DEMO “Toolbox”

Fillipe Goulart

November 19, 2015

Abstract

This is a simple description of how to use this set of m files containing implementations of the Differential Evolution for Multi-objective Optimization (DEMO) algorithm. It is supposed to work on Octave, which is a free Matlab “clone”, but it may work well on Matlab too.

1 Introduction: Just enough about Multi-objective Optimization

Welcome, newcomer! This is a set of m files to execute different variations of the Differential Evolution for Multi-objective Optimization [6] (a.k.a. DEMO), a Multi-objective Evolutionary Algorithm, in order to solve problems with multiple objectives.

“Solve” here is a pretentious term, because, in order to effectively solve a multi-objective problem, one must [1]:

- Compute efficient (or Pareto-optimal) points, candidates to final solution;
- Help a decision maker (DM) to choose a final solution that best pleases its[1] desires.

Notice that these steps need not to be taken in any specific order, that is, the DM may provide its preferences before, during, or after the quest for efficient points.

[1]This text refers to the decision maker by the pronoum “it” in order to emphasize its “genderless” character. That is, a man, a woman, a group of people or even a machine can be used to take decisions.

These philosophies generate the well-known *a priori*, interactive and *a posteriori* methods for solving these problems [5].

In this set of files, there are versions of the DEMO that either take the DM's preferences into account and focus on a smaller set of the Pareto front (which follow the *a priori* or interactive principle), and ones that try to approximate the whole front (the *a posteriori* ones). It is assumed that the DM express its desires by a reference point $\mathbf{z}^r$ [8]. The following algorithms are implemented:

- *A posteriori methods* (without preferences):
  - DEMO [6]: the regular DEMO with non-dominated sorting;
  - IBEA [9]: the DEMO using indicators instead.
- *A priori* or interactive (with preferences):
  - R-DEMO [2]: R-NSGA-II but using the DEMO instead;
  - PBEA [7]: IBEA but using a reference point;
  - PAR-DEMO(nds) [3]: the method proposed by the author using the non-dominated sorting;
  - PAR-DEMO(ϵ) [3]: the same method, but using indicators instead.

The next section describes briefly each folder and the files it contains.

2 What are the files in this "toolbox"?

Notice that there are two folders and a file in this project:

- `Algorithms`: this folder contains the m files of the methods described before, and some other auxiliary functions. They are all (hopefully) very well documented, so, in doubt, try a `help filename` or just open it. It's free!
- `DTLZ`: these are files to implement the well-known DTLZ benchmark. They are extensible to any number of objectives $m > 2$, and also contain some functions to return the Nadir and ideal points, and other cool stuff.
- `includepaths.m`: this is just to help add the previous folders to the working directory.

3 Fine, so, how do I use it?

The complete way is to read the documentation of each function. If you are too lazy to do that and just want a quick start, try the following examples.

3.1 Example 1: Find an approximation of the DTLZ1 Pareto front with 2 objectives

First, start by including all paths in the working directory:

```
>> includepaths
```

Now, create the function to be optimized and get its search space:

```
>> f = @(x) dtlz1(x, 2)
f =

@(x) dtlz1 (x, 2)

>> xrange = dtlz_range ('dtlz1', 2)
xrange =

0   1
0   1
0   1
0   1
0   1
0   1
```

Notice that the search space is limited between 0 and 1 for each variable, and that its dimension (number of rows here) is dependent on the number of objectives. Finally, let us solve it using the regular DEMO with its default parameters (see `help demo_opt` to get a list on its adjustable parameters and their defaults):

```
>> [fopt, xopt] = demo_opt(f, xrange)
```

After some time, you get a matrix wherein each column is an approximation to an efficient point. If you want, you can plot the results:

```
>> plot(fopt(1,:), fopt(2,:), 'o'), xlabel('f_1'), ylabel('f_2')
```

