Universiteit Leiden
Opleiding Informatica

Cooperative Swarm-based and Gradient-based Multi-objective Optimization:

Visualization and Comparison of Search Dynamics

Name: Wilco Verhoef

Date: 01/03/2015

1st supervisor: Michael Emmerich
2nd supervisor: André Deutz

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Cooperative Swarm-based and Gradient-based Multi-objective Optimization:
Visualization and Comparison of Search Dynamics

Wilco Verhoef
wilco@verhoef.nu
wilco.verhoef.nu

Supervised by: Michael Emmerich and André Deutz

March 1, 2015

Abstract. This thesis is about the numerical solution of multiobjective optimization problems in continuous spaces. The problem is to define a search direction and dynamical adaptation scheme for sets of vectors that serve as approximation sets. The dynamics of such algorithms will be visualized by a set of tools that is developed in this project. In this thesis two new algorithmic concepts are compared; a novel stochastic optimization algorithm based on cooperative particle swarms, and a deterministic optimization algorithm based on set-based gradients. Both approximate the Pareto front dynamically. Comparisons will be made on the efficiency and effectiveness. The results show that the approaches are both capable to produce approximations of the Pareto front, each of which is diverse. This research provides interesting new directions for developing algorithms for set-oriented optimization.

Contents

1 Introduction

Multi-Objective Optimization (MOO) is a class of optimization problems where the ideal solution should optimize multiple objectives simultaneously. In typical MOO problems, the objectives are conflicting. Thus, an optimal solution for one objective is necessarily not optimal for the others. With conflicting objectives, there is no single optimal solution for the problem. Instead, there is a whole set of solutions in the decision space that could be considered optimal. We call this set the efficient set. The image by the objective function is called the Pareto front. In MOO we strive to obtain close approximations of the efficient set in order to approximate the Pareto front.

MOO problems are common in numerous fields including engineering, science, industry, finance and logistics [6, 17, 15]. MOO algorithms are for example already extensively used in supply chain management [1]. Given this broad range of application areas, there is a big need for reliable MOO algorithms.

For this we will introduce new algorithmic concepts in this thesis for multi-objective optimization, namely a *Multi-Objective COoperative Particle Swarm* (MOCOPS) algorithm and a *Multi-Objective Gradient based Optimization* (MOGO) algorithm. In this thesis we will elaborate on the dynamics of these algorithms.

The specific contribution of this research is three-fold. We will analyze the effectiveness and efficiency of the new algorithmic concepts by using the hypervolume indicator as quality measurement. Furthermore we will compare the MOCOPS and MOGO algorithm. Thirdly this research will focus on a dynamic visualization of the algorithms.

This thesis is structured as follows. In Section 2 we will first introduce the definitions and notation that are used in the remainder of the thesis. Next in this section we will state the problem definition. Lastly in Section 2 there will be a subsection dedicated to related work. In Section 3 different variant of the multi-objective optimization algorithms are presented. In Section 4 the proposed algorithms will be tested with several benchmarks. First of all the test problems, and secondly the experiments setup will be defined. After defining the experiment, the results will subsequently be described and discussed. We sum up the conclusions in Section 5. Following the conclusion is the bibliography. Lastly the thesis ends with the appendices. These will consist of a manual of the application, and a section describing the code directory.

2 Background

2.1 Definitions and notation

The space of candidate solutions is called the *decision space*. The space of objective function values with the decision space as domain is called the *objective space* [8].

For optimization, it is desirable to have an unambiguous way of determining whether an arbitrary vector is considered better than another. For this reason, you can define the relations *weakly-Pareto-dominance* and *strictly-Pareto-dominance* between two vectors $\vec{\mathbf{x}}, \vec{\mathbf{y}} \in \mathbb{R}^m$. Weakly-Pareto-dominance is a relation between two vectors where one vector is considered better or equal than another. We will assume that our objective is minimization. Vector $\vec{\mathbf{x}}$ weakly-Pareto-dominates vector $\vec{\mathbf{y}}$ if and only if Equation 1 holds.

$$\forall_{i \in \{1,2,\ldots,m-1,m\}} \ \vec{\mathbf{x}}_i \leq \vec{\mathbf{y}}_i \tag{1}$$

A vector $\vec{\mathbf{y}}$ is considered *strictly-Pareto-dominated* by $\vec{\mathbf{x}}$ if and only if Equation 2 holds [11, 5, 16].

$$\forall_{i\in\{1,2,\ldots,m-1,m\}}\ \vec{\mathbf{x}}_i \leq \vec{\mathbf{y}}_i \wedge \exists_{i\in\{1,2,\ldots,m-1,m\}}\ \vec{\mathbf{x}}_i < \vec{\mathbf{y}}_i \tag{2}$$

In the remainder of this thesis we will use the latter definition for Pareto-dominance. Pareto-dominance can be considered as a partial order where dominance of $\vec{\mathbf{x}}$ over $\vec{\mathbf{y}}$ is denoted as $\vec{\mathbf{x}} \prec \vec{\mathbf{y}}$ [9].

Let $\mathbb{S} \subseteq \mathbb{R}^d$ and $\vec{f} : \mathbb{S} \to \mathbb{R}^m$. In optimization problems $\mathbb{S}$ represents the decision space, and $\vec{f}$ the objective functions. The Pareto front of the minimization problem is the non-dominated subset of the image of $\vec{f}$. The Pareto set is also known as the *Pareto front* or *Pareto optimum*. We define the Pareto set Y_{PF} in Equation 3.

$$Y_{PF} := \{\vec{f}(\vec{\mathbf{x}}) \mid \vec{\mathbf{x}} \in \mathbb{S} \wedge \nexists_{\vec{\mathbf{x}}\prime\in\mathbb{S}}\ \vec{f}(\vec{\mathbf{x}}\prime) \prec \vec{f}(\vec{\mathbf{x}})\} \tag{3}$$

The inverse image of the Pareto set with respect to $\vec{f}$ is called the *efficient set*. The efficient set X_E is defined in Equation 4.

$$X_E := \{\vec{\mathbf{x}} \in \mathbb{S} \mid \exists_{\tilde{\mathbf{y}}\in Y_{PF}}\ \vec{f}(\tilde{\mathbf{x}}) = \tilde{\mathbf{y}}\} \tag{4}$$

The *hypervolume indicator* is a generally usable measure for multi-objective optimization indicating how well the population approximates the Pareto front [13]. With this method, a reference point is chosen. Preferably the reference point is chosen such that it is dominated by all images of the decision space. For example the maximal point can be used as a reference point. Given the objective space Y and m objective functions, the maximal point $\bar{y}$ is defined by Equation 5 [11].

$$\bar{y} = (\max_{y\in Y} y_1, \ldots, \max_{y\in Y} y_m) \tag{5}$$

The hypervolume indicator of a population is equal to the hypervolume of the subset of $\mathbb{R}^m$ which consists of points which dominate the reference point and in turn are dominated by some point of the population.

