Model-based optimization with concept drifts

Jakob Richter, Junjie Shi, Jian-Jia Chen, Jörg Rahnenführer, Michel Lang
TU Dortmund, Deaprtment of Statistics, Dortmund, Germany
TU Dortmund, Deaprtment of Computer Science, Dortmund, Germany

Citation: 10.1145/3377930.3390175

BibTeX:

```
@inproceedings{DBLP:conf/gecco/RichterSCRL20,
  author    = {Jakob Richter and
               Junjie Shi and
               Jian{-}Jia Chen and
               J{\"{o}}rg Rahnenf{\"{u}}hrer and
               Michel Lang},
  title     = {Model-based optimization with concept drifts},
  booktitle = {{GECCO}},
  pages     = {877--885},
  publisher = {{ACM}},
  year      = {2020}
  doi={10.1145/3377930.3390175}
}
```

Model-based Optimization with Concept Drifts

Jakob Richter
richter@statistik.tu-dortmund.de
TU Dortmund University
Dortmund, Germany

Junjie Shi
junjie.shi@tu-dortmund.de
TU Dortmund University
Dortmund, Germany

Jian-Jia Chen
jian-jia.chen@cs.uni-dortmund.de
TU Dortmund University
Dortmund, Germany

Jörg Rahnenführer
rahnenfuehrer@statistik.tu-dortmund.de
TU Dortmund University
Dortmund, Germany

Michel Lang
lang@statistik.tu-dortmund.de
TU Dortmund University
Dortmund, Germany

ABSTRACT

Model-based Optimization (MBO) is a method to optimize expensive black-box functions that uses a surrogate to guide the search. We propose two practical approaches that allow MBO to optimize black-box functions where the relation between input and output changes over time, which are known as dynamic optimization problems (DOPs). The *window approach* trains the surrogate only on the most recent observations, and the *time-as-covariate approach* includes the time as an additional input variable in the surrogate, giving it the ability to learn the effect of the time on the outcomes. We focus on problems where the change happens systematically and label this systematic change *concept drift*. To benchmark our methods we define a set of benchmark functions built from established synthetic static functions that are extended with controlled drifts. We evaluate how the proposed approaches handle scenarios of no drift, sudden drift and incremental drift. The results show that both new methods improve the performance if a drift is present. For higher-dimensional multimodal problems the window approach works best and on lower-dimensional problems, where it is easier for the surrogate to capture the influence of the time, the *time-as-covariate approach* works better.

CCS CONCEPTS

• **Mathematics of computing** → *Bayesian computation*; *Continuous optimization*.

KEYWORDS

Model-based Optimization, Bayesian Optimization, Dynamic Optimization Problems

ACM Reference Format:
Jakob Richter, Junjie Shi, Jian-Jia Chen, Jörg Rahnenführer, and Michel Lang. 2020. Model-based Optimization with Concept Drifts. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390175

GECCO '20, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7128-5/20/07.
https://doi.org/10.1145/3377930.3390175

1 INTRODUCTION

We address the problem of global optimization of expensive black-box functions under the assumption of temporally evolving optima. We consider incrementally changing as well as suddenly changing functions in the sense of the definition of different concept drifts in machine learning [5]. Such functions are found in many domains, e.g. in the optimization of running production processes, where the quality of the production depends on controllable parameters which are usually subject of the optimization. Additionally, uncontrollable external parameters like temperature, humidity, etc. influence the production quality. Such external parameters can be measured directly or only latently e.g. through the time.

For static expensive black-box problems, Model-based optimization (MBO) has proven its worth [7]. MBO is an iterative algorithm that relies on a surrogate model to predict the outcome of the expensive black-box problem for unknown input values. The surrogate is a regression model that is fitted on previously observed outcomes and it is updated with a new result in each iteration. We aim to augment the MBO algorithm with two novel approaches to cope with dynamic optimization problems (DOPs). The goal is to obtain a generic approach where any DOP can be optimized with any surrogate model within the MBO framework. The first approach uses a sliding time window where only the most recent observations are used to train the surrogate model. The second approach includes the time as a covariate, allowing the surrogate to model the influence of the time directly.

We evaluate our approaches on synthetic test functions with controlled dynamic changes to eliminate any side effects while studying the properties of the new approaches. The performance is measured by the cumulative performance of all evaluations of the black-box during the optimization.

Related Work. The overview paper on optimization in dynamic environments [3] gives an extensive overview of optimization approaches and benchmarks for DOPs. The majority are evolutionary [10] and particle swarm [9] optimization approaches that require a large number of function evaluations. Changes of the objective function are assumed to happen only after one or multiple generations are evaluated, and are usually stochastic. Most of the many evaluation criteria listed in [3] are tailored to the characteristics of evolutionary optimization algorithms, e.g. *current best-of-generation evolution*, *mean best-of-generation*. As such they only take the best solution of a set of evaluations, e.g. a single generation,

into account. This means that many intermediate evaluations of the objective over time can perform badly without affecting the performance under such criteria.

Our scenario for DOPs is different. First, the evaluation of the objective takes a significant amount of time. Therefore, only a very limited number of evaluations is possible. Furthermore, we assume that the objective function has already changed after an evaluation and therefore running multiple evaluations on the same state of the objective function is not possible. Second, the objective function changes systematically over time.

In [11] a Bayesian Optimization approach for DOPs is presented that exclusively uses the LCB acquisition function in combination with a Gaussian process surrogate with a special kernel for the time dimension. Unfortunately there is no available implementation.

2 MODEL-BASED OPTIMIZATION (MBO)

The aim of MBO [7] is to find the global minimum $\boldsymbol{x}^*$ of a given objective function $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}). \tag{1}$$

The function is assumed to be an expensive black-box function with a d-dimensional input domain $\mathcal{X} \subset \mathbb{R}^d$, usually expressed by simple box constraints. We assume that the surrogate can approximate the true expensive black-box function. This surrogate is a regression method that is comparably inexpensive to evaluate. If we refer to MBO as Bayesian optimization, typically a Gaussian process regression (Kriging) is chosen as a surrogate [8]. To initialize the optimization, an initial design $\mathcal{D}$ of k points, laid out in a Latin hypercube design, is evaluated on the objective function and yields the outcomes $\boldsymbol{y}$. For the optimization the following steps are repeated until a predefined budget of iterations is exhausted:

(1) A Gaussian process is fitted on the Design $\mathcal{D}$ and the outcomes $\boldsymbol{y}$, yielding the surrogate.
(2) An acquisition function that is based on the surrogate is optimized to determine the most promising point $\boldsymbol{x}$ = $\arg\max_{\boldsymbol{x} \in \mathcal{X}} \text{acq}(\boldsymbol{x})$.
(3) $y = f(\boldsymbol{x})$ is evaluated and $\boldsymbol{x}$ and y are added to $\mathcal{D}$ and $\boldsymbol{y}$.

