Optimization and Engineering
https://doi.org/10.1007/s11081-024-09943-y

RESEARCH ARTICLE

SCR, an efficient global optimization algorithm for constrained black-box problems

Syed Ali Zaryab[1] · **Andrea Manno**[2] · **Emanuele Martelli**[1]

Received: 4 December 2023 / Revised: 24 July 2024 / Accepted: 19 November 2024

Abstract

This paper proposes a surrogate based, global-search derivative free algorithm which is specifically designed for computationally expensive black-box (simulation-based) optimization problems with constraints. The algorithm, called *SCR* (*Surrogate-CMAES-RQLIF*), uses kriging to generate surrogate models of the black-box objective function and black-box constraints. These surrogate models are optimized using the global-search algorithm *CMA-ES* with the quadratic penalty approach for the constraints. The quality of the kriging surrogates is checked, and the surrogates are updated at each iteration by adding the point found by *CMA-ES* and additional training points. Once the region of the global optimum has been approximately defined, local search is performed using the hybrid direct-search/model based algorithm *RQLIF*. After each iteration the points sampled by *RQLIF* and some additional points found within the optimal region are used to update the surrogate model. Tests on 25 unconstrained and 21 constrained literature test problems show that *SCR* outperforms benchmark optimization algorithms. The outstanding performance of *SCR* is also confirmed on two real-world black-box problems arising in process engineering with computationally expensive simulations: the techno-economic optimization of a CO_2 Purification Unit (CPU) and a Vacuum Pressure Swing Adsorption Unit (VPSA).

Keywords Derivative-free optimization · Surrogate-based optimization · Global optimization · Kriging · Process optimization

✉ Emanuele Martelli
emanuele.martelli@polimi.it

[1] Department of Energy, Politecnico Di Milano, Via Lambruschini 4, 20156 Milano, Italy

[2] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Via Vetoio, 67100 L'Aquila, Italy

Published online: 13 March 2025

1 Introduction

There are several optimization problems arising in engineering and economics in which the objective function and the problem constraints are computed by simulation codes (e.g., flow sheeting software, Computational Fluid Dynamics code, etc.). For such problems, the simulation code can be considered as a black box (Audet and Kokkolaras 2016) called by the optimization algorithm. Generally, simulation codes solve complex models using very specific algorithms. Some examples include finite element models in mechanical engineering (Besnerais et al. 2011; Meo and Zumpano 2008), Computational Fluid Dynamics (CFD) models in fluid dynamics (Madsen and Langthjem 2001; Olivero 2014); flowsheet models in process engineering (Martelli and Amaldi 2014; Zaryab et al. 2020), and systems of DAEs in control engineering (Peeters et al. 2018). These simulation-based functions are generally noisy (i.e., affected by the numerical noise) and lack gradient information. We are referring to the numerical noise generated by the internal numerical algorithms of the simulation code, i.e., variations of the output returned by the function not due to the variation of the input optimization variables but due to numerical issues of the simulation routines. For example, codes for process simulation (chemical and energy engineering) have iterative numerical algorithms (e.g., direct substitution) to guarantee convergence of the tearing variables to solve a fixed point problem (Biegler et al. 1997), to solve implicit systems of nonlinear equations within the process units, and to compute the properties of the streams (e.g., flash calculation Boston and Britt (1978)). Given the nonzero convergence tolerance of these numerical algorithms (e.g., 10 E-3), for fixed input optimization variables, there might be small variations of the simulation solution within such convergence tolerance. Additionally, there is the effect of the rounding errors which might propagate within the iterations, specially for ill-conditioned problems.

Moreover, simulation-based optimization problems are often computationally expensive. Thus, to tackle such black-box functions an efficient derivative-free algorithm is required which can find an (approximate) optimum within a limited number of function evaluations. This can be done by generating surrogate models of the black box functions.

