Sim-EDA: A Multipopulation Estimation of Distribution Algorithm Based on Problem Similarity

Krzysztof Michalak

Department of Information Technologies,
Institute of Business Informatics,
Wroclaw University of Economics, Wroclaw, Poland
krzysztof.michalak@ue.wroc.pl

Abstract. In this paper a new estimation of distribution algorithm Sim-EDA is presented. This algorithm combines a multipopulation approach with distribution modelling. The proposed approach is to tackle several similar instances of the same optimization problem at once. Each subpopulation is assigned to a different instance and a migration mechanism is used for transferring information between the subpopulations. The migration process can be performed using one of the proposed strategies: two based on similarity between problem instances and one which migrates specimens between subpopulations with uniform probability. Similarity of problem instances is expressed numerically and the value of the similarity function is used for determining how likely a specimen is to migrate between two populations. The Sim-EDA algorithm is a general framework which can be used with various EDAs.
The presented algorithm has been tested on several instances of the Max-Cut and TSP problems using three different migration strategies and without migration. The results obtained in the experiments confirm, that the performance of the algorithm is improved when information is transferred between subpopulations assigned to similar instances of the problem. The migration strategy which transfers specimens between the most similar problem instances consistently produces better results than the algorithm without migration.

Keywords: estimation of distribution algorithms, multipopulation algorithms, combinatorial optimization

1 Introduction

The algorithm presented in this paper combines a multipopulation approach based on the island model [2, 27] with an estimation of distribution mechanism used in the PBIL algorithm [1] (additionally adapted in the case of the TSP). The multipopulation approach can be useful when the problem requires finding more than one good solution. Such a situation may occur in multimodal

problems in which, in many areas of the search space, solutions can exist which are equally good or almost equally good in terms of the objective function. In such a case it might be preferred, from the decision-making point of view, to maintain several good solutions corresponding to various areas of the search space, rather than to select just one that outperforms the others by possibly just a small value. Multimodal problems have been so far solved in the literature using methods such as species conservation [11], algorithms using the island model [2] and methods based on small-world topologies [8]. It is also possible to apply EDAs to multimodal problems, even though the authors of the paper [17] note "the poor performance of most EDAs for globally multimodal problem optimization". They propose to combine models able to encode conditional dependencies (namely Bayesian Networks) with niching to improve the performance of EDAs for this class of problems. However, contrary to this paper, they use a clustering algorithm for grouping solutions instead of multiple populations with migration. In the paper [5] EDAs are combined with the island model with migration, however, problem instance similarity is not used for controlling the migration. Instead, three different topologies are used (star, ring and broadcast) to determine which subpopulations receive the migrating specimens.

Similarly as in the case of multimodal problems, in multiobjective optimization it is usually not enough to obtain a single good solution. Because, typically, the objectives in a multiobjective optimization problem are conflicting it is not possible to find one solution with the best values of all of them. Instead, a common approach is to try to approximate an entire Pareto front of non-dominated solutions and report the possible choices to the decision maker. The paper [23] presents various ways of applying multipopulation algorithms to multiobjective optimization problems. Discussed approaches include cooperating subpopulations and the multi-start approach [9, 12] in which independent instances of the optimization algorithm are started from different points or with different parameters.

Multipopulation algorithms are also often used for dynamic optimization problems. Because the environment changes, it is not desirable for the entire population to converge close to a single optimum. Multipopulation approaches allow some part of the population to improve the solution near the optimum while some part is left to explore other areas of the search space. Self-Adaptive Differential Evolution algorithm (jDE) [3], Shifting Balance GA (SBGA) [4], Forking Genetic Algorithms (FGAs) [24], and the multinational GA (MGA) [26] are among the methods that use the multipopulation approach for dynamic problems. Multipopulation approach has also been used with EDAs in the case of dynamic optimization problems by some authors [25, 29].

In genetic algorithms a population of specimens which represent solutions of a given problem found by the algorithm is processed. These specimens undergo a process during which genetic operators are used to produce new specimens. The Estimation of Distribution Algorithms, on the other hand, work by modelling a probability distribution that describes properties of good solutions of the problem. Instead of applying genetic operators, the probabilistic model is

updated based on the existing population, and then a new population is generated by sampling the updated model. In various EDAs different probabilistic models are used, ranging from one-dimensional distributions as in UMDA [20, 28] to decision trees [31], one- and multidimensional Gaussian models [7, 30], Boltzmann Machines [21] and Bayesian Networks [18].

The algorithm proposed in this paper combines the multipopulation island model with an estimation of distribution algorithm to solve a set of similar problem instances using migration based on similarities between these instances. In this paper the Sim-EDA algorithm is applied to two optimization problems: the Max-Cut problem and the Travelling Salesman Problem. In a recent paper [22] the Max-Cut problem is used in image segmentation. Solving several problem instances at once would in this case mean performing image segmentation on a set of similar, but not identical images. In the case of the TSP solving several similar problem instances may be necessary for example when analyzing the influence of various changes in the network of roads on the length of transport routes. This task could be performed, for example, when simulating a situation in a city when certain streets become blocked or more crowded due to construction works. In a recent paper [19] the idea of solving many similar problems has been applied to a real world problem in genetics. In the aforementioned paper a structural transfer approach is used in which structural information is extracted and used to bias model construction. In this paper information is transferred in the form of solutions which partake in the migration process. The examples described above show that in certain areas there is a need for solving several similar optimization problem instances at once. Improving the optimization results using the Sim-EDA algorithm may be beneficial to optimization tasks in these areas.

