Neurocomputing 598 (2024) 128043

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Original Software Publication

EDAspy: An extensible python package for estimation of distribution algorithms

Vicente P. Soloviev *, Pedro Larrañaga, Concha Bielza

Universidad Politécnica de Madrid, Artificial Intelligence Department, Campus de Montegancedo, Madrid, Spain

ARTICLE INFO

Communicated by S. Wang

Keywords:
Estimation of distribution algorithm
Evolutionary algorithm
Bayesian network
Benchmarking

ABSTRACT

Estimation of distribution algorithms (EDAs) are a type of evolutionary algorithms where a probabilistic model is learned and sampled in each iteration. EDAspy provides different state-of-the-art implementations of EDAs including the recent semiparametric EDA. The implementations are modularly built, allowing for easy extension and the selection of different alternatives, as well as interoperability with new components. EDAspy is totally free and open-source under the MIT license.

1. Introduction

Estimation of distribution algorithms (EDAs) [1] are a type of evolutionary algorithms [2] in which traditional mutation and crossover operators are replaced by a probabilistic model that is iteratively learned and sampled during the optimization process. EDAs have been successfully applied to a wide range of tasks [3–6]. See [7] for a review on EDAs applied to solve machine learning tasks. In recent meetings within the field of EDAs [8] a need for establishing an EDA reference library has been identified. EDAspy is proposed to satisfy this need for the scientific community working on this topic.

In this paper we present a python package in which several EDA implementations are efficiently designed. The different optimizers are easily called and can be tuned in a user friendly mode. Each EDA variant is built using different available modules, which can be customly selected to build a new implementation. These variants can be easily extended and interoperate with new components.

2. Background

Algorithm 1 shows the pseudocode of the EDA baseline. Firstly, random population (G_0) with size N is sampled (line 1). Secondly, population G_{t-1} is evaluated (line 3) and ranked (line 4) according to a given cost function $g(\cdot)$. Thirdly, a probabilistic model is learned from a fraction α of the best individuals, i.e., the top $\lfloor \alpha N \rfloor$ solutions (line 5). Finally, a new population is sampled (line 6). These four steps are iterately repeated until the stopping criterion is met.

Depending on the complexity of the probabilistic model and the nature of the optimization problem, different EDA variants are identified in the literature.

Algorithm 1 Estimation of distribution algorithm

Input: Population size N, selection ratio α, cost function g
Output: Best individual $\boldsymbol{x}'$ and cost found $g(\boldsymbol{x}')$

1: $G_0 \leftarrow N$ individuals randomly sampled
2: **for** $t = 1, 2, \ldots$ until stopping criterion is met **do**
3: Evaluate G_{t-1} according to $g(\cdot)$
4: $G^S_{t-1} \leftarrow$ Select top $\lfloor \alpha N \rfloor$ individuals from G_{t-1}
5: $f_{t-1}(\cdot) \leftarrow$ Learn a probabilistic model from G^S_{t-1}
6: $G_t \leftarrow$ Sample N individuals from $f_{t-1}(\cdot)$
7: **end for**

Univariate approaches assume independence among the variables, and a probability distribution is fitted independently to each of them. EDAspy uses independent Gaussian distributions, kernel density estimation (KDE) or categorical probability distributions, depending on the EDA variant and the nature of the data.

Multivariate approaches contemplate dependencies between the variables using different probabilistic models. EDAspy uses multivariate Gaussians or different types of Bayesian networks (BNs) [9], corresponding to different EDA versions.