The *hypervolume contribution indicator* of a particle is defined as the hypervolume of the total population, minus the hypervolume of the population without that particle. Particles which are part of the dominated set have a hypervolume contribution of 0. The hypervolume is always at least as big as the hypervolume contributions of the particles combined. The concepts of domination, hypervolume and hypervolume contribution are clarified in Figure 1.

A *fitness function* is a single-objective scalar function that denotes the value of a certain particle. A fitness function is used in heuristic optimization algorithms to compare different particles. Good heuristic algorithms will efficiently converge to a high fitness value. In turn a fitness function yields a low and high value in case the population is respectively unoptimized and highly optimized.

In the literature regarding optimization algorithms, different terms with similar meanings are used depending on the field of research. The literature regarding *Evolutionary Algorithms* (EAs) mention individuals, while the literature regarding *Particle Swarm Optimization* (PSO) algorithms mention particles. Below is a short summary of interchangeable terms. In this thesis an attempt has been made to match each field with each of their terms.

EA	PSO	Gradient optimization
fitness (function)	fitness (function)	objective function
individual	particle	(search) point
population	swarm	multiset

It was attempted to use a similar notational convention as commonly used in other literature regarding optimization such as for example the notational conventions used in EAs [3].

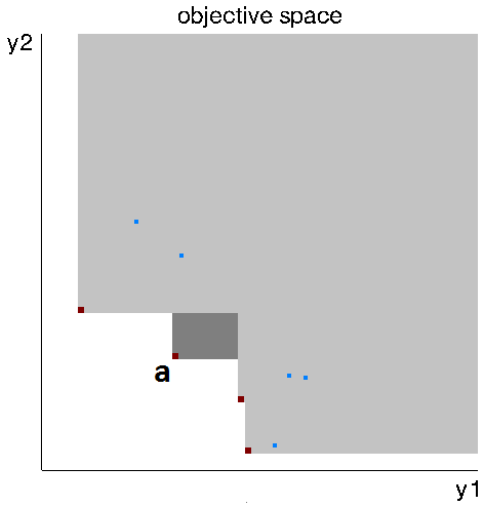

Figure 1: This is an example of a population in the objective space. Particles are displayed in red in case they are not dominated by another particle in the population. Particles are displayed in blue when they are part of the dominated set. The total hypervolume covered by the population is colored gray. The hypervolume contribution of particle a is colored dark gray.

Notation	Description
$\mathbb{S}$	Decision space
$\vec{\mathbf{x}} \in \mathbb{S}$	Representation of a particle in decision space
$\vec{\mathbf{y}} \in \mathbb{R}^m$	Representation of a particle in objective space
$\vec{f} : \mathbb{S} \to \mathbb{R}^m$	The objective function
μ	Population size
$\Phi : \mathbb{S}^\mu \times \mathbb{N} \to \mathbb{R}$	Fitness of a certain particle given a population
$HV : \mathbb{S}^\mu \to \mathbb{R}$	The hypervolume indicator of a population
$\Delta HV : \mathbb{S}^\mu \times \mathbb{N} \to \mathbb{R}$	The hypervolume contribution indicator of a particle
$i \sim u(\{a_1, \ldots, a_r\})$	A random discrete sample $i \in \{a_1, \ldots, a_r\}$
$x \sim u([0, 1])$	A uniformly random continuous sample $x \in [0, 1]$
$\vec{\mathbf{z}} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$	A vector of gaussian random samples in $[0, 1]$

2.2 Problem definition

For optimizing, minimization is assumed for all objectives throughout this thesis.

$$\vec{f}(\vec{\mathbf{x}}) \to \min \tag{6}$$

The research is limited to 2 dimensions in both the decision and objective space for simplicity.

$$\mathbb{S} \subseteq \mathbb{R}^2 \tag{7}$$

The aspiration of the optimization algorithms in this thesis is maximization of the hypervolume indicator. That will be the measurement used for benchmarking.

$$HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}) \to \max \tag{8}$$

2.3 Related work

Research about gradient optimization algorithms in MOO using the hypervolume indicator has been performed before [12]. In the research, computation complexity of the algorithms is reported for a varying numbers of dimensions.

In EA, NSGA-II is a well established method for MOO. NSGA-II introduced a fast non-dominated sorting[1] approach in combination with an elitist approach [7].

Research is performed in EA using the hypervolume indicator metric in combination with non-dominated sorting as fitness. The algorithm performed very well in comparison with other EAs such as NSGA-II [10].

3 Optimization algorithms

In Algorithm 1 is displayed how a population is initialized in a randomly distributed way. This initialization is used in both of the following algorithmic concepts.

Algorithm 1 Initialize randomly distributed population

for $i \in \{1, 2, \ldots, \mu - 1, \mu\}$ **do**
  $\vec{\mathbf{a}}_1 \sim u([0, 1])$
  $\vec{\mathbf{a}}_2 \sim u([0, 1])$
  $\vec{\mathbf{x}}^{(i)} \leftarrow \vec{\mathbf{a}}$
end for

3.1 Multi-objective cooperative particle swarm algorithm

PSO is an EA where a swarm of particles is stochastically driven. In conventional PSO algorithms, the swarm is driven by a certain leader. In single-objective optimization an exploitation of the position of the leader will typically converge to a local, or sometimes even global optimum. Such a solution will often approach the Pareto optimum in single-objective problems. In the traditional PSO algorithms a swarm will thus lead to unity, whereas in multi-objective optimization you need diversification to adequately approach the Pareto set[2]. Using traditional PSO for multi-objective optimization problems have been tried before [14], but we think the traditional PSO algorithms do not fit multi-objective optimization problems.