The final optimization result $\hat{\boldsymbol{x}}^*$ is the input that led to the minimal observed objective value.

The aim of an acquisition functions is to balance the evaluation of points where the surrogate predicts a low outcome (exploitation of the predicted minimum) and points where the surrogate is uncertain (exploration of new areas). Common choices for the acquisition functions are the *expected improvement*

$$\text{EI}(\boldsymbol{x}) = \text{E}(\max(y_{\min} - \hat{f}(\boldsymbol{x}), 0)) \tag{2}$$

and the *lower confidence bound* [7]:

$$\text{LCB}(\boldsymbol{x}, \lambda) = \hat{\mu}(\boldsymbol{x}) - \lambda \hat{s}(\boldsymbol{x}) \,, \tag{3}$$

with $y_{\min}$ as the minimum observed in $\boldsymbol{y}$ so far, $\hat{f}$ as the random variable following the posterior distribution of the function outcomes given by the surrogate, $\hat{\mu}(\boldsymbol{x})$ as the mean and $\hat{s}(\boldsymbol{x})$ as the uncertainty prediction of the surrogate, and λ as a tuning parameter. For a closed form of the EI see [7].

3 MBO WITH CONCEPT DRIFT

In contrast to the previous Section 2, we now assume that $f(\boldsymbol{x})$ as well as its optimum $\boldsymbol{x}^*$ changes over time t. Thus, the optimization problem (1) becomes

$$\boldsymbol{x}_t^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} f_t(\boldsymbol{x}) \,. \tag{4}$$

The relation between the input $\boldsymbol{x}$ and the outcome y changes (drifts) from one state (concept) to another over time. In this work we consider changes of two different kinds: A single instantaneous change of the objective functions is labeled as **sudden drift**. An **incremental drift** describes a change that happens constantly with a continuous trend. The concept is lent from the machine learning community [5]. We consider two extensions to the MBO framework to solve such DOPs.

3.1 Window Approach

In each iteration only evaluations observed in the window of the last t_Δ time units are considered for the design $\mathcal{D}$. Consequently, we have to treat the objective function as stochastic since the same $\boldsymbol{x}$ leads to different outcomes y at different points in time. Therefore, the EI cannot be used, since the reference value $y_{\min}$ is not necessarily the correct minimal value for the current concept of the objective function. We propose to use the augmented expected improvement (AEI) [6] that was proposed for stochastic optimization problems. For the AEI we first need to obtain the *effective best solution* [6]:

$$\boldsymbol{x}^{**} := \arg\min_{\boldsymbol{x} \in \mathcal{D}} \hat{\mu}(\boldsymbol{x}) \quad c \cdot \hat{s}(\boldsymbol{x}) \,, \tag{5}$$

with c as a tuning parameter that is usually set to 1. The AEI then calculates as follows:

$$\text{AEI}(\boldsymbol{x}) = \text{E}\left(\max(\hat{\mu}(\boldsymbol{x}^{**}) - \hat{f}(\boldsymbol{x}), 0)\right) \cdot \left(1 - \frac{\sigma_n}{\sqrt{\sigma_n^2 \quad \hat{s}^2(\boldsymbol{x})}}\right), \tag{6}$$

with σ_n^2 denoting the variance of the random error (nugget effect) of the stochastic objective function. The nugget effect is assumed to be normally distributed and estimated within the Kriging model that is used as a surrogate. The usage of $\boldsymbol{x}^{**}$ as a reference value encourages the re-evaluation of points close to potential optima. This is a desired effect to avoid overly optimistic estimates because of outliers. However, to prevent evaluations that do not further decrease the uncertainty, the additional correction term ensures that the AEI becomes smaller, when the uncertainty estimation $\hat{s}^2(\boldsymbol{x})$ converges to the variance of the random error σ_n^2. For a closed form of the AEI see [6]. The LCB can be used without adaptation for stochastic objectives.

The window-approach is supposed to lead to a frequent re-explorations of previously explored areas. While this is necessary to adapt to a concept drift, it can be inefficient in cases without drift. Similar to window-based approaches in time series analysis, the choice of t_Δ symbolizes a trade-off. If the window is large, much information is available to fit the surrogate. In presence of sudden changes, the surrogate reacts slower to changes. If the window is small on the other hand, the surrogate can adapt faster, but it might be comparably less accurate as it relies on less information.

3.2 Time-as-Covariate Approach

The time is included in the surrogate model as an additional dimension t. Interactions between $\boldsymbol{x}$ and t can be identified and the drift can be modeled by the surrogate. This approach relies on a certain extrapolation capability of the surrogate, as we are interested in predictions at time t_{now} but only have observations from the past. To propose a point for the current MBO-iteration, we optimize the acquisition function on the hyperplane with fixed time $t = t_{\text{now}}$. For the same reason as in Section 3.1 the EI is not a valid choice. Setting the covariate t to a fixed value allows us to adapt the AEI to obtain the *temporal expected improvement*. First we obtain the effective best solution for time $t = t_{\text{now}}$:

$$\boldsymbol{x}_t^{**} := \underset{\boldsymbol{x} \in \mathcal{D}}{\arg\min}\, \hat{\mu}_t(\boldsymbol{x}) \quad c \cdot \hat{s}_t(\boldsymbol{x}) \,. \tag{7}$$

