Bare Demo of IEEEtran.cls for IEEE Journals

Angus Kenny, Tapabrata Ray, and Hemant Kumar Singh

***Abstract*—The abstract goes here.**

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

I. Introduction

MANY real-world engineering design problems are intractable to analytical methods, and the production and testing of a prototype model can be probitively expensive, in terms of both time and cost. In many other applications, such models are not even feasible. For these types of problem instances, numerical simulations — such as finite element analysis (FEA) and computational fluid dynammics (CFD) — can be employed to predict the outcome of various design choices. Simulation optimisation (SO) is an important tool which allows large design spaces to be searched, the merits of various candidate solutions to be approximated and inferences to be made about how those candidates might be improved — all relatively cheaply.

The principle behind SO is to iteratively use historical information from previous numerical simulations to construct a surrogate model (such as a kriging model) of the random field representing the fitness of a solution, with respect to its input variables. This model is searched using some optimisation technique, to find a new candidate — or set of candidates — which the model predicts will produce a good outcome when simulated. The data from this new simulation is incorporated into the surrogate model and the process continues.

Although conducting a numerical simulation is not usually as expensive as producing and testing a physical prototype, it does incur a significant cost; with some complex FEA or CFD simulations taking several days to complete. Often, the allowed computational budget is given as a fixed amount, therefore it is important to ensure that this budget is used as efficiently as possible. In order to address this concern, the field of multi-fidelity simulation optimisation (MFSO) was developed.

Some numerical simulation processes allow control over the resolution, or fidelity, of the data they produce. For example, in FEA simulations this can be controlled by the mesh density, and in CFD simulations it can be controlled by the number of iterations the simulation is run for. While these lower-fidelity simulations are cheaper to produce, there is a trade-off in the accuracy of their predictions. However, by combining information from both high- and low-fidelity simulations, MFSO methods can produce good quality approximations for a much lower computational cost.

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.

Despite being a relatively nascent field of research, there are a number of different approaches to MFSO in the literature. Many of these approaches are based on Forrester's co-kriging technique which correlates the two sets of data, cheap and expensive, to produce a single prediction model. The principal way in which the different methods in this category deviate from each other, is in how they collect this data and also how they search the model for potential candidates.

Other than these co-kriging-based techniques, there are several other approaches taken by researchers, of which the following is not an exhaustive list: Lv et al. employ a canonical correlation analysis-based model, in which the least squares method is used to determine optimal parameters; Ariyarit and Kanazaki use a hybrid method which employs a kriging model to estimate local deviations and a radial basis function to approximate the global model; and, Xu et al. use a two-stage process which first uses ordinal transformation to transform the original multi-dimensional design space into a one-dimensional ordinal space and then samples from this ordinal space using a method based on the optimal computational budget allocation (OCBA) algorithm.

This paper proposes an iterative two-stage MFSO process called $InsertName$ for bound-constrained, single-objective multi-fidelity optimisation problems. It uses previously obtained information about promising areas of the search space to define a restricted neighbourhood using a guided differential evolution (DE) process on a kriging model of the low-fidelity data. This neighbourhood is then sampled from and searched, using a method derived from OCBA, to determine a set of candidates to 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 data, which is searched globally using DE to find a suitable candidate for high-fidelity simulation. Finally, this high-fidelity data is 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, $InsertName$ allows two-way information sharing between the sets of data — this is in contrast to

most other similar techniques, which either feature one-way information sharing, or treat the sets of data as completely independent of one another.

The performance of the $InsertName$ model is compared against a baseline co-kriging-based MFSO algorithm on two separate datasets. 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 standard test functions using the methods described in the paper by Wang et al. In addition to this, some important properties of $InsertName$ are also investigated.

The remainder of this paper is organised as follows. Section II provides the fundamentals and background of the proposed model, along with a description of the type of problems tackled and related work. The $InsertName$ algorithm is detailed in Section III, describing all of its constituent parts and detailing some similarities and differences with a related technique from the literature. Experimental design and datasets are discussed in Section IV, with Section V giving the results of these experiments and a discussion of their implications. Finally, Section VI provides the conclusion and outlines any future directions of research.

II. Background

This section details the type of problem $InsertName$ is designed to address; provides information about two critical components that are used in its construction; and gives a brief summary of another method which is similar to $InsertName$ from the literature.

