A Simple Yet Effective Approach to Robust Optimization Over Time

Lukáš Adam and Xin Yao

***Abstract*—Robust optimization over time (ROOT) refers to an optimization problem where its performance is evaluated over a period of future time. Most of the existing algorithms use particle swarm optimization combined with another method which predicts future solutions to the optimization problem. We argue that this approach may perform subpar and suggest instead a method based on a random sampling of the search space. We prove its theoretical guarantees and show that it significantly outperforms the state-of-the-art methods for ROOT.**

***Keywords*—Dynamic optimization; Robust optimization; Robust optimization over time; Uniform sampling; Particle swarm optimization**

I. Introduction

Classical optimization problems involve minimizing or maximizing a function f over a region X. Often, these problems depend on time t and random variables (also called environments) $\boldsymbol{\alpha}(t)$. These problems may be written as

$$\underset{x\in X}{\text{maximize}}\, f(\boldsymbol{x};\boldsymbol{\alpha}(t)). \tag{1}$$

We focus on the case where at time t only the history of $\boldsymbol{\alpha}(t)$ is known and where there is no information about its future distribution. Moreover, the objective may be accessed only via black-box evaluations without knowing the exact value of $\boldsymbol{\alpha}(t)$. The goal is to find the optimal solution to (1). Since the computation budget is limited, the solution at the current time should be found with the help of function evaluations at previous times.

The setting above describes the "solution tracking" where the solution may be recomputed and changed at every time instant. However, this is often not desirable or even impossible as a reimplementation of a solution may be physically impossible or may cause additional costs or inconvenience to users.

Another approach was proposed in [1] where the emphasis is not given to the performance up to the current time but over a future time period. Thus, the solution does not have to perform exceptionally well at present but it has to perform satisfactorily over time. The authors named this problem Robust optimization over time (ROOT).

This work was supported by National Natural Science Foundation of China (Grant No. 61850410534), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Peacock Plan (Grant No. KQTD2016112514355531), and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008),

Both authors are with Shenzhen Key Laboratory of Computational Intelligence, University Key Laboratory of Evolving Intelligent Systems of Guangdong Province, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. Email: adam@utia.cas.cz (corresponding author), xiny@sustech.edu.cn

A good ROOT solution should show a good performance in at least one of the two main performance criteria [2]. The first one is the average performance over a future time interval while the second one counts how long a solution performs better than a given threshold.

In this paper, we follow two goals. First, we propose a novel method. While the current state-of-the-art methods use a modification of particle swarm optimization, we propose to uniformly sample the search space and then improve the best point by a local search. The uniform search has the advantage that it gives theoretical bounds for the solution quality. Moreover, if the problem dimension is low, the sampled points may be the same for all time instants. This allows using prediction algorithms without having to reevaluate the functions at previous time instants.

Second, the ROOT papers usually did not describe the parameter initialization, boundary conditions or dynamics properly. They contain confusing notations and even plain mistakes. We conjecture that even though all ROOT papers used the same modified moving peak benchmark, they solved different problems due to different parameter setting. At the same time, the papers usually did not propose a comparison with the basic benchmark: the solution which performs best at the current time and ignores the future. We try to remedy this situation by describing the benchmark properly, showing a proper comparison with a basic solution approach, and by providing our codes online so that any inconsistency can be immediately clarified.[1]

The paper is organized as follows: the introduction is c6oncluded by a short literature survey. In Section II we propose our novel method and in Section III we try to codify the benchmark problems. Section IV consists of the numerical part. To keep the paper as clear as possible, multiple results were moved to the Appendix.

II. A simple approach to ROOT

In this section, we provide a literature overview, specify the problem formulation, propose a solution method and perform its basic analysis.

A. Literature overview

There are numerous alternatives to approaching (1). Stochastic optimization [3] maximizes f in expectation while robust optimization [4] maximizes it in worst-case. Dynamic optimization [5] models the evolution via an ordinary differential

[1] https://github.com/sadda/ROOT-Benchmark

equation while multi-stage programming [6] generalizes the stochastic optimization by considering a longer horizon. All of these fields assume the knowledge of the distribution of $\boldsymbol{\alpha}(t)$ and they are computationally rather expensive.

Concerning the literature overview for ROOT, [1] was the first paper to propose the ROOT problem. This paper did not consider any numerical results. [7] suggested new metrics requiring the knowledge of the optimal solution and tried to formalize the benchmark problem. [8] suggested the survival metric where the optimal solution does not need to be known. [9] investigated predicting the future by autoregressive series. [10] considered ROOT as a bi-objective problem of maximizing the survival time and the average future fitness. [2] provided a new benchmark with known solutions. [11] proposed a new method based on multi-swarm particle optimization. [12] investigated several methods for predicting future solutions. [13] proposed new techniques to predict future solutions and provided extensive literature overview and numerical study.

B. Problem formulation

We consider the time discrete ROOT problem, where we need to solve (1) for all $t \in \{1, \dots, T\}$. We consider a rather general case where at time t we can evaluate the objective value $f(\boldsymbol{x}, \boldsymbol{\alpha}(t))$ for any query point $\boldsymbol{x}$. We do not know the exact value of $\boldsymbol{\alpha}(t)$ or its future distribution but we can make use of all queries (function evaluations) from previous time instants $1, \dots, t-1$.