Then the temporal expected improvement for a fixed t is:

$$\text{TEI}_t(\boldsymbol{x}) = \text{E}\left(\max\left(\hat{\mu}_t(\boldsymbol{x}_t^{**}) - \hat{f}_t(\boldsymbol{x}), 0\right)\right) \,, \tag{8}$$

with $\hat{f}_t$ as the random variable following the posterior distribution of the function outcomes for time t given by the surrogate, $\hat{\mu}_t(\boldsymbol{x})$ as the mean and $\hat{s}_t(\boldsymbol{x})$ as the uncertainty prediction of the surrogate for a fixed t. Note that in contrast to the *window approach*, here the objective function f_t can bee seen as deterministic, because the time dependency is included. Therefore, we can set $\sigma_n = 0$ and the correction term from the AEI (6) becomes zero. Finally, the TEI can be calculated analytically as

$$\text{TEI}_t(\boldsymbol{x}) = \left(\hat{\mu}_t(\boldsymbol{x}^{**}) - \hat{\mu}_t(\boldsymbol{x})\right) \Phi\left(\frac{\hat{\mu}_t(\boldsymbol{x}^{**}) - \hat{\mu}_t(\boldsymbol{x})}{\hat{s}_t(\boldsymbol{x})}\right) \phi\left(\frac{\hat{\mu}_t(\boldsymbol{x}^{**}) - \hat{\mu}_t(\boldsymbol{x})}{\hat{s}_t(\boldsymbol{x})}\right) \,, \tag{9}$$

with ϕ and Φ as the standard normal density and distribution functions.

The adaptation of the LCB is merely technical to ensure that we only respect the surrogate's prediction for time t:

$$\text{LCB}_t(\boldsymbol{x}, \lambda) = \hat{\mu}_t(\boldsymbol{x}) - \lambda \hat{s}_t(\boldsymbol{x}) \,. \tag{10}$$

4 EXPERIMENTAL SETUP

The goal of the benchmark is to compare the optimization performance of the presented optimization approaches on DOPs with different types of drifts. Therefore, we need to define the DOPs that serve as benchmark functions and a performance measure.

4.1 Synthetic Dynamic Optimization Problems

We construct the benchmark according to the following principles: We use functions that are well-known in the optimization community. For each function three versions should exist: A baseline version with *no drift*, one with a *sudden drift*, and one with an *incremental drift*. The drift should be of a controlled fashion and not change the values of the optima but only their position in the domain space.

Therefore, we construct the DOPs from three parts: A static *objective function* $f(\boldsymbol{x})$, a *drift function* $\delta(t)$ that yields the state of the drift depending on the time and a *transformation function* $g(\boldsymbol{x}, w)$ that defines how the objective function is changed depending on the state of the drift. The constructed DOP will be a composed function of the form $f_{\delta,g,t}(\boldsymbol{x}) = f(g(\boldsymbol{x}, \delta(t)))$. The transformation function $g(\boldsymbol{x}, w) : [0,1]^d \to [0,1]^d$ transforms the input $\boldsymbol{x}$ according to the drift state w. The drift function $\delta(t) : [0,1] \to [0,1]$ maps the time t to the drift state w, e.g. $\delta(t) = 0.5$ implies that the time does not change the state of the function at all.

Objective Function: First, we take one of the black-box functions $f(\boldsymbol{x})$ from the Table 1. All of them are implemented in the R package `smoof` [2] and have already been used to generate DOPs [3, 11], but usually with stochastic changes. Functions that are defined on a 1, 2 and 5-dimensional input space are grouped in Set 1 and the others in Set 2. All functions in Set 1 have a parabolic surface combined with cosine waves of different amplitude and frequency as it can bee seen in Figures 2 and 3 for the *no drift* case. Consequently, they are multimodal. Functions of Set 2 are also multimodal but with fewer local minima and a more smooth surface. Functions that were originally constructed as maximization problems were flipped to become minimization problems. Furthermore, each function is standardized in two ways. The input space is scaled to $[0,1]^d$ and function values are scaled, so that $f(\boldsymbol{x}^*) = 0$ and the median performance is 1. The median performance is obtained by evaluating a grid of $\min(100^d, 10^6)$ values of $\boldsymbol{x} \in \mathcal{X}$ on the objective function and calculating the median of the outcomes.

Table 1: Objective functions that serve as base functions devided in two sets. d is the dimensionality of the input space $\mathcal{X}$, x^* is the location of the global optimum.

Function	d	$\boldsymbol{x}^*$
Set 1		
Ackley	1	$(0.5)'$
	2	$(0.5, 0.5)'$
	5	$(0.5, 0.5, 0.5, 0.5, 0.5)'$
Griewank	1	$(0.5)'$
	2	$(0.5, 0.5)'$
	5	$(0.5, 0.5, 0.5, 0.5, 0.5)'$
Rastrigin	1	$(0.5)'$
	2	$(0.5, 0.5)'$
	5	$(0.5, 0.5, 0.5, 0.5, 0.5)'$
Set 2		
Branin	2	$(0.12, 0.82)'$, $(0.54, 0.15)'$, $(0.96, 0.17)'$
Camelback	2	$(0.49, 0.68)'$, $(0.51, 0.32)'$
Goldstein-Price	2	$(0.5, 0.25)'$

Drift Function: The drift function $\delta(t)$ is utilized to introduce different types of drifts, depending on the elapsed time t: We consider the following drift functions $\delta(t) : [0,1] \to [0,1]$:

No Drift	$\delta_n(t) = 0.5$
Sudden Drift at $t = 0.5$	$\delta_s(t) = \mathbf{1}_{[0.5,1]}(t)$
Incremental Drift	$\delta_i(t) = -0.5 \cdot \left(\sin\left(\frac{\pi}{2} - \pi \cdot t\right) - 1\right)$,

with $\delta_s(0) = \delta_i(0) = 0$ and $\delta_s(1) = \delta_i(1) = 1$. The outcome of $\delta(t)$ will define the state w of the drift. All drift functions are illustrated in Figure 1.

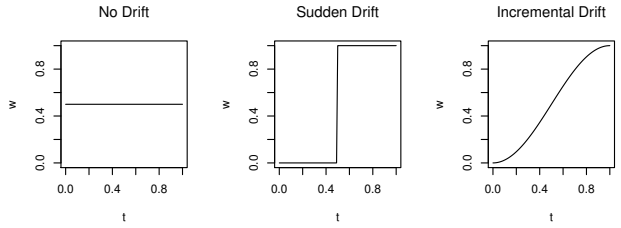

Figure 1: The three different drift functions δ_n, δ_s and δ_i.