In this subsection we will introduce our own interpretation of a MOCOPS. We propose a simple MOCOPS based on the hypervolume contribution. Compared to more traditional PSO algorithms, focus is shifted from exploitation to exploration.

Our proposed MOCOPS algorithm starts with randomly initializing every particle. After the initialization the algorithm will select a random particle of the population. This particle is mutated with a gaussian random sample. The fitness values of the particle before and after mutation are assessed. If the fitness value of the mutated particle is equal or better than the particle will be replaced. Otherwise the mutated particle will be rejected. The cycle continues with picking a random particle again. The MOCOPS algorithm is displayed in pseudocode in Algorithm 2. Care must be taken to ensure $\vec{\mathbf{a}} \in \mathbb{S}$.

In Equation 9 the fitness function of particle i when only enabling the Pareto front of the MOCOPS algorithm is shown. In this case the fitness values of all particles which are dominated by another particle in the population are 0.

[1] *Non-dominated sorting* is a method in which, recursively, different ranks of Pareto sets are formed in the population. First the Pareto set of the total population is formed. Then a new Pareto set is formed out of the remaining population. This repeats until the population consists entirely of different ranks of Pareto sets.

[2] In trivial situations where the objectives are not conflicting, no diversification is needed. In this case, the problem can easily be restated as single-objective problems instead.

Algorithm 2 MOCOPS

```
Initialize randomly distributed population
loop
  i ~ u({1, 2, ..., n − 1, n})
  z ~ N(0, I)
  a ← x^(i) + σ · z
  if Φ({x^(1), ..., x^(i−1), a, x^(i+1), ..., x^(μ)}, i) ≥ Φ({x^(1), ..., x^(i−1), x^(i), x^(i+1), ..., x^(μ)}, i)
  then
    x^(i) ← a
    y^(i) ← f(x^(i))
  end if
end loop
```

$$\Phi(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i) = \Delta HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i) \tag{9}$$

The hypervolume contribution indicator always will be 0 or more, thus $\Phi(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i)$ is in $[0, \infty)$.

Below the fitness function when enabling the whole population for the MOCOPS algorithm is shown. It is assumed that $\vec{\mathbf{y}}^{(i)} \in [0, \infty)^2$.

$$\Phi(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i) = \begin{cases} \Delta HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i), & \text{if } \Delta HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i) > 0 \\ -|\vec{f}(\vec{\mathbf{x}}^{(i)})|, & \text{if } \Delta HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i) = 0 \end{cases} \tag{10}$$

In this case the range is broader and also includes negative values, thus $\Phi(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i)$ is in $(-\infty, \infty)$. In this case a particle which is not dominated by another particle in the population will have a fitness of at least 0. A particle which is dominated by another particle in the population will have a fitness of less than 0.

Adaptive mutation

In general when the mutations are largely successful, the mutation rate σ can be improved for a faster convergence. However, when a high fraction of the mutations are unsuccessful, the mutation rate should be lowered instead for a faster convergence. Getting the mutation rate right can be crucial for efficiency. In earlier research it has been heuristically determined that adapting the mutation rate to a success rate of $\frac{1}{5}$ is a good balance [2]. This has been called the $1/5^{th}$ *succes rule*. In practice σ could be multiplied by a scalar every time the mutation rate is successful and multiplied by the 4^{th} root of that scalar such that σ remains stable with a success rate of $\frac{1}{5}$.

The MOCOPS algorithm with adaptive mutation is displayed in Algorithm 3.

Complexity class

Both the MOCOPS algorithms with and without adaptive mutation are in the complexity class depicted in Equation 11, given a 2-dimensional objective space. Reason for the computational complexity is formed by calculating the hypervolume contribution. This computation is beyond the scope of this thesis.

$$\mathcal{O}(\mu \log \mu) \tag{11}$$

Algorithm 3 MOCOPS with Adaptive Mutation

Initialize randomly distributed population
loop
  $i \sim u(\{1, 2, \ldots, n-1, n\})$
  $\vec{\mathbf{z}} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$
  $\vec{\mathbf{a}} \leftarrow \vec{\mathbf{x}}^{(i)} + \sigma \cdot \vec{\mathbf{z}}$
  if $\Phi(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(i-1)}, \vec{\mathbf{a}}, \vec{\mathbf{x}}^{(i+1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i) \geq \Phi(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(i-1)}, \vec{\mathbf{x}}^{(i)}, \vec{\mathbf{x}}^{(i+1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i)$ **then**
    $\vec{\mathbf{x}}^{(i)} \leftarrow \vec{\mathbf{a}}$
    $\vec{\mathbf{y}}^{(i)} \leftarrow \vec{f}(\vec{\mathbf{x}}^{(i)})$
    $\sigma \leftarrow \sigma \cdot 1.05$
  else
    $\sigma \leftarrow \sigma \cdot \sqrt[-4]{1.05}$
  end if
end loop

3.2 Multi-objective gradient based optimization algorithm

The MOGO algorithm is a deterministic algorithm where each search point is simultaneously directed by a gradient. For example the search point $\vec{\mathbf{x}}^{(i)}$ could be directly directed by the gradient of $\vec{f}$. This will converge to a local optimum. It is however expected that using the gradient this way will lead to little diversification. Therefore instead we will be using the gradient of $\vec{f}$ indirectly with use of the hypervolume contribution. This will enable each search point to follow the direction in which their own hypervolume contribution will increase most. This should lead to a good diversification.

