order and 2-order marginals over that space. Inspired by this concept, we propose a new EDA that implements a probabilistic model which learns k-order marginal probabilities between the variables ($k \geq 1$).

GECCO'11, July 12–16, 2011, Dublin, Ireland.

These models should be able to identify better the relations (dependencies) between the variables of the problem as k increases.

The remainder of the paper is as follows: In Section 2, we make a review of the most relevant existing EDAs for permutation problems. Our proposal, k-order marginals based EDA is introduced in Section 3. The current approach is tested with well-known permutation-based optimization problems in Section 4. Section 5 introduces a discussion on marginal based EDA approaches for permutation represented problems. Finally, conclusions are drawn in Section 6.

2. ESTIMATION OF DISTRIBUTION ALGORITHMS APPLIED TO PERMUTATION PROBLEMS

2.1 Permutation-based optimization problems

As introduced previously, many optimization problems find a natural representation of the solution as permutations. In combinatorics, permutations are 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 $\sigma_i = j$.

Although the solution of many problems are codified as permutations, the way that the information is represented by the indexes in the permutation may differ from one problem to another. Since the probabilistic model aims to learn the semantic information of the set of selected solutions, the performance of the algorithm is closely related to the codification of the individuals and to the ability of the probabilistic model to exploit the information given by those individuals.

For example, in the well-known Traveling Salesman Problem (TSP) [8], where the goal is to go over a number of cities looking for the shortest path in terms of time, distance, or any similar criterion, a solution is given by a sequence of the cities. In a problem with 4 cities, $\sigma = (3, 2, 4, 1)$ would be a solution, indicating that the initial city is 3, then 2, 4, 1, coming finally back to 3. Note that, $\sigma' = (1, 3, 2, 4)$ represents also the same city tour that σ does. We conclude that the relevant information is given by the relative ordering of the indexes in the permutation.

However, it does not happen for other types of problems such as Flow Shop Scheduling Problem [9] which consist of scheduling n jobs of m steps in m machines minimizing the processing time of all the jobs. In a problem of 4 jobs and 3 machines, the permutation $\sigma = (1, 2, 3, 4)$, represents that job 1 is processed first, next job 2 and so on. In this problem the objective value can not be decomposed and depends on the absolute position of all the jobs as well as the whole order of the jobs.

2.2 EDAs

As with EAs, EDAs have also been applied to permutation problems. For this purpose, different approaches in both discrete and continuous domains have been designed and/or adapted to deal with permutation-based problems.

Initial approaches tried to use EDAs designed for discrete domain codifying individuals by using path representation [23]. In this case, each variable can take a value in the range $\{1 \ldots n\}$, being n the problem size. Algorithms such as Univariate Marginal Distribution Algorithm (UMDA) [12], Estimation of Bayesian Networks Algorithm (EBNAs) [4], or Mutual Information Maximization for Input Clustering (MIMIC) [4] have been implemented with this encoding of the individuals for permutation-based problems.

However, these approaches have some drawbacks. As those EDAs do not deal with probabilities in the space of permutations, it is necessary to modify the sampling step to guarantee that each value will appear once and only once. In order to overcome this deficiency, different proposals have been presented. Nevertheless, the most common method to sample a probabilistic model in EDAs is the Probabilistic Logic Sampling algorithm [10]. 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 the 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. This sampling method always guarantees feasible solutions, however the information kept by the probabilistic models is modified at each step.

As regards the continuous domain, the most extended encoding is that based on random keys, introduced by Bean [3]. To encode a permutation of length n, each integer value σ_i in $\sigma = (\sigma_1, \ldots, \sigma_n)$ is assigned a value (key) from some real domain, which is usually taken to be the interval [0, 1]. Subsequently, the numbers in σ are sorted according to the keys to get the resulting permutation. The main advantage of random keys is that they always provide feasible solutions, since each encoding represents a permutation. Nevertheless, solutions are not processed in the permutation space, but in the largely redundant real-valued space. For example, strings (0.2, 0.1, 0.7) and (0.4, 0.3, 0.5) represent the same permutation.

This random key strategy has been jointly used with different EDAs for real-valued problems [5, 23]. In [15] 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).