The rest of this paper is structured as follows. Section 2 describes the Sim-EDA algorithm. Section 3 contains a description of the test problems used for experimental verification. In section 4 parameters used for experiments are detailed and the results are presented. Section 5 concludes the paper.

2 Algorithm Description

The Sim-EDA algorithm proposed in this paper is based on the island model [27] used by some multipopulation genetic algorithms. Instead of genetic operations, however, it uses a probabilistic model and follows a cycle of operations typically implemented in EDAs in which, in each generation, the population is constructed using the probabilistic model, the specimens are evaluated and then the probabilistic model is updated. In the Sim-EDA algorithm, after the specimens are evaluated, but before the probabilistic model is updated the specimens are migrated between subpopulations. In this algorithm a separate subpopulation is assigned to each problem instance along with a separate probabilistic model. Therefore, the working of the algorithm can be viewed as a number of EDA algorithm instances each solving one of the optimization problem instances which exchange information through migration of specimens. This approach is different than in EDAs that use a mixture of models, such as [10] because from this

mixture of models they generate a common population and in the Sim-EDA algorithm each model generates a different subpopulation. A mechanism of elitism is used in the Sim-EDA algorithm which stores the best specimen b_k for each problem instance separately. This specimen is updated after a new population is generated and evaluated. In turn, the updated best specimen b_k is added to a newly generated population in the next generation.

2.1 Algorithm details

An overview of the Sim-EDA algorithm is presented in Algorithm 1. Steps 1., 2.1. and 2.4. can be implemented in a way specific to the chosen EDA algorithm. EDAs with different probabilistic models, update procedures and sampling methods can be used depending on the solved problems. In this paper the PBIL algorithm [1] is used for the MaxCut problem [15], because it is well suited for problems defined on a $\{0,1\}^n$ decision space. For the TSP a different probabilistic model is used, that models probabilities of different cities to be successors of each of the cities in good tours. This model is used with a PBIL model update procedure and with a procedure that generates new permutations based on the modelled probabilities of predecessor-successor pairs. For different problems, of course, different EDAs could be used.

The Sim-EDA algorithm uses the following procedures.

BinaryTournament - Performs a binary tournament between two specimens and returns the winner. In the case of a single-objective problem this consists of simply comparing the two specimens' fitnesses. In the case of multiobjective problems the tournament can be based on the concept of Pareto domination.

CreatePopulation - generates a required number of specimens from the given probabilistic model.

Evaluate - evaluates specimens in a given population using a problem-specific goal function. Because during the migration phase specimens are evaluated with respect to a different problem than the one solved by their original population this method has an argument which defines the number of the problem instance to use for evaluation.

Improve - an optional step which performs an improvement or local search around specimens in a given population. Similarly as the evaluation procedure this procedure has an argument which defines the number of the problem instance to use for calculating the goal function.

InitializeModel - initializes a probabilistic model in a way specific to the chosen problem and the EDA algorithm.

SelectBestSpecimens - selects a given number of specimens which have the highest fitness function values from a given set

Algorithm 1 An overview of the Sim-EDA algorithm.

IN: N_{gen} - the number of generations
N_{pop} - the size of each subpopulation
N_{sub} - the number of subpopulations
N_{imig} - the number of migrated specimens

Calculate the matrix $S_{[N_{sub} \times N_{sub}]}$

// 1. Initialize probabilistic models and best specimens
for $d = 1, \ldots, N_{sub}$ **do**
  M_d = InitializeModel()
  $B_d = \emptyset$
end for

// 2. The main loop
for $g = 1, \ldots, N_{gen}$ **do**

  // 2.1. Generate new specimens
  for $d = 1, \ldots, N_{sub}$ **do**
    P_d = CreatePopulation(N_{pop}, M_d)
    Evaluate(P_d, d)
    Improve(P_d, d)
    $P_d = P_d \cup B_d$
    B_d = SelectBestSpecimens(P_d, 1)
  end for

  // 2.2. Select source populations
  for $d = 1, \ldots, N_{sub}$ **do**
    s = SelectSourcePopulation(S, d)
    P'_d = SelectBestSpecimens(P_s, N_{imig})
  end for

  // 2.3. Migrate specimens
  for $d = 1, \ldots, N_{sub}$ **do**
    Evaluate(P'_d, d)
    Improve(P'_d, d)
    for $x \in P'_d$ **do**
      w = the weakest specimen in P_d
      $P_d = P_d - \{w\}$
      b = BinaryTournament(w, x)
      $P_d = P_d \cup \{b\}$
    end for
  end for

  // 2.4. Update probabilistic models
  for $d = 1, \ldots, N_{sub}$ **do**
    M_d = UpdateModel(P_d, M_d)
  end for
end for

SelectSourcePopulation - chooses a source subpopulation from which specimens will be migrated to subpopulation d using the problem similarity matrix S and a chosen migration strategy. Migration strategies are described in section 2.2.