The MOGO algorithm starts with initializing a randomly distributed population. For every search point in the population a Jacobian matrix will be calculated. A Jacobian matrix consists of the first-order partial derivatives of a vector function. The values in a Jacobian indicate how the objective function values respond on a change of the input. In our case we have 2 dimensions for both our decision and objective space, and the Jacobian will thus have 2×2 dimensions. The Jacobian Matrix of $\vec{f}$ at $\vec{\mathbf{x}}$ is shown in Equation 12 [3].

$$\mathbf{J}(\vec{f}, \vec{\mathbf{x}}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\vec{\mathbf{x}}) & \frac{\partial f_1}{\partial x_2}(\vec{\mathbf{x}}) \\ \frac{\partial f_2}{\partial x_1}(\vec{\mathbf{x}}) & \frac{\partial f_2}{\partial x_2}(\vec{\mathbf{x}}) \end{pmatrix} \tag{12}$$

The hypervolume contribution derivative vector is shown in Equation 13. This vector shows how the hypervolume contribution of a certain particle is affected by a change of the objective values.

$$\nabla_y \Delta HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i) = \begin{pmatrix} \frac{\partial \Delta HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i)}{\partial f_1(\vec{\mathbf{x}}^{(i)})} \\ \frac{\partial \Delta HV(\{\vec{\mathbf{x}}^{(1)}, \ldots, \vec{\mathbf{x}}^{(\mu)}\}, i)}{\partial f_2(\vec{\mathbf{x}}^{(i)})} \end{pmatrix} \tag{13}$$

Multiplication of the matrix and vector yields a gradient of the $\nabla_y \Delta HV$ as shown in Equation 14. This vector shows how a change in the search space propagates in a change of the hypervolume contribution of a certain particle.

[3] 2 dimensions are assumed for $\vec{f}$, as well as $\vec{\mathbf{x}}$

$$\mathbf{J}(\vec{f},\vec{\mathbf{x}})\cdot\Delta\nabla_y HV(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)=$$

$$\nabla\Delta HV(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)=\begin{pmatrix}\frac{\partial\Delta HV(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)}{\partial x_1^{(i)}}(\vec{\mathbf{x}}^{(i)})\\ \frac{\partial\Delta HV(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)}{\partial x_2^{(i)}}(\vec{\mathbf{x}}^{(i)})\end{pmatrix} \tag{14}$$

Normalizing the gradient yields the unit vector $\widehat{\nabla\Delta HV}(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)$ as shown in Equation 15. Normalizing is done with the purpose of preventing that the optimization stagnates.

$$\widehat{\nabla\Delta HV}(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)=\frac{\nabla\Delta HV(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)}{|\nabla\Delta HV(\{\vec{\mathbf{x}}^{(1)},\ldots,\vec{\mathbf{x}}^{(\mu)}\},i)|} \tag{15}$$

The MOGO algorithm is displayed in Algorithm 4.

Algorithm 4 MOGO

```
Initialize randomly distributed population
loop
  for i ∈ {1, 2, ..., μ − 1, μ} do
    a⃗^(i) ← x⃗^(i) + σ · ∇Δ̂HV({x⃗^(1), ..., x⃗^(μ)}, i)
  end for
  for i ∈ {1, 2, ..., μ − 1, μ} do
    x⃗^(i) ← a⃗^(i)
    y⃗^(i) ← f⃗(x⃗^(i))
  end for
end loop
```

Complexity class

The MOGO is in the computational complexity class shown in Equation 16, just like the MOCOPS algorithm. Again the reason for this computational complexity lies in finding the hypervolume contribution. In contrast with the MOCOPS algorithm, the MOGO algorithm will move every particle simultaneously in every iteration.

$$\mathcal{O}(\mu\log\mu) \tag{16}$$

4 Evaluation

4.1 Test problems

4.1.1 Problem 1

The objective functions for problem 1 are depicted in Equation 17 and Equation 18. The reference point used for the hypervolume indicator will be the maximal point $(1.25, 1.25)$. Problem 1 is visualized in Figure 2.

$$f_1(\vec{\mathbf{x}})=(x_1)^2+(x_2-0.5)^2 \tag{17}$$

$$f_2(\vec{\mathbf{x}})=(x_1-1)^2+(x_2-0.5)^2 \tag{18}$$

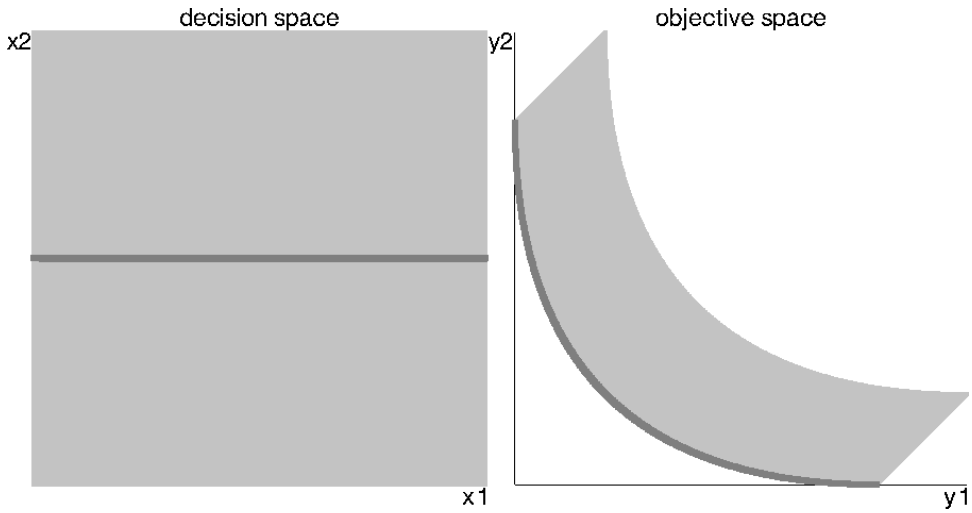

Figure 2: The decision space and objective space of test problem 1 are shown using a uniformly distributed population of 250000 particles. The area covered with dark gray particles denote the efficient and Pareto set among the population. The area covered with light gray particles resembles the subset which is dominated by the Pareto set of the population. The efficient set is horizontal.

4.1.2 Problem 2

The objective functions for problem 2 are depicted in Equation 19 and Equation 20. The reference point used for the hypervolume indicator will be the maximal point $(1, 1)$. Problem 2 is visualized in Figure 3.

$$f_1(\vec{\mathbf{x}}) = 1 - ((x_1)^2 + 1)(x_2)^2 \tag{19}$$

$$f_2(\vec{\mathbf{x}}) = 1 - ((x_2)^2 + 1)(x_1)^2 \tag{20}$$

4.2 Experiments setup

With $\mu = 100$ we are going to compare the different algorithms based on the hypervolume indicator with varying amount of iterations. The amount of iterations that will be used is 10, 50, 100, 500 and 1000. Each of these tests is run 100 times and averaged. The 10, 50, 100, 500 and 1000 iterations tests will be performed on both test problems. The MOCOPS will be benchmarked with and without adaptive mutation. The MOCOPS algorithms will have their mutation rate initialized with 0.2. The step size of the MOGO is initialized with 0.0008.

Also we will have a look at the dynamics of the algorithms in a visual way. We will compare the converged populations of the different algorithms.

4.3 Description of results

The results of the benchmark on problem 1 and 2 can respectively be found in Table 1 and Table 2. MOCOPS is the MOCOPS algorithm without adaptive mutations and MOCOPS A is the MOCOPS algorithm with adaptive mutation. The numbers represented in the tables representing the hypervolume indicator are visualized for problem 1 in Figure 4 and for problem 2 in Figure 5.

In Figure 6 and Figure 7 the converged populations of respectively the MOCOPS and MOGO algorithm are shown of test problem 2. It seems that the algorithms

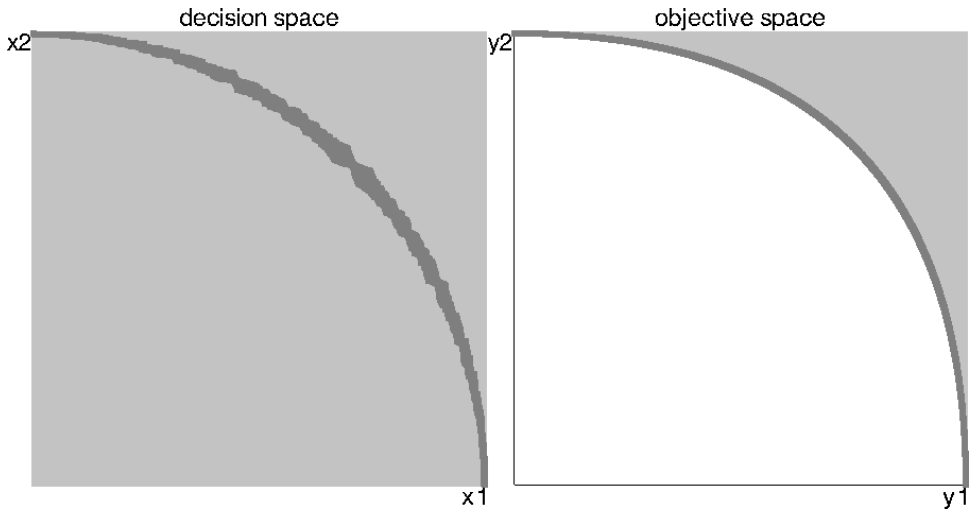

Figure 3: The decision space and objective space of test problem 2 are shown using a uniformly distributed population of 250000 particles. The area covered with dark gray particles denote the efficient and Pareto set among the population. The area covered with light gray particles resembles the subset which is dominated by the Pareto set of the population. The efficient set is curved.

Table 1: In this table the hypervolume indicator is shown using a varying amount of iterations on test problem 1.

Algorithm	10	50	100	500	1000
MOCOPS	1.36053	1.36552	1.37069	1.384	1.38793
MOCOPS A	1.36013	1.36583	1.37003	1.38032	1.38511
MOGO	1.36442	1.37472	1.38098	1.39014	1.39087

Table 2: In this table the hypervolume indicator is shown using a varying amount of iterations on test problem 2.

Algorithm	10	50	100	500	1000
MOCOPS	1.35933	1.36565	1.37033	1.38419	1.38787
MOCOPS A	1.36107	1.36587	1.37042	1.38027	1.38501
MOGO	1.36341	1.37507	1.38108	1.39015	1.39086

converge with a slightly different alignment. Both alignments are diverse and approximate the real Pareto set.

In Figure 8 the steps of the particles are shown as trails. This reveals that particles tend to evade other particles. This leads to a better diversification.

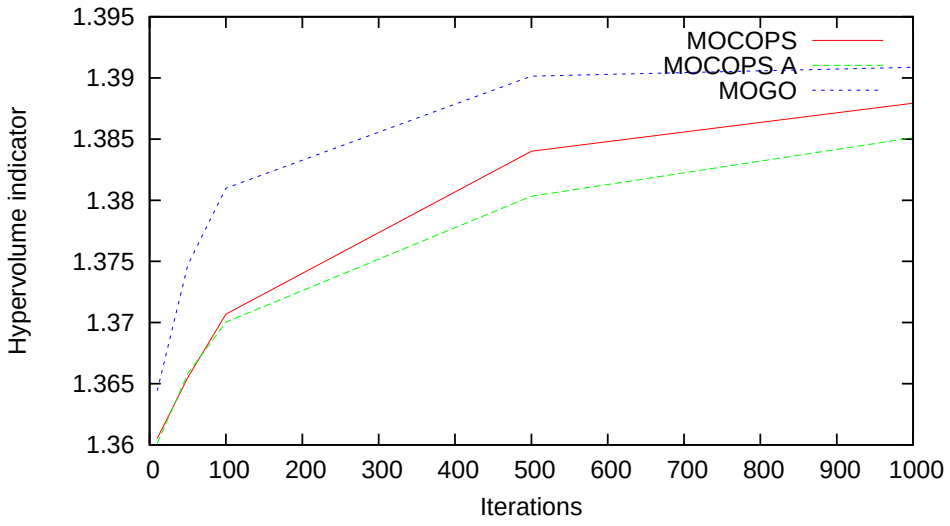

Figure 4: Graph depicting the hypervolume indicator of the different variants of optimization algorithms on problem 1 after a number of iterations.

