36th European Modeling & Simulation Symposium, *018*
21st International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2024 The Authors.
doi: 10.46354/i3m.2024.emss.018

Efficient Global Optimization for Dynamic Problems

Bernhard Werth[1,2*], Johannes Karder[1,2], Stefan Wagner[1] and Michael Affenzeller[1]

[1]Heuristic and Evolutionary Algorithms Laboratory
University of Applied Sciences Upper Austria, Softwarepark 11, 4232 Hagenberg, Austria
[2]Institute for Symbolic Artificial Intelligence
Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria

[*]Corresponding author. Email address: bernhard.werth@fh-hagenberg.at

Abstract

Efficient Global Optimization (EGO) is a very important black-box optimization framework for solving expensive optimization problems, which appear in high-fidelity simulation-based optimization. EGO operates by employing a Gaussian process model as an approximation to the computationally expensive black-box function. As we observe an increasing trend towards dynamic optimization problems that incorporate live data, combined with the fact that EGO is designed to solve static optimization problems, a need for adaptation arises. This paper analyzes and compares five EGO extensions that should aid the algorithm in dealing with dynamic changes. Results indicate that such dynamification is challenging and while successful for problem instances with the correct difficulty and change severity, it can have adverse effects if older data is unrepresentative of new scenarios.

Keywords: Surrogate; optimization; dynamic; EGO; heuristic

1. Introduction

Dynamic/ongoing optimization is of increasing relevance for practitioners that deal with Internet of Things (IoT), Industry 4.0 and/or digital twin applications (Silva et al., 2020). Specifically in areas where automated systems interact, like warehouse operations, logistics or cobot-assisted manufactoring (Zaatari et al., 2022; Granata et al., 2024) unforseen changes to the environment happen frequently and may impact the current plan. These scenarios are often inherently complex, time-critical and represent significant financial investments. Therefore, optimizing these systems directly as closed-form functions is often not feasible. Testing different configurations in reality might also disrupt operations or might simply be too slow. The approach of simulation-based optimization remedies this issue by employing metaheuristic black-box solvers in combination with arbitrarily complex simulation models (De Paula Ferreira et al., 2020).

These simulation-based optimization problems and objective functions are often both computationally expensive to evaluate and subject to significant changes over time as e.g. production scenarios progress, stochastic processes materialize or unforeseen changes occur.

In recent years, various real-world applications of dynamic simulation-based optimization have been published. Application cases range from traditional dynamic production scheduling (Jiang et al., 2022b), crane scheduling (Li et al., 2020), to much more ad-hoc applications such as water distribution in emergency situations (Zhang et al., 2020) or niche use cases like the fine calibration of particle accelerators (Kuklev et al., 2023). The shear broadness of different application scenarios highlights the need for transferable black-box optimization methods that are not inherently tied to a specific optimization problem.

Especially in the case of simulation-based optimization, exact optimization can be prohibitively expensive and surrogate models that approximate the simulation behavior need to be used to guide metaheuristic search algorithms. The *Efficient Global Optimization* (EGO) framework is one of the most used surrogate-assisted optimization algorithm schemes in such scenarios. Like most metaheuristic solvers, however, it is mainly geared towards optimization of static problems that do not change over time.

In this paper we investigate different approaches for adapting EGO to dynamic optimization. The main challenge is how data points obtained from previous problem states (epochs) can be incorporated into the surrogate model that tries to approximate the current state.

The following content is structured as follows. Section 2 gives an overview of dynamic optimization, focusing on surrogate-assisted methods and describes the EGO workflow. Section 3 explains different methods of utilizing past information within EGO. The experimental setup and obtained results are shown and discussed in Section 4. The paper concludes with Section 5.

2. Related Literature

Scientific literature of dynamic optimization problems (DOPs) is certainly less prevalent for their static counterparts, but in recent years the rise of digitalization and the advent of streaming data a number of methods for a wider array of DOPs and application cases have been created. The two-part survey by Yazdani et al. (Yazdani et al., 2021b,c) captures many different optimization strategies, benchmark problems and generators as well as analysis methods. The use of predictive models for dynamic optimization in the summarized methods is usually in the context of change prediction and not geared towards computationally expensive DOPs. The survey by Jiang et al. (2022a) focuses specifically on multi-objective optimization.

Concerning specifically the use of surrogate-assisted dynamic optimization for expensive problems, a smaller number of ideas was put forward. Gao and Bai (2022) utilize a spatio-temporal kernel with the EGO framework to optimize various test functions. Kuklev et al. (2023) focus on a specific problem with increased drift and counteract this via *kernel-switching*. Fan et al. (2020) utilize a transfer learning approach to reattain information from previous epochs and can avoid starting their data collection "from scratch" every time the problem changes.

The work by Fan et al. (2020) presents a surrogate-assisted differential evolution approach for a multi-stage optimization problem where new decision variables are introduced over time

In general, the Efficient Global Optimization algorithm and various of its adaptations often summarily called Bayesian Optimization, comprises the following steps (depicted in Figure 1):

1. **Initial sampling of data points**: This step is usually performed by using a space-filling design like a Latin Hypercube pattern as to cover as much of the search space as possible without waiting too many expensive evaluations.

