Martin Zaefferer, Boris Naujoks, Thomas Bartz-Beielstein

A Gentle Introduction to Multi-Criteria Optimization with SPOT

Schriftenreihe CIplus, Band 1/2013

Herausgeber: T. Bartz-Beielstein, W. Konen, H. Stenzel, B. Naujoks

Fachhochschule Köln
Cologne University of Applied Sciences

A Gentle Introduction to Multi-Criteria Optimization with SPOT

Martin Zaefferer, Boris Naujoks, and Thomas Bartz-Beielstein

Faculty for Computer and Engineering Sciences
Cologne University of Applied Sciences, 51643 Gummersbach, Germany
`firstname.lastname@fh-koeln.de`

Schriftenreihe CIplus
TR 1/2013. ISSN 2194-2870

Abstract. Multi-criteria optimization has gained increasing attention during the last decades. This article exemplifies multi-criteria features, which are implemented in the statistical software package SPOT. It describes related software packages such as moo and emoa and gives a comprehensive introduction to simple multi criteria optimization tasks. Several hands-on examples are used for illustration. The article is well-suited as a starting point for performing multi-criteria optimization tasks with SPOT.

1 Multi Criteria Optimization: A Simple Example

Optimization in general is fully integrated into todays life. We decide for the best prize when comparing similar groceries or choose the best quality when comparing different jackets. However, such decisions are generally more difficult, more complex than just deciding either for the best price or the best quality. Some sort of compromise solution might be preferable. This is where multi-criteria optimization comes into play.

Of course, the same holds for industrial processes. Input material has to be selected and the process itself can be steered in different directions via controlling some input variables. All these decisions will influence the quality of the final product, whereas quality can be expressed in form of different criteria or objectives (e.g., price, robustness, speed). As a consequence, a decision regarding the above mentioned input parameters can rarely be made without considering multiple criteria at once.

There are some aspects that make such decisions more complicated than classical single objective problems. The most apparent one is the loss of total order. Comparing two products or processes just based on their price is easy. The cheaper is the better. If quality comes into play, life is much harder. Several products might exist which represent different compromises between quality and price. For example, let us consider the choice of selecting a car. Car no. 1 may be the cheapest, but also the one with the highest milage, whereas car no. 2 is the one with the least milage but also most expensive one. There may exist several more cars which represent different compromises between milage and price. An example of that situation is represented in Fig. 1.

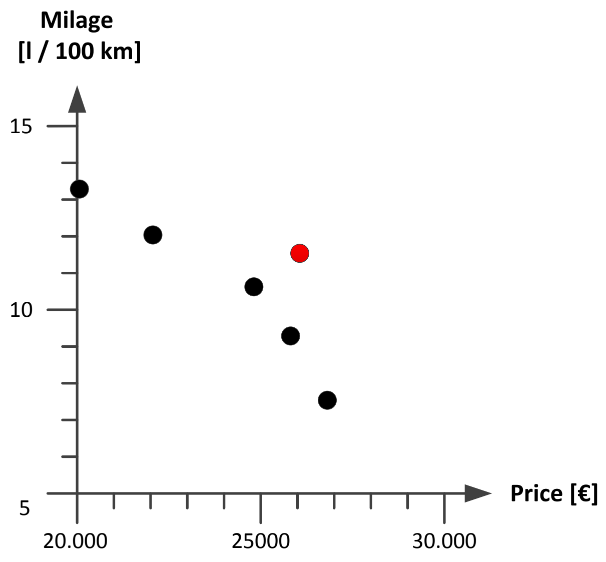


Fig. 1: Graphical comparison of 6 cars, regarding price and milage.

These solutions cannot be sorted by their relative quality to each other since the two goals (milage and price) are conflicting. The total order of solutions is lost. Only the car marked by the red dot is clearly worse, because there exist cars which are better with regards to both objectives.

Of course, things get more difficult (and even more computationally expensive) if more than just two criteria have to be considered. For example, one could consider additional criteria like environmental sustainability or safety of the car. At the same time, this decision process can be made regarding different input parameters. For the consumer, the choice that affects the different criteria is merely that of the six cars available. For the producer of the cars, the cars can be defined by different parameters like their material, shape or type of motor.

