Optimizing Computational Efficiency: A Comparative Study on Sequential Designs for Function Optimization

Group 2: Graciela McDonogh-Wong, Ashley Munayco, Ellen Wei, Kyle Lee
Statistics 143, Spring 2024

Abstract
Function optimization is a foundational area of study, integral to nearly all quantitative fields and central to most machine learning algorithms and predictive modeling projects. While computer models facilitate the exploration of complex function landscapes, they often incur high computational costs. To address this challenge, alternative approaches such as sequential design or adaptive design are employed. In this paper, we investigate sequential designs for function optimization, examining their effectiveness and efficiency in minimizing computation expense. We evaluate and inform our comparison by introducing three key factors: the type of surrogate model used, the infill criteria applied, and the initial design algorithm. Our results demonstrate the impact of these factors on the optimization process and offer insights into selecting optimal configurations for various optimization tasks.

1 Introduction

Optimizing complex functions often entails high computational costs, particularly when evaluations are time-consuming or expensive. Sequential designs are adept at navigating complex optimization landscapes, particularly when function evaluations are costly or noisy. They adaptively adjust their search strategy, making them robust in scenarios with high-dimensional or unknown objective functions.

In this paper, we focus on sequential designs for function optimization, investigating their effectiveness and efficiency in minimizing computation expenses. Specifically, we examine the impact of surrogate model selection, infill criteria, and initial design strategies on the optimization process. We conducted our analysis and study using R, which we used to create dozens of models that run through various permutations of our comparison criteria. Through our analysis, we aim to provide insights into selecting optimal configurations for diverse optimization tasks, ultimately enhancing the effectiveness of function optimization in practical applications.

2 Sequential Design Background

This section describes the general sequential design [1] and sequential model-based optimization (SMBO) setup, as well as other methods presented in lecture related to this topic.

2.1 Sequential Design Workflow

1. Select an initial design with n runs and obtain data by evaluating f at the chosen points.

$x_i, y_i = f(x_i)), \qquad i = 1,\ldots,n$

2. Fit a gaussian-process (GP) surrogate model on the data to all evaluated points and corresponding values.
3. Based on the GP model and infill criterion, determine a new point (x_{new}) such that x_{new} maximizes the expected improvement. The point should either have a good expected objective value or high potential to improve the quality of the surrogate model.
4. Obtain $y_{new} = f(x_{new})$
5. Add (x_{new}, y_{new}) to the data and re-build or update the model.
6. If the budget is not exhausted and no other termination criteria is met, repeat steps 3 to 5 until convergence or budget exhaustion.

2.2 EGO and TREGO

There are many existing methods and frameworks for sequential design for function optimization, notably Efficient Global Optimization (EGO) and Trust-Region Efficient Global Optimization (TREGO). EGO was first introduced in a 1998 paper in the Journal of Global Optimization by Jones et. al [2]. EGO was designed as a response to the high computational cost of many engineering optimization problems, and involves fitting response surfaces to data collected by evaluating the objective and while maintaining optimization accuracy.

TREGO expands upon EGO by alternating regular EGO steps and local steps within a trust region, developed in response to EGO's struggles to scale in higher dimensions. We initially included EGO and TREGO in our study, but due to the relatively similar workflow of sequential designs, as we progressed further in our analysis we chose to instead use one R package to manipulate the levels of our three criterion we are comparing.

This package was mlrMBO, which is a configurable R toolbox for Bayesian optimization. More discussion on mlrMBO and its role in this study will be discussed in Section 4 Experimental Setup.

3 Evaluation Criteria

In this section, we introduce the three evaluation criteria we used to assess the effectiveness of our various sequential designs.

3.1 Surrogate Models

Surrogate models are simplified models that approximate more complex and computationally expensive simulations, providing a cheaper way to estimate the objective function. Surrogate models predict the function values at untested points based on previously evaluated points. In our study, we used Kriging and Random Forest as our surrogate models.

Kriging, also known as gaussian process regression, is widely used in statistics as a method of interpolation that uses a Gaussian process to predict the value of the objective function. A Kriging model is a generalized linear regression model that accounts for the correlation in the residuals between the regression model and the observations [3]. We chose Kriging for its flexibility and accuracy in modeling complex, nonlinear functions. Its probabilistic nature allows it to quantify

uncertainty, which is beneficial for guiding the optimization process in sequential designs.

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the average prediction of the individual trees. Though the literature on Random Forests to construct surrogate models is less well studied than Kriging, multiple studies have shown RF models to perform as well as Kriging and other common surrogate models (like SVMs) [4]. Additionally, RF is a good alternative to Kriging in scenarios where the relationship between input and output variables is highly non-linear and noisy.

3.2 Infill Criterion

Infill criterion, also known as acquisition functions, guides the selection of new points to evaluate in the optimization process and attempts to trade-off exploitation and exploration. We used three different infill criteria in our study.

Expected improvement, or EI, can be defined as:

$$EI(x) := E(I(x)),$$

where the random variable $I(x)$ represents the potential improvement over the currently best observed function value.

Augmented expected improvement, or AEI, consists of the regular EI multiplied by a penalization function that accounts for the lower payoff of replicates. AEI was the most difficult infill criterion to work with, and we skipped a few of the initial planned models we had with it with higher dimension test functions due to issues.