To evaluate the solution $\boldsymbol{x}(t)$ quality at time t, we consider two metrics

$$\begin{aligned} F_{\text{aver}}(\boldsymbol{x}(t); t) &= \frac{1}{S} \sum_{s=0}^{S-1} f(\boldsymbol{x}(t); \boldsymbol{\alpha}(t+s)), \\ F_{\text{surv}}(\boldsymbol{x}(t); t) &= \min\{s \geq 0 \mid f(\boldsymbol{x}(t); \boldsymbol{\alpha}(t+s)) \leq f^*\}. \end{aligned} \tag{2}$$

The averaged objective metric F_{aver} measures the average from the future S values while the survival metric F_{surv} measures how long the objective stays above a threshold f^*. Note that both metrics make use of the objective function f at the current time (which can be evaluated) and at the future times (which can be only predicted).

A word of caution is needed here. The future values $\boldsymbol{\alpha}(t+s)$ in (2) are considered to be fixed but not known. In the field of stochastic optimization [3] this amounts to adding expectation with respect to $\boldsymbol{\alpha}$ to (2). Since in the numerical section, we will average the results with respect to different realizations of $\boldsymbol{\alpha}$, we should technically add this expectation to (2) as well. The key difference is that stochastic optimization assumes the future distribution to be known while we assume it to be unknown.

C. Proposed methods

Most of the existing methods for ROOT are based on particle swarm optimization. These papers do not provide any convergence proofs and require hyperparameter tuning. In this section, we propose two very simple methods which do not suffer from these issues. The first one solves (1) at the current time t without considering the past or the future while the second one tries to obtain a robust solution. Note that at every time instant t, we have the computational budget of N_{eval} evaluations of $f(\cdot; \boldsymbol{\alpha}(t))$.

The first method spends N evaluations on a global search and $N_{\text{loc}} = N_{\text{eval}} - N$ evaluations on a local search. The global search is performed by a uniform discretization of the search space into $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ and evaluating $f_n(t) = f(\boldsymbol{x}_n; \boldsymbol{\alpha}(t))$ for all $n = 1, \dots, N$. Then we find the index $n_{\max}$ where $f_n(t)$ has the maximal value and improve $\boldsymbol{x}_{n_{\max}}$ by any local search method within N_{loc} function evalutions. We provide a summary in Algorithm II.1.

Algorithm II.1 Hybrid uniform sampling and local search method for solving ROOT

Input: Number of function evaluation N_{eval}, number of function evaluations for the local search N_{loc}

Set $N \leftarrow N_{\text{eval}} - N_{\text{loc}}$
Discretize the search space X into $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$
for $t = 1, \dots, T$ **do**
    Evaluate $f_n(t) \leftarrow f(\boldsymbol{x}_n; \boldsymbol{\alpha}(t))$ for $n = 1, \dots, N$
    Find the index $n_{\max}$ with maximal value of $f_n(t)$
    Improve $\boldsymbol{x}_{n_{\max}}$ by local search in N_{loc} function evaluations to obtain optimal solution $\boldsymbol{x}_{\text{opt}}(t)$
end for
return $(\boldsymbol{x}_{\text{opt}}(1), \dots, \boldsymbol{x}_{\text{opt}}(T))$

The second method spends all N_{eval} evaluations on a global search. Again, we uniformly discretize the search space into $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_{\text{eval}}}\}$ and evaluate $f_n(t) = f(\boldsymbol{x}_n; \boldsymbol{\alpha}(t))$ for all $n = 1, \dots, N_{\text{eval}}$. The robust solution is selected by any method which takes into account the function values at a neighborhood or at previous time instants. Since the space discretization is the same at every time, besides $f_n(t)$ we also know $f_n(t-1), \dots, f_n(1)$ from previous iterations and we do need to invest any additional function evaluations. Thus, we may apply most of the methods from other ROOT papers for free. We provide a summary in Algorithm II.2.

Algorithm II.2 Uniform sampling method for solving ROOT

Input: Number of function evaluation N_{eval}

Discretize the search space X into $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N_{\text{eval}}}$
for $t = 1, \dots, T$ **do**
    Evaluate $f_n(t) \leftarrow f(\boldsymbol{x}_n; \boldsymbol{\alpha}(t))$ for $n = 1, \dots, N_{\text{eval}}$
    Based on $f_n(t), f_n(t-1), \dots$ for $n = 1, \dots, N_{\text{eval}}$ find robust solution $\boldsymbol{x}_{\text{rob}}(t)$
end for
return $(\boldsymbol{x}_{\text{rob}}(1), \dots, \boldsymbol{x}_{\text{rob}}(T))$

If the search space is $X = [x_{\min}, x_{\max}]^D$, then Appendix A implies that the procedure from Algorithm II.1 gives a solution which is optimal with the following bound

$$f(\boldsymbol{x}_{\text{opt}}(t); \boldsymbol{\alpha}(t)) \geq f^*(t) - \frac{L\sqrt{D}(x_{\max} - x_{\min})}{2(N^{\frac{1}{D}} - 1)}, \tag{3}$$

where $f^*(t)$ is the optimal solution at time t and L is the so-called Lipschitz constant of $f(\cdot; \boldsymbol{\alpha}(t))$. Since most ROOT

methods were tested for the two-dimensional case $D = 2$, the previous bound is rather tight. The solution quality is further improved by the local search.