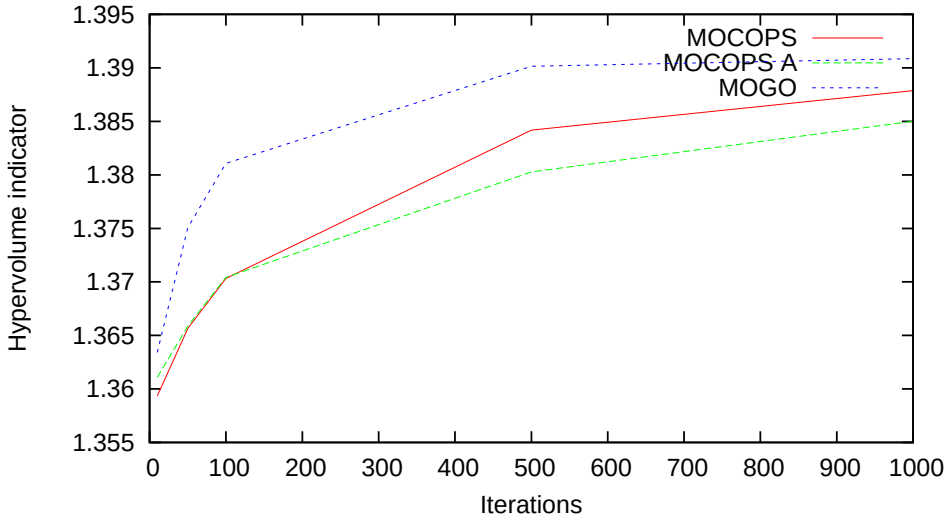

Figure 5: Graph depicting the hypervolume indicator of the different variants of optimization algorithms on problem 2 after a number of iterations.

4.4 Discussion of results

Looking at the benchmarks, it seems that the performance of the algorithms is comparable. The MOGO seems to perform slightly better than the MOCOPS variants. The visualization seems to offer more convenient information. Experimenting with the algorithms show that nearly the whole population quickly finds a spot on the Pareto front with the MOGO algorithm. In the MOCOPS algorithm some particles tend to lag behind. Since the MOGO algorithm updates the whole population simultaneously, it is much faster with larger populations. Also, since the algorithm

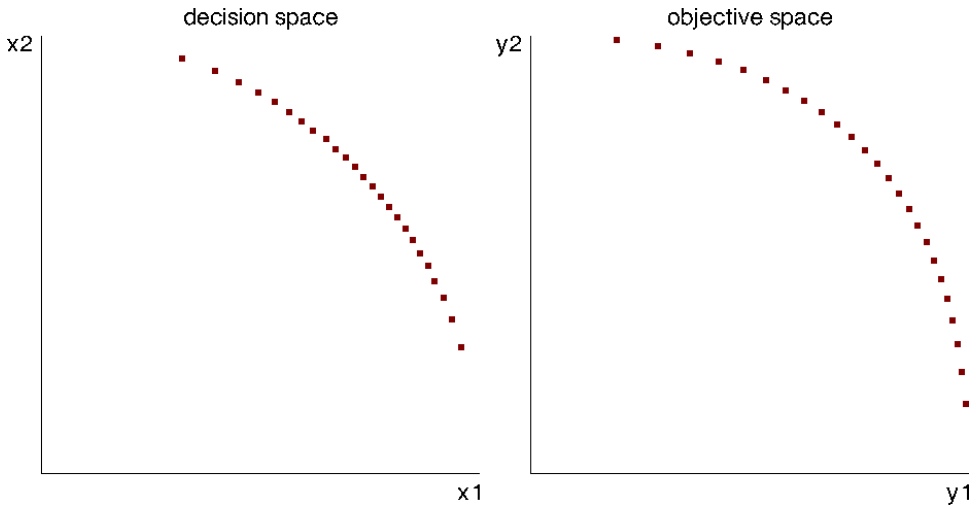

Figure 6: A screenshot of the converged population with the MOCOPS algorithm on test problem 2.

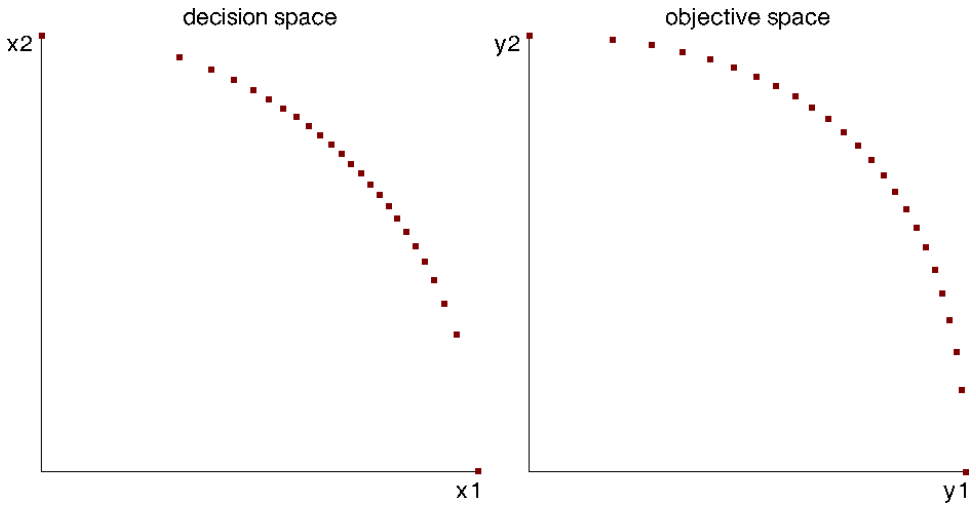

Figure 7: A screenshot of the converged population with the MOGO algorithm on test problem 2.

is deterministic it is more robust[4]. The MOCOPS algorithm turns out to converge slightly slower with adaptive mutation rate if the amount of iterations increase.

[4]Aside from the initialization, the MOGO algorithm is deterministic. This in contrary to the MOCOPS algorithm which is stochastic at every single iteration.