In this paper we propose a novel derivative-free, surrogate-based global optimization algorithm called *SCR* which is well suited for black-box problems with general nonlinear constraints. *SCR* is designed to solve the following general constrained optimization problem.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(\boldsymbol{x}) \\ s.t. : \; & \boldsymbol{lb} \leq \boldsymbol{x} \leq \boldsymbol{ub} \\ & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \\ & \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}, \end{aligned}$$

here $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function, $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of decision variables, $\boldsymbol{lb}, \boldsymbol{ub} \in \mathbb{R}^n$ are the lower and upper bounds on $\boldsymbol{x}$, $\boldsymbol{g} : \mathbb{R}^n \rightarrow \mathbb{R}^p$ represents the inequality constraints, and $\boldsymbol{h} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ represents the equality constraints. It must be noted here that the functions involved in the problem statement may be nonlinear and

nonconvex, either black-box functions (e.g., computed by simulation codes) or algebraic. Moreover, the black-box functions might be non-smooth and computationally expensive.

The novelty of *SCR* lies in the following two main features:

(1) Its search procedure combines the use of kriging surrogate (Lophaven et al. 2002), the global search algorithm *CMA-ES* (Hansen and Kern 2004) and the novel local search algorithm *RQLIF* (Manno et al. 2020) within an effective adaptive algorithm, as detailed in Sect. 3.
(2) *SCR* creates separate surrogate models for the black box objective function and each black-box constraint. As shown in Sect. 4, using separate surrogate models for the objective function and the black-box constraints improves appreciably the performance of the optimizer.

Although the *SCR* algorithm is heuristic and its convergence to a local or global optimum is not investigated, in the extensive tests reported in this paper it shows to be competitive compared to some other state-of-the-art derivative free algorithms. The paper is organized as follows. Section 2 reviews the available derivative-free algorithms while Sect. 3 presents the *SCR* algorithm. Section 4 shows the results of the tests performed to benchmark *SCR* against well-known derivative free global algorithms, namely the hybrid algorithm implemented in the latest *NOMAD* software (Audet and Dennis 2006), *CMA-ES* (Hansen and Kern 2004) and *RBFOPT* (Costa and Nannicini 2018). Sections 5 and 6 concern the application of *SCR* to two real-world process engineering problems: (i) the techno-economic optimization of a CO_2 purification unit (CPU), and (ii) the techno-economic optimization of a Vacuum Pressure Swing Adsorption system (VPSA).

2 Overview of derivative free algorithms

Derivative free optimization (*DFO*) algorithms have been widely used to solve many scientific (Abramson et al. 2008; Deming et al. 1974; Gray et al. 2004), medical (Marsden et al. 2008; Oeuvray 2005) and engineering design problems (Audet et al. 2008a; Bartholomew-Biggs et al. 2002; Fowler et al. 2008), where the derivative information is unavailable, unreliable or hard to obtain (computationally expensive). The development of *DFO* algorithms started with Spendley et al. (Spendley et al. 1962) along with Nelder and Mead (Nelder and Mead 1965) who introduced simplex based algorithms, and continued over time with the design of many *DFO* algorithms.

All these *DFO* algorithms can be divided into two categories: (i) direct search and (ii) model based. The direct search methods explore the search domain by evaluating the objective function at points determined along a set of search directions (Hooke and Jeeves 1961; Kolda et al. 2003; Audet and Hare 2017; Conn et al. 2009). Model-based algorithms create a surrogate model of the objective function which is used to help in the search process (Audet and Hare 2017; Conn et al. 2009). Algorithms can also be classified as stochastic and deterministic according to the potential adoption of random search strategies, however in the section we will focus on the direct search/model-based categorization.

2.1 Direct search algorithms

One of the first *DFO* algorithms was the Nelder-Mead simplex algorithm which is a direct local search method. This algorithm (Nelder and Mead 1965), starting from a set of starting points generating a simplex, iteratively generate new candidates to improve the best current solution by replacing the simplex point with lowest objective function value with a new one determined on the line from such point to the centroid of the simplex. Over the years many variants of the Nelder-mead simplex algorithm have been proposed in the literature (e.g., Kelley (1999); Tseng (1999); Conn et al. (2009)). A software implementation of the *DFO* algorithms called *DFO* Software package (Conn et al. 1997; Conn 1998) is available online. This software package works well with relatively small-scale optimization problems with a number of variables smaller than 50. *DFO* builds a quadratic model around the current best solution, and the model is then optimized with a trust region approach.