UpdateModel - updates the probabilistic model based on the current population. The implementation of this method depends on the chosen model and the representation of solutions used in specimens.

Details of the InitializeModel, CreatePopulation and UpdateModel procedures specific for test problems used in this paper are discussed in section 3.

2.2 Migration

The Sim-EDA algorithm uses an island model [2, 27] with migration. The subpopulations are assigned to different instances of an optimization problem and the migration process is controlled by a problem similarity matrix $S_{[N_{sub} \times N_{sub}]}$. Entries in the problem similarity matrix $S_{[N_{sub} \times N_{sub}]}$ should reflect how similar the problem instances are. Obviously, the way of calculating $S_{i,j}$ for a given pair of subproblems is specific to the problem representation. The two problems tackled in this paper both use a representation based on a weighted graph. Thus, i-th problem instance can be represented using a cost matrix $C^{(i)}_{[N \times N]}$ (where N is the problem size) that contains the lengths of the edges of the graph. The similarity between the i-th and the j-th problem instance can be calculated as a negated sum of squared differences between elements of the cost matrices:

$$S_{i,j} = -\sum_{p=1}^{N}\sum_{q=1}^{N}(C^{(i)}_{p,q} - C^{(j)}_{p,q})^2 \ . \tag{1}$$

This approach to utilizing problem instance similarity has been successfully used in previous works of the author of this paper for solving the TSP [13] and the firefighter problem [14]. In both previous papers the multipopulation approach with migration was used with evolutionary algorithms, not EDAs.

Migration can be performed using various strategies. In this paper three migration strategies are tested and compared to the Sim-EDA without migration in which each subpopulation works independently on a separate problem instance. The latter is, of course, equivalent to simply executing N_{sub} separate algorithm runs for solving each problem instance in turn.

The migration strategies used in this paper are as follows:

- **nearest** - a strategy in which N_{imig} specimens are migrated to a population P_d from such population P_s, $s \neq d$ that maximizes the value of $S_{d,s}$:

$$s = \operatorname*{argmax}_{t \in \{1,\ldots,N_{sub}\}-\{d\}} (S_{d,t}) \tag{2}$$

– **rank** - in this strategy all subpopulations except P_d are ranked in an increasing order according to values of $S_{d,t}$, $t \in \{1, \ldots, N_{sub}\} - \{d\}$ and the source population is selected randomly using the roulette wheel selection procedure with probabilities proportional to the ranks.
– **uniform** - a strategy in which the source population number s is selected randomly with uniform probability from $\{1, \ldots, N_{sub}\} - \{d\}$.
– **none** - no migration is performed. Used for comparison with other strategies.

The two first migration strategies require calculating the similarity matrix $S_{[N_{sub} \times N_{sub}]}$ as a preprocessing step before the optimization starts.

3 Test Problems

The experiments described in this paper were performed on two test problems: the Max-Cut problem and the Travelling Salesman Problem (TSP). Both problems can be represented using a weighted graph $G = \langle V, E \rangle$, with $w : V \times V \to \mathbb{R}$ representing weights of edges connecting vertices from V. The representations of solutions differ, however, in these problems and also the probabilistic models used in Sim-EDA are different. Further, we will define the problem size as the number of vertices in the graph $N = |V|$.

3.1 The Max-Cut problem

In the Max-Cut problem the goal is to find such a partition $V' \cup V'' = V$, $V' \cap V'' = \emptyset$ of the set of vertices of graph G for which the weight of the edges having one vertex in V' and the other in V'' is maximal. These edges would be the ones to be "cut" if we wanted to separate the graph G into two disjoint subgraphs with vertices from V' and V'' respectively.

Assume, that the weights in the graph are represented as a matrix $C_{[N \times N]}$ in which the element c_{ij} represents the weight of the edge between the i-th and the j. Formally, the Max-Cut problem can be stated as:

$$\begin{aligned} &\text{maximize } f(V') = \sum_{i \in V', j \in (V - V')} c_{ij} \ , \\ &\text{subject to } V' \in 2^V \ , \end{aligned} \tag{3}$$

in which we try to find the best subset V' of the set of vertices V with respect to the goal function (3). The other set V'' is, of course, the complement of V'. Test instances of the size $N = 12, 25, 50, 100, 250$ and 500 were used in the experiments for the Max-Cut problem. In the Max-Cut problem a binary array $\{0, 1\}^N$ can easily be used for representing a solution with 0s corresponding to V' and 1s to V''. The probabilistic model $\mathbb{P}$ is also an N-element array $[0, 1]^N$ in which the elements belong to the range $[0, 1]$ and represent the probabilities of a corresponding entry having a value of 1 in good solutions of the problem.