Transformation Function: Finally, we define $g(\boldsymbol{x}, w)$ which transforms the input in each dimension depending on the state w of the drift. The function is constructed in such a way that for $w = 0$: $g(\boldsymbol{x}, 0) = \boldsymbol{x}^{\frac{1}{3}}$, so that the optima are dragged to the left. For $w = 0.5$ no transformation should be applied so that $g(\boldsymbol{x}, 0.5) = \boldsymbol{x}$. For $w = 1$: $g(\boldsymbol{x}, 1) = \boldsymbol{x}^3$, which has the effect that all optima are dragged to the right. We choose the exponent as 3 to obtain a strong shift if w is 0 or 1. For a one dimensional objective function with an optimum at $\boldsymbol{x}^* = 0.5$, the optimum for $w = 0$ is at $g(0.5, 1) = 0.125$ and for $w = 1$ the optimum is at $g(0.5, 0) \approx 0.794$.

To obtain transformations for every weight $w \in [0, 1]$ we derived the following function

$$g\left(\boldsymbol{x}_l, w\right) = \boldsymbol{x}_l^k \text{ , with } k = -2 \cdot (w - 1.5)^{-1} - 1\,, \tag{11}$$

which transforms $\boldsymbol{x}$ for each dimension $l \in 1, \ldots, d$. If the global optimum $\boldsymbol{x}^*$ is known, its position at time t can be derived from the inverse of g. In fact the inverse of g solves to:

$$g^{-1}(\boldsymbol{x}, w) = g(\boldsymbol{x}, 1 - w) \tag{12}$$

Combination: To put all pieces together, we first plug the drift function $\delta(t)$ into our transformation function to obtain $g\left(\boldsymbol{x}, \delta(t)\right)$. Applying this scheme, we can combine any static objective function f with any drift function $\delta(t)$ and the transformation function g to obtain a DOP:

$$f_{\delta,g,t}(\boldsymbol{x}) = f\left(g\left(\boldsymbol{x}, \delta\left(t\right)\right)\right). \tag{13}$$

Figures 2 and 3 visualize how the functions and the optimum changes over time for the one and two-dimensional DOPs constructed out of the static objective functions.

4.2 Performance Measurement

To evaluate the performance of the optimizers we need an evaluation measure that considers the online characteristic of our dynamic optimization problems. In comparison to regular MBO we are not interested in the distance of the final result from the true optimum at the end of the optimization process, but in the distance from the optimum at any given time point t.

In our scenario for DOPs a function evaluation takes a significant amount of time. Therefore, we have to assume that the objective function has already changed after an evaluation. Accordingly, an evaluation criterion where only the best evaluation within a certain batch counts towards the error, as common in the DOP literature [3], would be an unreasonable choice. We decide to count every evaluation during optimization towards the performance.

In our benchmark scenario we know the true optimum y_t^* for any given point in time. Therefore, we calculate the *fitness error*

$$\text{err}_{\text{FE}} = f_t(\boldsymbol{x}_t) - y_t^* \tag{14}$$

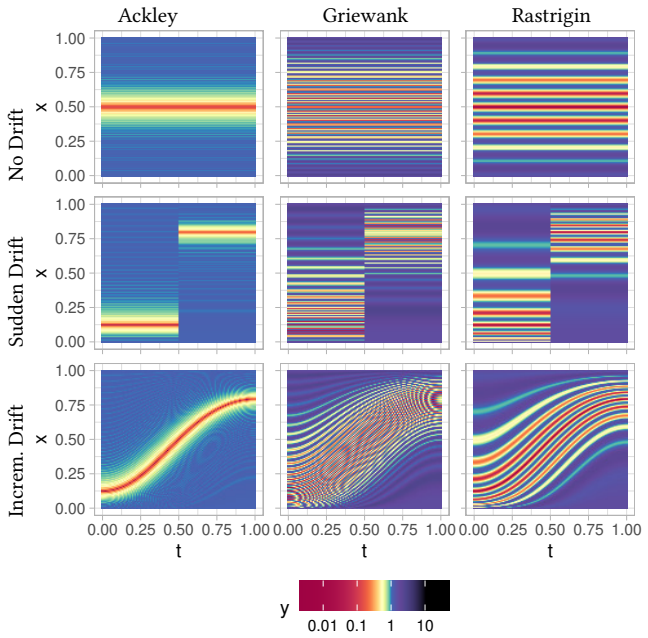

Figure 2: All $1d$ functions from Set 1 combined with the different drifts.

for the minimization of f_t with $\boldsymbol{x}_t$ as the proposal of the optimizer for time t. For the evaluation of the whole optimization run we average the errors over all time points which gives us the *mean fitness error* (MFE). This performance measure reflects a scenario where every evaluation counts towards the final performance. In the context of MBO this implies that exploration can negatively affect the overall error but can also be necessary to find regions that perform better than the currently best setting.

5 BENCHMARK

The benchmark consists of $(3 \cdot 3 \quad 3) \cdot 3 = 36$ *problems*: 3 objective functions in Set 1, each defined on three dimensions $d \in \{1, 2, 5\}$, 3 objective functions of dimensionality 2 in Set 2, and all of them without drift, with a sudden drift and with an incremental drift, constructed as described in Section 4.1.

We divide the time frame of $[0, 1]$ into 100 discrete time steps. After each evaluation of the objective function the time progresses one step. The initial design consists of a Latin hypercube sample of $4 \cdot d$ points evaluated at $t = 0$.

Each optimization is run with 50 stochastic repetitions. All optimizers start with the same set of 50 different initial designs for all 50 stochastic repetitions for each problem.

5.1 Optimization Algorithms

For the benchmark we consider a trivial baseline, the ordinary MBO and the proposed adaptations as presented in Section 3:

`constant`: From the initial design the best configuration is taken and evaluated at each time step.

bo: Sequential MBO executed regularly without notion of concept drifts that might occur.