Kolda et al. (2003) coined the term generating set search (*GSS*) for a large portion of the local direct search algorithms. These algorithms can be distinguished by their two main steps: a search step and a poll step. The search step evaluates points on a finite number of search direction starting from the current iterate with the goal of improving it. If the search step fails, the poll step is performed. By evaluating points on a set of generating directions which positively span $\mathbb{R}^n$. Assuming smoothness of the objective function, this process guarantees that the set of generating directions includes at least one decent direction at a non-stationary point. One of the most well known *GSS* is the Hooke and Jeeves method (Hooke and Jeeves 1961), in which a search step is used to exploit the promising directions of improvement of a successful poll step.

In the Mesh Adaptive Direct Search (*MADS*) method (Audet and Dennis 2006) the poll step of the *GSS* algorithm is further modified so that it can consider a variable set of poll directions whose union across all iterations is asymptotically dense in $\mathbb{R}^n$. In *MADS* poll points are generated using the poll size parameter and the mesh size parameter. The poll size parameter defines a region in which the points can be selected while the mesh size parameter creates a grid inside this region. *MADS* adopts dynamic ordering, i.e. precedence is given to previously successful poll directions. Many *MADS* variants are available according to the type of poll directions: *LTMADS* (Audet and Dennis 2006) considers randomly generated poll directions, whereas in *ORTHOMADS* (Abramson et al. 2009) orthogonal poll directions are generated in a deterministic way. *MADS* is also equipped with a variable neighborhood search (VNS) metaheuristic strategy (Audet et al. 2008b), which helps to escape from local optima. In *MADS* constraint handling can be done with the extreme barrier or progressive approaches (Audet and Dennis 2006). In the former, trial points are rejected from consideration, while in the latter the infeasible trial points are allowed but with an infeasibility threshold. *MADS* is shown in Audet and Dennis (2006) to be one of the best performing *GSS* algorithms as for a number of problems where other *GSS* methods stagnates, *MADS* is able converges to an optimal solution, furthermore it is particularly able to escape from saddle points (see Abramson and Audet (2006)). The *MADS, LTMADS* and *ORTHOMADS* methods are implemented in the *NOMAD* (Abramson

et al. 2024) software package (available in Matlab through the OPTI toolbox Currie, (2020)). The *NOMAD* software also offers a quadratic surrogate model that serves two purposes: guiding the search process and prioritizing search directions. Additionally, *NOMAD* can be equipped with a global search strategy by enabling the VNS (Variable Neighborhood Search) option. In this work *NOMAD* will be used in the experimental comparison to assess the performance of the *SCR* algorithm.

Other direct search methods can be divided into dividing rectangles, population-based (e.g., evolutionary algorithms), random search and hybrid algorithms. The *DIRECT* algorithm (Jones et al. 1993) is a global direct search algorithm, created as an extension of the Lipschitzian-based method for bound constrained problems. The major drawback of *DIRECT*, like for other branch and bound type approaches, is the large number of function evaluations resulting in large computational time.

A different approach concerns stochastic algorithms. There are many variants of stochastic algorithms, as their implementation is often straightforward compared to deterministic ones. Hit and run algorithms are a very simple type of stochastic algorithms proposed by Boneh and Golan (Failed 1979) and (Smith 1984), where at each iteration a candidate is generated based on a direction and a step. This candidate is then compared with the current iterate. If the new candidate is better, then the current iterate is updated.

Genetic algorithms or evolutionary algorithms, introduced by Holland (Holland 1975), are based on the principle of natural selection and survival of the fittest. These algorithms have a fitness function based on which the individuals of the population adapt and mutate in each iteration. Several techniques have been proposed to adapt the covariance matrix which is used to sample new points in genetic algorithms, one of these techniques is called the Covariance Matrix Adaptation (*CMA-ES*) proposed by Hansen (Hansen and Kern 2004). In this method the covariance matrix is adapted deterministically from the last move of the algorithm. In *SCR*, *CMA-ES* (Hansen and Kern 2004) is used to find the global minimum of the surrogate model which has been created using *DACE* toolbox (Lophaven et al. 2002).

Finally, in particle swarm algorithms, pioneered by Kennedy and Eberhart (Caballero and Grossmann 2008; Boukouvala and Floudas 2016), a swarm of particles each with a velocity vector are produced at each iteration by using rules considering some parameters like inertia, cognition, and social, and by assigning to particles some randomly generated weights.