3. Software framework

Fig. 1 represents the high order representation of the previously mentioned modules in EDAspy. In general, an EDA implementation is applied to a cost function to be minimized, and some results are

* Corresponding author.
E-mail address: vicente.perez.soloviev@alumnos.upm.es (V.P. Soloviev).

https://doi.org/10.1016/j.neucom.2024.128043
Received 12 February 2024; Received in revised form 19 April 2024; Accepted 7 June 2024
Available online 12 June 2024

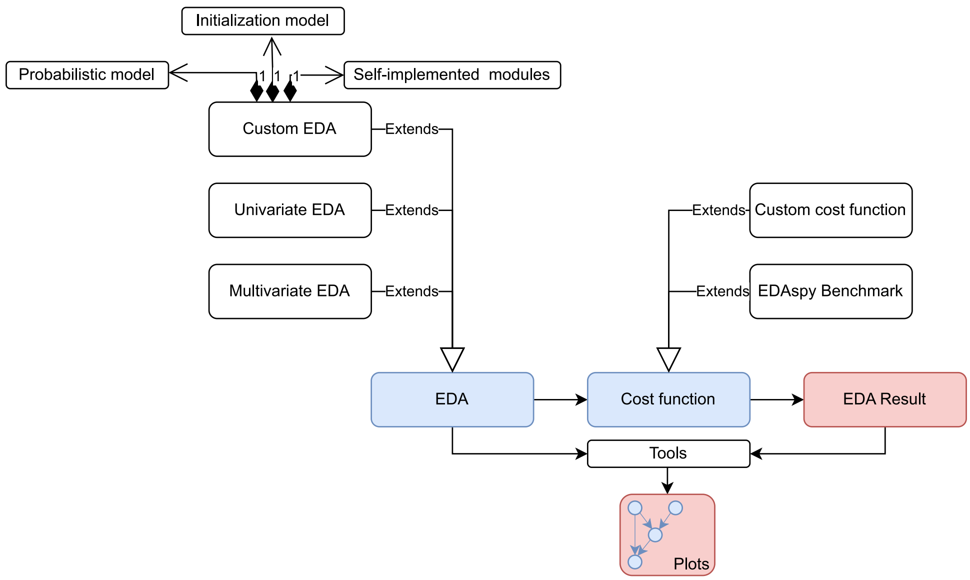


Fig. 1. High order organization of the EDAspy library.

found. There are several EDA implementations available in the library organized in univariate and multivariate modules, but it is also possible to build a customizable implementation by integrating the already available components with other modules (optionally) in the EDA object. Regarding the cost function, there are several benchmarks implemented. In addition, a custom cost function can be used. Once the optimizer has converged, several information and plots can be extracted from the execution.

Moreover, although the library has been built modular in order to allow the integration with new custom implementations, the EDA optimizer can be easily extended and built from scratch by the user without using *Custom EDA* module facilities.

`EDAspy` is organized in different modules:

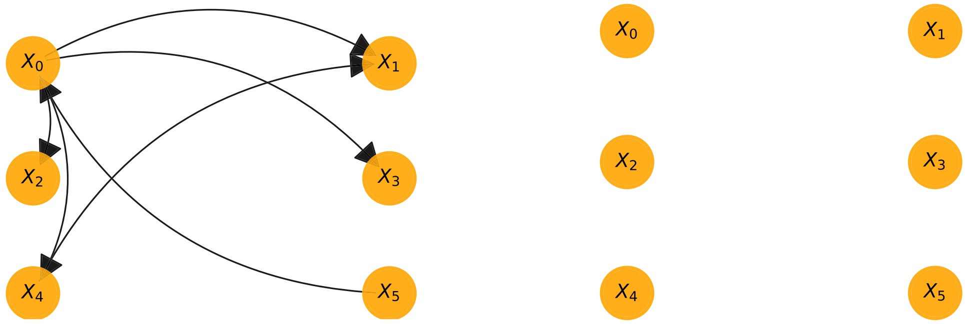
- **Benchmarks**. Different test functions for benchmarking and comparing the different optimizers are included. Toy discrete functions such as OneMax [10] and benchmark suites such as IEEE CEC 2014 [11] are included.
- **Univariate**. The following univariate approaches in which no dependencies between variables are considered: univariate marginal distribution algorithm (UMDA) for (i) binary [12] (UMDA_B), (ii) categorical (UMDA_D), and (iii) continuous optimization [13] (UMDA_C); (iv) kernel EDA [14] (u_KEDA); and (v) population-based incremental learning algorithm [15] (PBIL).
- **Multivariate**. The following multivariate approaches in which dependencies between variables are considered: (i) estimation of Bayesian network algorithm [1] (EBNA), (ii) estimation of multivariate normal algorithm [1] (EMNA), (iii) estimation of Gaussian network algorithm [16] (EGNA), (iv) semiparametric EDA [17] (SPEDA), and (v) multivariate kernel density EDA [17] (m_KEDA), (vi) Bayesian optimization algorithm (BOA) [18] in which a discrete BN, a multivariate Gaussian distribution, a Gaussian BN, a semiparametric BN, a kernel density estimated BN, and a discrete BN are iteratively learned, respectively.
- **Custom**: this module includes the different components to build a custom EDA variant and is divided into probabilistic and initialization models.
  - **Probabilistic model**. The following components are implemented for learning and sampling. Regarding univariate probabilistic models, (i) binary, (ii) discrete, (iii) Gaussian, and (iv) KDE models are considered. Regarding Bayesian networks, (v) Gaussian, (vi) semiparametric, (vii) KDE, and (viii) discrete models are available. Other models include (ix) multivariate Gaussian.
  - **Initialization model**. Uniform sampling meeting landscape user defined bounds, Latin hypercube sampling [19] and initialization from a given dataset are available to build the first population of the EDA.
  - **Self-implemented modules**. This includes modules implemented by users that can be integrated into the library.
- **Plotting tools**. The tools for graphically representing the probabilistic model embedded by the EDA are included in this module. Fig. 2 shows an example of two different probabilistic models. Panel (a) represents a Gaussian BN, in which dependencies between variables are considered, while panel (b) represent a univariate model, in which no dependencies are considered.

Regarding the multivariate EDA implementations, some of the probabilistic models are interfaced to `PyBNesian` library [20], which uses `C++` to speed up the back-end computations. All the algebraic computations in `EDAspy` are computed using `numpy` library [21], employing `C` to speed up the back-end computations. Moreover, the parallelization of the optimizer is available by using `multiprocessing` library [22,23], and can be optionally activated in all the EDA implementations.

4. Related work

Although there are several libraries in which different evolutionary algorithms are available, to the best of our knowledge we have not found comparable published libraries with different EDA implementations in `python`. However, here we list some libraries in which some EDA implementations are available.

- `mateda` [24] is a `matlab` library which allows building multivariate EDAs based on undirected probabilistic models and Bayesian networks. The purpose of the library is different from `EDAspy`. It offers a framework to build a multivariate EDA algorithm by modules, in which different components can be integrated. `mateda` implements categorical and Gaussian Bayesian networks, multivariate Gaussian distributions, Markov networks and mixtures of Gaussian distributions as probabilistic models. However, semiparametric and KDE Bayesian networks are missed, and the implementations for univariate approaches are omitted. Moreover, the last released version of `mateda` was in 2020.



(a) Gaussian BN structure (b) Univariate Gaussian structure

Fig. 2. Probabilistic models graphical representations.

Table 1
Summary of functionalities implemented in each library. Note that X^x denotes that the implementation is expected to be released in the near future.

	`EDAspy`	`mateda`	`inspyred`	`LEAP`
Language	`python`	`matlab`	`python`	`python`
UMDA$_C$	X		X	
UMDA$_D$	X			
UMDA$_B$	X			
u_KEDA	X			
PBIL	X			X^x
BOA	X			X^x
EMNA	X	X		
EGNA	X	X		
SPEDA	X			
m_KEDA	X			
EBNA	X	X		
Custom	X	X		

- `inspyred` [25] is a `python` library which implements general evolutionary algorithms such as genetic algorithms, evolutionary strategies, differential evolution and multi-objective genetic algorithms, among others. UMDA$_C$ is the only EDA implemented.
- `LEAP` [26] is a `python` library built for evolutionary computation and incorporates useful visualization modules. Regarding EDAs, the population-based incremental learning algorithm (PBIL) [15] and Bayesian optimization algorithm (BOA) [18] are expected to be available in future releases.

