An Iterative Two-stage Multi-fidelity Optimization Algorithm for Solving Computationally Expensive Problems

Angus Kenny, Tapabrata Ray, and Hemant Kumar Singh

***Abstract*—The abstract goes here.[Hemant: Angus, please take a shot at the first version of abstract.]**

[Angus: Angus can make comments using \angus{comment}]

[Ray: Ray can make comments using \ray{comment}]

[Hemant: Hemant can make comments using \hemant{comment}]

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

MANY real-world engineering design problems require estimation of responses that are intractable for exact or analytical methods. In such cases, two alternatives are commonly resorted to: numerical simulations or physical testing of a prototype [1]. When used in-loop during design optimization using iterative methods such as evolutionary algorithms (EAs), both of these methods tend to be prohibitively slow; as well as potentially cost and resource intensive [2]. Such optimization problems where each design evaluation incurs significant cost in any form are referred to as expensive optimization problems.

Simulation-based optimization (SO) refers to the methods that deal with optimization problems involving numerical simulations[1]. Typically, these methods make use of surrogates/approximations, also referred to as metamodels, to reduce the computational expense [3]. The basic principle is to use historical information from previously evaluated designs to build a surrogate model of the response landscape. The predicted values from these surrogate models can be then utilized to determine new candidate solutions that are likely to be competent when evaluated using the true (time consuming) simulation. Through this informed pre-selection, the number of expensive evaluations can be significantly reduced during the course of optimization.

Some numerical simulation processes allow control over the resolution, or *fidelity*, of the estimated responses. For example, in finite element analysis (FEA) or computational fluid dynamic (CFD) simulations, the mesh size can be controlled to yield solutions with different fidelities [4], [5]. A coarse mesh yields a low-fidelity (LF) performance estimate that is relatively fast to compute but less accurate, while a fine mesh yields a high-fidelity (HF) estimate that is relatively more time consuming but more accurate. Multi-fidelity methods (MF) are a special class of methods that attempt to efficiently combine the information from different fidelities with the overarching goal of identifying optimal design(s) that are good based on HE estimates.

A number of different approaches to MF optimization have been investigated in the literature. Many of these approaches are based on the co-kriging technique described by Forrester et al. [6] which correlates the two sets of samples, low- and high-fidelity, to produce a single prediction model. This technique is an extension of the autoregressive model first introduced by Kennedy and O'Hagan [7]. The two key aspects in which the different methods in this category deviate from each other is in how they collect these samples and how they search the model for potential candidates for evaluation. Laurenceau and Sagaut [8] investigated a number of different sampling methods for use with kriging and co-kriging models and illustrated their performance using an airfoil design problem. Huang et al. [9] combined a genetic algorithm with co-kriging to optimize wing-body drag reduction in aerodynamic design. Perdikaris et al. [10] incorporated elements of statistical learning into a co-kriging framework to cross-correlate ensembles of multi-fidelity surrogate models. Yang et al. [11] adopted a physics-informed approach, constructing models based on sparsely observed domain knowledge, representing unknowns as random variables or fields which are regressed using elements of co-kriging. Giraldo et al. [12] provided an extension to co-kriging for use when the secondary variable is functional, based on the work of Goulard and Voltz [13].

Among the approaches that do not use co-kriging models, some of the prominent ones include the following. Lv et al. [14] employed a canonical correlation analysis-based model, in which the least squares method was used to determine the optimal parameters. Ariyarit and Kanazaki [15] used a hybrid method which employed a kriging model to estimate local deviations and a radial basis function to approximate the global model for airfoil design problems. Hebbal et al. [16] and Cutajar et al. [17] both use machine learning techniques that treat the layers of a deep Gaussian

The authors are with the School of Engineering and Information Technology, The University of New South Wales, Australia.
Emails: {angus.kenny, t.ray, h.singh}@adfa.edu.au.

[1]For brevity, the discussion is restricted to simulation-based design, but the same principles can be applied to other expensive optimization problems such as those involving physical prototyping.

process as different fidelity levels to capture non-linear correlations between various fidelities. Xu et al. [18] used a two-stage process where in the first ordinal transformation was used to transform the original multi-dimensional design space into a one-dimensional ordinal space and then the sampled from this ordinal space using a method based on optimal computational budget allocation (OCBA) algorithm proposed by Chen and Lee [19]. Branke et al. [4] and Lim et al. [20] both adopt evolutionary approaches to solving MF problems. Bryson and Rumpfkeil [21] proposed a quasi-Newton method framework which can be used with many different surrogate models while Ng and Willcox [22] proposed a number of approaches for multi-fidelity optimization under uncertainty.

The majority of these approaches sample solutions *a priori* and evaluate them in low- and high-fidelity and build models based on these samples and use some global search method to optimize them. To ensure these constructed models are adequately representative, it is important to maintain a diversity of samples across the entire the design space. However, sampling without any prior knowledge can result in spending computational resources in areas which are less promising. Of those which do sample iteratively, information is typically only shared in one direction between the two datasets. Low-fidelity solutions are sampled randomly, or using some independent process, and used to inform where the high-fidelity solutions should be sampled from; but no information is then shared in the reverse direction, to inform the low-fidelity sampling process. Again, this can result in computational resources being consumed inefficiently in regions of the search space which do not yield high quality solutions, especially in high-dimensional problems.

To overcome these limitations and improve the performance for MF methods, this paper proposes an iterative two-stage, bound-constrained, single-objective multi-fidelity optimisation algorith, referred to here as MFITS. It uses previously obtained information about promising areas of the search space to define a restricted neighbourhood using a guided differential evolution (DE) [23] process based on a kriging model of the low-fidelity samples. This neighbourhood is then sampled from and searched, using a method derived from OCBA, to determine a set of candidates that undergo low-fidelity simulation. The information from these simulations is used to update the low-fidelity model and also a co-kriging-based surrogate model of the high-fidelity samples, which is searched globally using DE to find a suitable candidate for high-fidelity simulation. Finally, these high-fidelity samples are used to update the surrogate model and also to help determine the restricted neighbourhood in the next iteration. By using the high-fidelity simulation information to inform and restrict the region of interest while searching the low-fidelity model, MFITS maintains two-way information sharing between the sets of samples. The performance of the MFITS algorithm is compared against a baseline co-kriging-based MF algorithm on two separate suites of test functions. The first is a common set of multi-fidelity test-functions from the literature, and the second is a set of multi-fidelity test functions that are generated from the standard test functions using the methods described in the paper by Wang et al [24]. In addition to this, some important properties of MFITS are also investigated.

The remainder of this paper is organized as follows. Section II describes the nature of problems studied in this work, along with the background and related work that form some of the key components in the proposed solution approach. The MFITS algorithm is detailed in Section III, describing all of its constituent parts and detailing some similarities and differences with related techniques from the literature. Experimental design and test problems are discussed in Section IV, with Section V presenting the results of these experiments and a discussion of their implications. Finally, Section VI provides the conclusion and outlines some future directions for potential improvement.

II. Background and Related Work

This section details the type of problem MFITS is designed to address. It also provides information about two critical components that are used in its construction and summarizes its differences and similarities with another method which is similar to MFITS from the literature.

A. Problem type

The algorithm presented here is designed for bound-constrained, single-objective, multi-fidelity optimization problems with continuous variables. Let P be a problem instance with D decision variables and let $\boldsymbol{x} \in \prod_{i=1}^{D}[l_i, u_i]$ be a *solution* to P, represented by a real vector[2] $\boldsymbol{x}$ such that $l_i \leq x_i \leq u_i \quad \forall i \in [D]$, where $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^D$ are the lower- and upper-bounds, respectively[3]. The performance of the solution i.e., *objective value* of the solution to P is given by the function

$$f : \prod_{i=1}^{D}[l_i, u_i] \to \mathbb{R}, \ \boldsymbol{x} \mapsto f(\boldsymbol{x}), \tag{1}$$

which is typically smooth. When this objective function is optimized, the *optimal* solution to P is $\boldsymbol{x}^*$, such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \prod_{i=1}^{D}[l_i, u_i]$. Without loss of generality, the problems are presented in a minimization sense throughout this paper. To facilitate development and testing of optimization algorithms, a range of test problems have been formulated in the literature. They are very fast to evaluate and can be customized to gauge the performance of the algorithms across a variety of fitness landscapes, before their application to the real-world problems.

From the standard version of test functions discussed above, multi-fidelity benchmark functions are generated using additional transformations. The test functions in Lv et

[2]Bold type is used here to indicate a vector (indexed by superscript) and regular type is used for elements (indexed by subscript). E.g., $\boldsymbol{x}^i$ is the ith indexed vector, and x_j^i is the jth element of the ith vector.

[3]To conserve space, the following shorthand is used in this paper: $[k] = \{1, 2, \ldots, k\}$ and $[k^*] = \{0, 1, \ldots, k-1\}$.

al. [14] uses the customized addition, removal and modification of terms to produce a transformed function with a desired correlation to the original. While this method allows control over the shape of the fitness landscape, it requires analysing each test function individually and only permits a single level of fidelity for each transformation.

The second strategy introduces external noise which models the errors that occur when simulation fidelity is decreased. Wang et al. [24] analysed the behaviour of many numerical simulations under different fidelity conditions and formulated a generic transformation function to turn any test function f of the form in Equation 1 into a low-fidelity version $\tilde{f}$:

$$\tilde{f}: \prod_{i=1}^{D}[l_i, u_i] \times [0, 10000] \to \mathbb{R},\ (\boldsymbol{x}, \phi) \mapsto f(\boldsymbol{x}) + e(\boldsymbol{x}, \phi)\,, \tag{2}$$

where ϕ is the *fidelity level*, with $\tilde{f}(\boldsymbol{x}, 10000) = f(\boldsymbol{x})$ and $\tilde{f}(\boldsymbol{x}, 0)$ having the worst possible correlation to $f(\boldsymbol{x})$. The *error function*, $e(\boldsymbol{x}, \phi)$ — which returns a single real value — can be one of ten different functions, all of which are independent of $f(\boldsymbol{x})$.

Because the error function is always independent of $f(\boldsymbol{x})$, this method can be applied to any test function. Since ϕ is real-valued, many different fidelity levels can be modelled easily— although some analysis must still be performed if specific correlation coefficients are required. In this study, we use this approach to generate multi-fidelity problem instances for numerical experiments.

The cost ratio of high- to low-fidelity evaluations, R_c, is a problem characteristic that affects the relative number of low- and high-fidelity evaluations that can be performed within a given computational budget. When $R_c = 1$, there is no difference in cost between the two, meaning that every candidate solution can be evaluated in high-fidelity; however, as R_c tends toward zero, high-fidelity evaluations need to be used sparingly (while a much larger number of low-fidelity evaluations may be affordable), to avoid depleting the budget before a good solution can be obtained. For the specific problem instances presented in Section IV, $R_c = 0.2$ is used, which is in agreement with the experiments in [14]. That is to say, an HF evaluation is considered 5 times more expensive than an LF evaluation.

B. Kriging and co-kriging

Kriging model [1] originated from geostatistical methods used for ore valuation in mining research, but has since been applied across a number of domains. Kriging is an interpolation technique that uses a limited set of samples to predict the objective value at a point that has not been sampled yet.

Let P be a problem instance with D decision variables and let $X = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n\}$ be a set of n sample points with observed objective values $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$, where $\boldsymbol{x}^i \in \mathbb{R}^D$ for $i \in [n]$. A kriging model of these samples is a Gaussian process

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}) + Z(\boldsymbol{x}), \tag{3}$$

where $f(\boldsymbol{x})$ is a polynomial regression function based on $\boldsymbol{y}$ and encapsulates the main variations in the samples. The function $Z(\boldsymbol{x})$ is a Gaussian process with mean 0, which models the residual error. As the mean of $Z(\boldsymbol{x})$ is 0, $f(\boldsymbol{x})$ is the mean of $Y(\boldsymbol{x})$.

Low-fidelity samples are typically much cheaper to evaluate than high-fidelity ones, a fact which the *co-kriging* technique builds upon. Adapted from Kennedy and O'Hagan's autoregressive model [7], co-kriging correlates multiple sets of samples with different fidelities to produce a single model that approximates the high-fidelity performance. The residual error for a sample point evaluated at a given fidelity is recursively modelled as a function of the error of the same point evaluated at the fidelity below it — until some foundational model with the lowest fidelity is reached. Because of this, the Markov property must be assumed, i.e., given a sample point evaluated at some fidelity, no more information about that point can be gained by evaluating it at a lower fidelity.

Let $X_H = \{\boldsymbol{x}_H^1, \boldsymbol{x}_H^2, \ldots, \boldsymbol{x}_H^n\}$ be the set of n high-fidelity sample points with observed objective values $\boldsymbol{y}_H = \{y_{H(1)}, y_{H(2)}, \ldots, y_{H(n)}\}$ and $X_L = \{\boldsymbol{x}_L^1, \boldsymbol{x}_L^2, \ldots, \boldsymbol{x}_L^m\}$ be the set of m low-fidelity[4] sample points with observed objective values $\boldsymbol{y}_L = \{y_{L(1)}, y_{L(2)}, \ldots, y_{L(m)}\}$. A kriging model $Y_L(\boldsymbol{x})$ of the low-fidelity samples are constructed according to Equation 3. Using this, the high-fidelity model is represented as:

$$Y_H(\boldsymbol{x}) = \rho(\boldsymbol{x})\mu_L(\boldsymbol{x}) + Z_d(\boldsymbol{x}), \tag{4}$$

where $\rho(\boldsymbol{x})$ is a scaling factor, determined as part of the maximum likelihood estimation (MLE) of the second model; $\mu_L(\boldsymbol{x})$ is the mean of $Y_L(\boldsymbol{x})$; and $Z_d(\boldsymbol{x})$ is a Gaussian process which models the difference between $Y_H(\boldsymbol{x})$ and $\rho(\boldsymbol{x})\mu_L(\boldsymbol{x})$.

For a more in-depth mathematical treatment of kriging, co-kriging and how to use these models to make predictions, the interested readers are referred to [1], [6], [7], [25].

C. Optimal computing budget allocation

Stochastic simulation is a noisy process, requiring multiple simulation replications in order to accurately approximate the true fitness value of a given design. Assuming a normal distribution, the mean fitness value, μ, for a design is unknown, but it can be estimated by its sample mean $\hat{\mu}$. Given a set of k candidate designs and a finite computing budget T, the optimal computing budget allocation (OCBA) [19] method aims to find an allocation such that $N_1 + N_2 + \ldots + N_k = T$, where N_i is the total replications allocated to design i, in order to select the best design, $b \in [k]$, such that $\hat{\mu}_b < \hat{\mu}_i$ for all $i \in [k]$.

[4]Although only two fidelity levels are employed here, without loss of generality, the method can be extended to an arbitrary number of fidelities.

The probability that design b actually *is* the the best design is called the probability of correct selection (PCS). The PCS can be estimated using Monte Carlo simulation, but this is time-consuming. Therefore, the following problem is formulated in [19] to compute the approximate probability of correct selection (APCS):

$$\underset{N_1,\ldots,N_k}{\text{maximize}} \quad 1 - \sum_{i=1, i\neq b}^{k} P\{\tilde{\mu}_b < \tilde{\mu}_i\}\,,$$

$$\text{such that} \quad \sum_{i=1}^{k} N_i = T\,, \quad N_i \geq 0, \tag{5}$$

Based on the work in [19], the APCS is asymptotically maximized (as $T \to \infty$) when

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2, \tag{6}$$

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{N_i^2}{\sigma_i^2}}\,, \tag{7}$$

where N_i is the total replications allocated to design i and $\delta_{b,i} = \hat{\mu}_b - \hat{\mu}_i$, for all $i, j \in \{1, 2, \ldots, k\}$ with $i \neq j \neq b$.[Ray: sigma needs to be defined]

The implications of Equations 6 and 7 are such that additional computing resources are not only allocated to those designs with a small sample mean, but also to potentially promising designs that have a high variance. A high variance indicates uncertainty in the prediction, which the increased volume of replications will help to reduce.

Using the above results, Algorithm 1 details an iterative process to select a design, based on maximising APCS.

Algorithm 1: OCBA procedure

```
Input: $k$, number of designs; $T$, computing budget; $\Delta$, replications per
    update; $n_0$, initial replications.
Output: $b$, index of best design.
 1: $b \leftarrow \emptyset$ {Initialize $b$}
 2: $N_i \leftarrow n_0\,,\ \forall i \in [k]$ {Count initial replications}
 3: $\boldsymbol{S}^i \leftarrow Sim(n_0)\,,\ \forall i \in [k]$ {Perform initial replications}
 4: while $\sum_{i\in[k]} N_i < T$ do
 5:     $\boldsymbol{N'} \leftarrow \boldsymbol{N}$ {Store old allocations}
 6:     $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}} \leftarrow Stats(\boldsymbol{S}),\ \forall i \in [k]$ {Compute sample statistics}
 7:     $b \leftarrow \text{argmin}_i\ \hat{\mu}_i,\ \forall i \in [k]$ {Update $b$}
 8:     $\boldsymbol{N} \leftarrow Allocate(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}, \boldsymbol{N}, \Delta)$ {Allocate $\Delta$ replications}
 9:     $S_i \leftarrow Sim(N_i - N_i'),\ \forall i \in [k]$ {Perform additional simulations}
10: end while
```

In this algorithm, $Sim(n)$ is a function that returns the output of running the stochastic simulation n times; $Stats(\boldsymbol{S})$ computes the sample statistics for a set of simulation outputs; and $Allocate(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}, \boldsymbol{N}, \Delta)$ is a function that allocates Δ replications among the designs, in accordance with Equations 6 and 7.

D. MO^{2}TOS

Optimal computing budget allocation (OCBA) operates on the assumption that each stochastic simulation being performed will have the same input parameters, and does not consider the case where the fitness of a solution may be evaluated with different fidelities. The multi-fidelity optimization with ordinal transformation and optimal sampling (MO^{2}TOS) [18] framework is a two-stage algorithm that takes this into consideration, by combining ideas from OCBA with ordinal transformation [26].

Given a problem instance P and a finite computing budget T, the MO^{2}TOS algorithm takes advantage of the relatively low cost of low-fidelity evaluations to identify promising regions of the search space that can subsequently be exploited by iterative high-fidelity evaluations.

A large population of solutions is evaluated in low-fidelity and then ranked based on the evaluated values. This ranked population is then partitioned into equal-sized groups. The reason for ranking the solutions before partitioning, is to increase the probability that solutions which share a group are close to each other in the transformed objective space, as opposed to the original design space. As a result, regardless of where the solutions may be, it is more likely that solutions within a group will have a similar performance.

These partitioned groups are treated as de facto "designs" for the purposes of OCBA allocation. However, instead of multiple simulation replications being performed on the same design, the allocations will determine how many solutions are selected from a group for high-fidelity evaluation.

Initially, each n_0 solutions are selected — and removed — from each group, and evaluated in high-fidelity. These results are used to determine the sample statistics of the groups. Equations 6 and 7 are used to allocate a portion of the total budget, Δ, to the set of groups. Solutions are selected from these groups in accordance with this allocation, and evaluated in high-fidelity. The sample statistics are computed once again and the process repeats until the total budget T is consumed.

The value of Δ is typically quite small, as there may be significant bias between the low- and high-fidelity evaluated solutions. By keeping Δ small, it ensures that the sample statistics of the groups are updated more frequently, meaning that the allocations are based more on the statistics of each group, as opposed to their initial partitioning.

The MO^{2}TOS framework has some limitations, due to the fact that it samples the low-fidelity candidates *a priori* and it is a two-step process that operates directly on the high- and low-fidelity functions, as opposed to an iterative one. By sampling once, without any prior information, the sampling must occur uniformly across the entire design space. This means that computational budget is likely to be expended in areas which are not helpful to the search. Similarly, it is necessary to initially evaluate n_0 solutions with high-fidelity from each partitioned group, which can result in further unnecessary expenditure of computational budget.

Algorithm 5 in Section IV provides the basic structure of the MO^{2}TOS procedure used for numerical benchmarking in this paper and more details can be found in [18].

E. A motivating example

A key limitation of the MO^{2}TOS framework identified in this study is the fact that any solution that is evaluated in high-fidelity must be selected from the initial set of low-fidelity solutions. To demonstrate this, the one-dimensional, multi-modal, multi-fidelity test function from Xu et al. [18] is used as a motivating example (Equations 8 and 9).

When applying MO^{2}TOS to this problem, the design space was discretized into 10,000 feasible solutions in [18], all of which are evaluated and ranked using the low-fidelity function. Although low-fidelity evaluations are significantly cheaper than high-fidelity ones, they do not have zero cost, meaning the computational budget consumed by MO^{2}TOS in its initialization phase is already potentially large — depending on the relative cost of low- and high-fidelity evaluations. In situations where the computational budget is very limited and the ratio of high- to low-fidelity evaluations moves away from one, the limitations of MO^{2}TOS begin to emerge. This is illustrated in Figure 1, where MO^{2}TOS is applied to this problem with a budget of 200 low-fidelity-equivalent evaluations.

$$f_H(\boldsymbol{x}) = -\frac{\sin^6(0.09\pi x)}{2^{2((x-10)/80)^2}} - 0.1\cos(0.5\pi x) - 0.5\left(\frac{x-40}{60}\right)^2 - 0.4\sin\left(\frac{x+10}{100}\pi\right), \quad (8)$$

$$f_L(\boldsymbol{x}) = -\frac{\sin^6(0.09\pi x)}{2^{2((x-10)/80)^2}}. \quad (9)$$

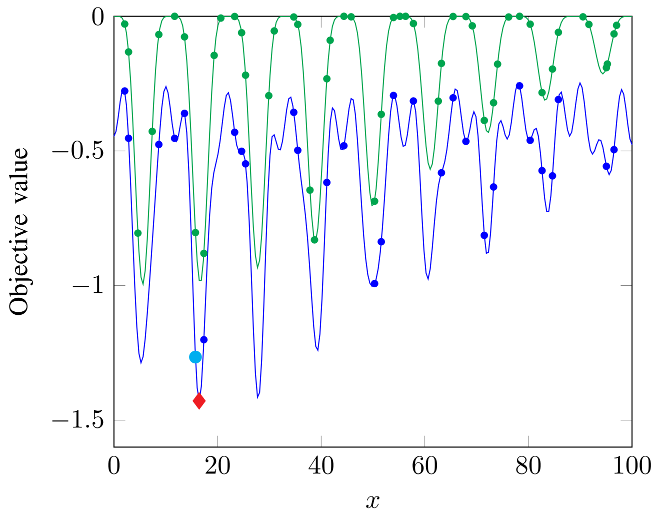


Fig. 1: None of the initial low-fidelity samples is $\boldsymbol{x}^*$, therefore the optimal solution cannot be obtained by MO^{2}TOS.

In this figure, the green and the blue lines are the low- and high-fidelity response surfaces, respectively; green points are the initial low-fidelity samples; blue points are the low-fidelity samples chosen for high-fidelity evaluation; the larger cyan point is the best solution found; and the red diamond is the optimal solution, $\boldsymbol{x}^*$. It can be seen from this example that regardless of how many the initial low-fidelity samples are evaluated in high-fidelity, there is no way for MO^{2}TOS to identify the optimal solution $\boldsymbol{x}^*$. The implication of this example is that MO^{2}TOS is an effective technique for choosing between a finite set of candidate solutions, but it is not as well suited to searching continuous design spaces. This shortcoming can be potentially overcome by increasing the density of the initial candidates. However, this is not trivial in terms of computational cost, and maintaining a constant density of coverage. As the number of decision variables increases exponentially, more initial samples will be required to maintain a given density; which will quickly become untenable.

III. MFITS Algorithm

The multi-fidelity optimization algorithm proposed in this paper relies on an iterative two-stage approach that maintains two separate surrogate models that share information between them. The first stage involves a search on a kriging based surrogate model for potential candidates, within a restricted region, to evaluate them using low-fidelity computations. The information from this search is used to update the first model and a co-kriging model that combines both low- and high-fidelity samples to approximate the high-fidelity objective function. In the second stage, a search is conducted using the co-kriging model to determine a suitable candidate for high-fidelity evaluation. These high-fidelity evaluations are used to update the co-kriging model and to determine the neighbourhood for the next search using the low-fidelity surrogate. The overarching framework is outlined in Algorithm 2, followed by the discussion of its key components.

Algorithm 2: MFITS procedure

```
Input: $P$, problem data; $N_L$, max size of low-fidelity population; $T$,
    maximum number of evaluations; $\rho$, parameter set for LocalOCBA.
Output: Best solution found $\boldsymbol{x}^\beta$.
 1: $X_v \leftarrow LHS(P),\ \forall v \in \{L, H\}$ {Generate initial populations}
 2: $A_v, t \leftarrow f_v(X_v, 0),\ \forall v \in \{L, H\}$ {Evaluate initial populations}
 3: $\boldsymbol{x}^\beta \leftarrow \emptyset$ {Initialize $\boldsymbol{x}^\beta$}
 4: while $t < T$ do
 5:     $M_L \leftarrow Krige(A_L)$ {Update low-fidelity kriging model}
 6:     $M_C \leftarrow CoKrige(A_L, A_H)$ {Update co-kriging model}
 7:     $\boldsymbol{x} \leftarrow GlobalSearch(M_C)$ {Globally search co-kriging model}
 8:     $\boldsymbol{\alpha}, t \leftarrow f_H(\boldsymbol{x}, t)$ {Evaluate and update total cost}
 9:     $A_H \leftarrow A_H \cup \{\boldsymbol{\alpha}\}$ {Add $\boldsymbol{\alpha}$ to high-fidelity archive}
10:     $\boldsymbol{x}^\beta \leftarrow \min(f_H(\boldsymbol{x}^\beta), f_H(\boldsymbol{x}))$ {Update best solution}
11:     $\varepsilon \leftarrow Sigmoid(t, T)$ {Determine size of neighbourhood}
12:     $A_L, t \leftarrow LocalOCBA(\rho, M_L, A_L, \boldsymbol{x}^\beta, \varepsilon)$ {Locally search
        low-fidelity model}
13:     if $|A_L| > N_L$ then
14:         $A_L \leftarrow Winnow(A_L, N_L)$ {Control population size}
15:     end if
16: end while
```

A. Initialization, evaluation and surrogate models

The function $LHS(P)$ returns an initial population generated from a latin hypercube sampling (LHS), the bounds of which are defined in P. LHS is used as it can provide well distributed solutions in the variable space without requiring exponential number of them, unlike full-factorial sampling. The solutions are evaluated using the function $f_v(X, t)$, at some fidelity $v \in \{L, H\}$, which returns a set of "archive" pairs comprising the solution and its objective value. It also updates the total cost incurred, t, taking into account the cost of evaluating the solutions at fidelity v.

Two surrogate models are built: $Krige(A_L)$ takes the low-fidelity samples and returns a kriging model approximating it, and $CoKrige(A_L, A_H)$ takes both the low- and high-fidelity archive and returns a co-kriging model that approximates the fitness landscape of the high-fidelity objective function. The function $GlobalSearch(M_C)$ takes the co-kriging model as input and returns the best solution that can find using some global search method (on the surrogate model). The solution identified from the above provcess is evaluated in high-fidelity and the information is used to update the co-kriging model, and also to determine the restricted neighbourhood to be used for search using the low-fidelity kriging model, discussed next.

B. *Restricted neighbourhood*

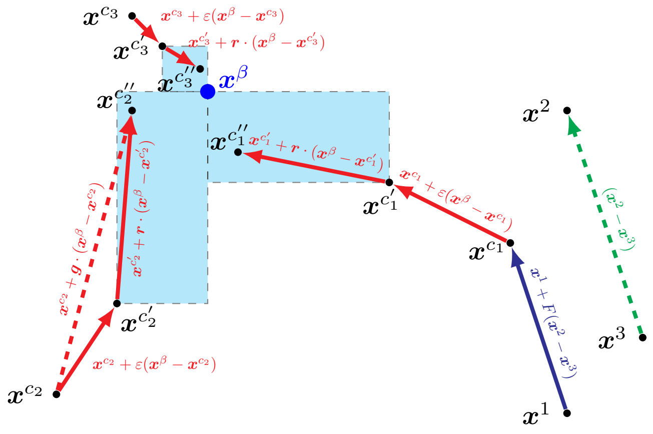
The correlation between the low- and high-fidelity samples can vary significantly between problems, meaning identifying a promising region in the low-fidelity kriging model does not necessarily mean identifying a promising region in the high-fidelity co-kriging model. Therefore, instead of searching the low-fidelity model globally, it makes sense to search for promising candidates within a restricted neighbourhood, defined using information from the high-fidelity model. As the goal of performing low-fidelity evaluations in co-kriging models is to gain information about the shape of the fitness landscape, more than it is to find the "best" low-fidelity solution, restricting the neighbourhood in the algorithm to focus on the region of interest — without getting "distracted" by global optima that might not be useful in the over-all search.

[Ray: Angus can you please take stock of all symbols being used and see any double use. For example I see capitaal x in F(x) and also small bold]In order to restrict the region of interest, a so-called *guided* differential evolution (DE) process is employed. In DE, a candidate child solution x_c is produced from three parent solutions $\boldsymbol{x}^1$, $\boldsymbol{x}^2$ and $\boldsymbol{x}^3$ using the following formula [23]:

$$\boldsymbol{x}^c = \boldsymbol{x}^1 + F(\boldsymbol{x}^2 - \boldsymbol{x}^3)\,, \tag{10}$$

where F is a constant called the mutation factor. This can also be thought of as scaled vector addition, therefore it is possible to direct the resulting child $\boldsymbol{x}^c$ towards a reference point (say $\boldsymbol{x}^\beta$) by translating $\boldsymbol{x}^c$ further.

Figure 2 illustrates this process. In this figure, it can be seen that $\boldsymbol{x}^{c_1}$ is produced using $\boldsymbol{x}^1$, $\boldsymbol{x}^2$ and $\boldsymbol{x}^3$, and then undergoes two translations to produce $\boldsymbol{x}^{c_1'}$ and finally $\boldsymbol{x}^{c_1''}$, the resultant child solution. Solutions $\boldsymbol{x}^{c_2}$ and $\boldsymbol{x}^{c_3}$ are produced using two other sets of solutions that are not pictured, and then translated using the same processes.



$\boldsymbol{x}^{c_3}$
$\boldsymbol{x}^{c_3} + \varepsilon(\boldsymbol{x}^\beta - \boldsymbol{x}^{c_3})$
$\boldsymbol{x}^{c_3'}$
$\boldsymbol{x}^{c_3'} + \boldsymbol{r} \cdot (\boldsymbol{x}^\beta - \boldsymbol{x}^{c_3'})$
$\boldsymbol{x}^{c_3''}$
$\boldsymbol{x}^\beta$
$\boldsymbol{x}^{c_2''}$
$\boldsymbol{x}^{c_1''}$
$\boldsymbol{x}^{c_1'} + \boldsymbol{r} \cdot (\boldsymbol{x}^\beta - \boldsymbol{x}^{c_1'})$
$\boldsymbol{x}^2$
$\boldsymbol{x}^{c_1'}$
$\boldsymbol{x}^{c_1} + \varepsilon(\boldsymbol{x}^\beta - \boldsymbol{x}^{c_1})$
$(\boldsymbol{x}^2 - \boldsymbol{x}^3)$
$\boldsymbol{x}^{c_2} + \boldsymbol{g} \cdot (\boldsymbol{x}^\beta - \boldsymbol{x}^{c_2})$
$\boldsymbol{x}^{c_2'} + \boldsymbol{r} \cdot (\boldsymbol{x}^\beta - \boldsymbol{x}^{c_2'})$
$\boldsymbol{x}^{c_1}$
$\boldsymbol{x}^1 + F(\boldsymbol{x}^2 - \boldsymbol{x}^3)$
$\boldsymbol{x}^{c_2'}$
$\boldsymbol{x}^3$
$\boldsymbol{x}^{c_2} + \varepsilon(\boldsymbol{x}^\beta - \boldsymbol{x}^{c_2})$
$\boldsymbol{x}^{c_2}$
$\boldsymbol{x}^1$

Fig. 2: The DE recombination process can be "guided" towards the best high-fidelity solution found so far.

At the beginning of the search, $\boldsymbol{x}^\beta$ is determined using a very coarse model built with sparse information, and cannot be trusted to indicate a promising region of the search space. As the search continues, the model becomes more accurate and the information it produces more trustworthy. Therefore, the algorithm should be explorative in the beginning of the search, but exploitative towards the end.

Takahama and Sakai [27] proposed the ε-constrained adaptive differential evolution (εADE) method for efficient constraint handling when solving constrained optimization problems, and it is this method which is the inspiration for the guided DE process in Figure 2. Simply put, εADE uses an adaptive variable ε to control how much constraint violation is allowed at any given time during the search. Early on in the search, ε is high, allowing more constraint violation and promoting exploration; as the search continues, ε tends toward zero, biasing the search towards exploiting regions it has already identified as promising.

In the guided DE process illustrated in Figure 2, a similar principle is adopted. Each solution $\boldsymbol{x}^{c_1}$, $\boldsymbol{x}^{c_2}$ and $\boldsymbol{x}^{c_3}$ is "nudged" toward $\boldsymbol{x}^\beta$ by some factor ε (in Figure 2, $\varepsilon = 0.4$), which is in the range $[0, 1)$, starting at 0 at the beginning of the search and tending towards 1 (but never actually reaching it) towards the end. This gives the first translation equation:

$$\boldsymbol{x}^{c'} = \boldsymbol{x}^c + \varepsilon(\boldsymbol{x}^\beta - \boldsymbol{x}^c)\,. \tag{11}$$

If this was the only translation applied, then all possible solutions produced by the guided DE process would lie on a line with endpoints at $\boldsymbol{x}^c$ and $\boldsymbol{x}^\beta$. This neglects a lot of the search space and is too greedy, especially in higher dimensional problems. To address this, a second translation is applied:

$$\boldsymbol{x}^{c''} = \boldsymbol{x}^{c'} + \boldsymbol{r} \cdot (\boldsymbol{x}^\beta - \boldsymbol{x}^{c'})\,, \tag{12}$$

where, $\boldsymbol{r} \in [0, 1]^D$ is a random vector with D components. By making $\boldsymbol{r}$ a vector and not a scalar, the set of solutions that can be produced is no longer restricted to a line; instead, any solution within the D-dimensional hyperrectangle with

with diagonal vertices at $\boldsymbol{x}^{\beta}$ and $\boldsymbol{x}^{c'}$ (indicated by the shaded rectangles in Figure 2) can be produced.

In combining these two translation operations, the guided DE makes an initial "bee-line" for $\boldsymbol{x}^{\beta}$ by some factor ε, determined by how much of the search has completed, effectively pruning a lot of the search space, before randomly searching the vicinity of $\boldsymbol{x}^{\beta}$ of to produce the resultant child solution $\boldsymbol{x}^{c''}$. Substituting Equation 11 into Equation 12 gives

$$\begin{aligned}\boldsymbol{x}^{c''} &= \boldsymbol{x}^{c} + (\varepsilon + (1-\varepsilon)\boldsymbol{r}) \cdot (\boldsymbol{x}^{\beta} - \boldsymbol{x}^{c}) \,, \\ &= \boldsymbol{x}^{c} + \boldsymbol{g} \cdot (\boldsymbol{x}^{\beta} - \boldsymbol{x}^{c}) \,, \end{aligned} \tag{13}$$

with $\boldsymbol{g} \in [\varepsilon, 1)^{D}$ being a random vector with D components, known as the *guide factor*. Equation 13 is illustrated by the dashed red line in Figure 2.

If this process is continued for all solutions in the population, the result is a cloud of candidate solutions in the vicinity of $\boldsymbol{x}^{\beta}$, the centre of the region of interest.

As mentioned, the value of ε should be small at the beginning of the search, and tending towards 1 and flattening by around 80% — consistent with the principles of εADE. One function which exhibits these properties is the logistic sigmoid curve:

$$\varepsilon = \frac{L}{1 + e^{-k(x - x_0)}} \,, \tag{14}$$

where x_0 is the x value of the sigmoid's midpoint, L is the curve's maximum and k is the steepness parameter, which controls how fast the function rises. The function $Sigmoid(t, T)$ uses the total cost and maximum cost to compute this ε value. Figure 3 gives a plot of the logistic sigmoid curve with $L = 0.99$, $k = 10$ and $x_0 = 0.2$. Computing ε for Equation 13 using Equation 14 ensures

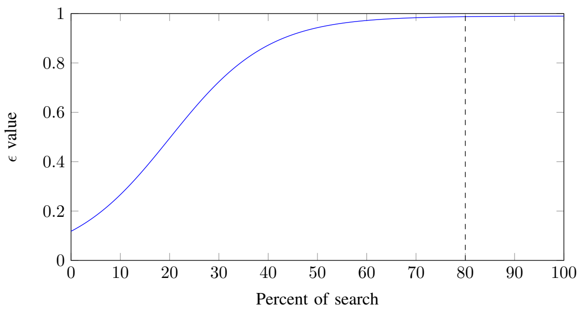


Fig. 3: The logistic sigmoid curve flattens at around 80%.

the cloud of candidate solutions around $\boldsymbol{x}^{\beta}$ is sparse at the beginning of the search, allowing for better global exploration, while focusing the search more tightly on promising regions towards the end. Crucially, the parameters needed to control the sigmoid curve are intuitive to set and do not require tuning to achieve the desired behavior.

C. *LocalOCBA*

Once the candidate solutions have been been generated in the vicinity of $\boldsymbol{x}^{\beta}$, some of them need to be selected for evaluation at low-fidelity. As already stated, the goal of evaluating solutions in low-fidelity is not to optimize the low-fidelity objective function, but to provide as much information for the model as possible. The optimal computing budget allocation (OCBA) algorithm selects from a set of solutions with the goal of reducing uncertainty within simulation models by adaptively allocating computing resources. This principle can be applied here, with some modifications, as outlined in Algorithm 3.

Algorithm 3: $LocalOCBA$ procedure

```
Input: $\rho = \{\Delta_1$, total to be sampled; $\Delta_2$, samples per statistics update$\}$;
    $M$, low-fidelity model; $A_L$, low-fidelity archive; $\boldsymbol{x}^{\beta}$, best
    high-fidelity solution ; $\varepsilon$, sigmoid value; $t$, total cost incurred.
Output: $A_L$, updated low-fidelity archive; $t$, updated total cost.
 1: $X \leftarrow GuidedDE(A_L, \boldsymbol{x}^{\beta}, \varepsilon)$ {Generate child population}
 2: $A_X \leftarrow f_M(X)$ {Approximate each child by model $M$}
 3: $G, q \leftarrow Partition(A_X)$ {Partition solutions}
 4: $S_i \leftarrow \emptyset,\ \forall i \in [q]$ {Empty groups for selected solutions}
 5: while $\sum_{i \in [q]} |S_i| < \Delta_1$ do
 6:     $\hat{\mu}_i, \hat{\sigma}_i \leftarrow$ sample statistics for $S_i$, $\forall i \in [q]$ (if $|S_i| < 2$, use $G_i$)
 7:     $R \leftarrow GetRatios(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})$ {compute allocation ratios}
 8:     $D \leftarrow Allocate(R, S, G) : \sum_{i \in [q]} |D_i| = \Delta_2$ {Allocate $\Delta_2$
        solutions according to ratios $R$}
 9:     $D_i, t \leftarrow f_L(D_i, t),\ \forall i \in [q]$ {Evaluate allocated solutions}
10:     $S_i \leftarrow S_i \cup D_i,\ \forall i \in [q]$ {Add to selected solutions}
11:     $G_i \leftarrow G_i \setminus D_i,\ \forall i \in [q]$ {Selection without replacement}
12: end while
13: $A_L \leftarrow A_L \cup \bigcup_{i \in [q]} S_i$ {Combine all selected solutions}
```

Here, the $GuidedDE(A_L, \boldsymbol{x}^{\beta}, \varepsilon)$ process is used to generate a set of candidate solutions, which are approximated using $f_M(X)$. This function takes a set of solutions and returns a set of pairs comprising the solution and its approximation on the (surrogate) model M. The solutions are ranked and partitioned using a k-means clustering algorithm, based on their approximated value. The purpose of ranking the solutions first is to increase the probability that solutions within a clustered group will tend to have a similar performance to each other, regardless of their proximity in the decision space. This helps to ensure diversity of selected solutions, while still preferring the more promising candidates. The function $Partition(X)$ returns a set of solution groups G and the number of groups q.

OCBA principles are used to select — and remove — from these groups to populate a set of empty groups S. First, sample statistics are computed for all groups of selected solutions in S. If there are fewer than two solutions in a group, then its corresponding group in G is used. The function $GetRatios(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})$ uses these statistics to compute allocation ratios in accordance for each group with standard OCBA practice. These ratios are used by $Allocate(R, S, G)$ to allocate Δ_2 solutions to be evaluated as low-fidelity and added to the selected solutions S.

Once Δ_1 solutions have been selected in the set S, they are evaluated in the low-fidelity. Then, they are added to the low-fidelity archive, and the updated archive is returned.

D. Archive size control

Due to the fact that many more low-fidelity solutions are added to the archive between each high-fidelity evaluation, the size of the low-fidelity archive can become too big for some kriging and co-kriging algorithms, or too concentrated if the search is focused on the same area for too long. Therefore, a maximum archive size N_L is set such that once it reaches that threshold, the population should be maintained at that level. Choosing a steady-state method such as ranking the solutions by their value and selecting the top N_L will cause the population to converge and lose diversity over time. As the purpose of maintaining the low-fidelity population is to provide the co-kriging model with information about the shape of the fitness landscape, this loss of diversity can be detrimental.

The function $Winnow(A_L, N_L)$ processes an archive of solutions and returns a winnowed archive with exactly N_L solutions. It does this by partitioning the solutions in the decision space into N_L different groups using a clustering algorithm, such as k-means clustering. Most of these clusters will contain only one solution which are directly added to the winnowed archive; for those that have more than one solution, only the solution with the best value is selected. This ensures that the archive never has more than N_L solutions, but diversity is maintained throughout the population.

E. Similarities and differences to MO^{2}TOS

There exist some similarities between MFITS and the MO^{2}TOS framework. For example, both use a two-step process of ordering a population of solutions and then selecting from them using ideas from OCBA. Despite the similarities, there are several key aspects which differentiate the two algorithms from each other.

The biggest difference is that MFITS involves an iterative sampling and evaluation process, whereas MO^{2}TOS is a two-step algorithm which is only executed once. As MO^{2}TOS only performs a single iteration, it is not able to use any prior knowledge to dynamically determine where it should concentrate its resources when evaluating the initial low-fidelity population. Therefore, it evaluates uniformly across the whole search space, and subsequently spends computational budget in areas which are not beneficial to the search. In contrast, MFITS uses information from previous iterations, to identify promising areas of the search space that it can exploit.

Another difference is that MO^{2}TOS operates on the high- and low-fidelity objective functions directly across the entire search space. Solutions are selected using information from low-fidelity evaluations, to be evaluated in high-fidelity. In contrast, MFITS uses its ranking and selection phases in order to select from a neighbourhood of solutions that have been identified as promising regions of the search space. These selections are informed by a kriging model of the low-fidelity function, and selected for low-fidelity evaluation. The high-fidelity evaluations are determined by a separate search that is performed on the co-kriging model, updated by the selected low-fidelity evaluations. Because of this, it is not necessary to perform the initial n_0 evaluations before computing the sample statistics, as the goal is not to optimize the low-fidelity objective function but rather just select from a set of ranked candidates.

The fact that MO^{2}TOS must determine all of its candidates *a priori* means that it can only sample from a predefined, finite set of solutions, and is unable to refine what it produces. To achieve a high precision, a large number of low-fidelity evaluations solutions may be required. Since low-fidelity evaluations have low but non-zero cost in relation to high-fidelity, the computational effort required in sampling large set of solutions in low-fidelity can still be excessive. This is illustrated in Figure 1. By doing on-line sampling and creating a surrogate model which can be used for global search, MFITS is, in theory, able to produce any feasible solution in the search space and improve upon the promising candidates by focusing the search on their local neighbourhood.

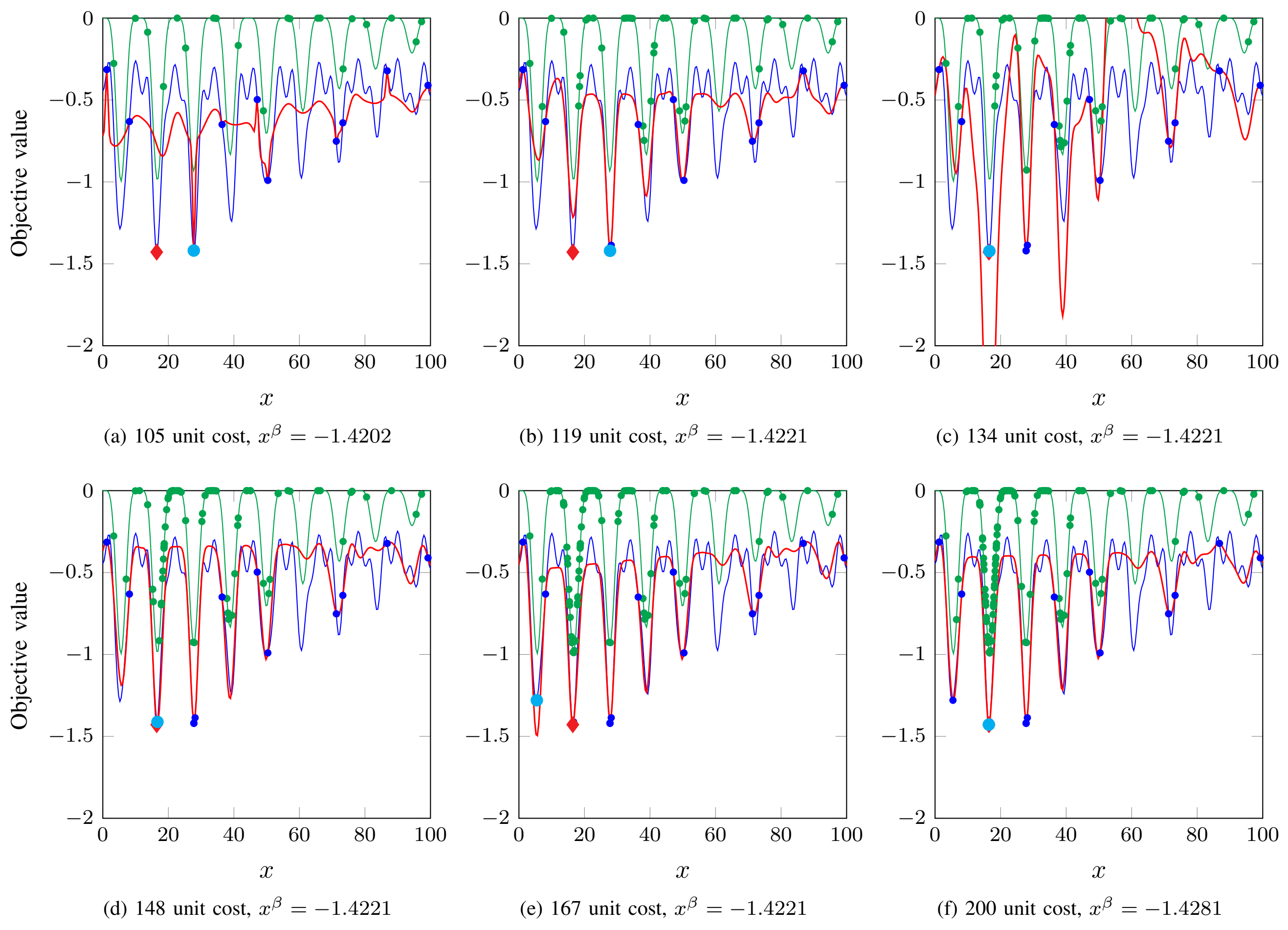
Finally, MO^{2}TOS partitions the ranked population into a fixed number of equal-sized groups, whereas MFITS uses a clustering algorithm to determine the number and size of partitions. This ensures that the average distance between groups in objective space is maximized.

F. Proof of concept

To visualise the operation of the proposed method and demonstrate its advantages, MFITS is applied to the motivating example given in Section II with a limited computational budget (200 low-fidelity-equivalent evaluations) to highlight the shortcomings of the MO^{2}TOS algorithm (Figure 4).

In this figure, the green and blue lines are the low- and high-fidelity response surfaces, respectively; the red line is the co-kriging approximation of the high-fidelity function, green and blue points represent the low- and high-fidelity archives, respectively; the larger cyan point is the next solution chosen for high-fidelity evaluation; and the red diamond is the optimal solution, $\boldsymbol{x}^*$. Each sub-figure gives a snapshot of the operation of MFITS on this example, for a given unit cost.

Initially (Figure 4a), the co-kriging approximation is poor, as there are very few samples to inform it and the global search on this model selects a candidate for high-fidelity evaluation which is in the valley adjacent to the one which contains $\boldsymbol{x}^*$. The search remains in this valley, until the co-kriging model has gathered enough information about the landscape to escape the local optima (Figure 4c), where the global search selects a solution that is closer to optimal. The search concentrates in this area until the model is updated again such that it selects a candidate in the valley to the left of the optimal solution (Figure 4e), which has a worse objective than the current best when evaluated. Finally, there is one more update (Figure 4f) which results in the final high-fidelity evaluation of -1.4281, which is near optimal. Xu et al. report the “optimum” as -1.4277, a result of their



(a) 105 unit cost, $x^\beta = -1.4202$ (b) 119 unit cost, $x^\beta = -1.4221$ (c) 134 unit cost, $x^\beta = -1.4221$
(d) 148 unit cost, $x^\beta = -1.4221$ (e) 167 unit cost, $x^\beta = -1.4221$ (f) 200 unit cost, $x^\beta = -1.4281$

Fig. 4: Snapshot of MFITS applied to Equations 8 and 9, with corresponding budget consumed and best solution, x^β.

descretization of the search space, meaning MO^{2}TOS was not able to find the optimum due to precision issues.

Something also to note about Figure 4f is the concentration of the low-fidelity samples on the left, an evidence that the guided DE process is using the computational budget efficiently, focusing only in the areas of the search it has identified as promising. The No Free Lunch Theorem [28] says there must be a trade-off for this concentration — and lack — of attention in certain regions, meaning that the co-kriging approximation is much poorer on the right-hand side of the plot than the left. However, this does not affect the search negatively as the quality of solutions in this region is poor as well.

IV. Numerical Experiments

The experiments were carried out on [Angus: insert details of machine and OS here]. All code was implemented and executed in MATLAB[Angus: version number], with kriging and co-kriging models constructed using the ooDACE toolbox.

The MFITS algorithm was compared against a baseline co-kriging algorithm and an implementation of the MO^{2}TOS [18]. All algorithms were run 30 times on each problem instance for the equivalent of 2000 low-fidelity evaluations, recording the best objective value, the mean and standard deviation of the resulting solutions produced by each trial.

To establish statistical significance of the results, a pairwise comparison was made using the Mann-Whitney-Wilcoxon (MWW) test [29], with the number of "wins", "losses" and "draws" recorded for each algorithm. For each pairwise comparison, let $\delta^p_{ab} = \mu^p_a - \mu^p_b$, with μ^p_i being the mean objective value for algorithm i on problem instance p over all trials. If the p-value for MWW is less than 0.05 and $|\delta^p_{ab}| > 10^{-5}$, algorithm a is said to have "won" when $\delta^p_{ab} < 0$, "lost" when $\delta^p_{ab} > 0$ and "drawn" if both initial conditions are not met.

A. Test problems

The experiments were run on a collection of bound-constrained, single-objective, multi-fidelity test functions that can be divided into two different test suites. The first suite (suite A) is a selection of problem instances taken from Lv et al. [14]. Problems $f10$ to $f17$ were chosen from this suite as they contain between three and eight decision variables, determined as a representative range of easy to difficult problems. The explicit definitions for both the high- and low-fidelity functions are provided in the Appendix.

The second set of problem instances (suite B) is generated using the well-known test functions of Griewank [30] and Michalewicz [31]. Each problem was instantiated with versions for three, five and eight decision variables. This matches the range of the first test suite, and also allows the performance of the algorithms to be judged across different sized problems in a more controlled environment. Since these test problems are not naturally multi-fidelity optimisation problems, low-fidelity funtions are constructed by applying the methods described in [24]. Two different error functions were tested for each instance, modelling two types of fidelity error: resolution and stochastic. An appropriate fidelity level ϕ for each problem was chosen by computing the square of the Pearson correlation coefficient r^2 for different fidelity values and selecting a value for ϕ such that r^2 was between 0.65 and 0.85, to be commensurate with the first test suite.

See Tables IV and V in Appendix A for test functions and error function equations, respectively.

B. MFITS parameters

The MFITS algorithm conforms to a modular framework where various search and modeling methods can be implemented. Its components can be divided into two categories: MFITS specific components, which are intrinsic to its operation and whose parameters are also intrinsic; and, generic components, such as global search techniques and modelling methods which can be substituted by other similar techniques, which come with their own set of parameters. The MFITS specific parameters are discussed in general in Section III.

To reduce the number of arbitrary parameters, initial population sizes are defined as a function of the problem size. The high-fidelity population has a size of $6D$, as this is the smallest number of high-fidelity solutions that can be used to generate a co-kriging model for all the instances, with the low-fidelity population being $18D$, which was chosen using sensitivity analysis to find the smallest value which still produces good solutions across the set of instances.

The maximum size that the low-fidelity archive can reach before it must be truncated is set to 400. This was chosen because values much higher than this significantly slowed the procedure down while yielding minimal-to-no improvement.

In the *LocalOCBA* procedure, the total number of solutions to be sampled, Δ_1, is set to 25 and the number of samples per statistics update, Δ_2, is set to 5. These were chosen to provide a good balance between speed and efficiency. The clustering method used to partition the solutions is k-means clustering, with the value for k being chosen using the so-called *elbow method* [32].

Differential evolution (DE) [23] was used to globally search the updated co-kriging model. The "classical" DE (DE/rand/1/bin) was run over 30 generations with a crossover rate of 0.9, a mutation factor of 0.5 and a population size of 100.

The kriging models used are from the ooDACE toolbox [33], with the default parameters.

C. Baseline co-kriging algorithm

Co-kriging is a popular approach used in solving multi-fidelity optimization problems. The outer-loop of the MFITS algorithm functions by iteratively updating a co-kriging model using data from a modified version of the OCBA procedure. In order to demonstrate the effectiveness of this new technique, the performance of MFITS is compared against the baseline simple co-kriging algorithm given in Algorithm 4.

Algorithm 4: Baseline co-kriging procedure

```
Input: $P$, problem data; $N_{L_{max}}$, max size of LF pop; $N_{e_{max}}$,
    maximum number of evaluations; $\Delta$, new LF per iteration.
Output: Best solution found $\boldsymbol{x}^\beta$.
 1: $X_v \leftarrow LHS(P),\ \forall v \in \{L, H\}$ {Generate initial populations}
 2: $\boldsymbol{x}^\beta \leftarrow \emptyset$ {Initialize $x_\beta$}
 3: while $N_e < N_{e_{max}}$ do
 4:     $A_L \leftarrow A_L \cup LHS(\Delta)$ {Add $\Delta$ new solutions to LF archive}
 5:     if $|A_L| > N_{L_{max}}$ then
 6:         $A_L \leftarrow winnow(A_L, N_{L_{max}})$ {Control population size}
 7:     end if
 8:     $M_C \leftarrow CoKrige(A_L, A_H)$ {Update co-kriging model}
 9:     $\boldsymbol{x} \leftarrow GlobalSearch(M_C)$ {Globally search co-kriging model}
10:     $\boldsymbol{\alpha}, N_e \leftarrow f_H(\boldsymbol{x}, N_e)$ {Evaluate and update total cost}
11:     $A_H \leftarrow A_H \cup \{\boldsymbol{\alpha}\}$ {Add $\boldsymbol{\alpha}$ to high-fidelity archive}
12:     $\boldsymbol{x}^\beta \leftarrow \min(f_H(\boldsymbol{x}^\beta), f_H(\boldsymbol{x}))$ {Update best solution}
13: end while
```

The functions here have the same definitions as in Algorithm 2. This procedure iteratively updates a co-kriging model by randomly sampling the low-fidelity data using a latin hypercube sampling (LHS) technique; however, the global search method is the same.

D. MO^{2}TOS algorithm

The second method used for comparison with MFITS is the MO^{2}TOS algorithm described in Section II. The source code for this algorithm could not be retrieved online, so an implementation in MATLAB based on the description in [18] was made, the pseudocode for which is presented in Algorithm 5.

Here, most of the subroutines are the same as those previously discussed in Sections II and III. The total computing budget, $N_{e_{max}}$, must be divided between the initial low-fidelity evaluations and the subsequent high-fidelity ones, with the proportion of the total computing budget allocated to the high-fidelity evaluations given by the parameter ρ. In [18], no guidelines were provided for setting this parameter; however sensitivity analysis experiments showed that a value of around 0.25 for ρ provided the best results for the given problem instances and the computing budget.

The function $LHS(P, N, \rho)$ takes the problem instance, total budget and ρ as input and returns an initial population of solutions sampled using LHS with size $\frac{N}{1+\rho}$. This

Algorithm 5: MO^{2}TOS procedure

```
Input: $P$, problem data; $N_{e_{max}}$, max LF equivalent evals; $n_0$, initial
    HF evals; $\rho$, HF eval proportion; $k$, number of partitions; $\Delta$, HF
    evals per statistics update.
Output: Best solution found $\boldsymbol{x}^\beta$.
 1: $X_L \leftarrow LHS(P, N_{e_{max}}, \rho)$ {Generate initial population}
 2: $A_L, N_e \leftarrow f_L(X_L, 0)$ {Evaluate initial LF population}
 3: $G \leftarrow Partition(A_L, k)$ {Partition solutions into $k$ groups}
 4: $D_i \leftarrow randSelect(G_i, n_0),\ \forall i \in [k]$ {Select initial HF candidates}
 5: $S_i, N_e \leftarrow f_L(D_i, N_e),\ \forall i \in [k]$ {Evaluate selected solutions}
 6: $G_i \leftarrow G_i \setminus D_i,\ \forall i \in [k]$ {Selection without replacement}
 7: while $N_e < N_{e_{max}}$ do
 8:     $\hat{\mu}_i, \hat{\sigma}_i \leftarrow$ sample statistics for $S_i, \forall i \in [k]$
 9:     $R \leftarrow GetRatios(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})$ {compute allocation ratios}
10:     $D \leftarrow Allocate(R, S, G) : \sum_{i\in[k]} |D_i| = \Delta$ {Allocate solutions}
11:     $D_i, N_e \leftarrow f_L(D_i, N_e),\ \forall i \in [k]$ {Evaluate allocated solutions}
12:     $S_i \leftarrow S_i \cup D_i,\ \forall i \in [k]$ {Add to selected solutions}
13:     $G_i \leftarrow G_i \setminus D_i,\ \forall i \in [k]$ {Selection without replacement}
14: end while
15: $\boldsymbol{x}^\beta \leftarrow \min\left(\bigcup_{i\in[k]} S_i\right)$ {Find best selected solution}
```

population is evaluated in low-fidelity and partitioned using $Partition(A, k)$. It takes an archive of solutions and a specified number of groups k as inputs, ranks the solutions by objective value and returns k equal-size groups based on their ranks. In [18], $k = 10$ was used for the experiments. The remainder of the procedure is similar to Algorithm 3, with $\Delta = 5$ corresponding to the value of Δ_2 in that algorithm.

V. Results and Discussion

The experiments in this section are used to compare the performance of MFITS against a base-line co-kriging algorithm and an implement ion of the MO^{2}TOS framework, as described in Section IV. Test suite A is taken from [14] and demonstrates the performance of MFITS on a variety of problems with different sizes; while test suite B is used to investigate the effect of problem size and error type. The results from each set of experiments are presented in a table with plots comparing the convergence of MFITS and the base-line co-kriging algorithm on a representative sample of the problem instances. MO^{2}TOS is not included in the convergence plots as decisions are made *a priori* and it is not an iterative process.

A summary of the results, their implications and statistical significance is also provided at the end of this section.

Test suite A

Table I presents the results from 30 runs of MFITS, the base-line co-kriging algorithm and MO^{2}TOS on test suite A. In this table, it can be seen that MFITS performs better than MO^{2}TOS for all instances. It also performs as well as or better than the base-line co-kriging algorithm with a few exceptions. For the tests on instances $f11$, $f14$ and $f16$, the base-line co-kriging algorithm was able to find at least one solution that was better than the best solution produced by MFITS, and for the instances $f10$ and $f14$ it produced better solutions on average. In all of these instances, the differences between the algorithms are small, with both performing very similarly; whereas MFITS outperforms the co-kriging by some margin in a lot of the other instances.

This is supported by the convergence plots in Figure 5. Here, it is clear that when the size of the problem is small, the performance of the two algorithms is very similar, with similar convergence rates; but as the size increases, both the average performance, and convergence rate, of MFITS over co-kriging improves significantly — with the most marked improvement being for problem $f17$.

Test suite B

The purpose of test suite$B$ was to observe the effect of problem size on the performance of MFITS, and to do so using two different error functions to ensure the results obtained are consistent and not over-fitted to a particular type of simulation error model. The results in Table II compare the performance of MFITS, MO^{2}TOS and the baseline co-kriging algorithm over 30 independent runs on the instances of test suite B. The test suite comprises of two problem instances from the literature, with varying numbers of decision variables using error functions e_2 and e_6 as described in [24].

Again, the table shows that MFITS outperforms MO^{2}TOS in all instances and the co-kriging baseline algorithm, with a few exceptions. For the Michaelwicz problem using error function e_6 and with the problem instantiated with five decision variables, the co-kriging base-line algorithm managed to find the best over-all solution. However MFITS still performed better on average. The only time when the co-kriging base-line outperformed MFITS on average was on the two smallest instantiations of the Griewank problem. The convergence plots in Figures 6a and 6d show that the difference between the performance of both algorithms for these instances is reasonably negligible. The remaining plots in Figure 6 agree with the results presented above for test suite A.

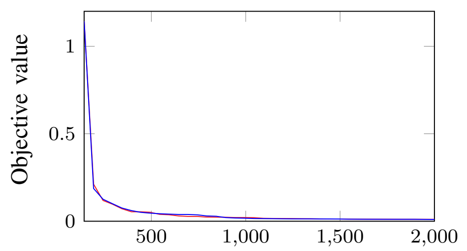
Discussion

Table III provides the results of a pairwise comparison of all three algorithms on both test suites, using the Mann-Whitney-Wilcoxon (MWW) statistical significance test [29]. This table clearly shows that MFITS produces significantly better solutions than both MO^{2}TOS and the base-line co-kriging algorithm when considered across all problem instances and trials.

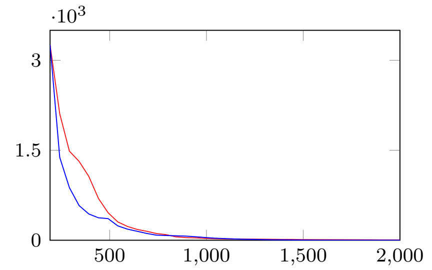
The results from both test suites suggest a similar pattern. For problems with few decision variables, there is very little difference between MFITS and the base-line co-kriging algorithm. As the size of the problem increases, the convergence rate and performance of MFITS improves over the simple co-kriging algorithm with the greatest improvements observed on the larger problem instances.

TABLE I: Results on dataset A comparing MFITS to MO^{2}TOS and the base-line co-kriging algorithm. Given are the number of decision variables (D), the square of the Pearson correlation coefficient (r^2), the best objective obtained, the mean best objective over the full set of runs (μ) and the corresponding standard deviation (σ).

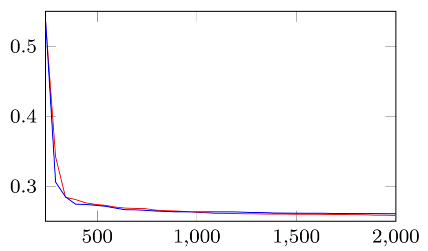
			Co-kriging			MO^{2}TOS			MFITS		
Instance	D	r^2	best	μ	σ	best	μ	σ	best	μ	σ
$f10$	3	0.64	**0**	**2.2960**	3.5445	4.9076	29.6201	33.3903	**0**	3.9189	5.3296
$f11$	3	0.74	**0.0001**	0.0110	0.0077	0.0199	0.2043	0.1710	0.0004	**0.0098**	0.0064
$f12$	4	0.79	-8.1107	-3.8007	1.3426	-3.8370	-1.7674	0.7283	**-9.5783**	**-5.8853**	1.5123
$f13$	4	0.89	0.6290	4.9366	5.0965	14.0887	159.2747	180.3111	**0.0519**	**0.3457**	0.1971
$f14$	5	0.75	**0.2509**	**0.2583**	0.0039	0.2815	0.4025	0.0605	0.2522	0.2607	0.0037
$f15$	6	0.78	104.2304	1700.28	2015.04	894.6061	6069.7722	4689.5802	**24.6278**	**152.9817**	144.6451
$f16$	8	0.82	**7.3904**	196.810	171.7771	152.3790	502.9491	261.0805	7.9240	**75.2898**	59.3423
$f17$	8	0.79	-2.5859	42.0074	153.4414	87.9290	293.7373	104.7241	**-3.0161**	**-2.8355**	0.0967



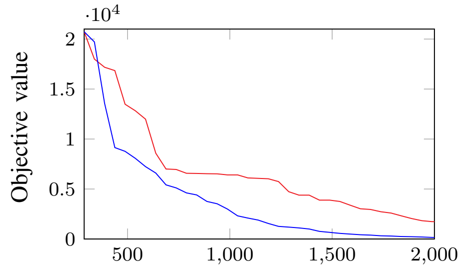
(a) $f11$ ($D = 3$)



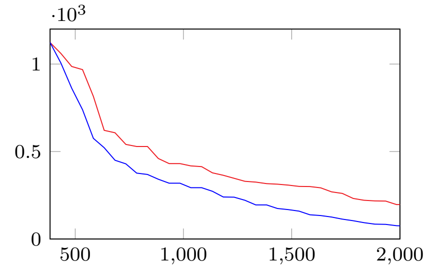
(b) $f13$ ($D = 4$)



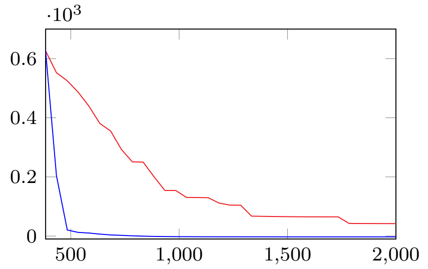
(c) $f14$ ($D = 5$)



(d) $f15$ ($D = 6$)



(e) $f16$ ($D = 8$)

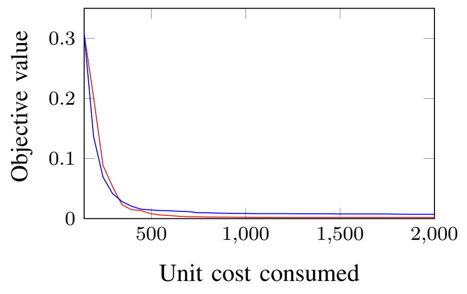


(f) $f17$ ($D = 8$)

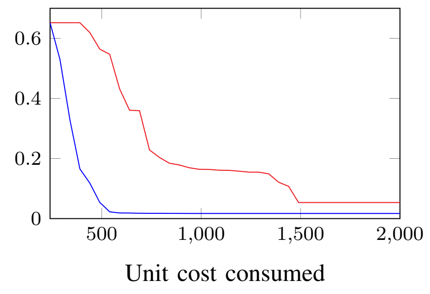
Fig. 5: Mean convergence plots for problem instances $f11$, $f13$, $f14$, $f15$, $f16$ and $f17$ of dataset A, over 30 runs.

TABLE II: Results on Griewank and Michalewicz test problems using Wang error functions 2 and 6 (indicated by subscript), comparing MFITS to MO^{2}TOS and the base-line co-kriging algorithm. Given are the number of decision variables (D), the square of the Pearson correlation coefficient (r^2), the best objective obtained, the mean best objective over the full set of runs (μ) and the corresponding standard deviation (σ).

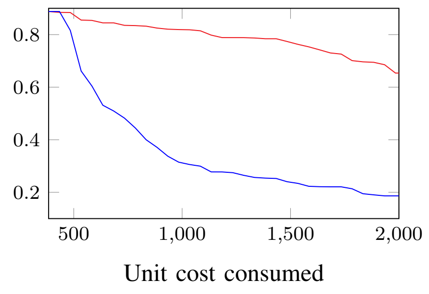
			Co-kriging			MO^{2}TOS			MFITS		
Instance	D	r^2	best	μ	σ	best	μ	σ	best	μ	σ
$Griewank_2$	3	0.73	**0**	**0.0018**	0.0083	0.0019	0.0735	0.0440	**0**	0.0072	0.0137
	5	0.54	**0**	0.0535	0.0723	0.1428	0.3886	0.1245	**0**	**0.0485**	0.1055
	8	0.37	**0**	0.6538	0.3363	0.5606	0.8036	0.1100	**0**	**0.1864**	0.2531
$Griewank_6$	3	0.74	**0**	**0.0056**	0.0114	0.0076	0.0564	0.0290	**0**	0.0104	0.0130
	5	0.76	**0**	0.0730	0.1145	0.1139	0.3792	0.1328	**0**	**0.0361**	0.0778
	8	0.76	**0**	0.4560	0.4238	0.5565	0.8108	0.1249	**0**	**0.2089**	0.3539
$Michalewicz_2$	3	0.77	-2.7239	-2.3582	0.2587	-2.4612	-1.8939	0.2366	**-2.7360**	**-2.5305**	0.1557
	5	0.73	-3.5164	-2.9470	0.2921	-3.2841	-2.4554	0.3928	**-3.5653**	**-2.9953**	0.3219
	8	0.64	-4.1813	-3.0214	0.4958	-4.2150	-3.0818	0.4406	**-4.5542**	**-3.3046**	0.5004
$Michalewicz_6$	3	0.76	-2.7194	-2.2412	0.2978	-2.4657	-1.8847	0.2927	**-2.7409**	**-2.4080**	0.2711
	5	0.83	**-3.6198**	-2.6695	0.3986	-3.2980	-2.5380	0.3328	-3.5511	**-2.9944**	0.3490
	8	0.86	-3.9978	-3.1032	0.4929	-3.9627	-3.1363	0.3772	**-4.2922**	**-3.1672**	0.4865



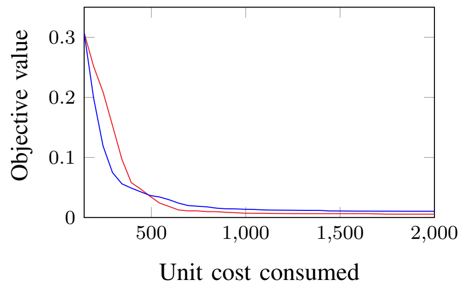
(a) $Griewank_2$ $(D = 3)$



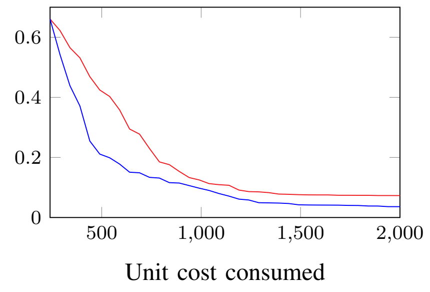
(b) $Griewank_2$ $(D = 5)$



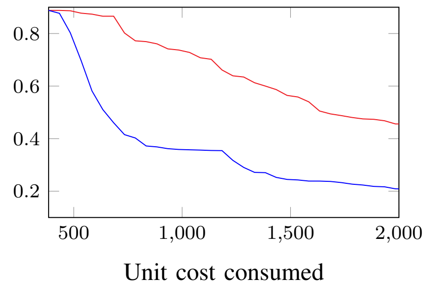
(c) $Griewank_2$ $(D = 8)$



(d) $Griewank_6$ $(D = 3)$



(e) $Griewank_6$ $(D = 5)$



(f) $Griewank_6$ $(D = 8)$

Fig. 6: Mean convergence plots for $Griewank$ problem instance of dataset B, over 30 runs.

TABLE III: Pairwise comparision of MFITS, MO^{2}TOS and the base-line co-kriging algorithm using the MWW test.

	Dataset A			Dataset B		
Algorithm	Win	Draw	Loss	Win	Draw	Loss
MFITS	**13**	2	1	**17**	5	2
Co-kriging	9	2	5	10	8	6
MO^{2}TOS	0	0	16	0	5	19

Although both algorithms contain the co-kriging technique as their principal constituent, the base-line algorithm uses a random sampling technique which gives it a disadvantage in problems with more decision variables, as it must cover an exponentially larger space with the same computational budget. By using the $LocalOCBA$ procedure to focus the sampling on areas of the search space that have been identified as promising, MFITS can use its budget more efficiently to exploit these regions, which is evidenced by the faster convergence rates for larger problem sizes.

The MO^{2}TOS algorithm performed worse than both MFITS and the base-line co-kriging algorithm for all problem instances. This is predominantly due to the limitations discussed in Section II and demonstrated in Figure 1. Because MO^{2}TOS can only select candidates for high-fidelity evaluation from the initial set of low-fidelity solutions, it requires a reasonably high density of coverage from initial sampling. In [18], up to 10,000 initial low-fidelity candidates were used for some problems which is well in-excess of the total computational budget of 2,000 low-fidelity-equivalent evaluations allowed in the experiments conducted here. As MO^{2}TOS must decide the finite set of solutions it can choose from *a priori*, it heavily relies on chance as to whether that initial set contains promising candidates or not. This is particularly evident in the results obtained for larger instances of test suite A, where the limited budget for initial samples must be spread over a high-dimensional space.

Finally, although the numerical results and convergence plots do indicate that there is not a significant difference between the performance of MFITS and the base-line co-kriging algorithm on the smaller instances; it is worth noting that for three out of four of these instances, the co-kriging baseline did perform slightly better on average. One possible explanation for this is that the $LocalOCBA$ procedure makes MFITS more greedy, and therefore more likely to become trapped in local optima — especially for multimodal instances like the Griewank problem. In these smaller instances, the random sampling of the base-line co-kriging algorithm is not as much of a disadvantage and it is more likely to come across good areas of the search space by chance.

VI. Conclusion and Future Work

Real-world engineering design problems often involve computationally intensive simulation-based analysis. In some cases, the fidelity of such simulations can be controlled i.e. a computationally less intensive and less accurate lower-fidelity analysis can be used instead of a computationally expensive and more accurate analysis.

In this paper a multi-fidelity simulation-based optimization algorithm is presented that can solve bound-constrained, single-objective problems called MFITS. It is based on an iterative, two-stage process and relies on commonly-used

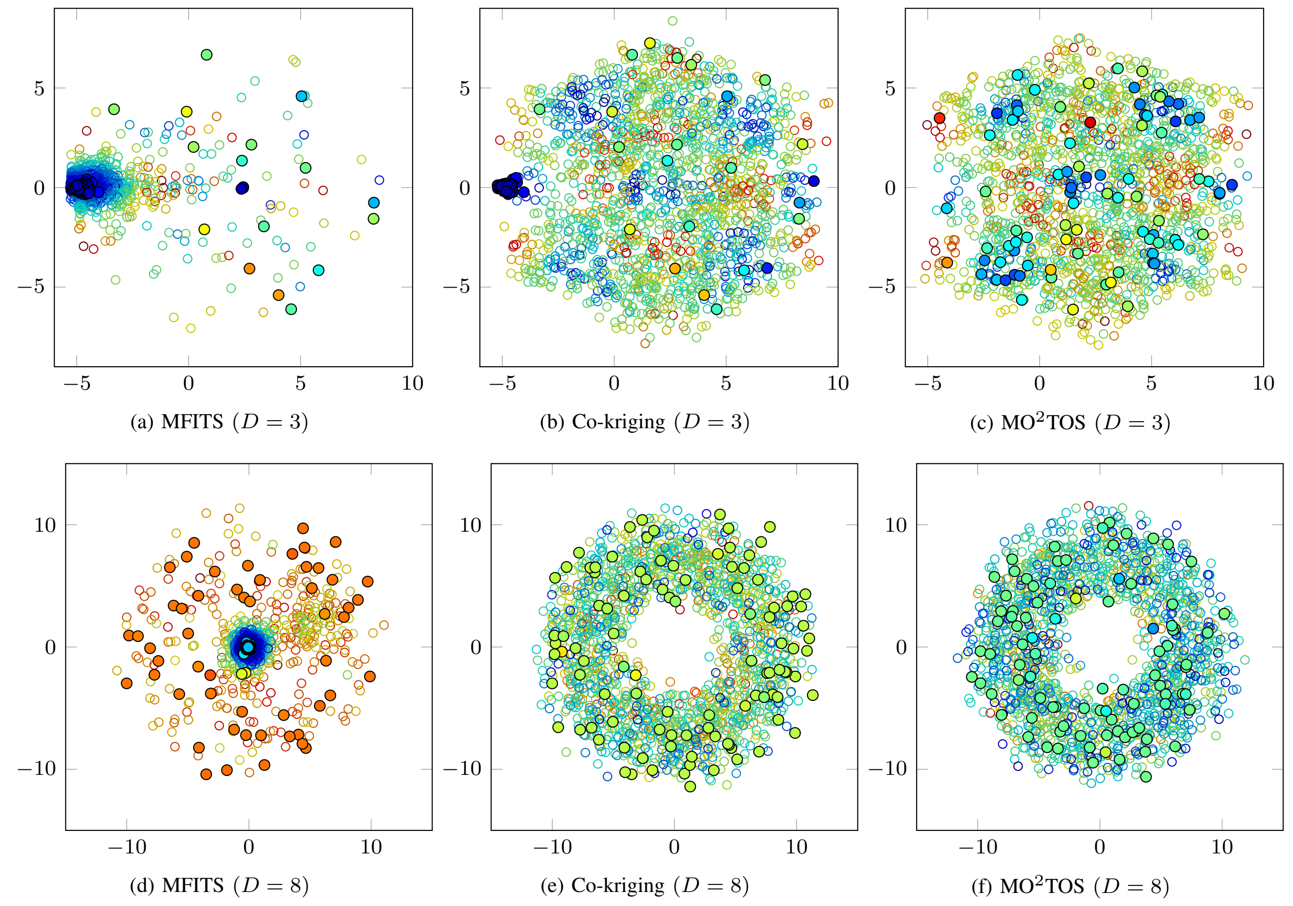


Fig. 7: Sammon mapping projection of samples onto the 2D plane for the Griewank function at $D = 3$ and $D = 8$. Low-fidelity samples are un-filled circles and high-fidelity samples are filled. Blue indicates higher quality solution and red indicates poorer quality.

co-kriging method coupled with a modified version of the optimal computing budget allocation algorithm that samples from a restricted neighbourhood. The key feature of the proposed approach is its means to enable a two-way information sharing. The information gained from the high-fidelity evaluations is used to determine the restricted neighbourhood from which the low-fidelity samples are selected. This allows for more efficient use of the computing budget as compared to contemporary approaches.

In order to demonstrate the effectiveness of the improved sampling technique of MFITS, numerical experiments were conducted on two separate test suites, one multi-fidelity test suite from the literature, and one test suite created by applying a generic method used to model simulation errors of two standard test functions. The results obtained by MFITS were compared with those from a base-line co-kriging and MO^{2}TOS algorithms. The experiments on the first suite demonstrated the versatility of MFITS on a variety of test functions, while those on the second test suite demonstrated the effects of problem size and error type on the performance of MFITS.

The results of the numerical experiments confirmed that MFITS produced solutions that were better than MO^{2}TOS and competitive with, or superior to, the base-line co-kriging algorithm across all instances for a limited computational budget. By sampling the search space more selectively when updating its surrogate model, MFITS was more efficient in its use of computing budget, resulting in faster convergence rates — especially for problems with larger number of variables.

Further research in this direction could include investigating the use of different global search techniques to optimize the co-kriging model and applying it to a wider range of test functions and real-world problems, including problems with integer and categorical decision variables, and with linear and non-linear constraints. Some of these directions are currently pursued by the authors.

Appendix A
Function Equations

Tables IV and V below give the equations for the test functions and error functions, respectively.

TABLE IV: Test functions used for experiments in this paper. Given are the instance name, the function, the number of decision variables (D), the bounds (B) and the reference the test function is taken from.

Instance	Function	D	B	Ref.
$f10$	$f_H(\boldsymbol{x}) = 100\left(\exp\left(-\frac{2}{x_1^{1.75}}\right) + \exp\left(-\frac{2}{x_2^{1.5}}\right) + \exp\left(-\frac{2}{x_3^{1.25}}\right)\right)$ $f_L(\boldsymbol{x}) = 100\left(\exp\left(-\frac{2}{x_1^{1.75}}\right) + \exp\left(-\frac{2}{x_2^{1.5}}\right)\right)$	3	$[0,1]^D$	[14]
$f11$	$f_H(\boldsymbol{x}) = 4\left(x_1 - 2 + 8x_2 - 8x_2^2\right)^2 + (3-4x_2)^2 + 16\sqrt{x_3+1}\,(2x_3-1)^2$ $f_L(\boldsymbol{x}) = 4\left(x_1 - 2 + 8x_2 - 8x_2^2\right)^2 + (3-4x_2)^2 + 5\sqrt{x_3+1}\,(2x_3-1)^2$	3	$[0,1]^D$	[14]
$f12$	$f_H(\boldsymbol{x}) = -\sum_{i=1}^{m}\left(\sum_{j=1}^{4}(x_j - C_{ji})^2 + \beta_i\right)^{-1}$ $f_L(\boldsymbol{x}) = -\sum_{i=1}^{m}\left(\sum_{j=1}^{4}(x_j - C_{ji})^2 + 0.9\beta_i\right)^{-1}$ $m = 10, \quad \beta = \frac{1}{10}(1,2,2,4,4,6,3,7,5,5)^T$ $C = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}$	4	$[0,10]^D$	[14]
$f13$	$f_H(\boldsymbol{x}) = (x_1-1)^2 + \sum_{i=2}^{d} i\left(2x_i^2 - x_{i-1}\right)^2$ $f_L(\boldsymbol{x}) = (x_1-1)^2 + x_2^4 + 4x_3^4 + 4x_4^4$	4	$[-10,10]^D$	[14]
$f14$	$f_H(\boldsymbol{x}) = \sum_{i=1}^{5}\left(0.3 + \sin\left(\frac{16}{15}x_i - 1\right) + \sin\left(\frac{16}{15}x_i - 1\right)^2\right)$ $f_L(\boldsymbol{x}) = \sum_{i=1}^{5}\left(0.3 + \sin\left(\frac{13}{15}x_i - 1\right) + \sin\left(\frac{13}{15}x_i - 1\right)^2\right)$	5	$[-1,1]^D$	[14]
$f15$	$f_H(\boldsymbol{x}) = \sum_{i=1}^{5}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i-1)^2\right)$ $f_L(\boldsymbol{x}) = \sum_{i=1}^{5}\left(100\left(x_{i+1} - x_i^2\right)^2 + 4(x_i-1)^4\right)$	6	$[0,1]^D$	[14]
$f16$	$f_H(\boldsymbol{x}) = \sum_{i=1}^{2}\left[\begin{matrix}(4x_{4i-3} - 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + \\ (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^2\end{matrix}\right]$ $f_L(\boldsymbol{x}) = \sum_{i=1}^{2}\left[\begin{matrix}(4x_{4i-3} - 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + \\ (x_{4i-2} - 2x_{4i-1})^4 + 4(x_{4i-3} - x_{4i})^2\end{matrix}\right]$	8	$[-4,5]^D$	[14]
$f17$	$f_H(\boldsymbol{x}) = \sum_{i=1}^{8}\left(x_i^4 - 16x_i^2 + 5x_i\right)$ $f_L(\boldsymbol{x}) = \sum_{i=1}^{8}\left(0.8x_i^4 - 16x_i^2 + 5x_i\right)$	8	$[-5,5]^D$	[14]
Griewank	$f_H(\boldsymbol{x}) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$\{3,5,8\}$	$[-5,5]^D$	[30]
Michalewicz	$f_H(\boldsymbol{x}) = \sum_{i=1}^{D}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right), \quad m = 10$	$\{3,5,8\}$	$[0,\pi]^D$	[31]

TABLE V: Error functions used, as defined in Wang et al. [24]. Function e_2 models resolution errors and e_6 models stochastic errors. Here, $\phi \in [0, 10000]$ determines the fidelity level and D is the number of decision variables.

Name	Function
e_2	$e_2(\boldsymbol{x},\phi) = \sum_{i=1}^{D} a(\phi)\cos\left(w(\phi)x_i + b(\phi) + \pi\right)$ where, $a(\phi) = \theta(\phi),\ w(\phi) = 10\pi\theta(\phi),\ b(\phi) = 0.5\pi\theta(\phi),\ \theta(\phi) = e^{-0.00025\phi}$
e_6	$e_6 = \mathcal{N}\left(\mu(\boldsymbol{x},\phi), \sigma(\phi)\right)$ where, $\mu^2(\phi) = 0,\ \sigma(\phi) = 0.1\vartheta(\phi),\ \vartheta(\phi) = e^{-0.0005\phi}$

References

[1] A. Forrester, A. Sobester, and A. Keane, *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[2] Y. Jin and B. Sendhoff, "A systems approach to evolutionary multi-objective structural optimization and beyond," *IEEE Computational Intelligence Magazine*, vol. 4, no. 3, pp. 62–76, 2009.

[3] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury, "Simulation optimization: a review of algorithms and applications," *Annals of Operations Research*, vol. 240, no. 1, pp. 351–380, 2016.

[4] J. Branke, M. Asafuddoula, K. S. Bhattacharjee, and T. Ray, "Efficient use of partially converged simulations in evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 52–64, 2016.

[5] D. J. Toal, "Some considerations regarding the use of multi-fidelity kriging in the construction of surrogate models," *Structural and Multidisciplinary Optimization*, vol. 51, no. 6, pp. 1223–1245, 2015.

[6] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Proceedings of the royal society a: mathematical, physical and engineering sciences*, vol. 463, no. 2088, pp. 3251–3269, 2007.

[7] M. C. Kennedy and A. O'Hagan, "Predicting the output from a complex computer code when fast approximations are available," *Biometrika*, vol. 87, no. 1, pp. 1–13, 2000.

[8] J. Laurenceau and P. Sagaut, "Building efficient response surfaces of aerodynamic functions with kriging and cokriging," *AIAA journal*, vol. 46, no. 2, pp. 498–507, 2008.

[9] L. Huang, Z. Gao, and D. Zhang, "Research on multi-fidelity aerodynamic optimization methods," *Chinese Journal of Aeronautics*, vol. 26, no. 2, pp. 279–286, 2013.

[10] P. Perdikaris, D. Venturi, J. O. Royset, and G. E. Karniadakis, "Multi-fidelity modelling via recursive co-kriging and gaussian–markov random fields," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 471, no. 2179, p. 20150018, 2015.

[11] X. Yang, D. Barajas-Solano, G. Tartakovsky, and A. M. Tartakovsky, "Physics-informed cokriging: A gaussian-process-regression-based multifidelity method for data-model convergence," *Journal of Computational Physics*, vol. 395, pp. 410–431, 2019.

[12] R. Giraldo, L. Herrera, and V. Leiva, "Cokriging prediction using as secondary variable a functional random field with application in environmental pollution," *Mathematics*, vol. 8, no. 8, p. 1305, 2020.

[13] M. Goulard and M. Voltz, "Geostatistical interpolation of curves: a case study in soil science," in *Geostatistics Tróia'92*. Springer, 1993, pp. 805–816.

[14] L. Lv, C. Zong, C. Zhang, X. Song, and W. Sun, "Multi-fidelity surrogate model based on canonical correlation analysis and least squares," *Journal of Mechanical Design*, vol. 143, no. 2, p. 021705, 2021.

[15] A. Ariyarit and M. Kanazaki, "Multi-fidelity multi-objective efficient global optimization applied to airfoil design problems," *Applied Sciences*, vol. 7, no. 12, p. 1318, 2017.

[16] A. Hebbal, L. Brevault, M. Balesdent, E.-G. Talbi, and N. Melab, "Multi-fidelity modeling with different input domain definitions using deep gaussian processes," *Structural and Multidisciplinary Optimization*, vol. 63, no. 5, pp. 2267–2288, 2021.

[17] K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, and J. González, "Deep gaussian processes for multi-fidelity modeling," *arXiv preprint arXiv:1903.07320*, 2019.

[18] J. Xu, S. Zhang, E. Huang, C.-H. Chen, L. H. Lee, and N. Celik, "Mo2tos: Multi-fidelity optimization with ordinal transformation and optimal sampling," *Asia-Pacific Journal of Operational Research*, vol. 33, no. 03, p. 1650017, 2016.

[19] C.-H. Chen and L. H. Lee, *Stochastic simulation optimization: an optimal computing budget allocation*. World scientific, 2011, vol. 1.

[20] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff, "Evolutionary optimization with dynamic fidelity computational models," in *International Conference on Intelligent Computing*. Springer, 2008, pp. 235–242.

[21] D. E. Bryson and M. P. Rumpfkeil, "Multifidelity quasi-newton method for design optimization," *AIAA Journal*, vol. 56, no. 10, pp. 4074–4086, 2018.

[22] L. W. Ng and K. E. Willcox, "Multifidelity approaches for optimization under uncertainty," *International Journal for numerical methods in Engineering*, vol. 100, no. 10, pp. 746–772, 2014.

[23] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[24] H. Wang, Y. Jin, and J. Doherty, "A generic test suite for evolutionary multifidelity optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 836–850, 2017.

[25] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[26] J. Xu, S. Zhang, E. Huang, C.-H. Chen, L. H. Lee, and N. Celik, "An ordinal transformation framework for multi-fidelity simulation optimization," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2014, pp. 385–390.

[27] T. Takahama and S. Sakai, "Efficient constrained optimization by the ε constrained adaptive differential evolution," in *IEEE congress on evolutionary computation*. IEEE, 2010, pp. 1–8.

[28] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[29] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[30] A. O. Griewank, "Generalized descent for global optimization," *Journal of optimization theory and applications*, vol. 34, no. 1, pp. 11–39, 1981.

[31] Z. Michalewicz, *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 2013.

[32] K. Leonard and J. R. Peter, "Finding groups in data: an introduction to cluster analysis," *Liu, H., Mroz, TA, and Van der Klaauw, W.(2010). Maternal employment, migration, and*, 1990.

[33] I. Couckuyt, T. Dhaene, and P. Demeester, "oodace toolbox: A flexible object-oriented kriging implementation," *Journal of Machine Learning Research*, vol. 15, pp. 3183–3186, 2014.