Pareto Explorer Framework

GUI-Documentation

Oliver Cuate[*]
CINVESTAV

Sebastian Peitz[†]
University of Paderborn

Abstract

Here, we present the documentation for the Graphical User Interface (GUI) of the Pareto Explorer (PE), a global/local exploration tool for the treatment of many objective optimization problems (MaOPs). This GUI focuses in the second step of PE, that is, the local search along the Pareto set/front of the given MaOP is performed into user specified directions. For this, we propose several continuation-like procedures that can incorporate preferences defined in decision, objective, or in weight space.

Contents

[*] ocuate@computacion.cs.cinvestav.mx
[†] speitz@math.upb.de

1 Introduction

In many applications the problem arises that several objectives have to be optimized concurrently leading to *multi-objective optimization problems* (MOPs). Due to the increasing complexity of practical problems, decision making processes are getting more and more sophisticated. Motivated by the advances in the design of algorithms for the numerical treatment of MOPs with few objectives and their huge success in applications, there is a recent trend to include more objectives into the optimization process. There is, however, one important characteristic of MOPs that has to be considered in their treatment, namely that the solution set (the Pareto set or its image, the Pareto front) typically form $(k-1)$-dimensional objects, where k is the number of objectives. This fact has a crucial impact on the approximation qualities of the Pareto sets and fronts with respect to k. Due to this reason, MOPs with more than four objectives are often termed *many objective problems* (MaOPs).

Continuous MOPs, as we consider in this work, can be stated as

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \; & F(x) \\ \text{s.t. } & g(x) \leq 0 \\ & h(x) = 0, \end{aligned} \tag{1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^k$, $F(x) = (f_1(x), \dots, f_k(x))^T$ is defined by the objective functions f_i, $i = 1, \dots, k$, and $g(x) = (g_1(x), \dots, g_m(x))^T$ and $h(x) = (h_1(x), \dots, h_p(x))^T$ are the inequality and equality constraints, respectively. $Q := \{x \in \mathbb{R}^n \; : \; g(x) \leq 0 \text{ and } h(x) = 0\}$ is called the domain of F.

Pareto Explorer (PE) is a global/local approach search for the effective treatment of MaOPs. The PE method consists of two main stages: first, one Pareto optimal solution x_0 is computed or selected out of a set of possible solutions for the given problem; and secondly, the Pareto landscape is explored around x_0, where a steering is performed according to the DMs' preferences. In general, the first part of the PE can be accomplished by any existing solver for M(a)OPs. In this work, we present the documentation for the second second stage. Here the PE will play its main role by restricting the search to a movement directed by the user's preferences. These preferences can be expressed in terms of directions in either decision or objective space, as well as in the space of the weight vectors. The key for these methods will be the work presented in [MS18] since (i) the proposed continuation method Pareto Tracer (PT) inherently has steering features that can be exploited for the current context, and since (ii) the results of [MS18] allow to explicitly compute the tangent spaces of both Pareto set and front at every regular solution. Though it is already known since the seminal work of Hillermeier ([Hil01]) that the Pareto sets and fronts locally form $(k-1)$-manifolds under certain assumptions, these are embedded in the $(n+k)$-dimensional space composed of the decision space and the associated weight space derived from the KKT equations. Much more detailed knowledge about the related tangent space has become available via a separation of both spaces as done in [MS18].

2 Pareto Explorer Method

The PE approach is mainly focused on the computation of a single trajectory along the landscape defined by the Pareto set/front. PE can hence be seen as a compilation of methods to follow the path given by the DM's needs, ultimately leading him/her to discover what he/she is looking for in a timely manner. The PE is thought to evolve as an interactive tool, where it can receive feedback from the user at any stage of the exploration. The PT, on the other hand, is designed to receive a fixed initial set of settings and run until completion, i.e., ideally until an approximation of the entire solution set is generated. Both techniques PT and PE complement each other and shall optimize their implementations based on their respective scopes.