We also explored confidence bound (CB) as an infill criterion, which provides a simpler approach to balancing exploration and exploitation as compared to EI and AEI. CB is calculated as the mean predicted value plus a multiple of the standard deviation.

3.3 Initial Design Algorithm

The initial design algorithm determines the initial set of points to evaluate before applying the sequential design method. The choice of initial points can significantly impact the efficiency and effectiveness of the optimization process.

Random Latin Hypercube Sampling (LHS) is a statistical method that generates a random sample of points, ensuring that each variable is sampled uniformly. Optimum LHS is a version of LHS that optimizes the placement of points to maximize the minimum distance between them, ensuring better coverage of the design space. Maximin LHS aims to **max**imize the **min**imum distance between points in the sample, ensuring that points are as spread out as possible within the design space.

4 Experimental Setup

4.1 Test functions

The Cosine Mixture Function [5] is a one-dimensional function with multiple local minima, defined as:

$$f(x) = -0.1 \sum_{i=1}^{d} cos(5\Pi x_i) - \sum_{i=1}^{n} x_i^2,$$

subject to $x_i \in [-1, 1]$. Despite the presence of many local minima, it was selected as there is a clear global minimum.

The Branin function [5] is a two-dimensional function with three local minima widely used in optimization literature, defined as:

$$f(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)cosx_1 + s$$

subject to $x_1 \in [-5, 10]$, $x_2 \in [0, 15]$.

The Hartmann function [5] is a six-dimensional function that has 6 local minima and is usually evaluated using a hypercube $x_i \in (0, 1)$, for all i = 1, …, 6. It is defined as [2]:

$$f(\mathbf{x}) = -\frac{1}{1.94}\left[2.58 + \sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2\right)\right]$$

4.2 Comparison Metrics

To evaluate the performance of the sequential design approaches, we created comparison tables isolating the following metrics:

1. Convergence
    a. Convergence is measured by the number of interactions required to reach the best solution, which we denoted as `best.ind` or BI
2. Computation Efficiency:
    a. Train Time (TT): total time taken to train the surrogate model that produced the points
    b. Propose time (PT): time needed to propose new points for evaluation based on the surrogate model and infill criterion
    c. Standard error (Sum SE): sum of the squared standard errors across iterations for the infill criterion

4.3 mlrMBO

mlrMBO is a highly configurable R toolbox for model-based / Bayesian optimization of black-box functions [6]. We used mlrMBO to compare various permutations of test function, surrogate model, initial design, and infill criteria. To organize our findings, we created four model comparison tables: Kriging + Cosine-Mixture, Kriging + Branin, Random Forest + Branin, and Kriging + Hartmann, which each consist of nine models with varying initial designs and infill criteria.

4.4 General MBO Workflow

1. Define objective function and its parameters using the package smoof
    a. Cosine mixture function (1D), Branin (2D), or Hartmann(6D)
2. Generate initial design
    a. Random LHS, Optimum LHS, or Maximin LHS
3. Define mlr learner for surrogate model
    a. Kriging (default), Random Forest
4. Set up a MBO control object.
5. Start the optimization with run=mbo().
6. Visualize runs: exampleRun, plotExampleRun

4.5 Additional R Packages

Though mlrMBO was the main package we worked with in our analysis, this package built on other existing packages, including `DiceKriging`[7], which allowed us to estimate, validate, and predict kriging models, `smoof`[8], which provides

generators for a high number of single and multi-objective test functions allowing us to to work with and model with them, and `checkmate` [9], which performs argument checks.

5 Results

5.1 Kriging + Cosine-Mixture Function

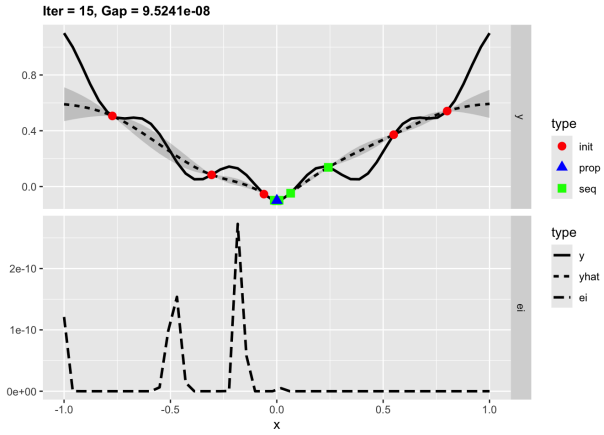
As aforementioned, we narrowed our study into four comparison tables. Our first table combined the one-dimensional cosine-mixture function with the kriging surrogate model. We ran nine models under these conditions, with different pairings of infill criteria and initial design, altogether comparing 36 models.

In one-dimension, the `plotExampleRun` in `mlrMBO` generates a plot with two panels. The top panel shows the actual objective function (y), which is a solid line, and the surrogate model's prediction, which is a dashed line with a confidence interval. The top panel also plots the optimization path, plotting the initial points and points proposed by the algorithm as it runs, showing how the optimization process goes through the search space.