The drawbacks of these direct approaches in the discrete and continuous domains encouraged the research community to implement specific EDAs for solving permutation-based problems. Bosman and Thierens introduced the ICE [5, 6] algorithm to overcome the bad performance of random keys in permutation optimization. The authors proposed using a crossover operator [5] instead of sampling new individual blocks from a Gaussian probability density function. This crossover operator reflects the dependency information learned in a factorization. In [5], marginal product factorizations are used, or equivalently, a product of multivariate joint probability density functions.

In [24] a new framework for EDAs called Recursive EDAs (REDA) is introduced. REDA is an optimization strategy of k stages which, at each stage, divides the set of the variables of an individual into two subsets and calls recursively to REDA to optimize each set separately. In the first call, one subset of nodes is fixed to evolve recursively over the other, and in the second step the opposite process is performed. The algorithm stops when the number of non-fixed variables

Table 1: 2-order marginals matrix

Positions \ Index Combinations	(1,2)	(1,3)	(1,4)	(2,1)	(2,3)	(2,4)	(3,1)	(3,2)	(3,4)	(4,1)	(4,2)	(4,3)
(1,2)	0.20	0.05	0.10	0.05	0.10	0.05	0.05	0.10	0.05	0.05	0.10	0.10
(1,3)	0.05	0.20	0.10	0.10	0.05	0.05	0.05	0.05	0.10	0.10	0.10	0.05
(1,4)	0.10	0.10	0.15	0.05	0.05	0.10	0.10	0.05	0.05	0.10	0.05	0.10
(2,3)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.10	0.05	0.05	0.05	0.10
(2,4)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.05	0.10	0.05	0.10	0.05
(3,4)	0.05	0.10	0.10	0.10	0.05	0.05	0.05	0.10	0.15	0.10	0.05	0.10

in the individual reaches a minimum threshold. As REDA is a general strategy, any EDA algorithm and encoding could be used. Particularly, the authors propose using EDAs such as UMDA or MIMIC that permit keeping the computation cost feasible. Regarding the encoding of the individuals, the authors propose using two different codifications, one based on random keys for continuous approaches, and the other a new codification for the discrete domain introduced by themselves. This new codification allows to learn probability distributions over permutations setting 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 $n!$.

Tsutsui et al. [28, 29, 30] attempted to create models over the space of permutations with different approaches. The first approach was called Edge Histogram Based Sampling Algorithm (EHBSA) [28, 29]. EHBSA builds an Edge Histogram Matrix (EHM), which counts the times that index i and index j appear consecutively in the permutation. The probabilistic model estimated from the adjacency of indexes could be considered as a sub group of 2-order marginals. A second approach called Node Histogram Based Sampling Algorithm (NHBSA) [30] was introduced later by the authors. While the EHBSA learns the adjacency of the indexes, the new approach focuses on learning the distribution of the indexes across the position of the permutation. As a result, a Node Histogram Matrix (NHM) is built, which in essence is a probabilistic model of 1-order marginals.

Regarding simulating new individuals, both algorithms sample the marginals matrix to create new solutions. In addition, the authors propose using a template-based sampling method to create new solutions. The method consists of choosing an individual from the previous generation, dividing it in c random segments and sampling the indexes of one of the segments, leaving the remaining indexes unchanged.

Tsutsui's approaches demonstrated that these algorithms based on 1-order and 2-order marginals yield better results than the previously mentioned EDA proposals in the solution of permutation-based problems. Extending this idea, we propose a k-order marginals-based probabilistic model able to determine the relations of the variables assuming k interactions for each variable.

3. K-ORDER MARGINALS EDA

Further developing the idea introduced by Tsutsui et al. [28, 29, 30], we present a k-order marginals based EDA which assumes k-length interactions between all variables in the problem.

Our model is given by a matrix M_k, where each entry of the matrix is given by each of the possible k-order marginal probabilities:

$$P(\sigma_{i_1} = j_1, \ldots, \sigma_{i_k} = j_k)$$

Table 2: Algorithm complexity progression by n and k

$n \backslash k$	1	2	3	4
10	100	4050	86400	1058400
20	400	72200	7797600	$5.63\text{x}10^{8}$
50	2500	3001250	$2.30\text{x}10^{9}$	$1.27\text{x}10^{12}$
100	10000	49005000	$1.56\text{x}10^{11}$	$3.69\text{x}10^{14}$

Table 1 shows an example of $n = 4$ (problem size) and $k = 2$ marginals matrix. The rows of the matrix represent the combinations of positions of the permutation and the columns represent the combinations of indexes. For instance, the entry $m_{2,4}$ of the matrix gives the joint probability that index 2 is in position 1 and index 1 is in position 3, i.e. $P(\sigma_1 = 2, \sigma_3 = 1)$.