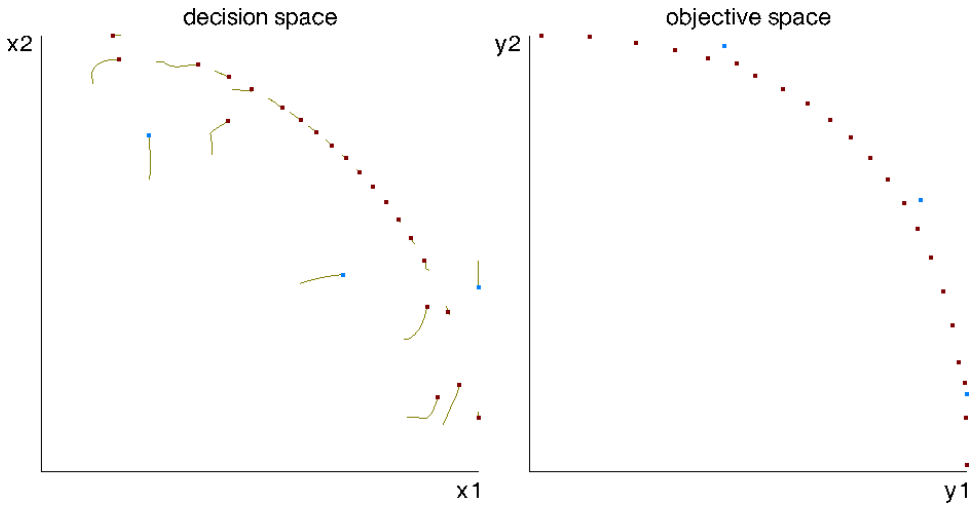

Figure 8: A screenshot showing the previous locations of the particles as trails using the MOGO algorithm.

5 Conclusion

It turns out both the algorithms perform well on the test problems. However, the MOGO algorithm outperforms the MOCOPS algorithm. This especially holds with a big population. The MOGO algorithm tends to get a nearly perfect diversification after a small amount of iterations. For future research it will be interesting to see how the MOCOPS will perform when the adaptive mutation is performed individually instead of globally. Experiments with more challenging test problems will also be interesting for future research. Test problems with for example local minima and more than 2 objective function will reveal more about the usefulness and robustness of the algorithms.

References

[1] Fulya Altiparmak, Mitsuo Gen, Lin Lin, and Turan Paksoy. "A genetic algorithm approach for multi-objective optimization of supply chain networks". In: *Computers & Industrial Engineering* 51.1 (2006), pp. 196 –215. ISSN: 0360-8352. DOI: http://dx.doi.org/10.1016/j.cie.2006.07.011.

[2] Anne Auger. "Benchmarking the (1+1) Evolution Strategy with One-fifth Success Rule on the BBOB-2009 Function Testbed". In: *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. GECCO '09. Montreal, Québec, Canada: ACM, 2009, pp. 2447–2452. DOI: 10.1145/1570256.1570342.

[3] Thomas Bäck and Hans-Paul Schwefel. "An overview of evolutionary algorithms for parameter optimization". In: *Evolutionary computation* 1.1 (1993), pp. 1–23.

[4] Ricardo Cabello. *three.js - Javascript 3D library*. English. 2015. URL: http://threejs.org/ (visited on 01/24/2015).

[5] Massimiliano Caramia and Paolo Dell'Olmo. *Multi-objective management in freight logistics*. Springer, 2008. DOI: 10.1007/978-1-84800-382-8.

[6] Carlos A Coello Coello and Gary B Lamont. *Applications of multi-objective evolutionary algorithms*. Vol. 1. World Scientific, 2004.

[7] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II". In: *Lecture notes in computer science* 1917 (2000), pp. 849–858.

[8] Michael Emmerich. "Multi-attribute decision analysis". English. 2014. URL: `http://www.liacs.nl/~emmerich/moda2014-multi-attribute-decision-analysis.pdf` (visited on 01/20/2015).

[9] Michael Emmerich. "Order and Dominance". English. 2014. URL: `http://www.liacs.nl/~emmerich/moda2014-order-and-dominance.pdf` (visited on 02/04/2015).

[10] Michael Emmerich, Nicola Beume, and Boris Naujoks. "An EMO algorithm using the hypervolume measure as selection criterion". In: *Evolutionary Multi-Criterion Optimization*. Springer. 2005, pp. 62–76.

[11] Michael Emmerich and André Deutz. "Multicriteria Optimization and Decision Making". English. 2015. URL: `http://www.liacs.nl/~emmerich/MODAReader20140312.pdf` (visited on 02/09/2015).

[12] Michael Emmerich and André Deutz. "Time Complexity and Zeros of the Hypervolume Indicator Gradient Field". English. In: *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III*. Ed. by Oliver Schuetze, Carlos A. Coello Coello, Alexandru-Adrian Tantar, Emilia Tantar, Pascal Bouvry, Pierre Del Moral, and Pierrick Legrand. Vol. 500. Studies in Computational Intelligence. Springer International Publishing, 2014, pp. 169–193. DOI: `10.1007/978-3-319-01460-9_8`.

[13] Mark Fleischer. "The measure of Pareto optima applications to multi-objective metaheuristics". In: *Evolutionary multi-criterion optimization*. Springer. 2003, pp. 519–533.

[14] Sanaz Mostaghim, Jürgen Branke, and Hartmut Schmeck. "Multi-objective Particle Swarm Optimization on Computer Grids". In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. GECCO '07. London, England: ACM, 2007, pp. 869–875. DOI: `10.1145/1276958.1277127`.

[15] Christos A Nicolaou, Nathan Brown, and Constantinos S Pattichis. "Molecular optimization using computational multi-objective methods". In: *Current Opinion in Drug Discovery and Development* 10.3 (2007), p. 316.

[16] V Pareto and Manuale di Economia Politica. *Societa Editrice Libraria, Milano, Italy, 1906. Translated into English by AS Schwier as Manual of Political Economy*. 1971.

[17] Gade Pandu Rangaiah. *Multi-objective optimization: techniques and applications in chemical engineering*. Vol. 1. World Scientific, 2008.

[18] Wilco Verhoef. *The interactive multi-objective optimization application*. English. 2015. URL: `http://wilco.verhoef.nu/projects/moo`.

Appendices

Manual of the application

The application and code is available online [18]. Using the application is straightforward and does not need any installation or configuration. The application can be started with any modern browser. The application is shown in Figure 9.