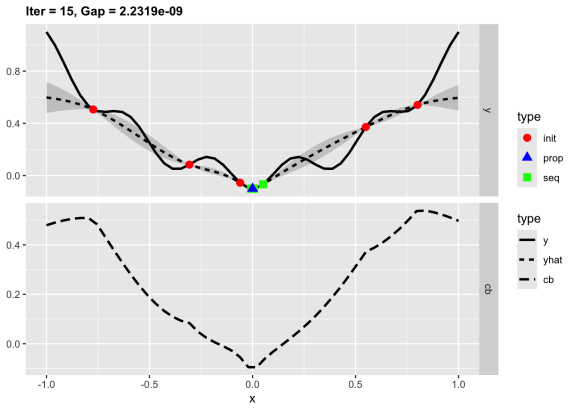
The second panel displays the acquisition function/infill criterion, which indicates where the model finds the next best point to sample is, balancing exploration and exploitation. Peaks in this plot suggest where the optimization algorithm is likely to sample next, as these points are expected to yield significant improvements. Each of the following plots show the model at its 15th iteration.

5.1.1. Plots

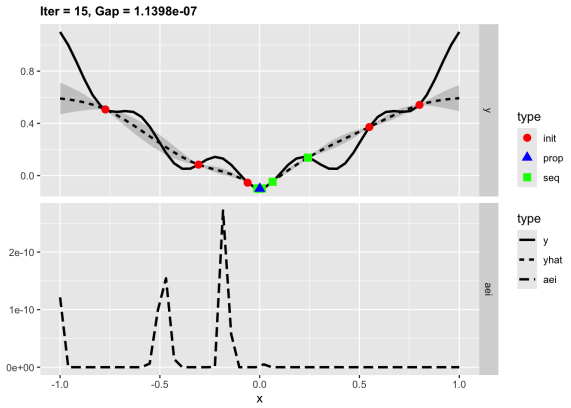
Model 1: EI + Random



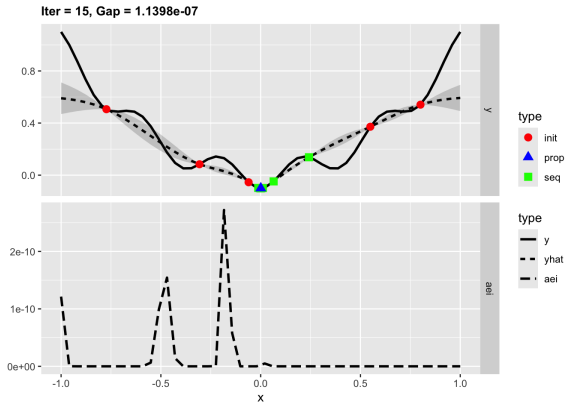
Model 2: CB + Random



Model 3: AEI + Random



Model 4: EI + Maximin



Additional graphs in appendix.

5.1.2 Comparison Table 1:

Model	BI	TT	PT	Sum SE
EI + Random	17	0.188	1.595	0.013
CB + Random	18	0.278	1.749	0.005
AEI + Random	18	0.166	2.158	0.013
EI + Maximin	14	0.127	1.44	0.124
CB + Maximin	18	0.159	1.423	0.083
AEI + Maximin	19	0.131	1.992	0.131
EI + Optimum	20	0.143	1.437	0.093
CB + Optimum	20	0.155	1.397	0.093
AEI + Optimum	20	0.145	2.175	0.093

BI: number of iterations required for the model to find the best solution; TT: train time - time taken to train the surrogate model; PT: propose time - time taken to propose the next point; Sum SE: sum of standard errors

5.1.3 Analysis

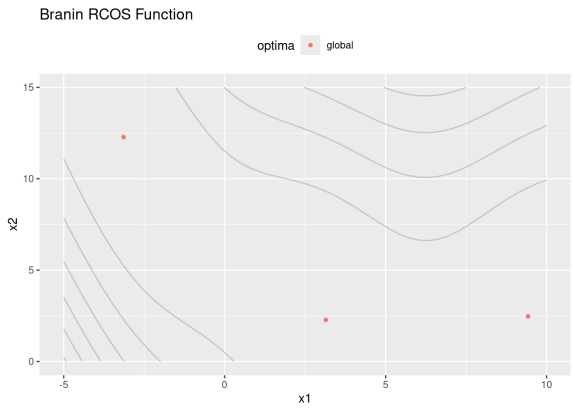
From the plots and comparison table, the combination of maximin LHS initial design and expected improvement achieved the best solution in the fewest iterations. This is shown in its plot, where the surrogate model line traces the true function line very well compared to other models at the same iteration, and in its table results: EI + Maximin required the fewest iterations, had the lowest train time, and had a somewhat low propose time. However, it is important to note that this model had a relatively high sum SE, suggesting it was less accurate than other methods despite its efficiency in iterations and time. CB + Random provided the highest accuracy, but had a relatively high training time. For scenarios where computation efficiency is important, EI + Maximin would be a good option, while CB + Random is a good choice for high accuracy scenarios. When both efficiency and accuracy are desired, AEI + Optimum is a good option.