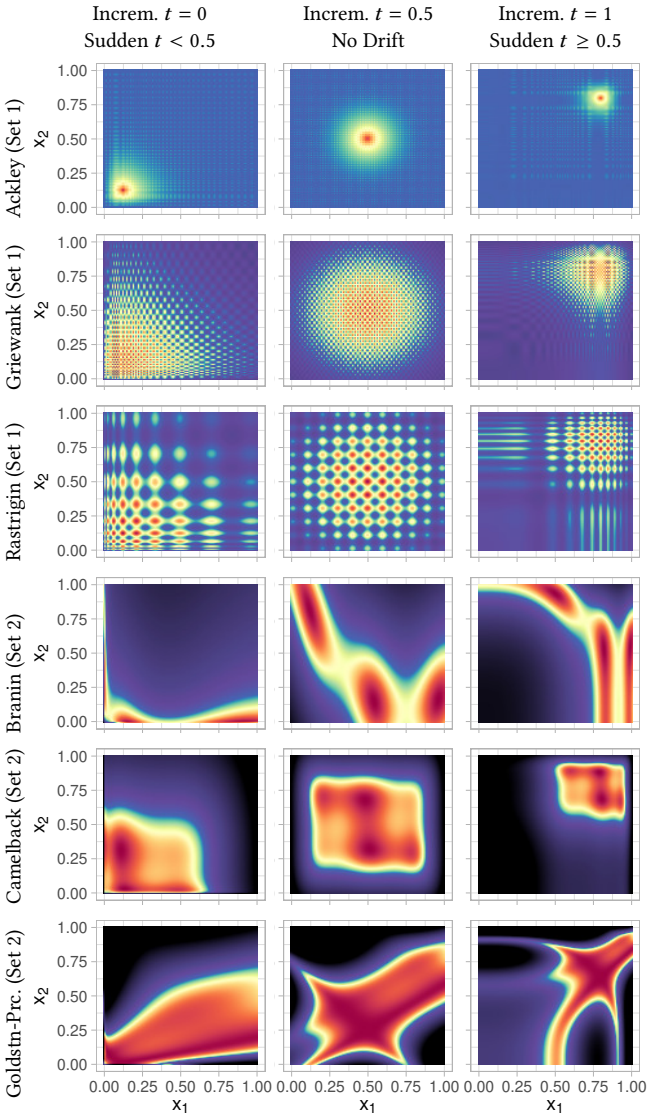

Figure 3: All $2d$ functions with incremental drift and their states at $t = 0$, $t = 0.5$ and $t = 1$. For $t = 0.5$ the state equals the constant state of the function with no drift. For $t = 0$ and $t = 1$ the states equal the two different states for the sudden drift. Same color scale as in Figure 2.

bo_tw: Drift-aware MBO with time-window (see 3.1) and two different window sizes of 20 and 40 time steps.

bo_tac: Drift-aware MBO with the time as covariate (see 3.2).

Each of the MBO algorithms is run with two different acquisition functions.

_cb2 The *confidence bound* with $\lambda = 2$ as defined in Eq. 3 for bo and in Eq. 10 for bo_tw and bo_tac.

_aei The *augmented expected improvement* as defined in Eq. 6 for bo and bo_tw.

_tei The *temporal expected improvement* as defined in Eq. 8 for bo_tac.

The proposal that is obtained by optimizing the acquisition functions is the value that counts towards the performance measure. All optimizers use Kriging with a Matern $\frac{5}{2}$-kernel for covariance estimation as a surrogate. For bo_tw the *nugget effect* is estimated during the surrogate fit, to account for the non-deterministic behavior of the objective function over time. For bo_tw and bo_tac the time steps are supplied as an additional input variable, whereas bo is kept unaware of the time. We use MBO as implemented in the R package mlrMBO [1] which uses the R package DiceKriging [12] for the surrogate.

5.2 Results

For each discrete time step we measured the *fitness error* (FE) of each optimizer on each problem. The optimization paths in Figure 4 show, for a subset of the problems, the FE values for each time step averaged over all stochastic repetitions.

For the problems with **no drift** we observe a particular pattern for the window-based optimizers. It is most noticeable for Camelback 2d that after each 20 (for bo_tw20) and after each 40 (for bo_tw40) iterations an error peak occurs. At these iterations the information from the initial design drops out of the time-window and the uncertainty predicted by the surrogate increases in certain regions. The re-exploration leads to an error peak as bad performing areas are reevaluated. This pattern repeats as information about bad performing configurations gets repeatedly lost. This behavior is also present for functions with drift.

Comparing the optimization curves of the aei/tei acquisition function with the lcb curves in the no drift scenario shows that often the EI-based acquisition functions do not get as low as the lcb curves and in some cases go up (Griewank 5d). We call this the *saturation* effect, where the acquisition functions prevents exploitative proposals and guides the optimizer towards exploration. Interestingly, adding the time as a covariate can prevent this effect. This can be seen for the Griewank 1d function, where bo_tac_tei can outperform bo_aei, even though no drift is present. bo_tw40 slightly performs better than bo_tw20 which can potentially contributed to the bigger design of the surrogate.

For problems with a **sudden drift** the change is clearly visible at iteration 50. The smaller window size of bo_tw20 allows to faster adapt to the change. For bo_tac the discontinuity challenges the Kriging regression, which expects input from a differentiable function. Only for the lower dimensional versions of the Griewank function and for Camelback 2d we can see that the error decreases for bo_tac.

For problems with **incremental drift** bo_tac_cb2 manages to constantly perform with low error rates, i.e. it is able to keep track of the changes. bo_tac_aei performs similarly but has issues on the very multimodal Griewank 5d function. The window-based approaches need a bit longer to adapt to the changes and naturally the smaller window adapts faster.

To interpret the results of all problems we combine functions of the same set, dimensionality and drift type into *groups*. For each *group* we generate a *preference graph* and calculate the *average ranks* to compare the performance of the optimizers jointly in one

group. For both evaluations the performance of each optimizer is calculated by the *mean fitness error* (Section 4.2) on each problem and for each single stochastic repetition.

The first evaluation method generates a **preference graph**. This directed graph shows which optimizer was able to statistically beat which optimizer in a pairwise comparison. The pairwise comparison is conducted with a one-sided paired-sample sign test on the MFE values of each repetition and of each problem across two competing optimizers. Each test is conducted at the 5% significance level without further adjustment for multiple testing, as this analysis is only meant as an exploratory visualization of the stochastic results. The results of all pairwise comparisons yield a directed graph. A transitive reduction on the results removes any direct edge if there also exist an indirect path in order to improve readability. The preference graphs for all grouped problems is shown in Figure 5.