It is important to take into account the complexity in terms of memory requirements of these models. For the matrix M_k the number of rows is given by the number of k size subsets from $\{1, \ldots, n\}$:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

Regarding the number of columns, this is given by the number of the ordered k size subsets from $\{1, \ldots, n\}$:

$$\frac{n!}{(n-k)!} \tag{2}$$

Therefore, we can say that the complexity of the algorithm in terms of memory is given by:

$$O((n/k)^{2k}) \tag{3}$$

Table 2 shows the number of entries of matrix M_k for different n and k combinations. As can be seen, this approach can be used for small values of k and/or n, because otherwise the memory requirements make it unaffordable. However, this is in agreement with algorithms based on k-order marginals in binary spaces where, while the memory requirements are lower, the number of operations are on the same order of complexity [11].

In the following sections we introduce the learning algorithm of the M_k matrix and the sampling technique used to generate new individuals.

3.1 Learning marginals

Each of the entries of the matrix M_k is calculated using maximum likelihood estimation from the set of selected individuals. Basically, it means that the probability $P(\sigma_{i_1} = j_1, \ldots, \sigma_{i_k} = j_k)$ is calculated from the number of times that the configuration $(\sigma_{i_1} = j_1, \ldots, \sigma_{i_k} = j_k)$ appears in the set of selected individuals. In our experiments we have used Laplace correction to avoid probabilities equal to 0 or equal to 1.

Figure 1: Sampling algorithm of M_k marginals matrix to create new individuals.

1. Initialize an individual $S = (-, -, ..., -)$.
2. Set order counter $r \leftarrow 1$.
3. Get the list of rows P from M_r that match S.
4. Uniformly at random pick a row from P.
5. Get the list of columns V from M_r that match S.
6. Modify the probability distribution in the row for the list of columns V.
7. Sample a column from V.
8. Update the individual S with the new assignment.
9. If $r == k$, go to Step 11.
10. Update order counter $r \leftarrow r + 1$.
11. If S is not completed, go to Step 3.
12. Return S.

3.2 Sampling

The sampling of new individuals must be carried out meeting the constraints of permutations, that is, each index must appear only once in the solution. The sampling procedure is carried out according to the algorithm defined in Figure 1. The process starts by randomly selecting a position. Then the 1-order matrix is sampled. Once the first value has been obtained, the procedure is repeated using the 2-order matrix, until k values are set (using k-order matrix). Then the remaining $n - k$ indexes will be sampled directly from the M_k (k-order) matrix.

At each step r, the sampling is carried out as follows. We select in M_r those rows whose positions match partially the currently assigned positions. Then, one of these rows is obtained uniformly at random, and an index is sampled using the probability distribution associated to the chosen row. It is important to bear in mind that the probability distribution in the row is not the original that is in M_r, but the one which is modified to assign probabilities higher than 0 to only the indexes that agree with the current assignment.

Let us look at an example of this procedure (Table 3) for the 4-city TSP problem introduced in the previous section and assuming 2-order marginals EDA. We start by sampling M_1. We choose a position uniformly at random, and then we obtain an index according to the probability distribution in the row corresponding to that position. Let's assume that we sample position 2 and index 3. Therefore, our partially sampled solution is $(-, 3, -, -)$. Now, in order to sample the next position we move to M_2. From all the rows in this matrix that have position 2, i.e, (1,2), (2,3) and (2,4), we choose uniformly at random one of them, for instance (2,3). Now we have to sample the probability distribution in that row. However, we have to take into account that only pairs of indexes that assign index 3 to position 2 are valid. Therefore, the probability distribution is modified to comply with that constraint. We can imagine that we obtain the assignment (3,4) for the positions (2,3). So, our partially sampled individual is now $(-, 3, 4, -)$. This process will continue in M_2 until we have a complete individual.

4. EXPERIMENTS

In order to study the performance of the k-order marginals EDAs, we have chosen some instances of the TSP and FSSP problems (already introduced in Section 2.1) and Quadratic Assignment Problem (QAP) [27]. QAP is described as the allocation of n facilities in n different locations, with the goal of minimizing the sum of the distances multiplied by the corresponding flows.