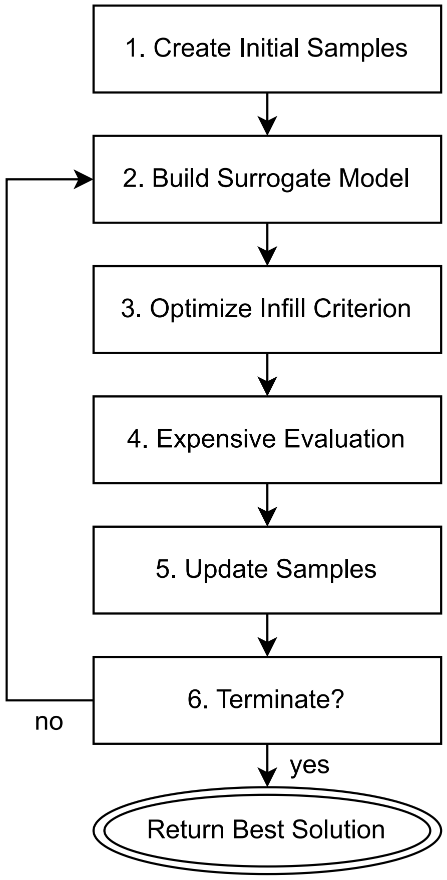

Figure 1. EGO steps

2. **Building the surrogate model**: While variations in the choice of model exist, the most common choice of model is a Gaussian process (GP) regression model that directly predicts the objective value that would be obtained from the expensive optimization problem. This step is sometimes itself quite arduous as Gaussian process models can have a number of hyperparameters that need to be tuned every time a model is built and the building process of the model involves the inversion of an $n * n$-matrix which can dampen performance if many training data points have to be included in the model. The advantages of this type of model are that Gaussian processes are fast to evaluate, provide a measure of their own uncertainty and gradients for both hyperparameters and predictions.
3. **Choosing the next sample**: The next solution candidate that is to be evaluated is chosen by optimizing an acquisition function (infill criterion) such as expected improvement (EI), probability of improvement (PI) or upper confidence bound (UCB) (Wang et al., 2017). Choosing new samples according to their EI enables exploration and exploitation of the search space. The expected improvement of a point correlates with the quality and availability of nearby points. EI is defined as

$$E[I(\mathbf{x})] = \underbrace{(f_{\min} - \hat{y})\Phi\left(\frac{f_{\min} - \hat{y}}{s}\right)}_{\text{exploitation}} + \underbrace{s\phi\left(\frac{f_{\min} - \hat{y}}{s}\right)}_{\text{exploration}} \quad (1)$$

, where $f_{\min}$ is the objective function value of the best sample observed so far, $\hat{y}$ and s are the objective value of $\boldsymbol{x}$ as predicted by the surrogate model and the uncertainty of this prediction, respectively, and Φ and ϕ the standard normal density and distribution functions.

4. **Evaluate the chosen sample** using the original objective function, which is usually an expensive task.
5. **Update the sample set** by adding the expensively evaluated sample.
6. **Terminate according to termination criterion**: Usually, EGO stops after a specified number of iterations. If a termination criterion is met, the best solution that was sampled is returned. If no termination criterion is met, continue with step 2 and rebuild the surrogate model using the updated sample set.

3. Methodology: Extending EGO for Dynamic Optimization

In its simplest form, EGO is implemented to execute the steps mentioned in Section 2. One of the fundamentals of EGO is to maintain a data set of evaluated inputs and respectively observed outputs, and EGO assumes that this data does not change. When dealing with dynamic problems, such assumptions cannot be made. Dynamic problem updates can have different effects on previously observed data: objective values can change and observed inputs can even be rendered invalid, as they become incompatible with the current problem state.

Figure 2 shows an intermediate state of a restarting EGO implementation, where old data points (shown in red) are discarded at epoch change. The black line shows the ground truth, i.e. the current configuration of the moving peaks (maximization) problem. The blue and green lines show the predictions and uncertainty of the learned GP model, respectively. The model considers only points of the current epoch for training (shown in orange). In the lower subfigure, the acquisition function is visualized. Ideally, EGO samples the next data point where the acquisition value is maximal.

Because the peaks have shifted, old data points do not perfectly match the ground truth anymore. An argument can be made that the old data points still contain some useful information for the model. The current GP model underestimates the variance of the ground truth and is therefore too conservative in its uncertainties. Furthermore, while the old data points do not match the new peaks, they can still be considered to be “near” the new peaks, which could guide EGO into promising regions. We therefore investigate different ways in which EGO can be applied to problems that exhibit such dynamicity.