The procedure InitializeModel() sets $\mathbb{P} = [0.5, 0.5, \ldots, 0.5]$. A new population is created in the CreatePopulation() procedure by generating a given number of binary vectors of length N with i-th coordinate set independently of the others to 1 with probability p_i and to 0 with probability $1 - p_i$. The UpdateModel() procedure performs a standard probabilistic model update used in the PBIL algorithm, which updates the model using the best genotype $g^{(+)}$ and the worst one $g^{(-)}$ selected using their fitness from the current population. The model update mechanism from PBIL is applied for each element p_i, $i = 1, \ldots, N$ in $\mathbb{P}$ separately. The parameters used in this process are two learning rate parameters: the positive learning rate η_+ and the negative learning rate η_-. Their sum is denoted $\eta = \eta_+ + \eta_-$. If $g_i^{(-)} = g_i^{(+)}$ the element p_i of $\mathbb{P}$ is set to:

$$p_i = p_i \cdot (1 - \eta_+) + g_i^{(+)} \cdot \eta_+ \ , \tag{4}$$

and if $g_{ij}^{(-)} \neq g_{ij}^{(+)}$ the element p_i is set to:

$$p_i = p_i \cdot (1 - \eta) + g_i^{(+)} \cdot \eta \ . \tag{5}$$

After probability update, each element g_i (except the elements on the diagonal) is mutated with probability P_{mut} by setting:

$$p_i = p_i \cdot (1 - \mu) + \alpha * \mu \ , \tag{6}$$

where:

α - a 0 or 1 value drawn randomly with equal probabilities $P(0) = P(1) = \frac{1}{2}$,

μ - a mutation-shift parameter controlling the intensity of mutation.

3.2 The Travelling Salesman Problem (TSP)

A real-life motivation for the TSP arises from various transportation problems. The TSP requires visiting N cities in such a way that each of them is visited exactly once and the tour taken is possibly the shortest. Given a cost matrix $C_{[N \times N]}$ the TSP can be formulated as:

$$\begin{aligned} &\text{minimize } f(\pi) = C_{\pi(N)\pi(1)} + \sum_{i=1}^{N-1} C_{\pi(i)\pi(i+1)} \\ &\text{subject to } \pi \in \mathit{\Pi}(N) \end{aligned} \tag{7}$$

where $\mathit{\Pi}(N)$ is the set of all permutations of numbers $1, \ldots, N$. A solution can be represented as a permutation and consequently the genotype of a specimen is an N-element integer array containing the permutation. A different representation of solutions than in the Max-Cut problem requires a different probabilistic model. In the case of the TSP the probabilistic model used in Sim-EDA is a matrix $\mathbb{Q}_{[N \times N]}$ in which an element $\mathbb{Q}[i, j] \in [0, 1]$ represents the probability that the number j is placed right after i in a permutation that constitutes a good

solution of the given TSP instance. This model is initialized by setting uniform probabilities for all pairs $i, j \in \{1, \ldots, N\}$ such that $i \neq j$:

$$\mathbb{Q}_{i,j} = \begin{cases} 0 & \text{for } i = j \\ \frac{1}{N-1} & \text{for } i \neq j \end{cases} \tag{8}$$

When a new specimen is created its genotype X is initialized as presented in Algorithm 2. This algorithm fills the genotype iteratively starting from a random value for $X[1]$. For each $k = 2, \ldots, N$ denote a set of numbers not yet assigned to X by $U = \{1, \ldots, N\} - \{X[1], \ldots, X[k-1]\}$. Two situations are possible. If there are some nonzero elements in $\mathbb{Q}_{i,j}$ for $i = X[k-1]$ and $j \in U$ then select one of the numbers from U as $X[k]$ with probabilities proportional to the respective elements in $\mathbb{Q}$. If all elements in $\mathbb{Q}_{i,j}$ are zero for $i = X[k-1]$ and $j \in U$ then select one of the numbers from U with uniform probability. This ensures that whenever the probabilistic model $\mathbb{Q}$ indicates that there are possible successors to $X[k-1]$ one of them is selected using elements from $\mathbb{Q}$. If there is no successor to $X[k-1]$ with non-zero probability according to the probabilistic model $\mathbb{Q}$ then one of the yet unused numbers is selected with uniform probability.

Algorithm 2 Initialization of a genotype for a new specimen for the TSP based on the probabilistic model $\mathbb{Q}$.

IN: $\mathbb{Q}$ - the probabilistic model
N - genotype length

OUT: X - a new genotype

$X[1] = \text{UniformSelection}(\{1, \ldots, N\})$
for $k = 2, \ldots, N$ **do**
  $U = \{1, \ldots, N\} - \{X[1], \ldots, X[k-1]\}$
  $i = X[k-1]$
  $S = \sum_{j \in U} \mathbb{Q}[i, j]$
  if $S > 0$ **then**
    $X[k] = \text{RouletteWheelSelection}(U, \{\mathbb{Q}[i, j]\}_{j \in U})$
  else
    $X[k] = \text{UniformSelection}(U)$
  end if
end for

The RouletteWheelSelection(A, B) procedure selects elements from A with probabilities proportional to values in B. The UniformSelection(A) procedure selects elements from A with uniform probability.

