SoftwareX 30 (2025) 102097

Contents lists available at ScienceDirect

SoftwareX

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

Software update

Version [2.0] — [pyMCMA: Uniformly distributed Pareto-front representation]

Marek Makowski [a,b,*], Janusz Granat [c], Andrii Shekhovtsov [c], Zbigniew Nahorski [a], Jinyang Zhao [d,e]

[a] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
[b] International Institute for Applied Systems Analysis, Laxenburg, Austria
[c] Warsaw University of Technology, Warsaw, Poland
[d] School of Business, East China University of Science and Technology, Shanghai, China
[e] PetroChina Planning and Engineering Institute, China National Petroleum Corporation, Beijing, China

ARTICLE INFO

Keywords:
Pareto-front representation
Multiple-criteria model analysis
Pareto-front visualization
Clustering
Pyomo modeling language
Structured modeling

ABSTRACT

pyMCMA is the Python implementation of a novel method for autonomous computation of the Pareto-front representation composed of efficient solutions distributed uniformly in terms of the distances between neighbor Pareto solutions. pyMCMA supports scientific, i.e., objective, model analysis by providing preference-free Pareto front representation.

The update provides new functionalities and enhancements. The former include clustering of the Pareto-front solutions. The enhancements include internal software improvements, optional customization of some parameters, as well as a new functionalities that might be used by advanced users.

Code metadata

Current code version	2.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00380
Permanent link to Reproducible Capsule	Not applicable
Legal Code License	GNU General Public License (GPL)
Code versioning system used	git
Software code languages, tools, and services used	Python 3.11, conda. The required Python packages are specified in the pyMCMA configuration on the conda-forge.
Compilation requirements, operating environments & dependencies	macOS or Linux or MS-Windows. All needed packages are installed at the pyMCMA installation
If available Link to developer documentation/manual	https://pymcma.readthedocs.io
Support email for questions	marek@iiasa.ac.at

1. Motivation for the update

The original publication on pyMCMA [1] describes the Python implementation of a novel method for autonomous computation of the Pareto-front representation composed of efficient solutions of a mathematical programming model, furthermore referred to as a core model. The representation is distributed uniformly in terms of the distances between neighbor Pareto solutions included in the representation.

The current pyMCMA implementation seamlessly integrates core models developed in the Pyomo [2] modeling environment. It was tested with LP (Linear Programming) and MILP (Mixed-Integer Linear Programming) models. The implemented methodology should also work with NLP (Non-Linear Programming) models, but the seamless integration of NLP models to pyMCMA requires support for solver configuration, which has not yet been explored.

The use of pyMCMA is very simple as it requires the specification in the configuration file of only two items: (1) location of the core model,

DOI of original article: https://doi.org/10.1016/j.softx.2024.101801.
* Corresponding author at: International Institute for Applied Systems Analysis, Laxenburg, Austria.
E-mail address: marek@iiasa.ac.at (Marek Makowski).

https://doi.org/10.1016/j.softx.2025.102097
Received 22 July 2024; Received in revised form 19 December 2024; Accepted 11 February 2025
Available online 10 March 2025

and (2) definition of criteria by selecting at least two corresponding variables of the core model and declaring whether a criterion should be minimized or maximized. There is no upper limit on the number of criteria. Moreover, there is no requirement on criteria independency.

The novel method developed for pyMCMA has no restrictions whatsoever on the properties of the Pareto set. In particular, the Pareto set can be discrete and/or nonconvex and/or disconnected.

Experience with pyMCMA application to diverse problems (one of them is presented in Section 3) and the corresponding discussions, as well as earlier applications of the interactive Multiple-Criteria Model Analysis (MCMA) [3], triggered the demand to extend the pyMCMA functionality. Improved support for the analysis of large Pareto front representations is the main enhancement of the software update. Therefore, it is discussed here in detail. Other extensions and improvements are summarized in Section 2.

The implemented clustering is especially useful for analysis of the Pareto front if the requested maximum distance between the neighbor solution is rather small; in such cases, the representation of Pareto front for problems with several criteria is typically composed of hundreds of thousands of solutions. This in turn causes difficulties in effective analysis, an issue recognized long ago; see, e.g., [4].

