Constrained single-objective optimization: combining GPs and Karush Kuhn Tucker conditions

Prof. Dr. Inneke Van Nieuwenhuyse

Data Science Institute, Hasselt University

Research Center for Operations Management, KU Leuven

Joint work with Jack Kleijnen (Tilburg University) and Wim Van Beers

Overview

- (Black box) constrained optimization
- Our approach:
  - Combination of GPR and KKT
  - Infill criterion =EI-KT
- Results on test problems
- Conclusions

(Black box) constrained optimization

We consider the following optimization problem:

$$\boxed{\begin{gathered}\min_{\boldsymbol{x}} w_0(\boldsymbol{x}) \\ w_{h'}(\boldsymbol{x}) \leq c_{h'} \ (h' = 1, \ldots, t-1) \\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}.\end{gathered}}$$

General single-objective *<u>nonlinear constrained</u>* *optimization problem*

- $\boldsymbol{x}$ = (x_1,…x_k) = controls (decision variables, actions)
- $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$: input constraints (box constraints)
- $w_0(\boldsymbol{x})$: objective function
- $w_{h'}(\boldsymbol{x})$: nonlinear constraint functions $(h' = 1, \ldots, t-1)$

Analytically intractable, but you have some "model" (physical experiment, simulation model,…) to accurately observe these → <u>EXPENSIVE</u> calculations (time, cost, resources,…)

(Black box) constrained optimization

Examples:

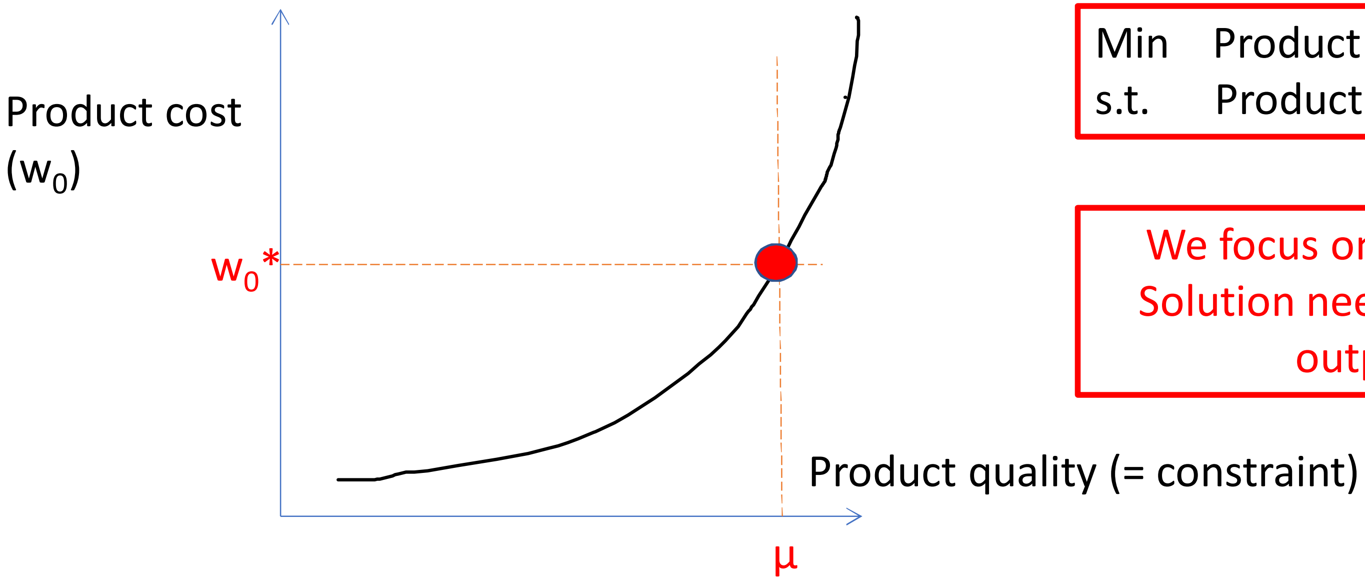
- **Production process**: process parameters → impact on product quality, product cost,…: problem =
  - minimize cost while quality needs to be *above* a minimum threshold level
  - maximize quality while cost needs to stay *below* a maximum threshold level

- **Warehousing**: reorder points and reorder quantities → impact on service level and inventory holding cost: problem =
  - minimize holding cost while service level needs to be *above* a minimum threshold level
  - maximize service level while holding cost needs to stay *below* a maximum threshold level

(Black box) constrained optimization

<u>In many constrained optimization problems, optimal solution makes (at least) 1 output constraint binding!</u>

- Because there is a *trade-off* between the goal output and this output constraint!



Min	Product cost
s.t.	Product quality >= μ

We focus on this type of problem!
Solution needs to lie on (at least 1) output constraint!