Table 1 summarizes the main differences between the listed libraries. Regarding univariate approaches, `inspyred` implements UMDA$_C$ and `LEAP` plans to integrate PBIL approaches in the near future, compared to the five implemented variants in `EDAspy`. Regarding multivariate approaches, `LEAP` will incorporate BOA approach, which is also implemented in `EDAspy`. The most competitive library is `mateda`, which overlaps with some of the implemented multivariate approaches. It also allows for building a custom EDA version with some additional probabilistic models. However, `mateda` is implemented in `matlab` and seems to be no longer updated.

5. Performance analysis

In this section we compare the performance of different continuous domain optimizers implemented in EDAspy. For the evaluation three different cost functions (to be minimized) have been selected from the benchmark suite in `EDAspy`: CEC14_3, CEC14_4 and CEC14_8, where the former is unimodal and the rest are multimodal functions.

Section 4 reviewed some existing software for EDAs in different programming languages. In this section we also compare the result found by the UMDA$_C$ approach implemented in `inspyred`. Although `mateda` and `LEAP` were also reviewed, the former is implemented in a different programming language, and thus it is not fair to be compared in terms of CPU time, and the latter does not currently include any of the implemented approaches.

All the optimizers have been configured equally in order to perform a fair comparison. Hyper-parameters and a more extended tutorial can be found in the original documentation.[1]

Since a statistical study is out of the scope of the paper (see [17] for a more complete analysis), we show a runtime and final solutions analysis of the different variants for continuous optimization in `EDAspy`.

Fig. 3 shows the mean best cost found after 5 independent executions. It is generally observed how in the three functions the best approaches are SPEDA, m_KEDA and EGNA, which find the minimal costs in the benchmarks. Previous analyses have shown that m_KEDA, SPEDA and EGNA approaches are able to achieve statistically significant improvements in terms of quality of solutions [17]. In the case of the UMDA$_C$ implementation from `inspyred` library, a slightly worse result is found in all the three benchmarks compared to the implementation provided in `EDAspy`.

Fig. 4 shows the mean CPU times of all the tested algorithms after 5 independent executions. Note that all the tested approaches have been configured in the same environment, that is, the number of function evaluations and hardware. It is observed that generally the higher the complexity of the probabilistic model embedded, the longer the CPU time required. However, PBIL is one of the slowest approaches in the comparison for CEC14_3. In this case, the multivariate version of KEDA is the most expensive algorithm in terms of CPU time, followed by SPEDA and EGNA. In the case of the UMDA$_C$ implementation from `inspyred` library, our implementation seems to be more efficient implemented in terms of CPU time consumption, keeping a good performance in terms of results found (Fig. 3).

6. Illustrative examples

The following examples are available in the original documentation[1], where different EDAs are applied to different tasks:

- Using UMDA$_C$ for continuous optimization. UMDA$_C$ is tested on a IEEE CEC 2014 benchmark.
- Using SPEDA for continuous optimization. SPEDA is tested on a provided benchmark and several convergence plots are shown.
- Using EGNA for continuous optimization. SPEDA is tested on a provided benchmark and the plotting tools module is used to graphically show the probabilistic model embedded into the EDA approach.

[1] https://github.com/VicentePerezSoloviev/EDAspy/blob/master/notebooks/CPU%20time%20analysis.ipynb.