The latter issue can be decomposed into two elements. The first is reducing the representation by pruning from the detailed representation solutions that are close (in terms of the criteria values) to other solutions; see, e.g. [5]. Such an approach is not needed in pyMCMA because it prunes close neighbor solutions on the fly. However, the second element, namely support of analysis of large sets of distinct Pareto solutions, remains the widely researched challenge.

Clustering is the typical approach to categorizing elements of a large set of solutions (generally data points) into clusters, where each cluster contains a subset of the set and is characterized by one solution; see, e.g., [6–8], and the references therein. There are several clustering techniques, including partitioning, hierarchical, grid-based, or density-based methods. The K-Means algorithm is a well-known partitioning method. The algorithm groups the data into a given number of clusters based on their proximity to each other, and defines the center of each cluster. The centers usually differ from the data points.

In the Pareto-front representation, the clustered data is defined by the Pareto solutions. Therefore, the K-Means algorithm is not suitable for such cases, because the cluster centers would not correspond to any Pareto solution. Hence, we applied the K-Medoids method [9], which defines the cluster centers at one of the Pareto solutions.

Although the addition of clustering was the main motivation for the update, we implemented several other pyMCMA enhancements outlined in Section 2. Most of them aim to support advanced users of MCMA (Multiple-Criteria Model Analysis) in optional customization of pyMCMA to specific needs. Others improve the handling of diverse problems not recognized by most users. The latter include non-unique solutions of selfish optimization for a criterion, which required enhancement of computation of the Pareto-set corners; see Sections 2 and 3 for more information and the corresponding discussion, respectively.

The remaining part of the paper discusses in Section 2 details of the software update, follows with an illustrative use case in Section 3, and concludes with Section 4.

2. Description of the software update

The update of pyMCMA improves the support for scientific, i.e., objective, multiple-criteria model analysis by providing new functionalities of the preference-free Pareto front representation provided by the original pyMCMA version.

From the user point of view, the most important part of the software update is the implementation of clustering of the Pareto front representation. Clustering is optional; it can be activated by an entry in the pyMCMA configuration file described in the online documentation. The number of clusters best supporting an analysis depends on the analyzed problem characteristics; therefore, it cannot be rationally predefined. We recommend starting with a small number of clusters and increasing the number as long as the results provide the desired insights.

We recall that each cluster center corresponds to a medoid, i.e., to one of the Pareto solutions. The medoid is equal to such a Pareto point (belonging to the cluster) whose sum of dissimilarities to all points in the cluster is minimal. Therefore, the medoid is the Pareto point most centrally located in the cluster. Hence, the corresponding cluster center can be interpreted as a cluster representation.

Diverse plots illustrating clustering results are discussed in Section 3. Additionally, pyMCMA provides for each cluster information about: ranges of criteria values of Pareto solutions included in the cluster, the center of the cluster, its radius and the number of included solutions.

The other improvements of the updated pyMCMA include:

1. Generation of the Pareto-set corners (aka selfish solutions) and the approximation of the Nadir point. This change is transparent for users but it is important for a correct Pareto-set representation in some problems with more than two criteria, when a criterion has a non-unique selfish optimization solution. Information about the Pareto-set corners is now available in the pyMCMA output. An example of such a situation is discussed in Section 3. A discussion of the approach to handling such problems is beyond the scope of this paper; it will be included in the planned paper on the underlying methodology.
2. Optional specification of the Aspiration/Reservation (A/R) criteria values. This optional feature might be useful for analysis of problems that require rather long optimization time; in such cases, it might be irrational to generate the whole Pareto front representation with fine resolution. Instead, for initial analysis, the Pareto front representation with a coarse resolution may provide a good basis for selecting the so-called Region-of-Interest (RoI), and augment the autonomously generated coarse representation by a detailed representation of only the selected RoIs. To support the latter, we implemented optional specifications of the A/R-based preferences. The A/R-based specification of the user preferences is widely used in the interactive MCMA; see, e.g. [3]. As explained in [1], this method is also used by pyMCMA, where the A/R values, for each iteration, are generated autonomously. Now, the pyMCMA users can optionally specify a set of the A/R values for generating, in selected RoIs, additional (to those computed by pyMCMA) Pareto-front solutions that fit the specific user preferences.
3. Use of diverse optimization solvers. In most cases, the solvers distributed with Pyomo [2] perform sufficiently well. However, users may want to use alternative solvers for computationally demanding optimization tasks. Such solvers can now be declared in the configuration file.
4. Optional modifications of the pyMCMA parameters. This functionality can be controlled in the configuration file and allows advanced users to refine their analysis.