A. Problem type

The algorithm presented here is designed to address bound-constrained, single-objective, multi-fidelity optimisation problems with continuous variables. Here, the terms bound-constrained and continuous imply that a variable may have an upper- or lower-bound which it cannot exceed, but that it may take any value (precision issues notwithstanding) between these bounds in the decision space; and that there are no regions in the objective space which are forbidden.

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[1] $\boldsymbol{x}$ such that $l_i \leq x_i \leq u_i$ for all $i \in [D]$, where $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^D$ are the lower- and upper-bounds, respectively[2]. The *objective value* of a 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 value is minimised, the so-called *optimal* solution to P is $\boldsymbol{x}^*$, such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \prod_{i=1}^{D}[l_i, u_i]$, although it may be maximised without loss of generality.

Examples of these types of problems abound in the field of engineering design [**?**], [**?**], [**?**] and often they require some form of numerical simulation to compute the objective value of their solutions. This numerical simulation can be very computationally expensive, therefore researchers often employ artificial test functions when developing algorithms. While not generally being an accurate model for the behaviour of real-world problems, test functions are very fast to evaluate and can be customised to gauge the performance of algorithms on a variety of fitness landscapes.

One advantage of numerical simulation is that the fidelity of evaluated objective values can often be controlled, trading accuracy of the simulation for computational budget [**?**]. This effect is difficult to model using test functions as a low-fidelity function must still have some correlation to the true function in order to be useful, but also must contain enough noise to obfuscate its behaviour. The experiments detailed in Section IV use datasets generated by two different methods from the literature.

The dataset in Lv et al. [1] uses the customised addition, removal and modification of terms to produce a transformed function with a desired correlation to the original. For example, the pair of high- and low-fidelity functions

$$f_H(\boldsymbol{x}) = \sum_{i=1}^{2}\left(x_i^4 - 16x_i^2 + 5x_i\right)\,,$$
$$f_L(\boldsymbol{x}) = 0.5\sum_{i=1}^{2}\left(x_i^4 - 16x_i^2\right)\,,$$

have a correlation coefficient of 0.79 over $\boldsymbol{x} \in [0,1]^2$. 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. [2] analysed the behaviour of many numerical simulations under different fidelity conditions and identified the following three types of errors:

1) *Resolution errors:* The landscape of resolution errors is multimodal due to the inconsistency of global and local errors. It is deterministic but gets smoother when the fidelity level increases.
2) *Stochastic errors:* The errors are stochastic, i.e., the fitness value of the same solution varies in different simulation runs. The stochastic error decreases as the fidelity level increases.
3) *Instability errors:* Representing failed simulations. A failure happens at a certain probability, and the probability decreases as the fidelity level increases.

From this, they formulated a generic transformation function to turn any test function f of the form in Equation 1 into a

[1]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.

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

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. Each models one of the identified errors, and are all 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 at all; and because 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.

B. Kriging and co-kriging

Originally arising from geostatistical methods used for ore valuation in mining research — but since being applied across a number of domains — is the so-called *kriging* method [3]. Kriging is an interpolation technique that uses a limited set of sampled data 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 this sample data is the Gaussian process

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

where, $f(\boldsymbol{x})$ is a polynomial regression function based on the sample data $\boldsymbol{y}$ and encapsulates the main variations in the data. 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 $Y(\boldsymbol{x})$.

The correlation matrix Ψ of $Y(\boldsymbol{x})$ is

$$\text{cor}[Y(\boldsymbol{x}^i), Y(\boldsymbol{x}^l)] = \exp\left(-\sum_{j=1}^{D} \theta_j \mid x_j^i - x_j^l \mid^{p_j} \right), \tag{4}$$

where p_j is a smoothness parameter (typically $p_j \in [1, 2]$) and θ_j determines how much influence decision variable j has for a given sample point. Both $\boldsymbol{p}$ and $\boldsymbol{\theta}$ have size D and are usually determined using maximum likelihood estimation (MLE). Given the set of sample points X and this correlation matrix Ψ, the predicted value $Y(\boldsymbol{x}')$ can be computed using Equation 3, employing regression to find $f(\boldsymbol{x}')$ and by interpolation on Ψ to find $Z(\boldsymbol{x}')$.