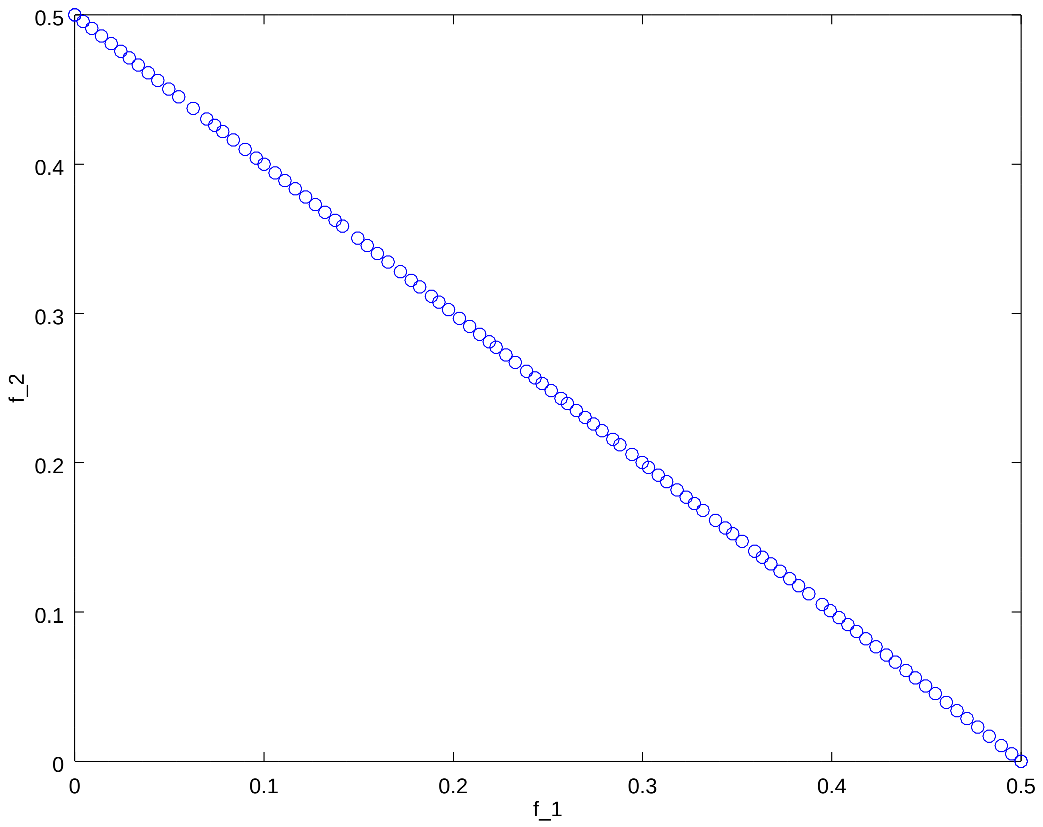

Figure 1: Different regions of interest for each method that includes the DM's preferences.

The results are in Figure 1, in which the points try to be as uniform as possible, while covering the entire extension of the front.

In order to get more numerical results, the function `dtlz_distance` computes the distance of each point to the true efficient *set* (yes, "set", in the variables space).

```
>> d = dtlz_distance (xopt, 'dtlz1');
>> min(d), mean(d), max(d)
ans =    3.4427e-20
ans =    3.1858e-19
ans =    1.1342e-18
```

Notice how close they are to the optimal region. This is usually common for problems with a small number of objectives.

3.2 Example 2: A problem with many objectives

The many-objective optimization field [4] comprises functions with more than 3 objectives. So, let us try to solve the DTLZ2 with 10 objectives with the regular DEMO and see what happens:

```
>> f = @(x) dtlz2(x, 10);
>> xrange = dtlz_range ('dtlz2', 10);
>> [fopt, xopt] = demo_opt(f, xrange);
>> d = dtlz_distance (xopt, 'dtlz2');
>> min(d), mean(d), max(d)
ans =  1.8075
ans =  2.2907
ans =  2.4995
```

Of course, now you cannot plot the results like before, since `fopt` is a matrix with 10 rows, but you can compute the distances to the efficient set. Here, these values are way bigger than before. This is expected, since algorithms based on non-dominated sorting usually do not scale well with the number of objectives. If we try an indicator based method like the IBEA, the results become:

```
>> [fopt2, xopt2] = demo_ibea_opt(f, xrange);
>> d2 = dtlz_distance (xopt2, 'dtlz2');
>> min(d2), mean(d2), max(d2)
ans =    4.3372e-06
ans =    2.8357e-05
ans =    1.6762e-04
```

which are much better. Usually, indicator based algorithms are less vulnerable to increases in the number of objectives.