The update of the probabilistic model $\mathbb{Q}$ is performed similarly as in the PBIL algorithm, but because the probabilistic model $\mathbb{Q}$ is a matrix and the solution representation in the TSP is a permutation an additional preprocessing step is required. Instead of updating the model directly from specimens S_{best} and S_{worst} the genotypes X_{best} and X_{worst} of these specimens are converted to

matrices Q_{best} and Q_{worst} in which an element $Q_{i,j}$ is set to 1 if i and j are consecutive elements in X (including $X[N] = i$ and $X[1] = j$) and otherwise to 0. Clearly, such matrix represents a probability distribution in which consecutive pairs found in a given genotype are given probability 1 and all others probability 0. The matrices Q_{best} and Q_{worst} are used instead of binary vectors in the standard probability model update procedure used in the PBIL algorithm.

Since the mutation in PBIL works independently on each element of the probabilistic model $\mathbb{P}$ the application of the same procedure to $\mathbb{Q}$ is straightforward.

Test instances of the size $N = 12, 25, 50, 100, 150, 200$ and 250 were used in the experiments for the TSP.

3.3 Generating similar problem instances

For the experiments N_{sub} similar instances of each problem for each problem size N were required. Each problem instance is represented as a cost matrix $C^{(d)}_{[N\times N]}$, for $d = 1, \ldots, N_{sub}$. The first cost matrix $C^{(1)}_{[N\times N]}$ was generated by drawing the elements above the diagonal from the uniform probability distribution $U[0, 100]$. To ensure the symmetry of the matrix the elements below the diagonal were initialized by copying the corresponding elements from above the diagonal. Each of the remaining cost matrices $C^{(d)}_{[N\times N]}$, for $d = 2, \ldots, N_{sub}$ was initialized by changing $1/N_{sub}$ (i.e. $1/20 = 5\%$) of the elements not placed on the diagonal in $C^{(d-1)}_{[N\times N]}$. The replaced elements were drawn from the uniform probability distribution $U[0, 100]$ and the symmetry of the new matrix was ensured by setting both corresponding elements above and below the diagonal to the new value. In all the cost matrices the elements on the diagonal were set to 0. The above procedure produces N_{sub} cost matrices $C^{(d)}_{[N\times N]}$, for $d = 1, \ldots, N_{sub}$ which differ less for closer values of the index d and differ more for values of the index d further apart.

4 Experiments and Results

The experiments described in this paper were aimed at verifying the effectiveness of the Sim-EDA algorithm in solving selected combinatorial problems. In particular, the goal was to investigate the influence of the migration procedure on the results of the optimization process. The Sim-EDA algorithm was run on several instances of two test problems: the Max-Cut problem and the Travelling Salesman Problem (TSP) using four migration strategies described in section 2.2. For each test problem and each migration strategy $N_{run} = 30$ runs of the algorithm were performed to allow statistical comparison of the results.

The parameters of the algorithm were set as follows. The number of problem instances (and thus the number of subpopulations) was set to $N_{sub} = 20$. Each subpopulation size was set to $N_{pop} = 100$ for the Max-Cut problem and

$N_{pop} = 1000$ for the TSP. The number of migrated specimens was set to 10% of the population size, thus $N_{imig} = 10$ for the Max-Cut problem and $N_{imig} = 100$ for the TSP.

In this paper the PBIL algorithm with negative learning rate was used as the EDA in the Sim-EDA framework. Parameters of the algorithm were set following the original paper on the PBIL algorithm [1]. This algorithm uses two learning rate parameters which in this paper were set to: learnRate = 0.1 and negLearnRate = 0.075. Two other parameters which control probabilities used in the mutation procedure were set to $P_{mut} = 0.02$ and $P_{shift} = 0.05$.

The migration process used in the Sim-EDA algorithm requires some extra computation time for both the similarity matrix calculation as the preprocessing step and for additional computations during the migration phase in each generation. Therefore, results obtained with various migration strategies should not be compared with respect to the same number of generations, but with respect to time. In Figures 1 and 2 examples of the performance of tested migration strategies with respect to time are presented.

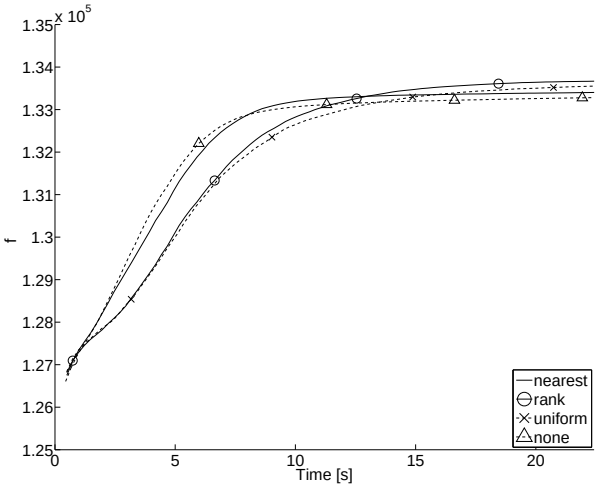

Fig. 1. Median results obtained for the Max-Cut problem with $N = 100$ nodes.