Low-fidelity data is typically much cheaper to produce than high-fidelity data, a fact which the *co-kriging* technique exploits to great effect. Adapted from Kennedy and O'Hagan's autoregressive model [4], co-kriging correlates multiple sets of data with different fidelities to produce a single model that approximates the high-fidelity data. 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 lowest fidelity is reached. Because of this, the Markov property must be assumed, that 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[3] 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 data is constructed according to Equation 3. Using this, the high-fidelity model is

$$Y_H(\boldsymbol{x}) = \rho(\boldsymbol{x})\mu_L(\boldsymbol{x}) + Z_d(\boldsymbol{x}), \tag{5}$$

where $\rho(\boldsymbol{x})$ is a scaling factor, determined as part of the 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 $\rho(\boldsymbol{x})\mu_L(\boldsymbol{x})$.

A more in-depth mathematical treatment of kriging and co-kriging can be found in [3]–[6].

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 for a design, μ, is unknown, but it can be estimated by its sample mean $\hat{\mu}$. This estimation becomes more accurate with an increased volume of replications; however, performing more replications consumes finite computing resources. Therefore, in the interest of efficiency, it is best to concentrate efforts on simulating promising designs.

Given a set of k candidate designs and a finite computing budget T, the optimal computing budget allocation (OCBA) method aims to find an allocation such that $N_1 + N_2 + \cdots + N_k = T$, 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 $i \in [k]$.

The probability that design b actually *is* the the best design is called the probability of correct selection (PCS),

$$PCS = P\{\tilde{\mu}_b < \tilde{\mu}_i, i \neq b\}, \tag{6}$$

where $\tilde{\mu}_i \sim \mathcal{N}\left(\hat{\mu}_i, \frac{\sigma_i^2}{N_i}\right)$ is the posterior distribution of the unknown mean of design i, μ_i.

The PCS can be estimated using Monte Carlo simulation, but this is time-consuming; therefore, the following problem

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

is formulated in [**?**] to compute the approximate probability of correct selection (APCS):

$$\begin{aligned} \underset{N_1,\ldots,N_k}{\text{maximise}} \quad & 1 - \sum_{i=1, i\neq b}^{k} P\{\tilde{\mu}_b < \tilde{\mu}_i\}\,, \\ \text{such that} \quad & \sum_{i=1}^{k} N_i = T\,, \quad N_i \geq 0, \end{aligned} \tag{7}$$

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

$$\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2, \tag{8}$$

$$N_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{N_i^2}{\sigma_i^2}}\,, \tag{9}$$

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 $i \neq j \neq b$.

The implications of Equations 8 and 9 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 increased volume of replications will help to address.

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

Algorithm 1: $OCBA$

Input: k, number of designs; T, computing budget; Δ, replications per update; n_0, initial replications.
Output: b, index of best design.

$b \leftarrow \emptyset$ {Initialise b}
$N_i \leftarrow n_0\,,\ \forall i \in [k]$ {Count initial replications}
$\boldsymbol{S}^i \leftarrow Sim(n_0)\,,\ \forall i \in [k]$ {Perform initial replications}
while $\sum_{i\in[k]} N_i < T$ **do**
  $\boldsymbol{N}' \leftarrow \boldsymbol{N}$ {Store old allocations}
  $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}} \leftarrow Stats(\boldsymbol{S}),\ \forall i \in [k]$ {Compute sample statistics}
  $b \leftarrow \text{argmin}_i\ \hat{\mu}_i,\ \forall i \in [k]$ {Update b}
  $\boldsymbol{N} \leftarrow Allocate(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}, \boldsymbol{N}, \Delta)$ {Allocate Δ replications}
  $S_i \leftarrow Sim(N_i - N_i'),\ \forall i \in [k]$ {Perform additional simulations}
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 8 and 9.

D. MO^2TOS

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 optimisation with ordinal transformation and optimal sampling (MO^2TOS) [7] framework is a two-stage algorithm that addresses this consideration, by combining ideas from OCBA with ordinal transformation [8].

Given a problem instance P and a finite computing budget T, the MO^2TOS 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 are evaluated in low-fidelity and then ranked by their evaluated value. 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 proximate to each other in the transformed objective space, as opposed to the original design space. As a result, regardless of where 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 group has n_0 solutions selected — without replacement — and evaluated in high-fidelity. These results are used to determine the sample statistics of the groups. Equations 8 and 9 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, while the total budget consumed is less than T.

The value of Δ is typically quite small, as there may be significant bias between the low- and high-fidelity evaluated data. 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 intial partitioning.