    - The resolution of the representation. The default value is equal to 5 (in the criteria achievement scale [0, 100]) and is suitable for most cases. However, for time-demanding optimization tasks, the values of 10 (or even 20) provide good enough insights.
      To illustrate the coarse representation, we use in the example discussed in Section 3 the resolution equal to 20.
    - Maximum number of iterations. The default value of 1000 is sufficiently large for most problems to obtain the required resolution of the representation. However, computing the representation for problems with many criteria and/or requested fine resolution for some shapes of the Pareto-front may require more iterations.

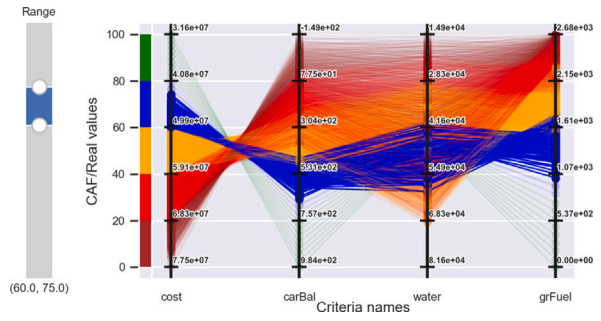

Fig. 1. Parallel plot of the Pareto-front (discussed in [1]).

- Number of clusters. The default value of 0 suppresses clustering.

5. The program flow control is reorganized in the new Python class `WrkFlow`. This change is transparent for most users, but might be relevant for those who want to modify the code.
6. The specification of the packages' version in the configuration of the `conda-forge` distribution of `pyMCMA` is improved. This was triggered by the report of a user who faced the problem with Python-packages version incompatibility caused by running the recommended `conda update --all` command. We will monitor the situation and make periodic changes in this configuration, as well as in the online documentation.

The above summarized new `pyMCMA` functionalities are optional. Any combination of them can be used by selecting the corresponding options in the `pyMCMA` configuration file. The `pyMCMA` configuration is documented in https://pymcma.readthedocs.io. Moreover, the template of configuration file is self-documented and is available after the `pyMCMA` installation.

The `pyMCMA` run output consists of thousands of lines; its content depends on the selected options, therefore, it cannot be discussed in a short paper. We point out that the output can be optionally redirected to a file when this is suitable for analysis.

3. Illustrative use case

The functionality of the key `pyMCMA` update element, namely the clustering, is illustrated by the model developed for the analysis of the technology pathways of China's liquid fuel production [10–12]. The same model was also used as an illustrative example in [1]. Therefore, by comparing the original and this paper, it is easy to recognize the advantages of the implemented clustering. We recall that the case considers four criteria:

- Cost: total cost (minimize);
- Water use (minimize);
- GreenFue (Green fuel;[1] produced by two, out of four considered, technologies) (maximize);
- Carbon balance (difference between carbon emission and capture; the latter is done by only one considered technology) (minimize).

Fig. 1 illustrates the trade-offs between the criteria achievements in many solutions. Although interactive brushing helps to recognize the general trade-offs between the costs and the other three criteria, it does not help in classifying subsets of solutions having diverse combinations of trade-offs. Such support is provided by clustering, which we illustrate in Fig. 2 by the 3D plots of the first three criteria in the above list.