For comparison of the migration strategies the following procedure has been adopted in this paper. Each strategy was allowed to run for $N_{gen} = 300$ generations. The average time t_0 in which the "none" strategy completed $N_{gen} = 200$ generations in $N_{run} = 30$ runs of the test was used as a reference point at which the results obtained by all the strategies were compared. Results for all tested strategies were recorded at time t_0, including any preprocessing time required by a given strategy and the time for migration. For each of the N_{sub} subpopulations in each of the $N_{run} = 30$ runs the best attained objective function value f_{best}

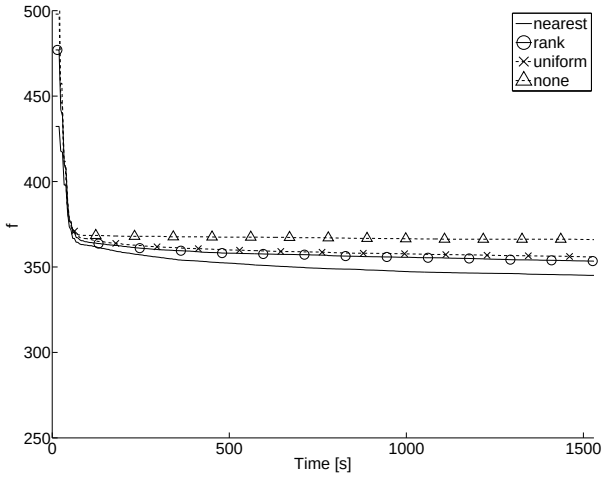

Fig. 2. Median results obtained for the TSP with $N = 100$ cities.

was recorded. Tables 1-2 present the mean and median values calulated from the recorded f_{best} values.

For statistical verification of the results the Wilcoxon rank test was used. The Wilcoxon test was chosen because it does not assume the normality of the distributions which may be hard to guarantee in practical applications. This test was recommended in a survey [6] on methods suitable for evaluating evolutionary and swarm intelligence algorithms. In Tables 1-2 the column "p-value" contains the p-value obtained by the Wilcoxon test for a null hypothesis that the migration strategy corresponding to a given row produces results with the same median as the "none" strategy. A low p-value indicates that it is statistically unlikely that the medians are equal. The column "interp." contains the interpretation of the p-value which is determined in the following way. If the median obtained by a given strategy is lower for the Max-Cut problem or higher for the TSP than that produced by the "none" strategy the interpretation is "worse". In the opposite case the interpretation depends on the result of the statistical test. If the obtained p-value is > 0.05 the interpretation is "insignificant" and if the obtained p-value is ≤ 0.05 the interpretation is "significant".

In the case of the Max-Cut problem the strategies using migration outperformed the one not using migration for all small instances ($N \leq 100$). With one exception (the "nearest" strategy for $N = 12$) this advantage is statistically significant. For large instances ($N = 250$ and 500) only the "nearest" strategy outperformed the strategy without migration, but this advantage was found to be statistically significant. Overall, the "nearest" strategy is the one that consistently outperformed the "none" strategy on all tested instances of the Max-Cut problem. With exception to the smallest problem instance ($N = 12$) the advantage of the "nearest" migration strategy is statistically significant.

Table 1. Results of the tests on the Max-Cut problem (higher values are better).

Instance size	Migration strategy	Mean	Median	std. dev.	Comparison to "none"	
					p-value	interp.
12	none	2052.2833	2043.2797	68.7213		
	nearest	2052.7526	2043.2797	67.1400	0.43307	insignificant
	rank	2054.2401	2043.2797	66.5184	0.0003651	significant
	uniform	2053.6531	2043.2797	67.5602	0.0066871	significant
25	none	8671.3856	8664.3626	136.6357		
	nearest	8686.8558	8674.5851	132.2364	1.68E-05	significant
	rank	8701.5194	8686.9078	131.8232	2.00E-20	significant
	uniform	8702.0109	8689.4084	133.8095	9.00E-21	significant
50	none	34169.0427	34157.2758	262.0070		
	nearest	34213.7399	34183.5902	259.1844	3.61E-06	significant
	rank	34262.5991	34233.2313	302.8604	4.88E-21	significant
	uniform	34252.5286	34246.1427	292.6219	1.73E-19	significant
100	none	133281.7688	133292.2493	505.6580		
	nearest	133405.0093	133433.3314	509.5968	7.09E-08	significant
	rank	133486.5788	133651.4388	817.3963	4.58E-19	significant
	uniform	133195.4960	133514.7050	1114.0702	1.27E-03	significant
250	none	817251.5645	817303.3203	1411.8478		
	nearest	818132.9153	818057.8073	1488.0998	7.94E-26	significant
	rank	813343.0263	815448.3402	6016.6470	3.58E-37	worse
	uniform	809292.8215	809954.2165	7203.6887	3.48E-79	worse
500	none	3214635.065	3214426.042	3388.4932		
	nearest	3214979.768	3215141.950	3875.1273	4.89E-02	significant
	rank	3187810.589	3190216.098	18081.6794	7.56E-98	worse
	uniform	3175792.355	3168177.289	20508.3657	8.48E-98	worse

For small instances of the TSP ($N = 12$ and 25) all the strategies with migration outperformed the "none" strategy, however, no statistical significance could be confirmed (cf. Table 2). For large instances ($N \geq 100$) the "nearest" strategy consistently produced significantly better results than those obtained when no migration was used. Overall, all the strategies with migration outperformed the "none" strategy in all cases except the "rank" strategy for $N = 50$. Not in all cases, however, the difference in results was large enough to allow stating statistical significance.