The field of Multi Criteria Optimization (MCO) deals with that type of problems. The Sequential Parameter Optimization Toolbox offers methods from that field, which employ surrogate models to speed up the optimization progress. The following Sec. 2 introduces the field of MCO as well as the nomenclature. Afterwards, Sec. 3 presents the programming environment R and some basic tools for MCO in that environment. To give an easy to use example, Sec. 4 introduces a simple MCO test function, and shows how to optimize this with classical MCO algorithms. The very same test functions is optimized with the Sequential Parameter Optimization Toolbox SPOT in Sec. 5. To give more details on the usage of SPOT additional features are shown in Sec. 6. Finally, the reader can test his understanding of the presented information in Sec. 7.

2 Short overview on MCO

Considering only one objective in applied optimization is a simplification that does not mirror the complexity of the underlying application in most (or almost all) cases. Normally, more than one and up to hundreds or thousands of objectives $f_1, \dots, f_n$ need to be considered. The most common way to deal with multiple objectives appears to be aggregation, e.g., to a weighted sum $f(x) = \sum_i w_i f_i(x)$. In contrast, multi-criteria or multiobjective optimization (MCO) offers a different way to handle multiple objectives in a more principled, maybe more effective way.

In MCO, a well-known concept is Pareto dominance, i.e., a solution x from a decision space A dominates another solution $y \in A$ iff x is better in at least one dimension of the decision space and not worse in all the others. More formally and considering minimization, this reads ($A \subset \mathbb{R}^n$, $i, j \in \{1, \dots, n\}$):

$$\begin{aligned} x \prec y \text{ iff } \forall i: \quad & f_i(x) \leq f_i(y) \quad \wedge \\ \exists j: \quad & f_j(x) < f_j(y). \end{aligned}$$

If a solution x is not dominated by any other solution in the search space (or generated by the algorithm), it is said to be *nondominated*, i.e.

$$\forall y \in A: \quad y \nprec x.$$

This concept allows for ranking sets of solutions in the multi-dimensional objective space. As a consequence, MCO algorithms aim for a set A^* of solutions with the property that every two solutions x and y from A^* are mutually non-dominated, i.e.

$$y \nprec x \quad \wedge \quad x \nprec y.$$

A set $\{x \in A \mid \nexists y \in A: \quad y \prec x\}$ of such solutions is also called a *Pareto-front.*

Coming back to the car example from above, all but the car marked in red in Fig. 1 are non-dominated, and thus represent the Pareto-front. The red point is dominated by two other dots.

2.1 Evolutionary Multiobjective Optimization Algorithms

Beside the sheer number of objectives, there is a structural change in the step from one to more objectives. The strict order of solutions in the single-objective objective space turns to a partial order in the multi-objective objective space (with respect to Pareto dominance). This is one of the reasons why population-based optimization approaches like EA are very successful in tackling MCO problems. This structural change also implies that besides Pareto dominance a secondary quality indicator is required for ranking and thus for rank-based selection in an EA. As a result, evolutionary multiobjective optimization algorithms (EMOA) became very popular over the last decade(Deb, 2001a; Coello Coello et al., 2007).

Such algorithms always keep a set of solutions, also called a population, during the optimization run and thus provide a decision maker with a set of comparably good, non-dominated solutions, not just one final best like in single-objective optimization. As a consequence, the decision maker has the chance and the burden to choose the final solution, the one to really implement, from a set of such solutions.

In recent years, the hypervolume indicator turned from a frequently used quality indicator to a well-established selection operator for EMOA (Zitzler & Thiele, 1998; Zitzler, 1999). The hypervolume of a Pareto front A^* is defined as the n-dimensional volume of the space spanned by the Pareto front and a reference point y_{ref}, which needs to be defined by the user:

$$\Lambda\left(\bigcup_{a\in A^*}\{y' \mid a \prec y' \prec y_{\text{ref}}\}\right)$$

with Λ being the Lebesgue measure of the given set.

Maximization of the hypervolume covered by the population results in maximization of possibe trade-offs. Implicitly this covers the traditional goals of convergence of the solution set to the optimal front as well as good solution spread. Most prominent instances of hypervolume based selection MCO algorithms are SMS-EMOA (Beume et al., 2007a), Hyp-E (Bader & Zitzler, 2011), as well as MO-CMA-ES (Igel et al., 2007).