Our approach (ML + OR)

Aims specifically at **expensive systems!**

- We approximate expensive system outcomes using machine learning model: Gaussian Process Regression (GPR) (ML element)
  - Smartly choose next decision vector(s) to simulate using infill criterion: Exploits GPR information to estimate how promising new decision vector would be
  - Multiple criteria exist: expected improvement, probability of improvement, differential entropy,…

- We propose a new infill criterion that combines expected improvement (EI) and preferably samples points that are close to or on the constraint(s)! (OR element: **KKT conditions**)

KKT conditions

= *first order necessary conditions* for a solution in nonlinear programming to be optimal

$$\min_{\boldsymbol{x}} w_0(\boldsymbol{x})$$
$$w_{h'}(\boldsymbol{x}) \leq c_{h'} \;\; (h' = 1, \ldots, t-1)$$
$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}.$$

Our focus: Problems with optimal solution on (at least one) binding output constraint

Assumption: all functions are differentiable

$$-\nabla_0(\boldsymbol{x}_*) = \sum_{h'' \in A_\lambda(\boldsymbol{x}_*)} \lambda_{h''}(\boldsymbol{x}_*) \nabla_{h''}(\boldsymbol{x}_*) + \sum_{g'' \in A_\mu(\boldsymbol{x}_*)} \mu_{g''}(\boldsymbol{x}_*) \nabla_{g''}(\boldsymbol{x}_*)$$
with $\lambda_{h''}(\boldsymbol{x}_*) \geq 0$ and $\mu_{g''}(\boldsymbol{x}_*) \geq 0$.

Set of binding input constraints

Set of binding output constraints

KKT conditions

GPR model of a given function also gives us estimate of the gradient of that function (Matlab DACE Toolbox)

$$\boxed{\begin{aligned} -\nabla_0(\boldsymbol{x}_*) &= \sum_{h'' \in A_\lambda(\boldsymbol{x}_*)} \lambda_{h''}(\boldsymbol{x}_*) \nabla_{h''}(\boldsymbol{x}_*) + \sum_{g'' \in A_\mu(\boldsymbol{x}_*)} \mu_{g''}(\boldsymbol{x}_*) \nabla_{g''}(\boldsymbol{x}_*) \\ \text{with } \lambda_{h''}(\boldsymbol{x}_*) &\geq 0 \text{ and } \mu_{g''}(\boldsymbol{x}_*) \geq 0. \end{aligned}} \quad (1)$$

How can we estimate the multipliers? Least-squares regression

$$\widetilde{\boldsymbol{\nu}} = (\boldsymbol{\Delta}'\boldsymbol{\Delta})^{-1}\boldsymbol{\Delta}'(-\nabla_0) = (\widetilde{\boldsymbol{\lambda}}', \widetilde{\boldsymbol{\mu}}')' \text{ with } \boldsymbol{\Delta} = (\nabla_{h''}, \nabla_{g''})$$

BUT: (1) will never *perfectly* hold (**error term due to regression**) → how to take KKT into account in the infill criterion?

KKT conditions

Ideally: (-)gradient estimated by LS model equals (-)gradient estimated by GPR

$$\boxed{-\widetilde{\nabla}_0(\boldsymbol{x}_*) = \sum_{h'' \in A_\lambda(\boldsymbol{x}_*)} \widetilde{\lambda}_{h''}(\boldsymbol{x}_*) \nabla_{h''}(\boldsymbol{x}_*) + \sum_{g'' \in A_\mu(\boldsymbol{x}_*)} \widetilde{\mu}_{g''}(\boldsymbol{x}_*) \nabla_{g''}(\boldsymbol{x}_*) \quad \text{with } \widetilde{\lambda}_{h''}(\boldsymbol{x}_*) \geq 0 \text{ and } \widetilde{\mu}_{g''}(\boldsymbol{x}_*) \geq 0.}$$

