Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

LEARNING TO FORGET: DESIGN OF EXPERIMENTS FOR LINE-BASED BAYESIAN OPTIMIZATION IN DYNAMIC ENVIRONMENTS

Jens Jocqué
Tom Van Steenkiste
Pieter Stroobant
Rémi Delanghe
Dirk Deschrijver
Tom Dhaene

IDlab
Ghent university - imec
Technologiepark-Zwijnaarde 126
Ghent 9052, Belgium

ABSTRACT

Various scientific and engineering fields rely on measurements in 2D spaces to generate a map or locate the global optimum. Traditional design of experiments methods determine the measurement locations upfront, while a sequential approach iteratively extends the design. Typically, the cost of traveling between sample locations can be ignored, for example in simulation experiments. In those cases, the experimental design is generated using a point-based method. However, if traveling towards the next sample location incurs an additional cost, line-based sampling methods are favored. In this setting, the sampling algorithm needs to generate a route of measurement locations. A common engineering problem is locating the global optimum. In certain cases, such as fire hotspot monitoring, the location of the optimum dynamically changes. In this work, an algorithm is proposed for sequentially locating dynamic optima in a line-based setting. The algorithm is evaluated on two dynamic optimization benchmark problems.

1 INTRODUCTION

Measurements in 2D or 3D spaces are ubiquitous in many science and engineering fields. These measurements can be aimed towards assessing the global response surface or towards locating the global optimum. In the first case, the measurements are space-filling and based on exploration of the measurement space whereas the last case is a global optimization problem. Typically, these measurements are performed by a device such as an Unmanned Aerial Vehicle (UAV), Unmanned Ground Vehicle (UGV), or robotic arm. Practical examples are cartography (Gademer et al. 2009), vegetation monitoring (Berni et al. 2009), agricultural field mapping (Valente et al. 2013), 3D mapping (Nex and Remondino 2014), electro-magnetic compatibility (EMC) (Deschrijver et al. 2012), etc.

Design of Experiments (DoE) encompasses a range of methods to determine the order and location of data acquisition which can be either physical measurements or simulations (Kleijnen 2008). Often, the focus of DoE methods is on exploring the entire measurement space, known as space-filling design.

Classic DoE methods are typically one-shot approaches. All sample locations and their order are determined before starting the measurements. There is a clear risk of choosing too many or too few sample locations, which can either result in a large measurement cost or insufficient information respectively. Real-life scenarios such as limited battery life or the possibility of mechanical failure (Carlson and Murphy 2005) enforce the limitations of one-shot approaches. This can be mediated by using a sequential DoE strategy instead of a one-shot approach. Sequential strategies iteratively improve the result by adding

new samples, having the advantage that the location of the following sample point can be improved by interpreting past measurements.

As classic DoE methods are used in simulation experiments or separated physical measurements, the constraint that the UAV, UGV, or robotic arm needs to travel through the design space to perform the measurement is often omitted. To incorporate this constraint, line-based approaches such as the one-shot Boustrophedon path (Choset 2000) or a Hilbert curve (Hilbert 1891) can be used.

The Adaptive Line-Based Sampling TRajectOrieS (ALBATROS) algorithm (Van Steenkiste et al. 2019) is an algorithm that combines this sequential and line-based sampling. This algorithm is a combination of a path planning algorithm and a path sampling algorithm and is a good solution for a space-filling algorithm when the amount of samples is not known upfront.

When the measurements are aimed towards finding a global optimum instead of constructing an accurate representation of the entire measurement space, Bayesian Optimization (BO) (Jones et al. 1998; Mockus 1974) is frequently used. Bayesian optimization is a sequential design strategy for global optimization of black-box functions. Often, the additional assumption is made that it is expensive to measure samples, either due to computational requirements or experiment duration. The typical flow of a BO algorithm is given in Figure 1.

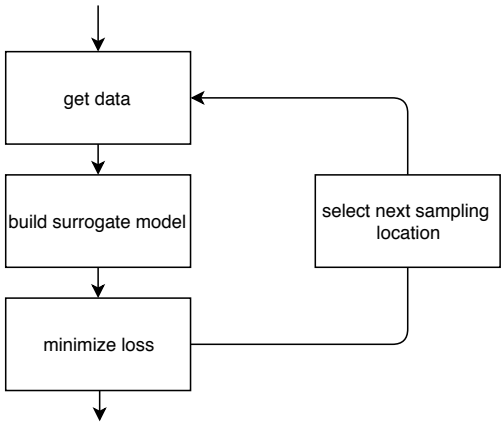
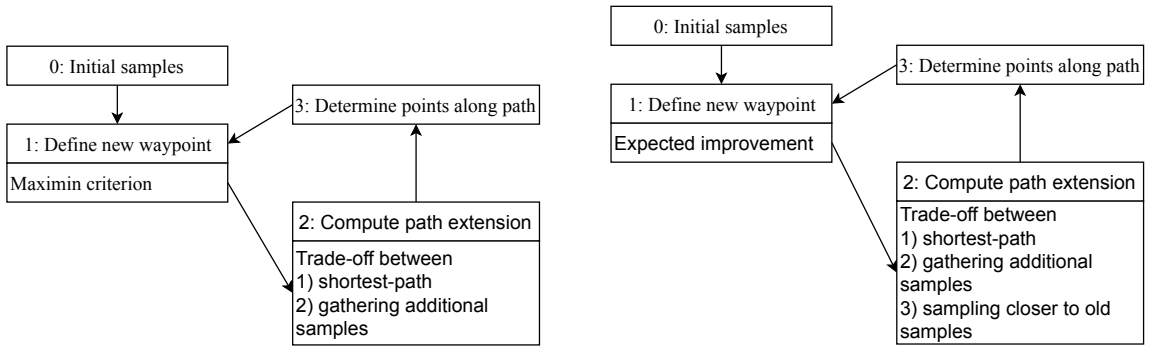