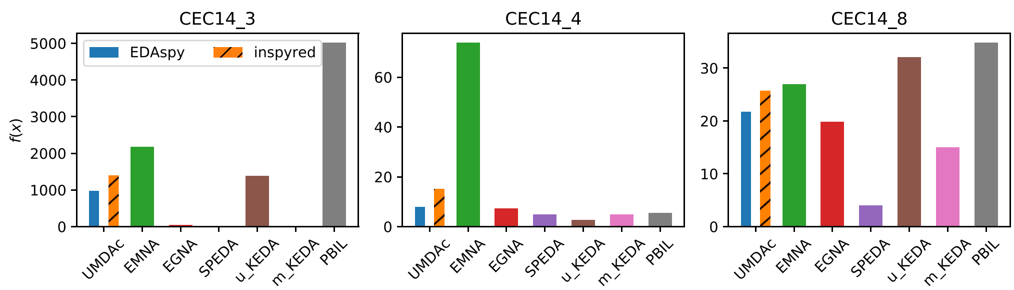


Fig. 3. Best cost found analysis of some EDA variants for continuous optimization. $UMDA_C$, EMNA, EGNA, SPEDA, univariate KEDA (u_KEDA), multivariate KEDA (m_KEDA) and PBIL are shown.

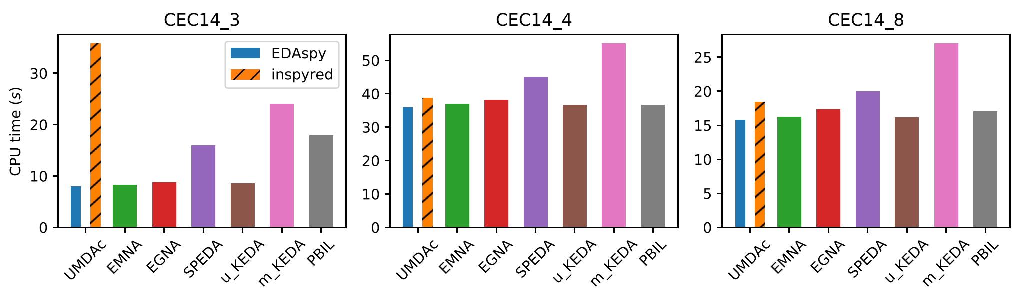


Fig. 4. CPU runtime analysis of some EDA variants for continuous optimization. $UMDA_C$, EMNA, EGNA, SPEDA, univariate KEDA (u_KEDA), multivariate KEDA (m_KEDA) and PBIL are shown.

Table A.2
Software metadata.

N	Software metadata description	Software metadata information
S1	Current software version	1.1.4
S2	Permanent link to executables of this version	https://github.com/VicentePerezSoloviev/EDAspy/releases/tag/1.1.3
S3	Legal software license	MIT License
S4	Computing platform/Operating system	Linux, OS X, Windows
S5	Installation requirements & dependencies	python 3.8–3.11, pybnesian, numpy, pandas, scikit_learn, scipy, pgmpy pyarrow, multiprocessing
S6	Link to user manual	https://edaspy.readthedocs.io/en/latest/
S7	Support email for questions	vicente.perez.soloviev@gmail.com

- Using EMNA for continuous optimization. EMNA is tested on a IEEE CEC 2014 benchmark.
- Using $UMDA_D$ for feature selection in a toy example. Given a dataset and a forecasting model, $UMDA_D$ is used to select the best subset of variables that optimizes the accuracy of the prediction.
- Categorical optimization using EBNA and $UMDA_D$. A categorical cost function is designed and optimized by EBNA and $UMDA_D$ approaches.
- Building my own EDA implementation. A tutorial on how to customize an EDA implementation is provided.
- CPU time analysis. All the continuous domain EDA variants are tested against the same IEEE CEC 2014 benchmark.

7. Conclusions

In this paper we present the first `python` library entirely dedicated to EDA implementations. `EDAspy` has been shown to be easy to use, and to integrate with custom implementations. Therefore, we hope that `EDAspy` can speed up the development of research on EDAs and their applications.

In addition to maintaining the code and solving bugs found by `EDAspy` users, future work would include adding more visualization tools for the optimization process and the implementation of other EDA variants.

CRediT authorship contribution statement

Vicente P. Soloviev: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Pedro Larrañaga:** Writing – review & editing. **Concha Bielza:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Innovation through the PID2022-139977NB-I00 and TED2021-131310B-I00 projects, and by the Autonomous Community of Madrid within the ELLIS Unit Madrid framework. Vicente P. Soloviev has been supported by the predoctoral grant FPI PRE2020-094828 from the Spanish Ministry of Science and Innovation.