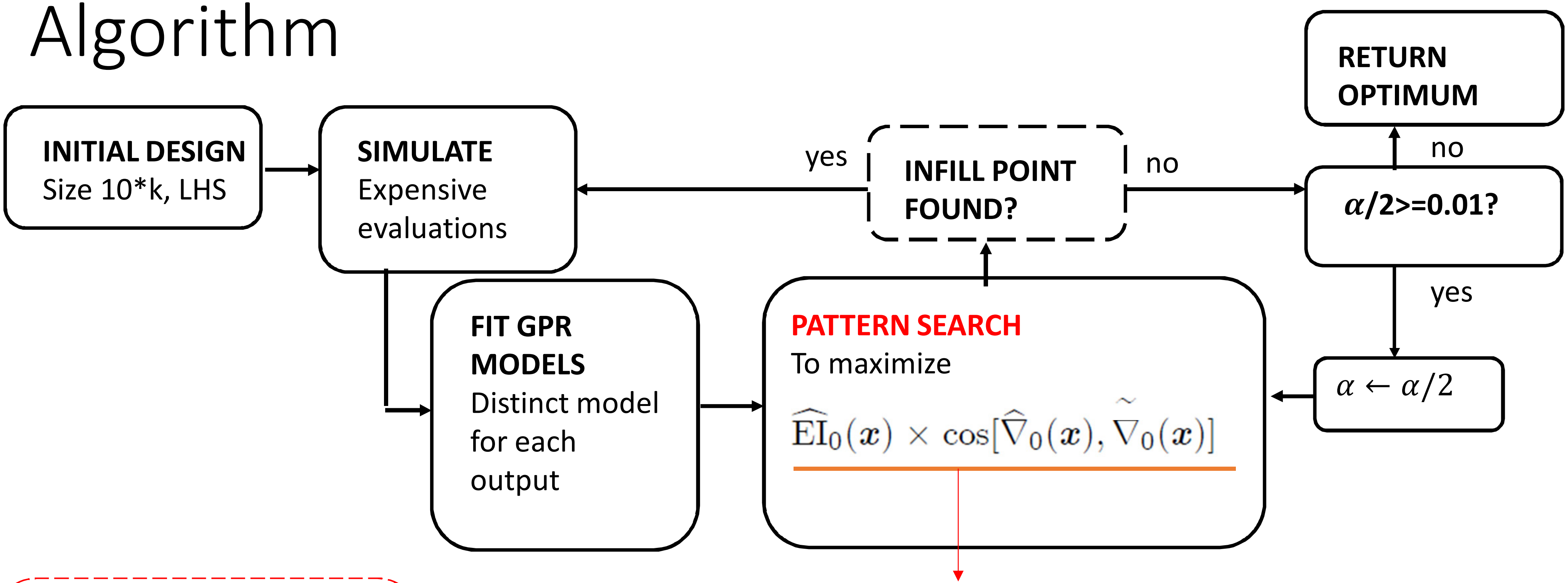
$= -\nabla_0(x_*)$

We quantify how close these two gradients are:

$$\cos[-\nabla_0(\boldsymbol{x}_*), -\widetilde{\nabla}_0(\boldsymbol{x}_*)] = \cos[\nabla_0(\boldsymbol{x}_*), \widetilde{\nabla}_0(\boldsymbol{x}_*)] = \frac{\nabla_0(\boldsymbol{x}_*) \bullet \widetilde{\nabla}_0(\boldsymbol{x}_*)}{||\nabla_0(\boldsymbol{x}_*)|| \times ||\widetilde{\nabla}_0(\boldsymbol{x}_*)||}.$$

= 1 in case of perfect fit!! Otherwise, we prefer close to 1

Algorithm



Output constraint is considered to be binding if

$$\frac{|\widehat{y}_{h'}(\boldsymbol{x}_*) - c_{h'}|}{s[\widehat{y}_{h'}(\boldsymbol{x}_*)]} \leq z_{1-\alpha/2}$$

Infill criterion: combines

- expected improvement at the new point x → balances exploration and exploitation
- Cos value at the new point x → “rewards” points that are close to the constraint

Results on toy example (Gramacy et al.2016)

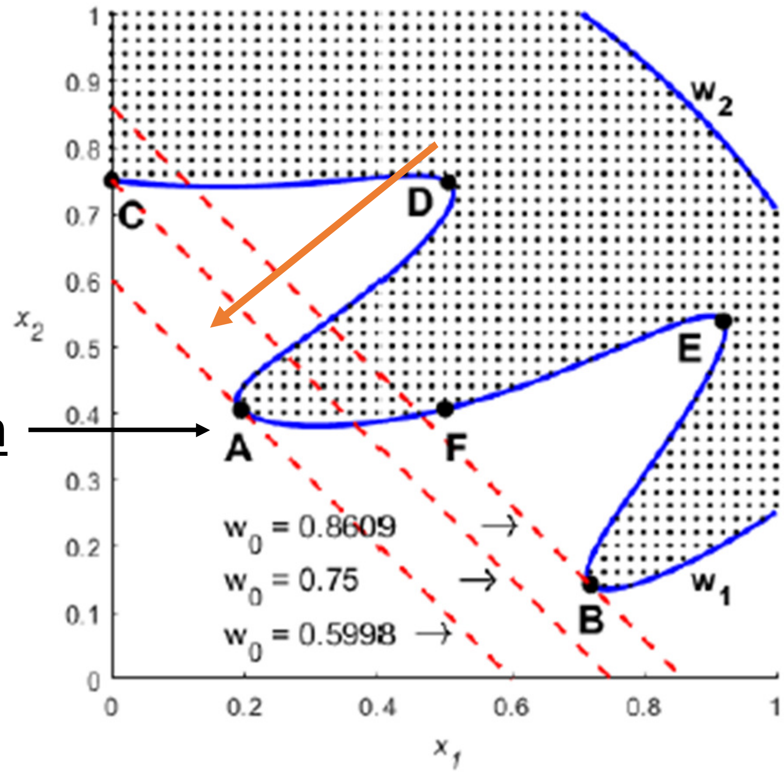
$$\min_{x} w_0(x) = x_1 + x_2$$

$$w_1(x) = \frac{3}{2} - x_1 - 2x_2 - \frac{1}{2}\sin[2\pi(x_1^2 - 2x_2)] \le 0$$

