Investigating selection strategies in multi-objective probabilistic model based algorithms

Andrei Strickler, Olacir Castro Jr., Aurora Pozo
Computer Science's Department
Federal University of Paraná
Curitiba, Paraná, Brazil
Email: {astrickler, orcjunior, aurora}@inf.ufpr.br

Roberto Santana
Department of Computer Science and Artificial Intelligence
University of the Basque Country
San Sebastian-Donostia 20080, Spain
Email: roberto.santana@ehu.es

***Abstract*—Recent advances on multi-objective evolutionary algorithm (MOEAs) have acknowledged the important role played by selection, replacement, and archiving strategies in the behavior of these algorithms. However, the influence of these methods has been scarcely investigated for the particular class of MOEAs that use probabilistic modeling of the solutions. In this paper we fill this void by proposing an analysis of the role of the aforementioned strategies on an extensive set of bi-objective functions. We focus on the class of algorithms that use Gaussian univariate marginal models, and study how typical selection and replacement strategies used together with this probabilistic model impact the behavior of the search. Our analysis is particularized for a set of bi-objective functions that exhibit a representative set of characteristics (e.g. decomposable, ill-conditioned, non-linear, etc.). The experimental results shows that MOEAs that use simple probabilistic modeling outperform traditional MOEAs based on crossover operators.**

***Keywords*-multi-objective optimization; estimation of distribution algorithms; Gaussian UMDA; selection;**

I. Introduction

One of the questions that make the design and application of evolutionary algorithms (EAs) and other optimization methods a challenging task is the wide variety of potential fitness landscapes where an algorithm could be applied. Since no algorithm is expected to perform well for all classes of functions, the creation of methods able to adapt to the characteristics of the search space seems a reasonable strategy for devising efficient search algorithms.

Among the methods that adapt the search of EAs to the characteristics of the fitness landscape, we can highlight the estimation of distribution algorithms (EDAs) [1]. These methods replace the application of genetic operators by the use of probabilistic models. The rationale is that the probabilistic model could capture the most salient features of the best solutions found during the search and generate new solutions that keep these characteristic patterns. Probabilistic modeling can explicitly detect and exploit some relevant features of the fitness landscape, e.g. the decomposition of the fitness function in groups of interacting variables.

Recently, probabilistic modeling has been also applied in the multi-objective domain. Several types of multi-objective estimation of distribution algorithms (MOEDAs) have been proposed [2], [3], [4]. A common approach to the design of MOEDAs is to replace the original application of the genetic operators (crossover and mutation, or only crossover) by a new step where a probabilistic model is learned from the solutions contained in the archive. Then, the model is used to generate (sample) new solutions. This approach often produces optimizers that improve the efficiency of MOEAs with simple genetic operators. However, the importance of the interaction between the methods used to select, replace, and archive the solutions with the probabilistic model is usually overlooked.

Research on single-objective optimization has shown that there is a close relationship between the type of selection methods applied and the quality of the probabilistic models learned and consequently the impact of model learning on the search. Furthermore, this relationship between the selection of solutions and model learning can depend on the characteristics of the functions being optimized. Some combination of selection and model learning methods can be more effective for a particular class of functions. In MOEAs, there are other factors that add together with probabilistic modeling in the efficiency of the search. These factors include replacement methods and archiving strategies. In this paper we present some initial steps in the investigation of the relationship between probabilistic learning and all other factors previously mentioned. We focus on the simplest model usually applied in EDAs, the univariate marginal model [5] and use a recently proposed benchmark comprising 55 bi-objective functions that represent different sources of difficulty for EAs [6]. In our study, we apply state-of-the-art benchmarking methods that allow us to extract sound, statistically supported, conclusions about the influence of the selection methods.