In summary, strategies with migration outperformed the "none" strategy in most cases. Especially, the "nearest" migration strategy consistently produced better results than those obtained without migration.

5 Conclusion

This paper presents an algorithm combining the Estimation of Distribution Algorithms with an island model. The proposed algorithm was designed with an intent of solving several instances of an optimization problem at once. In the case of the Max-Cut problem which can be applied to image segmentation [22] the proposed approach would allow processing several similar images at once. For other problems the proposed approach may be useful for simulating the ef-

Table 2. Results of the tests on the TSP (lower values are better).

Instance size	Migration strategy	Mean	Median	std. dev.	Comparison to "none"	
					p-value	interp.
12	none	217.3898	216.1576	13.3413		
	nearest	217.3898	216.1576	13.3413	1	insignificant
	rank	217.3898	216.1576	13.3413	1	insignificant
	uniform	217.3898	216.1576	13.3413	0.22656	insignificant
25	none	164.4112	156.5974	27.472		
	nearest	164.3935	156.5974	27.4952	0.38316	insignificant
	rank	164.3905	156.5974	27.4855	0.91015	insignificant
	uniform	164.3892	156.5974	27.486	0.40226	insignificant
50	none	223.0045	220.3383	27.4457		
	nearest	222.3719	219.5455	28.2546	0.088667	insignificant
	rank	224.3653	220.6636	27.8668	0.00072179	worse
	uniform	223.5734	219.0627	27.6526	0.30527	insignificant
100	none	365.9890	364.7559	28.1933		
	nearest	359.3402	360.5102	24.6559	4.2904E-18	significant
	rank	365.0730	364.5377	26.6054	0.012205	significant
	uniform	365.4464	364.6945	26.9252	0.21189	insignificant
150	none	404.8644	403.0477	26.8652		
	nearest	403.4235	401.3445	25.9356	1.6479e-008	significant
	rank	404.7463	402.9095	26.7110	0.03125	significant
	uniform	404.7797	402.9095	26.7887	0.125	insignificant
200	none	456.2797	456.6390	23.8788		
	nearest	455.8308	456.2452	23.5170	6.1035e-005	significant
	rank	456.2793	456.6390	23.8788	1	insignificant
	uniform	456.1204	456.5193	23.7636	0.0625	insignificant
250	none	510.5612	510.1462	24.3592		
	nearest	510.3769	510.0111	24.1334	0.0078125	significant
	rank	510.5425	510.1462	24.3338	1	insignificant
	uniform	510.5477	510.1462	24.3529	0.5	insignificant

fects of various changes in the input data. For example in the case of the TSP it is possible to determine how the changes in the graph of available routes will influence the length of the tour.

The proposed approach was tested on the Max-Cut problem instances of the size $N = 12, 25, 50, 100, 250$ and 500 and on the TSP instances of the size $N = 12, 25, 50, 100, 150, 200$ and 250. Strategies using migration outperformed the strategy without migration in most cases. In particular, the strategy of migrating specimens solely from the nearest subpopulation consistently produced better results than those obtained when no migration was used.

In this paper the algorithm was tested on synthetic data and the focus was on determining if the migration improves the performance of the algorithm and how effective various migration strategies are. Application to real-life problems has been left for a further work. Another interesting topic for further work is to use migration of probabilistic models instead of specimens. This approach has been shown to be effective in the paper [5], so it seems promising to develop methods combining probabilistic models migration with problem instance similarity measures. Another interesting idea comes from the field of study on the parameterized complexity [16], where machine learning methods are used to predict

the difficulty of a given problem instance based on the features of this instance. A similar approach could be used in Sim-EDA to decide, based on features of the instances of the optimization problem, between which subpopulations to migrate solutions.

References

1. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. rep., Pittsburgh, PA, USA (1994)
2. Bessaou, M., Petrowski, A., Siarry, P.: Island model cooperating with speciation for multimodal optimization. In: Schoenauer, M., et al. (eds.) Parallel Problem Solving from Nature PPSN VI. LNCS, vol. 1917, pp. 437–446. Springer Berlin Heidelberg (2000)
3. Brest, J., et al.: Dynamic optimization using self-adaptive differential evolution. In: IEEE Congress on Evolutionary Computation. pp. 415–422. IEEE (2009)
4. Chen, J., Wineberg, M.: Enhancement of the shifting balance genetic algorithm for highly multimodal problems. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation. pp. 744–751. IEEE Press, Portland, Oregon (2004)
5. delaOssa, L., Gamez, J.A., Puerta, J.M.: Migration of probability models instead of individuals: An alternative when applying the island model to EDAs. In: Yao, X., et al. (eds.) Parallel Problem Solving from Nature - PPSN VIII, LNCS, vol. 3242, pp. 242–252. Springer Berlin Heidelberg (2004)
6. Derrac, J., et al.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Comput. 1(1), 3–18 (2011)
7. Dong, W., Yao, X.: Unified eigen analysis on multivariate gaussian based estimation of distribution algorithms. Information Sciences 178(15), 3000–3023 (2008)
8. Giacobini, M., Preuss, M., Tomassini, M.: Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In: Proceedings of the 6th European Conference on Evolutionary Computation in Combinatorial Optimization. pp. 86–98. EvoCOP'06, Springer-Verlag, Berlin, Heidelberg (2006)
9. Jozefowiez, N., Semet, F., Talbi, E.G.: Target aiming pareto search and its application to the vehicle routing problem with route balancing. Journal of Heuristics 13(5), 455–469 (2007)
10. Lancucki, A., et al.: Continuous population-based incremental learning with mixture probability modeling for dynamic optimization problems. In: Corchado, E., et al. (eds.) Intelligent Data Engineering and Automated Learning - IDEAL 2014, LNCS, vol. 8669, pp. 457–464. Springer International Publishing (2014)
11. Li, J.P., et al.: A species conserving genetic algorithm for multimodal function optimization. Evol. Comput. 10(3), 207–234 (2002)
12. Mezmaz, M.S., Melab, N., Talbi, E.: Using the multi-start and island models for parallel multi-objective optimization on the computational grid. In: e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on. pp. 112–120 (2006)
13. Michalak, K.: Sim-EA: An evolutionary algorithm based on problem similarity. In: Corchado, E., et al. (eds.) Intelligent Data Engineering and Automated Learning IDEAL 2014, LNCS, vol. 8669, pp. 191–198. Springer (2014)