The $(\mu + 1)$ selection mechanism in SMS-EMOA (cf. Alg. 1) provides an elegant way to enlarge and diminish the population size online. Therefore this algorithm is used for the present study.

Algorithm 1: SMS-EMOA

```
1 $P_0 \leftarrow$ init() ;                         // Initialize $\mu$ individuals randomly
2 $t \leftarrow 0$ ;
3 repeat
4     $q_{t+1} \leftarrow$ generate($P_t$) ;              // Generate offspring by variation
5     $P_{t+1} \leftarrow$ reduce($P_t \cup \{q_{t+1}\}$) ;            // Select new population
6     $t \leftarrow t + 1$
7 until stopcriterium reached;
```

In Algorithm 2

$$\Delta_{\mathcal{S}}(s, \mathcal{R}_I) := \mathcal{S}(\mathcal{R}_I) - \mathcal{S}(\mathcal{R}_I \setminus \{s\}) \quad (1)$$

Algorithm 2: Reduce(Q)

```
1 $\mathcal{R} \leftarrow$ fast-nondominated-sort($Q$) ;      // all $I$ non-dominated fronts of $Q$
2 $r \leftarrow \mathrm{argmin}_{s \in \mathcal{R}_I}[\Delta_\mathcal{S}(s, \mathcal{R}_I)]$ ; // eliminate element with lowest $\Delta_\mathcal{S}(s, \mathcal{R}_I)$
3 $Q' \leftarrow Q \setminus \{r\}$ **return** $Q'$
```

calculates the sole contribution of a single solution s to the hypervolume of a Pareto front $\mathcal{R}_I$.

3 MCO in R: Language, Packages, Algorithms

In R, the open source programming environment for statistical computing, some packages implement functionality for MCO[1]. We introduce three packages, which implement several important functions and algorithms. There are more MCO-relevant packages, which are not covered here. We encourage additional research for interested users[2].

3.1 The mco package

The mco package provides several functions concerning MCO with genetic algorithms. Most importantly, it provides an implementation of the NSGA2 algorithm. Several measures of a Pareto fronts quality are available as functions (e.g., Spread, Hypervolume, Epsilon-Indicator). Additionally, several typical MCO test functions are provided (e.g., ZDT1-3).

3.2 The emoa package

The emoa package provides several useful functions, including several measures like hypervolume contribution or crowding distance. It also has several test functions that are not in the mco package.

3.3 The SPOT package

The SPOT package provides mainly the means to do multi objective surrogate model based optimization. Additionally, it uses the emoa package to construct a very basic SMS-EMOA implementation. More information will be given in section 5.

4 A simple Testfunction: ZDT2

The test function used in the following examples is part of the ”mco” R-package[3]. In this example, it is expected that this package is already installed. To install, uncomment the first two lines below. The mco package also contains the NSGA2 algorithm (Deb, 2001b). SPOT should be loaded, too, for the purpose of optimization.

```
> #install.packages("mco")
> #install.packages("SPOT")
> require("mco")
> require(SPOT)
```

[1] R, and all packages mentioned in this document, are available from the CRAN platform: http://cran.r-project.org/

[2] The http://www.rseek.org/ home page might be a first start point for a search on MCO in R.

[3] http://cran.r-project.org/web/packages/mco/

The ZDT functions are MCO testfunctions with scalable decision space dimension n (Zitzler et al., 2000). To guarantee a low number of function evaluations as well as easy visualization of results, both, the dimension of the decision and the objective space are set to 2 in following example. The function ZDT2 $\mathbb{R}^2 \to \mathbb{R}^2$ is defined as:

$$\begin{aligned} f_1 &= x_1 \\ g &= 1 + 9x_2 \\ f_2 &= g\left(1 - \left(\frac{x_1}{g}\right)^2\right) \\ 0 &\leq x_1 \leq 1 \\ 0 &\leq x_2 \leq 1 \end{aligned}$$

Both objectives f_1 and f_2 have to be minimized. In the R-package `mean()`, the ZDT2 function can be called as follows.

```
> x=c(0.5,0.4) #Input vector
> y=zdt2(x)
> print(y)
```

```
[1] 0.500000 4.545652
```

The objective space of the ZDT2 function can be visualized in separate surface plots. Therefore, we will consider the following function: $\mathbb{R}^2 \to \mathbb{R} : (x_1, x_2) \mapsto f_1(x_1, x_2)$. Note, we will use the `apply(x)` command to define an anonymous function.