3.3 Example 3: Focusing on a smaller portion of the Pareto front

The inclusion of a reference point $\mathbf{z}^r$ to describe the DM's preferences has the effect of concentrating the population in a smaller subset of the efficient front, which is called the *region of interest* (ROI) [3]. The size of this portion varies from method to method, but in each algorithm I included some default parameters given in their respective papers. One exception is the one I proposed [3], which does not require any parameter (apart from the ones of the basic DEMO).

So, enough of gibberish. Let us solve the DTLZ3 with 2 objectives, using as a reference point $\mathbf{z}^r = [0.6\ 0.2]^T$, for each algorithm that includes the DM's preferences.

```
>> zr = [0.6 0.2]';
>> f = @(x) dtlz3(x, 2);
>> xrange = dtlz_range ('dtlz3', 2);

% R-DEMO:
>> [fopt1, xopt1] = rdemo_opt(f, zr, xrange);

% PBEA:
>> [fopt2, xopt2] = demo_pbea_opt(f, zr, xrange);

% PAR-DEMO(nds)
>> [fopt3, xopt3] =demo_par_nds(f, zr, xrange);

% PAR-DEMO(epsilon)
>> [fopt4, xopt4] =demo_par_ind(f, zr, xrange);
```

Now, plot the results in order to compare the different ROI sizes:

```
>> subplot(2,2,1)
>> plot(fopt1(1,:), fopt1(2,:), 'o', zr(1), zr(2), '*'); axis([0 1 0 1])
>> title('R-DEMO')
>> subplot(2,2,2)
>> plot(fopt2(1,:), fopt2(2,:), 'o', zr(1), zr(2), '*'); axis([0 1 0 1])
>> title('PBEA')
>> subplot(2,2,3)
>> plot(fopt3(1,:), fopt3(2,:), 'o', zr(1), zr(2), '*'); axis([0 1 0 1])
>> title('PAR-DEMO(nds)')
>> subplot(2,2,4)
>> plot(fopt4(1,:), fopt4(2,:), 'o', zr(1), zr(2), '*'); axis([0 1 0 1])
>> title('PAR-DEMO(\epsilon)')
```

This should generate Figure 2. Check the differences in ROI sizes for each method.

3.4 Example 4: How preferences can improve an algorithm

We have seen in example 2 that the regular DEMO has worse results when the number of objectives is increased. In this example, let us compare its basic version

