Trust regions in Kriging-based optimization with expected improvement

Rommel G. Regis*
Department of Mathematics, Saint Joseph’s University, Philadelphia, PA 19131, USA

(Received 8 December 2014; accepted 8 August 2015)

The Kriging-based Efficient Global Optimization (EGO) method works well on many expensive black-box optimization problems. However, it does not seem to perform well on problems with steep and narrow global minimum basins and on high-dimensional problems. This article develops a new Kriging-based optimization method called TRIKE (Trust Region Implementation in Kriging-based optimization with Expected improvement) that implements a trust-region-like approach where each iterate is obtained by maximizing an Expected Improvement (EI) function within some trust region. This trust region is adjusted depending on the ratio of the actual improvement to the EI. This article also develops the Kriging-based CYCLONE (CYClic Local search in OptimizatioN using Expected improvement) method that uses a cyclic pattern to determine the search regions where the EI is maximized. TRIKE and CYCLONE are compared with EGO on 28 test problems with up to 32 dimensions and on a 36-dimensional groundwater bioremediation application in appendices supplied as an online supplement available at http://dx.doi.org/10.1080/0305215X.2015.1082350. The results show that both algorithms yield substantial improvements over EGO and they are competitive with a radial basis function method.

Keywords: global optimization; metamodel; surrogate model; Kriging; trust region

1. Introduction

One of the challenges in the field of engineering optimization is finding solutions to computationally expensive black-box optimization problems. There are many real-world engineering optimization problems where the values of the objective function to be maximized or minimized, and sometimes the values of the constraint functions, are outputs of time consuming computer simulations. Moreover, in many cases, accurate gradient information is unavailable so gradient-based approaches cannot be used. For these problems, the goal is to find the global minimum, or sometimes simply to find an improved solution, using as few function evaluations as possible.

A widely used approach for expensive black-box optimization involves metamodels (or surrogate models or response surface models) of the expensive functions. The Efficient Global Optimization (EGO) method by Jones, Schonlau, and Welch (1998) is a popular and highly cited metamodel-based method that uses Kriging metamodels for the objective function. EGO is meant for unconstrained or bound constrained global optimization of expensive functions and it selects function evaluation points by maximizing an Expected Improvement (EI) function based on the Kriging metamodel. Although Kriging assumes that the function values are outcomes of a stochastic process, the original EGO method is designed for deterministic functions. EGO

*Email: rregis@sju.edu
has been shown to work well on test problems and a variety of application problems. Moreover, it has been shown to converge to the global minimum under mild conditions (Vazquez and Bect, 2010). However, numerical experiments show that EGO does not perform as well as other methods on some low-dimensional benchmark problems with steep and narrow global minimum basins (Regis and Shoemaker, 2013) and also on some high-dimensional test problems (e.g. Chen et al., 2012).

This article develops a new Kriging-based optimization method called TRIKE (Trust Region Implementation in Kriging-based optimization with Expected improvement) that overcomes some of the limitations of the original EGO method. TRIKE implements a trust-region-like approach whereby it selects function evaluation points by maximizing the EI in a trust region centred at the current best solution and adjusts this trust region depending on the ratio of the actual improvement to the EI. This method combines the desirable statistical properties of Kriging prediction and the efficiency of derivative-free trust region algorithms (Conn, Scheinberg and Vicente, 2009). To the best of the author’s knowledge, EGO with the EI criterion has not been used in a trust region framework. This article also develops a variant of EGO called CYCLONE (CYClic Local search in OptimizationN using Expected improvement) that uses a cyclic pattern to determine the sizes of the search regions around the current best solution where the EI is maximized. TRIKE and CYCLONE are designed for difficult problems with steep and narrow global minimum basins and also high-dimensional problems (i.e. at least 10 dimensions), which relatively few papers on variants or extensions of EGO have addressed. These algorithms put more emphasis on local search, which other studies have found to be helpful in improving the performance of metamodel-based methods (e.g. Sasena, Papalambros, and Goovaerts 2002 and Regis and Shoemaker 2007). To improve performance further, these algorithms use a restart strategy from Regis and Shoemaker (2007) that has been shown to yield more robust and consistent results over multiple trials for Radial Basis Function (RBF) methods on difficult test problems. TRIKE and CYCLONE are compared with a standard implementation of EGO on 28 bound constrained test problems with dimensions ranging from 2 to 32 and on a 36-dimensional groundwater bioremediation application. The proposed algorithms are also compared with an RBF method called AQUARS-CGRBF (A QUAsi-multistart Response Surface-Controlled Gutmann-RBF) (Regis and Shoemaker, 2013) on the lower-dimensional test problems. Since the performance of metamodel-based methods can be quite sensitive to the choice of the initial space-filling design (Regis and Shoemaker, 2007), each algorithm is run for 30 trials corresponding to different space-filling designs on each of the 28 test problems. Furthermore, the various methods are compared in three different ways: (1) in terms of the mean number of function evaluations to get a near-optimal solution; (2) by means of data profiles (Moré, and Wild, 2009); and (3) by means of average progress curves. The numerical results show that the proposed methods yield consistent and substantial improvements over EGO and they are very competitive with AQUARS-CGRBF on the low-dimensional problems used. Finally, it is worth noting that extensive tests of metamodel-based methods involving many problems and multiple trials are not common in the literature, partly because of the enormous computational overhead incurred by these methods. In fact, in the seminal paper by Jones, Schonlau, and Welch (1998), the original EGO algorithm was run on only four test problems using only one trial per problem. Many other studies involving EGO and its variants also performed tests on only a handful of problems. Moreover, many of these studies reported results on the Branin and Hartman functions from the Dixon and Szegö (1978) test bed but excluded the more difficult Shekel problems that have deep and narrow global minimum basins that are also part of the same test bed. This study is among the few to do extensive tests involving EGO and the frequently ignored Shekel problems and also one of the few to apply EGO to high-dimensional problems with more than 30 decision variables. However, despite the previous statements, this article is in no way intended to be a criticism of the EGO algorithm and its variants. On the
contrary, it supports the original idea behind EGO and provides one way of getting consistently good results for Kriging-based methods that use the EI criterion on a wide variety of problems, including problems that used to be difficult for the original EGO algorithm.

2. Related work

Metamodelling or surrogate modelling is a commonly used approach for expensive black-box optimization, and Kriging (sometimes called Gaussian process modelling) (Sacks et al. 1989; Cressie 1993) is probably the most popular metamodelling technique. In Kriging, the function values obtained at sample points are treated as outcomes of a stochastic process. Kriging has been widely used in simulation-based optimization for more than a decade (e.g. Jones, Schonlau, and Welch 1998 and Simpson et al. 2001). As mentioned earlier, EGO (Jones, Schonlau, and Welch, 1998) uses Kriging metamodels to approximate the expensive black-box objective function and it selects function evaluation points by maximizing an Expected Improvement (EI) function. Reviews of metamodelling methods for engineering design optimization can be found in Wang and Shan (2007) and Shan and Wang (2010).

Various extensions and modifications of EGO and the EI criterion have been developed and applied to numerous practical problems. In fact, EGO is one of the most highly cited metamodel-based methods for global optimization. This section mentions some of these extensions but it is not intended to be exhaustive. For example, Sasena, Papalambros, and Goovaerts (2002), Parr et al. (2012) and Basudhar et al. (2012) developed extensions of EGO for constrained optimization. Huang et al. (2006), Forrester, Keane, and Bressloff (2006) and Picheny, Ginsbourger, and Richet (2010) extended EGO to handle noisy and expensive single-objective optimization. Knowles (2006), Ponweiser et al. (2008), Zhang et al. (2010) and Couckuyt, Deschrijver, and Dhaene (2014) developed extensions of EGO and the EI criterion to handle expensive multiobjective optimization. Kleijnen, van Beers, and van Nieunwuyenye (2012) developed an improved variant of EGO that uses an unbiased bootstrap estimator of the variance of the Kriging predictor. Marzat, Walter, and Piet-Lahanier (2013) combined EGO with a relaxation procedure to develop an algorithm for continuous minimax optimization of black-box functions. In terms of applications, Morgans et al. (2008) applied EGO to optimize the shape of a horn loaded loudspeaker to improve sound quality while Aleman, Romeijn, and Dempsey (2009) modified EGO to solve a beam orientation optimization problem in intensity-modulated radiation therapy (IMRT) treatment planning. Finally, Hamza and Shalaby (2014) developed a variant of EGO that generates multiple sample points that can be evaluated in parallel and applied it to a vehicle crashworthiness design problem.