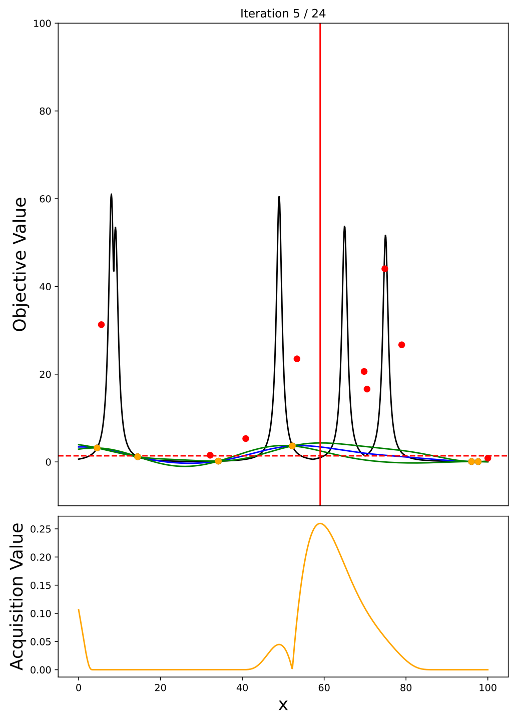

Figure 2. Reinitialized GP after epoch change.

3.1. Limited Memory (LM)

The simplest way to include outdated data points in a model is to ignore the fact that the results of reevaluating them in the current epoch might differ slightly from the information obtained in a previous epoch.

If the dynamic changes to the problem are small and gradual in nature, the impact of this error can be negligible compared to the information loss that not including the older data points can incur. For open-ended optimization the number of old data points to include must be limited, since in many cases the out-of-date-error can be assumed to increase over time and usually the training speed of Gaussian processes scales poorly with the number of data points. This limit can be implemented on a per-epoch or (if changes are numerous and small or not easily detectable) on a per-evaluation basis. Since for many simulation-optimization scenarios changes can be easily detected by monitoring the event queues that inform the simulation we opt to use a per-epoch cutoff in this work. One such example is Llorente and Djurić (2024) that use such a method to continuously solve a localization problem in a two-dimensional wireless sensor network.

3.2. Increased Uncertainty (IU)

When allowing EGO to reuse data points from previous iterations and previous epochs, the fact that the potentially outdated information is not as certain as information obtained in the current epoch should be accounted for. The most direct method is to deliberately introduce uncertainty/noise in the kernel of the Gaussian process. However, this has to be done with great care as manual introduction of noise can interfere with hyperparameter tuning and leaves open the question of how much noise should be added to each data point. In this paper, we opt to force the Gaussian process model to introduce noise on its own by duplicating old data points. The original data points keep their old objective values, why the duplicates are set to their mean objective value of all points of the current epoch, which is updated every iteration.

3.3. Transfer Learning (TL)

A new idea showcased by Liu et al. (2023) is to treat the inclusion of old data points as a *transfer learning* problem and utilize existing transfer learning techniques like Transfer Component Analysis (TCA) (Pan et al., 2010) to adapt older points to the current epoch. This approach essentially creates a transfer model that predicts adapted $\mathbf{x}$ and y values, which are then treated as training points for the actual surrogate model. A potential danger of this setup is the propagation of errors between the two models, where poor transfer performance can mislead the surrogate model, which in turn influences the solution candidates that will be evaluated in the current epoch and the data points to be included in the next update of the transfer model. The ability of the transfer model to adequately capture the change between epochs from a very limited set of samples heavily influences the performance of the overall algorithm. In the presented experimental study, a Support Vector Machine (SVM) (Platt et al., 1999) built on $5 * d$ data points is used as a transfer model.

3.4. Spatio-Temporal Modelling (STM)

Another relatively intuitive approach is to extend the feature vector of each data point with the epoch number at which the point was generated. The search for the optimal value of the acquisition function is subsequently restricted to the hyper plane of the current epoch. Special care must be taken concerning the scaling of this new time variable, since many classically used kernel functions for Gaussian processes assume isotropic behavior (that all decision variables are scaled to roughly the same scale). A solution to this is the combination of a spatial kernel that deals with distances in the actual search space and a temporal kernel that implements the "forgetfulness" of the Gaussian process (Nyikosa et al., 2018).

Figure 3 provides visual examples for the described strategies. Each plot shows the ground truth, i.e. a specific section of the well-known Rastrigin test function, before and after an epoch change. This change results in a shift in both axes between the gray dotted old epoch and the black solid new epoch. Samples obtained in the old epoch are marked in gray and current ones are marked in black, respectively. Data points used for training are plotted as filled circles (), unused data points are plotted as crosses (). The blue area designates the Gaussian process model and