We would like to summarize the benefits of our approach:

1) Equation (3) gives a guaranteed bound for the solution quality.
2) Since the same points are evaluated at all time instants, using any tracking or prediction mechanism from other ROOT papers requires no additional function evaluations.

III. Numerical benchmarks

In this section, we describe the moving peak benchmark commonly used in the ROOT literature. It is based on [14] and appeared in many papers [2], [7], [8], [9], [10], [11], [12], [13]. However, to the best of our knowledge, no complete and proper description was given in any of these papers. Since, as we will show later, even a small change in the problem setting may have a large impact on the optimal solution, we try to provide a rigorous statement of the benchmark problems.

A. Moving peaks benchmark 1

This benchmark considers M peaks of conic shape in $\mathbb{R}^D$. Peak m has center $\boldsymbol{c}^m$, height h^m and width w^m. Defining the random vector $\boldsymbol{\alpha} = (\boldsymbol{c}^m, h^m, w^m)_{m=1}^M$, the objective function

$$f_t^1(\boldsymbol{x}; \boldsymbol{\alpha}(t)) = \max_{m=1,\dots,M} (h_t^m - w_t^m \|\boldsymbol{x} - \boldsymbol{c}_t^m\|_{l_2}),$$

measures that the height of maximal peak at $\boldsymbol{x}$. We use the shortened notation $h_t = h(t)$.

The dynamics of the random vector is given by

$$\begin{aligned} h_{t+1}^m &= h_t^m + \sigma_h^m \cdot N(0,1), \\ w_{t+1}^m &= w_t^m + \sigma_w^m \cdot N(0,1), \\ \boldsymbol{c}_{t+1}^m &= \boldsymbol{c}_t^m + \boldsymbol{v}_{t+1}^m, \\ \boldsymbol{v}_{t+1}^m &= s^m \frac{(1-\lambda)\boldsymbol{r}_{t+1}^m + \lambda \boldsymbol{v}_t^m}{\|(1-\lambda)\boldsymbol{r}_{t+1}^m + \lambda \boldsymbol{v}_t^m\|}. \end{aligned} \tag{4}$$

Here, $N(0,1)$ denotes the normal distribution with zero mean and unit variance, $\boldsymbol{r}_t^m$ follows the uniform distribution on the D-dimensional sphere with radius s^m and $\sigma_h^m \geq 0$, $\sigma_w^m \geq 0$ and $\lambda \in [0,1]$ are fixed parameters. The peak height h_{t+1}^m differs from the previous height h_t^m by a random number drawn from the normal distribution with zero mean and standard deviation σ_h^m. Similar holds true for the widths. The center $\boldsymbol{c}_{t+1}^m$ moves from $\boldsymbol{c}_t^m$ by vector $\boldsymbol{v}_{t+1}^m$. If $\boldsymbol{v}_1^m$ has norm s^m, then we have

$$\begin{aligned} \lambda = 0 &\implies \boldsymbol{v}_{t+1}^m = \boldsymbol{r}_{t+1}^m, \\ \lambda = 1 &\implies \boldsymbol{v}_{t+1}^m = \boldsymbol{v}_t^m. \end{aligned}$$

Thus, $\lambda = 0$ implies that the movement of the peak centers is random while $\lambda = 1$ implies that the movement is constant in direction $\boldsymbol{v}_1^m$. In both cases the distance between the previous and new centers is s^m.

The random variables have their bounds. We require $h_t^m \in [h_{\min}, h_{\max}]$ and $w_t^m \in [w_{\min}, w_{\max}]$. The bounds for the centers $\boldsymbol{c}_t^m \in [x_{\min}, x_{\max}]^D$ are the same as for the search space. If the dynamics (4) pushes some variable out of its corresponding bounds, we project (clip) it back.

Finally, for initialization of (4) we need to know the initial centers $\boldsymbol{c}_0^m$, heights h_0^m, widths w_0^m and the initial speeds $\boldsymbol{v}_0^m$. Following previous papers, we initialize the centers randomly in the search space $[x_{\min}, x_{\max}]^D$, the heights and widths to some known values h_{init} and w_{init}, respectively and the initial speed is generated randomly at the D-dimensional sphere with radius s^m.

Note that in the literature there are some differences which we summarize in Appendix B.

B. Moving peaks benchmark 2

The second benchmark problem was defined in [2] by the objective

$$f_t^2(\boldsymbol{x}; \boldsymbol{\alpha}(t)) = \frac{1}{D} \sum_{d=1}^{D} \max_{m=1,\dots,M} \Big(h_t^{m,d} - w_t^{m,d} |x^d - c_t^{m,d}| \Big),$$

The upper index d denotes the d^{th} component of a vector. Then the D-dimensional problem can be decomposed into D one-dimensional problems. Moreover, since the heights are different in each dimension, the problem does not technically handle moving peaks anymore.