Appendix. Required metadata

A.1. Current executable software version

See Table A.2.

Table A.3
Code metadata.

N	Software metadata description	Software metadata information
C1	Current code version	1.1.4
C2	Permanent link to code/repository used of this code version	https://github.com/VicentePerezSoloviev/EDAspy
C3	Legal software license	MIT License
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	python 3.8–3.11
C6	Compilation requirements, operating environments & dependencies	Compatible python pybnesian, numpy, pandas, scikit_learn, scipy, pgmpy, pyarrow, multiprocessing
C7	Link to developer documentation/manual	https://edaspy.readthedocs.io/en/latest/
C8	Support email for questions	vicente.perez.soloviev@gmail.com

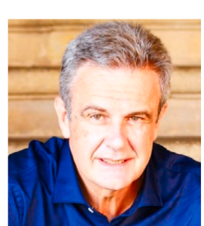
A.2. Current code version

See Table A.3.

References

[1] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2001.
[2] D. Dasgupta, Z. Michalewicz, Evolutionary Algorithms in Engineering Applications, Springer Science & Business Media, 2013.
[3] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J.L. Flores, J.A. Lozano, Y.V. de Peer, R. Blanco, V. Robles, C. Bielza, P. Larrañaga, A review of estimation of distribution algorithms in bioinformatics, BioData Min. 1 (2008) 1–12.
[4] J. Ceberio, A. Mendiburu, J.A. Lozano, A roadmap for solving optimization problems with estimation of distribution algorithms, Nat. Comput. (2022) 1–15.
[5] V.P. Soloviev, P. Larrañaga, C. Bielza, Estimation of distribution algorithms using Gaussian Bayesian networks to solve industrial optimization problems constrained by environment variables, J. Comb. Optim. 44 (2) (2022) 1077–1098.
[6] V.P. Soloviev, P. Larrañaga, C. Bielza, Quantum parametric circuit optimization with estimation of distribution algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2022, pp. 2247–2250.
[7] P. Larrañaga, C. Bielza, Estimation of distribution algorithms in machine learning: A survey, IEEE Trans. Evol. Comput. (2023).
[8] J.C. Uribe, B. Doerr, C. Witt, V.P. Soloviev, Estimation-of-Distribution Algorithms: Theory and Applications (Dagstuhl Seminar 22182), Dagstuhl Reports, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
[9] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, The MIT Press, 2009.
[10] M.S. Krejca, C. Witt, Lower bounds on the run time of the univariate marginal distribution algorithm on OneMax, in: Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, 2017, pp. 65–79.
[11] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Technical Report 635, Nanyang Technological University, Singapore, 2013, pp. 490–523.
[12] H. Mühlenbein, G. Paass, From recombination of genes to the estimation of distributions I. Binary parameters, in: International Conference on Parallel Problem Solving from Nature, Springer, 1996, pp. 178–187.
[13] H. Mühlenbein, J. Bendisch, H.-M. Voigt, From recombination of genes to the estimation of distributions II. Continuous parameters, in: International Conference on Parallel Problem Solving from Nature, Springer, 1996, pp. 188–197.
[14] N. Luo, F. Qian, Evolutionary algorithm using kernel density estimation model in continuous domain, in: 2009 7th Asian Control Conference, IEEE, 2009, pp. 1526–1531.
[15] S. Baluja, Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, School of Computer Science, Carnegie Mellon University Pittsburgh, PA, 1994.
[16] P. Larrañaga, R. Etxeberria, J. A. Lozano, J. M. Peña, Optimization in continuous domains by learning and simulation of Gaussian networks, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, 2000.
[17] V.P. Soloviev, C. Bielza, P. Larrañaga, Semiparametric estimation of distribution algorithms for continuous optimization, IEEE Trans. Evol. Comput. (2023).
[18] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, BOA: The Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference, Vol. 1, 1999, pp. 525–532.
[19] M.D. McKay, R.J. Beckman, W.J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics (2000) 55–61.
[20] D. Atienza, C. Bielza, P. Larrañaga, PyBNesian: An extensible python package for Bayesian networks, Neurocomputing 504 (2022) 204–209.
[21] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362.
[22] M.M. McKerns, L. Strand, T. Sullivan, A. Fang, M.A. Aivazis, Building a framework for predictive science, 2012, arXiv preprint arXiv:1202.1056.
[23] M. McKerns, M. Aivazis, Pathos: A framework for heterogeneous computing, 2010, See http://trac.mystic.cacr.caltech.edu/project/pathos.
[24] R. Santana, C. Bielza, P. Larrañaga, J.A. Lozano, C. Echegoyen, A. Mendiburu, R. Armananzas, S. Shakya, Mateda-2.0: A MATLAB package for the implementation and analysis of estimation of distribution algorithms, J. Stat. Softw. 35 (2010) 1–30.
[25] A. Tonda, Inspyred: Bio-inspired algorithms in Python, Genet. Program. Evol. Mach. 21 (1–2) (2020) 269–272.
[26] M.A. Coletti, E.O. Scott, J.K. Bassett, Library for evolutionary algorithms in Python (LEAP), in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, GECCO '20, Association for Computing Machinery, 2020, pp. 1571–1579.