Figure 1: Flow of Bayesian optimization algorithm.

Finding the global optimum with these assumptions becomes even more complex if the objective function changes through time. In literature, this is known as a dynamic optimization problem (DOP) (Floudas et al. 1999). Since many real world problems have a dynamic nature, it is important that DOPs are studied elaborately. An example application is fire detection and monitoring (Martínez-de Dios et al. 2006). To model the dynamics it is necessary for algorithms to incorporate well-considered aging concepts. Measurements that happened earlier, typically contain less information than new measurements. Data aging allows to mathematically model this information loss. Examples are found in intrusion detection systems (Gorodetsky et al. 2005), data stream mining algorithms (Chang and Lee 2005), and time series forecasting (Gardner 2006). The age of a sample is often expressed as data freshness.

In this work, an algorithm is proposed that combines the sequential and line-based approach of DoE methods with the BO solution for finding the optimum in a DOP by extending the ALBATROS algorithm (Van Steenkiste et al. 2019). This enables the algorithm to solve global optimization problems with a dynamic objective function.

This work is organized as follows. In Section 2 the novel algorithm is explained. Next, the experimental setup is discussed in Section 3 and in Section 4 the experiments and the results are explained. Finally, Section 5 concludes the paper and discusses the future work.



(a) ALBATROS (b) novel algorithm

Figure 2: Phases of the ALBATROS algorithm and the proposed algorithm.

2 DYNAMIC ALBATROS ALGORITHM

The proposed algorithm is a sequential line-based BO algorithm capable of finding optima for dynamic objective functions. It is implemented as an extension of the ALBATROS algorithm for 2D measurement spaces. The algorithm requires an initial set of samples that can be obtained through any classic DoE strategy. After initialization, the algorithm starts a sequential loop with identical steps to the ALBATROS algorithm as shown in Figure 2. The first step is the definition of the new waypoint towards which the path will be extended in one sequential iteration step, known as the path extension step. The original algorithm defined the waypoint as the point where the least amount of information had been collected, represented by the maximin criterion (Johnson et al. 1990), to ensure a quasi-uniform distribution of the information. Instead, the novel proposed algorithm builds a spatiotemporal Gaussian Process (GP) and defines the new waypoint using an acquisition function. As a second step, the algorithm computes the path from the current location towards the new waypoint. Here, the original ALBATROS made a trade-off between gathering samples along the way and the cost of the movement. This second step is fine-tuned as well in the new algorithm so that it tends to travel closer towards older samples. The last step of the sequential loop selects where to sample along the path.

For the remainder of this section, the proposed algorithm will be discussed elaborately. First, a brief overview is given of how the concept of time is interpreted by the algorithm. The implementation and simulation of time is vital for the simulation to be realistic. Second, the aging strategy is discussed. Here, it is explained how the model handles recent samples differently from older samples. Next, the initialization phase and the three phases in the sequential loop are discussed in detail.

2.1 Time Interpretation

Dynamic objective functions change through time. A dynamic optimization problem (DOP) was defined by (Cruz et al. 2011) as follows:

$$DOP = \left\{ \begin{array}{ll} \text{optimize} & f(\boldsymbol{x},t) \\ \text{s.t.} & \boldsymbol{x} \in F(t) \subseteq S, t \in T \end{array} \right\}$$

where:

- $S \in R^n$, S is the search space
- t is the time
- $f : S \times T \rightarrow R$, is the objective function that assigns a numerical value $f(\boldsymbol{x},t) \in R$ to each possible solution $\boldsymbol{x} \in S$ at time $t \in T$

- $F(t)$, is the set of feasible solutions $\boldsymbol{x} \in F(t) \subseteq S$ at time t.

A basic assumption of the algorithm is that the computational time is orders of magnitude smaller than the traveling measurement time. This assumption can be justified since BO is well suited to study objective functions that are expensive, either computationally or physically, to evaluate. To accomplish this, it is necessary that the algorithm keeps a notion of time. Our proposed solution is an initialization parameter which determines how long it takes to perform one measurement. This parameter must be chosen with care. Setting the measurement time too high would make it impossible to capture the dynamics of the objective function. Vice versa, setting this time too low could completely remove the time complexity by allowing a lot of measurements in a short time. Note that in real-life applications this parameter is problem specific and can not be chosen.