$$w_2(x) = -\frac{3}{2} + x_1^2 + x_2^2 \le 0$$

$$0 \le x_j \le 1 \ (j = 1, 2).$$

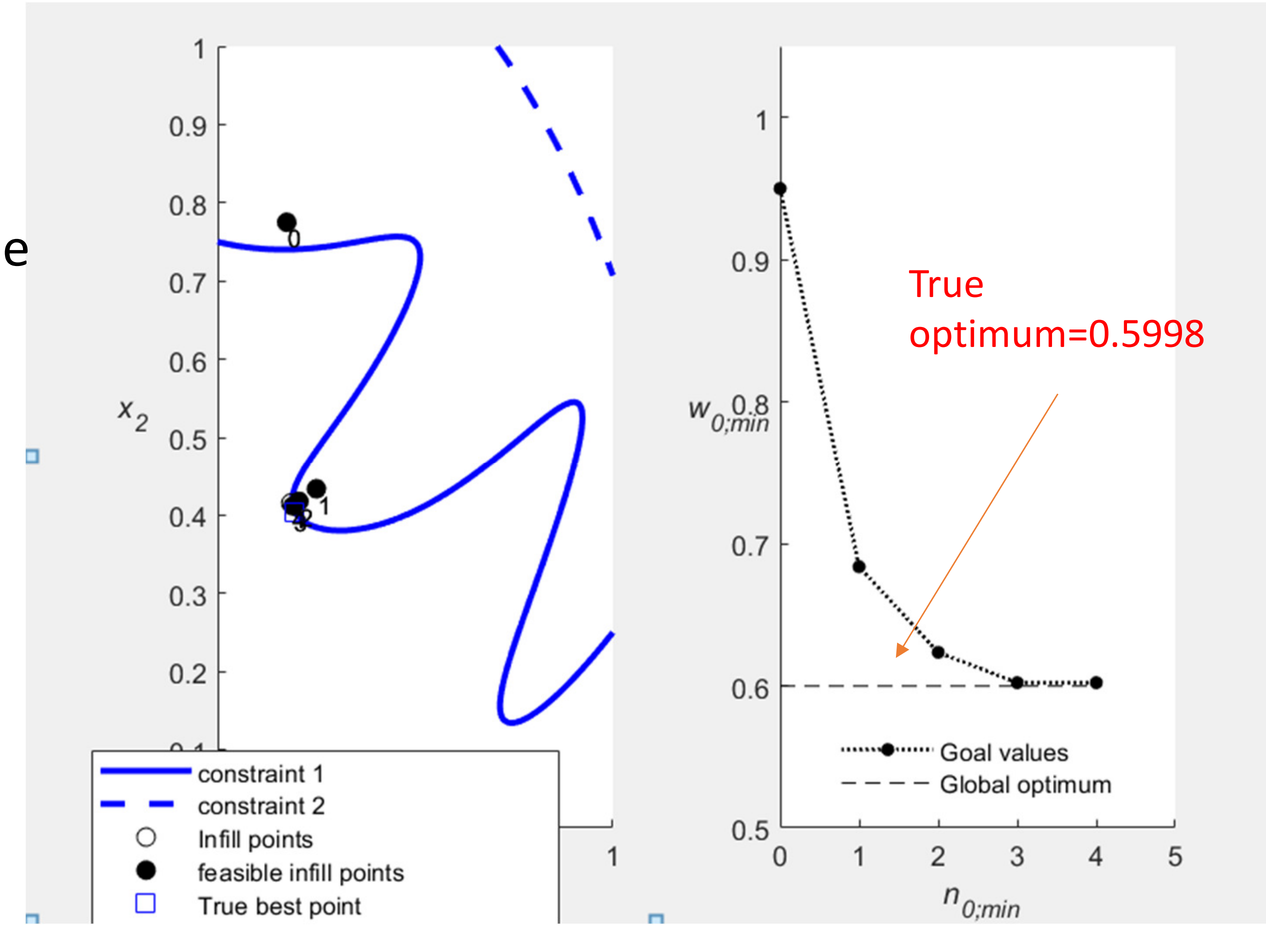
Global optimum



Gramacy, R.B., G.A. Gray, S. Le Digabel, H.K.H. Lee, P. Ranjan, G. Wells, and S.M. Wild (2016) Modeling an augmented Lagrangian for blackbox constrained optimization, *Technometrics*, 58, no. 1, pp. 1–11 (Comments by several authors, and rejoinder by Gramacy et al. pp. 12–29)