14. Michalak, K.: The Sim-EA algorithm with operator autoadaptation for the multi-objective firefighter problem. In: Ochoa, G., et al. (eds.) Evolutionary Computation in Combinatorial Optimization, LNCS, vol. 9026, pp. 184–196. Springer (2015)
15. Newman, A.: Max cut. In: Kao, M.Y. (ed.) Encyclopedia of Algorithms, pp. 1–99. Springer US (2008)
16. Nudelman, E., et al.: Understanding random SAT: Beyond the clauses-to-variables ratio. In: Wallace, M. (ed.) Principles and Practice of Constraint Programming CP 2004, LNCS, vol. 3258, pp. 438–452. Springer Berlin Heidelberg (2004)
17. Peña, J.M., Lozano, J.A., Larrañaga, P.: Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of bayesian networks. Evol. Comput. 13(1), 43–66 (2005)
18. Pelikan, M., Goldberg, D.E.: Hierarchical problem solving by the bayesian optimization algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference 2000. pp. 267–274. Morgan Kaufmann (2000)
19. Santana, R., Mendiburu, A., Lozano, J.: Structural transfer using EDAs: An application to multi-marker tagging SNP selection. In: Evolutionary Computation (CEC), 2012 IEEE Congress on. pp. 1–8 (2012)
20. Santana, R., Larranaga, P., Lozano, J.A.: Side chain placement using estimation of distribution algorithms. Artificial Intelligence in Medicine 39(1), 49–63 (2007)
21. Shim, V.A., et al.: Enhancing the scalability of multi-objective optimization via restricted boltzmann machine-based estimation of distribution algorithm. Information Sciences 248, 191–213 (2013)
22. de Sousa, S., Haxhimusa, Y., Kropatsch, W.: Estimation of distribution algorithm for the max-cut problem. In: Kropatsch, W.G., et al. (eds.) Graph-Based Representations in Pattern Recognition, LNCS, vol. 7877, pp. 244–253. Springer Berlin Heidelberg (2013)
23. Talbi, E.G., et al.: Parallel approaches for multiobjective optimization. In: Branke, J., et al. (eds.) Multiobjective Optimization, LNCS, vol. 5252, pp. 349–372. Springer Berlin Heidelberg (2008)
24. Tsutsui, S., Fujimoto, Y., Ghosh, A.: Forking genetic algorithms: Gas with search space division schemes. Evolutionary Computation 5(1), 61–80 (1997)
25. Uludag, G., et al.: A hybrid multi-population framework for dynamic environments combining online and offline learning. Soft Computing 17(12), 2327–2348 (2013)
26. Ursem, R.K.: Multinational GA Optimization Techniques in Dynamic Environments. In: Whitley, D., et al. (eds.) Genetic and Evolutionary Computation Conference. pp. 19–26. Morgan Kaufmann (2000)
27. Whitley, D., Rana, S., Heckendorn, R.B.: The island model genetic algorithm: On separability, population size and convergence. Journal of Computing and Information Technology 7, 33–47 (1998)
28. Yan, W., Xiaoxiong, L.: An improved univariate marginal distribution algorithm for dynamic optimization problem. AASRI Procedia 1, 166–170 (2012), AASRI Conference on Computational Intelligence and Bioinformatics
29. Yang, S., Yao, X.: Dual population-based incremental learning for problem optimization in dynamic environments. In: Proc. of the 7th Asia Pacific Symposium on Intelligent and Evolutionary Systems, pp. 49–56 (2003)
30. Yuan, B., Orlowska, M., Sadiq, S.: Extending a class of continuous estimation of distribution algorithms to dynamic problems. Optim. Lett. 2(3), 433–443 (2008)
31. Zhong, X., Li, W.: A decision-tree-based multi-objective estimation of distribution algorithm. In: Computational Intelligence and Security, 2007 International Conference on. pp. 114–118 (2007)