The authors in [2] considered several dynamics, we will mention only the one most similar to (4), namely

$$\begin{aligned} h_{t+1}^{m,d} &= h_t^{m,d} + \sigma_h^m \cdot N(0,1), \\ w_{t+1}^{m,d} &= w_t^{m,d} + \sigma_w^m \cdot N(0,1), \\ \boldsymbol{c}_{t+1}^m &= R(\theta_t^{D-1}, \dots, \theta_t^1) \boldsymbol{c}_t^m, \\ \theta_{t+1}^d &= \theta_t^d + \sigma_\theta \cdot N(0,1). \end{aligned} \tag{5}$$

The dynamics for the heights and widths are the same as in the first benchmark (4). The center are rotated based on the rotation matrix $R(\theta_t^{D-1}, \dots, \theta_t^1) = R^{D-1}(\theta_t^{D-1}) \dots R^1(\theta_t^1)$, where each rotation matrix $R^d(\theta_t^d)$ performs the rotation in the d-$(d+1)$ plane by angle θ_t^d.

We handle the technicalities similarly as for the first benchmark. If the variables get out of bounds, we project them back. We initialize the centers randomly in the search space $[x_{\min}, x_{\max}]^D$. Based on [2] the initial heights and widths and generated randomly from their bounds. However, the initial θ_1^d is set to θ_{init}.

IV. Experimental results

In this section, we describe the performance of our methods from Section II on the benchmarks from Section III. All displayed results are averaged over 5000 independent simulations of $\boldsymbol{\alpha}$.

A. Parameter setting

In Table I we show the used parameters. We first generated the random evolution of $\boldsymbol{\alpha}$ and then uniformly discretized the search space $[x_{\min}, x_{\max}]^D$ into $N_{\text{eval}} = 2500$ points. Algorithm II.1 randomly selected 2300 of these 2500 points at each t, evaluted $f(\cdot, \boldsymbol{\alpha}(t))$, selected the best value and invested the remaining 200 function evaluation into the local search made by the Matlab built-in function `fmincon`. Algorithm II.2 evaluated all 2500 points and replaced the function value

at a point by the average of all neigboring values with the maximal distance of 3 (points outside of search space were ignored). The solution with the highest average was deemed to be robust.

Table I
PARAMETER VALUES FOR BENCHMARK PROBLEMS

Parameter	Benchmark 1	Benchmark 2
N_{eval}	2500	2500
M	5	25
D	2	2
λ	{0,1}	-
$[x_{\min}, x_{\max}]$	$[0, 50]$	$[-25, -25]$
$[h_{\min}, h_{\max}]$	$[30, 70]$	$[30, 70]$
$[w_{\min}, w_{\max}]$	$[1, 12]$	$[1, 13]$
$[\theta_{\min}, \theta_{\max}]$	-	$[-\pi, \pi]$
σ_h	$U(1, 10)$	5
σ_w	$U(0.1, 1)$	0.5
σ_θ	-	1
h_{init}	50	$U(h_{\min}, h_{\max})$
w_{init}	6	$U(w_{\min}, w_{\max})$
θ_{init}	-	0

Even though it is possible to implement predicting future values by using function evaluations at previous time instants, we decided not to do so. The reason is that even this basic method significantly outperforms the state-of-the-art algorithms and adding the predictions could cloud the basic idea.

B. Numerical results

We compare three methods. *Mesh* and *Time-optimal* are based on Algorithm II.1 with the difference that *Mesh* does not perform the local search. *Robust* is based on Algorithm II.2. Numerical details are described in Section IV-A.

We compare the *Time-optimal* method to known results in Table III. On Benchmark 1 with $\lambda \in \{0, 1\}$ and Benchmark 2 we show the averaged objective F_{aver} with time window $S \in \{2, 6\}$ and the survival function F_{surv} with $\delta \in \{40, 50\}$; both defined in (2). We used the horizon $T = 100$ and the results shown are averages for all time instants with $t \in [20, 100]$. For all benchmarks and evaluation criteria, our results are significantly better than the best-known results. We comment more on how we collected the best-known results in Appendix C.

We can even show that our results are almost optimal. Consider Benchmark 1 with $\lambda = 0$. Discussion in Appendix D shows that the optimal solution has the expected value of approximately 65. Since the peak moves with stepsize $s^m = 1$ and the average width is 6.5, the objective drops to $65 - 6.5 = 58.5$ for the next time instant. But this gives the expected objective $\frac{1}{2}(65 + 58.5) = 61.75$ for $S = 2$ to which our value 61.13 from Table III is very close.

This intuition is confirmed in Table II where we show the gap between the optimal objective and the objective found. *Mesh* shows approximately half of the theoretical gap (3) while this gap is almost zero when we improve it by the local search via *Time-optimal*. This means that *Time-optimal* found the centre of the highest peak. We would like to stress that the information about the highest peak was not used during the optimization and we used it only a posteriori for evaluating performance.

Table II
GAP BETWEEN THE BEST POSSIBLE OBJECTIVE f_t^* AND THE OBJECTIVE FOUND BY OUR METHODS

	Maximal gap (3)	*Mesh*	*Time-optimal*	*Robust*
Benchmark 1	4.69	2.18	0.09	5.38
Benchmark 2	5.05	0.99	0.15	4.38

Tables II and III also suggest why other methods performed subpar:

1) Since the *Time-optimal* solution lies in the peak centre, it is a natural candidate for the robust solution as well. We believe that the commonly used particle swarm optimization was far away from the peak centre.
2) While incorporating objective tracking, the previous papers needed to reevaluate the point at previous time instants. This reduced the number of investigated points.