It is reasonable to expect that by increasing k (order), a larger population should be used to estimate an appropriated probabilistic model over the permutations space. For that reason, different population sizes have been tested for the instances. Particularly, seven different population sizes have been used starting from $10n$ (being n the problem size) and increasing this value by a factor 2 until an upper bound of $640n$ is reached.

From the set of possible parameter values commonly used for EDAs, we have set those described in Table 4. It must be noted that the values have been chosen without performing any previous test.

Table 4: Execution parameters set.

Parameter	Value
k-order	$\{1, 2, 3\}$
Population size range	$\{10n, 20n, 40n, 80n, 160n, 320n, 640n\}$
Selection size	Population size / 2
Offspring size	Population size - 1
Selection type	Ranking selection method
Elitism selection method	The best individual of the previous generation is guaranteed to survive
Stopping criteria	A maximum number of generations: $100n$

Additionally, in order to avoid premature convergence, we apply a technique to control evolution pressure introduced by Baluja in the PBIL algorithm [2]. This method allows to maintain information from the previous generation in order to control the evolution rate of the population. At each generation, the new probabilistic model is built merging, using a given α proportion, the previous probabilistic model and the one learnt from the current population:

$$M_k^t = (\alpha - 1)M_k^t + \alpha M_k^{t-1} \quad (4)$$

In our experiments, we set $\alpha = 0.9$.

Regarding the instances, the following have been chosen: Gröstel 17 cities TSP (*gr17*) [22] , FSSP instance of 20 jobs and 10 machines (*tai20_10*) [26] and asymmetric QAP instance of size 30 (*tai30b*) [26] .

Figure 2 shows the performance of $k = 1$, $k = 2$ and $k = 3$ (only for TSP) order marginals EDA for these instances. Tables 5, 6 and 7 show the optimal fitness found and the mean and standard deviation fitness over 10 repetitions of the experiments with k=1, k=2 and $k = 3$ order marginals for different population sizes.

As expected, the population requirements of the probabilistic model increase considerably as k does. Every plot in Figure 2 confirms that the performance of $k = 1$, $k = 2$ and $k = 3$ models improve when the population size is enlarged. However, once a critical population size is set, the mean results and standard deviations show more competitive results when k is higher. That is, even 1-order EDAs are able to find the similar optimal results as 2-order or 3-order do, the mean fitness values are more stable for higher k models.

This can also be checked in Tables 5, 6 and 7, where the best mean values are shown in bold for each population. It can be seen how more complex models provide the best results once a critical population size is reached. Resulting drawbacks are the computational cost of dealing with large

