**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.

**Algorithm 1** Estimation of distribution algorithm

<table>
<thead>
<tr>
<th>Input: Population size ( N ), selection ratio ( \alpha ), cost function ( g )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Best individual ( x' ) and cost found ( g(x') )</td>
</tr>
<tr>
<td>1: ( G_0 \leftarrow N ) individuals randomly sampled</td>
</tr>
<tr>
<td>2: for ( t = 1, 2, \ldots ) until stopping criterion is met do</td>
</tr>
<tr>
<td>3: Evaluate ( G_{t-1} ) according to ( g(\cdot) )</td>
</tr>
<tr>
<td>4: ( G^S_{t-1} \leftarrow ) Select top ( \alpha N ) individuals from ( G_{t-1} )</td>
</tr>
<tr>
<td>5: ( f_{t-1}(\cdot) \leftarrow ) Learn a probabilistic model from ( G^S_{t-1} )</td>
</tr>
<tr>
<td>6: ( G_t \leftarrow N ) individuals from ( f_{t-1}(\cdot) )</td>
</tr>
<tr>
<td>7: end for</td>
</tr>
</tbody>
</table>

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.

**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 \([3\text{–}6]\). 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 \( \alpha \) of the best individuals, i.e., the top \( \alpha N \) solutions (line 5). Finally, a new population is sampled (line 6). These four steps are iteratively 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.

**Acknowledgements**

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
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.

EDApy 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_C), and (iii) continuous optimization [13] (UMDA_D); (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 EDApy 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 EDApy. 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.
implementation is expected to be released in the near future. However, approaches. It also allows for building a custom EDA version with some additional probabilistic models. 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 m\_KEDA, which overlaps with some of the implemented multivariate approaches. It also allows for building a custom EDA version with some additional probabilistic models. However, m\_KEDA 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 m\_KEDA 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. 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.

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.
Using EMNA for continuous optimization. EMNA is tested on an 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.
A.2. Current code version

See Table A.3.

### References


Table A.3

<table>
<thead>
<tr>
<th>C</th>
<th>Code metadata description</th>
<th>Software metadata information</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Current code version</td>
<td>1.1.4 <a href="https://github.com/VicentePerezSoloviev/EDAspy">https://github.com/VicentePerezSoloviev/EDAspy</a></td>
</tr>
<tr>
<td>C2</td>
<td>Permanent link to code/repository used of this code version</td>
<td>git</td>
</tr>
<tr>
<td>C3</td>
<td>Legal software license</td>
<td>python 3.8-3.11</td>
</tr>
<tr>
<td>C4</td>
<td>Code versioning system used</td>
<td>Compatible python pybnesian, numpy, pandas, scikit,learn, scipy, pgmpy, pyarow, multiprocessing</td>
</tr>
<tr>
<td>C5</td>
<td>Software code languages, tools, and services used</td>
<td><a href="https://edaspy.readthedocs.io/en/latest/">https://edaspy.readthedocs.io/en/latest/</a> <a href="mailto:vicente.perez.soloviev@gmail.com">vicente.perez.soloviev@gmail.com</a></td>
</tr>
<tr>
<td>C6</td>
<td>Compilation requirements, operating environments &amp; dependencies</td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>Link to developer documentation/manual</td>
<td></td>
</tr>
<tr>
<td>C8</td>
<td>Support email for questions</td>
<td></td>
</tr>
</tbody>
</table>

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).