III. $InsertName$ Algorithm

The multi-fidelity optimisation algorithm proposed in this paper is an iterative two-stage process, which maintains two separate surrogate models that share information between them. The first surrogate is a kriging model of the low-fidelity data. This model is searched to find the best potential candidates, within a restricted region, to evaluate with low-fidelity. The information from this search is used to update the first model and a co-kriging model that combines both low- and high-fidelity data to approximate the high-fidelity objective function. This second model is globally searched to determine a suitable candidate for high-fidelity evaluation, and this data is used to update the co-kriging model and to determine the neighbourhood for the next search of the low-fidelity surrogate. Algorithm 2 gives a description of this process, and the functions within are defined in the following subsections.

Algorithm 2: $InsertName$

Input: P, problem data; $N_{L_{max}}$, max size of low-fidelity population; $N_{e_{max}}$, maximum number of evaluations; ρ, parameter set for LocalOCBA.
Output: Best solution found $\boldsymbol{x}^{\beta}$.

$X_v \leftarrow LHS(P),\ \forall v \in \{L, H\}$ {Generate initial populations}
$A_v, N_e \leftarrow f_v(X_v, 0),\ \forall v \in \{L, H\}$ {Evaluate initial populations}
$\boldsymbol{x}^{\beta} \leftarrow \emptyset$ {Initialise $\boldsymbol{x}^{\beta}$}
while $N_e < N_{e_{max}}$ **do**
  $M_L \leftarrow Krige(A_L)$ {Update low-fidelity kriging model}
  $M_C \leftarrow CoKrige(A_L, A_H)$ {Update co-kriging model}
  $\boldsymbol{x} \leftarrow GlobalSearch(M_C)$ {Globally search co-kriging model}
  $\boldsymbol{\alpha}, N_e \leftarrow f_H(\boldsymbol{x}, N_e)$ {Evaluate and update total cost}
  $A_H \leftarrow A_H \cup \{\boldsymbol{\alpha}\}$ {Add $\boldsymbol{\alpha}$ to high-fidelity archive}
  $\boldsymbol{x}^{\beta} \leftarrow \min(f_H(\boldsymbol{x}^{\beta}), f_H(\boldsymbol{x}))$ {Update best solution}
  $\epsilon \leftarrow Sigmoid(N_e, N_{e_{max}})$ {Determine size of neighbourhood}
  $A_L, N_e \leftarrow LocalOCBA(\rho, M_L, A_L, \boldsymbol{x}^{\beta}, \epsilon)$ {Locally search low-fidelity model}
  if $|A_L| > N_{L_{max}}$ **then**
    $A_L \leftarrow Winnow(A_L, N_{L_{max}})$ {Control population size}
  end if
end while

A. Initialisation, evaluation and surrogate models

The function $LHS(P)$ returns an initial population sampled from a latin hypercube with size $6D$ for high-fidelity and $18D$ for low-fidelity, where D is the number of decision variables defined in the problem data P. The bounds of the hypercube are also defined in P. Latin hypercube sampling is used in order to start with as broad a picture of the search space as possible. Solutions are evaluated using the function $f_v(X, N_e)$, 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, N_e.

Two surrogate models are used, $Krige(A_L)$ takes the low-fidelity data and returns a kriging model approximating it, and $CoKrige(A_L, A_H)$ takes both the low- and high-fidelity archive data and returns a co-kriging model that approximates the fitness landscape of the high-fidelity data. The function $GlobalSearch(M_C)$ takes the co-kriging model as input and returns the best solution that can be found using some global search method. This output solution is evaluated in high-fidelity, the information from which is used to update the co-kriging model, and also to determine the restricted neighbourhood used to locally search the low-fidelity kriging model.

B. Restricted neighbourhood

The correlation between the low- and high-fidelity data can vary wildly 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 this way allows 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.

In order to restrict the region of interest, a so-called *guided DE* process is employed. In differential evolution (DE), assuming complete recombination occurs, a candidate child solution x_c is produced from three parent solutions $\boldsymbol{x}^1$, $\boldsymbol{x}^2$ and $\boldsymbol{x}^3$ with the following formula:

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

with F being a constant called the mutation factor. This is simply scaled vector addition, therefore this produced child $\boldsymbol{x}^c$ can be "nudged" further towards a reference point (in this case x_β) by a similar process:

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

where $\boldsymbol{G}$ is the so-called *guide factor*. This is illustrated in Figure 1.

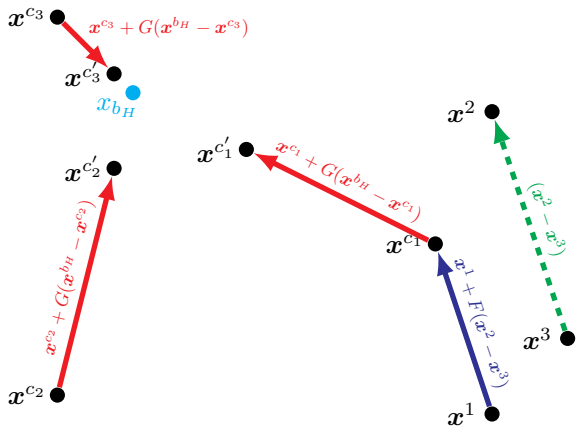

Fig. 1: The DE recombination process can be "guided" towards the best high-fidelity solution found so far.

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 translated using Equation 11 to produce $\boldsymbol{x}^{c_1'}$. Child 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 equation. If this process is continued, the result is a "cloud" of candidate solutions in the vicinity of $\boldsymbol{x}^{\beta}$, the centre of the region of interest.

The guide factor determines how far the resulting solution is translated in the direction of $\boldsymbol{x}^{\beta}$. At the beginning of the search, $\boldsymbol{x}^{\beta}$ is determined by optimising a very coarse model built with sparse information, and cannot be trusted to be indicative of 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. One function which exhibits these properties is the logistic sigmoid curve

$$\epsilon = \frac{L}{1 + e^{-k(x - x_0)}}\,, \tag{12}$$

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 "ramps up". The function $Sigmoid(N_e, N_{e_{max}})$ uses the total cost and maximum cost to compute this ϵ value. Figure 2 gives a plot of the logistic sigmoid curve with $L = 0.99$, $k = 10$ and $x_0 = 0.2$. Using

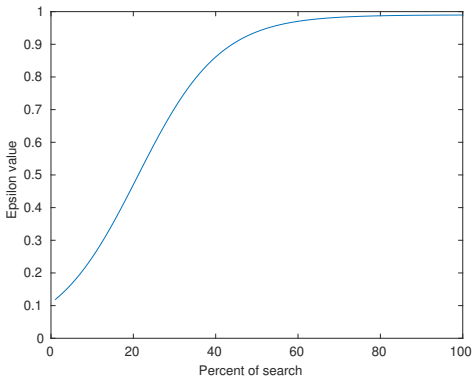

Fig. 2: The logistic sigmoid curve flattens at around 80%.

this value, $\boldsymbol{g}$ can be computed as:

$$\boldsymbol{g} = \epsilon + (1 - \epsilon)\boldsymbol{r}\,, \tag{13}$$

where $\boldsymbol{r} \in [0, 1]^D$ is a random vector with D components, making $\boldsymbol{g} \in [\epsilon, 1]^D$ also a random vector. This ensures 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, without adding any extra parameters.

C. LocalOCBA

Once the candidate solutions have been been generated in the vicinity of $\boldsymbol{x}^\beta$, they must be selected from. As already stated, the goal of evaluating solutions in low-fidelity is not to optimise 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 allocating computing resources). This principle can be applied here, with some modifications, as described in Algorithm 3.

Here, the $GuidedDE(A_L, \boldsymbol{x}^\beta, \epsilon)$ 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 some model M. The solutions are ranked and partitioned using a 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 a diversity of solutions will be selected, while still preferencing the more promising candidates. The function

Algorithm 3: *LocalOCBA*

Input: $\rho = \{\Delta_1$, total to be sampled; Δ_2, samples per statistics update$\}$; M, low-fidelity model; A_L, low-fidelity archive; $\boldsymbol{x}^\beta$, best high-fidelity solution ; ϵ, sigmoid value; N_e, total cost incurred.
Output: A_L, updated low-fidelity archive; N_e, updated total cost.

$X \leftarrow GuidedDE(A_L, \boldsymbol{x}^\beta, \epsilon)$ {Generate child population}
$A_X \leftarrow f_M(X)$ {Approximate each child by model M}
$G, k \leftarrow Partition(A_X)$ {Partition solutions}
$S_i \leftarrow \emptyset,\ \forall i \in [k]$ {Empty groups for selected solutions}
while $\sum_{i \in [k]} |S_i| < \Delta_1$ **do**
  $\hat{\mu}_i, \hat{\sigma}_i \leftarrow$ sample statistics for $S_i, \forall i \in [k]$ (if $|S_i| < 2$, use G_i)
  $R \leftarrow GetRatios(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})$ {compute allocation ratios}
  $D \leftarrow Allocate(R, S, G) : \sum_{i \in [k]} |D_i| = \Delta_2$ {Allocate Δ_2 solutions according to ratios R}
  $D_i, N_e \leftarrow f_L(D_i, N_e),\ \forall i \in [k]$ {Evaluate allocated solutions}
  $S_i \leftarrow S_i \cup D_i,\ \forall i \in [k]$ {Add to selected solutions}
  $G_i \leftarrow G_i \setminus D_i,\ \forall i \in [k]$ {Selection without replacement}
end while
$A_L \leftarrow A_L \cup \bigcup_{i \in [k]} S_i$ {Combine all selected solutions}

$Partition(X)$ returns a set of solution groups G and the number of groups k.

OCBA principles are used to select — without replacement — 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, they are added to the low-fidelity archive, and the updated archive is returned.

D. Population 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 population size $N_{L_{max}}$ should be 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 $N_{L_{max}}$ 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 very detrimental.

The function $Winnow(A_L, N_{L_{max}})$ takes an archive and a maximum size and returns a winnowed archive with exactly $N_{L_{max}}$ solutions. It does this by partitioning the solutions — in the decision space — into $N_{L_{max}}$ different groups using a clustering algorithm, such as k-means clustering. Most of these clusters will contain only one solution

which is 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 $N_{L_{max}}$ solutions, but diversity is maintained throughout the population.

E. Similarities and differences to MO^2TOS

There exist some similarities between $InsertName$ and the MO^2TOS 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 $InsertName$ is an iterative process, whereas MO^2TOS is a two-step algorithm which is only run through one time. As MO^2TOS only performs a single iteration, it cannot use any prior knowledge to determine where it should concentrate its resources when evaluating the initial low-fidelity population. Therefore, it must evaluate uniformly across the whole search space, and subsequently expend computational budget in areas which are not beneficial to the search. In contrast, $InsertName$ uses information from previous iterations, to identify promising areas of the search space that it can exploit.

Another difference is that MO^2TOS 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. $InsertName$ uses its ranking and selection phases in order to select from a neighbourhood of solutions that have been identified in a promising region 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 perfomed 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 optimise the low-fidelity objective function, just select from a set of ranked candidates.

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

Algorithm 4: Base-line co-kriging multi-fidelity optimisation algorithm

Input: D, number of variables for problem;$N_{L_{max}}$, max size of low-fidelity population $N_{e_{max}}$, maximum low-fidelity evaluations; Δ, new low-fidelity solutions added to archive per iteration.
Output: Best solution found $\boldsymbol{x}^\beta$.

$A_{v_1} \leftarrow LHS(v_2),\ \forall v \in \{(L, 18D), (H, 6D)\}$ {Generate initial populations using LHS}
$\boldsymbol{x}^\beta \leftarrow \emptyset$ {Initialise x_β}
while $N_e < N_{e_{max}}$ **do**
  $A_L \leftarrow A_L \cup LHS(\Delta)$ {Add Δ new solutions to low-fidelity archive}
  if $|A_L| > N_{L_{max}}$ **then**
    $A_L \leftarrow winnow(A_L, N_{L_{max}})$ {If population is too big, then winnow to required size}
  end if
  $M_C \leftarrow CoKrige(A_L, A_H)$ {Update co-kriging model}
  $\boldsymbol{x} \leftarrow RunEA(M_C)$ {Run EA on co-kriging model to get best solution}
  $A_H \leftarrow A_H \cup \{\boldsymbol{x}\}$ {Add x to high-fidelity archive}
  $\boldsymbol{x}^\beta \leftarrow \min(f_H(\boldsymbol{x}^\beta), f_H(\boldsymbol{x}))$ {Update best solution}
end while

IV. Numerical Experiments

- datasets (Lv et al. and maybe Wang and Jin's paper)
- details of machine and run on MATLAB
- number of runs
- parameters
- compared against base-line co-kriging algorithm
- give base-line co-kriging algorithm:
- same parameters as before

V. Results and Discussion

- present results and analysis
- convergence plots
- tables

VI. Conclusion and Future Work

- give conclusion
- future work includes adding constraints

References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] ——, "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.