Note that as explained at the end of Section II, our methods do not suffer from these problems.

We show additional results for Benchmarks 1 and 2 in Figures 3 and 4, respectively. For Benchmark 1 the columns show the results for $\lambda = 0$ (left) and $\lambda = 1$ (right) while for Benchmark 2 the columns show the random generation of initial centers (left) or the grid generation described in Appendix C (right). We can observe the following phenomena:

- The method with local search *Time-optimal* outperforms the method without the local search *Mesh* in all cases.
- The survival time for *Robust* is better than for *Time-optimal* only for one benchmark.
- The survival time is stable for Benchmark 2 while it increases with increasing time for Benchmark 1. The reason is that Benchmark 1 initializes the peak heights to 50 while Benchmark 2 initializes them randomly in $[30, 70]$. Thus, for the former case, the maximal peak height is much smaller for the initial time instants.
- The initialization or parameters have a large impact on the solution (comparison of left and right columns).
- Benchmark 2 is not affected by the boundary conditions for variables. This does not hold for Benchmark 1 where the survival time increases as the centres hit the boundary and stay there.

To summarize, the *Time-optimal* method, which does not utilize any tracking or future predictions, performs very well on both benchmarks. This raises the question of whether the moving benchmark problem is suitable for ROOT.

V. CONCLUSION

In this paper, we gave a proper description of the moving benchmark problem for ROOT and proposed a simple method to solve it. Our method significantly outperforms other methods. Since we believe that there are multiple deficiencies in

Table III
COMPARISON OF BEST KNOWN AND OUR RESULTS. ALL METHODS USE 2500 FUNCTION EVALUATIONS AT EACH TIME INSTANT. THE PROCESS OF COLLECTING THE BEST KNOWN RESULTS IS DESCRIBED IN APPENDIX C. ALL EXPERIMENTS WERE REPEATED 5000 TIMES.

Setting	From	Best known result				Our result			
		F_{aver}		F_{surv}		F_{aver}		F_{surv}	
		$S=2$	$S=6$	$\delta=40$	$\delta=50$	$S=2$	$S=6$	$\delta=40$	$\delta=50$
Benchmark 1 with $\lambda=1$	[8]	53.48	8.82	3.02	1.69	63.32	58.76	13.72	10.11
Benchmark 1 with $\lambda=0$	[9] [11]	-	-	8.35	4.25	61.13	54.77	10.42	5.91
Benchmark 2	[2] [12]	48.88	40.58	1.35	1.02	62.21	57.58	16.54	6.38

most ROOT papers, we suggest that the papers on ROOT should include the following information to facilitate further comparisons and analyses of proposed algorithms:

1) **Proper problem description**. Including parameters, special setting and initial conditions. This is needed for other authors to repeat the experiments.
2) **Codes available online**. When it is not possible to describe everything, codes online help significantly.
3) **Fair comparison**. In some papers, a comparison was done with different parameter setting. Including higher computational budget.
4) **Higher number of repetitions**. When the experiment is repeated 20 or 30 times as in most papers, the graphs are not smooth and it may be difficult to extract useful information from them.
5) **Comparison with a basic method**. Sometimes a simple solution (centre of the highest peak) performs well in a more complicated setting (robust solution).

Note that most papers investigated in this manuscript violated all these topics mentioned above.

APPENDIX

In the Appendix, we provide further technical results that support the main text.

A. Estimate on solution quality

We recall first two definitions. We say that a function g is Lipschitz on X with constant L if

$$|g(\boldsymbol{x}) - g(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$$

for all $\boldsymbol{x}, \boldsymbol{y} \in X$. We say that $\{\boldsymbol{x}^1, \dots, \boldsymbol{x}^S\}$ is δx-cover of X if for each $\boldsymbol{x} \in X$ there is some $s \in \{1, \dots, S\}$ such that $\|\boldsymbol{x} - \boldsymbol{x}^s\| \leq \delta x$. Then we have the following lemma.

Lemma A.1. *Consider an optimization problem*

$$\underset{\boldsymbol{x} \in X}{\text{maximize}} \quad g(\boldsymbol{x}), \tag{6}$$

where g is Lipschitz continuous with constant L. Denote $\boldsymbol{x}^1, \dots, \boldsymbol{x}^S$ to be a δx-cover of X and $\hat{\boldsymbol{x}} \in \operatorname{argmax}_{s=1,\dots,S} g(\boldsymbol{x}^s)$ to be the best sampled value. Then $\hat{\boldsymbol{x}}$ is an ε-optimal solution of (6) *in the sense of*

$$g(\hat{\boldsymbol{x}}) \geq \sup_{x \in X} g(\boldsymbol{x}) - L \cdot \delta x.$$

Proof. The existence of the δx-cover and the Lipschitz continuity of g imply that g is bounded from above on X. That means that there is a sequence $\{\boldsymbol{y}^n\}_{n=1}^{\infty} \subset X$ satisfying

$$g(\boldsymbol{y}^n) \geq \sup_{x \in X} g(\boldsymbol{x}) - \frac{1}{n}. \tag{7}$$