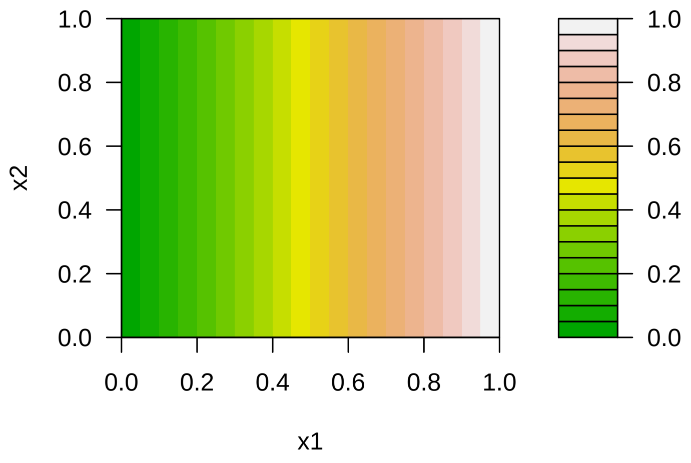
```
> apply(x,1,f)
```

will apply f to rows of x,

```
> apply(x,2,f)
```

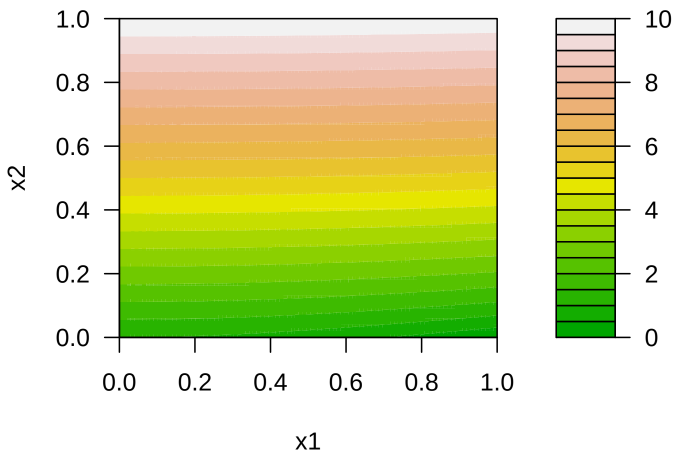
will apply f to columns of x.

```
> ## Define function that calls ZDT2 in a vectorized manner
> ## but returns only first objective:
> ZDT2obj1=function(x)apply(x,1,zdt2)[1,]
> ## Plot first objective in given boundaries
> ## (lo for lower, up for upper)
> spotSurfContour(f=ZDT2obj1,lo=c(0,0),up=c(1,1))
```



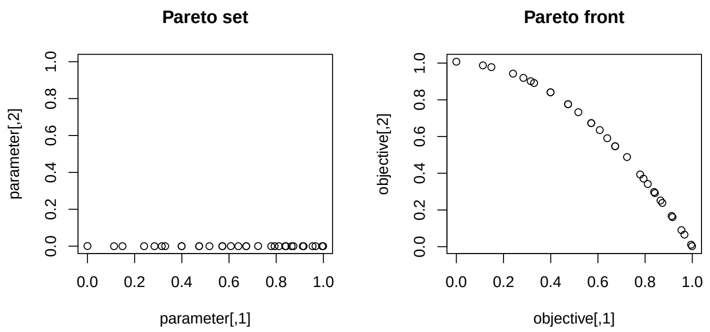
To visualize the impact of the second objective, we will consider the following function: $\mathbb{R}^2 \to \mathbb{R} : (x_1, x_2) \mapsto f_2(x_1, x_2)$:

```
> ## Define function that calls ZDT2 in a vectorized manner,
> ## but returns only second objective:
> ZDT2obj2=function(x)apply(x,1,zdt2)[2,]
> ## Plot second objective in given boundaries
> ## (lo for lower, up for upper)
> spotSurfContour(f=ZDT2obj2,lo=c(0,0),up=c(1,1))
```