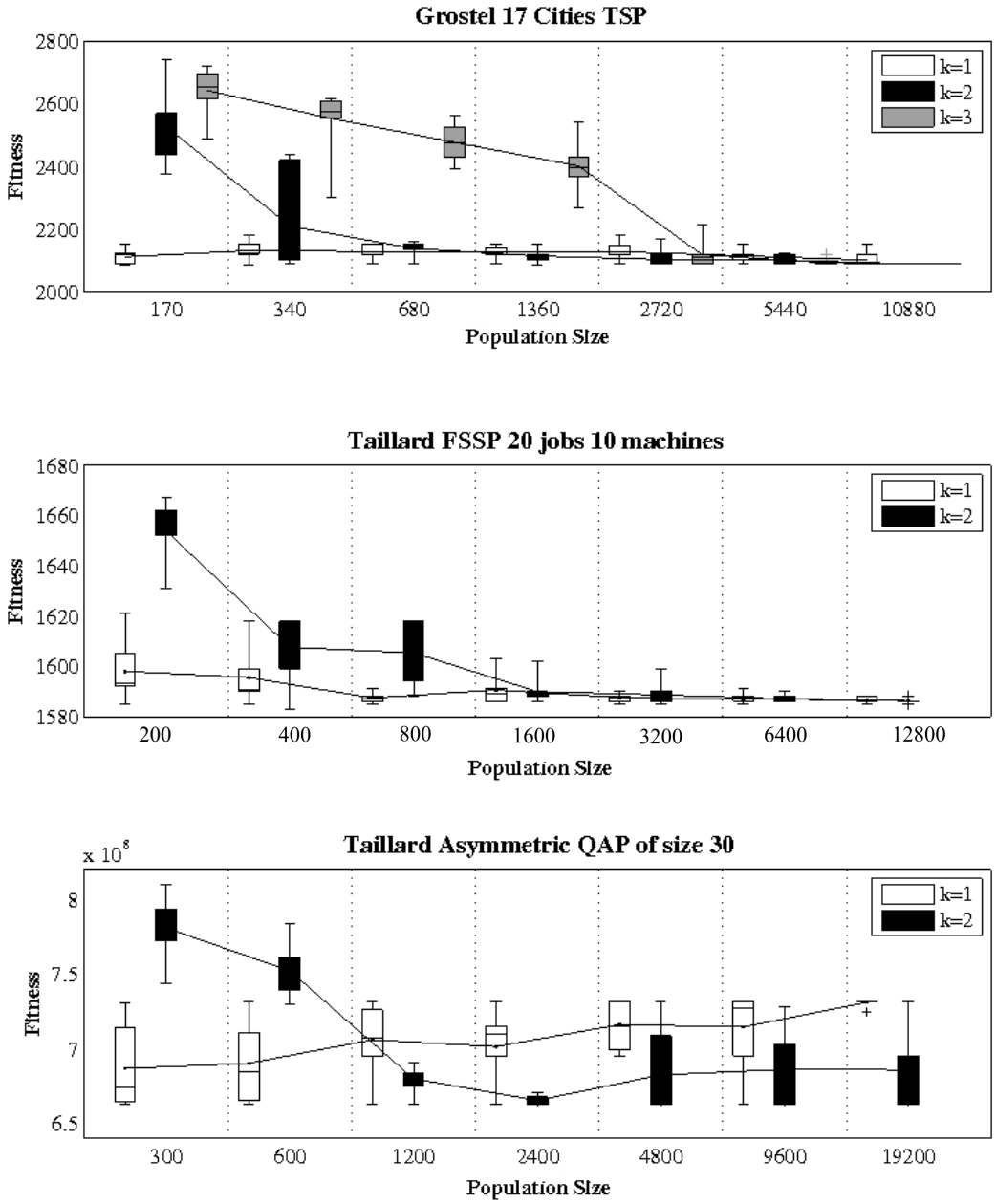

Figure 2: k-order marginals EDA solving *gr17* TSP, *tai20_10* FSSP and *tai30b* QAP instances.

Table 3: Example of creating an individual with the introduced sampling technique

Initialization
The individual is initialized as empty $S = (-, -, -, -)$

Step 1
The uniformly random selected position is 2.
M_1 marginals matrix:

	1	*2*	*3*	*4*
1	0.41	0.16	0.16	0.25
2	0.09	0.50	0.25	0.16
3	0.25	0.16	0.33	0.25
4	0.25	0.16	0.25	0.33

Sampled index is 3.

Step 2
Partially sampled individual is $S = (-, 3, -, -)$
The uniformly random selected combination of positions is $(2, 3)$.
M_2 marginals matrix:

	(1,2)	(1,3)	(1,4)	(2,1)	(2,3)	(2,4)	***(3,1)***	***(3,2)***	***(3,4)***	(4,1)	(4,2)	(4,3)
(1,2)	0.20	0.05	0.10	0.05	0.10	0.05	0.05	0.10	0.05	0.05	0.10	0.10
(1,3)	0.05	0.20	0.10	0.10	0.05	0.05	0.05	0.05	0.10	0.10	0.10	0.05
(1,4)	0.10	0.10	0.15	0.05	0.05	0.10	0.10	0.05	0.05	0.10	0.05	0.10
(2,3)	0.05	0.05	0.05	0.10	0.15	0.15	0.40	0.40	0.20	0.05	0.05	0.10
(2,4)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.05	0.10	0.05	0.10	0.05
(3,4)	0.05	0.10	0.10	0.10	0.05	0.05	0.05	0.10	0.15	0.10	0.05	0.10

Sampled indexes combination is $(3, 2)$.