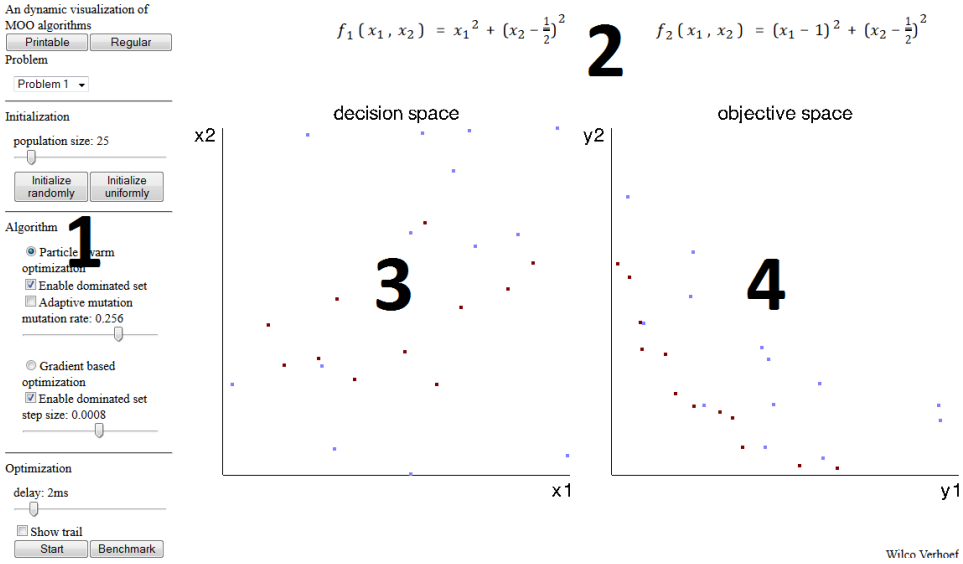

Figure 9: A screenshot of the interactive multi-objective optimization application.

1. This is the sidebar with the interaction. The sidebar is shown in detail in Figure 10.

   (a) Pressing the *Printable* button sets the background color to white. Pressing the *Regular* button will set the color scheme regular again.

   (b) In the problem section you can choose a test problem.

   (c) In the initialization section you can select the population size. You can initialize the population with the set population size by pressing one of the buttons. Pressing the *Initialize randomly* button will position the particles at random in the decision space. Pressing the *Initialize uniformly* button will try to position the particles uniformly in the decision space.

   (d) In the algorithm section you can choose the optimization algorithm by pressing *Particle swarm optimization* or *Gradient based optimization*. Deselecting the *Enable dominated set* will enable the use of the whole population. Pressing *Adaptive mutation* makes the MOCOPS algorithm use the $1/5^{th}$ succes rule. The mutation rate for the MOCOPS algorithm can be adjusted by using the slider. The step size of the MOGO can be adjusted similarly with the other slider.

   (e) In the optimization section you can select how many milliseconds delay every iteration should have using the slider. A bigger delay can make the dynamics of the algorithm clearer. The button *Start* will start the selected algorithm. Pressing the button *Stop* will stop the algorithm again.

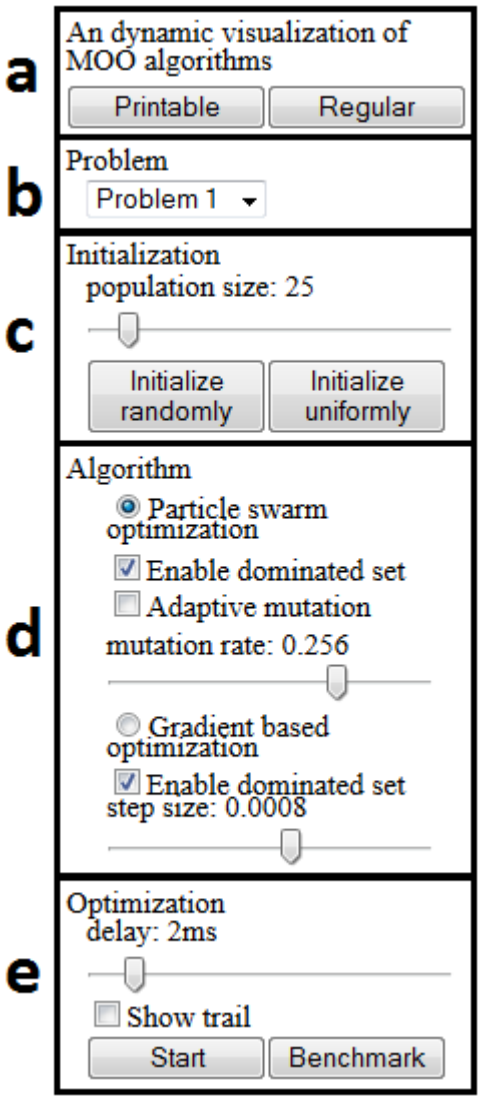

Figure 10: A screenshot of the application capturing the sidebar.

The *Benchmark* button will run some benchmarks and outputs the statistics in the browser console. In most browsers the console is accessible by pressing `F12`. The application automatically adapts to the window size. Full screen mode is available with `F11`.

2. The objective functions of the chosen test problem are shown here.

3. This graph is the decision space. The efficient set of the population is displayed red. The particles which are dominated by the particles in the efficient set are blue.

4. This graph is the objective space. The Pareto set of the population is displayed red. The particles which are dominated by the Pareto set are blue.

Code directory

The code directory contains the following files:

- `index.html`
  The *HTML* file specifying the *JavaScript* files and the `stylesheet.cc`. Open this file with a modern browser to start the application. The other files are dependencies and should be in the same directory.
- `stylesheet.css`
  This *CSS* file declares the layout of the page.
- `main.js`
  This file instantiates the `Controller` class from `controller.js`.
- `controller.js`
  This file instantiates the `Interface` class from `interface.js`.
- `interface.js`
  This file contains the class `Interface` and instantiates `Problem` from `problem.js`, `Optimizer` from `optimizer.js` and `Graphics` from `graphics.js` successively.
- `optimizer.js`
  This file contains the class `Optimizer`. `Optimizer` contains the optimization algorithms and instantiates `Individual` from `individual.js`. `Optimizer` also uses the `Matrix` class from `matrix.js`.
- `individual.js`
  This file contains the class `Individual`.
- `problem.js`
  This file contains the classes defining the test problems.
- `graphics.js`
  This file will visualize the graphs using `three.min.js` and `typeface.js`.
- `matrix.js`
  This file contains the `Matrix` class, used for matrix multiplications between the jacobian matrix and the hypervolume derivative.
- `benchmark.js`
  This file contains the `Benchmark` class, which can be used for benchmarking purposes.
- `typeface.js`
  This defines the typeface to be used in the canvas as labels of the graph.
- `three.min.js`
  `three.js` is a graphics engine for the web [4].