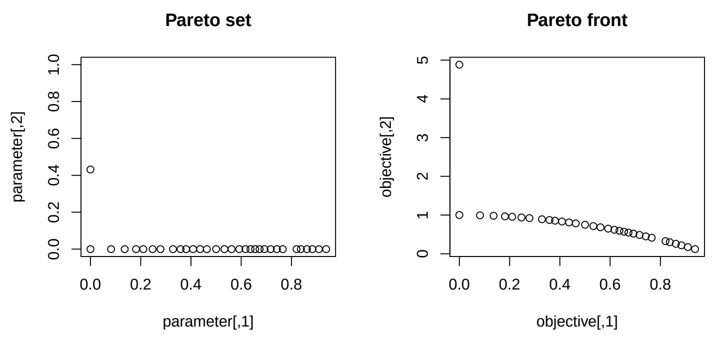
The first objectives landscape is completely linear, with all minima on the x2 axis. The second objective exhibits a slight curvature, with a minimum in the lower right corner of the plot. To find the Pareto front, the ZDT2 function can be optimized with the NSGA2 algorithm in R:

```
> resNSGA<-nsga2(zdt2,2,2,lower.bounds=c(0,0),
+          upper.bounds=c(1,1),popsize=32,generations=30)
> par(mfrow=c(1,2))
> objective=resNSGA$value
> parameter=resNSGA$par
> plot(parameter, main="Pareto set",ylim=c(0,1))
> plot(objective, main="Pareto front")
```



The plots indicate the approximated Pareto front and set. Here, all members of the Pareto set reside on or near to the second parameters lower boundary (e.g. $x_2 \approx 0$). Of course, any other adequate algorithm can be used for optimization as well. The following code calls a straight-forward SMS-EMOA (Beume et al., 2007b), which is shipped with the SPOT package.

```
> resSMS <- spotSmsEmoa(zdt2,
+         lower=c(0,0),
+         upper=c(1,1),
+         control=list(mu=32,maxeval=960))
> par(mfrow=c(1,2))
> objective=t(resSMS$value)
> parameter=t(resSMS$par)
> plot(parameter, main="Pareto set",ylim=c(0,1))
> plot(objective, main="Pareto front")
```



Both, SMS-EMOA and NSGA2 approximate the Pareto front roughly with 960 function evaluations. Using a reference point, the hypervolume can be computed for both Pareto fronts received from SMS-EMOA and NSGA2.

```
> volNSGA<-dominated_hypervolume(t(resNSGA$value),c(11,11))
> volNSGA

[1] 120.2826

> volSMS<-dominated_hypervolume(resSMS$value,c(11,11))
> volSMS

[1] 119.1131
```

5 MCO with SPOT

Up to version 0.1.1550, SPOT could only be applied to single objective optimization problems (Bartz-Beielstein et al., 2005). Despite this, many real world applications feature more than just one quality criterion. Therefore, SPOT was extended to be applicable to MCO. This application of SPOT is referred to as MSPOT.

The basic principle of SPOT remains the same for MCO problems. An initial design is created based on sampling in the decision space, e.g, by a Latin Hypercube Design (LHD) design. This design is evaluated on the target function. Based on this information, one surrogate model is build for each objective of the MCO problem. The models are exploited to suggest promising new design points, which are evaluated on the target function. Based on this new information, a better model can be build. This process is iterated until a termination criterion is reached.

Several surrogate models in SPOT can be used for MCO. To exploit the generated models, two tools are available in MSPOT. The first one is the naive sampling approach, the second one is the utilization of typical MCO algorithms. Both will be demonstrated in this document.

5.1 The naive sampling approach

One way to do multi objective optimization with SPOT, is to exploit the surrogate models by evaluating a large LHD. The "best" points of the design will be suggested for evaluation on the real target function. In this context, "best" is defined to be the lowest dominated sorting rank. If the rank of several points is the same, the hypervolume contribution of each single point will be considered to choose between them. To test this approach with MSPOT, a configuration list is created first:

```
> config=list()
```

The NSGA2 and SMS-EMOA algorithms used 960 function evaluations. This is quite a lot for SPOT, as building the models is rather expensive. In fact, SPOT is mostly used in problems that use only a small but costly number of function evaluations, like algorithm tuning or industrial real world applications. Therefore, the budget for SPOT is restricted to just 40 evaluations:

```
> config$auto.loop.nevals=40
```

Next, the size of the large LHD is specified with 1 000 points.

```
> config$seq.design.size=1000
```

In each sequential SPOT step, a certain number of design points will be evaluated on the target function. In this case, 10 points are chosen for each step.

```
> config$seq.design.new.size=10
```

Since the invoked test function is not noisy, old design points do not have to be reevaluated. As a consequence, repeats in the sequential or initial design are not needed. SPOT's OCBA feature should not be used with deterministic problems either (Chen, 1995) .

```
> config$seq.design.oldBest.size=0
> config$spot.ocba=FALSE
> config$seq.design.maxRepeats = 1
> config$init.design.repeats = 1
```

Two functions have to be chosen in the list. The first function is the surrogate model interface. For multi objective optimization ”spotPredictForrester”, ”spot-PredictMlegp”, ”spotPredictEarth”, ”spotPredictRandomForest” and ”spotPredictLm” are good choices. Since it is fast and robust, the Multivariate Adaptive Regression Spline Model (”spotPredictEarth”) is selected (Friedman, 1991) .

The second function specifies how the surrogate model is optimized. This is NA in this case, because only the sampling approach is used. Alternatively, it can be ”spotModelParetoOptim”, which will be demonstrated in the following section 5.2.

```
> config$seq.predictionModel.func="spotPredictEarth"
> config$seq.predictionOpt.func<-NA
```

Finally, SPOT needs some information about the target function. Its region of interest, in which the parameters are varied, has to be specified, as well as the name of target function itself. Additionally, a reference point can be given.

```
> config$alg.func=zdt2
> config$alg.roi=spotROI(lower=c(0,0),upper=c(1,1))
> config$mco.refPoint=c(11,11)
```

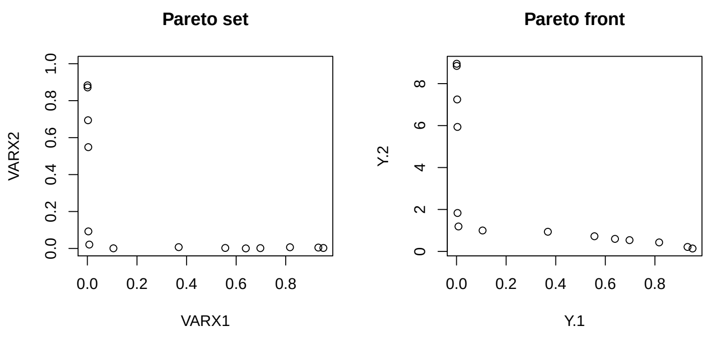
Now, using the above configuration, SPOT can be started.

```
> res1<-spot(spotConfig=config)
```

```
spot.R::spot started
```

The results can for instance be plotted as follows.

```
> par(mfrow=c(1,2))
> objective=res1$mco.val
> parameter=res1$mco.par
> plot(parameter, main="Pareto set",ylim=c(0,1))
> plot(objective, main="Pareto front")
```



5.2 Optimization of the surrogate models

The results observed in the previous section were far from good. Although they were based on a low number of function evaluations, they can be improved by choosing better settings. Therefore, SMS-EMOA is chosen to optimize the surrogate models. Moreover, instead of creating one large design in each step, SMS-EMOA is provided with a large budget.

```
> config$seq.design.size=10
> config$seq.predictionOpt.func="spotModelParetoOptim"
> config$seq.predictionOpt.method="sms-emoa"
> config$seq.predictionOpt.budget=1000
> config$seq.predictionOpt.psize=20
```

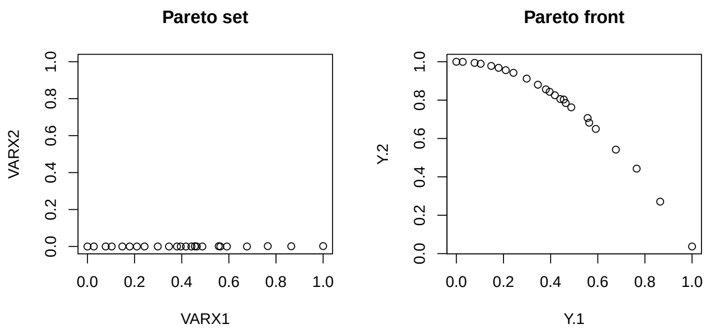
SPOT is started again with the altered configuration.

```
> res2<-spot(spotConfig=config)
```

```
spot.R::spot started
```

Again we can plot the results.

```
> par(mfrow=c(1,2))
> objective=res2$mco.val
> parameter=res2$mco.par
> plot(parameter, main="Pareto set",ylim=c(0,1))
> plot(objective, main="Pareto front")
```



Additionally, the hypervolume of the found fronts can be used to compare the different results generated.

```
> volsamp<-res1$mco.hvolume
> volopt<-res2$mco.hvolume
> volNSGA
```

```
[1] 120.2826
```

```
> volSMS
```

```
[1] 119.1131
```

```
> volsamp
```

```
[1] 118.7766
```

```
> volopt
```

```
[1] 119.9143
```

The hypervolumes generated are in a similar range. Of course, this is only a single experiment and would have to be reevaluated several times with different seeds to gain statistical significance. It has to be noted that MSPOT uses 40 evaluations of the target function only. Moreover, a complete archive of all non dominated solutions is kept for MSPOT. For a fair comparison, this should be compared against a similar archive of SMS-EMOA and NSGA2, instead of comparing it against the final populations.

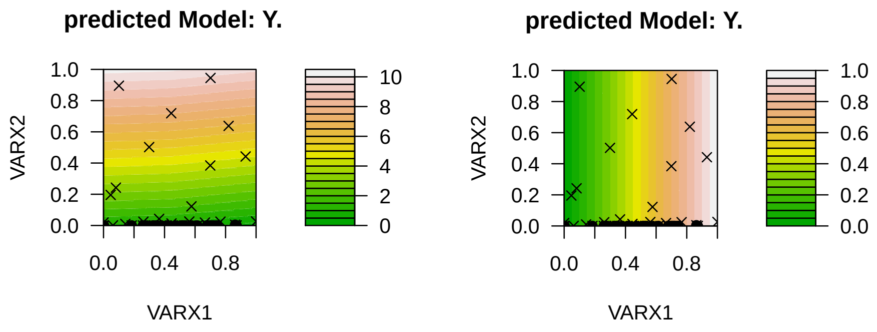
6 MSPOT: Advanced Topics

6.1 Surface plot report

SPOT can do more but just solve the optimization problem. It also provides the means to gain information on the behaviour of the problem landscape. One easy way to do so is to use the surface plot report functions, which are "spotReport3d" and "spotReportContour".

Running these functions yields:

```
> spot(spotConfig=append(list(report.func="spotReportContour",
+           report.interactive=FALSE),res2),spotTask="rep")
```



In these contour plots, crosses indicate a point sampled on the target function and black dots (here: very close to the horizontal axis) indicate points that belong to the approximated Pareto front.

With the above setting of `report.interactive = FALSE`, the surface plots will always show the behavior of the first two optimized parameters. If more than two are optimized, the user can use a simple GUI (see Fig. 2) to choose two parameters for plotting:

```
> spot(spotConfig=append(list(report.func="spotReportContour",
+           report.interactive=TRUE),res2),spotTask="rep")
```

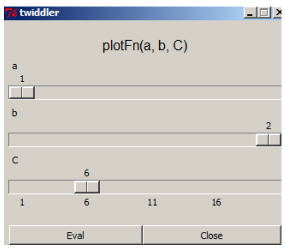


Fig. 2: Twiddler interface

Here, the user can choose which parameters to plot, using the sliders a and b. Slider C specifies which point from the Pareto front should be used to determine the remaining parameters, which are assumed to be constant for the plot. When just two parameters are optimized, this choice has no influence.

6.2 Different models for each objective

In the above examples, each objective was modeled by the same chosen surrogate modeling technique. It could however occur, that the user wishes to model the first objective with a Kriging model and the second with a Random Forest model.