The paper is organized as follows: In the next section we present the general framework of our study, where recombination operators and all other MOEAs components are divided into two main groups. Section II presents the univariate marginal probabilistic model, describing the methods used to learn and sample it. Section III describes the classical MOEAs that are used as baselines for our investigation. Section IV introduces the Comparing Continuous Optimizers (COCO) [6] benchmark, and describes its main characteristics. Section V presents the experimental framework and discusses the results of the experiments. In Section VI, we present the conclusions of the paper.

II. Probabilistic modeling using the Gaussian univariate model

Probabilistic modeling has been increasingly applied to MOEAs. In this approach, problem regularities captured by the probabilistic model are used to generate new solutions, thus trying to successfully deal with the limitations of traditional EAs. The probabilistic modeling can unveil useful modularities of the problem and orientate the search to promising areas of the search space.

One of the simplest approaches for probabilistic modeling is to assume that the problem variables are independent among them. Under this assumption, the probability distribution of any individual variable should not depend on the values of any other variables. EDAs of this type are usually called univariate EDAs and in addition to problems where some of the variables are independent, they can also solve problems with weak interactions between the variables.

The continuous univariate marginal distribution algorithm (UMDA$_c$) proposed by Larrañaga et al. [5] belongs to the class of univariate EDAs. It is an adaptation of the discrete UMDA [7] for continuous domain, where one Gaussian univariate model is learned for each variable. The univariate Gaussian model factorizes a distribution as:

$$f_l(x, \theta^l) = \prod_{i=1}^{n} f l(x_i, \theta_i^l), \quad \theta_i^l = \{\hat{\mu}_i^l, \hat{\sigma}_i^l\} \tag{1}$$

$$\hat{\mu}_i^l = \overline{x}_i^l = \frac{1}{N} \sum_{r=1}^{N} x_{i,r}^l \tag{2}$$

$$\hat{\sigma}_i^l = \sqrt{\frac{1}{N} \sum_{r=1}^{N} \left(x_{i,r}^l - \overline{x}_i^l\right)^2} \tag{3}$$

where $\hat{\mu}_i^l$ and $\hat{\sigma}_i^l$ are the mean and standard deviation, respectively, computed from the selected population in the l-th generation. $(x_{1,r}^l, x_{2,r}^l, \ldots, x_{n,r}^l)$ are the values of each variable i for any selected solution r [5].

The initial population (l = 0) is generated from a uniform distribution over the feasible search space. In the first generation (l = 1), μ_i^l and σ_i^l are estimated based on selected solutions from this random population. The model is then used to sample new solutions and the same loop is applied in each generation of the algorithm until some stop condition is satisfied. Only three algorithm parameters must be specified for an implementation of UMDA$_c^G$:the population size M, number of points to be selected (N), and the method to select these points (see Section III).

As baseline algorithms for comparison, we used MOEAs that apply the Simulated Binary Crossover (SBX) and Polynomial mutation [8] as crossover and mutation operators, respectively. The SBX simulates the behavior of the single-point crossover on binary strings. Given two parent individuals to be recombined, it generates the i-th component, $i = 1, \ldots, n$, of the children. The parameter η_c determines how well spread the children will be from their parents.

III. Methods for selecting the best solutions

This section briefly describes the MOEAs that will be used in the experimentation phase. The algorithms were selected because they implement selection and replacement strategies that are very different to each other. Some of these algorithms are among the most applied in the evolutionary computation community. The algorithms used were: 1) NSGA-II [9]; SPEA2 [10]; 3) FEMO [11]; and IBEA [12]. In the following, we explain in few words the distinguished features of each algorithm.

1) NSGA-II: The Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II) by Deb et al. [9] applies the rank assigned to each solution by non-dominated sorting as primary selection method. Non-dominated individuals are assigned rank one and the set of individuals with equal rank is called a front. Those individuals that are non-dominated once the first front has been removed are assigned as rank two. The third front is decided within the population discarding the first and the second front and so on. Individuals with equal ranks are evaluated using a secondary selection method called crowding distance (CD). The CD measures the distances to the next higher and lower values in each dimension.