The preference graph in Figure 5 shows that there is no clear winner across all scenarios. In case of **no drift** the drift-unaware `bo_cb2` performs well for Set 1. Also, the bigger window of the time-window approaches is beneficial if we compare the `bo_tw` approaches with the same acquisition function. The time-as-covariate approach only works well on the 1*d* problems of Set 1 and on the problems of Set 2. Even though no drift is present the drift-unaware `bo_aei` optimizer performs badly for the 1*d* and 2*d* problems of Set 1, which can be probably accounted to the mentioned saturation effect. The exploratory evaluations are also the reason why all `bo` optimizers perform worse than the *constant* baseline on Set 2.

In case of a **sudden drift** the window-based optimizers with the smaller windows perform better on Set 1. Also here, the time-as-covariate approach works comparably well on the 1*d* problems of Set 1 and the problems of Set 2.

For functions with **incremental drift** the optimizers with smaller window sizes again have an advantage on Set 1. The time-as-covariate approach `bo_tac_cb2` works comparably well across all problems of Set 1 and Set 2.

The second evaluation method generates a **rank table** based on the mean ranks of each optimizer. Therefore, we first rank the optimizers among each other for each *problem* and each stochastic replication by their MFE. Afterwards we calculate the mean of all ranks for each optimizer across all problems and replications within one *group* of problems. We conduct a statistical analysis on the underlying unranked data to find out which optimizers do not perform significantly worse then the best performing optimizer within a group. To verify whether there are statistically significant differences between the optimizers, we perform a non-parametric test procedure as recommended in [4]. First, the Friedman test is employed to test whether there are any statistical differences between all optimizers. If the null hypothesis of no differences between the optimizers cannot be rejected, we will underline all results to indicate that no algorithms are significantly different from each other. Second, if the null is rejected, we test each optimizer against the best performing one on the null hypothesis that the best performing optimizer is not better than the competitor. Therefore, the one-sided paired-sample sign test is applied on the MFE values of each repetition and each problem of the given group and the two competing optimizers. The test is adjusted for multiple comparisons using the Bonferroni-Holm-Correction. All tests are conducted at the 5% significance level. Results of optimizers that are not significantly worse then the best performing optimizer will be underlined to indicate that they are potential candidates for the best method.

The average ranks are given in Table 2. Again it shows that there is no clear winner across all scenarios. In situations with drift, `bo_tw20_aei` is a promising candidate for functions of Set 1 but performs badly on Set 2. For Set 2 `bo_tac_tei` is advantageous for both drift situations. It has to be noted that functions of Set 2 have regions of relatively high outcomes. Evaluations in such regions lead to high error rates, which negatively affect the overall MFE, as can be seen in Figure 4. Optimizers which tend to more exploratory

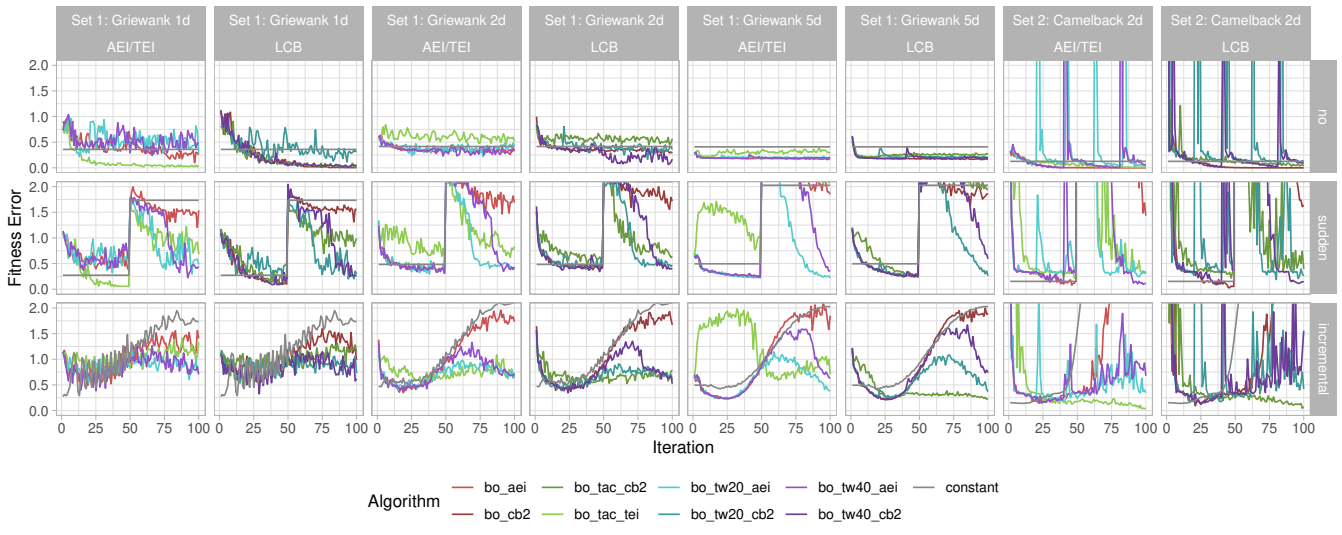

Figure 4: Optimization curves for a subset of the problems. Two columns form a pair, with optimizers that use the `aei` or `tei` acquisition function on the left and `cb2`-based optimizers on the right. Each row represents a different drift scenario.