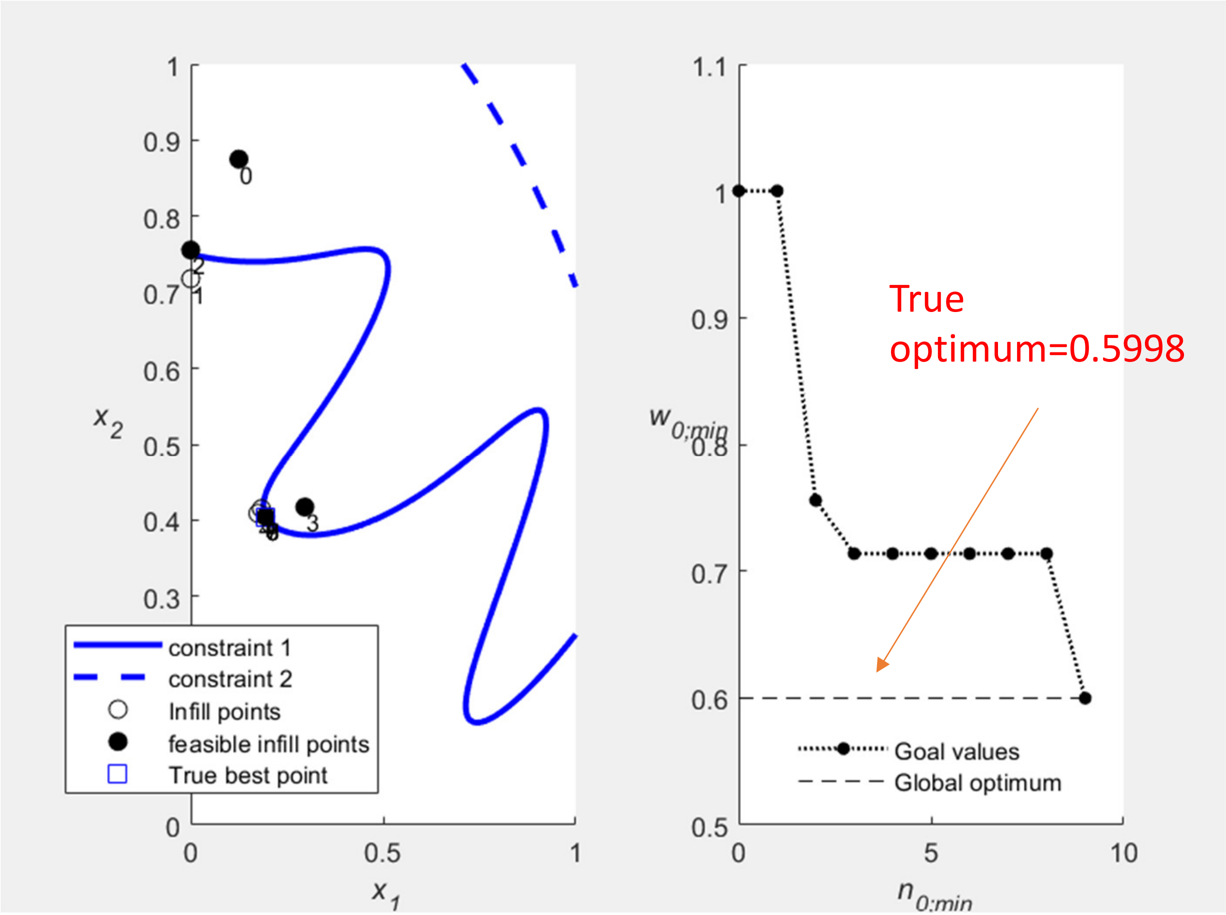
Results on toy example (Gramacy et al.2016)

- Example of search path
- We do m=50 macroreplications (each time different initial LHS)

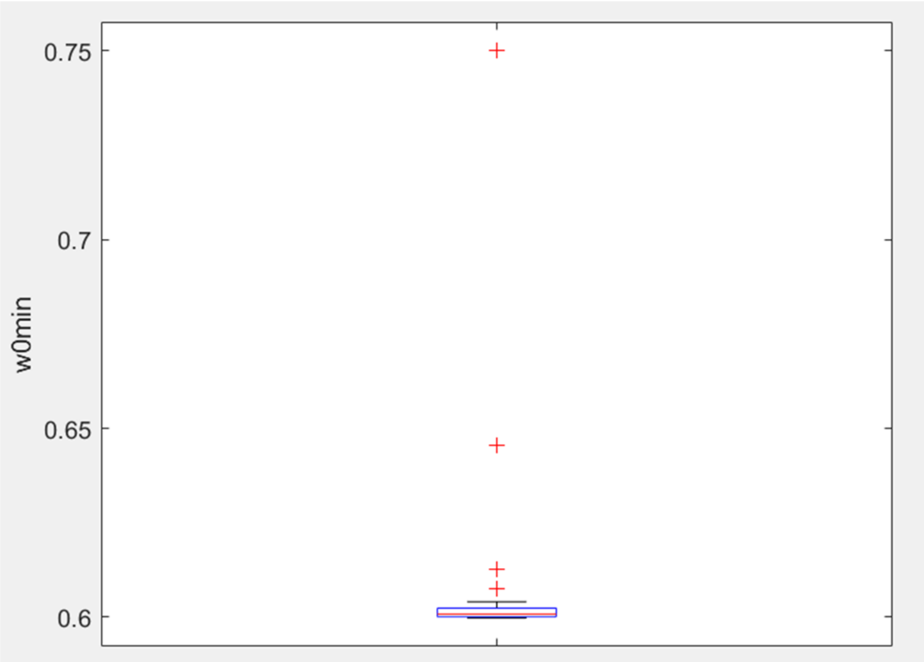


Results on toy example (Gramacy et al.2016)

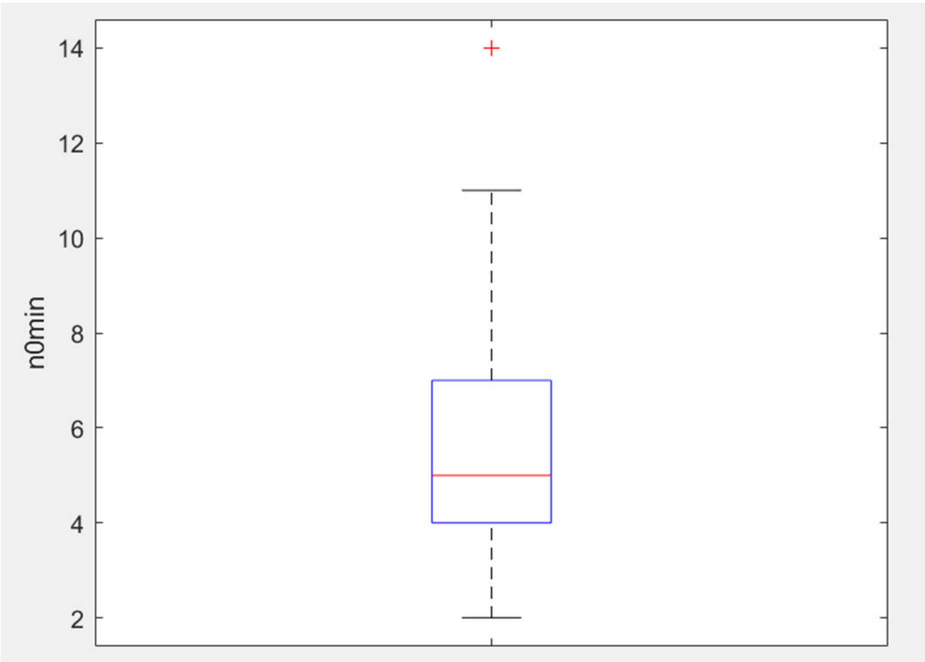
- Other search path example



Results on toy example (Gramacy et al.2016)



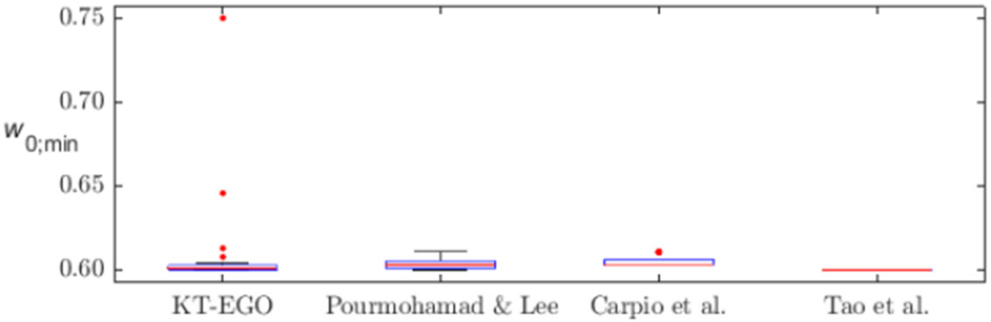
Boxplot final solution
(50 macroreplications)
median = 0.60085, mean = 0.6054

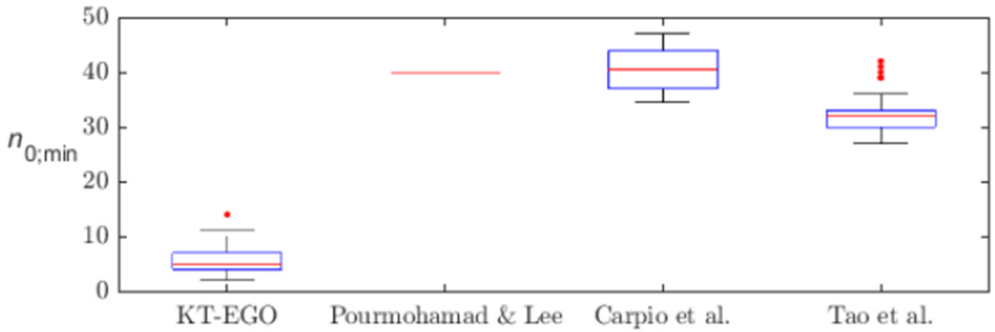


Number of iterations until
stopping criterion reached
(50 macroreplications)

Results on toy example (Gramacy et al.2016)

- Good performance on $w_{0;min}$
- Much faster than other recent algorithms





Carpio, R.R., R.C. Giordano, and A.R. Secchi (2018), Enhanced surrogate assisted framework for constrained global optimization of expensive black-box functions. *Computers & Chemical Engineering*, 118, pp. 91–102

Pourmohamad T. and H.K.H. Lee (2021), Bayesian optimization via barrier functions. *Journal of Computational and Graphical Statistics*, accepted

Tao, I., G. Zhao, and S. Ren (2020), An efficient Kriging-based constrained optimization algorithm by global and local sampling in feasible region. *Journal of Mechanical Design*, 142, no. 5, pp. 051401-1 – 051401-15

Results on spring example

$$\min_{x} w_0(x) = (x_1+2)x_2x_3^2$$

$$w_1(x) = 1 - \frac{x_2^3 x_1}{71,875x_3^4} \leq 0$$

$$w_2(x) = \frac{4x_2^2 - x_2x_3}{12,566(x_2x_3^3 - x_3^4)} + \frac{2.46}{12,566x_3^2} - 1 \leq 0$$

$$w_3(x) = 1 - \frac{140.54x_3}{x_2^2 x_1} \leq 0$$

$$w_4(x) = \frac{x_2 + x_3}{1.5} - 1 \leq 0$$

$$2 \leq x_1 \leq 15, 0.25 \leq x_2 \leq 1.30, 0.05 \leq x_3 \leq 0.20.$$

N = x_1
D=x_2
d=x_3

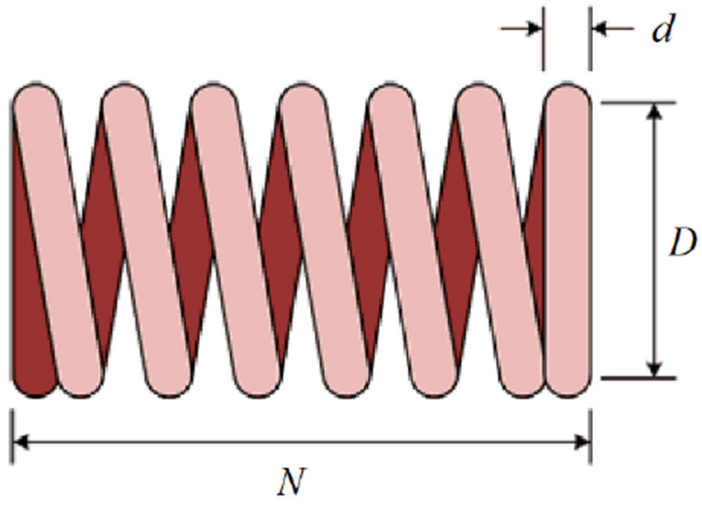
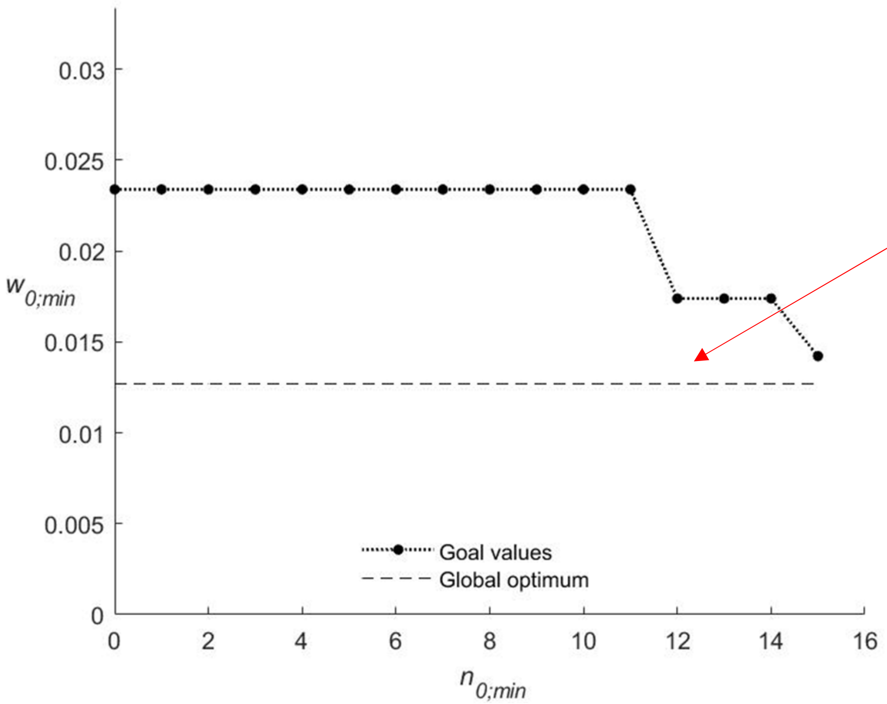


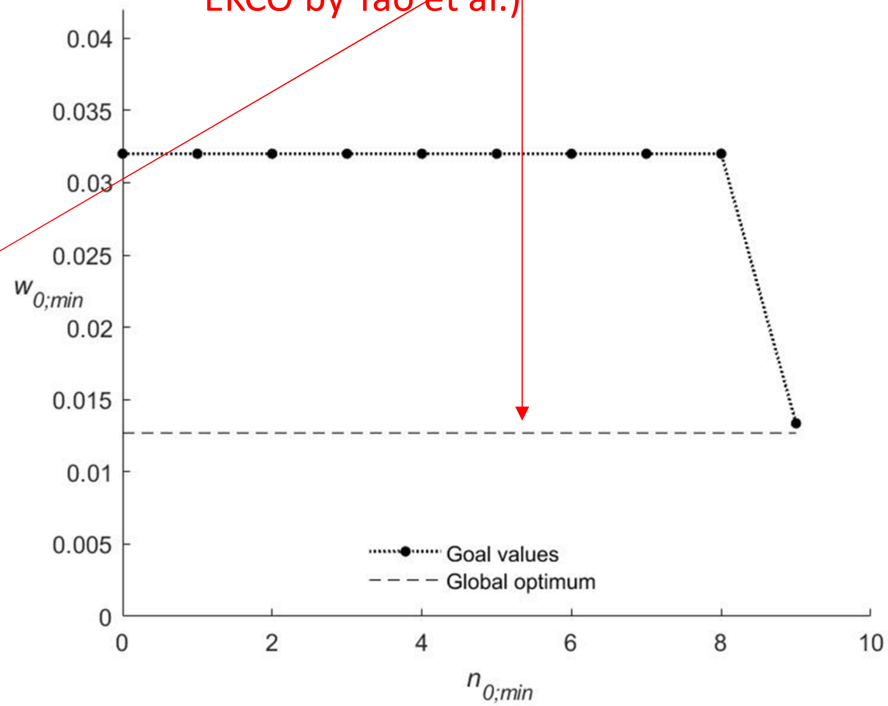
Figure 23: Mechanical engineering "spring" design problem

Results on spring example

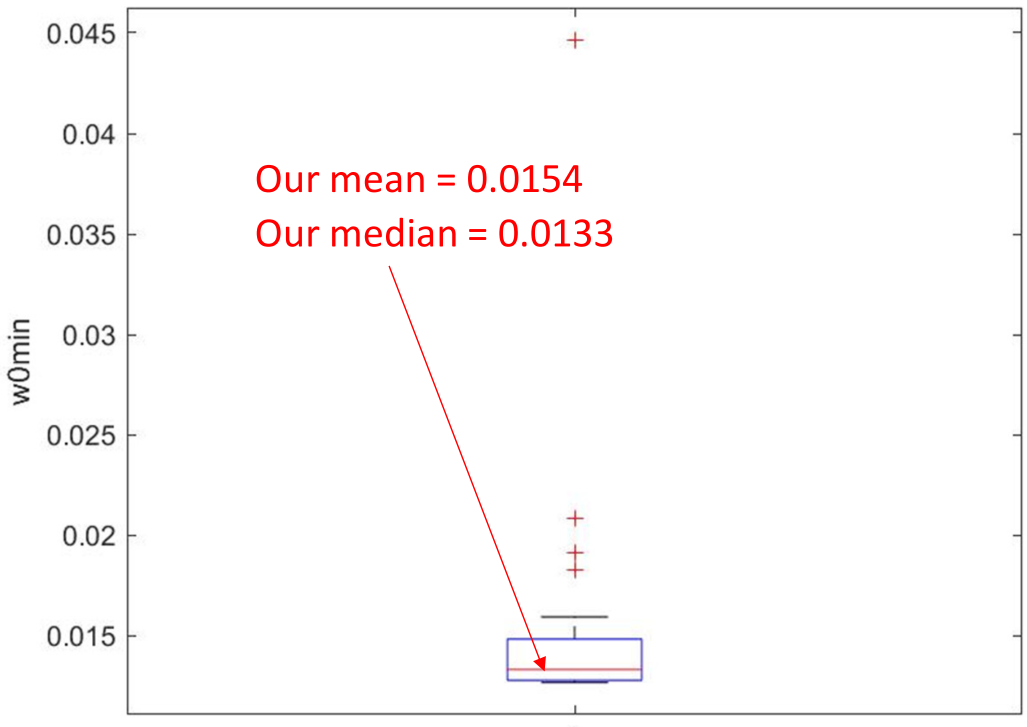
“True” optimum = 0.01269 (average of Table 3 Tao et al.)
(best in literature: 0.012664, in 92 iterations, EKCO by Tao et al.)

- Two arbitrary search paths

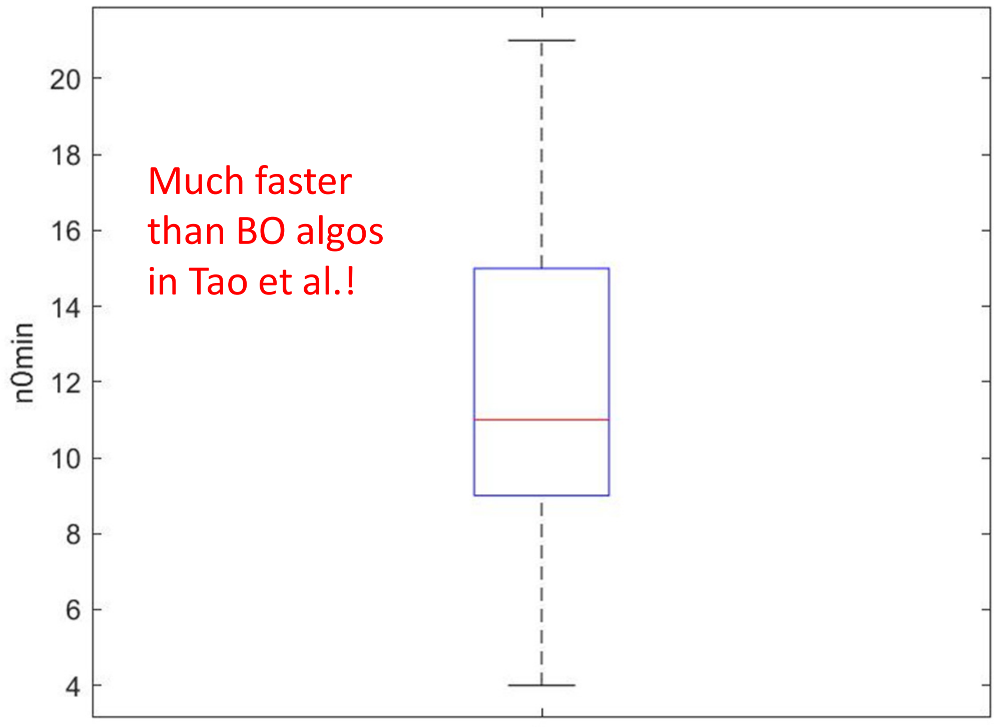




Results on spring example



Boxplot final solution
(25 macroreplications)



Number of iterations until
stopping criterion reached
(25 macroreplications)

Conclusions

- Results look promising, but probably room for improvement (see spring problem)
  - Due to stopping too soon?
  - Due to not exploring enough?
  - Due to pattern search options? Kernel choice? ….
- Other (engineering/real-life) test problems + compare performance with other recent algorithms
- Random simulation outputs