During the implementation of this algorithm, the continuous character of time was taken into account. This means that subsequent evaluations of the objective functions always happen at different timestamps. Indeed, when using only one measurement device, in real-life applications it is physically impossible to perform multiple measurements at the same timestamp. Existing solutions for DOP, such as evolutionary algorithms and particle swarm optimization are described and compared in (Moser and Chiong 2013). These algorithms do not incorporate this constraint, allowing multiple measurements at the same timestamp. In practice, this is equivalent to measuring with multiple probes simultaneously. Therefore, these algorithms are out of scope of the current work and not included in the comparison.

2.2 Aging Strategy

To model the age of a measurement sample, an exponential smoothing approach is used. A smoothing constant α (Brown et al. 1961) is usually chosen as $0 < \alpha < 1$ and is application specific. This concept, exponential smoothing, has been researched extensively and has been successfully used in the discrete time series forecasting domain. For more information, the reader is referred to (Gardner 2006).

We define exponential aging based on exponential smoothing as follows. When a sample is taken, it receives a sample freshness equal to one. Every time a new sample is taken, the freshness value decreases by $\alpha\%$ with respect to its previous value. This means that for old samples, the freshness value will approach zero. The freshness of a measurement at location $\mathbf{x}$ with value y can be represented mathematically as follows:

$$\text{freshness}(\boldsymbol{x}, y) = (1-\alpha)^i$$

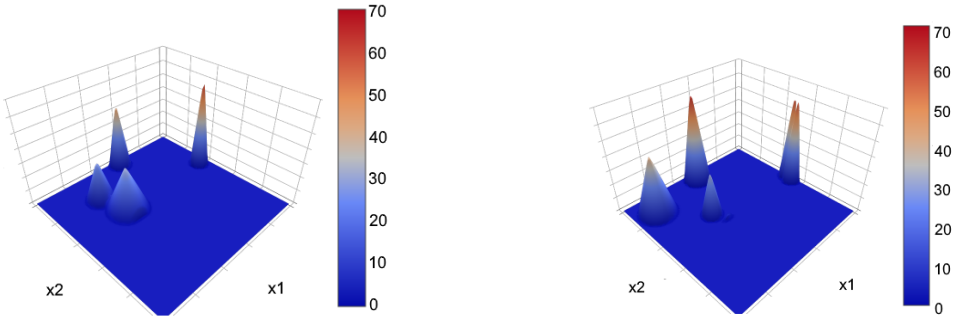
with i the amount of samples taken since $(\boldsymbol{x}, y)$ was sampled. For the most recent sample, $i = 0$.

2.3 Phase 0: Gather Initial Samples

Phase 0 is the initialization phase of the algorithm. A valid collection of initial samples in its most basic form consists of two samples on a line. More profound initialization strategies are possible as well. Well-known examples are the Boustrophedon path (Choset 2000) or Hilbert curve (Hilbert 1891) which are two line-based strategies aimed to cover the measurement space while minimizing the path cost. When a path is obtained using such an approach, samples can be taken along that path at equidistant steps. Which strategy is used for initialization, is application dependent.

If the algorithm is initialized with an initial general overview of the measurement space, it can focus on capturing the dynamics of the different optima. The challenge here is not only to find the global optima across the different local optima, but to track the movement, growth or shrinking of those optima. The difficulty of the problem is illustrated in Figure 3. At time t, the most right peak is the global optimum, but at time $t+1$ the objective functions has changed, resulting in the most right peak to be only a local optimum.

If there were no initialization phase there is a risk of a cold start problem. The algorithm would have to incorporate additional space-filling techniques first, in order to learn the locations of the different initial



(a) Objective function at time t

(b) Objective function at time $t+1$

Figure 3: Behavior of a dynamic optimization problem. Over time the different optima can move, grow or shrink.

local optima before it is able to capture the time dynamics of the measurement space. Although such an initial exploration is possible, it is out of scope for this work.

2.4 Phase 1: Define New Waypoint

When the initialization is complete, the algorithm starts with the sequential loop. The new waypoint is defined to maximize the amount of new information. ALBATROS defined this maximum increase of information using the maximin criterion (Johnson et al. 1990). Since this criterion is not sufficient for optimization problems, a more complex BO approach is used here.

First, a surrogate model is generated. This is done by building a spatiotemporal Gaussian process to model the latent objective function. The main advantages of a GP are that very few assumptions are made about the data and the capability of a GP to handle and estimate uncertainty. For more information about Gaussian processes we refer to (Rasmussen and Williams 2006). The input of the spatiotemporal GP is a n-dimensional coordinate vector $\mathbf{x}$ extended with a freshness value. The output is the measured value $y = f(\boldsymbol{x},t)$. When predicting a value, the coordinate vector is extended with a freshness of one and used as input to the GP model.

