IRIS: Iterative and Intelligent Experiment Selection

Raoufehsadat Hashemian
University of Calgary
Calgary, AB, Canada
rhashem@ucalgary.ca

Niklas Carlsson
Linköping University
Linköping, Sweden
niklas.carlsson@liu.se

Diwakar Krishnamurthy
University of Calgary
Calgary, AB, Canada
dkrishna@ucalgary.ca

Martin Arlitt
University of Calgary
Calgary, AB, Canada
martin.arlitt@ucalgary.ca

ABSTRACT

Benchmarking is a widely-used technique to quantify the performance of software systems. However, the design and implementation of a benchmarking study can face several challenges. In particular, the time required to perform a benchmarking study can quickly spiral out of control, owing to the number of distinct variables to systematically examine. In this paper, we propose **IRIS**, an **I**te**R**ative and **I**ntelligent Experiment **S**election methodology, to maximize the information gain while minimizing the duration of the benchmarking process. IRIS selects the region to place the next experiment point based on the variability of both dependent, i.e., response, and independent variables in that region. It aims to identify a performance function that minimizes the response variable prediction error for a constant and limited experimentation budget. We evaluate IRIS for a wide selection of experimental, simulated and synthetic systems with one, two and three independent variables. Considering a limited experimentation budget, the results show IRIS is able to reduce the performance function prediction error up to 4.3 times compared to equal distance experiment point selection. Moreover, we show that the error reduction can further improve through system-specific parameter tuning. Analysis of the error distributions obtained with IRIS reveals that the technique is particularly effective in regions where the response variable is sensitive to changes in the independent variables.

Categories and Subject Descriptors

System performance [**Performance benchmarking**]: Controlled Experimentation

Keywords

Experiment selection, Benchmarking, performance function, Kriging

ICPE'17, April 22-26, 2017, L'Aquila, Italy

DOI: http://dx.doi.org/10.1145/3030207.3030225

1. INTRODUCTION

Careful performance benchmarking is important to understand how computer systems will operate under different loads and configurations. Benchmark results are often used to optimize a system's configuration or to prepare a capacity plan, and must therefore be accurate. However, creating and modifying testbeds to accurately characterize applications under different configurations and workloads is often time consuming and expensive. As an example, a previous benchmarking study on interactive Web applications deployed on multicore hardware [6] required around 1,800 experiments to examine six independent variables in a full factorial setup, even with three of the variable being binary.

At a high level, careful performance benchmarking involves quantifying the performance impact that different system configurations and workload parameters, i.e., independent variables, have on various system performance metrics, i.e., response variables. To capture these relationships, i.e., the underlying performance function of the system under test, it is important to determine (i) where in the independent variable space to place experimental measurement points, and (ii) how to estimate the underlying function that relate these variables, given a set of such experimental points. Most prior work in this area have been focused on the second question, typically by comparing the accuracy of different function prediction techniques and their abilities to build statistically inferred models or interpolated functions of system performance [17] [3] [5]. In contrast, we focus on where in the independent variable space to place experimental measurement points, to achieve a desirable accuracy in a cost-effective manner.

In particular, we present **IRIS**, an **I**te**R**ative and **I**ntelligent Experiment **S**election methodology which reduces the number of experimental points needed for good performance function prediction and evaluate its effectiveness when applied to a variety of performance functions. IRIS allows us to focus on regions of greater interest more quickly and efficiently. Furthermore, in contrast to most existing experiment selection techniques, IRIS is iterative and in each step places new test points where they are expected to provide maximum information about the underlying performance relationships.]

To illustrate the value of careful, iterative point selection, consider a system in which the independent variable is the user population of a Web server and the response variable is the server's response time. A naive benchmarking approach

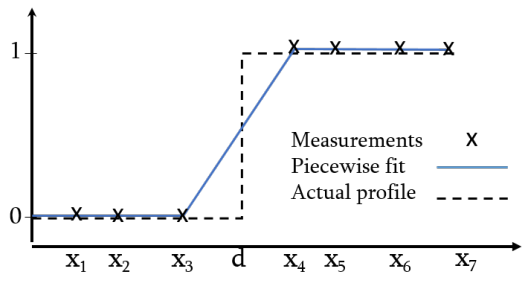

Figure 1: Sample point toy example.

would perform experiments for user population sizes evenly split over the expected range of population sizes. The problem with this approach is that the response time may be relatively constant for a wide range of (smaller) user populations, but may increase dramatically for a small range of (larger) user population sizes, for which the system utilization approaches one. The naive approach places unnecessary points in the flat region of the independent variable space, where performance function estimation is easy. A more efficient approach would instead place additional points in the region that sees the most change, and where it is more difficult to estimate performance accurately. If the approximate shape of the underlying performance function is known or can be quickly (and adaptively) learned, a more intelligent sampling strategy, based on the current knowledge and estimates of the underlying shape, can instead redistribute experimental points to the region with the bigger changes in the response variable. This can in turn allow for better tracking of the performance function and result in more accurate benchmarking. IRIS leverages this observation.

This paper makes three main contributions. First, we introduce the IRIS technique for performance benchmarking. IRIS carefully adapts the experimental sample points selection based on the current knowledge learned through existing system models and the results obtained from prior experiments. To capture the importance of these aspects, IRIS allows the user to determine the weight given to the initial point selection, e.g., based on offline system models, relative to the iterative point selection process. IRIS uses a single tuning parameter α that weights the desire to evenly cover the full independent variable space and placing particular focus on regions with larger changes in the response variable.

Second, we evaluate IRIS using several experimental, simulation based, and synthetic datasets for systems with up to three independent variables. These datasets represent a wide range of system performance functions, allowing us to provide insights into parameter tuning under different types of performance behaviours. The results show that IRIS can significantly outperform baseline approaches based on equal distance point selection. In particular, considering scenarios with limited experimentation budget, we measure up to 4.3 times error reduction for the systems with one independent variable and 2.8 times error reduction for the systems with multiple independent variables. Moreover, IRIS does not cause a significant degradation in prediction accuracy for any of the eight cases we examine.

Third, we perform an in-depth analysis of different aspects that impact the effectiveness of IRIS. In particular, we focus on the criteria behind the selection of tuning parameters, the effect of using an educated guess about the performance function for the initialization of IRIS, the individual effect of iterative versus weight based point selection, and the impact of IRIS on the prediction error distribution. Our analysis provides insights into how to best use IRIS to obtain the ideal trade-off between prediction accuracy and the number of experiments needed.

The remainder of this paper is organized as follows. Section 2 introduces IRIS and describes its behaviour under different dimensions of independent variables. Section 3 explains our evaluation process. Section 4 compares IRIS with baseline approaches for different systems. Section 5 presents results that further characterize the behaviour of IRIS. Section 6 discusses related work. Section 7 concludes the paper.

2. IRIS

2.1 Motivation

When using measurements to estimate the performance function for an underlying system, not all measurement points are of the same value. Some measurements provide little new information about the system, while other points may be very important to improve the overall accuracy of the performance function. To illustrate this point, consider the simple toy example illustrated in Figure 1. Here, the measured response variable y_j is sampled from an underlying step function $f_d(x)$, which is 1 when the independent variable x is greater than d, and 0 otherwise. Furthermore, assume that we use a piecewise linear function between the measurement points (x_j, y_j) to estimate the underlying function. In this case, the two measurement points closest to either side of the step at $x = d$ have the greatest impact on the accuracy of the model, whereas any points that are on the horizontally flat regions, i.e., between points with the same $y = f_d(x)$ value; either all $y = 0$ or all $y = 1$, provide no additional information.

