MECSOL
Joinville 2017
April 26-28, 2017
Joinville, Santa Catarina, Brazil

MOKO: An Open Source Package for Multi-Objective Optimization with Kriging Surrogates

Adriano Gonçalves dos Passos
Mechanical Engineering Department
Federal University of Technology - Parana, Curitiba, Parana, Brazil
adriano.utfpr@gmail.com

Marco Antônio Luersen
Mechanical Engineering Department
Federal University of Technology - Parana, Curitiba, Parana, Brazil
luersen@utfpr.edu.br

ABSTRACT

Many modern real-world designs rely on the optimization of multiple competing goals. For example, most components designed for the aerospace industry must meet some conflicting expectations. In such applications, low weight, low cost, high reliability, and easy manufacturability, are desirable. In some cases, bounds for these requirements are not clear, and performing a mono-objective constrained optimization might not provide a good landscape of optimal choices. For these cases, finding a set of Pareto optimal designs might give the designer a comprehensive set of options from where to choose the best design. This article shows the main features of an open source package, developed by the authors, to solve constrained multi-objective problems. The package, named `moko` (Multi-Objective Kriging Optimization), was built under the open source programming language R. Popular Kriging based multi-objective optimization strategies, as the expected volume improvement and the weighted expected improvement, are available in the package. In addition, a novel approach based on the exploration using a predicted Pareto front is implemented. The latter approach showed to be more efficient than the remainder ones in some didactic and real-life multi-objective applications performed by the authors with `moko`.

Keywords: Multi-Objective Optimization, Surrogates, Kriging, Open Source Package

1. INTRODUCTION

Multi-objective optimization is a field of interest for many real-world applications. Usually, projects have multiple and conflicting goals and, most of the time, the relationship between the decision space (design variables) and the outcome is highly complex. Around the 2000's, some comprehensive reviews were made that cover many of the basic ideas and challenges related to many-objective optimization [1–3]. Today, there are many efficient algorithms for dealing with multi-objective optimization, most of them being based on evolutionary techniques (evolutionary multi-objective optimization algorithm – EMOA). Some state-of-the-art algorithms can be found in [4–10]. However, even when using such efficient frameworks, if the cost of evaluation of the designs are too high, a surrogate approach is usually used to alleviate the computational burden.

To address this issue, the authors have developed an open-source package based on the Kriging surrogate model. The package, written in R language, presents three optimization algorithms. Two of them are implementations of well-known approaches of the literature, and one of them is a novelty first presented by the authors in [11].

In R, packages are the fundamental units of reproducible codes. They must include reusable R functions, the documentation that describes how to use them, and sample data. For `moko` development, the authors followed the guidelines and good practices provided by [12]. The work resulted in a package that has been accepted and is available[1] at the Comprehensive R Archive Network (CRAN), the official public repository for R packages.

2. SURROGATE MULTI-OBJECTIVE APPROACHES

As already mentioned, `moko` presents three different Kriging-based multi-objective frameworks, which are discussed in the following subsections. The derivation of the Kriging predictor and the design of experiments (DOE) concept are not covered in this paper. The reader can find a comprehensive mathematical description of these subjects in [13]. The Kriging models are built using the `DiceKriging` R package [14].

2.1 Multi-Objective Efficient Global Optimization (MEGO)

EGO, proposed by [15] for mono-objective optimization, consists in, from an initial set of samples $\mathbf{X}$, builds a Kriging model using the responses of a high-fidelity model, then the algorithm sequentially maximizes the expected improvement (EI) and updates the model at each iteration (including re-estimation of the hyperparameters).

The basic idea of the EI criterion is that by sampling a new point $\mathbf{x}^\star$ the results will be improved by $y_{\min} - y(\mathbf{x}^\star)$ if $y(\mathbf{x}^\star) < y_{\min}$ or 0 otherwise, where $y_{\min}$ is the lowest value of the responses $\mathbf{y}$ obtained so far. Obviously, the value of this improvement is not known in advance because $y(\mathbf{x}^\star)$ is unknown. However, the expectation of this improvement can be obtained using the information from the Kriging predictor. The EI criterion has important properties for sequential exploitation and exploration (filling) of the design space: it is null at points already visited (thus preventing searches in well-known regions and increasing the possibility of convergence); and at all other points it is positive and its magnitude increases with predicted variance (favoring searches in unexplored regions) and decreases with the predicted mean (favoring searches in regions with low predicted values).