2) SPEA2: The Strength Pareto Evolutionary Algorithm (SPEA2) by Zitzler et al. [10] uses two ranking criteria as well. It is an elitist algorithm with an external archive, which usually is set to be the population size. As primary selection method, a strength value is used. This value is given by the number of individuals in the population dominated by the current individual.

Based on the strength values a raw fitness is computed for each individual, as the sum of the strength values of every individual that dominates it. Thus, every non-dominated individual has the raw fitness equals to zero. In a second stage, a density estimation is performed based on the Euclidean distances between all individuals. The primary fitness value is the raw fitness plus the inverse of the sum of the distance to the k-nearest neighbor (KNN) [13]. The individuals with the best fitness are copied to the external archive for the next generation. In case there is more non-dominated solutions than the external archive size, the distance to the nearest neighbors for increasing k is used as criterion. Another case is that the non-dominated solutions are less than the external archive size, then it will be filled by dominated solutions.

3) FEMO: The Fair Evolutionary Multi-objective Optimizer (FEMO) [11] follows a strategy different to SEMO. Basically, it intends to increase diversity by allowing all selected individuals to produce a similar number of offspring. It uses a so-called fair sampling strategy, ensuring that all individuals would receive, approximately, the same amount of samples. To implement this strategy, a local mutation operator is applied to each solution. Additionally, a counter is kept for each individual in the population to measure the number of descendants it has created. The counter of an individual would be set to 0 whenever a new individual is

produced. After a new solution $x^{'}$ has been generated, it will be added to the archive. If there is not any other solution that dominates $x^{'}$ or has the same function values as $x^{'}$, means that solution $x^{'}$ will be part of the parent population for the next generation, and also means that the parent population are filled with the non-dominated solutions.

4) IBEA: In Zitzler's and Kunzli's [12] IBEA algorithm, the use of binary performance metrics that map an ordered pair of individuals to a scalar value were suggested as indicator functions. A suitable indicator has to be dominance preserving, which means that the indicator must not evaluate a vector better than another that dominates it. Two efficiently computable indicators have been suggested in [12].

- The additive ϵ-indicator comprises the translations in each dimension of objective space that are necessary to create a weakly dominated solution.
- The hypervolume indicator measures the dominated hypervolume that is only dominated by one vector and not by the other.

For both indicators, negative values mean that the first individual of the pair dominates the other. In each individual, its indicator values are charged in a sum of an exponential function to get a fitness value. For dominance preserving an indicator must not evaluate a vector with a fitness value worse than the fitness value of a vector that dominates it.

5) Recombination operator VS other components: A distinguished feature of our approach is considering on one hand the role of the recombination operator, and on the other hand the joint effect all other components of the MOEAs (e.g. selection, replacement, archiving). By doing this methodological separation of the MOEAs components we can focus our analysis on the impact that different strategies to select, replace, and archive the solutions have in the effectiveness of probabilistic modeling. All the MOEAs previously analyzed in this section allow this dissection in two main components.

One of our assumptions is that some procedures to select and archive the solutions may be more appropriate for exploiting the benefits of probabilistic modeling. The other premises is that success of the combination of probabilistic modeling and other MOEAs components will depend on the characteristics of the functions being optimized.

IV. Comparing Continuous Optimizers (COCO)

Fairly comparing optimization algorithms with different characteristics and capabilities is often a hard and tedious task. Usually researchers want to compare the algorithms under different domains of difficulty like non-separability, multi-modality and ill-conditioning, moreover it is important to be able to directly compare algorithms with different populational structures, like steady-state, single-population, multi-population and decomposition variants.

In order to automate the task of conducting scientifically sound experimental studies involving numerical optimizers, the Comparing Continuous Optimizers (COCO) [6] framework was developed. The platform provides benchmark suites, experimental templates and tools for processing and visualizing the outcome of one or more optimizers. The processing of quality indicators is based on runtimes, measured in number of objective function evaluations to reach one or several quality indicator target values. Recently, the platform was rewritten to deal with multi-objective problems and optimizers, for doing so, a new bi-objective suite of benchmark problems and new performance assessment mechanisms were proposed.