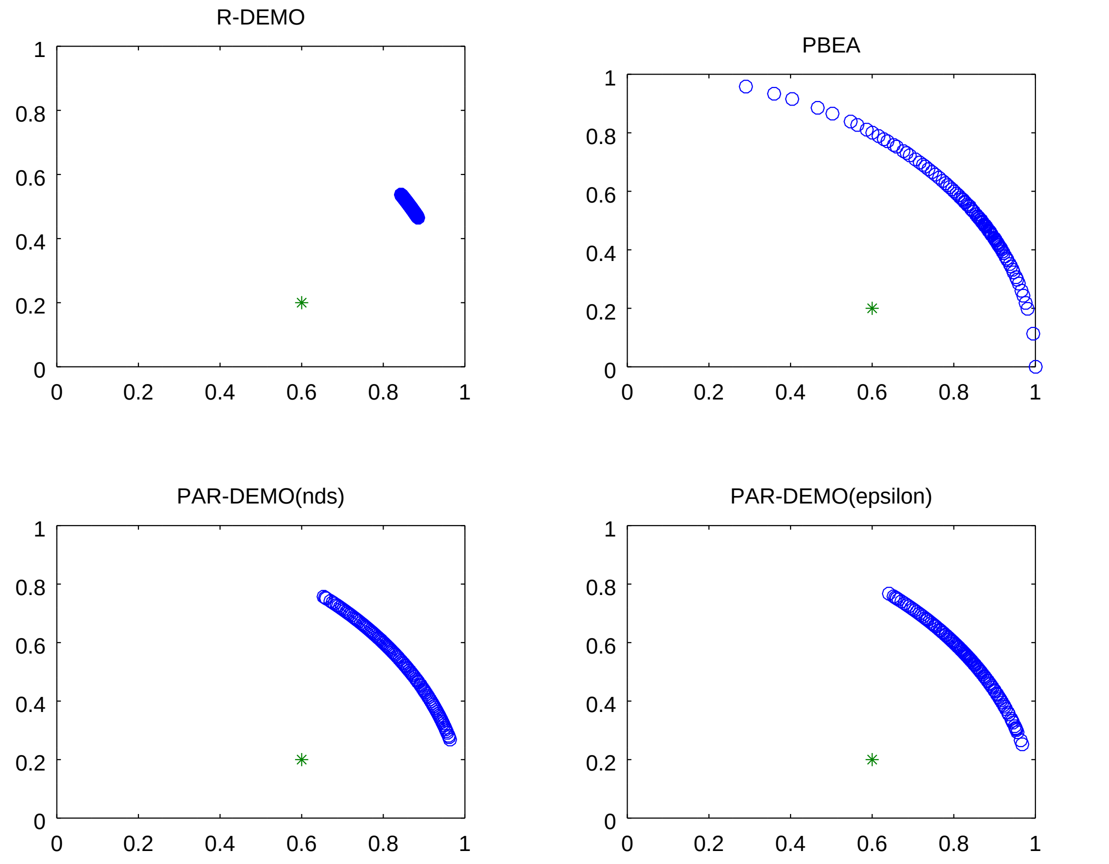

Figure 2: Different regions of interest for each method that includes the DM's preferences.

with the PAR-DEMO(nds), which simply includes a reference point and limits the region of interest of the efficient front.

For that, let us optimize again the DTLZ2 function with 10 objectives. For the PAR-DEMO(nds), use as a simple reference point

$$\mathbf{z}^r = [0.25 \; 0.25 \; \ldots \; 0.25]^T$$

```
% Create the function and the reference point
>> f = @(x) dtlz2(x, 10);
>> xrange = dtlz_range ('dtlz2', 10);
>> zr = 0.25(ones(10,1)) %or repmat(0.25, 10, 1) if this does not work
```

```
% Regular DEMO
>> [fopt, xopt] = demo_opt(f, xrange);
>> d = dtlz_distance (xopt, 'dtlz2');
>> min(d), mean(d), max(d)
ans =  1.4047
ans =  2.2914
ans =  2.4883

% PAR-DEMO(nds), which is the DEMO with preferences
>> [fopt2, xopt2] = demo_par_nds(f, zr, xrange);
>> d2 = dtlz_distance (xopt2, 'dtlz2');
>> min(d2), mean(d2), max(d2)
ans =    4.5072e-06
ans =    5.0641e-05
ans =    1.6245e-04
```

Notice how much closer the points are. This is a big improvement by just focusing on a smaller subset of the efficient front! Of course, in this case, we do not have information about the whole front, but this is exactly the point in an interactive (or *a priori*) method. Check our paper [3] if this intro got you interested.

References

[1] J Branke, K Deb, K Miettinen, and R Slowinski. *Multiobjective optimization: Interactive and evolutionary approaches*. 2008.

[2] Kalyanmoy Deb, J. Sundar, Rao N. Udaya Bhaskara, and Shamik Chaudhuri. Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. *International Journal of Computational Intelligence Research*, 2(3):273–286, 2006.

[3] Fillipe Goulart and Felipe Campelo. Preference-guided evolutionary algorithms for many-objective optimization. *Information Sciences*, 329:236 – 255, 2016. Special issue on Discovery Science.

[4] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. *2008 IEEE Congress on Evolutionary Computation IEEE World Congress on Computational Intelligence*, (March):2419–2426, 2008.

[5] K. Miettinen. *Nonlinear Multiobjective Optimization.* International Series in Operations Research & Management Science. Springer US, 1999.

[6] T Robič and B Filipič. DEMO: Differential evolution for multiobjective optimization. *Evolutionary Multi-Criterion Optimization*, pages 520–533, 2005.

[7] Lothar Thiele, Kaisa Miettinen, PJ Korhonen, and Julian Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17(3):411–436, 2009.

[8] Andrzej P. Wierzbicki. A mathematical basis for satisficing decision making. *Mathematical Modelling*, 3(5):391–405, January 1982.

[9] Eckart Zitzler and S Künzli. Indicator-based selection in multiobjective search. *Parallel Problem Solving from Nature-PPSN VIII*, (i):1–11, 2004.