5.2 Kriging + Branin Function

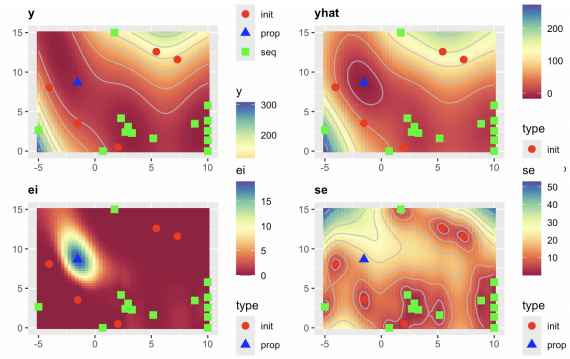
In two-dimensions, `plotExampleRun` outputs a four panel plot, which map the true function (y), surrogate model prediction (yhat), acquisition function (EI/CB/AEI) and the standard error. During initial testing and plotting, detailed further in the appendix, we found that at least 10 iterations were necessary for these plots to be meaningful, so the plots show the models at iteration 15. We had initially set out to create nine models/plots for this table, but while modeling due to issues had to skip models 12. AEI + Random (used standard error infill criteria instead), 14. CB + Maximin, 15. AEI + Maximin, 17. CB + Optimum, and 18. AEI + Optimum.

5.2.1 Plots

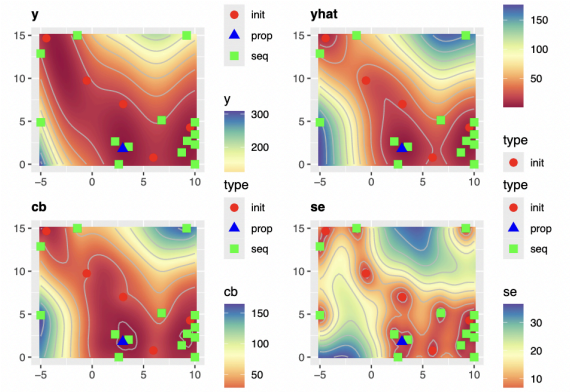
Branin RCOS Function



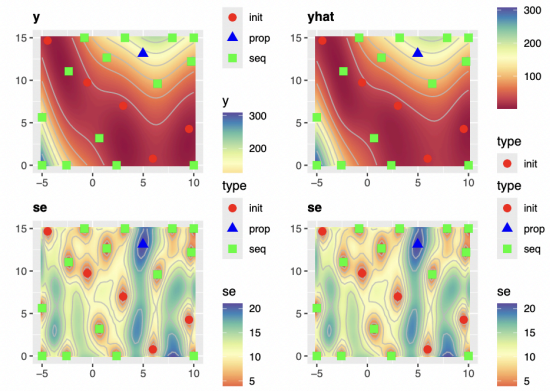
Model 10: EI + Random



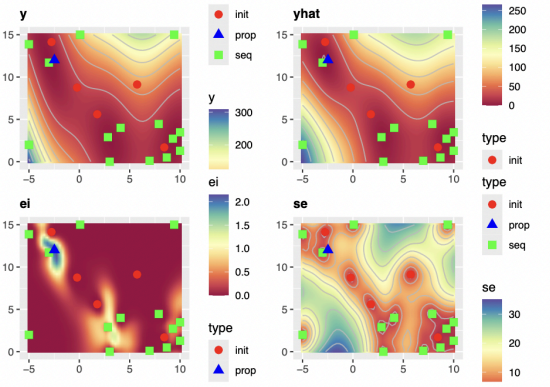
Model 11: CB + Random



Model 12: Standard Error + Random



Model 13: EI + Maximin



5.2.2 Comparison Table 2

Model	BI	TT	PT	Sum SE
EI + Random	3	0.479	2.246	14281.561
CB + Random	15	0.136	1.776	46954.761
SE + Random	3	0.13	1.719	4480.553
EI + Maximin	20	0.18	2.477	11242.49
EI + Optimum	18	0.479	2.246	14281.561

5.2.3 Analysis

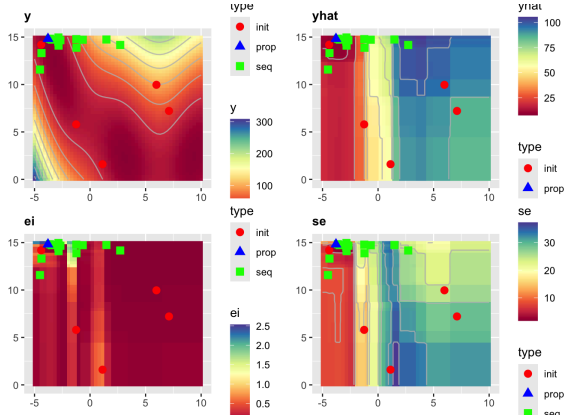
Upon analysis of the plots as compared with the Branin RCOS function, and of the convergence metrics in comparison table 2, EI + Random and EI + Optimum have the same performance metrics, indicating they may have similar optimization paths and model performances. SE + Random, which was a model we plotted after running into issues with AEI + Random, achieved the lowest sum SE, indicating the best prediction accuracy, and required very few iterations. For quick solutions, EI + Random or SE + Random appear most effective, with SE + Random being particularly successful in terms of prediction accuracy.

