Estimation of Distribution Algorithms: From Available Implementations to Potential Developments

Roberto Santana
Universidad Politécnica de Madrid
Madrid, Spain
roberto.santana@upm.es

ABSTRACT

This paper focuses on the analysis of estimation of distribution algorithms (EDAs) software. The important role played by EDAs implementations in the usability and range of applications of these algorithms is considered. A survey of available EDA software is presented, and classifications based on the class of programming languages and design strategies used for their implementations are discussed. The paper also reviews different directions to improve current EDA implementations. A number of lines for further expanding the areas of application for EDAs software are proposed.

Categories and Subject Descriptors

G.1 [**Optimization**]: Global optimization; G.3 [**Probabilistic methods**]

General Terms

Algorithms

Keywords

Estimation of distribution algorithms, probabilistic modeling, software, programming, virtual environment, mobile computing

1. INTRODUCTION

It is acknowledged that software availability contributes to the reproducibility of experimental results and the dissemination of research achievements. Advances in evolutionary computation (EC) are potentiated when the implementations of the introduced algorithms are made publicly available by the authors. Additionally, in order to boost practical applications of evolutionary algorithms, software developers should consider the conception of general and flexible software. Implementations of evolutionary algorithms will be more valuable if they could be applied to optimization problems arising in different contexts, and if they are intuitive and reusable, contributing to save the time of users less familiarized with EC theory.

One of the most active areas in EC research is the study of estimation of distribution algorithms (EDAs) [32, 45, 55]. Since its conception, research on EDAs has moved on multiple fronts. From their initial application to relatively simple binary problems, applications of EDAs have been extended to difficult non-binary discrete, continuous and mixed problems [36, 57]. Different variants of EDAs have been proposed for multi-objective [8, 59, 84] and dynamic functions [30]. Theoretical research on EDAs has also experienced a rapid development. However, since the use of probabilistic graphical models (PGMs) defended by EDAs developers is a fundamental shift in traditional research on evolutionary algorithms, a rapid embracement of this new methodology by the community of EC users is not expected. A related problem is that EDAs implementations are, in general, more complex than evolutionary algorithms based on the use of simple or heuristic genetic operators. Therefore, the availability of EDA software is very important for expanding the use of these algorithms.

The first part of this paper reviews available implementations of EDAs, analyzing their main characteristics and proposing a classification of these implementations that could be useful to users and developers. The paper does not survey the different classes of EDAs. For such type of analysis, the interested reader is suggested to consult [32, 55]. The second part of the paper focuses on the analysis of the features that could enhance the applicability of EDAs implementations. Although there are several software packages that provide implementations of EDAs, in some cases, the source code of the programs is not available. In other cases, the programs are only available upon request from the authors. The paper focuses on the analysis of those packages whose source code is available from internet.

The paper is organized as follows: In the next section, available EDA implementations are reviewed. Algorithms are grouped into three classes according to the programming language they have been implemented: C and C++ programs are grouped in the first class, Matlab programs in the second, and all implementations in any other language are grouped in the third class. In Section 3, a classification of EDA implementations according to the design strategy used to program the algorithms is introduced. Section 4 proposes several ways in which EDAs implementations could be enhanced. Section 5 discusses a number of ideas related to EDA implementations that could expand the application of

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

these algorithms. The conclusions of our paper and some lines for future research are given in Section 6.

2. REVIEW OF EDA IMPLEMENTATIONS

The analysis of the available software has shown that most implementations have been programmed in C++ and Matlab. This fact has motivated the classification of the EDA software presented in this section. The surveyed software are summarized in Table 1, that describes the software identifiers, the papers in which they have been presented and the programming language they have been implemented. In Table 2, the web pages from which one or more of the EDA implementations can be downloaded are listed.

2.1 C and C++ implementations

The first implementation of the extended compact genetic algorithm (ECGA) [24] was presented in [34]. It allows the user to define its own objective function. This implementation does not use advanced features of the C++ language such as templates and inheritance. Therefore, it makes easier to the user to modify the code and plug-in his own objective function using the C programming language alone.

An implementation of the χ-ary ECGA in C++ was introduced in [16]. χ-ary ECGA extends the application of non-overlapping marginal models to non-binary problems of χ-ary cardinality. The source code is also an extension of the original binary-coded ECGA and its previous implementation in C++ [34, 35].