2.1 Steering in Objective Space

The first proposed approach is to perform the steering in objective space, i.e., PE considers the case where the DM desires to change the objective values w.r.t. $F(x_0)$. That is, a direction $d_y \in \mathbb{R}^k$ is given with the aim to guide the search toward certain preferences that are only known in objective space. However, as the Pareto front is not known, it is of course unclear whether a movement in d_y can be actually performed. Then, since the linearized front at $F(x_0)$ can be computed, and since the underlying idea is to steer the search along the set of optimal solutions, it makes sense to perform a 'best fit' movement.

2.2 Steering in Decision Space

Similar to the steering in objective space, PE can perform a best fit movement along the Pareto set for a given preference direction in decision space. Analogously to the best fit movement in objective space, for a given point $x_i \in \mathcal{M}$ and a given a direction $d_x \in \mathbb{R}^n$ in *decision space*, one can project d_x onto the linearized Pareto set at x_i. This vector ν_i can now be used as predictor in a PC step of the PT.

2.3 Steering in Weight Space

The last movement we present here is the steering in weight space where the DM might be interested to gradually change the importance of the objectives. The key for this steering is the observation that a vector $\mu \in \mathbb{R}^k$ with $\sum_{i=0}^k \mu_i = 0$. Another way to see this is that for every two convex weights $\alpha^{(1)}, \alpha^{(2)} \in \mathbb{R}^k$ the difference vector $\Delta\alpha := \alpha^{(1)} - \alpha^{(2)}$ is of the above form. For $k = 2$ objectives, there are only two choices for μ after normalization: $\mu = (1, -1)^T$ and $\mu = (-1, 1)^T$. The first one means that the importance of f_1 should be increased and thus its values be decreased for the sacrifice of f_2. Hence, $\mu = (1, -1)^T$ should result in a movement left up the Pareto front from $F(x_0)$ while $\mu = (-1, 1)^T$ should result in a movement right down.

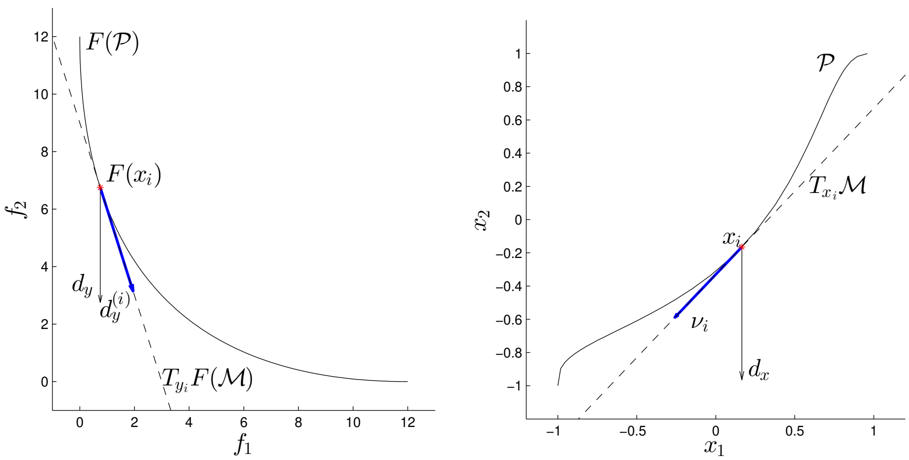

Figure 1: Left: Best fit direction $d_y^{(i)}$ for a given direction d_y in objective space. **Right**: Best fit direction in decision space for a given direction d_x in decision space.

3 The GUI

Here we describe the components of the GUI that easily allows to the user defining the preferences for the second stage of the PE. We assume that we have an initial optimal solution.

3.1 Main Window

The main window of the *Pareto Explorer* (cf. fig 2) consists of 3 parts:

1. The steering window (top left)
2. The figures of the Pareto set/front
3. The log window (bottom)

In the steering window, the movement direction of the *Pareto Explorer* can be specified. For details, see section 3.1.1. The figures show the Pareto set and front. Depending on the type of visualization (cf. section 5), either all computed points (old / new) or a comparison between the last point and the new point are shown. In the log window, information about the current point as well as the absolute and relative change in all dimensions between the current and the last point are shown. For details, see section 3.5.