Next, the surrogate model can be used to determine the new waypoint. This is done by defining an acquisition function. A good acquisition function is capable of exploiting previous measurements and exploring new regions. A frequently used acquisition function that makes a good exploration-exploitation trade-off is the expected improvement (EI) (Jones et al. 1998), given in (1) where f_{min} represents the current best minimum and Y represents the predicted value at point x. This is also the acquisition function used for defining the new waypoint.

$$E[I(x)] = E[max(f_{min} - Y, 0)] \tag{1}$$

2.5 Phase 2: Compute Path Extension

The second step of the sequential loop involves the generation of the path from the last sample (the current location) to the waypoint calculated in the previous step. First, a bounded Voronoi tessellation is constructed. A Voronoi tessellation (also called a Voronoi diagram) of a set of inducing points consists of a set of edges and a set of vertices. Here, the previous sample locations are used to generate the Voronoi tessellation. The edges represent equidistant locations between the two nearest measurement locations. A Voronoi cell of a

point is defined by the surrounding edges of that point. For more information about Voronoi tessellations and how to construct them, the reader is referred to (Aurenhammer 1991).

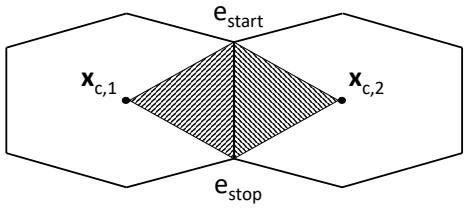


Figure 4: ALBATROS edge weight metric (Van Steenkiste et al. 2019).

After the construction of the Voronoi tessellation, the possible solutions of the path extension are limited to the edges of the tessellation. Since the edges are equidistant locations between two previous measurements, sampling and traveling along these edges is optimal with regarding to filling the space as equally as possible. ALBATROS transforms the Voronoi tessellation into a weighted graph with nonnegative weights. These weights are assigned by making a trade-off between the following, possibly conflicting, goals. The path needs to be as short as possible but it should also attempt to visit locations far away from previous samples. The weight of an edge was defined by ALBATROS as the surface of the triangles shown in Figure 4. After calculation of the weights, they are normalized by subtracting them from the maximum such that edges far from samples have lower weights. In this work, the proposed algorithm extends this approach by adding an extra freshness factor to the weights as described in formula (2).

$$w_{new}(e_{\text{start}}, e_{\text{stop}}) = w_{old}(e_{\text{start}}, e_{\text{stop}}) * \left(\frac{\text{freshness}(x_{c,1}) + \text{freshness}(x_{c,2})}{2}\right) \tag{2}$$

Finally, the path from the last sample to the new waypoint can be obtained by using any shortest-path algorithm for nonnegative weights. A frequently used solution for this problem is Dijkstra's algorithm (Dijkstra 1959). Using this path extension strategy ensures that the information gathering is optimally spread out while giving the preference to sampling closer to older samples.

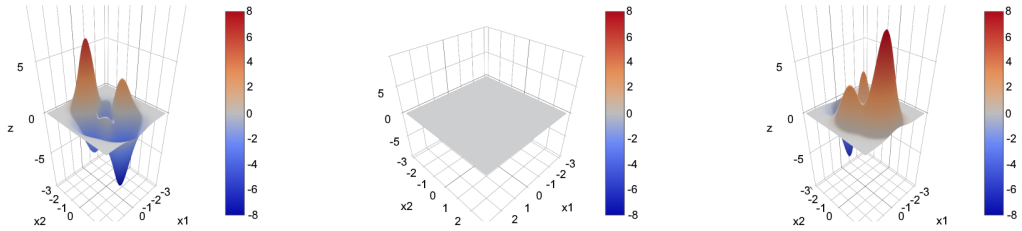
2.6 Phase 3: Determine New Points Along Path

Once a new waypoint is defined and a path towards that waypoint is computed, the algorithm determines where to sample along that path. The ALBATROS algorithm uses a heuristic to determine these sample locations. The points are chosen as far away as possible from the other points. Here, the proposed algorithm does not use this heuristic, instead samples are taken equidistantly. This fixed distance between subsequent samples is an additional parameter of the algorithm. This is the distance that the measurement device can traverse in one unit of time.

This final phase concludes the sequential loop. Next, another iteration of this sequential loop can be started if necessary. This is decided by any stopping criterion, such as time duration, traveled distance, or another information metric. If no additional iteration is started, the algorithm terminates.

3 EXPERIMENTAL SETUP

The algorithm was implemented in Python. To measure the performance of the algorithm, the offline error was chosen as a metric (Branke and Schmeck 2003). The formula is given in (3). This metric has been used extensively and has been used as a standard performance measure for dynamic problems by many researchers. For examples, the reader is referred to the survey paper (Moser and Chiong 2013). An important advantage of this metric is that, since it is an average, it does not punish exploration at any specific time instant (Nyikosa et al. 2018). This is an important property since exploration might not lead to good solutions in the near future, but it can be necessary for good solutions in the long run. Another



(a) $t = 0.0$ (b) $t = 0.5$ (c) $t = 1.0$

Figure 5: Illustration of the dynamic character of the objective function.

property of this metric is that it favors algorithms that find relatively good solutions early on (Moser and Chiong 2013).

$$\text{offline}_{\text{error}} = \frac{1}{T}\sum_{t=1}^{T}\left(F(\text{real}_{\text{opt}}(t)) - F(\text{model}_{\text{opt}}(t)\right) \tag{3}$$

Since ALBATROS originally did not interpret the performed measurements and focused on the space-filling aspect, changes had to be made to the algorithm to be able to compare them. Before computing the metric, a surrogate model is built using the measured samples. Note that this surrogate model was not interpreted by the ALBATROS algorithm in any way and did not influence any decisions of the ALBATROS algorithm. Its sole purpose was interpolating the measured data points to calculate the metric.