The EGO algorithm can be easily adapted to a multi-objective framework by scalarizing the objective vector into a single function [16]. The constrains of the optimization problem can be considered simply by building independent metamodels for each constraint and multiplying the EI of the composed objective function by the probability of each constraint to be met [17]. The MEGO algorithm can be summarized as follows:

1. Generate an initial DOE $\mathbf{X}$ using an optimized *Latin hypercube*;
2. Evaluate $\mathbf{X}$ using high-fidelity models and store responses of $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ and $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ for the m-objectives and p-constraints;
3. **while** computational budget not exhausted **do**:
   (a) Normalize the responses to fit in a hypercube of size $[0,1]^m$;
   (b) For each $\mathbf{x} \in \mathbf{X}$ compute a scalar quantity by making $f_\lambda = \max_{i=1}^{m}(\lambda_i f_i) + \rho \sum_{(i=1)}^{m} \lambda_i f_i$, where λ is drawn uniformly at random from a set of evenly distributed unit vectors and ρ is an arbitrary small value which we set to 0.05;

[1] `https://CRAN.R-project.org/package=moko`

   (c) Build Kriging models for f_λ and for the constraints $\mathbf{g}$;
   (d) Find $\mathbf{x}^\star$ that maximizes the *constrained expected improvement*: $\mathbf{x}^\star = \arg(\max(\mathrm{EI}_C(\mathbf{x})))$;
   (e) Evaluate the "true" values of $\mathbf{f}(\mathbf{x}^\star)$ and $\mathbf{g}(\mathbf{x}^\star)$ using high-fidelity models and update the database.

4. **end while**

Here, in order to handle the constrains, the EI is computed using a custom modified version of the `DiceOptim::EI` function provided in [18]. Also, in all approaches, the optimized *Latin hypercube* is built using the R package `lhs` [19].

2.2 Expected Hypervolume Improvement (EHVI – HEGO)

For comparison purposes, the expected hypervolume improvement (EHVI) is used as infill criterion. The EHVI is based on the theory of the hypervolume indicator [20], a metric of dominance of non-dominated the solutions have. This metric consists in the size of the hypervolume fronted by the non-dominated set bounded by reference maximum points. I that sense, the EHVI is the expected improvement at the hypervolume size we would get by sampling a new point $\mathbf{x}^\star$. Here, the EHVI is computed using the R package `GPareto` [21]. The original `GPareto::EHI` function does not account for constrains, so a custom modification was implemented (available in the `moko` package). The algorithm used here (HEGO – Hyper Efficient Global Optimization) is similar to the MEGO and can be summarized as follows:

1. Generate an initial DOE $\mathbf{X}$ using an optimized *Latin hypercube*;
2. Evaluate $\mathbf{X}$ using high-fidelity models and store responses of $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ and $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ for the m-objectives and p-constraints;
3. **while** computational budget not exhausted **do**:
   (a) Normalize the responses to fit in a hypercube of size $[0,1]^m$;
   (b) For each of the m-objectives and p-constraints, build a Kriging model;
   (c) Find $\mathbf{x}^\star$ that maximizes the *constrained expected hypervolume improvement*: $\mathbf{x}^\star = \arg(\max(\mathrm{EHVI}_C(\mathbf{x})))$;
   (d) Evaluate the "true" values of $\mathbf{f}(\mathbf{x}^\star)$ and $\mathbf{g}(\mathbf{x}^\star)$ using high-fidelity models and update the database.
4. **end while**

For this and for the previous approach, a simulated annealing (SA) algorithm is used to maximize the infill criteria. SA algorithm is provided by the R package `GenSA` [22].