3.1.1 Steering

Two possibilities exist to steer on the Pareto set / front, either by defining a steering vector using the steering window or by clicking on the figures. The second option is only available for some types of visualization: *Bars, Value vs. Index, Wheel* (cf. section 3.4).

When using the **Steering window**, three steering methods are available (cf. section 2):

Decision space: Move in the prescribed direction in decision space

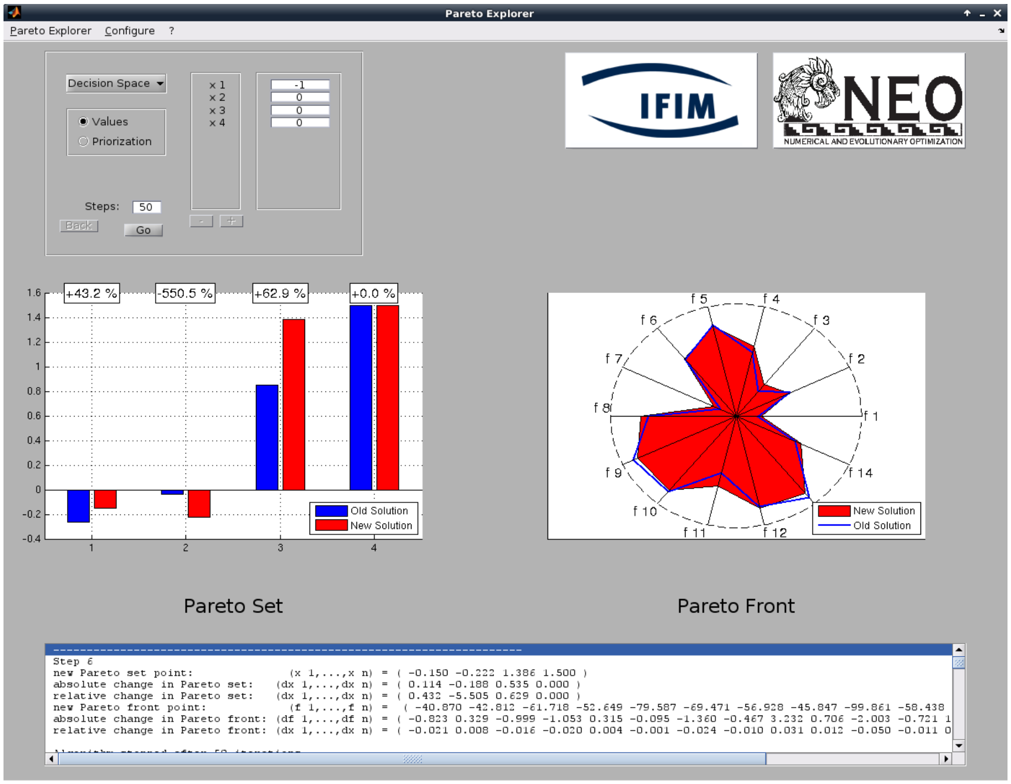

Figure 2: Main window of the Pareto Explorer

Objective space: Move in the prescribed direction in objective space

μ space: Explicitly define μ. Note that μ has to fulfill the condition $\sum_{i=1}^{k} \mu_i = 0$

For each steering method, there is either the possibility to explicitly type in the components of the steering vector by selecting *Values* (cf. fig. 2, $\{x_1, x_2, x_3, x_4\} = \{-1, 0, 0, 0\}$) or to priorize different components over others via sliders by selecting *Priorization*. Steering may be performed in three different spaces

Remark: Note that if the prescribed direction is infeasible because it would result in non-optimal values, the vector is projected onto a feasible direction. If a corner of the Pareto set is reached or the projection is 0, the *Pareto Explorer* stops. This information will be given in the log window. For details on the stopping criteria, see section **??**.