Other Kriging-based methods include Informational Approach to Global Optimization (IAGO) (Villemonteix, Vazquez, and Walter, 2009), which uses minimizers entropy as a criterion for selecting the next function evaluation point. Moreover, Hemker et al. (2008) developed a branch-and-bound algorithm that uses DACE surrogates for mixed-integer expensive black-box optimization and applied it to some water resources benchmarking problems. Glaz et al. (2009) showed the benefit of multiple surrogates, including Kriging metamodels, to approximate expensive functions. Viana, Haftka, and Watson (2013) developed MSEG (Multiple Surrogate Efficient Global Optimization), which also uses multiple surrogates, including Kriging, to generate multiple function evaluation points instead of a single one for each iteration. Finally, Kriging metamodels have been used to assist pattern search (Booker et al. 1999), scatter search (Egea et al. 2009), evolutionary algorithms (e.g. Ratle 2001, Won and Ray 2005, Emmerich, Giannakoglou, and Naujoks 2006 and Annicchiarico 2007), and particle swarm optimization algorithms (e.g. Parno, Hemker, and Fowler 2012).
As mentioned earlier, the TRIKE algorithm is a Kriging-based method that uses a trust-region-like approach where the next iterate is obtained by maximizing the EI function over a box-shaped trust region centred at the current best sample point. The size of this trust region increases, decreases or stays the same according to the ratio of the actual improvement to the EI. Model-based trust region methods are now among the most popular approaches for the derivative-free local optimization of expensive functions (e.g. Conn, Scheinberg, and Vicente 2009, Powell 2002, Oeuvray and Bierlaire 2009 and Wild, Regis, and Shoemaker 2008). TRIKE uses ideas from these methods to develop a Kriging-based method that improves on the performance of traditional EGO. One main difference between TRIKE and model-based trust region methods is that TRIKE uses the EI instead of the predicted improvement as the denominator in the ratio used to decide how to modify the size of the trust region. In contrast, derivative-free trust region algorithms (Conn, Scheinberg and Vicente, 2009) simply use the predicted improvement based on the model being used. Moreover, TRIKE uses all available sample points in building the Kriging model of the objective function while derivative-free trust region methods typically use a subset of the sample points for building a local model of the objective function. Finally, some of these derivative-free trust region methods have theoretical convergence guarantees (e.g. Conn, Scheinberg, and Vicente 2009, Oeuvray and Bierlaire 2009 and Wild and Shoemaker 2013) while TRIKE is a heuristic local search method.

The trust-region-like approach in TRIKE facilitates the generation of sample points that improve the objective function value by promoting local search, which previous studies have shown to be helpful for metamodel-based methods. For example, Sasena, Papalambros, and Goovaerts (2002) found that infill criteria for EGO that put more emphasis on local search tend to locate optima in fewer iterations than those that emphasize global search. Moreover, Regis and Shoemaker (2007) found that modifying the infill criterion of the RBF method by Gutmann (2001) to ensure an adequate amount of local search substantially reduces the number of function evaluations to find the global minimum on test problems, especially those that have deep and narrow basins for the global minimum. In addition, Cheng and Wang (2012) found that incorporating a trust region approach substantially improved the performance of a Mode Pursuing Sampling (MPS) algorithm on high-dimensional problems. However, to the best of the author’s knowledge, a trust-region-like approach has never been used in combination with a Kriging-based method based on the EI criterion.

Radial Basis Function (RBF) interpolation is another metamodelling approach for global optimization (Gutmann, 2001) and also for local optimization (Oeuvray and Bierlaire, 2009; Wild, Regis, and Shoemaker 2008). RBF methods tend to locate the global minimum of benchmark test problems more quickly and more frequently by employing a complete restart using a new space-filling design whenever the algorithm has stalled rather than allowing them to continue until the computational budget is exhausted (Regis and Shoemaker, 2007). Hence, a similar restart strategy is used for the TRIKE algorithm. Moreover, Regis and Shoemaker (2013) employed a quasi-multistart approach to improve the performance further of the RBF method by Gutmann (2001) and the TRIKE approach will be shown below to be competitive with this RBF approach on the low-dimensional problems.

3. Efficient global optimization and expected improvement function

In the Kriging surrogate model described in Jones, Schonlau, and Welch (1998) and Jones (2001) (sometimes called the DACE stochastic process model), the values of the black-box function \( f \) are assumed to be the outcomes of a stochastic process. In particular, before \( f \) is evaluated at any point, it is assumed that the function value at any point \( x \) in the domain is a realization of
a random variable $Y(x)$ that is normally distributed with mean $\mu$ and variance $\sigma^2$. Moreover, for any two points $x_i$ and $x_j$, it is assumed that the correlation between $Y(x_i)$ and $Y(x_j)$ can be modelled by

$$\text{Corr}[Y(x_i), Y(x_j)] = \exp\left(-\sum_{\ell=1}^{d} \theta_\ell |x_i\ell - x_j\ell|^p \right), \quad (1)$$

where $\theta_\ell, p_\ell (\ell = 1, \ldots, d)$ are parameters of the model. This correlation model is only one of many types of correlation function that can be used in Kriging metamodels. Note that when $x_i$ and $x_j$ are close, $Y(x_i)$ and $Y(x_j)$ will be highly correlated according to this model. As $x_i$ and $x_j$ become farther apart, the correlation drops to zero.

Given a set of $n$ points $x_1, \ldots, x_n \in \mathbb{R}^d$, the uncertainty about the values of $f$ at these points can be modelled by using the random vector $Y = (Y(x_1), \ldots, Y(x_n))^T$. Note that $E(Y) = J_{n \times 1} \mu$, where $J_{n \times 1}$ is the $n \times 1$ vector of all ones, and $\text{Cov}(Y) = \sigma^2 R$, where $R$ is the $n \times n$ matrix whose $(i,j)$ entry is given by (1).

Suppose the function $f$ has been evaluated at the points $x_1, \ldots, x_n \in \mathbb{R}^d$. Let $y_1 = f(x_1), \ldots, y_n = f(x_n)$ and let $y = (y_1, \ldots, y_n)^T$ be the vector of observed function values. Fitting the Kriging model described by Jones (2001) through the data points $(x_1, y_1), \ldots, (x_n, y_n)$ involves finding the Maximum Likelihood Estimates (MLEs) of the parameters $\mu, \sigma^2, \theta_1, \ldots, \theta_d, p_1, \ldots, p_d$. In practice, the MLEs of these parameters are typically obtained by solving a numerical optimization problem. Now, to determine the value of the Kriging predictor at a new point $x^*$, Jones (2001) provided the formula

$$\hat{y}(x^*) = \hat{\mu} + r^T R^{-1} (y - J_{n \times 1} \hat{\mu}),$$

where $\hat{\mu} = J_{n \times 1} R^{-1} y$ and $r = (\text{Corr}[Y(x^*), Y(x_1)], \ldots, \text{Corr}[Y(x^*), Y(x_n)])^T$. Moreover, a measure of error of the Kriging predictor at $x^*$ is given by

$$s^2(x) = \hat{\sigma}^2 \left[ 1 - r^T R^{-1} r + \frac{(1 - r^T R^{-1} r)^2}{J_{n \times 1} R^{-1} J_{n \times 1}} \right],$$

where $\hat{\sigma}^2 = (y - J_{n \times 1} \hat{\mu})^T R^{-1} (y - J_{n \times 1} \hat{\mu}) / n$.

In the EGO method by Jones, Schonlau, and Welch (1998), the next point for function evaluation is a point $x$ that maximizes an expected improvement function $\text{EI}(x)$ over the domain. Here,

$$\text{EI}(x) = (f_{\text{min}} - \hat{y}(x)) \Phi \left( \frac{f_{\text{min}} - \hat{y}(x)}{s(x)} \right) + s(x) \phi \left( \frac{f_{\text{min}} - \hat{y}(x)}{s(x)} \right), \quad (2)$$

where $\Phi$ and $\phi$ are the cdf and pdf of the standard normal distribution, respectively.

### 4. Kriging-based optimization using the expected improvement criterion in trust regions

#### 4.1. Algorithm description

Below is a description of the TRIKE algorithm, which is a Kriging-based method that uses a trust-region-like approach where the next iterate is a maximizer of the EI in (2) over a box-shaped trust region centred at the current best sample point. The size of this trust region increases, decreases or stays the same according to the ratio of the Actual Improvement (AI) to the EI. Note that
derivative-free trust region algorithms (Conn et al. 2009) use the predicted improvement based on the metamodel instead of the EI as the denominator in this ratio.

**TRIKE (Trust Region Implementation in Kriging-based optimization with Expected improvement)**

**Inputs:**

(1) A real-valued deterministic black-box function \( f \) whose values are given by a simulator.
(2) The search space, which is defined as a closed hypercube \( D = [a, b] \subset \mathbb{R}^d \).
(3) Initial points \( I = \{x_1, \ldots, x_{n_0}\} \subset D \) from a space-filling experimental design.
(4) The threshold parameter \( \epsilon_{EI} > 0 \) for the expected improvement function.
(5) The maximum number of function evaluations allowed, denoted by \( N_{max} \).
(6) Initial length, minimum length and maximum length of the box-shaped trust region, denoted by \( r_0, r_{min}, \) and \( r_{max} \), respectively.
(7) Threshold for the ratio of the AI to EI, denoted by \( \eta \).
(8) Contraction factor \( 0 < \gamma < 1 \) for the box-shaped trust region.
(9) Expansion factor \( \gamma_1 > 1 \) for the box-shaped trust region.
(10) Validity factor \( C > 1 \) used before deciding to reduce the trust region.

**Output:** The best point encountered by the algorithm.

**Step 1.** (Evaluate Initial Points) Compute \( f(x_i) \) for each \( i = 1, \ldots, n_0 \). Let \( f_{best} \) be the best point found so far and let \( f_{best} := f(x_{best}) = \min_{1 \leq i \leq n_0} f(x_i) \). Set \( n := n_0 \).