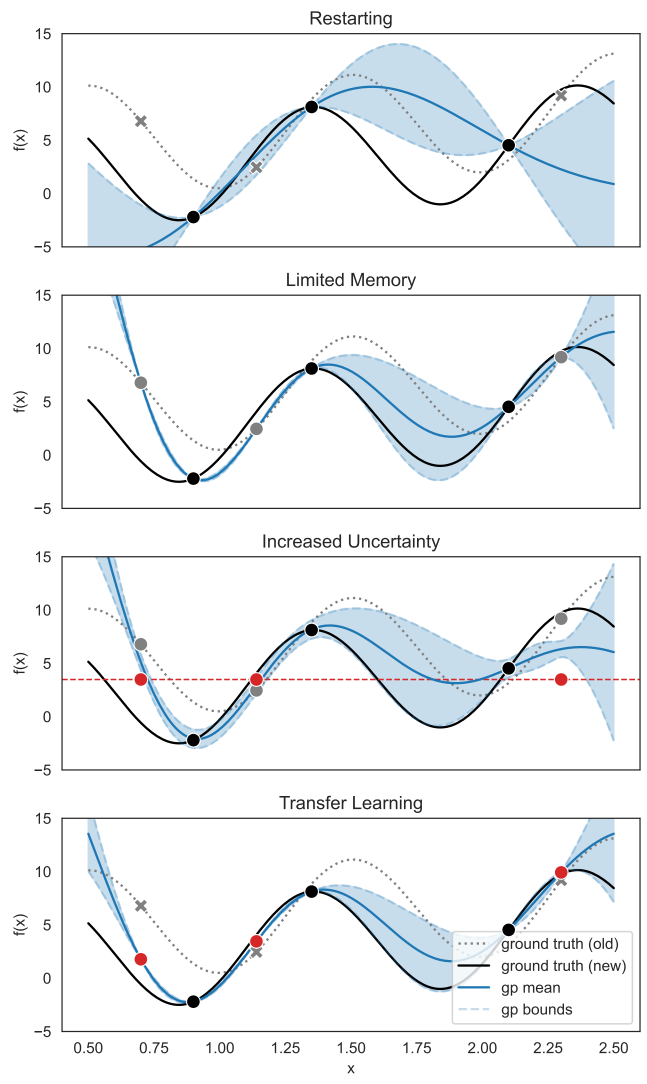

Figure 3. Restarting, LM, IU and TL strategies.

its uncertainties. Figure 4 shows the same example for the spatio-temporal model, where the inclusion of time as a dimension implies that the model now needs to approximate a surface rather than a line. The upper subplot displays the ground truth while the lower subplot shows the model's approximation.

The restarting plot shows a GP that does not use old data points for training and therefore loses some information. The second subplot shows a GP that uses the LM strategy and interprets old data as fully valid, which leads to a reasonable result on the rightmost region of the search space, but introduces significant error for the leftmost region. An application of the IU strategy can be seen in subplot three. The additional red data points force the GP to assume a certain level of noise. Note that all red data points share the same $f(x)$ value, which is equal to the mean $f(x)$ value of all current data points. The last subplot shows the GP that is built with points generated by transferring outdated samples, shown in red.

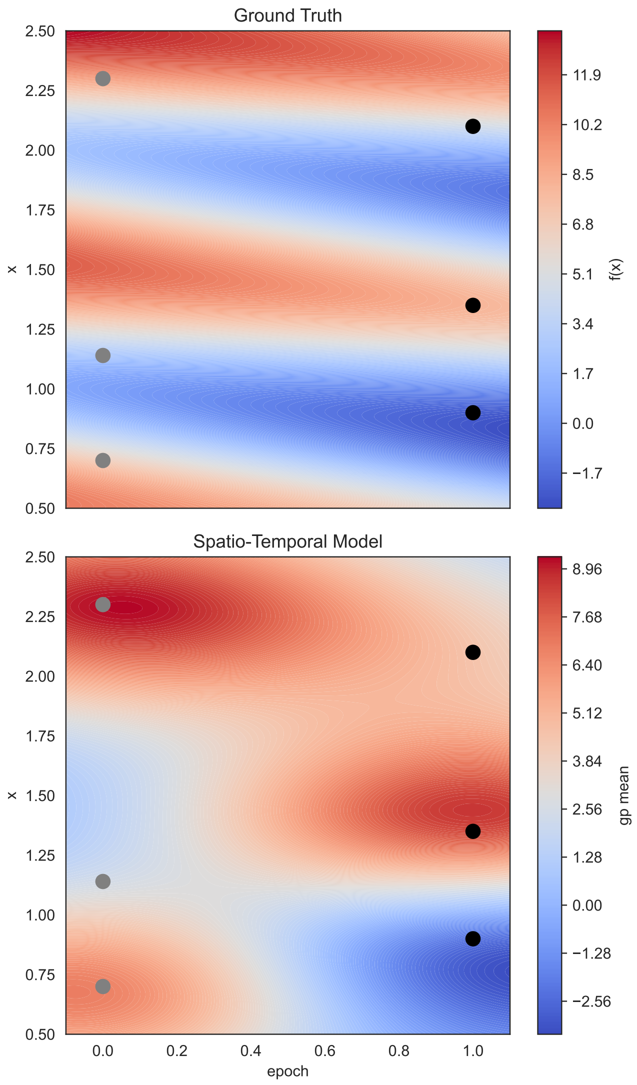

Figure 4. STM example.