5.3 Random Forest + Branin Function

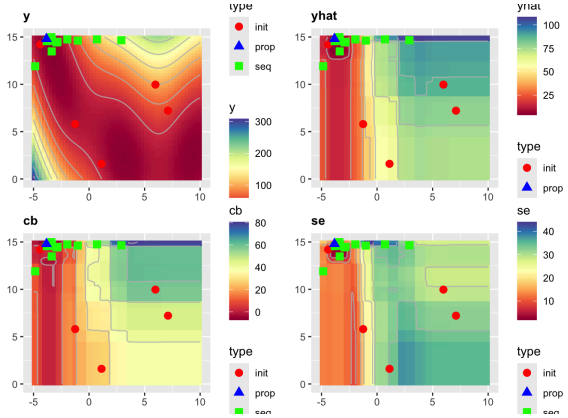
In this section, we combined the surrogate model random forest with the 2D Branin function. We had initially set out to create nine models/plots for this table, but while modeling due to issues had to skip the three models that used AEI as infill criterion: Model 21. AEI + Random, 24. AEI + Maximin, and 27. AEI + Optimum.

5.3.1 Plots

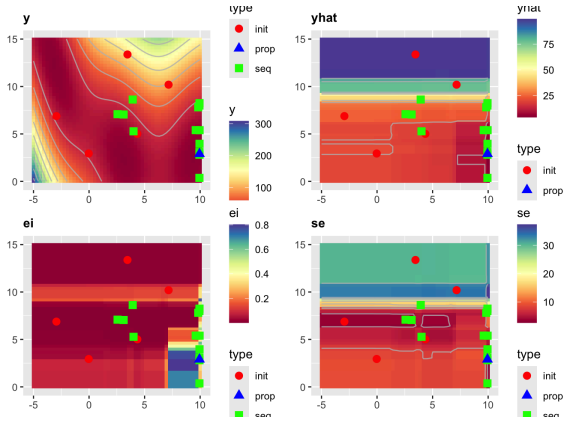
Model 19: EI + Random



Model 20: CB, Random



Model 22: EI, Optimum



5.3.2 Comparison Table 3

Model	BI	TT	PT	Sum SE
EI, Random	20	0.041	5.39	13398.031
CB, Random	6	0.043	4.889	21802.724
EI, Optimum	13	0.041	4.921	7385.306
CB, Optimum	20	0.039	5.104	5320.314
EI, Maximin	18	0.038	4.683	15334.908
CB, Maximin	16	0.043	4.61	13954.106

5.3.3 Analysis

EI + Optimum seems to be the best model configuration based on its balanced performance and low sum SE. The fastest to converge was CB + Random, which also had the highest SE, suggesting lower accuracy. All of the models in this table had similar train times, indicating the choice of initial design and infill criteria had more of an effect on propose time and standard error rather than training time.

5.4 Kriging + Hartmann Function

For our last comparison table, we combined the Kriging surrogate model with the six-dimensional Hartmann test function, Given the complexity and high dimensionality of this function, we opted to not include any plots to maintain clarity and focus on the comparison metrics.

5.4.1 Comparison Table 4

Model	BI	TT	PT	Sum SE
EI + Random	25	0.167	1.636	0.526
CB + Random	20	0.183	1.617	0.657
AEI + Random	21	0.179	1.877	0.58
EI + Maximin	25	0.232	1.543	0.041
CB + Maximin	25	0.185	1.526	0.257
AEI + Maximin	25	0.195	1.905	0.109
EI + Optimum	23	0.16	1.611	0.616
CB + Optimum	24	0.179	1.572	0.422
AEI + Optimum	25	0.162	2.066	0.514

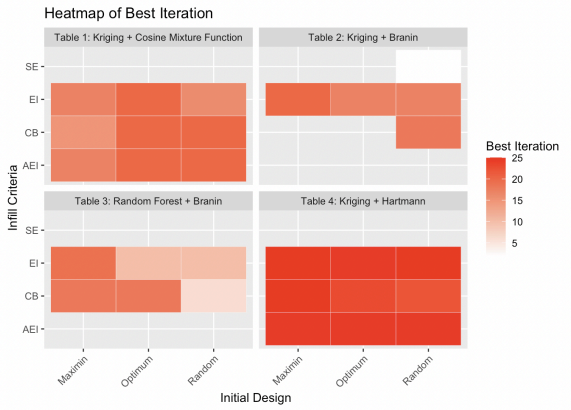
5.4.2 Analysis

As expected given the complexity of the test function, across the board these models required more iterations. EI + Maximin appears to be a good choice for model configuration due to its low SE and typical amount of iterations required. For faster convergence, CB + Random is an option, but could result in lower prediction accuracy.

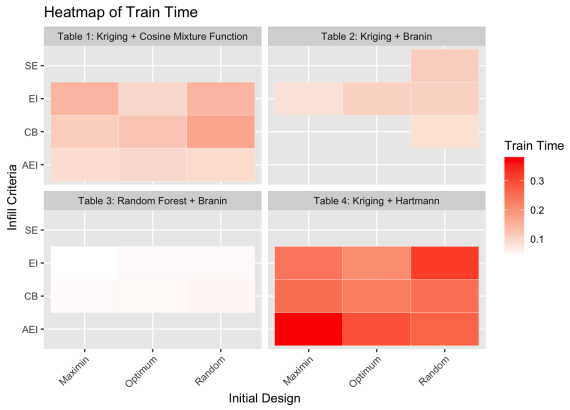
5.5 Heatmaps

Through visuals and comparison tables, we have explored 29 models, which we compiled into the heatmaps below. The heatmaps visually represent the performance metrics of the models across the different combinations our study explored. Each heatmap uses shades of red to indicate the differences in our four key comparison metrics: best iteration, train time, propose time, and sum SE.