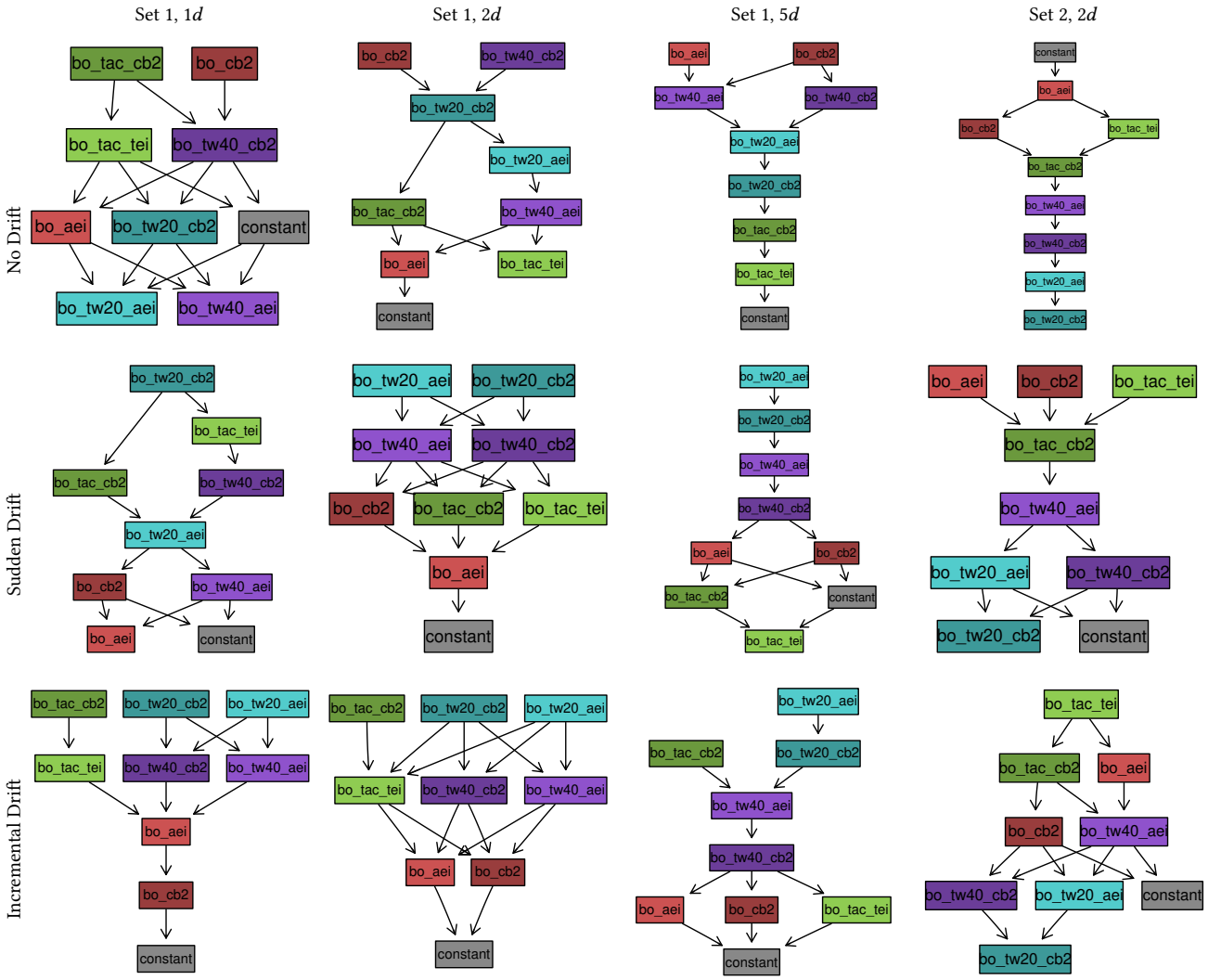

Figure 5: The preference graphs for each group of problems illustrate which optimizers were able to beat each other tested on 50 repititions across 3 (Set 1) and 2 (Set 2) problems.

evaluations are potentially punished more on Set 2. On the other hand those exploratory evaluations are necessary to adapt to concept drifts. For problems with incremental drift, `bo_tac_cb2` seems to be able to balance exploration end exploitation and the difference of the average ranks to the first are relatively small.

The performance of the different acquisition functions for each optimizer show, that its choice rarely is crucial for the average performance. This can also be seen in the preference plots in Figure 5. Here the same optimizer with different acquisition functions are on a similar level with few exceptions. For the time-as-covariate approach the LCB appears to be the better choice and for the window approaches we see a slight advantage for the AEI criterion.

6 SUMMARY

The benchmark compared two new Model-based optimization approaches for dynamic optimization problems (DOPs) against two baselines. The *window approach* reduces the design of the surrogate to observations within a recent time frame, so that it models the current state of the function. The *time-as-covariate approach* includes the time as an additional covariate in the surrogate, which allows the surrogate to directly model the influence of the time on the function outcomes. Both approaches were benchmarked with the lower confidence bound acquisition function and either the augmented expected improvement or the temporal expected improvement.

We constructed a benchmark that covers established benchmark functions under three different drift scenarios (no drift, sudden drift and incremental drift) and can easily be extended. We defined a

Table 2: Average rankings of the optimizers on each subset of problems for each drift scenario. For each problem group the rank of the average ranks is given in brackets. Results that are not significantly different from the best optimizer are underlined.

			AEI				CB2				
Drift	Set	d	bo_aei	bo_tac_tei	bo_tw20_aei	bo_tw40_aei	bo_cb2	bo_tac_cb2	bo_tw20_cb2	bo_tw40_cb2	constant
no	Set 1	1	5.99 (6)	3.23 (3)	7.55 (9)	7.31 (8)	**2.77** (1)	2.94 (2)	6.05 (7)	3.34 (4)	5.81 (5)
		2	6.28 (7)	7.01 (9)	4.79 (4)	5.15 (5)	2.83 (2)	5.41 (6)	4.13 (3)	**2.74** (1)	6.66 (8)
		5	3.15 (2)	7.99 (9)	4.48 (5)	3.57 (4)	**2.56** (1)	6.88 (7)	5.45 (6)	3.24 (3)	7.68 (8)
	Set 2	2	2.43 (2)	3.29 (3)	7.61 (8)	5.93 (6)	3.32 (4)	4.71 (5)	8.81 (9)	6.83 (7)	**2.06** (1)
sudden	Set 1	1	8.05 (9)	2.88 (2)	4.77 (5)	5.94 (6)	6.73 (7)	3.35 (4)	**2.24** (1)	3.30 (3)	7.74 (8)
		2	6.91 (8)	5.89 (6)	**1.92** (1)	4.43 (4)	6.33 (7)	5.51 (5)	2.03 (2)	4.13 (3)	7.85 (9)
		5	5.27 (6)	7.94 (9)	**1.99** (1)	3.64 (3)	5.15 (5)	7.01 (8)	2.61 (2)	4.39 (4)	7.01 (7)
	Set 2	2	4.28 (4)	**3.86** (1)	5.48 (7)	4.88 (5)	4.21 (2)	4.22 (3)	6.79 (9)	5.47 (6)	5.81 (8)
incremental	Set 1	1	6.66 (7)	4.07 (5)	3.37 (2)	4.05 (4)	7.19 (8)	4.04 (3)	**3.12** (1)	4.17 (6)	8.34 (9)
		2	7.20 (7)	4.18 (4)	**2.94** (1)	4.25 (6)	7.46 (8)	3.56 (3)	3.17 (2)	4.19 (5)	8.06 (9)
		5	6.65 (8)	5.97 (6)	**2.65** (1)	4.21 (4)	6.46 (7)	3.76 (3)	3.09 (2)	4.37 (5)	7.83 (9)
	Set 2	2	4.02 (4)	**2.61** (1)	6.20 (7)	4.65 (5)	4.00 (3)	3.20 (2)	7.96 (9)	6.21 (8)	6.15 (6)

