LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

technische universität dortmund

SFB 876 Providing Information by Resource-Constrained Data Analysis

`mlrMBO`: A Toolbox for Model-Based Optimization of Expensive Black-Box Functions

Bernd Bischl[1] **Jakob Richter[2]** Jakob Bossek[3] Daniel Horn[2] Michel Lang[2]

June 28, 2016

[1]Department of Statistics, LMU Munich

[2]Faculty of Statistics, TU Dortmund University

[3]Westfälische-Wilhelms Universität Münster, Germany

When to use mlrMBO?

When to use `mlrMBO`?

Answer: When `optim(par,` $f(x)$`)` is not enough!

When to use `mlrMBO`?

Answer: When `optim(par,` $f(x)$`)` is not enough!

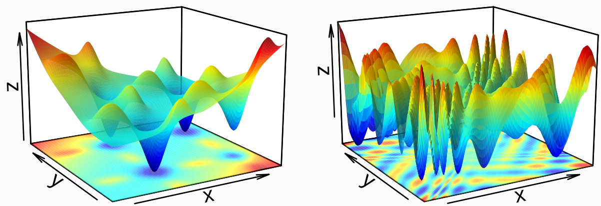
$f(x)$ is *expensive*.

(→ that's why GAs won't work!)

$f(x)$ is not convex.

$f(x)$ is noisy.

`par` is not only numeric.

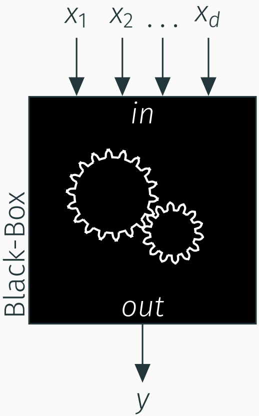


✗ `optim()`

✓ **`mlrMBO`**

Model-Based Optimization

Expensive Black-Box Optimization



$$y = f(\boldsymbol{x}) \ , \quad f : \mathbb{X} \to \mathbb{R}$$

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbb{X}} f(\boldsymbol{x})$$

- y, target value
- $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^d$, domain
- $f(\boldsymbol{x})$ function with considerably long runtime
- Goal: Find optimum $\boldsymbol{x}^*$

Basic Idea

Function evaluations are expensive, so keep number of black-box evaluations low

- Try to predict function values by regression model.
  $\rightarrow$ surrogate model
- Search for points leading to finding the optimum on the surrogate model.
- Update surrogate model with evaluated points.

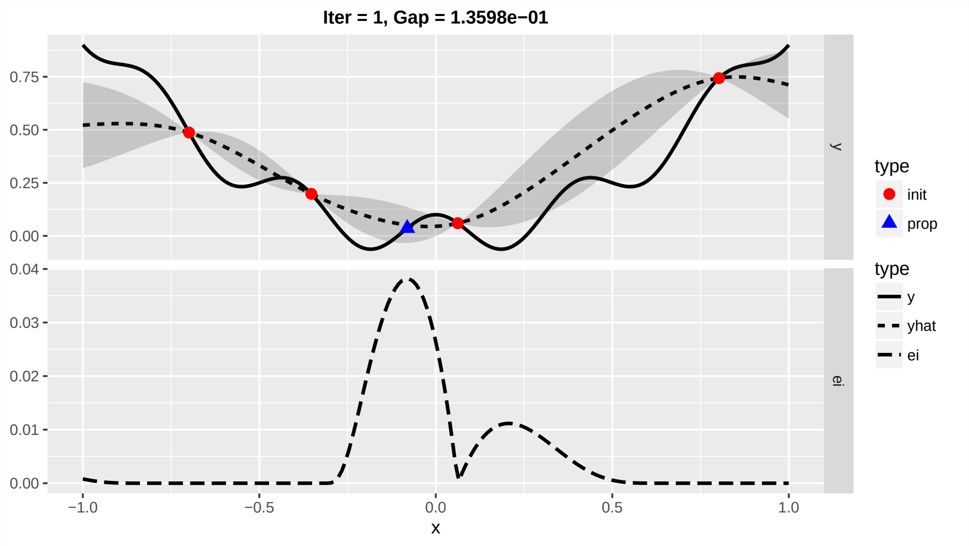
Search mechanism balances exploitation and exploration.

- Just evaluate $\mathbf{x}$ where
  - Predicted function value is low: $\searrow \hat{y}(\mathbf{x})$
  - Uncertainty is high: $\nearrow \hat{s}^2(\mathbf{x})$

  $\Rightarrow$ infill criterion: $Inf(\mathbf{x})$
- Popular choice proposed by Jones et al. (1998):
  Improvement: $I(\mathbf{x}) = max(0, |f(\mathbf{x}^*) - f(\mathbf{x})|)$
  Expected Improvement $EI(\mathbf{x}) = E(I(\mathbf{x}))$

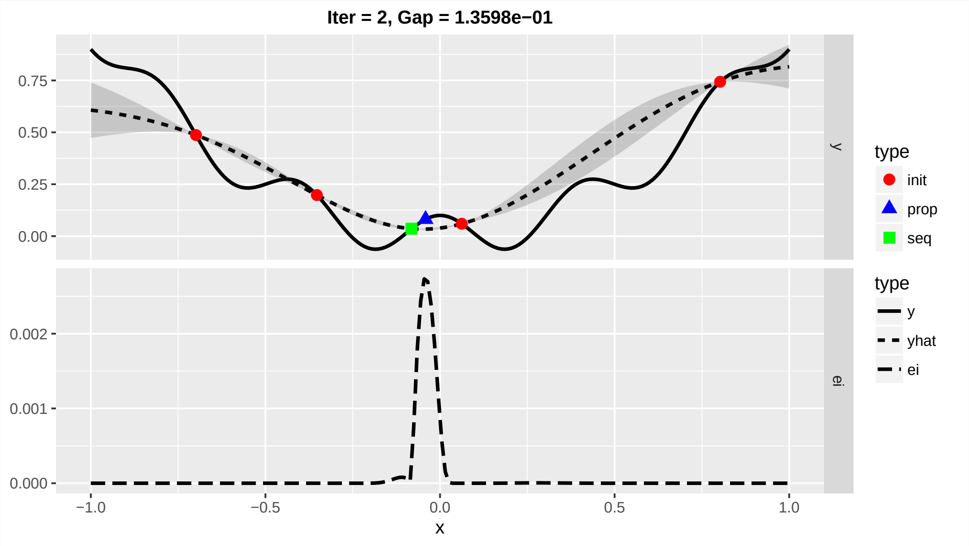
MBO Visualization

Evaluate initial design ● to generate surrogate model $\hat{y}$ - - - - - -, propose new point ▲ based on maximum of *EI* – – – –.

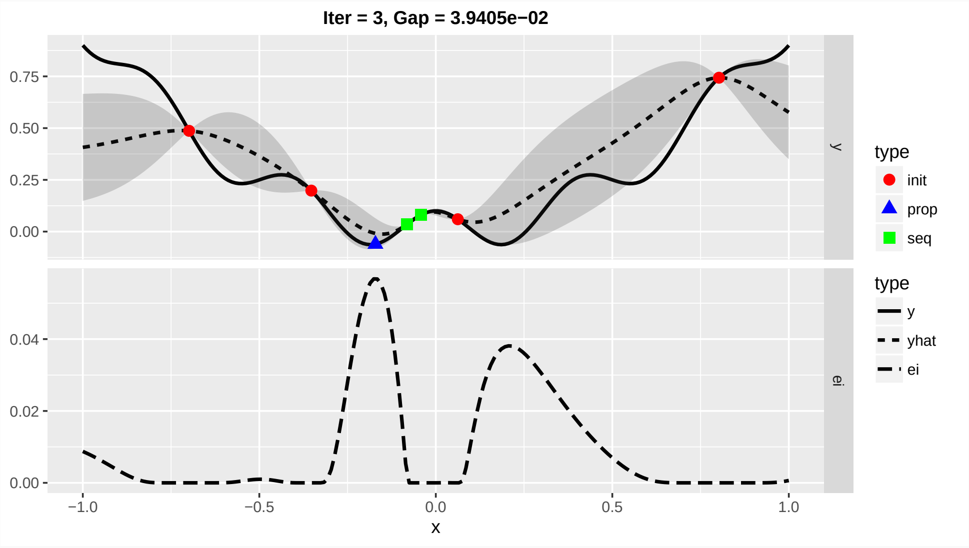


MBO Visualization

Update surrogate model $\hat{y}$ ------ with evaluated point ■, propose new point ▲ based on maximum of *EI* ----.

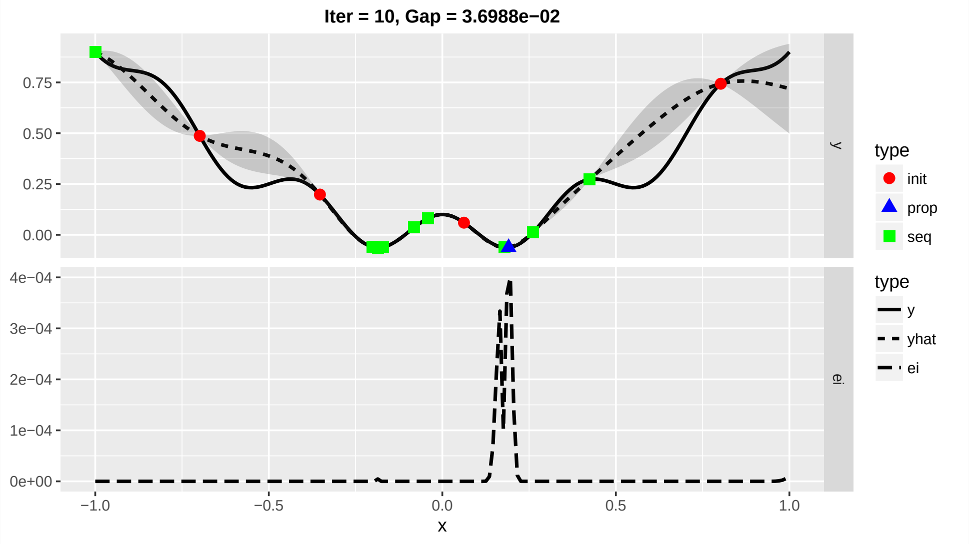


Continue ...

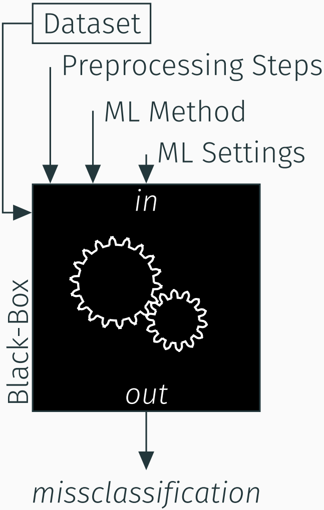

Iter = 3, Gap = 3.9405e−02
0.75
0.50
0.25
0.00
y
0.04
0.02
0.00
ei
−1.0
−0.5
0.0
0.5
1.0
x
type
init
prop
seq
type
y
yhat
ei

MBO Visualization

…until budget is exhausted.



Application

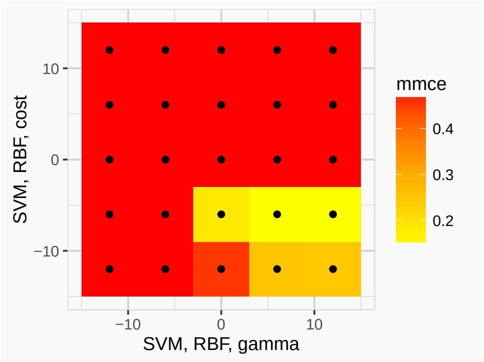


`mlrMBO` can be used for:

- Expensive Black-Box Optimization
- Hyperparameter Tuning for Machine Learning Methods
- Machine Learning Pipeline Configuration
- Algorithm Configuration
- ...

- Still common practice: grid seach
  For a SVM it might look like:
    - $C \in (2^{-12}, 2^{-10}, 2^{-8}, \ldots, 2^{8}, 2^{10}, 2^{12})$
    - $\gamma \in (2^{-12}, 2^{-10}, 2^{-8}, \ldots, 2^{8}, 2^{10}, 2^{12})$
    - Evaluate all $13^2 = 169$ combinations $C \times \gamma$
- Bad beacause:
    - optimum might be ”off the grid”
    - lots of evaluations in bad areas
    - lots of costy evaluations
- How bad? $\hookrightarrow$

Hyperparameter Tuning



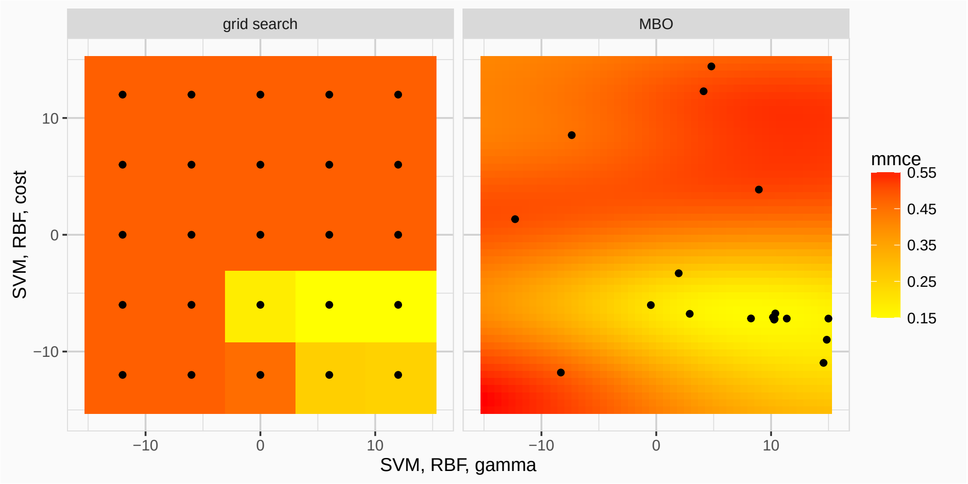
- Because of budget restrictions grid might even be smaller!
- Unpromising area quite big!
- Lots of costy evaluations!

With **`mlrMBO`** it's not hard to do it better! $\hookrightarrow$

Hyperparameter Tuning

```
# Define classification learner and its Hyper Parameter search space
lrn = makeLearner("classif.svm")
ps = makeParamSet(
  makeNumericParam("cost", -15, 15, trafo = function(x) 2^x),
  makeNumericParam("gamma", -15, 15, trafo = function(x) 2^x))
# Define Tuning Problem
mbo.ctrl = makeMBOControl()
mbo.ctrl = setMBOControlTermination(mbo.ctrl, iters = 10)
surrogate.lrn = makeLearner("regr.km", predict.type = "se")
ctrl = mlr::makeTuneControlMBO(learner = surrogate.lrn,
  mbo.control = mbo.ctrl, same.resampling.instance = FALSE)
rdesc = makeResampleDesc("Subsample", iters = 10)
res.mbo = tuneParams(lrn, sonar.task, rdesc, par.set = ps,
  control = ctrl, show.info = FALSE)
```

Grid Search vs. MBO

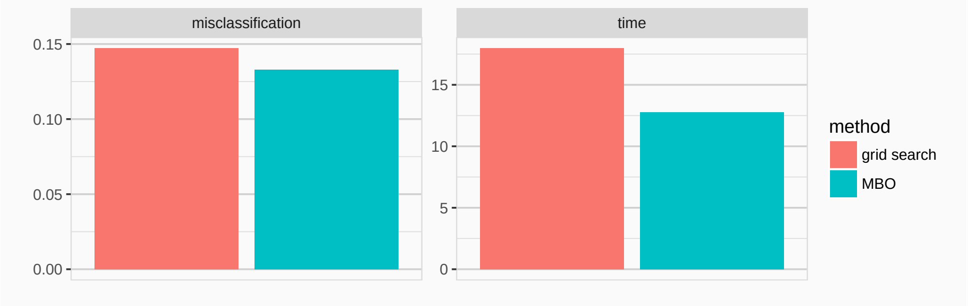


Hyperparameter Tuning

Compare results:

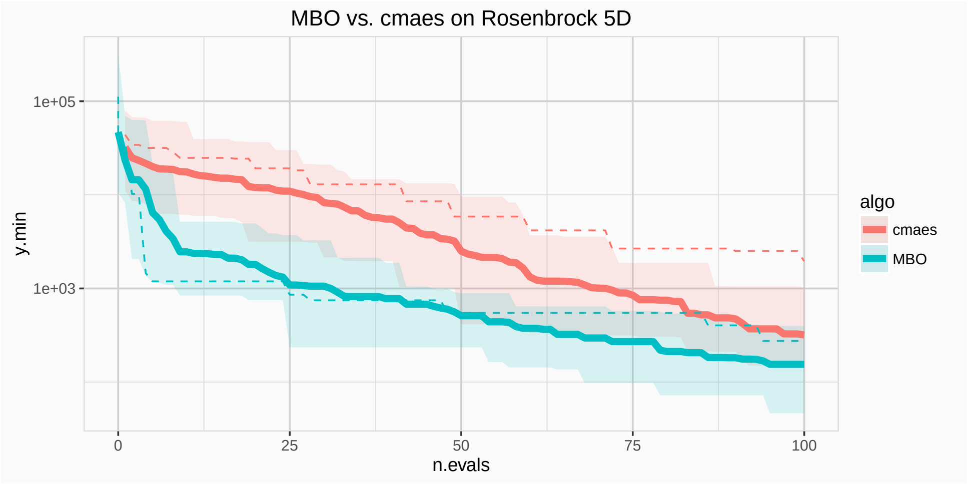
```r
# Grid Tuning Result:
res.grid

## Tune result:
## Op. pars: cost=64; gamma=0.0156
## mmce.test.mean=0.147

# Tuning Costs (Time):
sum(getOptPathExecTimes(res.grid$opt.path))

## [1] 17.967
```

```r
# MBO Tuning Result:
res.mbo

## Tune result:
## Op. pars: cost=1.32e+03; gamma=0.00938
## mmce.test.mean=0.133

# Tuning Costs (Time):
sum(getOptPathExecTimes(res.mbo$opt.path))

## [1] 12.764
```



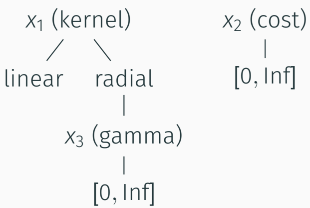
Compare to CMAES

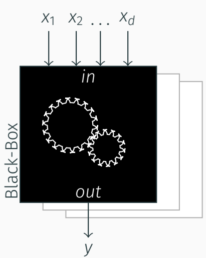
It's not hard to beat grid search! How about a state of the art optimizer?



Extensions

Extensions





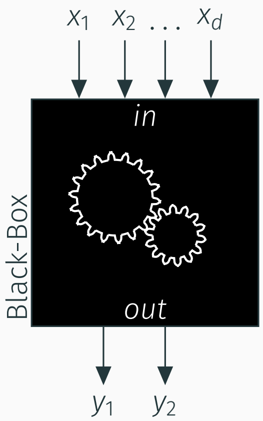
- Different surrogat models to support mixed valued domain $\mathbb{X}$

```r
ps = makeParamSet(
  makeDiscreteParam(id = "kernel",
            values = c("linear", "radial")),
  makeNumericParam(id = "cost"),
  makeNumericParam(id = "gamma",
            requires = quote(kernel!="linear"))
)
```

- Different Infill-Criteria: *mean*, *EI*, *CB*, ...
- Batch proposal for easy parallelization

```r
library(parallelMap)
ctrl = makeMBOControl(
  propose.points = 4L, ...)
# ...
parallelStartMulticore(4)
mbo(...)
parallelStop()
```

Advanced Extensions



$$\min_{\mathbf{x} \in \mathbb{X}} \mathbf{f}(\mathbf{x}) = \mathbf{y} = (y_1, ..., y_m) \text{ with } \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$$

- $\mathbf{y}$ *dominates* $\tilde{\mathbf{y}}$ if

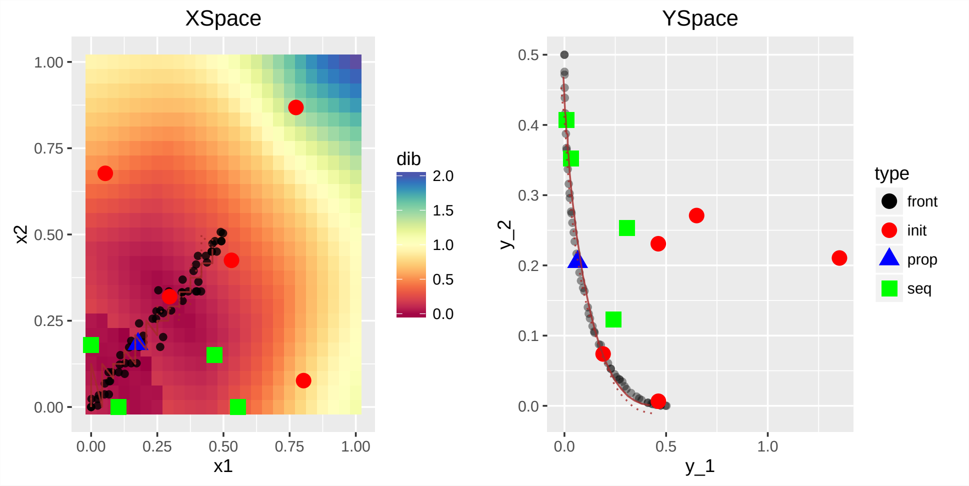
$$\forall i \in \{1, ..., m\} : y_i \leq \tilde{y}_i$$
$$\text{and } \exists i \in \{1, ..., m\} : y_i < \tilde{y}_i$$

- Set of non-dominated solutions:

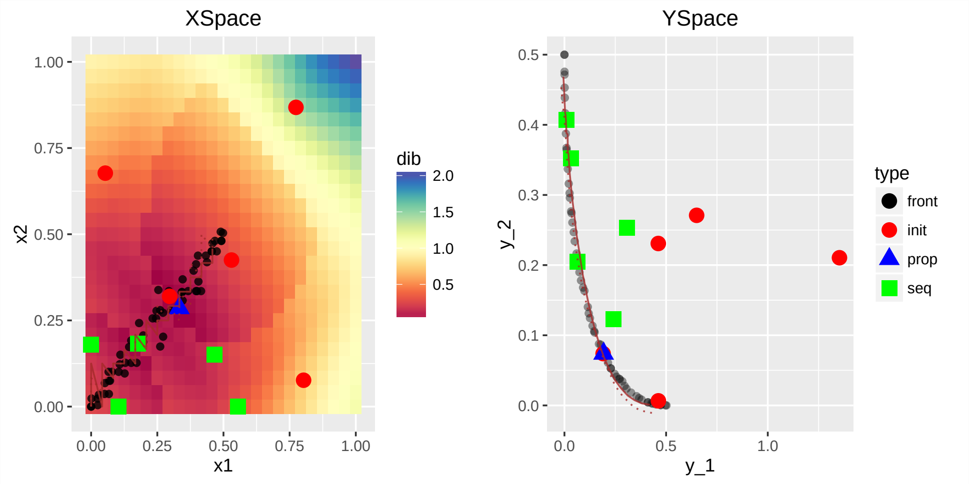
$$\mathcal{X}^* := \{\mathbf{x} \in \mathcal{X} | \nexists \tilde{\mathbf{x}} \in \mathcal{X} : \mathbf{f}(\tilde{\mathbf{x}}) \text{ dominates } \mathbf{f}(\mathbf{x})\}$$

- $\mathcal{X}^*$ is called Pareto set, $\mathbf{f}(\mathcal{X}^*)$ Pareto front
- Goal: Find $\hat{\mathcal{X}}^*$ of non-dominated points that estimates the true set $\mathcal{X}^*$ Different methods for **`mlrMBO`** discussed in Horn et al. (2015).

Pareto Front Optimization



Pareto Front Optimization



Conclusion

Conclusion

Why use `mlrMBO`?

- Effecient model based optimizer
- Powerfool toolbox for a wide varaiaty of set-ups
- Different black box scenarios covered
- Improved exploration of search space within time budget
- **`mlrMBO`** is easy to use!

Outlook

- improve user friendlyness
- improve parallel computation

We use R: Find us on GitHub

- `github.com/mlr-org/mlr`
- `github.com/mlr-org/mlrMBO`

References

Bischl, Bernd et al. (2014). "MOI-MBO: Multiobjective Infill for Parallel Model-Based Optimization". In: *Learning and Intelligent Optimization Conference*. Florida. DOI: 10.1007/978-3-319-09584-4_17.

Horn, Daniel et al. (2015). "Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark". In: *Evolutionary Multi-Criterion Optimization*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello. Vol. 9018. Lecture Notes in Computer Science. Springer International Publishing, pp. 64–78. ISBN: 978-3-319-15933-1.

References II

Jones, Donald R., Matthias Schonlau, and William J. Welch (1998). "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4, pp. 455–492. DOI: 10.1023/A:1008306431147.