The new bi-objective benchmark suite was called "bbob-biobj". This suite was created by combining a subset of the 24 single-objective problems of the original "bbob" test suite. Combining the 24 original functions without permutations would result in 300 problems. However, having so many functions would be impracticable in terms of the overall running time. Hence, two representative functions of each of the five domains of difficulty available were chosen in order not to introduce bias towards any specific domain. These pairwise combinations resulted in 55 bi-objective functions of the final "bbob-biobj" suite. These 55 functions are grouped in 15 classes according to the domains of difficulty of its component subproblems as presented on Table I. Each of these functions is provided in six dimensions (2, 3, 5, 10, 20 and 40) and with a large number of possible instances [14].

One of the most remarkable characteristics of COCO is its new performance assessment mechanism that considers a quality indicator based on the hypervolume of the external archive A_t instead of the objective value of a single-objective function. This external archive contains all non-dominated solutions obtained so far in an algorithm run. Using an archive is a relevant practice in real-world application, moreover using an external archive for performance assessment allows comparing algorithms with different or even changing population sizes or structures. The normalized external archive is evaluated by a quality indicator that can be either the negative hypervolume, with the nadir as reference point if the nadir is dominated by at least one point in the archive. Otherwise the quality of the archive is given by the distance between the closest point in A_t to the region of interest delimited by the nadir point. The target values are based on a target precision ΔI and a reference hypervolume indicator value I^{ref}, which is an approximation of the I_{HV}^{COCO} indicator value of the Pareto front [14]. The results of the performance assessment are presented as empirical distribution functions, where the proportion of problems solved within a specified budged (in x-axis) is displayed.

V. Experiments

In this section we evaluate the different variants of MO-UMDA$_c$ on the COCO framework. First, we present a brief description of the implementation and parameters used by

Table I
CLASSES OF FUNCTIONS INCLUDED IN THE COCO BENCHMARK

Class	Functions	Domain F1	Domain F2
1	f1,f2,f11	separable	separable
2	f3,f4,f12,f13	separable	moderate
3	f5,f6,f14,f15	separable	ill-conditioned
4	f7,f8,f16,f17	separable	multi-modal
5	f9,f10,18,f19	separable	weakly-structured
6	f20,f21,f28	moderate	moderate
7	f22,f23,f29,f30	moderate	ill-conditioned
8	f24,f25,f31,f32	moderate	multi-modal
9	f26,f27,f33,f34	moderate	weakly-structured
10	f35,f36,f41	ill-conditioned	ill-conditioned
11	f37,f38,f42,f43	ill-conditioned	multi-modal
12	f39,f40,f44,f45	ill-conditioned	weakly-structured
13	f46,f47,f50	multi-modal	multi-modal
14	f48,f49,f51,f52	multi-modal	weakly-structured
15	f53,f54,f55	weakly-structured	weakly-structured

the algorithm. Then, a comparison between the MO-UMDA$_c$ variants and the baseline algorithms that incorporate SBX is presented. Finally, we present a detailed investigation of MO-UMDA$_c$ for the different classes of bi-objective functions described in the previous section.

A. Implementation and parameters of the algorithms

All the algorithms were implemented using the PISA framework [15] which is a platform and programming language independent framework for search algorithms. It is mainly focused on multi-objective optimization algorithms. It is organized in two major modules: the variator and the selector. These two modules will communicate with each other through text files containing common parameters. The parameters used by the different MO-UMDA$_c$ variants were.

- NSGAII and SPEA2: Binary-tournament
- IBEA: indicator = Hypervolume; kappa = 0.05; ρ = 1.1

The common parameters used for all standard MOEAs used in this paper are: Mutation rate: 1 %, Crossover rate: 100, and Distribution index (eta_mutation and eta_recombination): 20. It is worth noting that the implementation of MO-UMDA$_c$ does not comprise the application of a mutation step. All the algorithms were allowed to run for 10000 function evaluations. The number of generations was set to 200 and the population size was 50. Results were post-processed using the COCO post-processing capabilities.