Identifier	Reference	Language
MBOA, AMBOA	[49, 48]	C++
BOA	[53, 51, 52]	C++
Tree-EDA	[58]	C++
RM-EDA	[84]	C++,M
Hybrid-EDA	[83]	C++,M
Mateda	[65, 67]	M
ECGA	[34, 35, 69, 69, 70]	C++,M
LiO	[40]	Java
Evoptool	[79]	C++
AMALGAM	[7]	C
ParadisEO	[11, 77]	C++
gtEDA	[29]	M

Table 1: Description of surveyed EDA software, implemented in C, C++, Matlab (M) and Java.

Different implementations of tree-based EDAs [4, 56] are available. The software presented in [58] is available for download. A simple implementation of the Bayesian optimization algorithm (BOA) [52, 53] is presented in [50]. A more sophisticated implementation incorporating decision graphs is presented in [51].

The evolutionary optimization tool (Evoptool) [79] is an optimization toolkit that implements different evolutionary optimization algorithms. EDA implementations includes univariate EDAs such as compact GA (cGA) [25], population-based incremental learning (PBIL) [3], and UMDA [45]. Implementations of bivariate EDAs include MIMIC [14] and COMIT [5]. Additionally, different versions of Markov network based EDAs [64, 73, 75] are also implemented. They include the univariate and bivariate versions of DEUM [74, 76]. External algorithm implementations can be wrapped through patches. Using this procedure, Evoptool includes the implementation of the simple BOA [50]. Evoptool also includes several benchmark problems.

ParadisEO [11, 77] is a white-box object-oriented software framework dedicated to the flexible design of metaheuristics for optimization problems of both discrete and combinatorial nature. It contains general classes that allows the definition of generic discrete and continuous EDAs.

The mixed BOA (MBOA) [49] and the adaptive mixed BOA (AMBOA) [48] are conceived for the solution of mixed discrete-continuous problems. In contrast to MBOA, AMBOA copes with improved performance in the continuous domain due to variance-adaptation. The C++ implementations of these two EDAs are parallel.

Implementations of hybrid EDAs that use genetic operators probabilistic modeling are also available. A C++ implementation of an EDA with guided mutation is proposed in [83]. The guided mutation is proposed by incorporating the location information (the actual position in the search space) and the global statistical information, which is represented by a univariate marginal distribution model [83].

There are not many implementations of multi-objective EDAs. A C++ implementation of the regularity model-based multiobjective estimation of distribution algorithm (RM-EDA) [84] for continuous problems is available. RM-EDA is based on a regularity of the Pareto set in the decision space. At each generation, RM-EDA models a promising area in the decision space by a probability distribution whose centroid is a 1-D piecewise continuous manifold. ParadisEO-MOEO [33] is an extension of ParadisEO to multi-objective problems. It is a general-purpose software framework dedicated to the design and the implementation of evolutionary multiobjective optimization techniques. EDAs can be also implemented for this multi-objective module.

The C implementation of the adapted maximum-likelihood Gaussian model iterated density estimation evolutionary algorithm (AMaLGaM-IDEA) [7] includes different approaches to the solution of continuous optimization problems. In AMaLGaM-IDEA, the factorizations of the Gaussian distribution is either not factorized, factorized using a Bayesian factorization that is learned in a greedy fashion, or factorized using the univariate factorization [7].

2.2 Matlab implementations

An interface for Matlab of the C++ implementation of the χ-ary extended compact genetic algorithm is described in [69]. The source code is an extension of the original binary-coded ECGA and its previous implementation in C++ [34, 35]. A pure Matlab implementation of ECGA is presented in [70]. The implementation allows the user to modify the fitness function and run eCGA on user-defined problems.

A general toolkit for estimation of distribution algorithms (gtEDA) is presented in [29]. gtEDA implements a number of univariate and bivariate discrete EDAs. The software implements a number of toy problems and allows the user to test his own objective function. No multivariate EDA is included in the current implementation. A Matlab implementation of the RM-EDA [84] is also available.

Mateda2.0 [65, 67] implements single and multi-objective discrete and continuous optimization problems using EDAs based on undirected and directed probabilistic graphical models. It is conceived as an open package to allow users to incorporate different combinations of selection, learning, sampling, and local search procedures. Additionally, it includes methods to extract, process and visualize the structures learned by the probabilistic models.