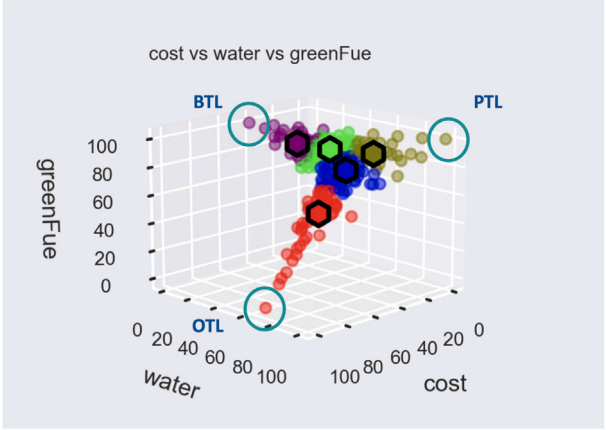

Fig. 2. Five clusters of the coarse Pareto-front representation. Criteria performances are in the achievement scale, where 0 and 100 correspond to the worst and best criterion values, respectively. The same scale is used in two Figures below.

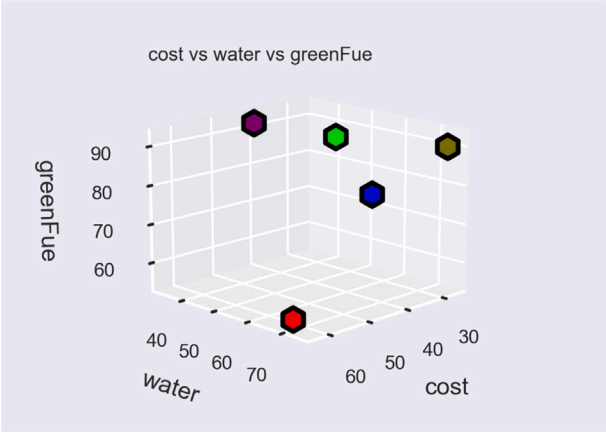

Fig. 3. Centers (medoids) of the clusters.

For the remaining part of the discussion, we use the results obtained for 5 clusters, and the representation resolution equal to 20. The latter choice was made to illustrate that even such a coarse Pareto front approximation provides a useful initial view on the basic Pareto front characteristics. We also note that in the Figures the criteria values are shown in the criteria achievement scale, where 0 and 100 correspond to the worst and best criterion values, respectively.[2]

Fig. 2 shows the 3D plot of the Pareto solutions. It illustrates two issues: (1) the non-uniqueness of selfish solutions, and (2) results of clustering. We comment on each of these issues.

The selfish solutions are provided by optimization of each criterion separately; they are used for definition of the Pareto front extreme points (aka corners). A traditional approach defines only one selfish solution for each criterion, i.e., only one corresponding corner. However, non-unique selfish solutions, if correctly treated, define (for more than two criteria) all relevant corners, which is necessary for proper specification of extreme points of Pareto set. Fig. 2 shows two selfish solutions for criterion `greenFue`. Both solutions have the same value of this criterion. However, they have substantially different values of the remaining two criteria:

- The extreme violet-marked solution is the worst in using water but is moderately expensive;

[1] Green fuel is a liquid fuel produced using only renewable sources, which significantly reduces greenhouse gas emissions compared to fossil fuels.

[2] Note that the best achievement for minimized criteria (e.g., water) is obtained for the minimum (within the Pareto set) criterion value.

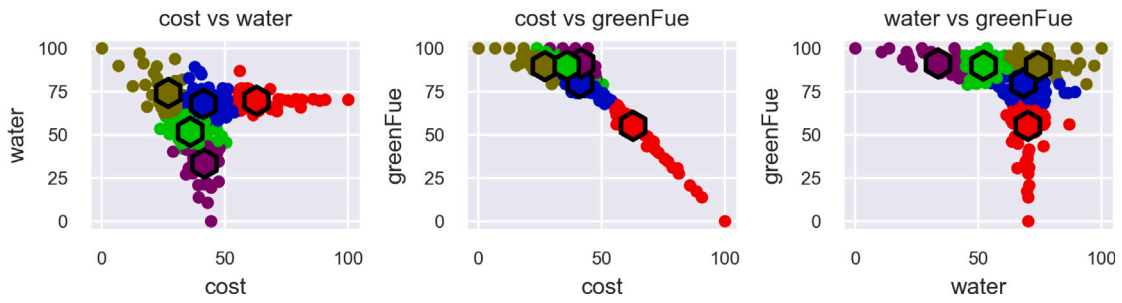

Fig. 4. Clusters' 2D projections for pairs of criteria.