Taking the above toy example one step further, let us consider the case where we have an initial set $\mathcal{S}$ of measurement points, and we want to use additional experiments to determine more precisely where the step takes place. In this case, assuming that we have at least one point on either side of the step, a sensible algorithm would be to use a binary search, in which a measurement is added at the midpoint $x' = \frac{x_a + x_b}{2}$ between the two points $x_a = \max_{j \in \mathcal{S}}(x_j | y_j = 0)$ and $x_b = \min_{j \in \mathcal{S}}(x_j | y_j = 1)$ closest to and on either side of the step. This procedure is then repeated until the maximum error $(x_b - x_a)$ of the current estimate of the x-value of the step is less than some desired maximum error or until we have reached our measurement point budget; in which case the error is minimized, given a certain point budget.

In addition to showing that not all points have the same value, this example also shows that the accuracy of the model can be significantly improved by adapting where the measurements are placed based on the current knowledge of the system. For example, a naive solution that spreads all measurement points evenly across the full measurement region would require exponentially more measurement points to achieve the same accuracy of where the step is located. To see this, note that the error of the binary search (above) reduces as $O(2^{-|\mathcal{S}|})$, whereas the maximum error when spreading all points evenly is proportional to $O(\frac{1}{|\mathcal{S}|})$.

While real systems typically have more than one point of interest and typically involve more than one independent variable, this example clearly shows that (i) there are significant advantages to selecting measurement points carefully when building a performance function of the underlying system, and (ii) there are advantages to making use of past measurements when selecting future measurement points. In the following, we describe how IRIS generalizes the approach illustrated on the above toy example to multiple independent variables (dimensions) and more general relationships between the response variable and the independent variables. Considering a limited point budget to keep the duration of the benchmarking exercise from spiraling out of control, IRIS carefully places measurement points based on the current knowledge of the system, to maximize the information gain and achieve the overall goal of providing the best possible performance function for the system. Equivalently, when the accuracy of the performance function can be explicitly evaluated, the same technique can easily be used to solve the (equivalent) problem of minimizing the number of measurements needed to achieve a given accuracy.

2.2 Methodology Overview

Our solution is simple and intuitive. It splits the point selection process into two phases. First, an *initial point selection* phase is used to obtain an initial sample point set $\mathcal{S}_i$, with $N_i = |\mathcal{S}_i|$ sample points. In the simplest case, these points are evenly spread across the independent variable space. This is possible even with very limited system knowledge. However, we can typically do better than this. For example, a user with a large number of initial measurements can leverage a queuing model or other system knowledge to carefully select points in regions of particular interest. While step functions around which a user may want to place points such as in the toy example are rare in real systems, most systems have known regions, e.g., high-utilization regions, with larger response time differences than in other regions. Later in this section, we describe how a queuing model can be used to carefully select the initial points to target some of these regions. In general, however, the initial point selection phase is only used to obtain an initial set of sample points and initialize the identification of region(s) of interest.

Second, an *iterative refinement* phase is used to identify the regions of most interest and to iteratively refine the measurement point selection. For this phase, we assume that we have $N \geq N_i$ current sample measurements and a total sample point budget N_t. During each of the remaining $(N_t - N)$ iterations, a new sample point is placed so as to greedily improve the accuracy of the performance function. More specifically, in each step, the independent variable space is divided into regions, e.g., the x-axis segments between the sample points in our toy example, and a weighted gain G_j is calculated for each region based on the variation in the response variable and the size of the region. The intuition here is that regions with bigger weighted gains are better candidates to add measurement points. At the end of each iteration, a new sample point is always added to the centroid (defined in Section 2.3) of the region with the biggest weighted gain. Section 2.3 provides the technical details on how regions are defined in multiple dimensions, and how the weighted gains are calculated for problems with different dimensionality using the current set of sample points.

2.3 Multi-dimensional Weighted Gains

To determine the region that may most benefit from an additional measurement point, we calculate a weighted gain for each region. To weight the importance of regions with large variations in the response variable y with the desire to have sufficient points in larger regions with smaller variations in the response variable, we calculate the gain factor G_j of region j as the product

$$G_j = A_j^{\alpha} \times R_j^{1-\alpha}, \tag{1}$$

where A_j is the normalized size[1] of region j, R_j is a normalized measure of the maximum observed difference in the response variable, and α ($0 \leq \alpha \leq 1$) is a parameter that weights the importance of these two factors. With smaller α values, IRIS gives higher importance to placing experiments in regions where there are likely to be larger changes in the response variable.

One independent variable: First consider cases with a single independent variable x. In this case, the size A_j of region j is equal to the difference in x-values between neighbouring measurement points and the response spread factor R_j is equal to the absolute difference in y-values between neighbouring measurement points. For example, assuming an ordered list of N measurement points for which $x_j \leq x_{j+1}$, we can now calculate the two factors as $A_j = x_{j+1} - x_j$ and $R_j = |y_{j+1} - y_j|$, for each of the $N-1$ regions. Referring back to the toy example, we note that the gain G_j of equation (1) would only be non-zero for the region that we would like to split into two regions, as long as $\alpha < 1$. This process ensures that we focus on the region of most interest.

For more general systems, there typically are multiple regions with non-zero gain. In these cases, our approach will greedily add a new point in the region with the greatest weighted gain. Since each new measurement point splits a region into multiple smaller regions, both A_j and R_j will be smaller for this region, effectively increasing the relative weight given to other regions, which may yet need to be split to reach a desired accuracy.

Two independent variables: In this case, the experiment candidates are selected from candidate regions in a two-dimensional plane. Here, each candidate region is defined as a triangle, with the corners of the triangle corresponding to close-by measurement points from the current sample set. For our triangulation, we use Delaunay triangulation [2]. Delaunay triangulations are relatively easy to calculate, guarantee a unique planar triangulation of the independent variable space, and generalize to multiple dimensions. By maximizing the minimum angle of all the angles of the triangles, Delaunay triangulations also provide triangles consisting of points with high proximity. This is important in our context, as we will select regions that will be further split.

For our gain calculation, the size factor A_j is calculated as the area of each triangle and the response difference factor R_j is calculated as the maximum absolute difference between the three corners of the triangle. For point selection, we consistently add an additional measurement point at the

[1]In the 1D case the size of a region is defined as a length, in the 2D case it is defined as an area, and in higher dimensions it is defined as a volume.

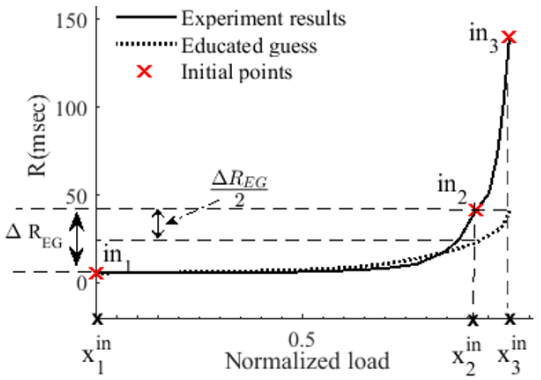

Figure 2: Initial point selection phase.

centroid of the region with the largest gain, where the centroid is used to capture the center point of each triangle.

Many independent variables: To further generalize the above Delaunay triangulation approach to the case when there are $K > 2$ independent variables to examine, we define each region as a convex hull of $K + 1$ points in the K-dimensional space. Similar to the two-dimensional case, the use of Delaunay triangulation ensures that the circumhypersphere of any triangle does not contain any interior points. For each region, the size factor A_j is calculated as the volume of each K-dimensional triangle and the response difference factor R_j is calculated as the maximum absolute difference between any pair of the the response variables as observed at the corners of the triangle. Finally, we again define the centroid as the center point of the hypersphere and place an additional measurement point at this centroid.

2.4 Detailed Demonstration

We now demonstrate the two phases of our methodology by applying the algorithm to a simple experimental dataset, focusing on a single independent variable. The system under study is a multicore Web server. The goal is to measure the user experienced response time, i.e., the response variable, as a function of the number of concurrent users or load, i.e., the independent variable. To understand the actual system function, we have previously conducted extensive experimentation, in which we measured the response time for a large number of experiments with equally spaced load values [7]. We normalize the load to be a number between 0 and 1. In Figure 2, the curve named "Experiment Results" shows the measured response time values as a function of normalized load.

(*i*) **Initial point selection:** In this phase, an educated guess is used to determine an initial set of N_i sample points. System models can help us identify reasonable estimates of the regions of interest and provide us with initial boundaries of those regions. A simple way to come up with such an educated guess is to run experiments for boundary values and use linear interpolation. An alternative method to place the initial points would be to use performance bound analysis [9] to determine the values for the upper and lower bounds of the response variable and the corresponding values of the independent variables. A more advanced approach would be to use a queuing model or other approximate models of the system to estimate the response variable in the desired range of independent variables. Such models or educated guesses of the underlying performance function can then be used to place the N_i initial points so as to cover variations in both the independent and response variables. In general, this can be done by splitting the parameter range so that the maximum (expected) gain G_j of any of the resulting regions is minimized. For our discussion here, we will assume a single independent variable x and $\alpha = 0$ for the initial point selection. In this case, the points x_j^{in} are placed so that they split the (expected) y-range of the $N_i - 1$ intervals evenly.

Figure 2 depicts the initial point selection phase for our sample system. The plot shows the actual performance function of the system, i.e., the curve named "experiment results", the educated guess, and the initially selected points. As mentioned earlier, the system under study is a multicore web server. The educated guess is obtained from a layered queuing model [14] of the system, with model parameters determined using estimated demands of hardware, software, and network resources. Due to inaccuracy of the demand estimation process, the final model has more than 100% errors in some regions. However, the model is still useful for determining the initial points. For this example, we start with three initial points ($N_i = 3$), labeled in_1, in_2, and in_3 in the figure.

Let's look closer at the initial point selection. First, based on the educated guess (EG), the value of response variable, $R_{EG}(x)$, varies for a range of ΔR_{EG}. We divide this y-range into two equal intervals by adding a single point in the middle of the y-interval at $\frac{\Delta R_{EG}}{2}$. Given the expected points x_1^{in} and x_3^{in} with the minimum and maximum load of interest, the second initial point is selected as $x_2^{in} = R_{EG}^{-1}(R_{EG}(x_1^{in}) + \frac{\Delta R_{EG}}{2})$, where $R_{EG}(x_1^{in})$ is the expected response time for the low load case and $R_{EG}^{-1}()$ is the inverse function of $R_{EG}()$. Finally, experiments are conducted to determine the actual values of the response variables, y_j^{in}, as shown by the red crosses in Figure 2.

The best number of initial points depends on the shape of the educated guess and the properties of the system under study. In general, since the initial point selection is performed with $\alpha = 0$, the systems that benefit from a smaller value of α can benefit the most from the initial point selection phase. In Section 5.3, we present an example of how the initial point selection phase can influence the overall error reduction achieved through IRIS.

(*ii*) **Iterative point selection:** There are two input parameters for this phase of the algorithm: (i) a sample point budget (N_t), which is the maximum number of experiments we can run, and (ii) the factor α that weights the importance of independent and response variables in selecting the next point. In this section, we assume that both parameters are known prior to the start of the iterative phase.

Having selected an initial point set, an ordered list of already measured (x_j, y_j) points are available, where initially $1 \leq j \leq N_i$. With this as the base case, the algorithm then calculates the gain factor G_j, using equation (1), for all intervals. The centroid of the interval corresponding to the maximum G_j is selected as the next point to be examined. After the first iteration, the list of available points grows to $N_i + 1$ points. The iterative phase continues until the number of available points reaches the N_t limit.

We next demonstrate the point selection process for our example system considering different values of α. Figures 3(a) to 3(c) show the actual performance functions reflecting the experiment data as well as the selected points for $\alpha = 0$, $\alpha = 0.5$, and $\alpha = 1$. For ease of comparison,

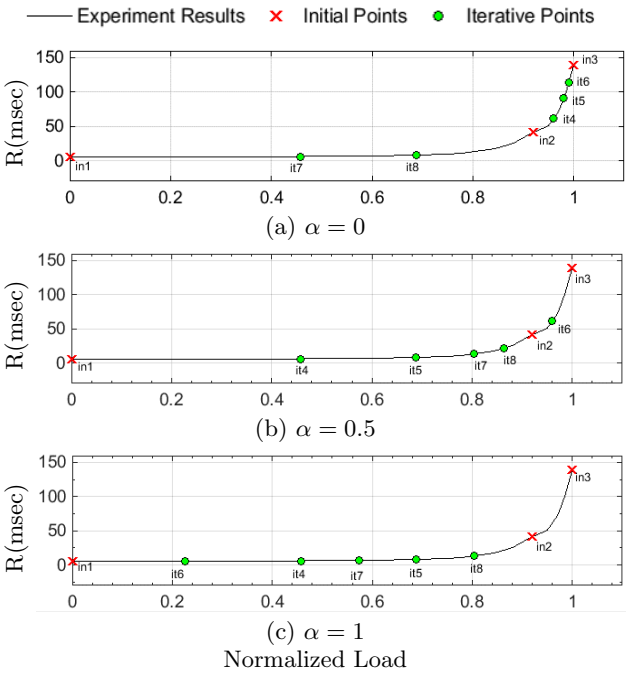

Figure 3: Iterative point selection phase.

in all cases, we have used the same $N_i = 3$ initial points shown with red crosses and a total sample point budget of $N_t = 8$ shown as green circles. In Figure 3(a), where $\alpha = 0$, the first iteratively selected point, labeled as $it4$ to indicate that it is added during the iterative phase and is the 4^{th} selected point, is selected in the right-most interval since it has the maximum value of R_j of the different candidate intervals, and the interval with maximum R_j is always prioritized when $\alpha = 0$. After this point is added to the set, the algorithm examines three intervals in the second iteration. Again, the next point, $it5$, is selected as the centroid of the right-most interval. The iterative phase continues until the number of selected points reaches N_t.

Considering the changes in the position of selected points from Figure 3(a) for $\alpha = 0$ to 3(c) for $\alpha = 1$, we note that as the α values increases, the selected points are more spread along the x-axis. This illustrates how the α parameter can be used as a tuning knob to put more or less emphasis on the regions with higher changes on the response variables. For example, with $\alpha = 1$ the interval with the largest x-range difference (A_j) is always split, and with $\alpha = 0$ the interval with the largest y-range difference (R_j) is always split.

3. EVALUATION METHODOLOGY

For the purpose of evaluation, we have applied IRIS to several systems with different characteristics. This section provides details of the general evaluation process. In particular, we outline a common baseline protocol against which we compare our technique, describe the input parameters, and explain evaluation metrics and the techniques we used to estimate the performance function.

3.1 Equal Distance Point Selection

A common way of selecting the values of independent variables to perform benchmarking studies is through *equal distance* (EQD) point selection. With this approach, the possible range of each independent variable is divided into $N-1$ equally sized intervals. Assuming D dimensions, each dimension d ($1 \leq d \leq D$) divided into $N_d - 1$ intervals, this approach results in $\prod_{j=1}^{D} N_j$ evenly spread experiment points.

A major drawback of EQD is the cost of adding additional points to achieve a desirable accuracy. As a simple example, consider a system with a single independent variable, where the initial estimate of the required number of points is five. Now assume that after running the experiments for five equally distanced points and applying a function estimation technique, the accuracy of the estimated function is not desirable. If there is additional budget available for adding more experiment points, and one wants to spend the additional points through the EQD approach, there are two ways to proceed. First, one can replace the three middle points with four new points that are equally spaced. Second, one can add a single point in the middle of each interval. In both cases, four new experiment points are required to increase the accuracy of the predicted function while still maintaining the equal distance property. Therefore, one requires nine points for identifying the performance function at this stage. These examples describe cases where the available point budget is spent in multiple rounds of EQD. We refer to this approach as *multi-stage* EQD.

We also consider a penalty-free version in which any previously performed measurements not used, as in the first of the above approaches, are ignored. This corresponds to the optimistic case when we always guess the correct number of sample points to use from the start and place all the point budget in a single round of EQD. We refer to this optimistic approach as *single-stage* EQD.

In our case studies, we compare both *multi-stage* EQD and *single-stage* EQD with IRIS, emphasizing that *single-stage* EQD is optimistic in most cases. For the systems with a single variable, the number of equally distanced points is set to N_t. For the cases with more than one independent variable, we consider all combinations of number of intervals in different directions that lead to a total of N_t points.

3.2 IRIS Parameters

As described in Section 2.4, there are four main input parameters for IRIS: (i) an educated guess of the expected performance function, (ii) the number of initial points N_i, (iii) the weight tuning gain parameter α, and (iv) the total sample point budget N_t. In the case studies presented in Section 4, we keep the first three inputs constant to facilitate a direct comparison. For the evaluation, we typically assume minimum knowledge about the system and try to minimize the effect of the initial point selection. In particular, we assume that the educated guess is simply a linear relationship between the independent and response variables and use $N_i = 3^D$, where D is the number of dimensions. Hence, for the single dimensional case we have $N_i = 3$. Section 4 considers the performance using default settings, before Section 5 evaluates the effect of tuning these parameters on the final results. For each case study, we explore a range of N_t values. The minimum of this range depends on the dimensionality of the system. The maximum is determined as the point budget beyond which there is no significant improvement in the prediction errors from the performance functions estimated by all the techniques.

For each case study, five values of α ($0, 0.25, 0.5, 0.75$, and $\alpha = 1-\epsilon$) are examined for a wide range of N_t values. Recall that $\alpha = 0$ leads the algorithm to only consider the change in the response variable R_j. Conversely, $\alpha = 1$ considers the other extreme and only considers the area A_j of each region in the gain calculations. Selecting $\alpha = 1 - \epsilon$ ensures that in case IRIS encounters equal areas, it selects the region with maximum variability in the response variable R_j.[2]

3.3 Evaluation Metrics

All methods are compared based on their relative accuracy for fixed numbers of point budgets. To calculate the accuracy metric we use the following three steps for each dataset and algorithm considered.

1. Run the algorithm: The output of this step is the set of selected points. Each point is a set of values for the independent variables and the corresponding response variable obtained from the experiment dataset.

2. Estimate the performance function: The set of points selected in the previous step is fed to a function estimation technique to estimate the performance function. The prediction for each point x_j obtained from this function is referred to as $R_{PRD}(x_j)$. The function estimation techniques we consider are explained in Section 3.4.

3. Calculate the Average Absolute Error (AAE): At this step, the values predicted by the performance function are compared with their corresponding experimentally measured values for all the available points in the experiment dataset. The AAE is calculated as follows:

$$AAE = \frac{\sum_{j=1}^{n} |R_{PRD}(x_j) - R(x_j)|}{\sum_{j=1}^{n} R(x_j)} \cdot 100, \qquad (2)$$

where n is the number of input points in the experiment dataset. For our evaluation, we consider constrained experiment budgets; i.e., $N_t << n$.

For head-to-head comparisons, we repeat the same process with the two EQD point selection approaches in step 1. For each dataset, the process is repeated for a range of point budgets (N_t). Finally, to quantify the effectiveness of IRIS compared to the EQD baselines, we define the *Error Reduction* (ER) ratio as:

$$ER = \frac{(\overline{AAE}_{baseline} - \overline{AAE}_{IRIS})}{\overline{AAE}_{IRIS}}, \qquad (3)$$

where the $\overline{AAE}$ is the average of all AAE values measured for setups with different point budgets (N_t). To calculate $\overline{AAE}$, we only consider N_t's that are common between the IRIS and baseline methods.[3] Note that the baseline method can be any of the two EQD approaches.

3.4 Function Estimation Techniques

In this study, we used two function estimation techniques, namely, the commonly used *Cubic Spline* interpolation technique for one-dimensional systems and *Kriging* for multi-dimensional systems. The Cubic Splines method can also be applied for multi-dimensional systems. However, it requires a grid of points in all dimensions to perform the interpolation. Recall that IRIS is based on scattered and iterative point selection. Therefore, we select the *Kriging* [4] technique as it is better equipped to handle scattered point selection in multi-dimensional scenarios. Previous studies [17] have also confirmed the applicability of Kriging for estimating performance functions. Kriging has several variants such as simple Kriging, ordinary Kriging and universal Kriging. We select the ordinary Kriging approach as it requires no initial information about the trend of the estimated performance function. We use the UQLab [15] implementation of Kriging for MATLAB.

[2]We use $\epsilon = 0.001$ for all the datasets.

[3]Due to the stepwise nature of *multi-stage* EQD approach, only a subset of point budget values are possible with this approach.

4. RESULTS

To evaluate the effectiveness of IRIS, we perform several case studies based on datasets collected through experiments, simulations, or synthetic performance functions. We evaluate the approach for a total of eight datasets based on four systems with a single independent variable, three systems with two independent variables and one system with three independent variables.

4.1 Single Independent Variable

We consider four systems with vastly different shapes of the underlying performance functions. Each dataset consists of a family of performance functions with similar high-level characteristics but different properties, e.g., different sizes of the flat regions. This allows us to determine the relationship between the tuning parameters, in particular α, and the properties of the performance function, as we will elaborate later in Section 5.1. The plots in Figure 4 show the normalized response variable (y) as a function of the normalized independent variable (x) for all four families of performance functions considered. Note that both the independent and response variables are normalized by rescaling to the range of $[0, 1]$.

The performance functions of the first dataset, shown in Figure 4(a), are the results of a simulation study of a closed system with two resources and two user classes. The independent variable (x-axis) is the service time for resource one while the response variable (y-axis) is the response time of the class that poses the largest service demand on that resource. Each curve $C1$, $C2$ and $C3$ in Figure 4(a) are the results of simulations for various combinations of populations for the two classes. Further discussions about the analysis of the system's behavior can be found in Chapter 7 of the textbook by Lazowska *et al.* [9]. These *s-shaped* curves represent cases where the response variable is initially constant then has a sharp increment before entering a saturation mode.

The second dataset, shown in Figure 4(b), is collected from the experimental system described in Section 2.4. Here, a large number of concurrent web requests are served using a multicore server, and the primary performance trade-off of interest is the impact that the normalized load, i.e., normalized number of concurrent users per core, has on the mean response time. We refer to this system as the *load-response time* dataset. The curves, $C1$, $C2$ and $C3$ show the actual experimental results for this system with two, four and eight processor cores, respectively. The three curves differ in terms of the extent of the flat region as well as the rate at which the response time increases with load. The main characteristics of this system is that the response variable

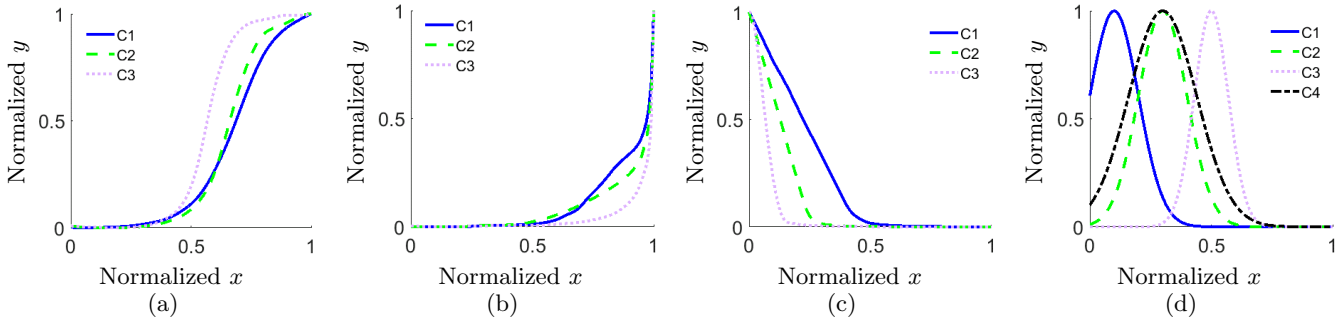

Figure 4: 1D systems: (a) S-shaped, (b) Load-response time, (c) Hockey stick, and (d) Bell-shaped.

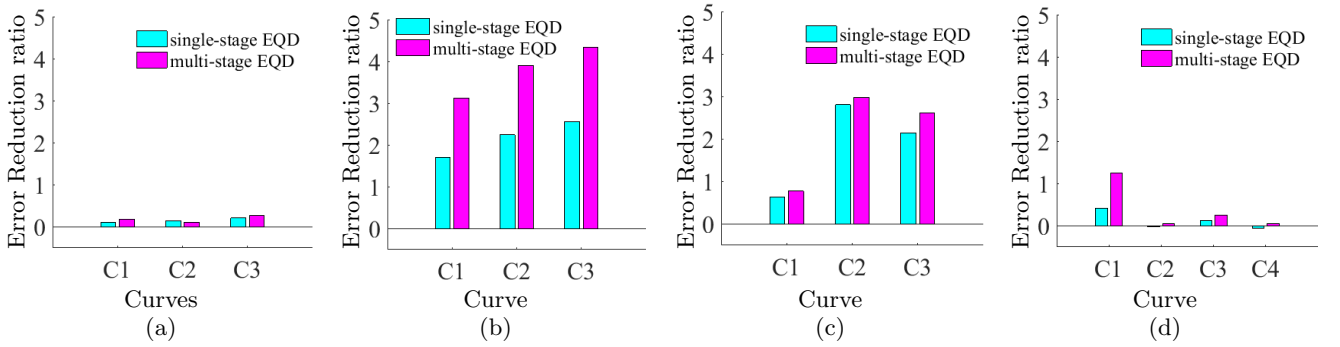

Figure 5: ER ratios for 1D systems: (a) S-shaped, (b) Load-response time, (c) Hockey stick, and (d) Bell-shaped.

increases sharply for a small range of the independent variable.

Third, we consider a group of performance functions generated by simulating a closed queuing model with three resources. The goal of this simulation is to identify the impact of user think time on the average response time. The results of this study are depicted in Figure 4(c). We refer to this family of curves as *hockey stick* as they have a section with constant slope as well as a flat region. The combinations of the resource demand values used in the simulation results in three curves that are different in terms of the extent of the flat region. Similar to the load-response time system, this dataset contains a region where the response variable does not change significantly with the independent variable. However, in contrast to the load-response time system, the increase in the response variable is only gradual in the hockey stick dataset.

The last dataset, shown in Figure 4(d), includes a group of *bell-shaped* synthetic functions representing normal distributions. We change the mean and standard deviation to vary the position of the maximum response and the width of the bell in the four curves. Compared to the previous datasets, the bell shapes display both an increase and decrease in the response variable over the range of the independent variable.

Figures 5(a) to 5(d) show the Error Reduction (ER) ratio for IRIS compared to the two EQD baselines. We again note that the *single-stage* EQD is highly optimistic in most cases, is not always feasible, but provides the most competitive baseline comparison in one dimensional cases. A negative ER value implies that IRIS causes a larger AAE compared to the baseline method. The ER ratios in Figure 5 are calculated for $\alpha = 0.5$. We will discuss the effect of other values of α in Section 5.1.

Figure 5 reveals that IRIS leads to a positive ER ratio for a vast majority of cases. Figure 5(a) for the s-shaped dataset suggests that our approach led to a maximum of $ER = 0.3$, which is equivalent to a 30% error reduction. The maximum ER value is measured for the curve with the steepest slope and the largest flat region. IRIS yields dramatic gains for the load-response time dataset. As shown in Figure 5(b), IRIS leads to an ER of 4.3 and 2.3 compared to *multi-stage* and *single-stage EQDs*, respectively. The results also reveal that the ER ratio increases as we move from $C1$ with the smallest flat region to $C3$ with the largest flat region. A similar trend is observable for the hockey stick dataset, as shown in Figure 5(c). For this system, the ER ratios are slightly less than the load-response time dataset. Finally, the results for the bell-shaped curves, as depicted in Figure 5(c), reveal a few cases where IRIS with $\alpha = 0.5$ sees slightly worse performance than *single-stage* EQD, i.e., negative ER values. However, in general, IRIS improves the accuracy for the bell-shaped curves as well. In particular, for C1 it reduces errors by as much 130%. It should also be noted that IRIS allows incremental point selection, allowing the technique to be used iteratively until meeting a desired accuracy or stopping criteria. In contrast, single-stage EQD is optimistic in that it always "guesses" the correct number of sample points.

Overall, these results show that IRIS can result in up to 2.3 times improvement in terms of AAE, compared to the

single-stage EQD. The improvements are even higher relative to *multi-stage* EQD. Moreover, the maximum achievable improvement depends on the shape of the system's actual performance function. Selecting the value of α based on the expected system behaviour can further increase the ER ratio, as we will discuss in Section 5.1.

4.2 Multiple Independent Variables

In this section we consider datasets with multiple independent variables. The first three datasets are systems with two independent variables while the last one is a simulation system with three independent variables. Figure 6 shows the systems with two independent variables. Similar to the single variable systems, each dataset consists of surfaces with similar high-level characteristics that differ in specific properties such as the area of the flat surface or the slope.

The system functions of the first dataset, shown in Figure 6(a), are the results of a simulation study of a closed system with two resources and two user classes [9]. The first and second independent variables (x_1 and x_2) are the service times for resource one and two, respectively. The response variable (y) is the response time of the class that poses the large service demand on resource two. Each surface represents the results of the simulation for a combination of populations for the two classes. The focus of our analysis is on the s-shaped surfaces of the response variable that is almost constant with respect to x_1 and is initially constant then has a sharp increment and then enters a saturation mode with respect to x_2.

The second dataset, depicted in Figure 6(b), is a load-response time dataset with two load parameters as independent variables. In this case the system functions are collected from simulating an open queuing model with an m server resource and two user classes. The request arrival rate for each user class is considered as an independent variable and the average response time over all classes of users as the response variable. The surfaces $S1$ to $S3$ are the results of running the simulation for $m = 1$, $m = 2$ and $m = 4$.

The last dataset, shown in Figure 6(c), represents a group of three synthetic Gaussian surfaces with different means and standard deviations. Similar to the single variable bell-shaped system, this dataset is selected to examine surfaces with a maximum point in the middle of the range of independent variables. $S1$ and $S2$ have similar maximum points but differ in terms of the radius of the bell. $S3$ has a similar radius with $S2$ but a different position for the maximum point.

Figures 7(a) to 7(c) show the ER ratios calculated for the systems with multiple independent variables. Note that IRIS provides a positive ER ratio when applied to all three systems. The maximum ER ratio was measured for the s-shaped dataset, as shown in Figure 7(a). From Figure 7(b), the ER ratios for the load-response time dataset are between 0.5-0.7 (50%-70%) and 0.3-0.4 (30%-40%) relative to *multi-stage* EQD and *single-stage* EQD, respectively. While the ratios are all positive, they are lower compared to the s-shaped dataset. This can be explained by the large size of the flat region in this dataset. In Section 5.4, we will discuss a better criteria to evaluate the effect of applying IRIS for datasets with large flat regions. Finally, applying IRIS results in considerable error reductions for the bell-shaped surfaces, as depicted in Figure 7(c). Comparing the results for $S1$ and $S2$, the ER ratio is larger for the wider surface $S1$. It is also larger for the symmetric curves $S1$ and $S2$ compared to $S3$. We explore the relationship between properties of the bell-shaped surfaces and the effectiveness of IRIS in more detail in Section 5.1.

Next, we apply IRIS to a dataset with 3 independent variables. This dataset is obtained from the results of a simulation study of a closed system with two resources and two user classes presented in Chapter 7 of the textbook by Lazowska *et al.* [9]. The example illustrates a case where the combination of service demand values for the two classes on the two resources results in the response time of class one to increase as the service time for resource two decreases. We used this remarkable behaviour to create a surface with a hump-shaped maximum point in the middle of the parameter plane. The service time of resource two and the ratio of class populations for the two classes are selected as the first and second independent variables. We extend the parameter space to the third dimension by running the simulation for a range of user populations. The final system is different from the previous systems in that it simultaneously displays bell-shaped, s-shaped and the load-response behaviours.

Table 1 shows the ER ratios for various values of α when compared with *multi-stage* and *single-stage* EQD baselines. The results show that except for the $\alpha = 0$ case and relative to the highly infeasible *single-stage* EQD case, all the ER ratios are positive and vary between $0.21 < ER < 2.79$. The largest error reduction was achieved with $\alpha = 1 - \epsilon$.

Table 1: ER ratios for 3D system.

α	*multi-stage* EQD	*single-stage* EQD
0	0.21	-0.28
0.25	1.75	0.63
0.5	1.99	0.77
0.75	2.76	1.22
$1 - \epsilon$	2.79	1.24

5. DISCUSSION

This section provides additional experiments to illustrate various properties of IRIS. We first investigate the best choice of the tuning parameter α based on the expected shape of the system's actual performance function. Second, we isolate the effect of the iterative point selection to further understand this component's contribution to the improvements seen with IRIS. Third, we show how a partially accurate educated guess about the system under study can improve the ER ratio. Finally, we perform an in-depth analysis of the error distribution to identify the regions of the independent variable space that benefit the most from applying IRIS.

5.1 Tuning the Gain Parameter

The results presented in Section 4 show that IRIS is able to reduce the AAE for a vast majority of system functions with $\alpha = 0.5$. We now investigate how the initial knowledge about shape of the system function can be used to select an even better α. To quantify the effect of α on the AAE, we apply IRIS to the one-dimensional and two-dimensional datasets for a range of gain parameter α values, $0 < \alpha < 1$.

In general, we find that α should be small when the underlying function has a convex knee, e.g., the load-response time

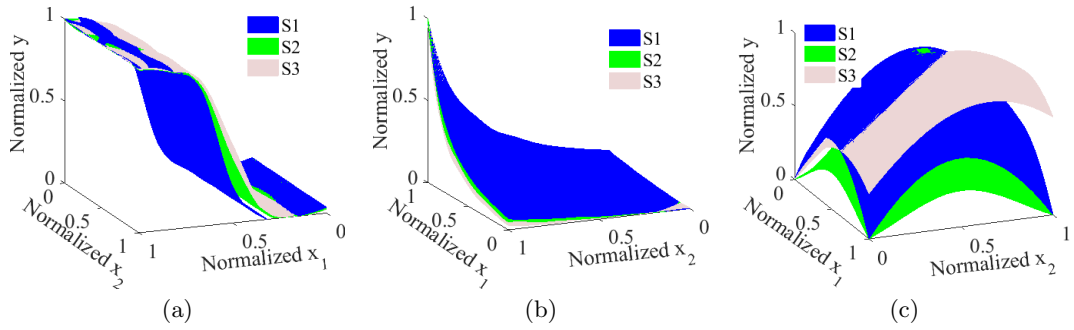

Figure 6: 2D systems: (a) S-shaped, (b) Load-response time, and (c) Bell-shaped.

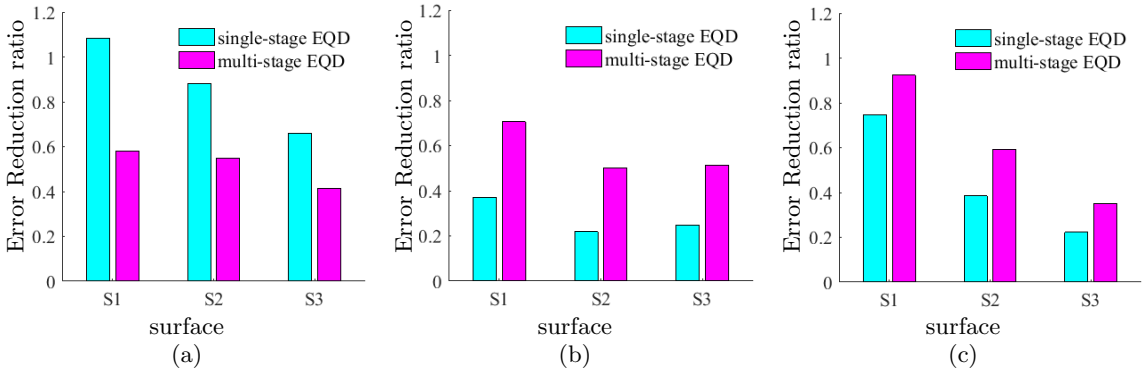

Figure 7: ER ratios for 2D systems: (a) S-shaped, (b) Load-response time, and (c) Bell-shaped.

or hockey stick systems, and should be larger in the rarer cases when the system has a symmetric maximum point, e.g., the bell-shaped systems. The results for one example of each type are illustrated in Figures 8 and Figures 9 for the one-dimensional and two-dimensional systems, respectively.

Figure 8(a) shows the AAE as a function of the α parameter for the load response time dataset. Here, larger values of α, say above 0.6, result in a larger AAE. This behaviour is expected, considering the shape of the three curves in this dataset, and confirms our intuition that greater weight should be given to the region with large changes in the response time (i.e., the R_j term in equation 1). Using a large α value for this dataset is sub-optimal since it places too many points in the flat low-load region. Instead, using smaller α values appears desirable as this allows more points to be placed closer to the region of most interest.

In contrast, from Figure 8(b), for the bell-shaped dataset the optimal α depends on the position of the maximum point as well as the size of the flat region. More specifically, for curves $C2$ and $C4$ with a more symmetric shape and smaller flat regions, the higher values of α result in lower errors. In contrast, for $C1$ and $C3$ with the larger flat region, setting $\alpha < 0.8$ can minimize the AAE. The difference between the two systems shows the value of careful selection of the α parameter. For systems where the existing system knowledge suggests that there is a convex sharp knee we may want to pick a smaller α, and in the rarer cases in which the system knowledge suggests a concave and symmetric maximum point we may want to select a larger α.

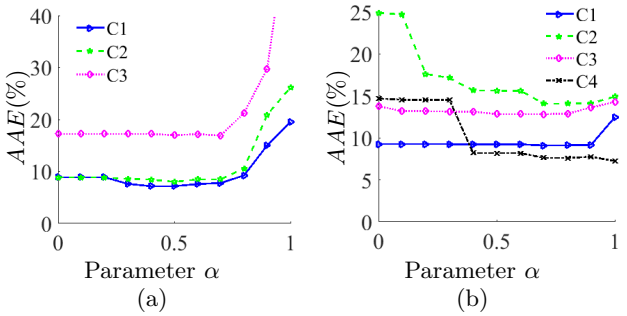

Figure 8: Effect of α on AAE for 1D systems: (a) Load-response time, and (b) Bell-shaped.

The two-dimensional results shown in Figures 9(a) and 9(b) confirm that these observations extend to multiple dimensions. For the load response time dataset, the surfaces have a large flat region that can suffer from setting α to lower values. Therefore, as shown in Figure 9(a), an intermediate α (say $\alpha = 0.5$) is best in this case. For the three bell-shaped surfaces without flat regions, the optimal setting is $\alpha = 1$.

5.2 Effect of Iterative Point Selection

As mentioned in Section 2, IRIS aims to improve prediction accuracy by applying two main modifications compared to EQD point selection: (i) selecting the points iteratively, and (ii) selecting the position of the next point based on

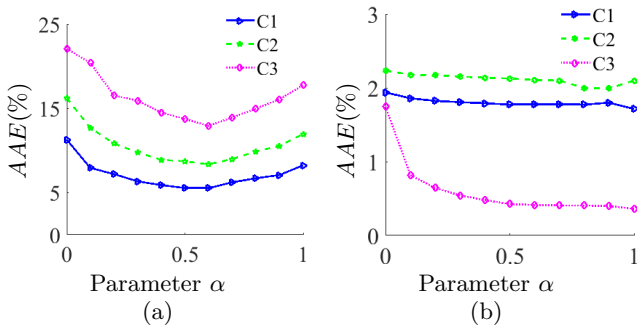

Figure 9: Effect of α on AAE for 2D systems: (a) Load-response time, and (b) Bell-shaped.

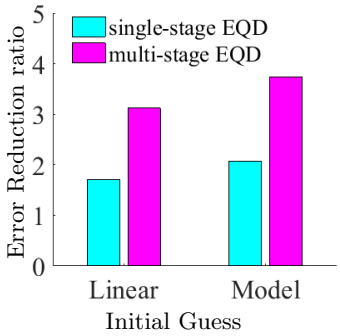

Figure 11: Linear guess vs. model based guess.

the observed changes in the response variable. To isolate the effect of the first modification alone, we evaluate IRIS with $\alpha = 1 - \epsilon$, in which case points are spread as evenly as possible ignoring the changes in the response variable. Due to space constraints, we show results only for the one-dimensional load-response time systems but similar results are observed for the other systems as well. We calculate the average AAE across all of the curves in this dataset. Figure 10 shows the AAE as a function of the total sample point budget N_t. With $\alpha = 1 - \epsilon$, IRIS would always place a new point at the midpoint of the largest current interval. It is therefore no surprise that the IRIS result matches the *single-stage* EQD results for the cases when $N_t = 5$, 9,and 17. In all these cases the distances between the points selected by IRIS should be equal. For the other N_t cases, IRIS leads to significantly lower AAE compared to both *multi-stage* and *single-stage* EQD. This emphasizes the importance of iterative point selection.

The results also confirm that IRIS can achieve a desired level of accuracy with a smaller point budget compared to both EQD baselines. For instance, considering a desired accuracy level of 40%, IRIS needs to spend $N_t = 6$ points, while the *single-stage* and *multi-stage* EQD need $N_t = 9$ and $N_t = 19$ points, respectively.

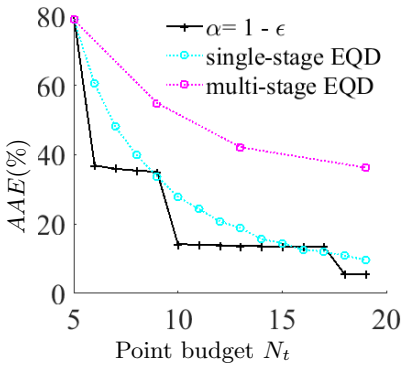

Figure 10: Effect of iterative point selection.

5.3 Effect of Initial Point Selection

In the results presented in Section 4, we tried to minimize the effect of the initial point selection by assuming that there is limited knowledge about the system under study. Specifically, we assume linear relationships between response and independent variables. In this section we show the effect of using a more educated guess about the system in the initial phase of IRIS.

The demonstration in Section 2.4 suggests that a careful selection of the initial points can be beneficial for system functions that generally benefit from smaller α values. Consequently, we consider the single dimension load-response dataset to explore the impact of careful initial point selection. As mentioned in Section 2.4, we use a layered queueing model for this dataset to obtain a realistic, educated guess of the system's performance function.

Figure 11 shows the ER ratios for the linear guess, i.e., the results presented in Section 4.1, and our educated guess based on the model. For example, considering *single-stage* EQD, selecting the initial points based on the model's educated guess can increase the ER ratio by around 17% over a linear initial guess. Similar results are obtained for other curves in this dataset as well as all surfaces of the load-response time dataset with two independent variables.

5.4 Error Distributions

In Section 4, we compared IRIS against the two baseline algorithms considering the differences between their "average" prediction errors (AAE). The results reveal that while applying IRIS can lead to a significant error reduction in most cases, the ER ratio improvements are more moderate for a subset of datasets. For instance, applying IRIS to the load-response time dataset with two dimensions results in relatively modest ER ratios between 0.3 to 0.7, depending on the surface. This may seem counter-intuitive given the sharp and convex knee point in the surfaces of this dataset. Our hypothesis is that for this case the AAE metric is biased towards the prediction errors for the large flat regions of the surfaces since the metric gives the same weight to all points in the two dimensional plane. Therfore, it does not capture the fact that the knee point may be of greater interest to practitioners, where IRIS can significantly outperform the EQD techniques. We confirm this hypothesis by analyzing the distribution of prediction errors throughout the two-dimensional space.

Figures 12(a), 12(b), and 12(c), show the colour map of error values when predicting surface $S3$ of the load-response time dataset for IRIS with $\alpha = 0.5$, *single-stage* EQD, and *multi-stage* EQD, respectively. The contour lines in the graphs show the values of the response variable in that surface. The figures show that with IRIS the error values for a large area of the independent variable space, corresponding

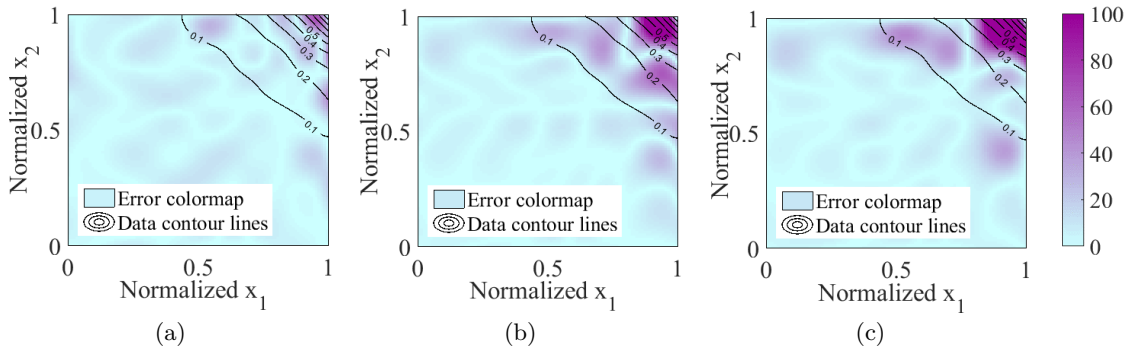

Figure 12: Color map of errors: (a) IRIS with $\alpha = 0.5$, (b) *single-stage* EQD and (c) *multi-stage* EQD.

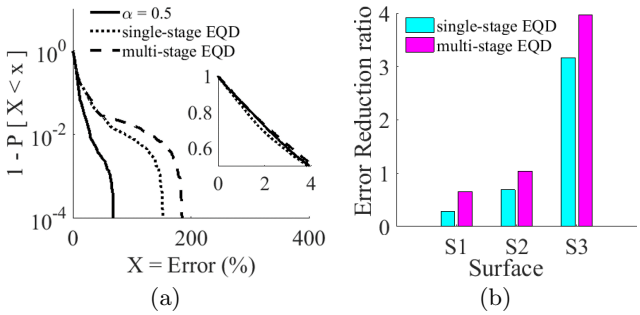

Figure 13: (a) ccdf of errors and (b) ER ratios for $0.1 < R < 0.4$.

to $0 < R < 0.1$, are somewhat higher compared to the *single-stage* EQD as indicated by the larger dark areas. However, the error values for a smaller area of the space, corresponding to $0.1 < R < 1$, are significantly lower compared to the *single-stage* EQD. A similar trend is observed for *multi-stage* EQD. Recall that for $S3$ IRIS achieves ER ratios of 0.4 and 0.2, over the *multi-stage* EQD and *single-stage* EQD, respectively. This implies that despite the slightly higher error values in the large flat area IRIS still improves the average errors by up to 40%.

These observations are further quantified by illustrating the complementary cumulative distribution function (ccdf) of errors as depicted in Figure 13(a). The main plot in this figure indicates that the baselines have significantly heavier tails than IRIS. Considering $1-P[X < x] < 10^{-1}$, errors are three to four times higher for the two baselines. In contrast, a closer look at the small, zoomed plot reveals that up to the 50th percentile of error, i.e., smaller error values in the flat region, the error values are slightly higher for IRIS compared to the two baselines.

These results reveal that IRIS is more successful in reducing prediction errors for the area of the knee point where the response time increases sharply for a small change in the load. In most scenarios practitioners are interested in the behaviour of the underlying system for this particular range of the response variable. We now calculate the effectiveness of IRIS when predicting the system function in this region of interest. Figure 13(b) shows the ER ratios for the three surfaces while only considering $0.1 < R < 0.4$ as the desirable range of response times. For this region, the ER ratio for IRIS is up to 6 times higher comparing the ER ratios in Figures 13(b) and 7(b) for the full region. This again demonstrate the high accuracy of IRIS in the regions often of most interest.

6. RELATED WORK

Experiment design techniques focus on identifying the independent variables that most influence the response variable [1, 12]. Typically, an initial set of experiments is conducted and used to estimate a regression model that relates the response variable to individual independent variables, as well as to interactions between the independent variables. Insights from the model are then used to eliminate independent variables that do not significantly influence the response variable from subsequent experiments. IRIS complements experiment design techniques. In particular, given the identified independent variables of interest, IRIS can help with the placement of experiment points.

Response Surface Methodology (RSM) [11, 10] is a technique to identify the independent variable settings at which the response variable attains the optimum, i.e., minimum or maximum, value. An experiment design phase is first carried out to establish an initial model of the response variable as a function of the independent variables. This model is then used to approximately determine the region that contains the optimal response. Next, more experiments are conducted in this region, aiming to refine the initial model. Finally, the refined model is used to estimate the independent variable settings that cause the optimal response. Jamshidi *et al.* [8] also propose a technique similar to RSM to identify optimum configuration settings for applications. In contrast to IRIS, RSM-based techniques are focus on a particular region and are not designed for identifying a performance function that spans the entire independent variable space.

Similar to IRIS, Reussner *et al.* [13] propose an experimentation technique that seeks to continually improve performance prediction accuracy for a suite of Message Passing Interface (MPI) benchmarks. Their technique iteratively estimates regions where the performance function has the highest errors and conducts additional experiments in those regions to reduce these errors. In contrast to IRIS, this technique focuses only on MPI benchmarking. Furthermore, unlike IRIS, it can only consider one independent variable at a time when refining the performance function.

Courtois and Woodside [3] and Westermann *et al.* [16] propose similar iterative experiment selection techniques for building a regression model based performance function. While these techniques can consider more than one inde-

pendent variable at a time, they require multiple experiment points to be placed in each iteration so as to quantify the errors of the model in various regions. In contrast, IRIS carries out experiments more sparingly by adding a single experiment point in each iteration. As we show in Section 5.2, this can facilitate using fewer experiments to achieve a given degree of accuracy. Furthermore, IRIS does not rely on a model but instead uses actual measurements to drive its point selection. Consequently, it eliminates the effect of modelling errors on the point selection.

7. CONCLUSIONS

This paper presents an iterative and intelligent experimentation technique called IRIS. The main objective of IRIS is to obtain the best possible insights on a system's performance for a given experimentation budget. Given an initial experimentation plan based on an educated guess, IRIS iteratively and intelligently adds further experiments in regions of the independent variable space where the independent variables have the most effect on the response variable. IRIS exposes a single tunable gain parameter α that allows an experimenter to trade off the desire to evenly cover the independent variable space and placing particular focus on regions with larger changes in the response variable.

Extensive experiments involving systems with single and multiple dimensions show that IRIS significantly outperforms baseline techniques that rely on equal distance point selection for a vast majority of cases. Moreover, there was no significant degradation in accuracy for other cases. For a given experimentation budget, performance functions estimated based on experiments suggested by IRIS can yield average prediction errors for the response variable that are lower by up to a factor of 4.3. Furthermore, due to the iterative nature of IRIS, fewer experiments are needed to achieve a desired level of response variable prediction accuracy. IRIS parameterized with lower values of α, e.g., $\alpha = 0.5$, are particularly effective in improving system performance predictions in regions that practitioners typically find interesting, e.g., the "knee" regions where response times can change considerably. The gains from IRIS when considering only such regions is up to 6 times more than that when considering all regions.

A sensitivity analysis of α settings shows that in general α should be small when the underlying system performance function has a convex knee and should be larger in the rarer cases when the system has a symmetric maximum point. Our analysis also shows that gains from IRIS can increase even more if the initial educated guess leverages domain knowledge or approximate performance models of the system under study. We note that IRIS takes negligible time to make its point selection decisions.

Future work will focus on applying IRIS to systems with higher dimensionality. We will also focus on leveraging our insights regarding the relationship between α and the system performance function to dynamically optimize the value of α during experimentation. Finally, we will also explore further the impact of the initial guess on prediction accuracy.

8. REFERENCES

[1] G. E. Box and D. W. Behnken. Some new three level designs for the study of quantitative variables. *Technometrics*, 2(4):455–475, 1960.

[2] S.-W. Cheng, T. K. Dey, and J. Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.

[3] M. Courtois and M. Woodside. Using regression splines for software performance analysis. In *Proc. of WOSP '00*, pages 105–114. ACM, 2000.

[4] N. Cressie. The origins of kriging. *Mathematical geology*, 22(3):239–252, 1990.

[5] J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

[6] R. Hashemian, D. Krishnamurthy, and M. Arlitt. Overcoming web server benchmarking challenges in the multi-core era. In *Proc. of ICST '12*, pages 648–653. IEEE, 2012.

[7] R. Hashemian, D. Krishnamurthy, M. Arlitt, and N. Carlsson. Improving the scalability of a multi-core web server. In *Proc. of ICPE '13*, pages 161–172. ACM, 2013.

[8] P. Jamshidi and G. Casale. An uncertainty-aware approach to optimal configuration of stream processing systems. *arXiv preprint arXiv:1606.06543*, 2016.

[9] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., 1984.

[10] K. Molka and G. Casale. Experiments or simulation? a characterization of evaluation methods for in-memory databases. In *Proc. of CNSM '15*, pages 201–209. IEEE, 2015.

[11] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization using Designed Experiments*, volume 705. John Wiley & Sons, 2009.

[12] R. L. Plackett and J. P. Burman. The design of optimum multifactorial experiments. *Biometrika*, pages 305–325, 1946.

[13] R. Reussner, P. Sanders, L. Prechelt, and M. Müller. SKaMPI: A detailed, accurate MPI benchmark. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 52–59. Springer, 1998.

[14] J. A. Rolia and K. C. Sevcik. The method of layers. *IEEE Transactions on Software Engineering*, 21(8):689–700, 1995.

[15] Uqlab: The framework for uncertainty quantification. http://www.uqlab.com/.

[16] D. Westermann, J. Happe, R. Krebs, and R. Farahbod. Automated inference of goal-oriented performance prediction functions. In *Proc. of ASE '12*, pages 190–199. ACM/IEEE, 2012.

[17] D. Westermann, R. Krebs, and J. Happe. Efficient experiment selection in automated software performance evaluations. In *Computer Performance Engineering*, pages 325–339. Springer, 2011.