Clicking on the figures:

3.1.2 Menus

- **Pareto Explorer**

  New: Restarts the GUI, thereby deleting all data and resetting all options to default

  Open Case: Loads a previously saved *mat* file, including options and already computed data.

Save Case: Saves all existing information to a *mat* file, including options and already computed data.

Export: Calls the export window (cf. section 3.6), where data, graphics and log data may be exported.

Window Size: Allows to change the size of the main window.

Exit: Ends the *Pareto Explorer*.

- **Configure**

  Model: Load / Configure the model (cf. section 3.2)

  Algorithm: Configure the *Pareto Explorer* algorithm (cf. section 3.3)

  Visualization: Configure the visualization (cf. section 3.4)

  Log: Configure the log window (cf. section 3.1.2)

Configuration: (sections to)

3.2 Loading / Configuring a model

The model can be configured via *Configure* → *Model* (Ctrl + M).

Files have to be given as an m-file of the form [y] = *model(x)* or, alternatively, with multiple outputs ([y, J] = *model(x)*). This depends on the option chosen in the box *Input Options*.

3.2.1 Input Options

All inputs separately: Four files can be selected, each with a single input and a single output (y = *f(x)*). A *model file is mandatory*, the other files are optional. If no file for the Jacobian is given, derivatives are computed using finite differences. If an equality constraint file is given but not a file for the respective gradients, these are computed using finite differences as well.

Model + Jacobian & Equality Constraint + Jacobian: With this option, two files can be given, each with two outputs ([y, J] = *model(x)* / [h, Dh] = *constraints(x)*). It is possible to return the derivatives as empty ([]). Nevertheless, the second output needs to be specified.

All inputs in one file: With this option, one file with four outputs ([y, J, h, Dh] = *model(x)*) needs to be specified. It is possible to return one or more fields as empty ([]). Nevertheless, four outputs need to be specified.

3.2.2 Dimension of the problem

In the box *Dimension of the problem* the correct dimensions of the image and the objective space need to be specified. In case the dimensions are incorrect, an error might occur when hitting the *OK* button since the model is evaluated to compute the initial point.

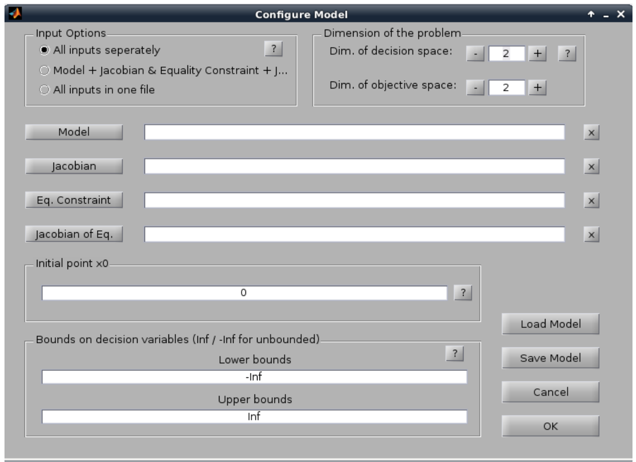

Figure 3: Model configuration window

3.2.3 Initial point x0

Here, the initial point of the exploration is defined. If all parameters have the same initial value (e.g. $x_0 = \{0, 0, 0, 0\}$), it is possible to enter this value once (cf. fig 3, $x_0 = 0$). The other option is to enter a value for each dimension, separated by ";" (e.g. $x_0 = 1; 2; 3; 4$).

3.2.4 Bounds on decision variables

Here, upper and lower bounds for the decision space may be specified. The functionality is the same as for the initial value x_0. For unbounded problems, insert $\pm Inf$.

3.3 Configuring the algorithm

The *Pareto Explorer* algorithm can be configured via *Configure* → *Algorithm* (Ctrl + A).