Vicente P. Soloviev received the M.Sc. degree in Artificial Intelligence from Universidad Politécnica de Madrid, in 2020 and he is currently a Ph.D. student at Universidad Politécnica de Madrid in the Computational Intelligence Group. He teaches some subjects related to Artificial Intelligence for the B.Sc. in Computer Science at Universidad Politécnica de Madrid. His research interests include the areas of probabilistic graphical models, metaheuristics for optimization, quantum machine learning, quantum heuristics, and real applications such as industry 4.0. Vicente holds a pre-doctoral FPI contract awarded by the Spanish Ministry of Science and Innovation since 2021.



Pedro Larrañaga received the M.Sc. degree in Mathematics (Statistics) from the University of Valladolid and the Ph.D. degree in Computer Science from the University of the Basque Country (excellence award). He has been a Full Professor in Computer Science and Artificial Intelligence with the Universidad Politécnica de Madrid (UPM), since 2007. Before moving to UPM, his academic career developed at the University of the Basque Country (UPV-EHU) through several faculty ranks: Assistant Professor, from 1985 to 1998, Associate Professor, from 1998 to 2004, and a Full Professor, from 2004 to 2007. He has published over 200 papers in high-impact factor journals. He has supervised over 35 Ph.D. theses. His research interests include the areas of probabilistic graphical models, metaheuristics for optimization, data mining, classification models, and real applications, such as biomedicine, bioinformatics, neuroscience, industry 4.0, and sports. He is Fellow of the European Association for Artificial Intelligence, since 2012, and a Fellow of the Academia Europaea, since 2018. He has received the 2013 Spanish National Prize in computer science and the prize of the Spanish Association for Artificial Intelligence in 2018 and the 2020 Machine Learning Award from Amity University (India).

Concha Bielza received her M.S. degree in Mathematics from Universidad Complutense de Madrid, in 1989, and her Ph.D. degree in Computer Science from Universidad Politécnica de Madrid (UPM) in 1996 (extraordinary doctorate award). She is currently (since 2010) a Full Professor of Statistics and Operations Research with the Departamento de Inteligencia Artificial, UPM. Her research interests are primarily in the areas of probabilistic graphical models, decision analysis, metaheuristics for optimization, classification models, and real applications, such as biomedicine, bioinformatics, neuroscience and industry 4.0. She has published more than 150 papers in high impact factor journals and has supervised 22 Ph.D. theses. She was awarded the 2014 UPM Research Prize and the 2020 Machine Learning Award from Amity University (India).