performance measure that counts every evaluation towards the average error, which is uncommon for DOPs, where usually just the best of a batch is taken into consideration.

The results show that in scenarios with drift the proposed MBO extensions are able to beat the drift-unaware baselines. However, none of the proposed optimizers is a clear winner across all scenarios and each method has its strengths and weaknesses. For more complex functions and functions of higher dimensionality the window-based MBO approach appears to be the most promising candidate. The time-as-covariate approach performs best on the $1d$ and on simpler $2d$ problems with fewer local minima. A clear difference between the acquisition functions could not be observed. There might be a slight advantage of the AEI criterion for the window approach and an advantage of the LCB for the time-as-covariate approach.

Altogether, the results show that with the proposed approaches MBO can be applied on DOPs in order to obtain online proposals that minimize the overall error. That no method is a clear winner indicates that there is still room for improvement. It remains unclear whether the optimizer performs poorly because the surrogate models the objective function badly or because the acquisition function proposes the wrong points. The results indicate that for some cases the combination of a certain surrogate and an acquisition function that drives exploration leads to bad performance. Here it is necessary to find a better balance of exploration of new areas within the search space and exploitation of estimated optima.

We note that the defined error measure influences how a drift-aware MBO algorithms should be designed. In the current setting each evaluation of the objective counts towards the mean fitness error. This punishes exploration since every evaluation that is exploratory but does not lead to a detection of a drift has a negative impact on the error. We could observe this behavior by comparing scenarios with and without drift for a certain optimizer. The fitness error went up in case of no drift because of exploration but stayed low in case of an incremental drift because the exploration led to better configurations. Therefore, one would have to fine tune the threshold that steers exploration vs. exploitation (e.g. the λ parameter of the LCB acquisition function) for each DOP individually. In contrast, exploration is less harmful when we evaluate static optimizers by looking at the one best found configuration. If we define our problem so that not each evaluation counts towards the error we could adapt the optimizers in such a way that non-counting evaluations are used for exploration.

Regarding the window approach, the smaller window size was always beneficial for the presented problems, except if no drift occurred. This motivates to use adaptive window sizes. For online machine learning it is common to detect concept drifts with a method that is independent of the machine learning method itself, e.g. by detecting increasing residuals. Such external concept drift detectors could help to determine the optimal window size for the incremental drift or trigger a deletion of the outdated design if a sudden drift is detected.

The proposed window approach is restricted to changes over time but for the time-as-covariate approach the time is symbolic for any external parameter that can be measured but not controlled. However, this feature is not further studied in this paper.

We have to highlight that the benchmarks were restricted to problems where the optimal value stays unchanged and only the position of the optimum changes. This makes it easier to detect unwanted behaviors of the optimizer, since an increasing error directly implies that the evaluated values move away from the optimum. However, the benchmark can be easily extended to a scenario where also the optimal value changes. This shift likely poses a completely new challenge for the optimizers.

ACKNOWLEDGMENTS

This work was partly supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876, A3.

REFERENCES

[1] Bernd Bischl, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. 2017. mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions. *arXiv:1703.03373 [stat]* (March 2017), 1–26. arXiv:stat/1703.03373

[2] Jakob Bossek. 2017. Smoof: Single- and Multi-Objective Optimization Test Functions. *The R Journal* 9, 1 (2017), 103–113.

[3] Carlos Cruz, Juan R. González, and David A. Pelta. [n.d.]. Optimization in Dynamic Environments: A Survey on Problems, Methods and Measures. 15, 7 ([n. d.]), 1427–1448. https://doi.org/10.1007/s00500-010-0681-0

[4] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. 7 (2006), 1–30. Issue Jan. http://www.jmlr.org/papers/v7/demsar06a.html

[5] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* 46, 4 (March 2014), 44:1–44:37. https://doi.org/10.1145/2523813

[6] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. [n.d.]. Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. 34, 3 ([n. d.]), 441–466. https://doi.org/10.1007/s10898-005-2454-3

[7] Donald R. Jones. 2001. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization* 21, 4 (2001), 345–383. https://doi.org/10.1023/A:1012771025575

[8] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* 13, 4 (Dec. 1998), 455–492. https://doi.org/10.1023/A:1008306431147

[9] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. 2017. A Survey of Swarm Intelligence for Dynamic Optimization: Algorithms and Applications. *Swarm and Evolutionary Computation* 33 (April 2017), 1–17. https://doi.org/10.1016/j.swevo.2016.12.005

[10] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. 2012. Evolutionary Dynamic Optimization: A Survey of the State of the Art. *Swarm and Evolutionary Computation* 6 (Oct. 2012), 1–24. https://doi.org/10.1016/j.swevo.2012.05.001

[11] Favour M. Nyikosa, Michael A. Osborne, and Stephen J. Roberts. 2018. Bayesian Optimization for Dynamic Problems. *arXiv:1803.03432 [stat]* (March 2018). arXiv:stat/1803.03432

[12] Olivier Roustant, David Ginsbourger, and Yves Deville. 2012. DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *Journal of Statistical Software, Articles* 51, 1 (2012), 1–55. https://doi.org/10.18637/jss.v051.i01