B. Comparison of traditional MOEAs vs MO-UMDA$_c$

Figure 1 shows a comparison between traditional MOEAs and MO-UMDA$_c$ variants. The figure represents the number of functions (out of 55) for which the MOEAs achieved the required level of precision in the best solution found. The results shown correspond to the problem with the smallest dimensionality in the space of decision variables (2D).

An analysis of Figure 1 reveals that for all functions all algorithms are able to find the first level of precision. However, as the level of precision is increased, the success rate significantly decreases. From the second level of precision is evident that all the MO-UMDA$_c$ variants outperform the results of traditional MOEAs with SBX. Therefore, to investigate the influence that the different classes of difficulty produce we will concentrate our analysis on the MOEAs that use probabilistic modeling. Figure 1 also shows that probabilistic modeling has a different effect according to the strategy used to select and replace the solutions. NSGAIIu is clearly the best algorithm and FEMO are the worst for all levels of precisions.

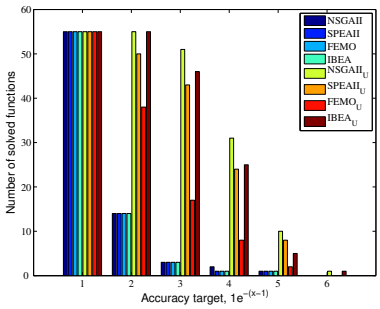

Figure 1. Comparison between traditional MOEAs and MO-UMDA$_c$ variants. Number of functions for which the MOEAs achieved the required level of precision in the best solution found.

C. Analysis of the influence per problem class

In the following step we investigate whether the impact of using the Gaussian univariate model with different selection strategies depends on the difficulty of the functions. We grouped the results of the algorithms for the 55 functions according to the 15 classes of functions described in Table I. Then, we determined whether some function classes were more difficult to solve than others.

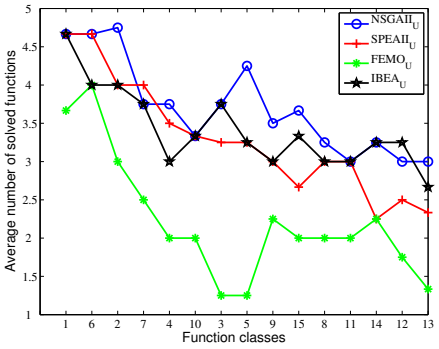

Figure 2. Number of times the MO-UMDA$_c$ variants find the optimum (all levels of precisions are considered) for each of the 15 classes of functions described in Table I. Dimension of the problem 2D.

Figure 2 and Figure 3 show the number of times that the MO-UMDA$_c$ variants found the optimum (all levels of precision are considered) for each of the 15 classes of functions described in Table I. Results are shown for problems of dimension 2D (Figure 2) and 3D (Figure 3). The X axis has been sorted from the problem classes with highest success rate (considering all the algorithms together) to the class of problems with smallest success rates.

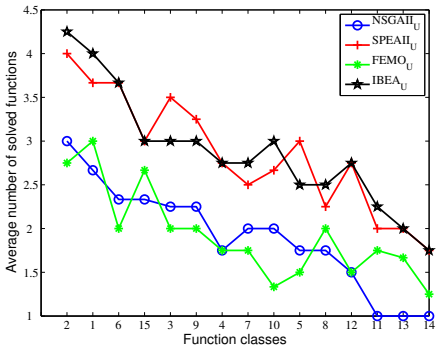

Figure 3. Number of times the MO-UMDA$_c$ variants find the optimum (all levels of precisions are considered) for each of the 15 classes of functions described in Table I. Dimension of the problem 3D.