- The extreme khaki-marked solution is the most water-efficient but it is the most expensive.

As mentioned in Section 1, an appropriate definition of the corners is essential for a proper Pareto front definition.

Fig. 2 shows the 3D plot of the Pareto solutions. The colors mark the memberships of the solutions in one of the clusters. The clusters' centers (medoids) are marked with hexagons. Additionally, Fig. 2 shows extreme solutions of the Pareto front representation. Labels BTL, PTL and OTL[3] correspond to the names of the technologies that are the only ones in each of these solutions. Comparing the two selfish solutions providing most of the Green Fuel, we see that the PTL is more expensive and most water efficient, while the BTL is cheaper but uses more water than the PTL.

Fig. 3 shows only the centers of the clusters. Note the scales in the latter Figure. Namely, the ranges of the scale are reduced to enhance the visibility by removing the empty part of this Figure. Finally, we mention that the 3D Figures are interactive during the program execution; therefore, it is also easy to change the angle of view on every selected Figure.

The interpretation of the clusters is quite straightforward. The red cluster contains rather cheap solutions that use medium amounts of water and produce a rather small fraction of the green fuel. The khaki, green, and violet clusters are composed of solutions that produce a rather large fraction of the green fuel. They differ mainly by the cost and the amount of water used; namely, the khaki cluster contains the most expensive (but relatively water-efficient) solutions, while the violet cluster is composed of cheaper but water-inefficient solutions. The green cluster gathers solutions with cost and water use between the corresponding values in the violet and khaki clusters. Finally, the blue cluster is composed of solutions that produce more green-fuel than those in the red cluster but less than the solutions in the other three clusters.

Fig. 4 shows the 2D projections of the clusters for pairs of criteria. The color markers of the solutions have the same meaning as in the 3D plots. The 2D projections are useful for pairwise analysis of the criteria trade-offs. For example, the cost-water projection shows that the rather cheap solutions (red cluster) are also relatively water-efficient; the violet solutions have middle costs, while the khaki solutions are more expensive than most of the other solutions. Similar conclusions can be drawn from the analysis of the two other criteria pairs.

Finally, we note that the run-output contains diverse numerical characteristics of the solutions that can be used for precise analysis. Moreover, the solutions are stored and, therefore, available for problem-specific analysis. The stored information is also useful for analysis in the core-model variables space. For example, one can analyze the underlying portfolios of technologies corresponding to each Pareto solution. Such an analysis shows that each corner solution is obtained by a single technology, while other solutions involve diverse levels of investment in building capacities of the corresponding technologies in different planning periods. More information on these issues is available in [1] and in the documentation available at https://pymcma.readthedocs.io.

[3] The first letter indicates the main input of the corresponding technology producing liquid fuel: B: biomass, P: power, O: crude oil.

4. Conclusions

The presented software update significantly extends the `pyMCMA` functionality and thus improves support for scientific, i.e., objective, multiple-criteria model analysis. The preference-free Pareto-front representation equitably treats all Pareto-efficient solutions. The main enhancement of the update consists of optional clustering, which supports an in-depth analysis of diverse trade-offs between criteria achievements represented by clusters composed of subsets of the Pareto solutions. In addition, the centers of the clusters help define the characteristics of each cluster. Other enhancements summarized in Section 2 increase the robustness and flexibility of `pyMCMA`, the latter helping to tailor the `pyMCMA` functionality to diverse specific needs.

The authors plan to periodically implement further `pyMCMA` improvements and release the corresponding versions in the conda-forge repository. Information on the forthcoming `pyMCMA` versions will be available in the corresponding versions of the online documentation.

CRediT authorship contribution statement

Marek Makowski: Writing – original draft, Software, Methodology, Conceptualization. **Janusz Granat:** Writing – review & editing, Supervision, Software, Project administration, Funding acquisition, Conceptualization. **Andrii Shekhovtsov:** Writing – review & editing, Visualization, Validation, Software. **Zbigniew Nahorski:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization. **Jinyang Zhao:** Writing – review & editing, Validation, Data curation.

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.

Acknowledgments

The authors gratefully acknowledge the financial support of the National Science Centre, Poland [2018/30/Q/HS4/00764], and of the National Natural Science Foundation of China (71961137012, 7214006). The authors thank the journal editor and the three anonymous reviewers of the original `pyMCMA` article published in SoftwareX journal, as well as the three anonymous reviewers of the first version of this paper, for their comments and recommendations, which substantially helped to improve the original submission, as well as this update.

References

[1] Makowski M, Granat J, Shekhovtsov A, Nahorski Z, Zhao J. pyMCMA: Uniformly distributed Pareto-front representation. SoftwareX 2024;27:101801. http://dx.doi.org/10.1016/j.softx.2024.101801.

[2] Bynum ML, Hackebeil GA, Hart WE, Laird CD, Nicholson BL, Siirola JD, et al. Pyomo–optimization modeling in python. third ed.. vol. 67, Springer Science & Business Media; 2021.

[3] Granat J, Makowski M. Interactive specification and analysis of aspiration-based preferences. European J Oper Res 2000;122(2):469–85.

[4] Morse J. Reducing the size of the nondominated set: Pruning by clustering. Comput Oper Res 1980;7(1):55–66. http://dx.doi.org/10.1016/0305-0548(80)90014-3.

[5] Petchrompo S, Coit DW, Brintrup A, Wannakrairot A, Parlikad AK. A review of Pareto pruning methods for multi-objective optimization. Comput Ind Eng 2022;167. http://dx.doi.org/10.1016/j.cie.2022.108022.

[6] Brusco MJ. Partitioning methods for pruning the Pareto set with application to multiobjective allocation of a cross-trained workforce. Comput Ind Eng 2017;111:29–38. http://dx.doi.org/10.1016/j.cie.2017.06.035.

[7] Aguirre O, Taboada H. A clustering method based on dynamic self organizing trees for post-Pareto optimality analysis. Procedia Comput Sci 2011;6:195–200. http://dx.doi.org/10.1016/j.procs.2011.08.037, complex adaptive sysytems.

[8] Zio E, Bazzo R. A clustering procedure for reducing the number of representative solutions in the Pareto front of multiobjective optimization problems. European J Oper Res 2011;210(3):624–34. http://dx.doi.org/10.1016/j.ejor.2010.10.021.

[9] Schubert E, Rousseeuw PJ. Faster k-medoids clustering: Improving the pam, clara, and clarans algorithms. In: Amato G, Gennaro C, Oria V, Radovanović M, editors. Similarity search and applications. Cham: Springer International Publishing; 2019, p. 171–87.

[10] Zhao J, Yu Y, Ren H, Makowski M, Granat J, Nahorski Z, et al. How the power-to-liquid technology can contribute to reaching carbon neutrality of the China's transportation sector? Energy 2022;261:125058. http://dx.doi.org/10.1016/j.energy.2022.125058, published online: August 22 2022.

[11] Ding B, Makowski M, Nahorski Z, Ren H, Ma T. Optimizing the technology pathway of China's liquid fuel production considering uncertain oil prices: A robust programming model. Energy Econ 2022;115:106371. http://dx.doi.org/10.1016/j.eneco.2022.106371, published online: October 21 2022.

[12] Ding B, Makowski M, Zhao J, Ren H, Zakeri B, Ma T. Analysis of technology pathway of China's liquid fuel production with consideration of energy supply security and carbon price. J Manag Sci Eng 2023;8:1–14. http://dx.doi.org/10.1016/j.jmse.2022.07.002, published online: August 7 2022.