4. Experiments & Results

The following results will use the well-known Moving Peaks benchmark generator by Branke (1999). While more modern generalized versions of this generator exist (Yazdani et al., 2021a) they are geared towards generating highly complex, irregular, asymmetric problems with higher degrees of decision variable interaction. While these types of problems can certainly appear in practice, they are usually poorly suited to surrogate-assisted optimization and the presented experiments should capture and compare the effects of different memory strategies rather than particularities of specific difficult landscapes.

The Moving Peaks problem defines a number of peaks with h_i, w_i and p_i, j being the height, width parameter and position in the j-th dimension of the i-th peak. The total objective value of this maximization problem is then defined as the maximum contribution of any peak for all solution candidates as seen in Equation 2. All parameters

Table 1. Problem and algorithm parameter setting

Parameter	Value/Range
Dimension d	1 .. 5
Decision variable range	0 .. 100
Epochs max_e	10
Epoch length	$10 * d$
Number of peaks	5
Peak heights	30 .. 70
Peak widths	0.1 .. 2.5
Shift strength	1
Initial sample size s	$5 * d$
Kernel	Anisotropic RBF + White Noise
Acquisition function optimizer	L-BFGS-B (10 restarts)
Hyper-parameter optimizer	L-BFGS-B

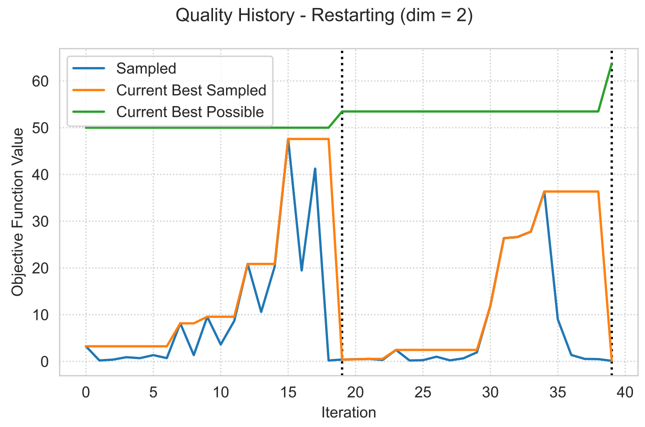

Figure 5. Timeline of two epochs.

of a peak undergo slight variations whenever an epoch changes.

$$q(x,t) = \max_i \frac{h_i(t)}{1 + w_i(t) * \sum_j^d (x_j - p_{j,i}(t))^2} \tag{2}$$

Since the moving peaks benchmark displays no global trend in large parts of the search space, dynamically shifted variations of the Rastrigin (Rastrigin, 1974) function are tested.

Table 1 lists the problem and algorithm parameters that are considered fixed over all comparisons.

The evaluated algorithm, including all of its memory strategies, have been implemented in Python. Gaussian processes are provided by “scikit-learn” (Pedregosa et al., 2011). To have a baseline for comparison, the Python package “bayesian-optimization” (Nogueira, 2014) is used, and identified by the term “Control” in all experiments.

Figure 5 displays a typical timeline of two epochs of a restarting EGO on a two-dimensional problem. The achieved quality increases as time progresses but resets drastically after the epoch change. The drop of the blue curve at the end of the second epoch is indicative of exploration behavior. The green line shows the best objective function value that can be achieved.

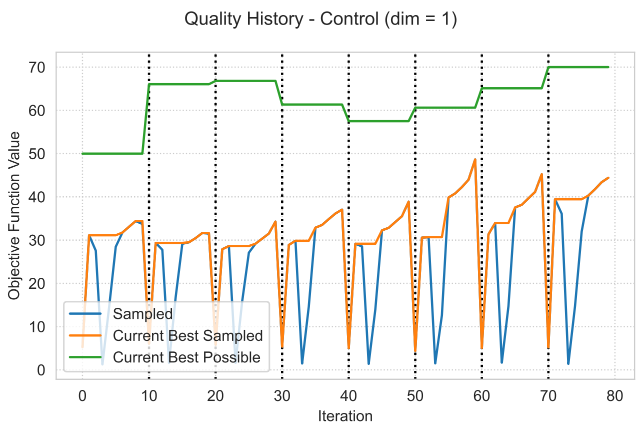

Figure 6. Timeline of a single Control run.

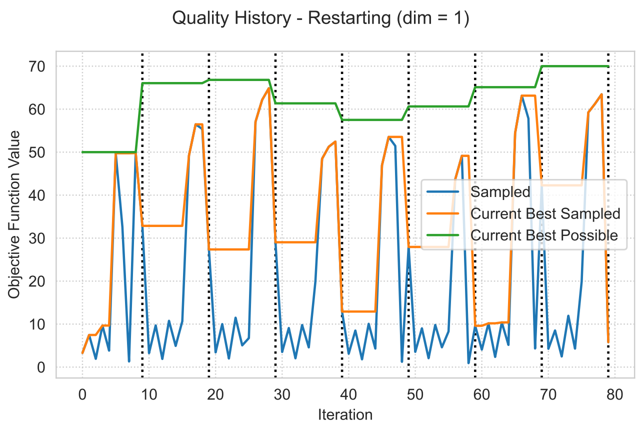

Figure 7. Timeline of a single Restarting run.

Figure 10 reports the mean *Mean Pre-Shift Error* (MPSE) for all runs started in the outlined experiments. For each single run, all epochs are analyzed. The pre-shift error within an epoch corresponds to the minimal distance between the best found and best possible objective value during that period. As every algorithmic setup is executed with three repetitions and for dimensions $1 - 5$, all computed pre-shift errors are then averaged, resulting in MPSEs, grouped by dimension and plotted for every algorithm extension.

- The first and foremost trend is the *curse of dimensionaltiy* which states that the performance of model-based optimization techniques decreases with the number of decision variables of the underlying problem. The overall performance of all algortihms for five-dimensional problems displays no notable difference anymore.
- In all Rastrigin-based experiments, the performance seems to be independent of the actual shift strength. Furthermore, no strategy distinguishes itself as a clear winner. Since the inclusion of old data seems to be of neither benefit not detriment, a possible explanation is that all EGO variations can exploit the strong global structure of the problem.
- In all Moving Peaks-based experiments, IU and TL are performing worst in most cases. This can potentially be attributed to the nature of the peak shifts. The resulting transfer function obtained from subtracting two shifted peaks is non-linear and can be even more complex than the original objective function, possibly requiring more points to adequately model the transfer than are needed to solve the original optimization problem.
- The Restarting and Control strategies are effectively static optimization approaches, and therefore unaffected by shift. As suggested by Alza et al. (2023), there are cases in which simply restarting an optimizer might be preferable to many conventional dynamic approaches.
- Looking at the results obtained in low-shifting Moving Peaks problems, LM and Tai are competitive with the static strategies and in some cases even outperform them.

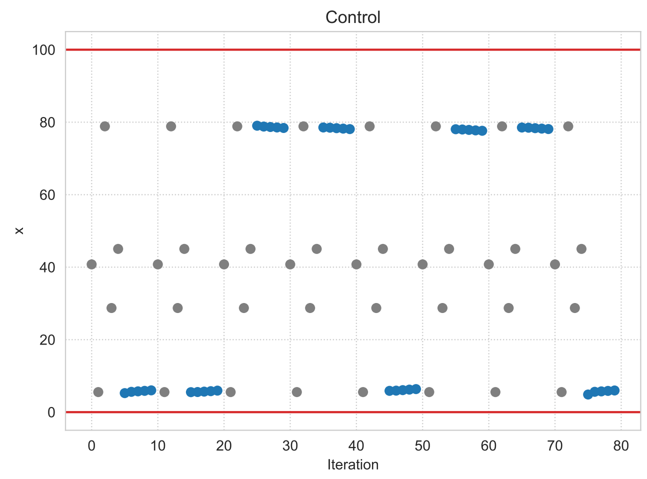

Figure 8. Sample distribution of a Control run.

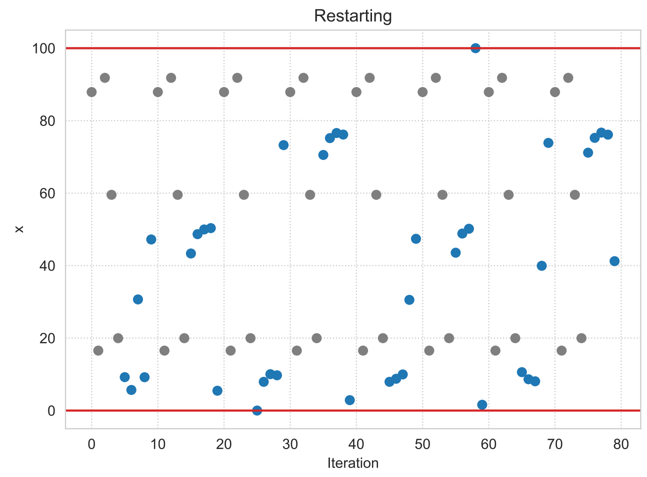

Figure 9. Sample distribution of a Restarting run.

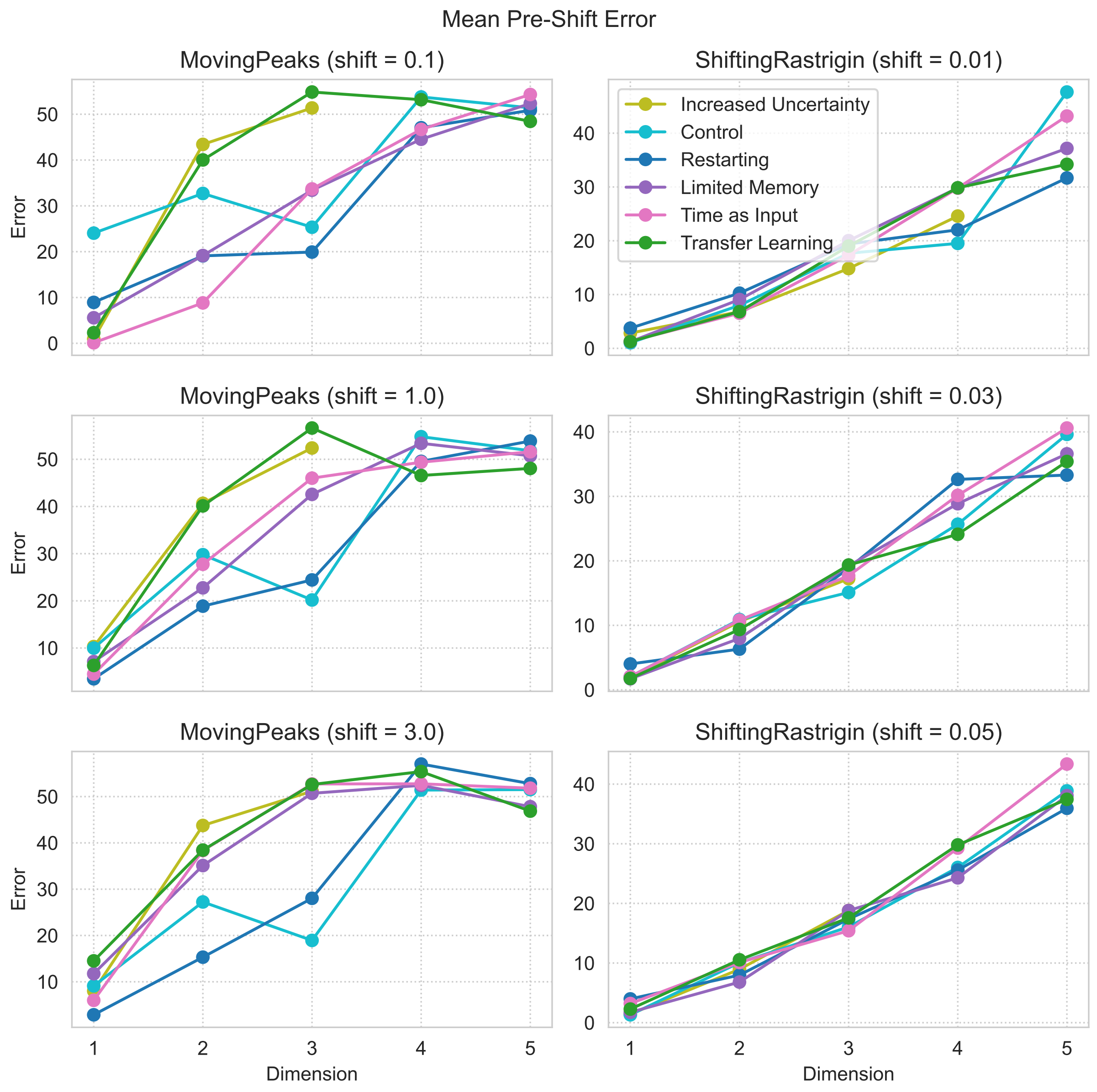

Figure 10. Mean pre-shift error.

- Most strategies perform similarly well on the one-dimensional Moving Peaks problems, indicating it being a rather easy to solve problem.

A note has to be made on the performance differences between the control an the restarting algorithm. While both versions follow the same idea of forgetting all old data points, reevaluating the initial set and operating from there with only new information, the setting of hyper-parameters and exploration of the acquisition function space differ. Figures6 and 7 provide examples of the quality curves obtained by singular runs of control and restarting respectively on the one-dimensional Moving Peaks problem. The control strategy was "luckier" in its initial set (the first 5 samples in each epoch), but very little progress is made after initialization. The restarting strategy seemingly jumps to quite superior qualities after initialization is done. An explanation of this behavior can be obtained by looking at the corresponding sample distributions of both algorithms (where in the search-space the algorithm decides to sample) in Figures 8 and 9 for these runs. The control strategy most likely suffers from either improper hyper-parameters for their GP models or fails to search the regionally very flat acquisition function, causing the algorithm to be "stuck" very close to the currently best initial sample point. Since both strategies lose all information as soon as the epoch changes, the apparent upwards trend in quality for the control strategy is not indicative of the algorithm recovering from this state, rather than simply some peaks moving closer to the initial sample points.

5. Conclusions & Outlook

In this work we presented, compared and evaluated multiple approaches how EGO can be extended to dynamic optimization problems. The five different EGO extensions include *limited memory*, *creation of noise*, *restarts*, *spatio-temporal modeling* and *transfer learning*. Although these methods differ significantly on how they incorporate past information and complexity, their overall impact on the achieved qualities is mostly limited to lower dimensional use cases which is consistent with previous reports on the effect of dimensionality on surrogate-based optimization techniques. It also highlights the need for more tailored approaches with the capacity to create longer memories in the algorithm in the hopes of obtaining the increased amount of information need to cover higher-dimensional search spaces. Lastly, this study only covers a limited set of test functions, with very uniform changes. Real-world situations where changes can appear in wide varieties and often display trends or regimes will necessitate case-specific selection and tuning of memory schemes and the EGO variation working on the resulting models. Also, evaluating EGO using other models instead of Gaussian processes could be interesting, for example, long short-term memory recurrent neural networks (Hochreiter and Schmidhuber, 1997) might be especially suited for dynamic surrogate modeling.

6. Funding

This research was funded in part by the Austrian Science Fund (FWF) [I 5315-N]. The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

References

Alza, J., Bartlett, M., Ceberio, J., and McCall, J. (2023). On the elusivity of dynamic optimisation problems. *Swarm and Evolutionary Computation*, 78:101289.

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, pages 1875–1882. IEEE.

De Paula Ferreira, W., Armellini, F., and De Santa-Eulalia, L. A. (2020). Simulation in industry 4.0: A state-of-the-art review. *Computers & Industrial Engineering*, 149:106868.

Fan, X., Li, K., and Tan, K. C. (2020). Surrogate assisted evolutionary algorithm based on transfer learning for dynamic expensive multi-objective optimisation problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Gao, T. and Bai, X. (2022). Bayesian optimization-based three-dimensional, time-varying environment monitoring using an uav. *Journal of Intelligent & Robotic Systems*, 105(4):91.

Granata, I., Faccio, M., and Boschetti, G. (2024). Industry 5.0: prioritizing human comfort and productivity through collaborative robots and dynamic task allocation. *Procedia Computer Science*, 232:2137–2146.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jiang, S., Zou, J., Yang, S., and Yao, X. (2022a). Evolutionary dynamic multi-objective optimisation: A survey. *ACM Computing Surveys*, 55(4):1–47.

Jiang, Z., Yuan, S., Ma, J., and Wang, Q. (2022b). The evolution of production scheduling from industry 3.0 through industry 4.0. *International Journal of Production Research*, 60(11):3534–3554.

Kuklev, N., Sun, Y., Shang, H., Borland, M., and Fystro, G. (2023). Robust adaptive bayesian optimization. *Proc. IPAC*, 23:4377–4380.

Li, J., Xu, A., and Zang, X. (2020). Simulation-based solution for a dynamic multi-crane-scheduling problem in a steelmaking shop. *International Journal of Production Research*, 58(22):6970–6984.

Liu, Y., Liu, J., Ding, J., Yang, S., and Jin, Y. (2023). A surrogate-assisted differential evolution with knowledge transfer for expensive incremental optimization problems. *IEEE Transactions on Evolutionary Computation*.

Llorente, F. and Djurić, P. M. (2024). Dynamic random feature gaussian processes for bayesian optimization of time-varying functions. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9756–9760. IEEE.

Nogueira, F. (2014). Bayesian Optimization: Open source constrained global optimization tool for Python.

Nyikosa, F. M., Osborne, M. A., and Roberts, S. J. (2018). Bayesian optimization for dynamic problems. *arXiv preprint arXiv:1803.03432*.

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2010). Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Rastrigin, L. A. (1974). Systems of extremal control. *Mir, Moscow*.

Silva, A. P., Rodríguez, O., de la Mota, F., and FI, S. A. (2020). Simulation of elements of industry 4.0 applied to the production processes of a company dedicated to the manufacture of plastic products. In *European Model-*

ing & Simulation Symposium, pages 84–92. CAL-TEK Srl.

Wang, H., van Stein, B., Emmerich, M., and Back, T. (2017). A new acquisition function for bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512. IEEE.

Yazdani, D., Branke, J., Omidvar, M. N., Li, X., Li, C., Mavrovouniotis, M., Nguyen, T. T., Yang, S., and Yao, X. (2021a). Ieee cec 2022 competition on dynamic optimization problems generated by generalized moving peaks benchmark. *arXiv preprint arXiv:2106.06174*.

Yazdani, D., Cheng, R., Yazdani, D., Branke, J., Jin, Y., and Yao, X. (2021b). A survey of evolutionary continuous dynamic optimization over two decades—part a. *IEEE Transactions on Evolutionary Computation*, 25(4):609–629.

Yazdani, D., Cheng, R., Yazdani, D., Branke, J., Jin, Y., and Yao, X. (2021c). A survey of evolutionary continuous dynamic optimization over two decades—part b. *IEEE Transactions on Evolutionary Computation*, 25(4):630–650.

Zaatari, S. E., Wang, Y., Hu, Y., and Li, W. (2022). An improved approach of task-parameterized learning from demonstrations for cobots in dynamic manufacturing. *Journal of Intelligent Manufacturing*, 33(5):1503–1519.

Zhang, Q., Zheng, F., Chen, Q., Kapelan, Z., Diao, K., Zhang, K., and Huang, Y. (2020). Improving the resilience of postdisaster water distribution systems using dynamic optimization framework. *Journal of Water Resources Planning and Management*, 146(2):04019075.