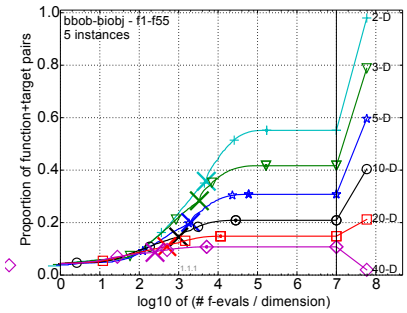

Figure 4. Empirical cumulative distribution (ECD) of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by decision space dimension (FEvals/DIM) for the 58 targets of the NSGA-IIu algorithm $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10-^{4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$. Results for all 55 functions and all dimensions.

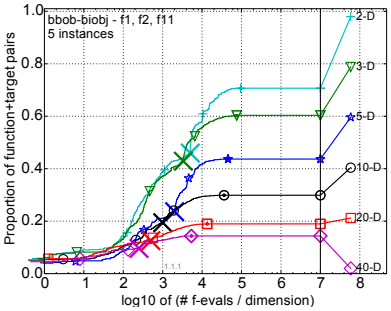

Figure 5. ECD for NSGA-IIu as described in Figure 4. Results for class 1 (separable, separable) functions and all dimensions.

It can be seen in Figures 2 and 3 that three easiest function classes (1, 2, 6) all contain separable functions. Four of the five most difficult functions (8, 11, 12, 13, 14) contain a multi-modal domain. The only class of functions containing a multi-modal domain that is not among the most difficult is class separate-multimodal, suggesting that having a separate domain can considerably ease the MOP for MO-UMDA.

While the explicit factorization made by MO-UMDA$_c$ makes reasonable that separate functions will be easier to optimize, finding that multi-modality is the most influential factor in the deterioration of algorithm behavior, it is an interesting fact, given that the other criteria of difficulty are also now to be challenging.

Two other important findings can be extracted from Figure 2 and Figure 3. The first is that while separable and multi-modal functions have a similar impact in all the MO-UMDA$_c$ variants, the impact of the functions is not always the same on the algorithms. For example, for dimension 2D, SPEAIIu outperforms IBEAu in function class 4 (separable - multi-modal). However, in function class 3 (separable - moderate), it is algorithm IBEAu the one that outperforms SPEAIIu. Our second finding is that one algorithm can have a good behavior for a given number of decision variables and degrade its performance when the number of variables increases. This happens to NSGAIIu, whose behavior deteriorates for 3D.

To analyze the scalability of the algorithm we inspect how the performance of the algorithms changes for the other dimensions considered in the experiments. Due to space restrictions, we present the results only for NSGAIIu. These results are shown in Figure 4 (for all the functions) and, in Figure 5, for the group of separable-separable functions. As expected, the success rate of the algorithm deteriorates with the increase in the dimensions. However, this deterioration is smaller for the class of separable functions, for which the trend evidenced for 2D and 3D is kept.

As a final step in the investigation of the algorithms, we studied the variability of the algorithm behaviors within the easiest and hardest classes of problems. What we want to determine is whether the good (respectively poor) performance for a given class of problems is due to a single function in each class, or all the functions within a class produce a similar impact in the behavior of the algorithm. Figure 6 and Figure 7 show this variability for the classes separable-separable and multimodal-multimodal. While in the first class of functions there is some variability, in the second class the algorithm fails to reach a high precision in a similar way for the three functions included in the class.

VI. Conclusions

In this paper we have conducted for the first time an analysis of the effect that different selection and replacement strategies have in the behavior of MOEAs using probabilistic modeling. We have focused on UMDA$_c$, a simple EDA that uses univariate Gaussian factorizations. Our experimental results have shown that even this simple algorithm is able to outperform traditional MOEAs based on crossover.

We have also addressed the relevant question of determining whether and to which extent different characteristics of the functions influence the behavior of MOEAs. Our experimental results show that separable functions are the easiest,

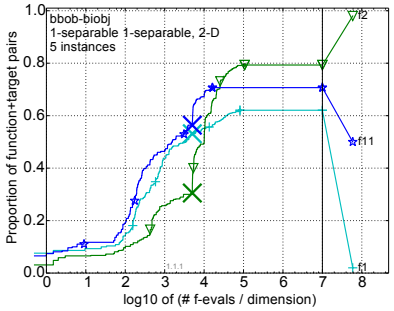

Figure 6. ECD for NSGA-IIu as described in Figure 4. Results for class 1 (separable, separable) functions in two dimensions.

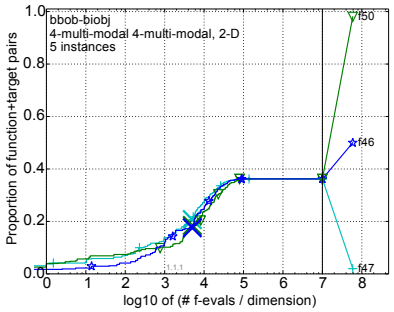

Figure 7. ECD for NSGA-IIu as described in Figure 4. Results for class 13 (multi-modal, multi-modal) functions in two dimensions.

and multimodal functions are the hardest for all the MOEAs analyzed. However, the choice of the selection method can provoke probabilistic modeling be more effective for some particular classes of functions. This effect seems to be stronger when considering the selection mechanisms used by SPEAIIu and IBEAu. Another issue that should be taken into account is that the ranking of the algorithms can change with the dimension of the problem considered.

VII. Acknowledgments

This work received support by CNPq Productivity Grant Nos. 306103/2015-0, CNPq Program Science Without Borders Nos.: 400125/2014-5, and CAPES program 99999.010574/2014-00, all financed by the Brazil Government. It has also received support by IT-609-13 program (Basque Government) and TIN2013-41272P (Spanish Ministry of Science and Innovation).

References

[1] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston/Dordrecht/London: Kluwer Acad. Publ., 2002.

[2] R. Cheng, Y. Jin, and K. Narukawa, "Adaptive reference vector generation for inverse model based evolutionary multiobjective optimization with degenerate and disconnected pareto fronts," in *Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 127–140.

[3] H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga, "Multi-objective optimization based on joint probabilistic modeling of objectives and variables," *IEEE Transactions on Evol. Comp*, vol. 18, no. 4, pp. 519–542, 2014.

[4] L. Marti, J. Garcia, A. Berlanga, C. A. Coello, and J. M. Molina, "On current model-building methods for multi-objective estimation of distribution algorithms: Shortcommings and directions for improvement," Department of Informatics of the Universidad Carlos III de Madrid, Madrid, Spain, Tech. Rep. GIAA2010E001, 2010.

[5] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization by learning and simulation of Bayesian and Gaussian networks," Department of Computer Science and Artificial Intelligence, University of the Basque Country, Technical Report EHU-KZAA-IK-4/99, 1999.

[6] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *CoRR*, vol. abs/1603.08785, 2016. [Online]. Available: http://arxiv.org/abs/1603.08785

[7] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization," *Evol. Comp*, vol. 1, no. 1, pp. 25–49, 1993.

[8] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, apr 2002.

[10] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[11] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb, "Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem," in *Parallel Problem Solving From Nature—PPSN VII*. Springer Berlin Heidelberg, 2002, pp. 44–53.

[12] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Inter. Conf. on Parallel Problem Solving from Nature (PPSN VIII*. Springer, 2004, pp. 832–842.

[13] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[14] D. Brockhoff, T. Tušar, D. Tušar, T. Wagner, N. Hansen, and A. Auger, "Biobjective Performance Assessment with the COCO Platform," *ArXiv e-prints*, May 2016.

[15] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA - A Platform and Programming Language Independent Interface for Search Algorithms," in *Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, ser. LNCS, vol. 2632. Berlin: Springer, 2003, pp. 494–508.