This is possible, by using the following configuration:

```
> config$seq.predictionModel.func="spotPredictMCO"
> config$mco.configs=list(list(seq.predictionModel.func="spotPredictForrester"),
+                  list(seq.predictionModel.func="spotPredictRandomForest"))
> res3<-spot(spotConfig=config)
```

```
spot.R::spot started
```

```
> res3$mco.hvolume
```

```
[1] 110.097
```

Here, the mco.configs element is a meta-list containing several configuration lists. Any settings not specified in `mco.configs` will be taken from the main configuration list (here: config) or set to default values. Instead of selecting different models, one can use the `mco.configs` meta list to choose different settings for two models of the same type.

7 Tasks

Imagine you have two target functions, both with a spherical shape, but with their center at different points. Assume that both functions are defined as:

$$f_1(x_1, x_2) = (x_1 - 2)^2 + (x_2 + 4)^2 \tag{2}$$

$$f_2(x_1, x_2) = (x_1 + 1)^2 + (x_2 - 3)^2 \tag{3}$$

Together, these functions form a two-objective optimization problem, where both f_1 and f_2 are minimized. Please try to solve the following tasks:

1. Implement the optimization problem in R.
2. Plot the target functions separately.
3. Without using the computer any further, what do you expect the Pareto front to look like? Describe and/or draw your idea.
4. Optimize the problem, using the NSGA2 Algorithm in R.
5. Plot Pareto front and Pareto set of the NSGA2 results.
6. Optimize the problem using SPOT.
7. Plot the Pareto front and set, compare with NSGA2 results. Why do results differ? Also, Compare with expectation from question 3.
8. Plot the surrogate models with SPOT and check whether they represent the actual target functions. If not, why?
9. In your opinion, which is the most suited way to solve this specific optimization problem, and why?

8 Solutions

Solutions to the tasks can be requested from the authors via E-Mail.

Bibliography

Bader, J. & Zitzler, E. (2011). HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation*, 19(1), 45–76.

Bartz-Beielstein, T., Lasarczyk, C., & Preuß, M. (2005). Sequential parameter optimization. In B. McKay & others (Eds.), *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1 (pp. 773–780). Piscataway NJ: IEEE Press.

Beume, N., Naujoks, B., & Emmerich, M. (2007a). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3), 1653–1669.

Beume, N., Naujoks, B., & Emmerich, M. (2007b). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3), 1653–1669.

Chen, C. H. (1995). An effective approach to smartly allocate computing budget for discrete event simulation. In *Proceedings of the 34th IEEE Conference on Decision and Control* (pp. 2598–2605).

Coello Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, 2nd edition.

Deb, K. (2001a). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. New York NY: Wiley, 1 edition.

Deb, K. (2001b). *Multi-Objective Optimization using Evolutionary Algorithms*. New York NY: Wiley.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *Ann. Stat.*, 19(1), 1–141.

Igel, C., Hansen, N., & Roth, S. (2007). Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1), 1–28.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2), 173–195.

Zitzler, E. & Thiele, L. (1998). Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben (Ed.), *Parallel Problem Solving from Nature (PPSN V)* (pp. 292–301).: Springer, Berlin.

Kontakt/Impressum

Diese Veröffentlichungen erscheinen im Rahmen der Schriftenreihe "CIplus". Alle Veröffentlichungen dieser Reihe können unter
www.ciplus-research.de
oder unter
http://opus.bsz-bw.de/fhk/index.php?la=de
abgerufen werden.

Köln, Januar 2012

Herausgeber / Editorship

Prof. Dr. Thomas Bartz-Beielstein,
Prof. Dr. Wolfgang Konen,
Prof. Dr. Horst Stenzel,
Dr. Boris Naujoks
Institute of Computer Science,
Faculty of Computer Science and Engineering Science,
Cologne University of Applied Sciences,
Steinmüllerallee 1,
51643 Gummersbach
url: www.ciplus-research.de

NA
Prof. Dr. Thomas Bartz-Beielstein,
Institute of Computer Science,
Faculty of Computer Science and Engineering Science,
Cologne University of Applied Sciences,
Steinmüllerallee 1, 51643 Gummersbach
phone: +49 2261 8196 6391
url: http://www.gm.fh-koeln.de/~bartz/
eMail: thomas.bartz-beielstein@fh-koeln.de

ISSN (online) 2194-2870