2.3 Variance Minimization of the Kriging-predicted Front (VMPF)

The proposed framework, VMPF, is based on the iterative improvement of the predicted Pareto set fidelity. Here, the idea is, from a given initial set of Kriging models (one for each cost or constraint function), to build a Pareto front using the predictor's mean of each model as input functions. From the estimated front $\mathbf{P}$, the design with higher variance $\mathbf{x}^\star$ (i.e. the most isolated on the decision space) has its "true" value evaluated using the high fidelity model. A new set of Kriging models are then rebuilt and the process repeats until a stopping criterion is met. The proposed algorithm can be summarized as follows:

```
1. Generate an initial DOE $\mathbf{X}$ using an optimized *Latin hypercube*;
2. Evaluate $\mathbf{X}$ using high-fidelity models and store responses of $\mathbf{f} = [f_1, f_2, \ldots, f_m]^T$ and $\mathbf{g} = [g_1, g_2, \ldots, g_p]^T$ for the $m$-objectives and $p$-constraints;
3. while computational budget not exhausted do:
    (a) For each of the $m$-objectives and $p$-constraints, build a Kriging model;
    (b) Generate a Pareto set $\mathbf{P}$ using the mean predictor of the Kriging models using a state-of-art multi-objective optimization algorithm (such as NSGA-II);
    (c) Find $\mathbf{x}^\star \in \mathbf{P}$ that maximizes the variance of the Kriging predictor: $\mathbf{x}^\star = \arg(\max(\mathrm{s}_{\mathrm{km}}(\mathbf{x})))$;
    (d) Evaluate the "true" values of $f(\mathbf{x}^\star)$ and $g(\mathbf{x}^\star)$ using high-fidelity models and update the database.
4. end while
```

Here, the NSGA-II implementation used is the one provided by the R package mco [23].

3. MOKO PACKAGE: SOME FEATURES AND USAGE

The installation of moko package can be simply done by downloading it directly from CRAN servers by running:

```r
install.packages('moko')
library(moko)
```

For all three multiobjective optimization algorithms implemented in moko package, the first step is to create a set of initial sampling points (the DOE - design of experiments). This can be done in different ways, however, we suggest using an optimized Latin hypercube. A randomly generated unit Latin hypercube of dimension k with n samples can be created using only base functions of R by running:

```r
doe <- replicate(k, sample(0:n, n))/n
```

Alternatively, an optimized Latin hypercube can be generated using the R package lhs [19] by running:

```r
doe <- lhs::optimumLHS(n, k)
```

The second step (also common for all algorithms) is to evaluate the high fidelity model responses (objectives and constrains) on the DOE points. The way that moko was built, all algorithms expect a single function that returns a vector of responses. If the user has multiple functions for the ojectives and constraints, they must be combined into a single one such as:

```r
fun <- function(x){
    f1 <- objective1(x)
    f2 <- objective2(x)
    f3 <- objective3(x)
    g1 <- constraint1(x)
    g2 <- constraint2(x)
    return(c(f1, f2, f3, g1, g2))
}
```

Also note that `fun` expects, as input argument, a single design vector `x`.

From here, the algorithms diverge by a little, however, the differences are mainly kept under the hood of the optimization algorithm functions. To achieve this, the authors developed an object class named `mkm` (multi-objective Kriging model). This object is a structured list of `km` objects (Kriging models from the `DiceKriging` package). The `mkm` object for a `fun` that returns the first two values as objectives can be created by

```
doe <- lhs::optimumLHS(n, k)
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objective = 1:2))
```

Note that if `modelcontrol$objective == NULL` (i.e. the index of the objectives are not provided) the function will assign that every response of `fun` is an object (i.e. unconstrained multi-objective optimization problem). Also, the user only needs to input the indexes of the objective responses, the algorithm will assume that the remainder functions are constrains.

Now, to run `nsteps` iterations of any implemented method to an object of class `mkm` one can run

```
new_model_MEGO <- MEGO(model, fun, nsteps)
new_model_HEGO <- HEGO(model, fun, nsteps)
new_model_VMPF <- VMPF(model, fun, nsteps)
```

Further details can be found on each function help page. For example, the help page of `VMPF` function can be loaded with

```
?VMPF
```

4. EXAMPLES

Analytical examples selected from the literature are used to access the robustness of the proposed multi-objective optimization framework. These examples are easy-to-compute optimization problems, which allows comparisons with a direct optimization approach performed here by the NSGA-II algorithm.

The quality of the Pareto sets are compared using the inverted generational distance (IGD) metric [24]. The IGD can be defined as

$$\mathrm{IGD}(\mathbf{T},\mathbf{P}) = \frac{1}{|\mathbf{T}|}\sum_{\mathbf{t}\in\mathbf{T}} \min(d(\mathbf{t},\mathbf{p}))_{\mathbf{p}\in\mathbf{P}}, \tag{1}$$

where $\mathbf{T}$ and $\mathbf{P}$ are the true and the current Pareto sets, $|\mathbf{T}|$ is the number of designs in the true Pareto set and $\mathbf{t}$ and $\mathbf{p}$ are normalized vectors of length m of the m-objectives of the true and the actual Pareto sets, respectively, and $d(\,.\,)$ is a distance metric that here is the Manhattan's. Hence, IGD corresponds to the average distance between all designs in the true set and the closest design of the current set. Thus, the lower the IGD value, the better the method is.

4.1 Binh and Korn Problem

The Binh and Korn optimization problem is defined as [25]

$$\begin{aligned}
\text{find: } & \{x_1, x_2 \,|\, 0 \leq x_1 \leq 5 \land 0 \leq x_2 \leq 3\},\\
\text{to minimize: } & f_1 = 4x_1^2 + 4x_2^2\\
\text{and minimize: } & f_2 = (x_1 - 5)^2 + (x_2 - 5)^2,\\
\text{subject to: } & g_1 = (x_1 - 5)^2 + x_2^2 \leq 25,\\
\text{subject to: } & g_2 = (x_1 - 8)^2 + (x_2 + 3)^2 \geq 7.7,
\end{aligned}$$

For this example, the “true” Pareto front is obtained by direct optimization using the NSGA-II algorithm with a population size of 500 over 100 generations. The resulting near-uniform Pareto set for $|\mathbf{T}| = 500$ is shown in Figure 1.

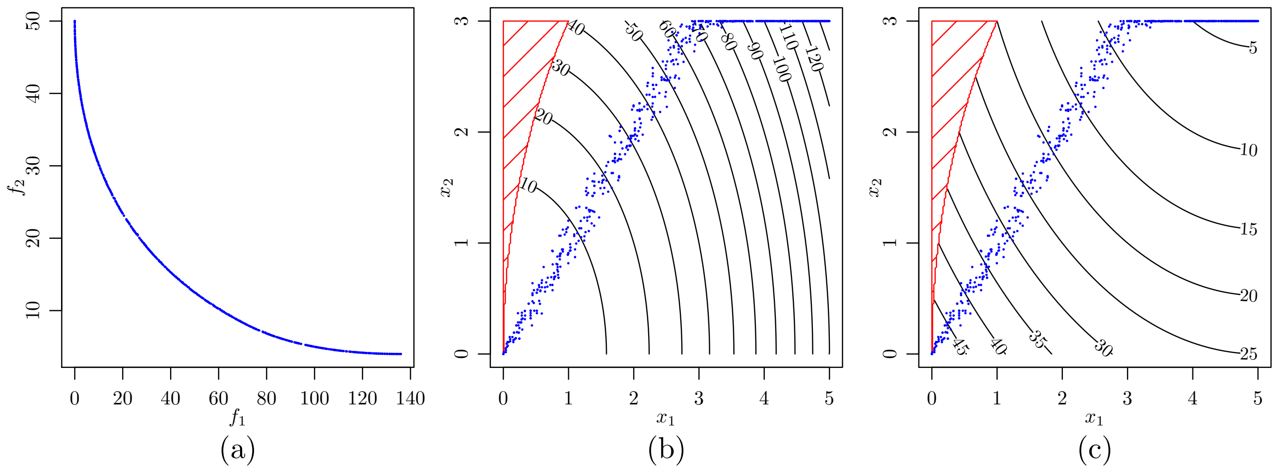

Figure 1. True Pareto front for the Binh and Korn example.

Implementation

Since the Binh and Korn function (and several others) is already implemented in `moko` package, to generate the “true” Pareto front we use the NSGA-II algorithm simply running the following:

```
fun <- function(x) Binh(x * c(5,3))
fun.fn <- function(x) fun(x)[1:2]
fun.cons <- function(x) - fun(x)[-(1:2)]
tpf <- mco::nsga2(fun.fn, lower.bounds = c(0,0), upper.bounds = c(1,1),
                  constraints = fun.cons,
                  idim = 2, odim = 2, cdim = 2, popsize = 500)
```

Note that the domain was scaled to the unit square, thus the input must be multiplied by $\{5,3\}$. This step is optional, however it is highly recommended since it makes the coding much cleaner.

To pass the constraints to the NSGA-II optimizer one must first split the function by creating an objective function and a constraint function. Both functions receive a vector x with size 2 and return vectors also with size 2. Also, `mco::nsga2` consider a violation of the constraint every return that is lesser than 0. This is the reason for the minus sign preceding fun(x) on `fun.cons`.

For this example we chose a computational budget of 60 high-fidelity model evaluations. Among those, 15 are spent building the initial DOE using an optimized Latin hypercube. The remaining budget is used with infill points. The same DOE is used for the three strategies in order to avoid random initialization effects. This routine is implemented as follows:

```
n0 = 15
n1 = 45
d = 2
doe <- lhs::optimumLHS(n = n0, k = d)
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objectives = 1:2))
model_MEGO <- MEGO(model, fun, n1, quiet = FALSE)
model_HEGO <- HEGO(model, fun, n1, quiet = FALSE)
model_VMPF <- VMPF(model, fun, n1, quiet = FALSE)
```

Results

For each strategy, 50 independent runs are performed. Figure 2 shows the IGD histories during the optimization problem. For each run, the IGD values are averaged every 5 evaluations to avoid overcrowding the graph. The resulting data, gathered from the 50 independent runs, are represented with boxes. The mean is highlighted by a bold horizontal line and the whiskers covers ±1.5 IQR (interquartile range).

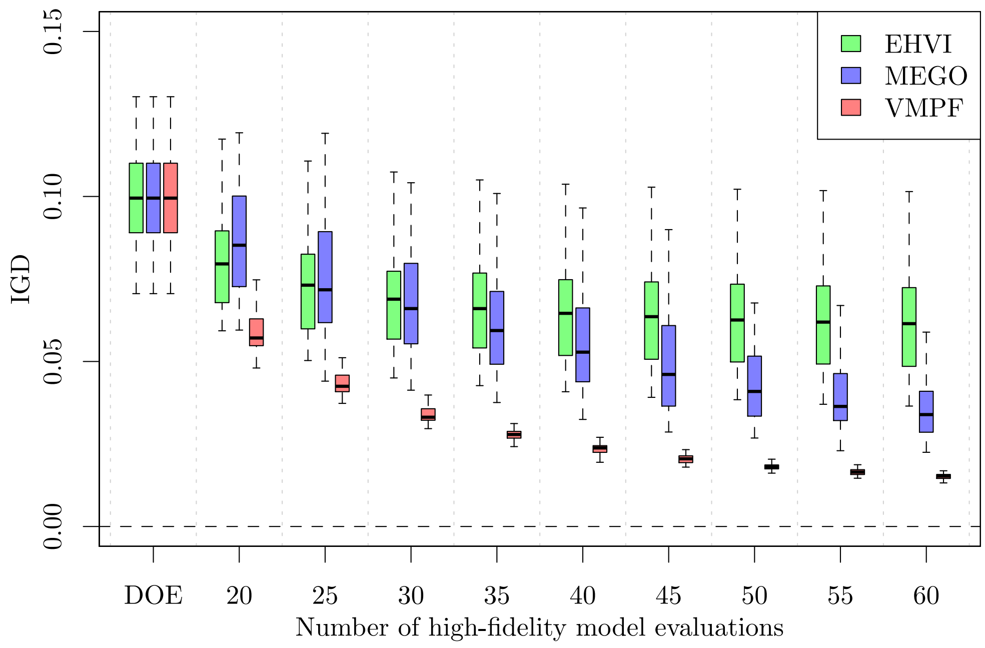

Figure 2. History of 50 independent runs of each proposed framework for the Binh and Korn optimization problem.

Among the three Kriging-based strategies, VMPF presented, in all iterations, the lowest values for the IGD criterion. The final IGD results for the three techniques are: VMPF = 0.015 ± 0.001, MEGO = 0.037 ± 0.012 and HEGO = 0.061 ± 0.016.

4.2 The Nowacki Beam

Based on a problem described by Nowacki [26], the aim is to design a tip loaded cantilever beam for minimum cross-sectional area and minimum bending stress. The beam length is $l = 1500\,\text{mm}$ and it is subject to a tip load force of $F = 5000\,\text{N}$. The cross-section of the beam is rectangular, with breadth b and height h, which are the design variables. The design is constrained by 5 requisites and the optimization problem can be formally defined as the following:

$$
\begin{aligned}
\text{find: } & \{b,h \,|\, 10 \le b \le 50 \wedge 50 \le h \le 250\},\\
\text{to minimize A: } & A = bh,\\
\text{and minimize B: } & \sigma = \frac{6Fl}{b^2h},\\
\text{subject to 1: } & \delta = \frac{12Fl^3}{Ebh^3} \le 5,\\
\text{subject to 2: } & \sigma = \frac{6Fl}{b^2h} \le 240,\\
\text{subject to 3: } & \tau = \frac{3F}{2bh} \le 120,\\
\text{subject to 4: } & \text{AR} = \frac{h}{b} \le 10,\\
\text{subject to 5: } & F_{\text{crit}} = \frac{4}{l^2}\sqrt{G I_T E I_Z \frac{1}{(1-\nu^2)}} \ge 2F,
\end{aligned}
$$

where $I_T = (b^3h + hb^3)/12$ and $I_Z = (b^3h)/12$.

The material used in the original problem is a mild steel with yield stress $\sigma_Y = 240\,\text{MPa}$, Young's modulus $E = 216.62\,\text{GPa}$, Poisson ratio $\nu = 0.27$ and shear modulus $G = 86.65\,\text{GPa}$. For consistency, all values are physically interpreted in the unit system [mm, N, MPa]. For this problem, the "true" Pareto front is obtained by direct optimization using the NSGA-II algorithm with a population size of 500 over 100 generations. The resulting near-uniform Pareto set for $|\mathbf{T}| = 500$ is shown in Figure 3.

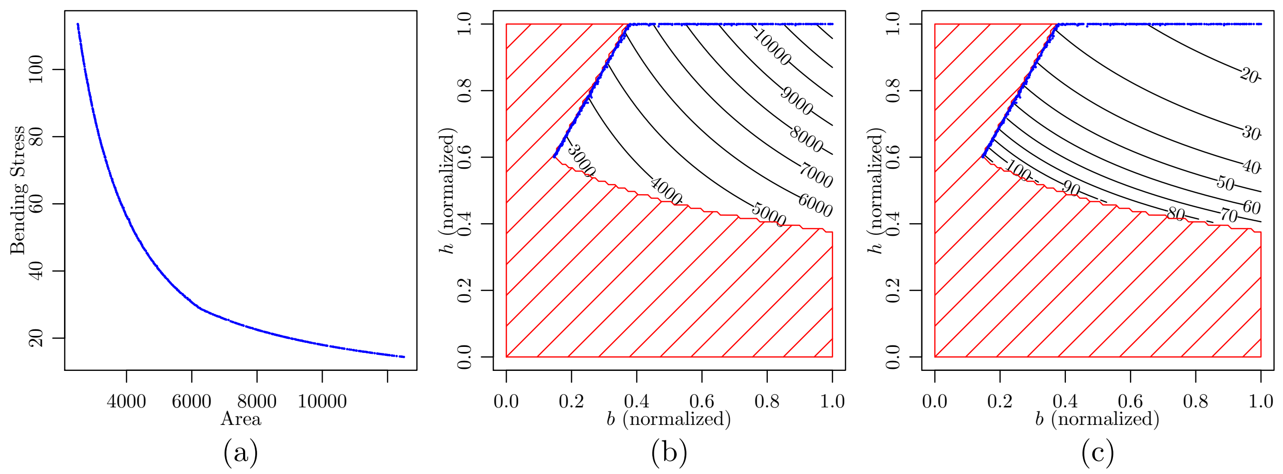

Figure 3. True Pareto front for the Nowacki beam problem.

Implementation

The Nowacki beam problem is also available in `moko` package. By default, the domain of `nowacki_beam` function is defined in the unit square. The `box` parameter can be used to define the real domain where the unit square will be mapped. This can be done with the following code:

```r
library(moko)
fun <- function(x) nowacki_beam(x, box = data.frame(b=c(10,50), h=c(50,250)))
fun.fn <- function(x) fun(x)[1:2]
fun.cons <- function(x) - fun(x)[-(1:2)]
tpf <- mco::nsga2(fun.fn, lower.bounds = c(0,0), upper.bounds = c(1,1),
                  constraints = fun.cons,
                  idim = 2, odim = 2, cdim = 5, popsize = 500)
```

For this example, a computational budget of 80 high-fidelity model evaluations is set. Among those, 20 are spent building the initial DOE using an optimized Latin hypercube. The remaining budget is used with infill points. The same DOE is used for the three strategies in order to avoid random initialization effects. This routine is implemented as follows:

```r
n0 = 20
n1 = 60
d = 2
doe <- lhs::optimumLHS(n = n0, k = d)
res <- t(apply(doe, 1, fun))
model <- mkm(doe, res, modelcontrol = list(objectives = 1:2))
model_MEGO <- MEGO(model, fun, n1, quiet = FALSE)
model_HEGO <- HEGO(model, fun, n1, quiet = FALSE)
model_VMPF <- VMPF(model, fun, n1, quiet = FALSE)
```

Results

For this example, the results are arranged similar to the previous one and can be found in Figure 4.

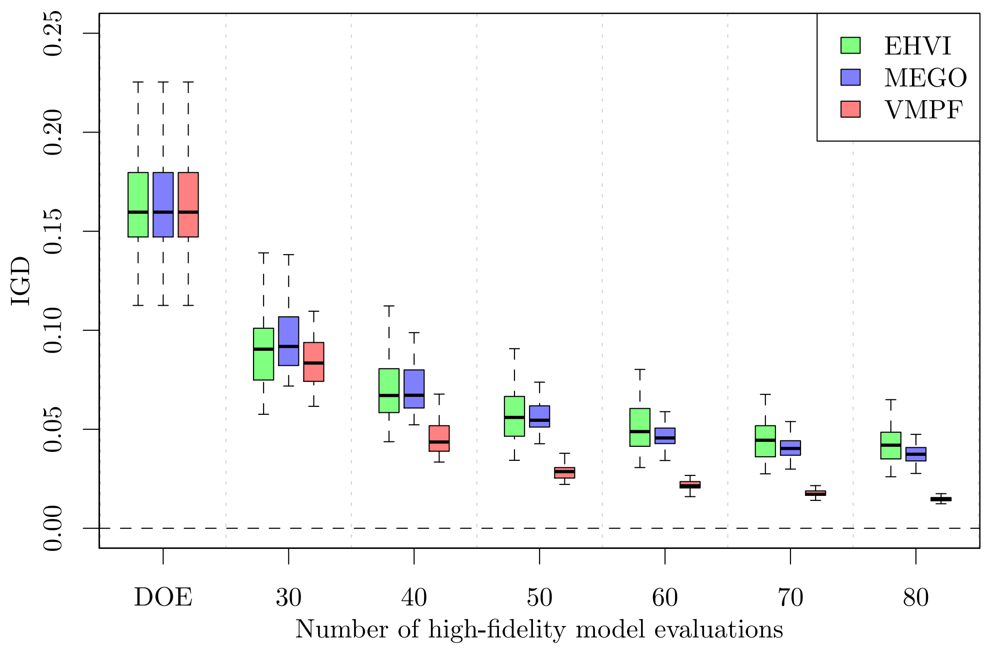

Figure 4. History of 50 independent runs of each proposed framework for the Nowacki beam optimization problem.

Among the three Kriging-based strategies, VMPF presented, in all iterations, the lowest values for the IGD criterion. The final IGD results for the three techniques are: VMPF = 0.015 ± 0.001, MEGO = 0.038 ± 0.005 and HEGO = 0.042 ± 0.010.

5. CONCLUDING REMARKS

In this paper, three Kriging based multi-objective optimization techniques are described (MEGO, EHVI and VMFK). The corresponding implementations can be found in the `moko` package, available on the CRAN servers at `https://CRAN.R-project.org/package=moko`. Two test problems were studied: the Binh and Korn and the Nowacki beam problems. These problems are based on analytical functions, both with two design variables and two objectives. The first problem (Binh and Korn) has two constrains while the Nowacki beam has five. In therms of high-fidelity model calls, all methods showed to be much more efficient than the direct NSGA-II algorithm. Among the Kriging based approaches, the VMFK algorithm significantly outperformed the other two in the cases studied in this paper. The advantages of VMFK showed to be substantial in both performance (IGD mean) and robustness (IGD standard deviation). Also, the authors reckon R language is a powerful easy-to-implement and easy-to-use open source platform for numerical optimization.

REFERENCES

[1] Van Veldhuizen, D. A. and Lamont, G. B. (1998). “Multiobjective evolutionary algorithm research: A history and analysis.” Technical report, Citeseer.

[2] Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications. Ph.D. thesis, Swiss Federal Institute of Technology Zurich.

[3] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2002a). “Scalable multi-objective optimization test problems.” In Evolutionary Computation, 2002. CEC’02. Proceedings of the 2002 Congress on, volume 1, pages 825–830. IEEE.

[4] Corne, D. W., Jerram, N. R., Knowles, J. D., Oates, M. J., *et al.* (2001). “PESA-II: Region-based selection in evolutionary multiobjective optimization.” In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001).

[5] Zitzler, E., Laumanns, M., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L., and Thiele, L. (2001). “SPEA2: Improving the strength Pareto evolutionary algorithm.” Technical report, TIK.

[6] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002b). “A fast and elitist multiobjective genetic algorithm: NSGA-II.” Evolutionary Computation, IEEE Transactions on, 6(2):182–197.

[7] Emmerich, M., Beume, N., and Naujoks, B. (2005). “An EMO algorithm using the hypervolume measure as selection criterion.” In Evolutionary Multi-Criterion Optimization, pages 62–76. Springer.

[8] Beume, N., Naujoks, B., and Emmerich, M. (2007). “SMS-EMOA: Multiobjective selection based on dominated hypervolume.” European Journal of Operational Research, 181(3):1653–1669.

[9] Deb, K. (2014). “Multi-objective optimization.” In Search methodologies, pages 403–449. Springer.

[10] Chen, B., Zeng, W., Lin, Y., and Zhang, D. (2015). “A new local search-based multiobjective optimization algorithm.” Evolutionary Computation, IEEE Transactions on, 19(1):50–73.

[11] Passos, A. G. and Luersen, M. A. (2016). “Kriging-Based Multiobjective Optimization of a Fuselage-Like Composite Section with Curvilinear Fibers.” In EngOpt 2016 - 5th International Conference on Engineering Optimization.

[12] Wickham, H. (2015). R packages. " O’Reilly Media, Inc.".

[13] Forrester, A., Sobester, A., and Keane, A. (2008). Engineering design via surrogate modelling: a practical guide. John Wiley & Sons, Pondicherry, India.

[14] Roustant, O., Ginsbourger, D., and Deville, Y. (2012). “DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization.” Journal of Statistical Software, 51(1):1–55.

[15] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). “Efficient global optimization of expensive black-box functions.” Journal of Global Optimization, 13(4):455–492.

[16] Knowles, J. (2006). “ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems.” Evolutionary Computation, IEEE Transactions on, 10(1):50–66.

[17] Sasena, M. J., Papalambros, P., and Goovaerts, P. (2002). “Exploration of metamodeling sampling criteria for constrained global optimization.” Engineering Optimization, 34(3):263–278.

[18] Ginsbourger, D., Picheny, V., Roustant, O., with contributions by Clément Chevalier, and Wagner, T. (2013). DiceOptim: Kriging-based optimization for computer experiments. R package version 1.4.

[19] Carnell, R. (2012). LHS: Latin Hypercube Samples. R package version 0.10.

[20] Zitzler, E. and Thiele, L. (1998). “Multiobjective optimization using evolutionary algorithms – a comparative case study.” In Parallel problem solving from nature, pages 292–301. Springer.

[21] Binois, M. and Picheny, V. (2016). GPareto: Gaussian Processes for Pareto Front Estimation and Optimization. R package version 1.0.2.

[22] Xiang, Y., Gubian, S., Suomela, B., and Hoeng, J. (2013). “Generalized simulated annealing for global optimization: the GenSA Package.” The R Journal, 5(1):13–29.

[23] Mersmann, O. (2014). MCO: Multiple Criteria Optimization Algorithms and Related Functions. R package version 1.0-15.1.

[24] Shimoyama, K., Jeong, S., and Obayashi, S. (2013). “Kriging-surrogate-based optimization considering expected hypervolume improvement in non-constrained many-objective test problems.” In Evolutionary Computation (CEC), 2013 IEEE Congress on, pages 658–665. IEEE.

[25] Binh, T. T. and Korn, U. (1997). “MOBES: A multiobjective evolution strategy for constrained optimization problems.” In The Third International Conference on Genetic Algorithms (Mendel 97), volume 25, page 27.

[26] Nowacki, H. (1980). “Modelling of design decisions for CAD.” In Computer Aided Design Modelling, Systems Engineering, CAD-Systems, pages 177–223. Springer.

RESPONSIBILITY NOTICE