Due to the definition of δx-cover, for each n there is some $s(n) \in \{1, \dots, S\}$ such that $\|\boldsymbol{x}^{s(n)} - \boldsymbol{y}^n\| \leq \delta x$. This implies

$$\begin{aligned}\max_{s=1,\dots,S} g(\boldsymbol{x}^s) &\geq g(\boldsymbol{x}^{s(n)}) = g(\boldsymbol{x}^{s(n)}) - g(\boldsymbol{y}^n) + g(\boldsymbol{y}^n) \\ &\geq g(\boldsymbol{y}^n) - L\delta x \geq \sup_{x \in X} g(\boldsymbol{x}) - \frac{1}{n} - L\delta x,\end{aligned}$$

where the second inequality follows from the Lipschitz continuity of g and the last inequality from (7). Since n is arbitrary, the lemma statement follows. □

To apply this to (3), it suffices to realize that uniform sampling with N points form a δx-cover for $[x_{\min}, x_{\max}]^D$ with

$$\delta x = \frac{\sqrt{D}(x_{\max} - x_{\min})}{2(N^{\frac{1}{D}} - 1)}.$$

B. Differences in benchmark problems from other papers

In this section, we comment on small details in the benchmark description. All the mentioned papers wrote $\boldsymbol{r}$ instead of $\boldsymbol{r}_t^m$ in (4). However, since they commented on random movement, we believe that the time-dependence has to be stressed because otherwise, the centres would move in a fixed direction.

The complete problem description also includes what happens when peak height, weight or centre get outside the allowed boundary. While some of the paper described that they are projected back onto the boundary, [7] noted that they are "bounced back", most of the papers did not describe what happens in such a situation. However, this may have a huge impact on the solution.

Most of the papers generated the initial random vector $\boldsymbol{r}_1^m$ by generating all components randomly in $[-1, 1]$ and then normalized the vector into the length of s^m. However, this is not equivalent to generating randomly on the sphere with a radius of s^m. Figure 1 shows the angle between the generated vector and the vector $(1, 0)$ in the two-dimensional case. The approach from the earlier papers gives a much higher chance

for the (normalized) vectors around $(\pm 1, \pm 1)$. The reason is that the square is "bigger" than the circle in these directions.

Finally, [2] initialized the initial centres of 25 peaks by selecting 5 random points in each dimension and then performing Cartesian product. As we show in Figure 4, this yields hugely different results from randomly generating in the domain.

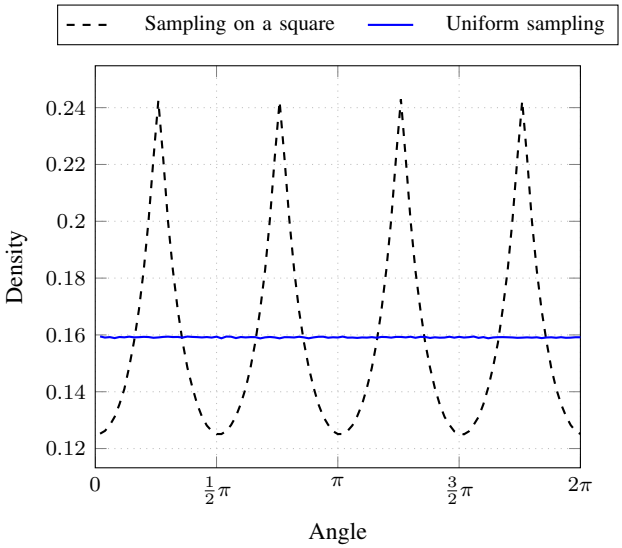

Figure 1. The sampling on a square, used in previous papers, does not result in uniform sampling.

Finally, [11] used a different function count. While the original and our approach recomputed the solution at every time step and then computed its survival based on the future values [11] recomputed the solution only when it dropped below the threshold δ. This resulted in the fact that they used approximately 8 times more function evaluations.

C. Selecting the best known results

In this section we describe how we collected the best known results from Table III. Benchmark 1 with $\lambda = 1$ is taken from [8], Benchmark 1 with $\lambda = 0$ from [11] and Benchmark 2 from [2]. Note that [11] compared himself with the results from [8], [9], [10] and showed that their results are superior. For Benchmark 2 we considered only the random movement which in [2] was denoted as TP_{13}. Finally we did not compare ourself to the promising-looking results from [13] because they used different setting for the stepsize s.

Note that due to the issues described earlier, it may have happened that the setting for our and their papers is different. However, we tried to minimize this possibility.

D. Height of the heighest peak

In Figure 2 we intitialize M peaks with initial heights $h_1^m = h_{\text{init}} = 50$. We apply the dynamics (4) and observe the average height of the highest peak for time instants $t \in [1, 20]$. We see that rather soon the average height stabilizes at 65 for $M = 5$ and close to the maximal value $h_{\max} = 70$ for $M = 25$. This is the optimal value for F_{aver} for $S = 1$.

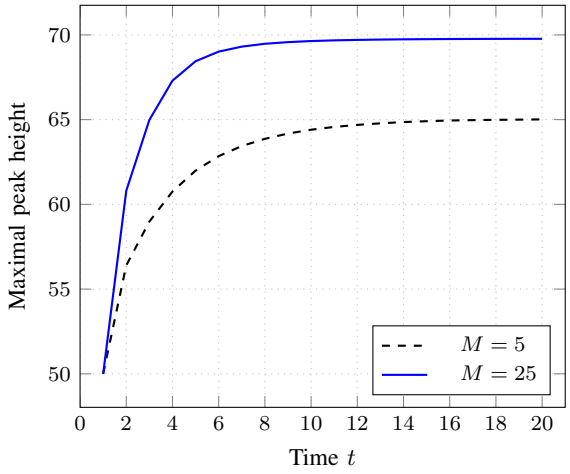

Figure 2. The average height of the heighest of M peaks.

References

[1] X. Yu, Y. Jin, K. Tang, and X. Yao, "Robust optimization over time—a new perspective on dynamic optimization problems," in *IEEE Congress on evolutionary computation*. IEEE, 2010, pp. 1–6.

[2] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Robust optimization over time: Problem difficulties and benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 731–745, 2015.

[3] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[4] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*. Princeton University Press, 2009, vol. 28.

[5] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995, vol. 1, no. 2.

[6] M. V. Pereira and L. M. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical programming*, vol. 52, no. 1-3, pp. 359–375, 1991.

[7] H. Fu, B. Sendhoff, K. Tang, and X. Yao, "Characterizing environmental changes in robust optimization over time," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[8] ——, "Finding robust solutions to dynamic optimization problems," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2013, pp. 616–625.

[9] Y. Jin, K. Tang, X. Yu, B. Sendhoff, and X. Yao, "A framework for finding robust optimal solutions over time," *Memetic Computing*, vol. 5, no. 1, pp. 3–18, 2013.

[10] Y. Guo, M. Chen, H. Fu, and Y. Liu, "Find robust solutions over time by two-layer multi-objective optimization method," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1528–1535.

[11] D. Yazdani, T. T. Nguyen, J. Branke, and J. Wang, "A new multi-swarm particle swarm optimization for robust optimization over time," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 99–109.

[12] P. Novoa-Hernández, D. A. Pelta, and C. C. Corona, "Approximation models in robust optimization over time-an experimental study," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–6.

[13] D. Yazdani, T. T. Nguyen, and J. Branke, "Robust optimization over time by learning problem space characteristics," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 143–155, 2018.

[14] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3. IEEE, 1999, pp. 1875–1882.

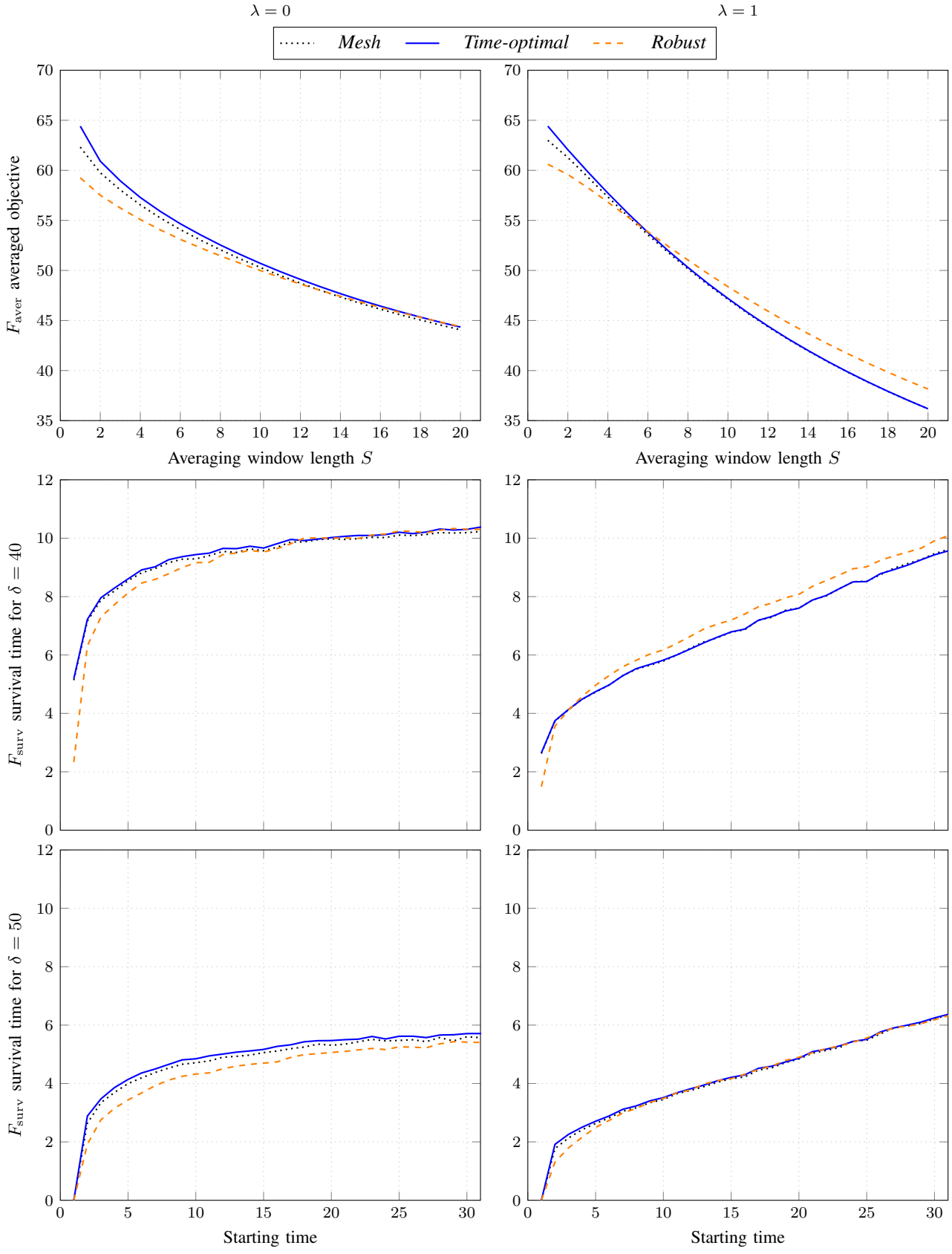

Figure 3. Results for Benchmark 1 with $\lambda = 0$ (left) and $\lambda = 1$ (right). We show the averaged objective F_{aver} as a function of the averaging time window S (top) and the survival function F_{surv} for thresholds $\delta = 40$ (middle) and $\delta = 50$ (bottom). Note that the metrics are defined in (2).

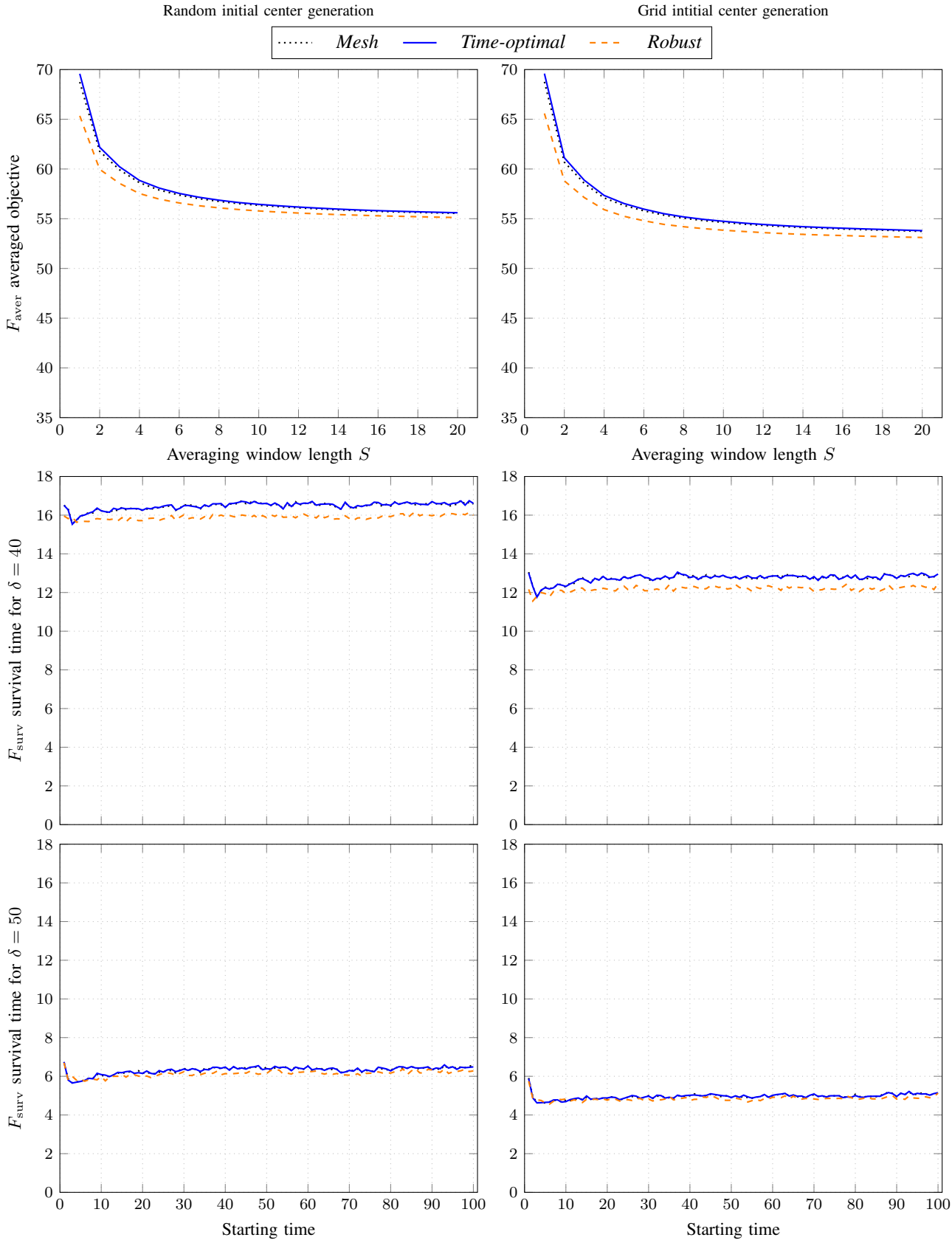

Figure 4. Results for Benchmark 2 with random center generation (left) and the grid center generation described in Appendix C (right). We show the averaged objective F_{aver} as a function of the averaging time window S (top) and the survival function F_{surv} for thresholds $\delta = 40$ (middle) and $\delta = 50$ (bottom). Note that the metrics are defined in (2).