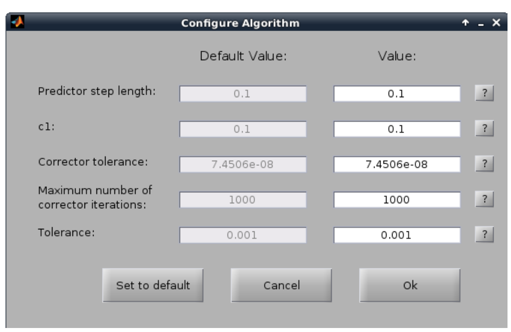

Figure 4: Algorithm configuration window

Here, parameters for the numerical behavior of the *Pareto Explorer* method can be specified (cf. fig. 4). For each parameter, the default value is given as well as an explanation of it's purpose (*?* buttons).

`opts.d`
: is the desired step length in objective space.

This parameter represents an approximation to the distance between the image of two consecutive solutions, i.e.

$$d \approx \|F(x_i) - F(x_{i-1})\|$$

- Default value: `opts.d = 0.1`;

`opts.c1`

constant for the corrector step length control.

The corrector uses a kind of Armijo condition to control the step size in the Newton method for multiobjective optimization problems. This condition allows to get a step size t small enough, for a direction v, such that v is a descent direction and the following inequality is satisfied:

$$f_j(x_i + tv) \leq f_j(x_i) + c_1 t v^T \nabla f_j(x_i),$$

for all $j = 1, \ldots, k$ at the point x_i.

- Default value: `opts.c1 = 0.1;`

`opts.correct_tol`

is the tolerance used to finish the corrector.

This tolerance is used in the newton method for multiobjective optimization problems and it has a relationship with the accuracy of solutions. A small value for the tolerance means better accuracy, but it increases the computational cost.

- Default value: `opts.correct_tol = 5 * sqrt(eps`[1]`);`

`opts.correct_its`

is the max number of iterations for the corrector.

Each new point which is computed by the corrector is considered as a new iteration, and the method takes the total of points for this stop criterion.

- Default value: `opts.correct_its = 1000;`

`opts.tol`

is the tolerance used to finish the continuation algorithm.

This tolerance is considered by stop criteria of the Pareto Explorer method.

- Default value: `opts.tol = 1e-3;`

3.4 Visualization options

The visualization can be configured via *Configure* → *Visualization* (Ctrl + V).

Different types of visualization can be selected for the decision space (left) and the objective space (right), respectively:

[1] Here **eps** is the Floating-point relative accuracy in Matlab. Is the distance from 1.0 to the next largest double-precision number, that is eps $= 2^{-52}$.

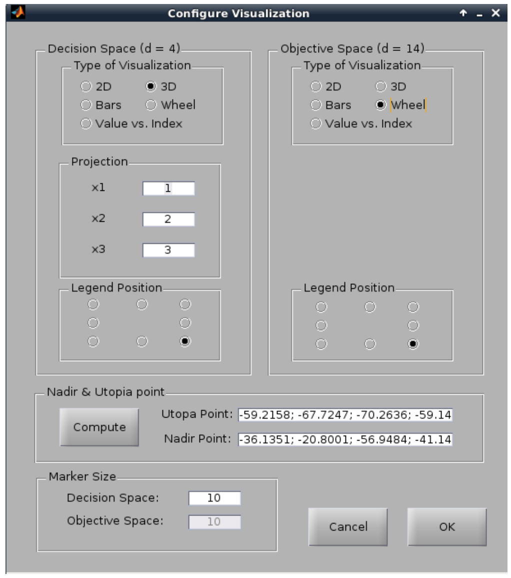

Figure 5: Visualization configuration window

2D/3D: Creates a 2D/3D plot of parameter / function values. In case the respective dimension is higher, a projection window is shown (cf. fig. 5) where the parameter / function values of interest can be chosen. Points that already existed will be displayed in blue, the new computed points will be displayed in red

Bars: Creates a bar plot with a bar for each dimension. The new values are displayed in red, the values from the last step are displayed in blue (cf. fig. 2, left plot). If the number of bars is ≤ 10, the relative change in % is shown in the figure itself.

Wheel: Creates a wheel plot where the parameters/objectives are plotted on axes in radial direction (cf. fig 2, right plot). The new values are displayed in red, the values from the last step are displayed with a blue line. The values are normalized such that the maximal value is in the center and the minimal value is on the outer circle. For this reason, this option is only available for the decision space if all variables have an upper and a lower bound and for the objective space if the Nadir and Utopia point are known. The Nadir and Utopia point can be specified in the visualization options (cf. fig. 5) or, if unknown, calculated. Note that for this purpose, a single objective optimization problem has to be solved for each objective separately.

Value vs. Index: Creates a plot where all parameter / function values are plotted on the y-axis vs their respective index. The new curve is displayed in red, the curve from the last step is displayed in blue. This option may be useful when the decision space is a discretized control trajectory, e.g. in optimal control problems

If the chosen type of visualization has markers (**2D, 3D, Value vs. Index**), their size may be specified in the box *Marker Size*. Also, the *Legend Position* may be selected for both the decision and the objective space.

Remark: The Nadir and Utopia point are updated each time the plots are created. This is necessary for the Nadir point, for the Utopia point only if the point was not explicitely computed or when the computation was incorrect, e.g. due to local minima.

3.5 Log

In the log options (cf. fig. 6), the information occurring in the log window in the bottom of the main GUI (cf. fig. 2, bottom) can be specified. Note that a more extensive log is created which can be exported to a text file (cf. section 3.6).

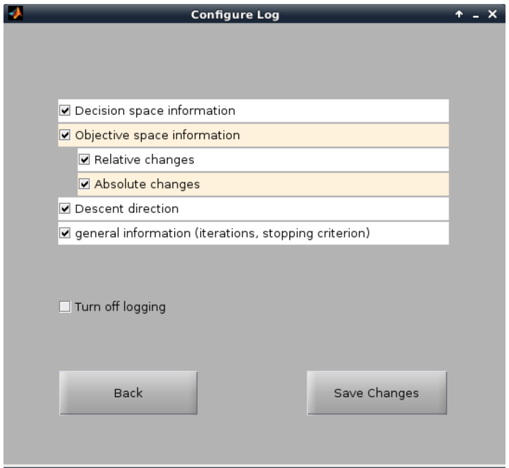

Figure 6: Log configuration window

3.6 Export possibilities

In the export window, all information generated by the *Pareto Explorer* may be exported (cf. fig. 7):

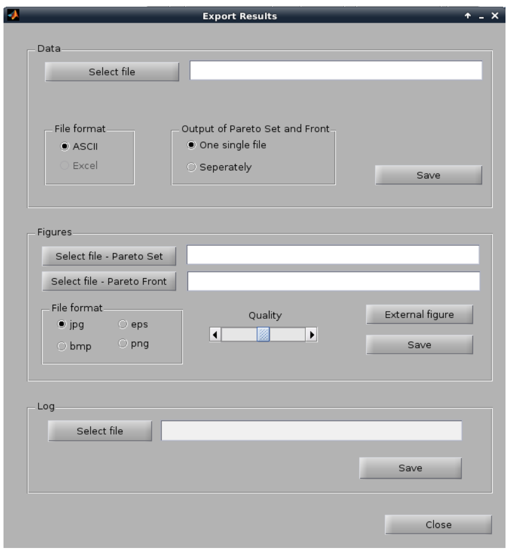

Figure 7: Export window

Top: Computed Points on the Pareto set / front to a text file

Middle: Figures to various file formats; It is also possible to create the figure in an *External figure* to allow further customization

Bottom: Log data to a text file

References

[Hil01] C. Hillermeier. *Nonlinear Multiobjective Optimization - A Generalized Homotopy Approach.* Birkhäuser, 2001.

[MS18] A. Martin and O. Schütze. Pareto tracer: A predictor–corrector method for multi-objective optimization problems. *Eng Opt*, 50(3):516–536, 2018.