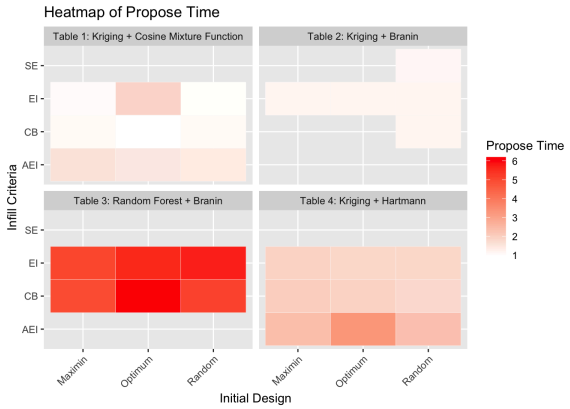
5.5.1 Best Iteration



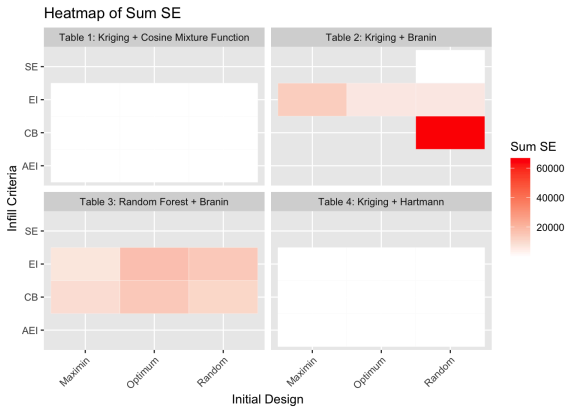
5.5.2 Train Time



5.5.3 Propose Time



5.5.4 Sum SE



From these heatmaps, we can observe a few patterns that support our previous results. In terms of being the most supportive to work with, EI and Random LHS consistently led to successful models, and is a good choice in terms of best iteration.

These heatmaps also allow for easy comparison between the two surrogate models, Kriging and Random Forest, which were both paired with the Branin test function. As shown across the heatmaps, Kriging and Random Forest performed similarly in terms of best iteration, and Random Forest tended to have lower sum SE. However, propose time for Random Forest was much higher than Kriging, highlighting the tradeoff between the complexity of the models and the computation resources needed.

6 Conclusion

6.1 Combined Analysis

Across the four tables, EI + Maximin shows consistently high accuracy across different functions and models, particularly when working with the Kriging surrogate model. The effectiveness of this paired approach can be attributed to its balanced exploration and exploitation mechanism, enhancing convergence towards the global optimum.

Confidence Bound (CB) models often had lower SE values, suggesting they provide a stable and accurate approach across different functions. Augmented Expected Improvement (AEI) models had higher propose times and were effective in some high-dimensional settings, but overall proved very difficult to work with and led to some failed models.

Optimum LHS designs generally provide consistent performance across infill criteria scenarios but may require more iterations (with the specific infill criterion

EI, Maximin LHS designs performed well). This consistency suggests that Optimum LHS designs can serve as a reliable initial sampling strategy, particularly when computational resources are a concern.

Finally, between the two surrogate models, Kriging generally had lower propose times compared to Random Forest, suggesting that it was more computationally efficient.

6.2 Challenges & Recommendations

As mentioned in Section 5, we encountered issues with 8 of the initial 36 models. Specifically, the data generation during the prediction process was not functioning correctly, likely due to potential missing values or problems within the input design. All of the models that were skipped utilized either CB or AEI as the infill criterion. Notably, while the mlrMBO package allows the establishment of custom infill criteria, its default options are EI and LCB (lower confidence bound), suggesting it likely behaves better with the default infill criterion. We opted to shift one of the models with issues, model 12, from AEI + Random to SE + Random.

Another challenge was figuring out which specific models to include, given the multitude of possible permutations. After narrowing our scope, we settled on creating 36 models, but, as aforementioned, ran into issues with a few of them, resulting in our running of 29 models. Though we were able to review many different combinations and learn more about sequential design in this way, we ultimately spent more time working with the Kriging surrogate model than Random Forest, so equal exploration to Kriging and Random Forest was not fully accomplished. Also as we scaled the dimensionality higher we ran into more problems during testing in R; so, this led to the test function Hartmann (6D) only being used with Kriging.

While the Kriging model is a popular choice due its flexibility, we would still recommend more in depth comparisons between additional surrogate models like SVMs. We recommend the use of the package mlrMBO; though there is a slight learning curve, once you become familiar with the package, it proves to be very flexible during testing for the three criteria looked at during this study, i.e., surrogate models, infill criteria, and initial design algorithm. Plus, while not explored in this report, mlrMBO has more extensive options such as multiple termination criteria.

7 References

[1] Xu, H., (2024, April). Sequential Designs and Efficient Global Optimization [Lecture and Powerpoint slides]. https://bruinlearn.ucla.edu/courses/187220/files/17007005?module_item_id=6716679

[2] Jones, D.R., Schonlau, M. & Welch, W.J. Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization 13, 455–492 (1998). https://doi.org/10.1023/A:1008306431147

[3] Wu, X.,Wang, C. & Kozlowski, T. (2017). Kriging-based Surrogate Models for Uncertainty Quantification and Sensitivity Analysis.