**Step 2.** While \( n < N_{max} \) and \( (EI \geq \epsilon_{EI}) \) do

**Step 2.1** (Fit/Update Kriging Metamodel) Fit or update a Kriging metamodel \( s_n(x) \) using all available data points \( \{(x_i, f(x_i)) : i = 1, \ldots, n\} \).

**Step 2.2** (Maximize Expected Improvement over Trust Region) Let \( x_{n+1} \) be a maximizer of EI(\( x \)) over \( D_n := [x_{best} - r_n/2, x_{best} + r_n/2] \cap [a, b] \) and let EI = EI(\( x_{n+1} \)).

**Step 2.3** (Perform Function Evaluation) Compute \( f(x_{n+1}) \).

**Step 2.4** (Compute Actual Improvement) Compute AI = max\( (f_{best} - f(x_{n+1}), 0) \).

**Step 2.5** (Update Best Solution) If \( f(x_{n+1}) < f_{best} \), set \( x_{best} = x_{n+1} \) and \( f_{best} = f(x_{n+1}) \).

**Step 2.6** (Adjust Length of Box-Shaped Trust Region) If AI/EI \( \geq \eta \), then \( r_{n+1} = \min(\gamma r_n, r_{max}) \); Else if AI = 0 and there are \( d + 1 \) affinely independent points within the box \( [x_{best} - C r_n/2, x_{best} + C r_n/2] \), then \( r_{n+1} = \max(\gamma_1 r_n, r_{min}) \).

**Step 2.7** (Increment Counter of Function Evaluations) Reset \( n := n + 1 \).

End.

**Step 3.** (Return the Best Solution Found) Return \( x_{best} \) and \( f_{best} \).

Input (1) is the deterministic black-box objective function \( f \) to be minimized. In many practical situations, the values of \( f \) are outputs of a computationally expensive simulation. Input (2) is the search space \( D \), which is a closed hypercube. TRIKE can also be applied when \( D \) is a closed hyperrectangle since such a region can be transformed into the unit hypercube \( [0, 1]^d \) by rescaling the variables. Input (3) is a set of initial space-filling design points \( I \) where \( f \) will be evaluated to obtain data needed to construct the initial Kriging metamodel. A commonly used space-filling design is the Latin Hypercube Design (LHD), and here, a Symmetric Latin Hypercube Design (SLHD) (Ye, Li, and Sudjianto, 2000) is used. Input (4) is the threshold expected improvement \( \epsilon_{EI} \) used to terminate the algorithm. That is, the algorithm is terminated when the maximum value of EI(\( x \)) over the current trust region is less than \( \epsilon_{EI} \). Input (5) is the computational budget \( N_{max} \), which is the maximum number of function evaluations that can be performed. Input (6) consists of the initial, minimum and maximum length of the box-shaped trust region around the current best solution. Finally, Inputs (7)–(10) are similar to parameters commonly used in model-based trust region algorithms as described in Conn et al. (2009).
TRIKE begins by evaluating the objective function at the points of the space-filling design (Step 1). Then the algorithm loops through the iterations (Step 2) and returns the best solution found (Step 3). Each iteration begins by forming the Kriging metamodel described in Section 3 (Step 2.1). Next, El(\(x\)) is maximized over the current trust region \(D_n\) to obtain the next function evaluation point \(x_{n+1}\) (Step 2.2). The objective function is then evaluated at \(x_{n+1}\) (Step 2.3), the actual improvement \(\Delta I\) is calculated (Step 2.4), and then the current best point \(x_{\text{best}}\) is updated (Step 2.5). Next, the ratio \(\Delta I/\text{EI}\) of the actual improvement to the expected improvement is calculated and compared with a threshold \(\eta\) (Step 2.6). If this ratio is at least \(\eta\), then the current length of the box-shaped trust region is increased using the factor \(\gamma_1\). If this ratio is zero (i.e. if there is no actual improvement) and there are \(d+1\) affinely independent points within a box of length \(Cr_n\) centred at \(x_{\text{best}}\), where \(C > 1\), then the length of the box-shaped trust region is decreased using the factor \(\gamma_0\). (A set of points \(y_0, y_1, \ldots, y_k \in \mathbb{R}^d\) are said to be affinely independent if the displacements from a fixed point \(y_1 - y_0, \ldots, y_k - y_0\) are linearly independent.) In all other cases, the length of the box-shaped trust region stays the same. Although \(d+1\) affinely independent points are not required to fit the Kriging model described in Section 3, the purpose of this requirement when there is no actual improvement is to ensure that there are enough data points within a suitable neighbourhood of \(x_{\text{best}}\) before reducing the trust region. This requirement is used by derivative-free trust region methods, including ORBIT (Wild et al. 2008), and the required number of points is at least enough to fit a linear model even though no such model is actually fitted. The parameter \(C\) is referred to as a validity factor. Finally, the iteration ends by incrementing the counter for the number of function evaluations (Step 2.7).

4.2. Restart strategy

Regis and Shoemaker (2007) noted that it is sometimes better to restart a surrogate-based algorithm using a new initial space-filling design rather than to let it continue until it finds the global optimum. This restart strategy sounds counterintuitive in that complete restarts using a new space-filling design appear wasteful in the computationally expensive setting. However, numerical evidence presented in Regis and Shoemaker (2007) strongly suggests that restarting from scratch when the algorithm appears to have stalled is a much better alternative to allowing the algorithm to continue, especially on difficult test problems with steep and narrow global minimum basins. Part of the success of this seemingly counterintuitive strategy stems from the fact that surrogate-based methods are quite sensitive to the initial space-filling design used. On a given test problem, it is possible that for one space-filling design, a surrogate-based method finds the global minimum quickly, while for a different space-filling design, the same method takes a very long time to find the global minimum. By doing restarts when the algorithm has not been making progress or is no longer expected to make progress, the number of function evaluations to reach the global minimum is substantially reduced on average. Hence, TRIKE is restarted whenever the EI falls below a threshold \(\epsilon_{\text{EI}}\). The resulting algorithm is referred to as TRIKE-Restart.

Although the convergence of EGO to the global minimum has been established (Vazquez and Bect, 2010), the numerical results obtained suggest that the convergence of EGO might be slow on some well-known test problems. Moreover, the trust-region-like strategy proposed in TRIKE puts more emphasis on local search and so there is no more guarantee that the convergence result holds. Fortunately, the use of a restart strategy makes it possible for a stochastic algorithm to converge again to the global minimum in a probabilistic sense as explained below.

The success of a restart strategy can be explained by observing that, under certain assumptions, the number of runs of an algorithm needed to find the global minimum or a near optimal solution follows a geometric probability distribution. Below is one version of a somewhat known result...
that applies to optimization algorithms that implement a multistart or restart strategy, including the TRIKE-Restart algorithm. In the proposition below, an algorithm is considered stochastic if it involves a random initialization procedure. For example, given a fixed initial space-filling design, the TRIKE algorithm is really deterministic. However, if one randomly varies the initial space-filling design used, then TRIKE now behaves like a stochastic algorithm. Moreover, a near-optimal solution is a feasible point whose objective function value is within some pre-specified fraction of the global minimum value. Finally, a computational budget \( \beta \) is included since, in practice, an optimization algorithm is usually run up to a maximum number of function evaluations so that it stops either on its own or because the budget is reached.

**Proposition 4.1** Consider the problem \( \min \{ f(x) : x \in D \} \), where \( D \subseteq \mathbb{R}^d \) and \( f : D \to \mathbb{R} \) is a deterministic function such that the global minimum of \( f \) over \( D \) exists. Suppose a stochastic algorithm \( A \) finds a near-optimal solution of this problem within \( \beta \) function evaluations with probability \( p_{\text{opt}}(\beta) \). Furthermore, assume that independent runs of \( A \) are performed, each with a computational budget of \( \beta \) function evaluations, and let \( \Gamma(A, \beta) \) be a random variable that represents the number of these independent runs of \( A \) before a near-optimal solution is found. Then \( \Gamma(A, \beta) \) has a geometric distribution with success probability \( p_{\text{opt}}(\beta) \) and

\[
P(A \text{ finds a near-optimal solution within } k \text{ runs}) = P(\Gamma(A, \beta) \leq k) = 1 - (1 - p_{\text{opt}}(\beta))^k.
\]

Moreover, if \( 0 < p_{\text{opt}}(\beta) \leq 1 \), then \( \lim_{k \to \infty} P(\Gamma(A, \beta) \leq k) = 1 \). 

**Proof** It is obvious why \( \Gamma(A, \beta) \) follows a geometric distribution with success probability \( p_{\text{opt}}(\beta) \). Moreover, since the runs of \( A \) are independent, it follows that \( P(\Gamma(A, \beta) > k) = P(A \text{ did not find a near-optimal solution during the first } k \text{ runs}) = \prod_{i=1}^{k} P(A \text{ did not find a near-optimal solution in the } i\text{th run}) = (1 - p_{\text{opt}}(\beta))^k \), and so

\[
P(\Gamma(A, \beta) \leq k) = 1 - P(\Gamma(A, \beta) > k) = 1 - (1 - p_{\text{opt}}(\beta))^k.
\]

Finally, \( \lim_{k \to \infty} 1 - (1 - p_{\text{opt}}(\beta))^k = 1 \) since \( 0 \leq 1 - p_{\text{opt}}(\beta) < 1 \). 

In practice, \( p_{\text{opt}}(\beta) \) is hard to estimate. However, Proposition 4.1 says that, as long as \( p_{\text{opt}}(\beta) > 0 \), the stochastic algorithm will eventually find a near-optimal solution.

## 5. Cyclic local search in Kriging-based optimization using the expected improvement criterion

Below is a description of the CYCLONE algorithm, which is a variant of EGO wherein the size of the search boxes around the current best solution is determined by cycling through a list of user-specified sizes. This cycle of lengths for the search boxes should include a combination of small and large values. Small search boxes promote local search while large search boxes (sometimes identical to the entire search space) promote global search.

**CYCLONE (CYClic Local search in OptimizatioN using Expected improvement)**

**Inputs:**

1. A real-valued deterministic black-box function \( f \) whose values are given by a simulator.
2. The search space, which is defined as a closed hypercube \( D = [a, b] \subset \mathbb{R}^d \).
3. Initial points \( \mathcal{I} = \{x_1, \ldots, x_{n_0}\} \subset D \) from a space-filling experimental design.
4. The threshold parameter for the expected improvement function, denoted by \( \epsilon_{\text{EI}} \).
(5) The maximum number of function evaluations allowed, denoted by \( N_{\text{max}} \).

(6) Cycle of relative sizes of the search boxes around the current best point, denoted by \( \Theta = \langle \rho_1, \rho_2, \ldots, \rho_k \rangle \), where \( 0 < \rho_i \leq 2 \) for \( i = 1, \ldots, k \). This is called a search pattern.

Output: The best point encountered by the algorithm.

**Step 1.** (Evaluate Initial Points) Compute \( f(x_i) \) for each \( i = 1, \ldots, n_0 \). Let \( x_{\text{best}} \) be the best point found so far and \( \ell_{\text{best}} := f(x_{\text{best}}) = \min_{1 \leq i \leq n_0} f(x_i) \). Set \( n := n_0 \).

**Step 2.** While \( n < N_{\text{max}} \) and \( (\text{EI} \geq \epsilon_{\text{EI}}) \) do

**Step 2.1** (Fit/Update Kriging Metamodel) Fit or update a Kriging metamodel \( s_n(x) \) using all available data points \( \{(x_i, f(x_i)) : i = 1, \ldots, n\} \).

**Step 2.2** (Determine Search Box) Set \( r_n = \rho_j \ell([a, b]) \), where \( \ell([a, b]) \) is the length of one side of \([a, b]\) and \( \rho_j \) is the value in the cycle \( \Theta = \langle \rho_1, \rho_2, \ldots, \rho_k \rangle \) that corresponds to this iteration. Then, set \( D_n := [x_{\text{best}} - (r_n/2), x_{\text{best}} + (r_n/2)] \cap [a, b] \).

**Step 2.3** (Maximize Expected Improvement over Search Box) Let \( x_{n+1} \) be a maximizer of \( \text{EI}(x) \) over \( D_n \) and let \( \text{EI} = \text{EI}(x_{n+1}) \).

**Step 2.4** (Perform Function Evaluation) Compute \( f(x_{n+1}) \).

**Step 2.5** (Update Best Solution) If \( f(x_{n+1}) < f_{\text{best}} \), then set \( x_{\text{best}} = x_{n+1} \) and \( f_{\text{best}} = f(x_{n+1}) \).

**Step 2.6** (Increment Counter of Function Evaluations) Reset \( n := n + 1 \).

End.

**Step 3.** (Return the Best Solution Found) Return \( x_{\text{best}} \) and \( f_{\text{best}} \).

As before, CYCLONE begins by evaluating the objective function at the points of the space-filling design (Step 1). Then the algorithm loops through the iterations (Step 2) and returns the best solution found (Step 3). Each iteration begins by forming the Kriging metamodel (Step 2.1). Next, the search box around the current best solution is determined (Step 2.2). Then the expected improvement function \( \text{EI}(x) \) is maximized over the current search box \( D_n \) to obtain the next function evaluation point \( x_{n+1} \) (Step 2.3). Next, the objective function is evaluated at \( x_{n+1} \) (Step 2.4) and the current best point \( x_{\text{best}} \) is updated (Step 2.5). Finally, the iteration ends by incrementing the counter for the number of function evaluations (Step 2.6). In Step 2.2, note that, when \( \rho_j = 2 \), the resulting \( D_n \) is the entire search space. Hence, CYCLONE with a search pattern \( \Theta = \langle 2 \rangle \) is identical to the original EGO algorithm.

As with the TRIKE algorithm, CYCLONE is also implemented with restarts to improve its performance further. The resulting algorithm is referred to as CYCLONE-Restart.

6. Numerical experiments

6.1. Test problems

TRIKE and CYCLONE are compared with a standard implementation of EGO on 28 test problems with dimensions ranging from 2 to 32 and also on a 36-dimensional groundwater bioremediation problem (Yoon and Shoemaker, 1999) called GWB36. Tables A.1 and A.2 in Appendix A (supplied in the online supplement—see the section ‘Supplemental data’ before the references list) provide some information on these test problems. Appendix B (supplied in the online supplement—see the section ‘Supplemental data’ before the references list) provides a description of the groundwater application.

Table A.1 in Appendix A shows 18 lower-dimensional test problems (2 to 6 decision variables), which are the ones used in Regis and Shoemaker (2013). These include the seven Dixon and Szegö (1978) test functions (Branin, Goldstein-Price, Hartman3, Shekel5, Shekel7, Shekel10...
and Hartman6), six problems that have the same mathematical form as the Shekel problems from Dixon and Szegö (1978), and five instances of the Gordom–Wixom–Schoen test functions (Schoen, 1993), labelled GWSS or GWSC.

Table A.2 in Appendix A shows 10 higher-dimensional test problems with 30 or 32 decision variables. The first six problems in this table (Ackley, Rastrigin, Griewank, Keane, Levy, Michalewicz) have many local minima and are well known in the engineering optimization community. The last four problems (Extended Rosenbrock, Extended Powell Singular, Trigonometric, Broyden Tridiagonal) each have only one local minimum and are taken from the Moré, Garbow, and Hillstrom (1981) test set.

The test functions mentioned above are not really computationally expensive to evaluate. However, algorithms may be compared in terms of their potential for truly expensive functions by allowing them to run for only a limited number of function evaluations. The relative performance of algorithms on some of these test functions are expected to be similar to their relative performance on truly expensive real-world objective functions whose characteristics resemble those of the test functions. Additionally, performing these comparisons on computationally cheap test problems makes it possible to perform multiple trials and compare average case performance.

6.2. Experimental setup

To evaluate the effectiveness of the proposed algorithms (TRIKE-Restart and CYCLONE-Restart), they are compared with a standard implementation of EGO on 28 test problems and on a groundwater bioremediation application. Moreover, to test if restarts make a difference, TRIKE-Restart and CYCLONE-Restart are also compared with TRIKE and CYCLONE, respectively, on the 18 lower-dimensional test problems. However, to allow TRIKE and CYCLONE to run until the computational budget is exhausted, the expected improvement threshold $\epsilon_{EI}$ is not used for termination.

TRIKE and CYCLONE are not run on the 10 higher-dimensional test problems because their performances are expected to be similar to those for the restart versions when the computational budget is relatively limited and also because of the enormous computational expense involved in running multiple trials of these algorithms on several test problems. Note that restarts will hardly occur on the higher-dimensional problems given the relatively limited number of function evaluations. In addition, the proposed algorithms are compared with a recently developed RBF method called the AQUARS-CGRBF method (Regis and Shoemaker, 2013). AQUARS-CGRBF performed much better than many alternative methods on the 18 lower-dimensional test problems, so comparison with this method provides some indirect comparison with the other alternatives.

All algorithms are run in MATLAB® 7.11.0 using an Intel® Core™ i7 CPU 860 2.8GHz desktop PC. In particular, EGO, CYCLONE and TRIKE (with or without restarts) are implemented through the Surrogates Toolbox by Viana (2010). In this toolbox, the EI function is maximized by means of a Genetic Algorithm (GA). Other solvers could be used to maximize the EI, but GA is the solver used in Viana (2010). Each algorithm is run 30 times on each of the 28 test problems and the groundwater application. Each run of an algorithm begins with a Symmetric Latin Hypercube Design (SLHD) of size $2(d + 1)$ containing a subset of $d + 1$ affinely independent points as was done in the AQUARS algorithms in Regis and Shoemaker (2013). All algorithms use the same SLHD on a given trial to ensure fair comparisons. On the lower-dimensional problems, the algorithms are run up to a maximum of 2000 function evaluations to allow many of them to get within 1% of the global minimum value. On the higher-dimensional problems, the algorithms are run only up to a maximum of 500 function evaluations because the overhead computational
expense for the metamodels is quite large and also because none of the algorithms is expected to find the global minimum within the limited budget anyway.