2.3 Other language implementations

Among EDA implementations in Java is the LiO package [40] which includes GAs, particle swarm optimization (PSO) methods and EDAs implementations. The LiO library was initially designed to avoid implementing pieces of code which are frequently used when solving problems by means of metaheuristics and EC. In this sense, LiO can be seen as a source of objects necessary to build search algorithms [40]. LiO includes the implementation of several univariate and multivariate EDAs and allows the creation of generic EDAs.

3. APPROACHES TO EDA IMPLEMENTATION

From the analysis of the software presented in the previous section three different approaches to the implementation of EDAs are identified:

- Implementation of a single EDA.
- Independent implementation of multiple EDAs.
- Common modular implementation of multiple EDAs.

Many of the available implementations fall in the first class [34, 35, 70]. This type of software is usually conceived to test a new EDA or extend its current scope of application. The flexibility of these implementations to be combined with other EDAs is, in most of cases, rather limited.

To the second class of implementation belong software packages that provide independent implementations of different EDAs [29, 79]. Usually, other type of evolutionary algorithms are also included in these implementations. When the algorithms are independently implemented, none or few common programming code is shared by their implementations. This approach is particularly suitable when the set of implemented EDAs is very diverse. In some cases, the software allows the algorithms to be compared in similar conditions, (e.g. function testbeds and experimental benchmarks are provided).

The third class of implementations groups programming code conceived to take advantage of the modular structure shared by most of EDAs [11, 40, 66]. In this approach, the EDA components (e.g. learning and sampling methods) are independently programmed. EDA implementations can be then constructed as a particular combination of the components. Modularity also allows the conception and validation of new EDA proposals that combine different components.

A paradigmatic example of this type of implementations is LiO, in which, from the point of view of design, a resource is an object whose behavior is known, and is thought to be as independent as possible from the rest of the resources. Since most of EDAs share a common scheme of functioning, differing only on the probabilistic model used to learn the features of the population, they are all implemented in LiO by one class. The only exception is the PBIL algorithm whose incremental learning step is different from most EDAs.

4. ENHANCING EDA IMPLEMENTATIONS

There are several ways current EDAs can be enhanced [68, 71]. This section focuses on general implementation enhancements with a direct impact in the application of the algorithms. The following is a list of ideas that could be beneficial when incorporated to EDAs software.

- Parallel implementations.
- Automatic methods to compute the population size.
- Support for statistical analysis of experimental results.
- Predicting the time of convergence for the algorithm.
- Restart strategies based on different search situations.
- Advanced visualization of the EDA evolutionary path.
- Mining of the probabilistic models and information extraction from the EDA search.

Although much work has been devoted to parallel EDAs [15, 37, 38, 41, 47], only a few of the available EDAs implementations are parallel [11, 47]. The use of parallelism is critical for more efficient algorithms since available computational resources are increasingly parallel. However, since there exist different alternatives for EDA parallelization [15, 38, 42] the choice for the type of parallelization to use should be clearly stated. Furthermore, two or more forms of parallelism could be incorporated to the programs.

There are several papers that theoretically and empirically analyze the population sizes required by EDAs for convergence [54, 61, 82]. Nevertheless, with a few exceptions [47], EDAs implementations do not provide the user with automatic procedures to compute the critical or suggested population size. Computation of the critical population size using the bisection method or other procedures should be incorporated as a tool to validate the EDA behavior and investigate their scalability. Similarly, support for statistical analysis of the experimental results, i.e. implementation of statistical tests or summaries of the algorithms results that could be easily used as input for statistical analysis software should be part of EDA implementations.

Stop criteria used for most of EDAs implementations mainly refer to the number of generations, reaching an *a priori*-set fitness value or producing a very homogeneous population. Some modular EDA implementations [65], allows the user to implement stop criteria using historical information about the EDA search. However, there are other studies that propose predictive models of the EDA behavior [19, 22, 44, 46, 54]. At least for certain classes of functions, these predictive models could be employed to detect the stagnation and convergence of the algorithms. It will be very useful that EDA implementations could incorporate this type of predictive models for intelligently setting stop conditions or providing the user information about the current state of the search. For example, using some of these models, the EDA software could provide, at each generation, which is the probability that the algorithm will improve the current best fitness value in the next generations.

When stagnation of the search has been detected, restart strategies are recognized as a relatively efficient procedure to continue the search. Restart strategies based on different search situations have been included in some EDA implementations [7, 83]. Nevertheless, they should be incorporated to other EDA software as a way to enhance the robustness of EDA applications.

Visualization of different aspects of the EDA search can contribute to a more flexible design of the applications and

Reference	Language	Web pages
[49, 48]	C++	http://jiri.ocenasek.com/#Download
[53, 51, 52, 58]	C++	http://medal.cs.umsl.edu/software.php
[83, 84]	C++,M	http://cswww.essex.ac.uk/staff/zhang/code
[65, 67]	M	http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/MATEDA.html
[34, 35, 69, 69, 70]	C++,M	http://illigal.org/category/source-code/
[40]	Java	http://www.dsi.uclm.es/simd/SOFTWARE/LIO/
[79]	C++	http://airwiki.elet.polimi.it/index.php/Evoptool:_Evolutive_Optimization_Tool
[7]	C	http://homepages.cwi.nl/bosman/source_code.php
[11, 77]	C++	http://paradiseo.gforge.inria.fr/

Table 2: EDA software available from internet.

to identify previously unknown information of the problem domain. Most of EDA implementations support only basic visualization of the algorithms behavior. It is very common to plot the best and average fitness of the population at each generation. Nevertheless, visualization of other type of information (e.g. the evolution of the graphical models structure) should be incorporated to the software.

One of the most promising EDA applications is the automatic extraction of problem information from the analysis of the information produced during the EDA search [26, 27, 66]. In some real-world problems, the information extracted from the EDA evolution could be as important as the optimization results. In [65], methods to automatically find most frequent substructures in the graphical models learned by EDAs along the evolution are implemented. One difficulty associated to the implementation of such methods, is that the relevance of the information extracted from the models can be very problem dependent. Nonetheless, confronting the user with regularities identified and extracted from the models could contribute to effective knowledge discovery.

There are other areas [57, 68] where established and current research on EDAs could be translated into novel features of the EDA implementations. Such features could improve the efficiency of EDAs, enhance the EDA user's experience, and provide valuable information about the characteristics of the problem.

5. EXPANDING THE SCOPE OF EDA APPLICATIONS

This section analyzes some research areas where the implementation of EDAs could lead to expanding the applicability of these algorithms. Among these research areas are the following:

- Make EDA implementations available in commonly used scripting programming languages.
- Add EDA implementations to problem-domain specialized optimization packages.
- Take profit of the probabilistic modeling resources implemented in R and Matlab to investigate novel EDAs variants.
- Determine the feasibility of implementing EDAs in PDIs, mobile telephones and other portable devices.
- Study potential applications of EDA implementations in virtual environments.

There are three main reasons that point to the convenience of implementing EDAs in scripting languages such as Perl [80] and Python [31]. The first reason is that these languages serve to interface many different devices, protocols, applications, and file formats [43], allowing to address diverse optimization problems, possibly implemented in different languages. The second reason is that there are several modules already implemented in these scripting languages that could be reused for the implementation of EDAs, in particular modules related with EAs [6, 17, 43, 72] and machine learning techniques [1, 23, 72, 81]. Finally, and perhaps the most important reason, is that Perl and Python are the languages of choice for many scientific communities. EDAs implemented in these languages could be tested in a wide variety of real-world optimization problems.

EDAs have been applied with good results to a variety of domains. These applications have followed, in most of cases, a single case approach. One example of these domains is Bioinformatics, where an impressive record of EDA applications have been reported [2]. However, none of the specialized packages used to solve problems from Bioinformatics incorporates an EDA module. Focusing on the implementation of generic EDA modules, that can be applied to different types of problems with minor changes and within a specialized programming environment can be worth in terms of the popularization of these optimization algorithms.

Software such as R [13] and Matlab [78] provide sophisticated implementations of probabilistic modeling and machine learning methods. The available probabilistic modeling algorithms can be used to ease the conception and validation of different EDA approaches. This is one of the pillars of the MATEDA implementations [65, 67]. The same idea can be developed using R or any other software for which a library of statistical and machine learning methods is available.

Some of the technological developments of recent years include the emergence of new powerful computational platforms such as graphical cards, cell broad band engines,and the popularization of mobile computing devices such as mobile telephones, PDI and other portable devices. These platforms are candidates for the implementation of EDAs and other EAs. The challenge is the identification of the target problems. There are many common day tasks that involves the solution of an optimization problem. Other possible domains of applications for EDAs would be image evolution or gaming. Future research on EDA implementations should evaluate the most appropriate programming solutions for the most recent computational platforms.

Some recent research has successfully implemented EAs in GPGPU cards [21, 39, 62], and mobile phones [12, 20]. Algorithms like UMDA have also been implemented in the cell broadband engine used by PlayStation 3 game console [60]. It is reasonable to expect that mobile telephones and

other devices will soon contain implementations of probabilistic graphical models to be used in the solution of predictive tasks in different contexts. Therefore, it will be a good idea to implement EDAs in Android [18] and other languages used by these devices. A related question is how to find the best ways to allow the new class of EA users an intuitive interaction with the EA software. Portable devices may be instrumental in expanding the range of applications for interactive evolutionary algorithms.

Virtual environments such as Second Life [63] have been increasingly used as an arena for the investigation of machine learning algorithms. These environments provide different benchmarks for the interactions between human users and artificial intelligence creatures or bots. An appealing attribute of these environments is the constant flux of information that resembles what occurs in the *real* physical world. In virtual environments like Second Life, sensors actively searches information about avatars movements and behaviors, and artificial intelligence algorithms take decisions based on this information. Virtual environment are an excellent ground to investigate genetic-based machine learning algorithms that use probabilistic models such as advanced classifier systems [9, 10]. The main limitation in the implementation of EDAs in virtual environments is related to the constraints in the computational power provided by these environments. Programming languages like Linden scripting [28] set limits to the number of variables and to the complexity of the used data structures. Nevertheless, current computational power is at least sufficient to allow the implementation of simple EDAs such as UMDA.

6. CONCLUSIONS

EDA software is an important component for expanding the application of these algorithms and improving their performance in different domains. In this paper I have reviewed different issues related with the implementation of EDAs. Although some EDA implementations are available from the authors upon request, I have focused on those implementations which can be directly accessed from internet. I have proposed a classification of these implementations and argued for the conception of modular implementations which allow the evaluation of different EDA variants. The paper also includes proposals for enhancing current EDAs implementations and extending the scope of EDAs applications to other unexplored areas.

For many years, research on EDAs has mainly focused in improving the EDAs components (learning, sampling, selection, etc.) and the way they interact. In particular, theoretical and empirical analysis of a wide variety of PGMs have received much attention. The question of how to implement EDAs has not received a similar attention. However, implementations are not merely a tool to test the research ideas. Clever implementations are a direct way to incorporate the research results into the practice, and are the ultimate tool to determine the advantages of the proposed algorithms over competitive methods. For EDAs, rethinking the way software is conceived may contribute to potentiate their use by other communities. Similarly, by investigating how to program EDAs for the emergent computational platforms, we can expand the potential range of application of these methods and set the path for new directions to advance the research on these algorithms.

7. ACKNOWLEDGMENTS

This work has been partially supported by the TIN2010-20900-C04-04, Consolider Ingenio 2010 - CSD2007-00018, and the CajalBlueBrain projects (Spanish Ministry of Science and Innovation).

8. REFERENCES

[1] D. Albanese, S. Merler, G. Jurman, R. Visintainer, and C. Furlanello. MLPY machine learning Py, 2010. `http://mloss.org/software/view/66/`.

[2] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J. L. Flores, J. A. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, and P. Larrañaga. A review of estimation of distribution algorithms in bioinformatics. *BioData Mining*, 1(6):doi:10.1186/1756–0381–1–6, 2008.

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

[4] S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In D. H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 30–38, San Francisco, CA., 1997. Morgan Kaufmann.

[5] S. Baluja and S. Davies. Fast probabilistic modeling for combinatorial optimization. In *Proceedings of 15th National Conference on Artificial Intelligence AAAI-98*, Madison, Wisconsin, July 1998. American Association for Artificial Intelligence.

[6] D. Blank, D. Kumar, L. Meeden, and H. Yanco. The Pyro toolkit for AI and robotics. *AI magazine*, 27(1):39, 2006.

[7] P. A. Bosman. On empirical memory design, faster selection of Bayesian factorizations and parameter-free Gaussian EDAs. In *Proceedings of the 11th Genetic and Evolutionary Computation Conference GECCO-2011*, pages 389–396. ACM, 2009.

[8] P. A. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.

[9] M. Butz, M. Pelikan, X. Llorá, and D. E. Goldberg. Effective and reliable online classification combining XCS with EDA mechanisms. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 249–274. Springer, 2006.

[10] M. V. Butz, M. Pelikan, X. Llorá, and D. E. Goldberg. Automated global structure extraction for effective local building block processing in XCS. *Evolutionary Computation*, 14(3):345–380, 2006.

[11] S. Cahon, N. Melab, and E. Talbi. ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.

[12] I. Carreras and D. Linner. Self-evolving applications over opportunistic communication systems. In

Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on, pages 153–158. IEEE, 2010.

[13] M. Crawley. *The R book*. John Wiley & Sons Inc, 2007.

[14] J. S. De Bonet, C. L. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 424–430. The MIT Press, Cambridge, 1997.

[15] L. de la Ossa, J. A. Gámez, and J. M. Puerta. Migration of probability models instead of individuals: An alternative when applying the island model to EDAs. In *Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242, pages 242–252. Springer, 2004.

[16] L. de la Ossa, K. Sastry, and F. G. Lobo. χ-ary extended compact genetic algorithm in C++. IlliGAL Report 2006013, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2006.

[17] W. deLandgraaf, A. Eiben, and V. Nannen. Parameter calibration using meta-algorithms. In *Proceedings of the 2007 Congress on Evolutionary Computation CEC-2007*, pages 71–78. IEEE Press, 2007.

[18] J. DiMarzio. *Android: a programmer's guide*. McGraw-Hill Osborne Media, 2008.

[19] C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano. A quantitative analysis of estimation of distribution algorithms based on Bayesian networks. *IEEE Transactions on Evolutionary Computation*, 2011. Accepted for publication.

[20] J. Fajardo and C. Oppus. A mobile disaster management system using the Android technology. *WSEAS Transactions on Communications*, 9(6):343–353, 2010.

[21] M. Franco, N. Krasnogor, and J. Bacardit. Speeding up the evaluation of evolutionary learning systems using GPGPUs. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1039–1046. ACM, 2010.

[22] C. González, J. A. Lozano, and P. Larrañaga. Mathematical modeling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions. *International Journal of Approximate Reasoning*, 31(4):313–340, 2002.

[23] A. Gouws. A Python implementation of graphical models. Master's thesis, Faculty of Engineering. Stellenbosch University, 2010.

[24] G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.

[25] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. IlliGAL Report No. 97006, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, 1997.

[26] M. Hauschild, M. Pelikan, C. Lima, and K. Sastry. Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. In D. Thierens et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2007*, volume I, pages 523–530, London, UK, 2007. ACM Press.

[27] M. Hauschild, M. Pelikan, K. Sastry, and D. E. Goldberg. Using previous models to bias structural learning in the hierarchical BOA. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation GECCO-2008*, pages 415–422, New York, NY, USA, 2008. ACM.

[28] J. Heaton. *Introduction to Linden scripting language for Second Life*. Heaton Research, Inc., 2007.

[29] Y. Hua, W. Wen-Quan, and L. Zhong. General toolkit for discrete estimation of distribution algorithms. In *Proceedings of the 2010 International Conference of Information Science and Management Engineering (ISME)*, volume 2, pages 212–215. IEEE.

[30] M. Kobliha, J. Ocenasek, and J. Schwarz. Bayesian optimization algorithm in dynamic environment. In *Proceedings of the Mendel 2005 11th Internacional Conference on Soft Computing*, pages 15–20, Brno, CZ, 2005.

[31] H. Langtangen. *Python scripting for computational science*. Springer Verlag, 2004.

[32] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[33] A. Liefooghe, L. Jourdan, and E. G. Talbi. A unified model for evolutionary multiobjective optimization and its implementation in a general purpose software framework: ParadisEO-MOEO. Technical Report Research Report RR-6906, INRIA, 2009.

[34] F. G. Lobo and G. R. Harik. Extended compact genetic algorithm in C++. IlliGAL Report No. 99016, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.

[35] G. Lobo Fernando, S. Kumara, and R. Harik Georges. Extended compact genetic algorithm in C++ (version 1.1). IlliGAL Report No. 2006012, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2006.

[36] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Springer, 2006.

[37] J. A. Lozano, R. Sagarna, and P. Larrañaga. Parallel estimation of distribution algorithms. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 125–142. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[38] J. Madera, E. Alba, and A. Ochoa. Parallel estimation of distribution algorithms. In E. Alba, editor, *Parallel Metaheuristics*, pages 203–222. John Wiley & Sons, 2005.

[39] O. Maitre, L. A. Baumes, N. Lachiche, A. Corma, and P. Collet. Coarse grain parallelization of evolutionary algorithms on GPGPU cards with EASEA. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1403–1410, New York, NY, USA, 2009. ACM.

[40] J. L. Mateo and L. de la Ossa. LiO an easy and flexible library of metaheuristics. Technical Report

DIAB-06-04-1, Department of Computing Systems, Escuela Politecnica Superior de Castilla La Mancha, Albacete, Spain, 2007.

[41] A. Mendiburu, J. Lozano, and J. Miguel-Alonso. Parallel implementation of EDAs based on probabilistic graphical models. *IEEE Transactions on Evolutionary Computation*, 9(4):406–423, 2005.

[42] A. Mendiburu, J. Miguel-Alonso, J. A. Lozano, M. Ostra, and C. Ubide. Parallel EDAs to create multivariate calibration models for quantitative chemical applications. *Journal of Parallel Distributed Computation*, 66(8):1002–1013, 2006.

[43] J. Merelo Guervós, P. Castillo, and E. Alba. Algorithm:: Evolutionary, a flexible Perl module for evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, pages 1–19, 2009.

[44] H. Mühlenbein and T. Mahnig. Convergence theory and applications of the Factorized Distribution Algorithm. *Journal of Computing and Information Technology*, 7(1):19–32, 1998.

[45] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lectures Notes in Computer Science*, pages 178–187, Berlin, 1996. Springer.

[46] J. Ocenasek. Entropy-based convergence measurement in discrete estimation of distribution algorithms. In J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, pages 39–50. Springer, 2006.

[47] J. Ocenasek, E. Cantú-Paz, M. Pelikan, and J. Schwarz. Design of parallel estimation of distribution algorithms. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 187–204. Springer, 2006.

[48] J. Ocenasek, S. Kern, N. Hansen, and S. M. and. P. Koumoutsakos. A mixed Bayesian optimization algorithm with variance adaptation. *Lecture Notes in Computer Science*, pages 352–361, 2004.

[49] J. Ocenasek and J. Schwarz. Estimation of distribution algorithm for mixed continuous-discrete optimization problems. In *Proceedings of the 2nd Euro-International Symposium on Computational Intelligence*, pages 227–232, Kosice, Slovakia, 2002. IOS Press.

[50] M. Pelikan. A simple implementation of Bayesian optimization algorithm in C++ (version1. 0). IlliGAL Report No. 99011, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 1999.

[51] M. Pelikan. The Bayesian optimization algorithm (BOA) with decision graphs. IlliGAL Report No. 2000025, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, May 2000.

[52] M. Pelikan. *Bayesian optimization algorithm: From single level to hierarchy.* PhD thesis, University of Illinois, 2002.

[53] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, volume I, pages 525–532, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

[54] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Bayesian optimization algorithm, population sizing, and time to convergence. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000*, pages 275–282, 2000.

[55] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[56] M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, London, 1999. Springer.

[57] M. Pelikan, K. Sastry, and E. Cantú-Paz, editors. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Studies in Computational Intelligence. Springer, 2006.

[58] M. Pelikan, K. Sastry, and D. E. Goldberg. Implementation of the dependency-tree estimation of distribution algorithm in C++. MEDAL Report No. 2006010, Missouri Estimation of Distribution Algorithms Laboratory (MEDAL), 2006.

[59] M. Pelikan, K. Sastry, and D. E. Goldberg. Multiobjective estimation of distribution algorithms. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 223–248. Springer, 2006.

[60] C. Pérez-Miguel, J. Miguel-Alonso, and A. Mendiburu. Porting estimation of distribution algorithms to the cell broadband engine. *Parallel Computing*, 36(10-11):618–634, 2010.

[61] P. Pošík. Preventing premature convergence in a simple EDA via global step size setting. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Parallel Problem Solving from Nature - PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 549–558, Dortmund, Germany, 2008. Springer.

[62] P. Pospíchal, J. Jaros, and J. Schwarz. Parallel Genetic Algorithm on the CUDA Architecture. *Applications of Evolutionary Computation*, pages 442–451, 2010.

[63] M. Rymaszewski. *Second life: The official guide.* Sybex, 2007.