[4] Preuss, M., Wagner, T., Ginsbourger, D. (2012). High-Dimensional Model-Based Optimization Based on Noisy Evaluations of Computer Games. In: Hamadi, Y., Schoenauer, M. (eds) Learning and Intelligent Optimization. LION 2012. Lecture Notes in Computer Science, vol 7219. Springer, Berlin, Heidelberg.

[5] Bingham, D., Surjanovic, S., (n.d.). Virtual Library of Simulation Experiments, Test Functions and Datasets. https://www.sfu.ca/~ssurjano/optimization.html

[6] Bischl B, Richter J, Bossek J, Horn D, Thomas J, Lang M (2017). *mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions*. https://arxiv.org/abs/1703.03373.

[7] Roustan, O., Ginsbourger, D., Deville, Y. (2012). "DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization." *Journal of Statistical Software*, 51(1), 1–55. https://www.jstatsoft.org/v51/i01/.

[8] Bossek, J. (2017). "smoof: Single- and Multi-Objective Optimization Test Functions." *The R Journal*. https://journal.r-project.org/archive/2017/RJ-2017-004/index.html.

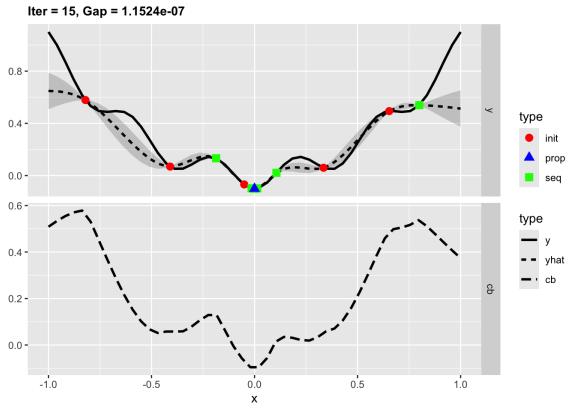
[9] Lang, M. (2017). "checkmate: Fast Argument Checks for Defensive R Programming." *The R Journal*, **9**(1), 437–445. doi:10.32614/RJ-2017-028.