4 RESULTS AND DISCUSSION

Our novel algorithm was compared with the original ALBATROS algorithm using the offline error metric. First, the proposed algorithm is compared using peaks that are slowly being mirrored. The second dataset consists of multiple peaks that move, grow and shrink randomly.

4.1 Mirrored Peaks

First, the proposed algorithm is compared on a modified peaks function (Borchers, H. W. 2018). The original peaks function is static and is given in (4). It is transformed to a dynamic function by adding a time dimension as illustrated in formula (5). This function mimics the behavior of multiple peaks and valleys slowly being mirrored through time and is illustrated in Figure 5. Initially, when $t = 0$ there are multiple peaks and valleys. Slowly all peaks and valleys decrease and converge to a flat surface given by $z = 0$. This surface is reached halfway, at $t = 0.5$. At this tipping point, all peaks become a valley and vice versa. Next, the peaks and valleys grow until $t = 1$. An animation of the evolution of the peaks function is available at https://www.youtube.com/user/sumolab.

$$f(x_1,x_2) = 3\times(1-x_1)^2\times e^{-(x_1^2)-(x_2+1)^2} - 10\times(\frac{x_1}{5} - x_1^3 - x_2^5)\times e^{-x_1^2-x_2^2} - \frac{1}{3}\times e^{-(x_1+1)^2-x_2^2} \tag{4}$$

$$f_{\text{dynamic}}(\boldsymbol{x},t) = t\times f(\boldsymbol{x}) + (t-1)\times f(\boldsymbol{x}) \quad \text{with } t\in[0,1] \tag{5}$$

The experiment was performed in 2D and limited to the $[0,1]^2$ space. The proposed sampling distance was set to 0.2. The measurement probe can travel 0.2 distance units in one time unit. The value of the aging factor α was set to 0.005. As stated before, the time goes from zero to one and performing a

single measurement takes 0.005 time instances, allowing 200 samples to be measured for this experiment. Computation time is neglected in the experiments because of the previously made assumption that sampling time and movement time is orders of magnitude larger. At the end of every sequential loop, the offline error is calculated. Both algorithms stop the sequential loop when $t = 1$ is reached.

To prevent possible cold start issues, both algorithms were initialized with the samples shown in Figure 6. Other space-filling DoE methods would suffice as well since the aim here is to get an initial view of the objective function as described in section 2.3.

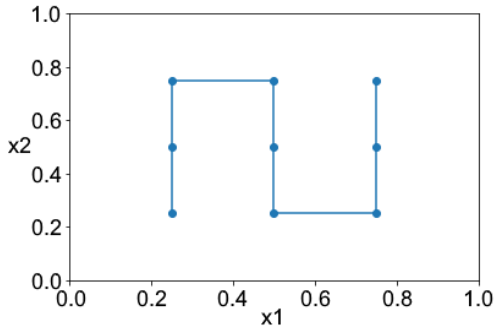


Figure 6: Initial samples for experiments.

The offline error comparison is shown in Figure 7. The red, dashed line is the ALBATROS algorithm and the blue line is the proposed novel algorithm for dynamic optimization problems. Initially the score is the same since no peak has been detected in the initial samples. When $t = 0.11$, the novel algorithm has identified a good approximation of the optima as illustrated by Figure 8. This figure shows the traversed trajectory and the resulting surrogate model. The blue lines represent the Voronoi tessellation, the black points represent the samples and the black dashed line is the traveled trajectory. The yellow point is the previous waypoint while the cyan points are the recent samples proposed in the last sequential cycle. It is clear that the traveled trajectory is not optimal in terms of space-filling properties. The trajectory makes a trade-off between exploration and exploitation, resulting in better modeling of the peaks.

In contrast, the ALBATROS algorithm has not yet detected any peaks because the more space-filling trajectory was traversing the boundaries of the grid as illustrated by Figure 9. Since the objective function contains few information at the borders, the surrogate model is a rather poor approximation. As the time increases, the ALBATROS algorithm offline error improves, but the proposed algorithm continues to outperform it as it is able to exploit previously gathered knowledge. As seen in Figure 10, the proposed algorithm has a much closer approximation of the peaks.

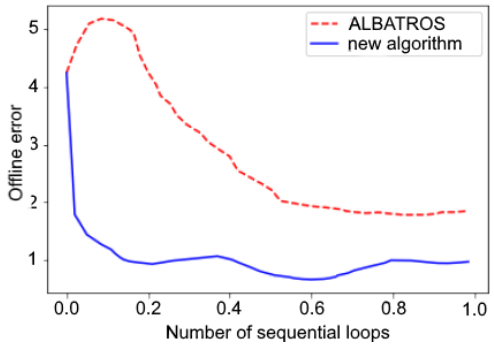
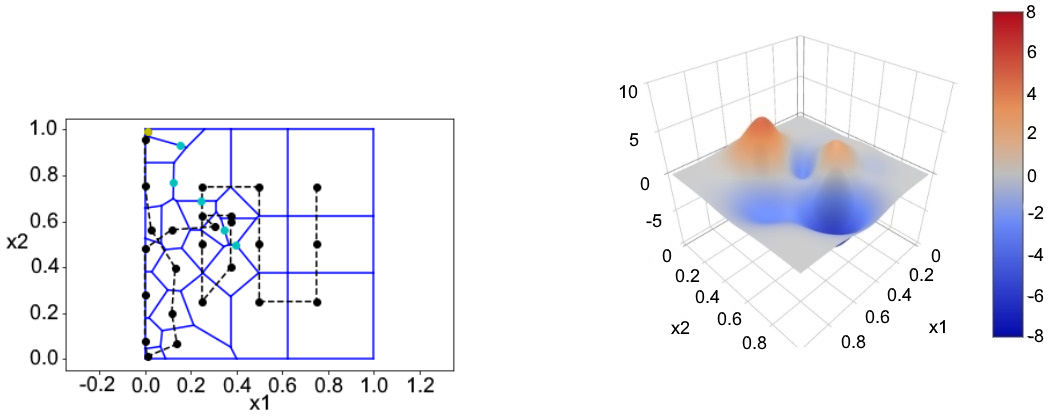
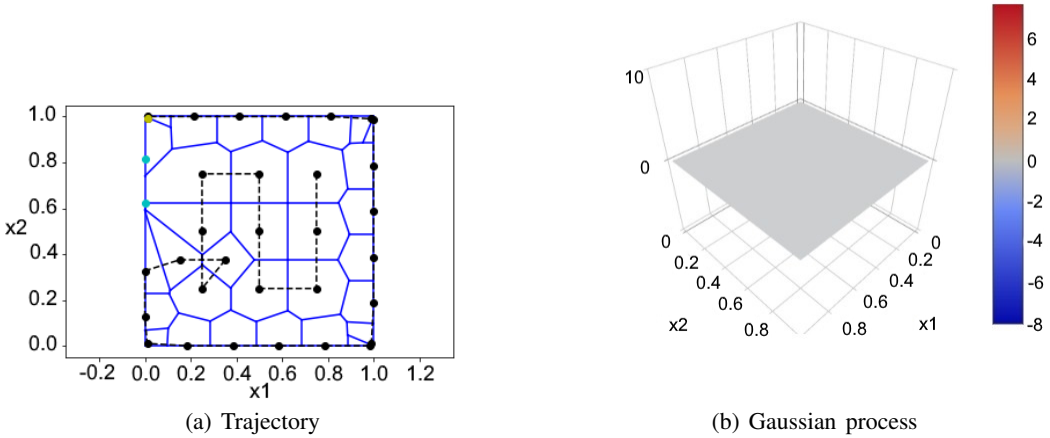


Figure 7: Comparison of the offline error between ALBATROS and the novel algorithm on the mirrored peaks data.



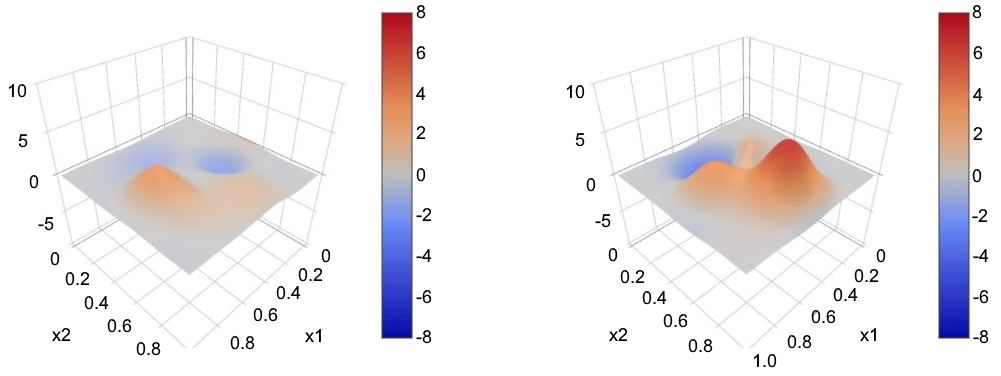
(a) Trajectory (b) Gaussian process

Figure 8: Trajectory and Gaussian process of the proposed algorithm after 5 sequential cycles, $t = 0.11$.



(a) Trajectory (b) Gaussian process

Figure 9: Trajectory and Gaussian process of the ALBATROS algorithm after 5 sequential cycles, $t = 0.125$.



(a) ALBATROS (b) Proposed algorithm

Figure 10: The surrogate model of the ALBATROS algorithm and the proposed algorithm with the mirrored peaks dataset, $t = 1$.

4.2 Moving Peaks Benchmark

Second, a well-studied benchmark is used. The moving peaks benchmark (Branke 1999) consists of multiple peaks where the height, width and position of each peaks changes through time. There are three standard scenarios defined with respectively 5, 10 and 50 peaks and different height, width and positions parameters. For our experiments, two peaks defined by the cone function (6) were used. With, $N = 2$ dimensions, $\boldsymbol{p}$ the peak coordinates, h the height and w the width of the peak. A two-dimensional grid was defined with coordinates in $[0,1]^2$ space. The height of the peaks varied between 10 and 50. Moves between subsequent timestamps are defined by drawing three random numbers from a uniform distribution. The first two are picked from the interval $[-0.05, 0.05]$ and the last one is picked from the $[-3,3]$ interval. These respectively represent the change in the spatial coordinates and the height. An animation of the evolution of the benchmark function is available at https://www.youtube.com/user/sumolab.

$$f(\boldsymbol{x}) = h - w\sqrt{\sum_{i=1}^{N}(x_i - p_i)^2} \tag{6}$$

The sampling distance was set to 0.2 and α was set to 0.005. Performing a single measurement takes 0.005 and as stated before, computation time is neglected. The offline error is calculated at the end of every sequential loop. In contrast to the previous experiments, the algorithms do not stop when the time reaches one. Instead, the algorithms continue to run until the difference in offline error stagnates. The initialization samples are identical to the previous experiment and shown in Figure 6.

In Figure 11 the offline error comparison is shown. The novel proposed algorithm outperforms the ALBATROS algorithm if executed long enough. It is important to run the experiments long enough to compensate the random character of the moving peaks benchmark. Starting from ca. $t = 0.30$, the proposed algorithm has a lower offline error. The offline error difference between ALBATROS and the proposed algorithm remains more or less constant during the remainder of the experiment. To illustrate the exploitation, Figure 12 is shown. On the left, the ALBATROS trajectory is given while on the right, the novel algorithm is shown. It is clear that the proposed algorithm exploits previous measurements instead of solely exploring the target space as the ALBATROS algorithm does.

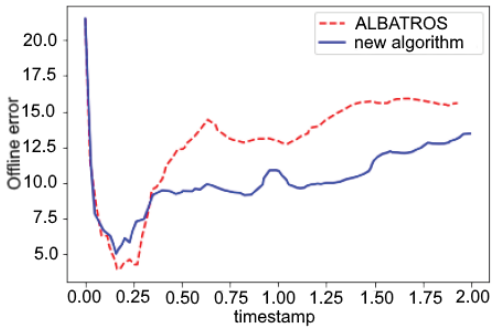
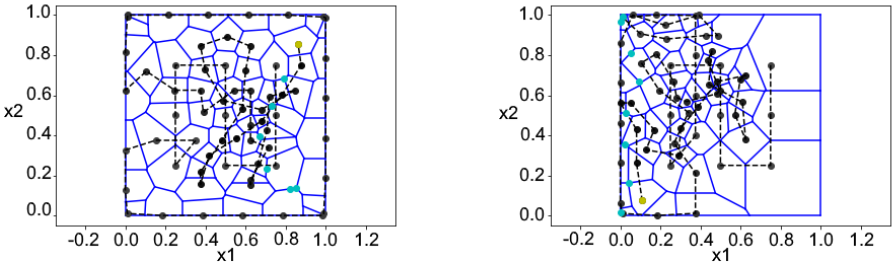


Figure 11: Comparison of the offline error between ALBATROS and the novel algorithm on the moving peaks benchmark.



(a) ALBATROS (b) Proposed algorithm

Figure 12: The trajectory of the ALBATROS algorithm and the proposed algorithm on the moving peaks benchmark when $t = 0.3$.

5 CONCLUSION

A novel algorithm for solving dynamic optimization problems was presented. The algorithm is a sequential line-based sampling algorithm designed to find and track optima over time. This algorithm performs best when measuring samples is expensive and includes a cost of the traversing the trajectory. The effectiveness of the proposed algorithm was shown using a 2D dynamic dataset and the moving peaks benchmark.

Future work should be done on expanding the algorithm to multiple coordinate dimensions since it is now limited to two dimensions. Additional work should be performed to determine the value of α automatically. Now, the value of α is a parameter of the algorithm but good estimations of this parameter require prior knowledge of the data.

REFERENCES

Aurenhammer, F. 1991. "Voronoi Diagrams - a Survey of a Fundamental Geometric Data Structure". *ACM Computing Surveys (CSUR)* 23(3):345–405.

Berni, J. A., P. J. Zarco-Tejada, L. Suárez, and E. Fereres. 2009. "Thermal and Narrowband Multispectral Remote Sensing for Vegetation Monitoring From an Unmanned Aerial Vehicle". *IEEE Transactions on Geoscience and Remote Sensing* 47(3):722–738.

Borchers, H. W. 2018. "pracme: Practical Numerical Math Functions". rdrr.io/rforge/pracma/. accessed 24th March 2019.

Branke, J. 1999. "Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems". In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Volume 3, 1875–1882.

Branke, J., and H. Schmeck. 2003. *Designing Evolutionary Algorithms for Dynamic Optimization Problems*, 239–262. Berlin, Heidelberg: Springer.

Brown, R. G., R. F. Meyer, and D. A. D'Esopo. 1961. "The Fundamental Theorem of Exponential Smoothing". *Operations Research* 9(5):673–687.

Carlson, J., and R. R. Murphy. 2005. "How UGVs Physically Fail in the Field". *IEEE Transactions on Robotics* 21(3):423–437.

Chang, J. H., and W. S. Lee. 2005. "estWin: Online data stream mining of recent frequent itemsets by sliding window method". *Journal of Information Science* 31(2):76–90.

Choset, H. 2000. "Coverage of Known Spaces: The Boustrophedon Cellular Decomposition". *Autonomous Robots* 9(3):247–253.

Cruz, C., J. R. Gonzalez, and D. A. Pelta. 2011. "Optimization in Dynamic Environments: A Survey on Problems, Methods and Measures". *Soft Computing* 15(7):1427–1448.

Deschrijver, D., F. Vanhee, D. Pissoort, and T. Dhaene. 2012. "Automated Near-Field Scanning Algorithm for the EMC Analysis of Electronic Devices". *IEEE Transactions on Electromagnetic Compatibility* 54(3):502–510.

Dijkstra, E. W. 1959. "A Note on Two Problems in Connexion with Graphs". *Numerische Mathematik* 1(1):269–271.

Floudas, C. A., P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gümüş, S. T. Harding, J. L. Klepeis, C. A. Meyer, and C. A. Schweiger. 1999. *Dynamic Optimization Problems*, 351–412. Boston: Springer.

Gademer, A., F. Mainfroy, L. Beaudoin, L. Avanthey, V. Germain, C. Chéron, S. Monat, and J.-P. Rudant. 2009. "Solutions for near real time cartography from a mini-quadrators UAV". In *Remote Sensing for Environmental Monitoring, GIS Applications, and Geology IX*, Volume 7478, 1–12. International Society for Optics and Photonics.

Gardner, E. S. 2006. "Exponential Smoothing: The State of the Arte Part II". *International Journal of Forecasting* 22(4):637 – 666.

Gorodetsky, V., O. Karsaev, V. Samoilov, and A. Ulanov. 2005. "Asynchronous Alert Correlation in Multi-agent Intrusion Detection Systems". In *Computer Network Security*, edited by V. Gorodetsky, I. Kotenko, and V. Skormin, 366–379. Berlin, Heidelberg: Springer.

Hilbert, D. 1891. "Ueber die Stetige Abbildung Einer Line auf ein Flächenstück". *Mathematische Annalen* 38(3):459–460.

Johnson, M. E., L. M. Moore, and D. Ylvisaker. 1990. "Minimax and Maximin Distance Designs". *Journal of Statistical Planning and Inference* 26(2):131–148.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions". *Journal of Global Optimization* 13(4):455–492.

Kleijnen, J. P. 2008. *Design and Analysis of Simulation Experiments*, Volume 20. Cham: Springer.

Martínez-de Dios, J., L. Merino, F. Caballero, A. Ollero, and D. Viegas. 2006. "Experimental Results of Automatic Fire Eetection and Monitoring With UAVs". *Forest Ecology and Management* 234(1):1–10.

Mockus, J. 1974. "On Bayesian Methods for Seeking the Extremum". In *Proceedings of the IFIP Technical Conference*, 400–404. London, UK: Springer-Verlag.

Moser, I., and R. Chiong. 2013. *Dynamic Function Optimization: The Moving Peaks Benchmark*, 35–59. Berlin, Heidelberg: Springer.

Nex, F., and F. Remondino. 2014. "UAV for 3D Mapping Applications: a Review". *Applied Geomatics* 6(1):1–15.

Nyikosa, F. M., M. A. Osborne, and S. J. Roberts. 2018. "Bayesian Optimization for Dynamic Problems". *arXiv preprint arXiv:1803.03432*.

Rasmussen, C. E., and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. Cambridge: MIT Press.

Valente, J., D. Sanz, J. Del Cerro, A. Barrientos, and M. Á. de Frutos. 2013. "Near-Optimal Coverage Trajectories for Image Mosaicing Using a Mini Quad-Rotor over Iregular-Shaped fields". *Precision Agriculture* 14(1):115–132.

Van Steenkiste, T., J. van der Herten, D. Deschrijver, and T. Dhaene. 2019. "ALBATROS: Adaptive Line-Based Sampling Trajectories for Sequential Measurements". *Engineering with Computers* 35(2):537–550.

AUTHOR BIOGRAPHIES

JENS JOCQUE is a master thesis student at Ghent university - imec, IDLab. His email address is jens@jocque.be.

TOM VAN STEENKISTE is a PhD student at Ghent university - imec, IDLab. His email address is TomD.VanSteenkiste@ugent.be.

PIETER STROOBANT is a PhD student at Ghent university - imec, IDLab. His email address is Pieter.Stroobant@UGent.be.

REMI DELANGHE is a master thesis student at Ghent university - imec, IDLab. His email address is Remi.Delanghe@UGent.be.

DIRK DESCHRIJVER is a professor at Ghent university - imec, IDLab. His email address is Dirk.Deschrijver@UGent.be.

TOM DHAENE is a professor at Ghent university - imec, IDLab. His email address is Tom.Dhaene@UGent.be.