In terms of parameters, TRIKE-Restart and CYCLONE-Restart both used an EI threshold of $\epsilon_{\text{EI}} = 0.001$. CYCLONE and CYCLONE-Restart used a search pattern of $SP = (2, 0.4, 0.2, 0.1, 0.05, 0.02, 0.01)$. The box-shaped trust regions for TRIKE and TRIKE-Restart have an initial length of $r_0 = 2\ell([a, b])$, minimum length of $r_{\min} = (1/64)\ell([a, b])$, and maximum length of $r_{\max} = 2\ell([a, b])$. Moreover, TRIKE and TRIKE-Restart use $\eta = 1$, $\gamma_0 = 1/2$, $\gamma_1 = 2$ and $C = 2$. AQUARS-CGRBF used the same parameters and type of RBF model as in Regis and Shoemaker (2013).

Finally, before presenting the results, it is worth emphasizing that another important contribution of this article is its extensive testing of EGO and the proposed Kriging-based methods on many test problems, including relatively high-dimensional problems. Due to the enormous computational overheads involved, most studies test a metamodel-based method on only a handful of low-dimensional problems. In fact, EGO was originally tested by Jones, Schonlau, and Welch (1998) on only four of the Dixon and Szegö (1978) test problems and using only one trial per problem. The largest problem in this test bed has only six decision variables. Similarly, Sasena, Papalambros, and Goovaerts (2002) tested EGO with various infill criteria on four test problems with up to two decision variables and on a simulation-based application with three decision variables. In contrast, this study ran EGO and the proposed algorithms on 29 problems (28 test problems with up to 32 decision variables and one 36-dimensional simulation-based groundwater application) using 30 trials per problem, resulting in $29 \times 30 = 870$ trials for each algorithm. Regis and Shoemaker (2007) and also this study showed that the performance of metamodel-based methods is quite sensitive to the choice of the initial space-filling design. Hence, the use of multiple trials with different initial space-filling designs provides a more accurate measure of algorithm performance.

7. Results and discussion

7.1. Comparison of results on the lower-dimensional test problems

Table 1 shows the average number of function evaluations (over 30 trials) that each algorithm took to get within 1% of the global minimum value on the 18 lower-dimensional test problems (2 to 6 decision variables). If all 30 trials got within 1% of the global minimum value, then the number inside the parenthesis is the standard error of the mean, i.e. the standard deviation divided by the square root of the number of trials. If one of the trials did not get within 1% of the global minimum value, then the number inside the parenthesis is the number of trials (out of 30) that did not get within 1% of the global minimum value within 500 function evaluations for EGO, CYCLONE and TRIKE, and within 2000 function evaluations for the other methods. The reason for these different computational budgets is that one iteration of EGO, CYCLONE or TRIKE becomes much slower as more sample points are accumulated. However, the restarts in CYCLONE-Restart and TRIKE-Restart prevent the accumulation of sample points since previous sample points are ignored. Hence, these other methods do not incur the same amount of computational overhead as EGO, CYCLONE and TRIKE.

Before proceeding, note that the results for EGO in Table 1 are the same ones reported in Regis and Shoemaker (2013). These results are averages over 30 trials using different SLHDs of size $2(d + 1)$. On the other hand, the original results for EGO on the four test problems in Jones, Schonlau, and Welch (1998) (Brainin, Golstein-Price, Hartman3 and Hartman6) are obtained from one trial using a particular type of space-filling design. Moreover, since the performance of RBF methods is quite sensitive to the choice of the initial space-filling design, it is
Table 1. Average number of function evaluations (over 30 trials) to get a relative error of less than 1%. If all 30 trials got a relative error of < 1%, then the number inside the parenthesis is the standard error of the mean. Otherwise, it is the number of trials that did not get a relative error of < 1% within 500 function evaluations for EGO, CYCLONE and TRIKE and within 2000 function evaluations for the other methods.

| Test problem | EGO   | CYCLONE | TRIKE  | CYCLONE-Restart | TRIKE-Restart | AQUARS-CGRBF |
|--------------|-------|---------|--------|-----------------|---------------|---------------|
| Branin       | 40.73 (2.03) | 17.33 (0.41) | 19.90 (1.31) | 21.23 (1.57) | 25.63 (2.96) | 39.43 (1.73) |
| GP           | 115.93 (10.20) | 62.70 (9.89) | 61.00 (11.40) | 56.73 (6.43) | 53.77 (7.60) | 65.77 (6.31) |
| GWSS_Spiked  | 136.23 (8.62) | > 294.20 (11) | > 324.20 (17) | 179.67 (36.90) | 75.27 (10.81) | 64.40 (5.85) |
| Shekel2A     | 73.53 (2.62) | 43.23 (5.91) | 34.67 (5.91) | 36.67 (4.96) | 26.60 (2.59) | 26.87 (1.84) |
| Shekel2B     | 73.07 (3.96) | 73.83 (7.94) | 43.23 (6.74) | 70.30 (14.04) | 56.50 (9.38) | 32.53 (2.13) |
| Shekel2C     | > 189.20 (6) | > 207.47 (5) | > 243.30 (9) | 52.03 (7.02) | 46.23 (5.94) | 38.00 (3.28) |
| Shekel3A     | > 475.07 (24) | > 393.00 (22) | > 406.13 (24) | 167.27 (22.17) | 186.33 (38.39) | 122.83 (8.88) |
| Shekel3B     | > 60.77 (9.74) | > 245.23 (12) | > 265.07 (13) | 74.00 (9.87) | 105.80 (16.47) | 76.20 (5.54) |
| Shekel3C     | > 500.00 (30) | > 282.10 (14) | > 265.07 (13) | 74.00 (9.87) | 105.80 (16.47) | 76.20 (5.54) |
| GWSC(4,10)   | > 60.77 (9.74) | > 245.23 (12) | > 269.00 (13) | 97.63 (17.53) | 89.67 (17.14) | 54.73 (6.12) |
| Shekel5      | > 500.00 (30) | > 328.10 (16) | > 318.43 (18) | 236.20 (40.46) | 202.43 (42.06) | 164.67 (11.96) |
| Shekel7      | > 500.00 (30) | > 294.20 (15) | > 334.50 (19) | 194.87 (22.76) | 125.03 (16.69) | 121.10 (7.34) |
| Shekel10     | > 500.00 (30) | > 268.60 (11) | > 319.03 (18) | 236.20 (40.46) | 202.43 (42.06) | 164.67 (11.96) |
| GWSC(5,8)    | > 60.77 (9.74) | > 245.23 (12) | > 252.90 (13) | 97.63 (17.53) | 89.67 (17.14) | 54.73 (6.12) |
| Hartman3     | > 208.27 (11) | > 191.80 (10) | > 183.53 (9) | 61.07 (6.09) | 54.47 (5.81) | 129.80 (17.31) |
| GWSS(6,8)    | > 500.00 (30) | > 406.47 (23) | > 312.83 (15) | 165.63 (21.01) | 172.90 (19.36) | 223.87 (13.92) |

not surprising that this is also the case for EGO. Taking all these things into consideration, the EGO results in Table 1 are not necessarily inconsistent with the original EGO results on the four problems. In fact, an average over multiple space-filling designs is more likely to be an accurate indicator of algorithm performance than the result of one trial. Moreover, Table 1 shows that EGO did not get within 1% of the global minimum value of these Shekel problems in all 30 trials, confirming that this method has difficulty with this type of problem. As explained below, the proposed Kriging-based methods are able to overcome these difficulties and provide robust and consistent performance on the Shekel problems.

Now from Table 1, TRIKE-Restart is better than EGO on 17 of the 18 lower-dimensional test problems (all except GWSC(4,10)) while CYCLONE-Restart is better than EGO on 15 of these problems (all except GWSS_Spiked, Shekel2B and GWSC(4,10)). Moreover, TRIKE-Restart is better than TRIKE (without the restart) on 15 of the 18 lower-dimensional problems (all except Branin, Shekel2B and Hartman3). Also, CYCLONE-Restart is better than CYCLONE (without the restart) on 15 of the lower-dimensional problems (all except Branin, Shekel2B and Hartman3). In summary, this table shows that both TRIKE-Restart and CYCLONE-Restart are substantial improvements over standard EGO and that removing the restart results in deterioration of performance.

Table 1 also shows the comparison among the TRIKE and CYCLONE algorithms (with and without restarts) and the AQUARS-CGRBF algorithm (Regis and Shoemaker, 2013). AQUARS-CGRBF is a metamodel-based method that uses RBF models of the objective to prevent the algorithm from visiting previously found local minima and to facilitate the search for the global minimum. AQUARS-CGRBF performed much better than many alternative methods, including CGRBF-Restart (Regis and Shoemaker, 2007), multistart Mesh Adaptive Direct Search (MADS) (Audet and Dennis 2006) with DACE surrogates (Abramson, 2007), and multistart sequential quadratic programming (SQP) that uses finite difference gradients. The table shows that TRIKE-Restart is very competitive with AQUARS-CGRBF. In particular, TRIKE-Restart is better than AQUARS-CGRBF on 7 of the 18 test problems (Branin, Goldstein-Price, Hartman3, Shekel7, GWSC(5,8), Hartman6 and GWSS(6,8)) and it has comparable performance with the latter on
three more test problems (Shekel2A, Shekel3B and Shekel110). Note also that TRIKE-Restart is better than or comparable with AQUARS-CGRBF on six of the Dixon and Szegö (1978) problems (all except Shekel5). Since TRIKE-Restart is very competitive with AQUARS-CGRBF, it also compares favourably with the other alternatives that were compared to AQUARS-CGRBF in Regis and Shoemaker (2013).