8 Appendix

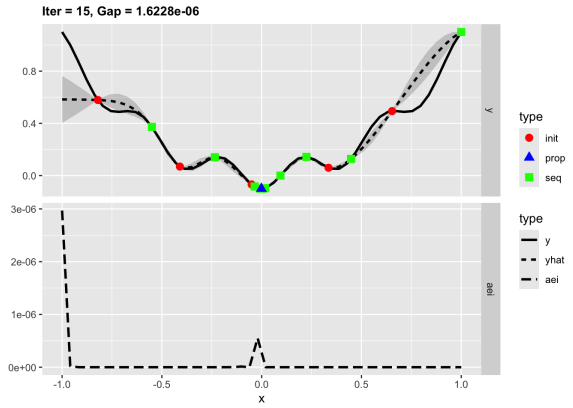
8.1 Additional Model Plots

Comparison Table 1: Kriging + Cosine-Mixture

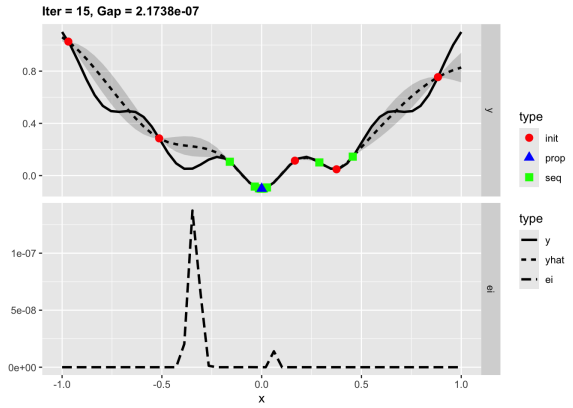
Model 5. CB + Maximin



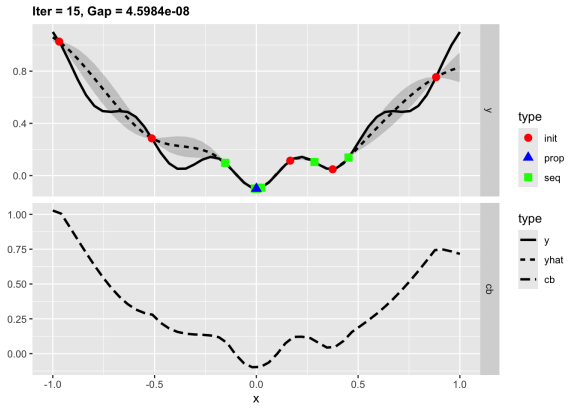
Model 6: AEI + Maximin



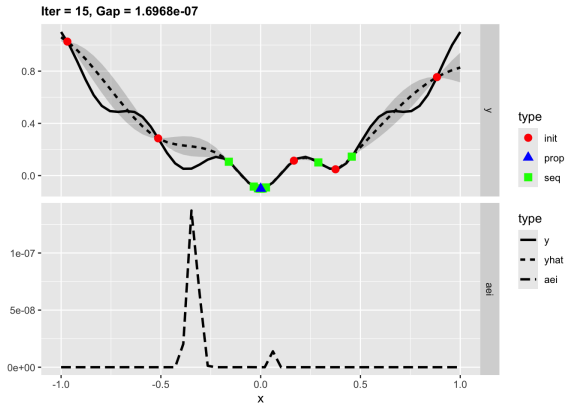
Model 7: EI + Optimum



Model 8: CB + Optimum

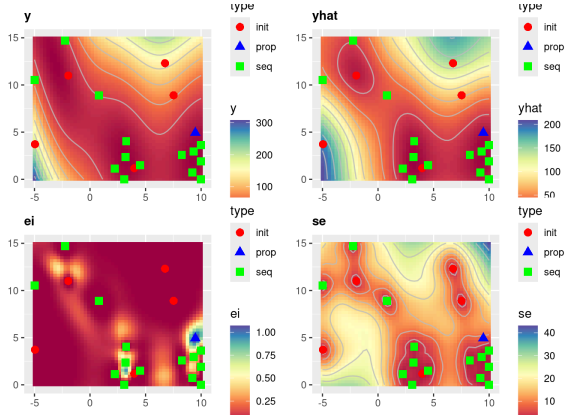


Model 9: AEI + Optimum



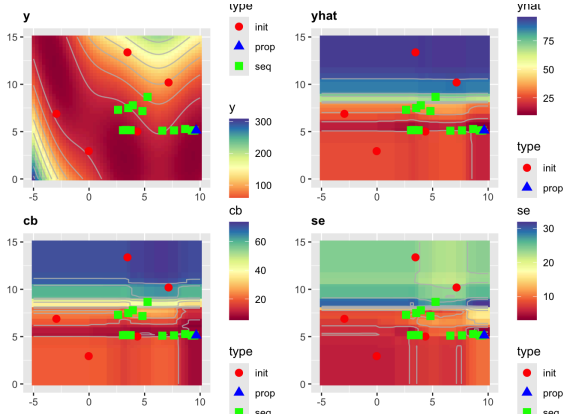
Comparison Table 2: Kriging + Branin

Model 16: EI + Optimum

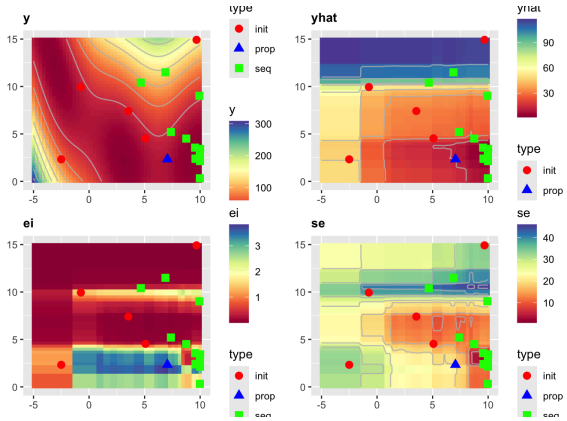


Comparison Table 3: RF + Branin

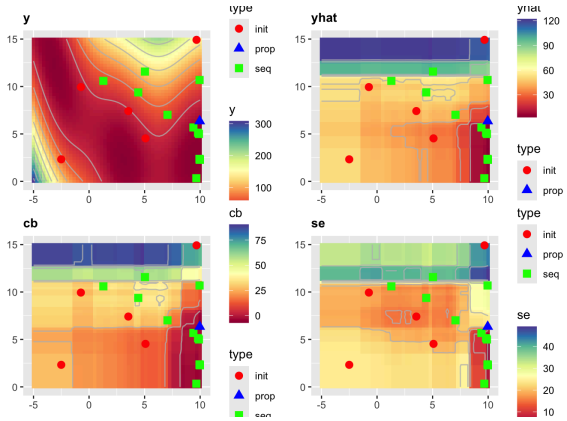
Model 23: CB + Maximin



Model 25: EI + Optimum



Model 26: CB + Optimum

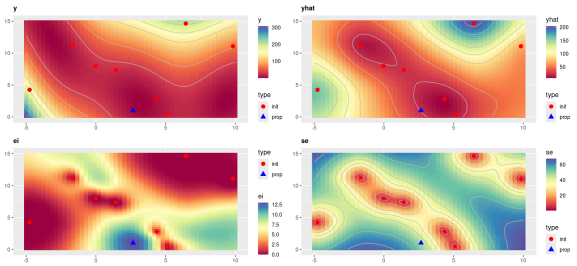


8.2 Branin Plots

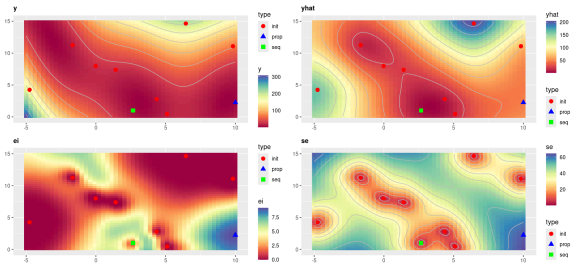
As noted in section 5, we decided on 15 iterations as our standard when modeling. This was due to initial testing of plotting the Branin function in mlrMBO at various iterations (1, 2, 10). Post plotting, we saw that we would clearly need at least 10, and

decided on 15 iterations. For reference, below are our initial plots.

1 Iteration:



2 Iterations:



10 Iterations: