A Framework for Multi-Model EDAs with Model Recombination

Stefan Niemczyk[2], Raymond Chiong[3], and Mingxu Wan[1]

[1] University of Science and Technology of China (USTC), Hefei, Anhui, China
[2] Distributed Systems Group, University of Kassel, Kassel, Germany
[3] Swinburne University of Technology, Melbourne, Australia

Abstract. Estimation of Distribution Algorithms (EDAs) are evolutionary optimization methods that build models which estimate the distribution of promising regions in the search space. Conventional EDAs use only one single model at a time. One way to efficiently explore multiple areas of the search space is to use multiple models in parallel. In this paper, we present a general framework for both single- and multi-model EDAs. We propose the use of clustering to divide selected individuals into different groups, which are then utilized to build separate models. For the multi-model case, we introduce the concept of model recombination. This novel framework has great generality, encompassing the traditional Evolutionary Algorithm and the EDA as its extreme cases. We instantiate our framework in the form of a real-valued algorithm and apply this algorithm to some well-known benchmark functions. Numerical results show that both single- and multi-model EDAs have their own strengths and weaknesses, and that the multi-model EDA is able to prevent premature convergence.

1 Introduction

Traditional Evolutionary Algorithms (EAs) are based directly on the idea of survival of the fittest [25]. Only the strongest candidate solutions of each generation survive and become the parents for the next generations. Estimation of Distribution Algorithms (EDAs) work in a different way. These algorithms do not optimize candidate solutions, but learn how to *create* good solutions [16]. Instead of applying the conventional mutation and crossover operators, EDAs typically use selected candidate solutions to build a statistical model which is then sampled in order to create new points in the search space.

The models used in EDAs usually have a univariate probability distribution and, in the continuous case a unimodal one. Such a distribution can only represent one single basin of attraction for one optimum. Sooner or later, the algorithm has to abandon investigating all but one interesting region in the search space and converges. EDAs using multimodal distribution models, on the other hand, are often very complicated and thus, brittle.

In this paper, we present a framework that enables us to generate multiple (univariate/unimodal) probability models to explore different areas of the search space in parallel while, at the same time, maintain the simplicity and generality of the approach. Via this framework, we introduce a new real-valued EDA and evaluate its performance in both single- and multi-model cases. We show that the multi-model version has the ability to prevent premature convergence. The trade-off, however, is the slower convergence speed when it comes close to the global optimum.

The rest of this paper is organized as follows: in Section 2, we discuss some related work in detail. The proposed framework is presented in Section 3. We then report the numerical experiments carried out and highlight some of the main results obtained in Section 4. Finally, we draw conclusions in Section 5 and outline possible future works.

2 Related Work

Although being part of the EA family, EDAs are largely different from the traditional EAs (see [9, 12, 16, 17, 19, 25]). Instead of improving possible solutions step by step, EDAs try to evolve a model that describes how a perfect solution should look like. The central idea is that such a model is defined by several parameters which will converge during the optimization process. The span of possible values which can be sampled from it will become smaller and smaller over time. Eventually, the model should turn out to be so specific that only the global optimum can be sampled. However, many real-world problems are multimodal in nature and have many local or global optima.

One of the main challenges in EDAs is that the algorithms may lose diversity too quickly [23] and thus converge towards a local optimum. A simple way to increase diversity in EDAs is by mutating the model itself, leading to shifts in the sampled region. If the newly explored parts of the search space are inferior, the algorithms are likely to find a way back to the previous parameters. Otherwise, they escape the local optimum. This approach is relatively old and has already been applied in previous studies [2, 20].

Diversity can also be created without permanently mutating the model. Instead, a model mutation may be applied which only affects the sampling of one single genotype and is reverted thereafter. This way, the risk that a good model may get lost is circumvented. Such an operator is called the *sampling mutation* [23, 24].

These mechanisms, however, cannot prevent convergence to a single optimum. They are good mostly for unimodal optimization. Our approach, on the other hand, does not only maintain a diverse population; it is suitable for multimodal optimization without needing any further modification.

Other methods for improving diversity include those that make use of clustering techniques. One such example is the Evolutionary Bayesian Classifier-based Optimization Algorithm (EBCOA) proposed by Miquélez et al. [11]. In the EBCOA, the population is divided into a fixed number $|K|$ of classes before the model building phase. This is achieved by splitting the population *pop* into equal-sized groups of individuals from the fittest to the least fit one and assigning a label $k(p)$ to each individual $p \in pop$. Eventually, only a subset C of the $|K|$ classes are selected to facilitate learning. This can be justified because it emphasizes the differences between the classes and reduces noise [11, 23]. However, if the problem is multimodal, clusters limited by iso-fitness planes will span a wide area and be ill-shaped.

Lu and Yao [10] introduced a basic multi-model EDA scheme for real-valued optimization. This scheme utilizes clustering in a way similar to our work presented here. However, they focused mainly on numerical optimization whereas our aim is to have a general framework with possible instantiations of different algorithms for numerical optimization. Platel et al. [18] proposed a quantum-inspired EA, which is an EDA for bit-string based search spaces. This rather complicated algorithm utilizes a structured population similar to the use of demes in EAs, making it a multi-model EDA. Gallagher et al. [7] extended the Population-Based Incremental Learning (PBIL) algorithm [2] to real-valued optimization by using an Adaptive Gaussian mixture model density estimator. This approach can deal with multimodal problems too but, in our opinion, is more complex than multi-model algorithms that utilize clustering.

Our framework is general and does not bound to numerical or binary optimization. Also, the idea of model recombination has not been used in [7, 10, 18]. More EDA approaches that utilize clustering in different ways can be found in [1, 6, 14, 21]. Similar to the related work already discussed, they lack the generality and features of the approach presented here.

3 The Framework

In the optimization domain, it is generally not possible to determine whether the best solution currently known is situated on a local or a global optimum and thus, if the convergence is acceptable.

As aforementioned, single-model EDAs may run the risk of premature convergence to a local optimum. Finding a general mechanism for utilizing multiple models which *repel* each other may thus be a more efficient way to prevent premature convergence. Our framework provides an easy blueprint for creating EDAs that can use an arbitrary number of simple, unimodal models.

We present an idea of uniting classical EAs which utilize mutation and crossover with EDAs which consist of model construction and sampling steps. This framework uses multiple models at the same time – on one hand, these models correspond to the stochastic models in EDAs that represent good regions in the search space (that are used to sample new candidate solutions too); on the other hand, each of the models is also treated like a single individual in a classical EA, can be recombined with other individuals (models) and/or mutated (which equals the model sampling step).

The major contribution here is the introduction of the recombination operation into EDAs. Crossover operators are one of the main reasons why EAs excel in many domains. The typical types of crossover from Genetic Algorithms, the ternary crossover from Differential Evolution, the ρ-ary recombination from $(\mu/\rho \overset{+}{,} \lambda)$ Evolution Strategies [3, 25], or the sub-tree crossover of Genetic Programming (to even the population dynamics of Particle Swarm Optimization), can easily be applied within our framework. It thus becomes possible to utilize the well-known strengths of these operations which, so far, were not available in EDAs.

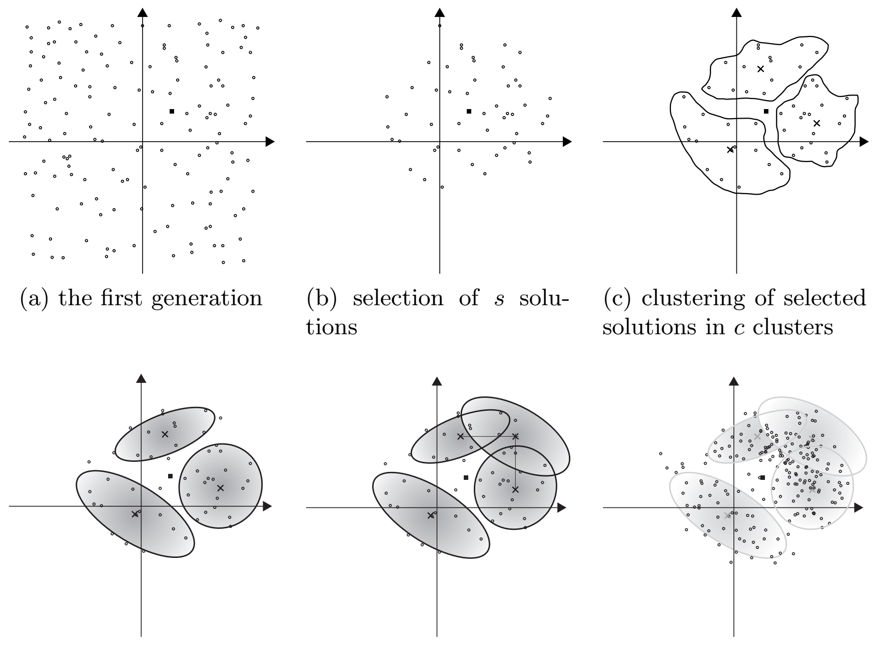
3.1 The Multi-Model EDA

The multi-model EDA is achieved in a very simple way: by building and optimizing n different models simultaneously. The flow of the algorithm can be described in eight steps as follows:

1. The first generation will be generated by sampling $n * m$ random new points uniformly distributed over the whole search space (Figure 1a).
2. The fitness of each candidate solution is evaluated.
3. After all candidate solutions have been evaluated, the best s individuals are selected by truncation selection (Figure 1b).
4. The selected points are clustered in c clusters (Figure 1c). It is important to note that c may be *1)* a fixed parameter, *2)* subject to self-adaption, or *3)* determined by the clustering algorithm itself on the fly.
5. One model is computed for each cloud of points (Figure 1d).
6. $n - c$ additional models are generated by model recombination (Figure 1e).
7. From each of the n models, m new points are sampled (i.e., $n * m$ new points in total, see Figure 1f).
8. If the termination criterion is not met, the algorithm continues at step 2.

By using multiple models created from point sets resulting from clustering the candidate solutions, we assume it is likely that some of the clusters do not reside on local optima. Furthermore, and perhaps more importantly, clusters located closely together in the search space may repel each other, thus increasing the chance of the search to escape local optima.

Assume, for instance, a one-dimensional real search space and normal distributions characterized by expected values μ and standard deviations σ as models, as sketched in Figure 2. If two clusters border to each other, the resulting models will significantly intersect (line z in Figure 2). Sampling model 2 may result in individuals occurring on the left of z, in the shaded region marked with L. After the next clustering step, the surviving individuals in L will likely be assigned to a cluster $1'$ (replacing 1), regardless of whether they stem from 1 or 2. The



(a) the first generation

(b) selection of s solutions

(c) clustering of selected solutions in c clusters

(d) the model building phase

(e) recombine two models to a new one

(f) a new generation is sampled

Fig. 1: Visualization of the steps of a search run of one particular real-valued instance (RVMMEDA) of our framework. The circles are candidate solutions, the $\times$ are cluster centroids and the square is the global optimum.

samples left of z are thus (likely) "lost" for model 2. Since this happens only on the left, the mean μ_2 will shift towards the right (μ_2'). The same will happen for model 1 with all samples right of z. Model 1 and 2 will not converge but be forced to move away from each other. This force works against a selection pressure that would cause a conventional EDA to converge. If a local optimum would be located at z, there is a good chance that the algorithm can escape. Premature convergence hence becomes less likely and the chance to find the basin of attraction of the global optimum increases.

3.2 The Real-Valued Multi-Model EDA (RVMMEDA)

The RVMMEDA is a trivial real-valued implementation of our new multi-model EDA and follows exactly the eight steps given before. A model is determined by the mean vector and the covariance matrix of a cluster of points. As in [4, 5, 15], we use the k-means [8] algorithm and a model representing a multi-dimensional normal distribution defined by the mean vector μ and the covariance matrix Σ

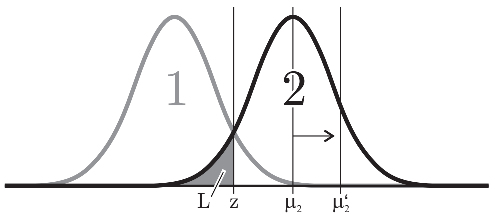


Fig. 2: Models repelling each other in a multi-model EDA.

of the candidate solutions in a cluster. Sampling new points from such a model is performed as follows:

1. Draw a standard normally-distributed random value for each dimension.
2. Scale these values with the square root of the Eigen value belonging to the corresponding dimension of Σ.
3. Rotate the point by multiplying and adding up the values for each dimension with the values of the Eigen vector belonging to the corresponding dimension of Σ, i.e., the iso-probability ellipsoids are rotated to the correct alignment.
4. Move the point by adding μ in order to ensure that the expected value of the samples equals the arithmetic mean of the points used for constructing the model.

Besides building a model from a cluster of points, new models can be created by recombining the existing ones. Here, we use a simple approach where two mean vectors are selected and a dominant crossover operation is applied similar to the one used in Evolution Strategies (see [3]). After this, a new covariance matrix will be computed from the two parent mean vectors and the newly generated vector.

3.3 A Framework Unifying EAs and EDAs

By introducing model recombination we create a framework that unifies EAs and EDAs. EAs are population-based optimization algorithms. Assume that the population of an EA applied to a real-valued optimization problem consists of *ps* individuals and that the complete population is replaced by its offspring in each iteration. In this case, the EA will create *ps* new candidate solutions in each generation by either mutating one parent individual or recombining two existing candidate solutions.

In the proposed multi-model EDA, n models exist in parallel and each of them is sampled exactly m times, hence resulting in $m * n = ps$ new points in every generation. If $n = ps$ and $m = 1$, each model corresponds to exactly one candidate solution. The model sampling process then avails to a mutation operation and the model recombination would equal a crossover operator in an EA. In this case, the multi-model EDA becomes a basic EA. In the other extreme end where $n = 1$ and $m = ps$, the multi-model EDA then proceeds in exactly the same way as an ordinary, single-model EDA.

Our multi-model EDAs thus bridge the gap between conventional EAs and EDAs, enabling the possibility to define intermediate forms between the two. An optimizer based on this may self-adapt and decide whether it would prefer to act more as an EA or EDA, depending on the current situation.

It should be noted that, although we instantiate our framework in the form of a real-valued algorithm for continuous search spaces, the fundamental idea is by no means limited to that. The algorithm could easily be applied to bit-string based search spaces, for example, by imposing it on top of the hBOA [17, 21] algorithm. Alternatively, we could also use it for Genetic Programming by building and sampling models according to [20].

4 Experiments and Results

4.1 Experimental Settings

To analyze the performance of both single- and multi-model EDAs in our framework, we performed experiments with five well-known numerical benchmarks [25]: the *Griewank*, the *Michalewicz*, the *Rosenbrock*, the *Summation Cancellation*, and the *Stair* functions for two different search space dimensions $d \in \{5, 25\}$. The population size ps was fixed to 1000 and no more than 10000 generations were performed. For the number of models n and the number of clusters c we tested all values in 1..10 while keeping $c \leq n$. In addition, we tried all mating pool sizes s from $\{200, 300, 400, 500, 600\}$. For each configuration, at least 10 to a maximum of 30 independent runs were performed.

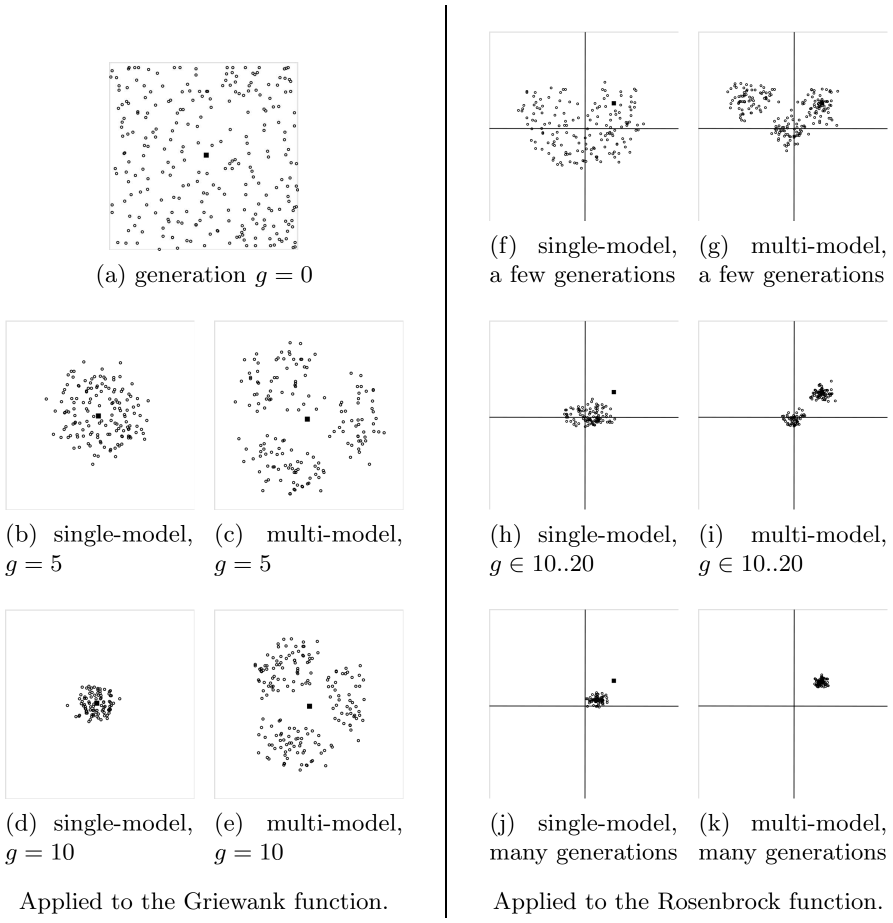
4.2 Experimental Results

Due to the resulting massive amount of experiments, we can only outline the main findings and trends that are of interest. A full report is provided in [13].

For the *Griewank* function, the minimum was found by the single-model EDA in every run. When more than one model was used, the global optimum was not found. The results became worse with increasing number of models and applications of the recombination operator. Here, we observed that the clusters have come very close to the global optimum and virtually surrounded it. However, the model-repelling mechanism worked so well that the clusters did not converge (see the left side of Figure 3). In practice, this would lead to the discovery of *robust* near-optimal solutions that, even with slight perturbations, retain their good features.

The *Michalewicz* function was hard for both single- and multi-model EDAs to solve. None of the approaches could fully solve it. They were only able to approach the global optimum with a precision of 10^{-4}.

Neither the single- nor the multi-model EDA found the global optimum of the *Rosenbrock* function more than five times in 30 runs. The RVMMEDA has


(a) generation $g = 0$

(b) single-model, $g = 5$

(c) multi-model, $g = 5$

(d) single-model, $g = 10$

(e) multi-model, $g = 10$

Applied to the Griewank function.

(f) single-model, a few generations

(g) multi-model, a few generations

(h) single-model, $g \in 10..20$

(i) multi-model, $g \in 10..20$

(j) single-model, many generations

(k) multi-model, many generations

Applied to the Rosenbrock function.

Fig. 3: The progress of single- and multi-model EDAs (the little square is the optimum).

been more efficient in terms of the mean of the achieved fitness, especially in the settings of a low crossover rate. As the *Rosenbrock* function has a long curved channel, most of the points were sampled in this channel by both EDAs after a few generations. The single-model EDA converged to a local optimum at $x_i = 0$ after the channel was reached. The RVMMEDA, however, was able to explore multiple optima at once, as shown in (the right side of) Figure 3. It can therefore escape the local optimum. After several additional generations, it sampled all points in the area around the global optimum. This clearly shows the strength of the proposed approach and the cluster repelling mechanism.

For the *Summation Cancelation* function, similar behavior as in the *Griewank* function was observed. The single-model EDA performed well with a high degree of successful runs and the multi-model EDA, for some settings, found the optimum only in 1 out of 30 runs. Here, we observed a higher average fitness when at least one model is created with crossover. However, if the number of models created this way in each generation is too high, the average fitness decreases again.

On the *Stair* function, the single-model EDA was unable to find the global optimum in any single run, whereas the RVMMEDA easily located it in all 30 runs across many configurations. From the results, we observed that for all configurations solving this problem more than one time, there was at least one model created with recombination. Together with the results from the *Summation Cancelation* function, this strongly indicates that the utility of model crossover is favorable. The *Stair* function is known to be a hard problem for a conventional EDA which usually converges after climbing only a few steps, misled by the neutrality on the stairs. The RVMMEDA, on the other hand, uses its crossover operator to jump onto other stairs.

In a nutshell, the single-model EDA has performed better on the *Griewank* and *Summation Cancelation* functions, while the multi-model EDA has done better on the *Stair* and (partially on) *Rosenbrock* functions. In addition, the multi-model EDA has demonstrated great potential in preventing premature convergence.

5 Conclusions and Future Work

In this paper, we have introduced a general and versatile framework for single- and multi-model EDAs. Our multi-model EDA is a new paradigm that aims to prevent the search process from getting stuck at local optima. Instead of just one area, it can explore different interesting regions of the search space at once.

By using the RVMMEDA, a specific algorithm derived from this framework, we studied the performance of single- and multi-model EDAs. Five different benchmark functions were used in the evaluation. Numerical experiments on these functions showed that the RVMMEDA variant is extremely good at preventing premature convergence. The *Rosenbrock* function serves as a good example of this, where the RVMMEDA has outperformed the single-model EDA. The *Stair* function is another such example which additionally shows that model recombination can be highly effective. However, the strength of the RVMMEDA could also be its drawback: it can quickly detect the basin of attraction of the global optimum, but convergence to the optimum itself is very slow. This drawback can also be its strength though, as the optima discovered tend to be robust (i.e., will retain their good characteristics even when perturbed).

In future work, we will test our framework on bit-string based search spaces by applying it to the benchmark given in [22]. We will further extend the framework with self-adaptation capabilities: when the clusters remain close to each other for some time, their numbers should automatically and slowly be reduced towards one in order to benefit from the better convergence behavior of a single-model EDA. We will introduce this method for general search spaces and also provide specific instantiations.

Bibliography

[1] C.W. Ahn and R.S. Ramakrishna. Clustering-based probabilistic model fitting in estimation of distribution algorithms. *IEICE Transactions on Information and Systems*, E89-D(1):381–383, 2006.

[2] S. Baluja. Population-based incremental learning – a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegy Mellon University, Pittsburgh, PA, USA, June 2, 1994.

[3] H.G. Beyer and H.P. Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[4] P.A.N. Bosman and D. Thierens. Mixed idas. Technical Report UU-CS-2000-45, Utrecht University, 2000.

[5] P.A.N. Bosman and D. Thierens. Advancing continuous idas with mixture distributions and factorization selection metrics. In *GECCO'01*, pages 208–212. Morgan Kaufmann, 2001.

[6] A. Cao, Y. Chen, J. Wei, and J. Li. A hybrid evolutionary algorithm based on edas and clustering analysis. In *Chin. Ctrl. Conf.*, pages 754–758. IEEE, 2007.

[7] M. Gallagher, M.R. Frean, and T. Downs. Real-valued evolutionary optimization using a flexible probability density estimator. In *GECCO'99*, pages 840–846, Orlando, USA, 1999. Morgan Kaufmann.

[8] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data – An Introduction to Cluster Analysis*, volume 59. Wiley Interscience, 1990.

[9] P. Larrañaga and J.A. Lozano, editors. *Estimation of Distribution Algorithms – A New Tool for Evolutionary Computation.* Springer, 2001.

[10] Quiang Lu and Xin Yao. Clustering and learning gaussian distribution for continuous optimization. *IEEE Transactions on Systems, Man, and Cybernetics Part C*, 35(2):195–204, 2005.

[11] T. Miquélez, E. Bengoetxea, and P. Larrañaga. Evolutionary computation based on bayesian classifiers. *International Journal of Applied Mathematics and Computer Science*, 14(3):335–349, 2004.

[12] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions i. binary parameters. In *PPSN IV*, pages 178–187. Springer, 1996.

[13] S. Niemczyk and T. Weise. A general framework for multi-model estimation of distribution algorithms. Technical report, University of Kassel, 2010. URL `http://www.it-weise.de/documents/files/NW2010AGFFMMEODA.pdf`.

[14] T. Okabe, Y. Jin, B. Sendhoff, and M. Olhofer. Voronoi-based estimation of distribution algorithm for multi-objective optimization. In *CEC'04*, volume 2, pages 1594–1601. IEEE, 2004.

[15] M. Pelikan and D.E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In *PPSN VI*, pages 385–394. Springer, 2000.

[16] M. Pelikan, D.E. Goldberg, and F.G. Lobo. A survey of optimization by building and using probabilistic models. Technical Report 99018, IlliGAL, 1999.

[17] M. Pelikan, K. Sastry, and E. Cantú-Paz, editors. *Scalable Optimization via Probabilistic Modeling – From Algorithms to Applications.* Springer, 2006.

[18] M.D. Platel, S. Schliebs, and N. Kasabov. Quantum-inspired evolutionary algorithm: A multimodel eda. *IEEE Trans. on Evol. Comp.*, 13(6):1218–1232, 2009.

[19] R. Armañanzas et al. A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6), 2008.
[20] R. Sałustowicz and J. Schmidhuber. Probabilistic incremental program evolution: Stochastic search through program space. In *9th European Conference on Machine Learning*, pages 213–220. Springer, 1997.
[21] K. Sastry and D.E. Goldberg. Multiobjective hboa, clustering, and scalability. In *GECCO'05*, pages 663–670. ACM, 2005.
[22] T. Weise et al. A tunable model for multi-objective, epistatic, rugged, and neutral fitness landscapes. In *GECCO'08*, pages 795–802. ACM, 2008.
[23] D. Wallin and C. Ryan. Maintaining diversity in edas for real-valued optimisation problems. In *FBIT'07*, pages 795–800. IEEE, 2007.
[24] D. Wallin and C. Ryan. On the diversity of diversity. In *CEC'07*, pages 95–102. IEEE, 2007.
[25] T. Weise. *Global Optimization Algorithms – Theory and Application.* it-weise.de, 2009. URL `http://www.it-weise.de/`.

```
%
@inproceedings{WNCW2011AFFMMEWMR,
  authors    = {Thomas Weise and Stefan Niemczyk and Raymond Chiong and
                Mingxu Wan},
  title      = {A Framework for Multi-Model EDAs with Model Recombination},
  booktitle = {Proceedings of the 4th European Event on Bio-Inspired
                Algorithms for Continuous Parameter Optimisation in
                Applications of Evolutionary Computation -- Proceedings of
                EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP,
                EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Part 1},
  month      = apr # {27--29,},
  year       = {2011},
  location   = {Torino, Italy},
  publisher = {Springer-Verlag GmbH: Berlin, Germany},
  series     = {Lecture Notes in Computer Science (LNCS)},
  volume     = {6624},
  pages      = {304--313},
  isbn       = {978-3-642-20524-8},
  doi        = {10.1007/978-3-642-20525-5_31},
}
```