In addition, Table 1 shows that TRIKE-Restart is much better than CYCLONE-Restart and the particular implementation of the latter is not as competitive with AQUARS-CGRBF as TRIKE-Restart. In particular, TRIKE-Restart finds a near optimal solution more quickly than CYCLONE-Restart on 14 of the 18 lower-dimensional problems (all except Branin, GWSS(3,10), Shekel3C and GWSS(6,8)). Moreover, CYCLONE-Restart is better than AQUARS-CGRBF on only 6 of the 18 test problems.

### 7.2. Effects of the localization and restart strategies

The performance improvements due to the localization and restart strategies in TRIKE-Restart and CYCLONE-Restart are consistent with the results obtained in Regis and Shoemaker (2007), wherein similar strategies are applied to the RBF method by Gutmann (2001). Regis and Shoemaker (2007) showed that, on some test problems, particularly those with deep and narrow global minimum basins such as the Shekel problems, the RBF method by Gutmann yields iterates that tend to be far from previous sample points and usually in the boundary of the search space. This puts too much emphasis on global search and not enough on local search, which is also important for quickly finding the global minimum. Closer examination of the trajectories of EGO on the Shekel problems suggest that a similar phenomenon is at work here. That is, the EI function tends to be maximized at the boundaries of the search space and far from previous sample points. TRIKE and CYCLONE enforce local search by requiring that the EI be maximized in smaller boxes centred around the current best point from time to time. If the current best point is not a local minimizer, then there must be a better solution in a small neighbourhood around that point and local search makes finding this better point easier.

Figure 1 shows the mean of the minimum distance of the current sample point from previous sample points after every function evaluation for EGO, CYCLONE and TRIKE on Shekel10 and GWSS(6,8). Figure C.1 of Appendix C (supplied in the online supplement—see the section ‘Supplemental data’ before the references list) shows corresponding plots for Shekel5, Shekel7, GWSC(5,8) and Hartman6. These plots show that EGO produces iterates that are farther from previous sample points compared with the iterates generated by CYCLONE or TRIKE on these problems, except Hartman6. This confirms that EGO puts too much emphasis on global search on these problems. Moreover, note that the cyclic pattern in the plots for CYCLONE is a result of the cyclic nature of the sizes of the search boxes where the EI is maximized. On the other hand, the sizes of the search boxes for TRIKE vary more gradually depending on how the algorithm is progressing.

Next, by implementing the localization strategy in TRIKE, the algorithm behaves more like a heuristic local search method and so it has a tendency to converge to a local minimum. Moreover, although CYCLONE balances global and local search by varying the size of the search region around the current best point in a cyclic manner, the algorithm could still take a long time to find the global minimum as can be seen from Table 1. Depending on the initial space-filling design chosen, the search could become biased towards certain regions early in the search, and a very large number of evaluations could be needed to drive the algorithm towards the region of the global minimum. From Table 1, the performance of TRIKE and CYCLONE (and also EGO) can be quite sensitive to the choice of the initial space-filling design and some designs lead to the global minimum more quickly than others. Performing a restart from scratch using
a new space-filling design when the EI threshold is reached possibly allows the algorithm to obtain a more favourable initial space-filling design that leads to the global minimum more quickly.

As explained in Section 4.2, an algorithm that can find a near-optimal solution with positive probability can benefit from multiple restarts since the number of optimization runs needed to find a near-optimal solution follows a geometric probability distribution. By recording the number of runs of either CYCLONE or TRIKE that are needed within CYCLONE-Restart or TRIKE-Restart to find a function value within 1% of the global minimum value for a given test problem, it is possible to estimate the success probability for this geometric distribution. In particular, if $T_1, T_2, \ldots, T_k$ are the random variables representing the number of runs to get within 1% of the global minimum value for $k$ trials of an algorithm $\mathcal{A}$ on a particular test problem, then the MLE of the success probability is $k / \sum_{i=1}^{k} T_i$. Table C.1 of Appendix C shows the MLEs for the success probabilities of CYCLONE and TRIKE (stopped when $\epsilon_{EI} < 0.001$) within CYCLONE-Restart and TRIKE-Restart, respectively. All that is required for Proposition 4.1 to guarantee that CYCLONE-Restart and TRIKE-Restart will find the global minimum in a probabilistic sense is for the success probability to be strictly positive, and this seems to be the case for both algorithms since the smallest value in Table C.1 is 0.17.

The benefits of the restart strategy are evident from Table 1. In particular, it is important on difficult problems with steep and narrow global minimum basins as can be seen from the performance of TRIKE and CYCLONE (with and without restart) on the Shekel problems. It is also important on problems with many local minima as can be seen from the results on the GWSS and GWSC problems. This is because TRIKE (without restart), and to a lesser extent CYCLONE (without restart), have a tendency to get trapped in a local minimum since their main strategies put more emphasis on local search. This results in relatively poor performance for these algorithms on problems with many local minima. On the other hand, EGO focuses more on global search. For example, Table 1 shows that EGO is better than CYCLONE and TRIKE on GWSS_Spiked and GWSC(4,10), both of which have many local minima. However, by performing restarts, TRIKE and CYCLONE are able to achieve results that are somewhat comparable to EGO on these problems. Moreover, the performances of EGO on GWSS(3,10), GWSC(5,8) and GWSS(6,8) suggest that its global strategy is not enough to locate the global minimum quickly. By doing restarts, both TRIKE and CYCLONE allow for the possibility of
using an initial space-filling design that can lead to the global minimum more quickly. With enough restarts, and provided the objective function is not terribly pathological (i.e. the basin of attraction of the global minimum is not very tiny), both TRIKE-Restart and CYCLONE-Restart will eventually locate this basin and find the global minimum with the help of their local search mechanisms.

7.3. Performance and data profiles

TRIKE and CYCLONE are also evaluated using performance and data profiles (Dolan and Moré, 2002; Moré, and Wild 2009). The profiles are created separately for the lower-dimensional and higher-dimensional problems since some of the algorithms are not run on the higher-dimensional problems. Let \( P \) be the set of problems for a given profile, where a given problem \( p \) corresponds to a particular test problem and a particular starting point. In this case, the starting point is the first point in the SLHD used to initialize all methods. For example, since there are 18 lower-dimensional optimization problems and 30 starting points (corresponding to the 30 trials), there are \( 18 \times 30 = 540 \) problems for the performance and data profiles for the lower-dimensional problems. Moreover, let \( S \) be the set of solvers. For the lower-dimensional problems, six solvers are used (EGO, CYCLONE, TRIKE, CYCLONE-Restart, TRIKE-Restart and AQUARS-CGRBF). For any pair \((p, s)\) of a problem \( p \) and a solver \( s \), the performance ratio is \( r_{p,s} = t_{p,s}/\min\{t_{p,s} : s \in S\} \), where \( t_{p,s} \) is the number of function evaluations required to satisfy the convergence test defined below. Obviously, \( r_{p,s} \geq 1 \) for any \( p \in P \) and \( s \in S \). Moreover, for a given problem \( p \), the best solver \( s \) for this problem attains \( r_{p,s} = 1 \). Furthermore, by convention, \( t_{p,s} = \infty \) whenever solver \( s \) fails to yield a solution that satisfies the convergence test.

Now, for any solver \( s \in S \) and for any \( \alpha \geq 1 \), the performance profile of \( s \) with respect to \( \alpha \) is the fraction of problems where the performance ratio is at most \( \alpha \) (Dolan and Moré, 2002), i.e. \( \rho_s(\alpha) = |\{p \in P : r_{p,s} \leq \alpha\}|/|P| \). Note that \( \rho_s(1) \) is the fraction of problems for which solver \( s \) is the best among all the solvers in \( S \). In general, algorithms that have higher values of the performance profiles are those that perform better than others on the problems in \( P \).

When function evaluations are computationally expensive, algorithms are typically compared given a fixed and relatively limited number of function evaluations. Hence, the convergence test proposed by Moré and Wild (2009) uses a tolerance \( \tau > 0 \) and the minimum function value \( f_L \) obtained by any of the solvers on a particular problem within a given number \( \mu_f \) of function evaluations, and it checks if a point \( x \) obtained by a solver satisfies \( f(x^{(0)}) - f(x) \geq (1 - \tau)(f(x^{(0)}) - f_L) \), where \( x^{(0)} \) is a starting point corresponding to the problem under consideration. Here, \( x \) is required to achieve a reduction that is \( 1 - \tau \) times the best possible reduction \( f(x^{(0)}) - f_L \). In the numerical comparisons below, \( \tau = 0.01 \).

Next, given a solver \( s \in S \) and \( \alpha > 0 \), the data profile of a solver \( s \) with respect to \( \alpha \) (Moré and Wild 2009) is given by \( d_s(\alpha) = |\{p \in P : t_{p,s}/(n_p + 1) \leq \alpha\}|/|P| \), where \( t_{p,s} \) is the number of function evaluations required by solver \( s \) to satisfy the above convergence test on problem \( p \) and \( n_p \) is the number of decision variables in problem \( p \). For a given solver \( s \) and any \( \alpha > 0 \), \( d_s(\alpha) \) is the fraction of problems for which the solver \( s \) generated a point satisfying the convergence test within \( \alpha(n_p + 1) \) function evaluations. As with performance profiles, algorithms with higher values of the data profiles are the better algorithms on the problems in \( P \).

Figure 2 shows the data and performance profiles of TRIKE and CYCLONE, with or without restarts, on the 18 lower-dimensional test problems. The performance profiles were obtained after 300 function evaluations while the data profiles were obtained up to the equivalent of 150 simplex gradient estimates. In these profiles, both TRIKE and CYCLONE are substantial improvements over EGO on the lower-dimensional test problems. Moreover, both profiles show that TRIKE-Restart and CYCLONE-Restart are, in turn, substantial improvements over TRIKE.
and CYCLONE. In particular, after the equivalent of about 100 simplex gradient estimates, the data profiles show that TRIKE-Restart and CYCLONE-Restart satisfy the convergence test for about 95% of the problems compared to about 65% of the problems for TRIKE and CYCLONE, and only about 50% of the problems for EGO. From the performance profiles after 300 function evaluations, the fraction of problems for which TRIKE-Restart is the best (given by the value of $\rho_s(1)$) is about 50% while the corresponding fractions for CYCLONE-Restart and TRIKE are around 30–40% and the corresponding fractions for CYCLONE and EGO are about 22 and 12%, respectively. Furthermore, the fraction of problems where the performance ratio is at most $\alpha = 3$ after 300 function evaluations is more than 85% for both TRIKE-Restart and CYCLONE-Restart while the corresponding fractions for TRIKE and CYCLONE are around 55–60% and the corresponding fraction for EGO is only about 35%. In summary, TRIKE-Restart and CYCLONE-Restart are both dramatic improvements over standard EGO and they are substantial improvements over TRIKE and CYCLONE. Note that these findings are consistent with the results of the comparisons in Section 7.1.

The data and performance profiles in Figure 2 also show slightly better performance for TRIKE-Restart and CYCLONE-Restart over AQUARS-CGRBF on the lower-dimensional test problems. Moreover, these profiles show that TRIKE-Restart is slightly better than CYCLONE-Restart on the lower-dimensional problems.

Figure 3 shows the data and performance profiles of TRIKE-Restart and CYCLONE-Restart on the higher-dimensional test problems. In both profiles, TRIKE-Restart is much better than CYCLONE-Restart, which in turn is also much better than standard EGO. In particular, after the equivalent of 16 simplex gradient estimates in the data profiles, TRIKE-Restart satisfies the convergence test on about 70% of the problems compared to about 20% of the problems for CYCLONE-Restart and only about 6% of the problems for EGO. Moreover, from the performance profiles after 500 simulations, the fraction of problems for which TRIKE-Restart is the best (given by the value of $\rho_s(1)$) is more than 75% while the corresponding fractions for CYCLONE-Restart and standard EGO are about 25 and 6%, respectively. Furthermore, the fraction of problems where the performance ratio is at most $\alpha = 1.5$ after 500 function evaluations is about 80% for TRIKE-Restart while the corresponding fractions for CYCLONE-Restart and standard EGO are around 30 and 6%, respectively.

In all, the data and performance profiles clearly show that TRIKE-Restart and CYCLONE-Restart are both substantial improvements over standard EGO on the 28 test problems. Moreover, they are competitive with AQUARS-CGRBF on the lower-dimensional problems. In addition,
Figure 3. Data and performance profiles calculated from running the algorithms on the 10 higher-dimensional test problems from 30 different starting points; $\tau = 0.01$. The test problems have at least 30 decision variables.

TRIKE-Restart is slightly better than CYCLONE-Restart on the lower-dimensional problems but the former is much better than the latter on the higher-dimensional problems.

7.4. Comparison of progress curves on the higher-dimensional problems

Figure 4 and Figure D.1 of Appendix D (supplied in the online supplement—see the section ‘Supplemental data’ before the references list) show the plots of the mean of the best objective function value obtained by each algorithm on GWB36 and on each of the higher-dimensional problems as the number of function evaluations increases. These plots are called average progress curves. The error bars are 95% $t$-confidence intervals for the mean. That is, the length of each side of the error bar is equal to 2.045 times the standard deviation of the best function value divided by the square root of the number of trials. The factor 2.045 is the critical value...
corresponding to a 95% confidence level for a $t$-distribution with 29 degrees of freedom. Since all algorithms start with the same set of initial points (an SLHD of size $2(d + 1)$) in each trial, all curves begin at the same height. The best algorithms correspond to the curves that go down faster compared to the others as the number of function evaluations increases.

When function evaluations are computationally expensive and the problem has many decision variables, average progress curves can also be used to compare the performance of algorithms. Note that finding the global minimum of a relatively high-dimensional objective function given a very limited computational budget is unrealistic. Hence, comparing the number of function evaluations required to reach the global minimum within some tolerance, as was done for the lower-dimensional problems, is not possible. Moreover, in this article, ‘computationally expensive’ means that the total running time of each algorithm is completely dominated by the time spent on function evaluations and so it makes sense to compare the performance of algorithms at a fixed and very limited number of function evaluations. For example, if each function evaluation of the Ackley function takes an hour, then the average progress curves provide a good idea of where each algorithm stands after 50 hours, 100 hours or at some other computational budget.

The average progress curves in Figure 4 and Figure D.1 of Appendix D show that both TRIKE-Restart and CYCLONE-Restart are consistently much better than EGO on all ten higher-dimensional test problems. However, on the groundwater application GWB36, TRIKE-Restart was substantially much better than EGO but CYCLONE-Restart did not show any real advantage over EGO. Moreover, the average progress curves show that TRIKE-Restart is better than CYCLONE-Restart on GWB36 and on seven of the higher-dimensional problems (all except Ackley, Keane and Levy). In summary, the average progress curves on the higher-dimensional problems show that TRIKE-Restart is generally better than CYCLONE-Restart and they are both generally better than standard EGO. Note that these results are consistent with the findings using data and performance profiles from the previous section.

7.5. Sensitivity analysis

The proposed algorithms require some user-specified control parameters and their performances might depend on the setting of these parameters. However, a full sensitivity analysis of each proposed algorithm is computationally demanding and is beyond the scope of this paper. Hence, this section focuses on the sensitivity of TRIKE-Restart and CYCLONE-Restart with respect to just one important control parameter. In practice, the other control parameters are simply set to reasonable values or to values commonly used by other algorithms that also use the same parameters.

In the case of TRIKE-Restart, some of its user-specified parameters are the same ones used by derivative-free trust region methods. For example, the contraction and expansion factors are typically set to $\gamma_0 = 1/2$ and $\gamma_1 = 2$, respectively. Moreover, the validity factor $C = 2$ is the setting used in the derivative-free trust region method by Oeuvray and Bierlaire (2009). This section performs sensitivity analysis of TRIKE-Restart with respect to the parameter $\eta$, which is the threshold for the ratio of the AI to the EI that is used to decide whether to increase the size of the trust region in a given iteration.

Figure 5 shows the data profiles for TRIKE-Restart with the three different values of $\eta$ on the 18 lower-dimensional test problems (Figure 5(a)) and on the 10 higher-dimensional test problems (Figure 5(b)). The values of $\eta$ include the default value of $\eta = 1$ used in the earlier comparisons. Performance profiles are also given in Figure E.1 of Appendix E (supplied in the online supplement—see the section ‘Supplemental data’ before the references list). The data and performance profiles on the test problems indicate that TRIKE-Restart is not sensitive to
the threshold ratio $\eta$ on the lower-dimensional problems. However, on the higher-dimensional problems, the setting $\eta = 0.5$ results in slightly worse performance compared to $\eta = 1$ or $\eta = 2$. Hence, TRIKE-Restart appears to be somewhat sensitive to this parameter on higher-dimensional problems. A possible explanation is that a lower ratio threshold will result in more frequent expansion of the trust region. A larger trust region tends to result in more global search, and in the higher-dimensional setting, this makes it less likely to find an improving solution.

Next, CYCLONE-Restart depends on the user-specified parameter Search Pattern, which consists of the predetermined lengths of the search boxes around the current best point as the algorithm goes through the iterations. To analyse how sensitive CYCLONE-Restart is to the search pattern, numerical experiments are also performed where this algorithm is run with two additional search patterns on the same test problems.

The search pattern used in the comparisons with alternatives is denoted by $\text{SP1} = (2, 0.4, 0.2, 0.1, 0.05, 0.02, 0.01)$. The two additional search patterns are $\text{SP2} = (2, 0.8, 0.4, 0.4, 0.2, 0.2, 0.1)$ and $\text{SP3} = (0.2, 0.1, 0.1, 0.05, 0.05, 0.02, 0.01)$. Note that SP2 is more global than the original search pattern SP1 since its search boxes are generally larger. On the other hand, SP3 is more local than SP1.

Figure 6 shows the data profiles for CYCLONE-Restart with the three different search patterns on the 18 lower-dimensional test problems (Figure 6(a)) and on the 10 higher-dimensional test problems (Figure 6(b)). Performance profiles are also given in Figure E.2 of Appendix E. The data and performance profiles on the test problems indicate that the search patterns with more local search (SP1 and SP3) are better than the more global search pattern SP2, especially on the lower-dimensional problems. These results provide further evidence of the importance of local search in improving the performance of Kriging-based methods that use the EI criterion.

### 7.6. Parallelization strategies

To improve the performance of the proposed algorithms on truly expensive problems further, parallelization strategies can also be implemented and this will be the subject of future work. For example, assuming that the time for each function evaluation is roughly constant, one parallelization strategy is to allow each method to generate multiple sample points within each iteration, as many as there are available processors. In the case of TRIKE or CYCLONE (with or without restarts), a natural choice is to use multi-points EI (or $q$-EI) (Ginsbourger, Le Riche, and Carraro, 2010) to generate multiple sample points within the current search box for simultaneous
function evaluation in parallel. Another possibility is to adopt the strategy by Viana et al. (2013) of using multiple surrogates (e.g. Kriging models with different covariance functions) to generate multiple sample points. By maximizing EI using different surrogates, the solutions obtained are potentially different. Moreover, in the case of CYCLONE (with or without restarts), the EI can be maximized within search boxes of different sizes in a single iteration, with some backup procedure in the event that the same sample points are obtained.

An alternative parallelization strategy is to perform multiple independent runs of either TRIKE or CYCLONE asynchronously on multiple processors, with or without communication, where each run uses a different initial space-filling design. As pointed out in Section 4.2, the performance of surrogate-based methods can be sensitive to the choice of the initial space-filling design, so these different runs could potentially yield very different optimization trajectories that explore different parts of the search space, especially when the dimension is high and the number of function evaluations is relatively limited.

8. Summary and conclusions

This article proposed the TRIKE and CYCLONE algorithms with restarts, which are both Kriging-based methods that select function evaluation points by maximizing the expected improvement function over a smaller region in the neighbourhood of the current best solution instead of the entire search space. The TRIKE algorithm uses a trust-region-like approach wherein the size of a box-shaped trust region varies adaptively depending on performance. That is, the trust region is expanded if the ratio of the actual improvement to the expected improvement is greater than some threshold. On the other hand, the trust region is reduced if there is no actual improvement and there are enough data points to fit a linear model within some neighbourhood of the current best point. The CYCLONE algorithm varies the size of the search box around the current best solution by cycling through a predetermined search pattern. These new algorithms are designed to address some of the performance issues of the well-known EGO algorithm on high-dimensional problems and on problems with deep and narrow global minimum basins. The proposed algorithms are tested on 28 test problems with dimensions ranging from 2 to 32 and on a 36-dimensional simulation-based groundwater application and are compared with a standard implementation of EGO and with AQUARS-CGRBF (Regis and Shoemaker 2013), which is an RBF method that has been shown to perform consistently well compared to many alternative methods.
The data and performance profiles of the algorithms for all test problems, the average progress curves for the groundwater application and the higher-dimensional problems, and the table of results showing the number of function evaluations to get within 1% of the global minimum value on the lower-dimensional problems, all consistently indicate that TRIKE-Restart and CYCLONE-Restart are both substantially much better than standard EGO on these problems. This shows that the localization strategies in these two algorithms yield substantial performance improvements for EGO. Furthermore, the results show an advantage of TRIKE-Restart over CYCLONE-Restart especially on the higher-dimensional problems. This suggests that the more adaptive trust region strategy for adjusting the sizes of the search regions for the next iterate is much better than a predetermined cycle of sizes for the search regions. In addition, the analysis of the sensitivity of CYCLONE-Restart to the search pattern and also the sensitivity of TRIKE-Restart to the ratio threshold $\eta$ provide further evidence that a more local search strategy yields better performance than a purely global search strategy when function evaluations are limited. In summary, both TRIKE-Restart and CYCLONE-Restart are very promising algorithms for expensive black-box optimization that extend the applicability of Kriging-based methods to problems with deep and narrow global minimum basins and also to high-dimensional problems.

Supplemental data

Supplemental data for this article can be accessed at http://dx.doi.org/10.1080/0305215X.2015.1082350.

References

Abramson, M. A.. 2007. NOMADm Version 4.6 User’s Guide. Department of Mathematics and Statistics, Airforce Institute of Technology, Wright-Patterson Air Force Base, OH. Unpublished manuscript.
Aleman, D. M., H. E. Romeijn, and J. F. Dempsey.. 2009. “A Response Surface Approach to Beam Orientation Optimization in Intensity Modulated Radiation Therapy Treatment Planning.” INFORMS Journal on Computing 21 (1): 62–76.
Annicchiarico, W.. 2007. “Metamodel-Assisted Distributed Genetic Algorithms Applied to Structural Shape Optimization Problems.” Engineering Optimization 39 (7): 757–772.
Audet, C., and J. E. Dennis Jr.. 2006. “Mesh Adaptive Direct Search Algorithms for Constrained Optimization.” SIAM Journal on Optimization 17 (2): 188–217.
Basudhar, A., C. Dribusch, S. Lacaze, and S. Missoum.. 2012. “Constrained Efficient Global Optimization with Support Vector Machines.” Structural and Multidisciplinary Optimization 46 (2): 201–221.
Booker, A. J., J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset.. 1999. “A Rigorous Framework for Optimization of Expensive Functions by Surrogates.” Structural Optimization 17 (1): 1–13.
Chen, L.-L., C. Liao, W. Lin, L. Chang, and X.-M. Zhong.. 2012. “Hybrid-Surrogate-Model-Based Efficient Global Optimization for High-Dimensional Antenna Design.” Progress in Electromagnetics Research 124: 85–100.
Cheng, G., and G. Wang. 2012. “Trust Region Based MPS Method for Global Optimization of High Dimensional Design Problems.” In 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 23–26 April 2012, Honolulu, HI, Paper No. AIAA 2012-1590. http://dx.doi.org/10.2514/6.2012-1590.
Couckuyt, I., D. Deschrijver, and T. Dhaene.. 2014. “Fast Calculation of Multiobjective Probability of Improvement and Expected Improvement Criteria for Pareto Optimization.” Journal of Global Optimization 60 (3): 575–594. http://dx.doi.org/10.1007/s10898-013-0118-2.
Dixon, L. C. W., and G. Szegö.. 1978. “The Global Optimization Problem: An Introduction.” In Towards Global Optimization 2, edited by L. C. W. Dixon and G. Szegö (1–15). Amsterdam: North-Holland.
Dolan, E. D., and J. J. Moré.. 2002. “Benchmarking Optimization Software with Performance Profiles.” Mathematical Programming 91 (2): 201–213.
Conn, A. R., K. Scheinberg, and L. N. Vicente.. 2009. Introduction to Derivative-Free Optimization. Philadelphia, PA: SIAM.
Cressie, N. 1993. Statistics for Spatial Data. New York: Wiley.
Egea, J. A., E. Vazquez, J. R. Banga, and R. Marti.. 2009. “Improved Scatter Search for the Global Optimization of Computationally Expensive Dynamic Models.” Journal of Global Optimization 43 (2-3): 175–190.
Emmerich, M. T. M., K. Giannakoglou, and B. Naujoks.. 2006. “Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels.” IEEE Transactions on Evolutionary Computation 10 (4): 421–439.
Forrester, A. I. J., A. J. Keane, N. Bressloff.. 2006. “Design and Analysis of ‘Noisy’ Computer Experiments.” AIAA Journal 44 (10): 2331–2339.
Viana, F. A. C.. 2010. *SURROGATES Toolbox User’s Guide, Version 2.1*. http://sites.google.com/site/felipeacviana/surrogatestoolbox.

Viana, F. A. C., R. T. Haftka, and L. T. Watson.. 2013. “Efficient Global Optimization Algorithm Assisted by Multiple Surrogate Techniques.” *Journal of Global Optimization* 56 (2): 669–689.

Villemonteix, J., E. Vazquez, and E. Walter.. 2009. “An Informational Approach to the Global Optimization of Expensive-to-Evaluate Functions.” *Journal of Global Optimization* 44 (4): 509–534.

Wang, G. G., and S. Shan.. 2007. “Review of Metamodelling Techniques in Support of Engineering Design Optimization.” *ASME Journal of Mechanical Design* 129 (4): 370–380.

Wild, S. M., R. G. Regis, and C. A. Shoemaker.. 2008. “ORBIT: Optimization by Radial Basis Function Interpolation in Trust-Regions.” *SIAM Journal on Scientific Computing* 30 (6): 3197–3219.

Wild, S. M., and C. A. Shoemaker.. 2013. “Global Convergence of Radial Basis Function Trust-Region Algorithms for Derivative-Free Optimization.” *SIAM Review* 55 (2): 349–371.

Won, K. S., and T. Ray.. 2005. “A Framework for Design Optimization Using Surrogates.” *Engineering Optimization* 37 (7): 685–703.

Ye, K. Q., W. Li, and A. Sudjianto.. 2000. “Algorithmic Construction of Optimal Symmetric Latin Hypercube Designs.” *Journal of Statistical Planning and Inference* 90 (1): 145–159.

Yoon, J. H., and C. A. Shoemaker.. 1999. Comparison of Optimization Methods for Ground-Water Bioremediation.” *Journal of Water Resources Planning and Management* 125 (1): 54–63.

Zhang, Q., W. Liu, E. Tsang, and B. Virginas.. 2010. “Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model.” *IEEE Transactions on Evolutionary Computation* 14 (3): 456–474.