Step 3
Partially sampled individual is $S = (-, 3, 2, -)$
The uniformly random selected combination of positions is $(3, 4)$.
M_2 marginals matrix:

	(1,2)	(1,3)	(1,4)	***(2,1)***	*(2,3)*	***(2,4)***	(3,1)	(3,2)	(3,4)	(4,1)	(4,2)	(4,3)
(1,2)	0.20	0.05	0.10	0.05	0.10	0.05	0.05	0.10	0.05	0.05	0.10	0.10
(1,3)	0.05	0.20	0.10	0.10	0.05	0.05	0.05	0.05	0.10	0.10	0.10	0.05
(1,4)	0.10	0.10	0.15	0.05	0.05	0.10	0.10	0.05	0.05	0.10	0.05	0.10
(2,3)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.10	0.05	0.05	0.05	0.10
(2,4)	0.05	0.05	0.05	0.10	0.15	0.15	0.10	0.05	0.10	0.05	0.10	0.05
(3,4)	0.05	0.10	0.10	0.66	0.05	0.33	0.05	0.10	0.15	0.10	0.05	0.10

Sampled indexes combination is $(2, 1)$.

Step 4
Partially sampled individual is $S = (-, 3, 2, 1)$
Remaining index is placed in position 1 to end the sampling of the solution. As a result of the sampling process, the new individual is $S = (4, 3, 2, 1)$.

populations. Nevertheless, small-size problems are suitable to be solved by the algorithm. For that reason in the experiments carried out, only the smallest problem size instance was run for 3-order marginals.

A singular behavior has also been seen in these experiments. Initially we expected that higher population sizes would allow the model to represent the search space better, and therefore, the results of the EDA were expected to be better. However, if we analyze the mean values of Table 7 for $k = 1$ and $k = 2$, the same behavior is repeated: initially the mean decreases (as we are minimizing) while we increase the population size, and suddenly the mean begins to increase slightly.

5. DISCUSSION

In this paper we designed an EDA for permutation-based problems. We have based our models on k-order marginals in order to set a parallelism between these models and those used for integer problems. However, there are subtle differences that make this task tricky.

We have used the matrix M_k to sample individuals. Nevertheless, we do not have an explicit expression for the probability distribution we are sampling. In fact, the only information we have is that marginals of the sampled distribution coincide with M_k.

Following the parallelism with EDAs designed for integer problems, we would like to have, given the k-orders marginals, the probability with maximal entropy. However, in the case of permutations, this is difficult to find [1]. It is not clear from the above reference paper when it is possible to sample that probability distribution without an explicit expression for it.

6. CONCLUSIONS AND FUTURE WORK

In this work, we have introduced a new approach of EDAs based on k-order marginals to solve optimization problems represented by permutations. We presented a brief review

Table 5: Results of k-order marginals EDA for *gr17* TSP instance.

Pop. Size	$k=1$			$k=2$			$k=3$		
	Best	*Mean*	*Dev*	*Best*	*Mean*	*Dev*	*Best*	*Mean*	*Dev*
170	2085	**2112.5**	23.6	2377	2526.8	107.0	2489	2642.7	67.5
340	2085	**2132.1**	26.4	2090	2210.9	153.4	2303	2553.5	92.1
680	2090	**2129.0**	23.2	2090	2138.9	26.5	2392	2477.0	56.0
1360	2090	2127.7	20.1	2085	**2115.5**	19.3	2269	2401.7	71.1
2720	2090	2128.6	31.2	2090	**2103.8**	25.7	2090	2118.9	37.9
5440	2090	2115.6	18.2	2090	2102.3	15.9	2090	**2093.0**	9.4
10880	2090	2102.3	21.7	2090	**2090.0**	0.0	2090	**2090.0**	0.0

Table 6: Results of k-order marginals EDA for *tai20_10* FSSP.

Pop. Size	$k=1$			$k=2$		
	Best	*Mean*	*Dev*	*Best*	*Mean*	*Dev*
200	1585	**1598.0**	12.5	1631	1653.8	12.9
400	1585	**1595.5**	11.4	1583	1607.6	14.7
800	1585	**1587.4**	1.9	1588	1605.3	12.0
1600	1586	**1590.6**	5.6	1586	1589.9	4.9
3200	1585	**1587.5**	1.7	1585	1588.6	4.1
6400	1585	**1586.8**	1.7	1586	1587.0	1.4
12800	1585	1586.7	1.1	1585	**1586.2**	1.0

Table 7: Results of k-order marginals EDA for *tai30b* QAP. All the values are $\times 10^5$.

Pop. Size	$k=1$			$k=2$		
	Best	*Mean*	*Dev*	*Best*	*Mean*	*Dev*
300	6624.4	**6866.5**	263.8	7438.4	7808.9	188.1
600	6624.4	**6902.1**	277.9	7299.3	7520.6	155.3
1200	6624.4	7058.9	217.2	6626.7	**6798.0**	84.8
2400	6624.4	7013.9	222.1	6624.4	**6653.1**	314.5
4800	6951.4	7161.3	154.2	6624.4	**6824.4**	294.5
9600	6624.4	7146.3	246.4	6624.4	**6860.8**	273.4
19200	7241.3	7309.9	241.0	6624.4	**6857.3**	273.2

of the most known EDAs and described the most promising EDAs that motivated our approach. We completed some experiments to understand the performance of k-order marginals modeling a probability distribution over the permutation search space.

As we have seen, the models we have considered do not obtain outstanding results. We think that this could be due to two issues. The first one is the population size. In order to learn 2-order or higher order marginals, the population size has to be really big. Secondly, we wonder when it makes sense to keep the relation between the indexes of two positions that are far apart in the individual. In this sense, we are considering the possibility of restricting the k-order marginals to positions that are close to some threshold in the individual. This change would additionally reduce the memory and execution requirements of the algorithms.

In the literature, there exist other schemes that could be used to model probability distributions over permutations. Particularly, the Mallows model and generalized Mallows models are considered as the counterpart of the Gaussian distribution for permutation problems. In the future we plan to evaluate the use of these models in the field of EDAs.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge the generous assistance and support of Ekhine Irurozqui in this work. This work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN 2008-06815-C02-01 and TIN2010-14931 and projects MICINN and COMBIOMED network in computational biomedicine (Carlos III Health Institute). Josu Ceberio holds a grant from the Basque Government.

8. REFERENCES

[1] S. Agrawal, Z. Wang, and Y. Ye. Parimutuel betting on permutations. In *Parimutuel Betting on Permutations*. 4th International Workshop On Internet And Network Economics (WINE), Dec. 2008.

[2] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie Mellon Report, CMU-CS-94-163, 1994.

[3] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *INFORMS Journal on Computing*, 6(2):154–160, 1994.

[4] 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.

[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] P. A. N. Bosman and D. Thierens. Permutation optimization by iterated estimation of random keys marginal product factorizations. In *PPSN*, pages 331–340, 2002.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison/Wesley, Reading MA, 1989.

[8] D. E. Goldberg and R. L. Jr. Alleleslociand the traveling salesman problem. In J. J. Grefenstette, editor, *ICGA*, pages 154–159. Lawrence Erlbaum Associates, 1985.

[9] J. Gupta and J. E. Stafford. Flow shop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711, March 2006.

[10] M. Henrion. Propagating uncertainty in bayesian networks by probabilistic logic sampling. In *UAI*, pages 149–164, 1986.

[11] I. Inza, P. Larrañaga, and B. Sierra. Feature Subset Selection by Estimation of Distribution Algorithms. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 269–294. Kluwer Academic Publishers, 2002.

[12] 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.

[13] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[14] 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.

[15] 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.

[16] J. A. Lozano and A. Mendiburu. Solving job schedulling with Estimation of Distribution Algorithms. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 231–242. Kluwer Academic Publishers, 2002.

[17] 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.

[18] M. Pelikan and D. E. Goldberg. *Genetic algorithms, clustering, and the breaking of symmetry*. Parallel Problem Solving from Nature - PPSN VI. Lecture Notes in Computer Science 1917, 2000.

[19] 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.

[20] 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.

[21] M. Pelikan, S. Tsutsui, and R. Kalapala. Dependency trees, permutations, and quadratic assignment problem. Technical report, Medal Report No. 2007003, 2007.

[22] G. Reinelt. Tsplib.

[23] 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.

[24] T. Romero and P. Larrañaga. Triangulation of bayesian networks with recursive estimation of distribution algorithms. *Int. J. Approx. Reasoning*, 50(3):472–484, 2009.

[25] 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.

[26] E. Taillard. Eric taillard's dataset. *European Journal of Operational Research*, 64(278):85, 1993.

[27] M. J. B. Tjalling C. Koopmans. Assignment problems and the location of economic activities. Technical Report 4, Cowles Foundation for Research in Economics, Yale University, 1955.

[28] S. Tsutsui. Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. In *PPSN*, pages 224–233, 2002.

[29] 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.

[30] S. Tsutsui, M. Pelikan, and D. E. Goldberg. Node histogram vs. edge histogram: A comparison of pmbgas in permutation domains. Technical report, Medal, 2006.