[64] R. Santana. A Markov network based factorized distribution algorithm for optimization. In *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 337–348, Dubrovnik, Croatia, 2003. Springer.

[65] R. Santana, C. Bielza, P. Larrañaga, J. A. Lozano,

C. Echegoyen, A. Mendiburu, R. Armañanzas, and S. Shakya. MATEDA: Estimation of distribution algorithms in MATLAB. *Journal of Statistical Software*, 35(7):1–30, 2010.

[66] R. Santana, C. Bielza, J. A. Lozano, and P. Larrañaga. Mining probabilistic models learned by EDAs in the optimization of multi-objective problems. In *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference GECCO-2009*, pages 445–452, New York, NY, USA, 2009. ACM.

[67] R. Santana, C. Echegoyen, A. Mendiburu, C. Bielza, J. A. Lozano, P. Larrañaga, R. Armañanzas, and S. Shakya. MATEDA: A suite of EDA programs in Matlab. Technical Report EHU-KZAA-IK-2/09, Department of Computer Science and Artificial Intelligence, University of the Basque Country, February 2009.

[68] R. Santana, P. Larrañaga, and J. A. Lozano. Research topics on discrete estimation of distribution algorithms. *Memetic Computing*, 1(1):35–54, 2009.

[69] K. Sastry, L. de la Ossa, and F. G. Lobo. χ-ary extended compact genetic algorithm for Matlab in C++. IlliGAL Report 2006014, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2006.

[70] K. Sastry and A. Orriols-Puig. Extended compact genetic algorithm in Matlab. IlliGAL Report 2007009, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2007.

[71] K. Sastry, M. Pelikan, and D. E. Goldberg. Efficiency enhancement of estimation of distribution algorithms. In M. Pelikan, K. Sastry, and E. Cantú-Paz, editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, Studies in Computational Intelligence, pages 161–186. Springer, 2006.

[72] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. PyBrain. *The Journal of Machine Learning Research*, 11:743–746, 2010.

[73] S. Shakya. Markov random field modelling of genetic algorithms. Technical report, The Robert Gordon University, Aberdeen, UK, 2004.

[74] S. Shakya. *DEUM: A framework for an Estimation of Distribution Algorithm based on Markov Random Fields*. PhD thesis, The Robert Gordon University. School of Computing, Aberdeen, UK, 2006.

[75] S. Shakya and R. Santana. An EDA based on local Markov property and Gibbs sampling. In M. Keijzer, editor, *Proceedings of the 2008 Genetic and evolutionary computation conference (GECCO)*, pages 475–476, New York, NY, USA, 2008. ACM.

[76] S. K. Shakya, J. A. McCall, and D. F. Brown. Updating the probability vector using MRF technique for a Univariate EDA. In E. Onaindia and S. Staab, editors, *Proceedings of the Second Starting AI Researchers' Symposium*, pages 15–25, Valencia, Spain, 2004. IOS press.

[77] E.-G. Talbi. *From Design to Implementation: A Unified View of Metaheuristics*. Wiley, 2009.

[78] The MathWorks, Inc. *MATLAB – The Language of Technical Computing, Version 7.5*. The MathWorks, Inc., Natick, Massachusetts, 2007.

[79] G. Valentini, L. Malago, and M. Matteucci. Evoptool: An extensible toolkit for evolutionary optimization algorithms comparison. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC-2010)*, pages 1–8. IEEE.

[80] P. Wainwright, S. Cozens, A. Calpini, A. Corliss, and J. Merelo-Guervos. *Professional Perl Programming*. Wrox Press Ltd. Birmingham, UK„ 2001.

[81] B. Wilczyński and N. Dojer. BNFinder: exact and efficient method for learning Bayesian networks. *Bioinformatics*, 25(2):286, 2009.

[82] T.-L. Yu, K. Sastry, D. E. Goldberg, and M. Pelikan. Population sizing for entropy-based model building in genetic algorithms. In D. Thierens et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2007*, volume I, pages 601–608, London, UK, 2007. ACM Press.

[83] Q. Zhang, J. Sun, and E. P. K. Tsang. Evolutionary algorithm with guided mutation for the maximum clique problem. *IEEE Transactions on Evolutionary Computation*, 9(2):192–200, 2005.

[84] Q. Zhang, A. Zhou, and Y. Jin. RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63, 2008.