2.2 Model based algorithms

In the model-based approach, the algorithms create and optimize a high-fidelity surrogate model to guide the optimization process of the real functions. Local-search algorithms, like *NEWUOA* (Powell 2006; Lucidi and Sciandrone 2002), typically use quadratic models of the black-box function while global-search algorithms, like *RBFOPT* (Costa and Nannicini 2018) and *EGO* (Jones et al. 1998) use more complex models capable of approximating the function over the whole search space. In case of response surface methods (*RSMs*) the objective function f is approximated by a response surface f' (Barton 1994). There are two types of response surfaces: (i)

non-interpolating and (ii) interpolation (Jones 2001). The non-interpolation response surfaces are generally low order polynomials which are created by minimizing the sum of square deviation between f and f'. While the interpolating response models are functions which pass through the sampled responses of the objective function. Kriging and radial basis functions (*RBFs*) are both examples of the interpolating response models. Since engineering simulation models might be highly nonlinear and cannot be replicated by a low order polynomial function, we will only consider interpolating response models.

Kriging methods (Matheron 1967) are generally used in geostatistics, these methods model deterministic response as the realization of a stochastic process by means of a kriging basis function. Whereas *RBFs* can approximate a function by creating interpolating models based on radial functions. Powell (1987) introduced the use of *RBFs* for unconstrainted derivative free optimization. Through the years, numerous *RBF* strategies have been proposed for unconstrained and bound-constrained optimization problems, such as the *ARBF* algorithm (Holmström et al. 2008*),* the *Gutmann-RBF* (Gutmann 2001) and their improvements, like the *CG-RBF* algorithm (Regis and Shoemaker 2007*)*. On the other hand, only a limited number of algorithms have been proposed for problems with nonlinear constraints. Regis and Shoemaker (Regis and Shoemaker 2005) extended their *RBF* algorithm for problems with known constraints (not black-box). The algorithm, called *CORS-RBF*, proceeds by selecting the next sampling point as the one that minimizes the current response surface model subject to the given constraints and to additional constraints that the point be of some distance from previously evaluated points. Caballero and Grossmann (Caballero and Grossmann 2008) developed a kriging-based algorithm specifically for optimizing modular processes. The black-box modules of the process (which return the objective function and a set of constraints) are approximated using kriging functions and then the optimization problem is solved as a nonlinear program using *SNOPT*. Boukouvala and Floudas proposed *ARGONAUT* (Boukouvala and Floudas 2016), a general-purpose surrogate-based global optimization algorithm for problems with both algebraic and black-box constraints. For the black box functions and each of the black-box constraints the model selects the best surrogate function from a set of predefined functions (linear, quadratic, Kriging, etc.). The algorithm uses also variable selection, bound tightening and constrained sampling techniques.

2.3 Hybrid algorithms

Many hybrid direct search/model based algorithms are available in the literature. One of them is *RQLIF* proposed by Manno et al. (Manno et al. 2020). *RQLIF* is a local search derivative free algorithm exploiting regularized quadratic and linear implicit filtering models to cope with the possible noise in the black box function and to obtain good quality solutions within relatively few function evaluations. The quadratic and linear models used to accelerate convergence are constructed with the objective function local information collected during the iterations.

Multilevel coordinate search (*MCS*) (Huyer and Neumaier 1999) is a hybrid direct search algorithm which, similarly to *DIRECT*, divides the search space into boxes but,

differently from *DIRECT*, the base evaluation point can be located anywhere in the corresponding box (not necessarily in the center). To accelerate convergence, at each iteration *MCS* uses a local search algorithm which creates a quadratic model of the system using base points. This model is then minimized on a suitable box to find a promising search direction.

PGSCOM (Martelli and Amaldi 2014) is another example of hybrid algorithm which combines the positive features of Constrained Particle Swarm, Generating Set Search and Complex. This algorithm is designed for non-differentiable and discontinuous black-box problems with linear and nonlinear relaxable and unrelaxable constraints.

Another example of effective hybrid algorithm is the latest version of the *NOMAD* software package (Digabel et al. 2021) which, by default, integrates the orthoMADS polling step with both VNS for global search and quadratic models. The quadratic models are used as ordering criterion for the search directions and as search strategy.

3 SCR algorithm

The *SCR* algorithm has been devised to solve black box optimization problems where involved functions are generally noisy, lack gradient information, and contain black-box and algebraic nonlinear inequality constraints. The block flow diagram of *SCR* is provided in Fig. 1. During the initialization phase of the *SCR* algorithm, Latin

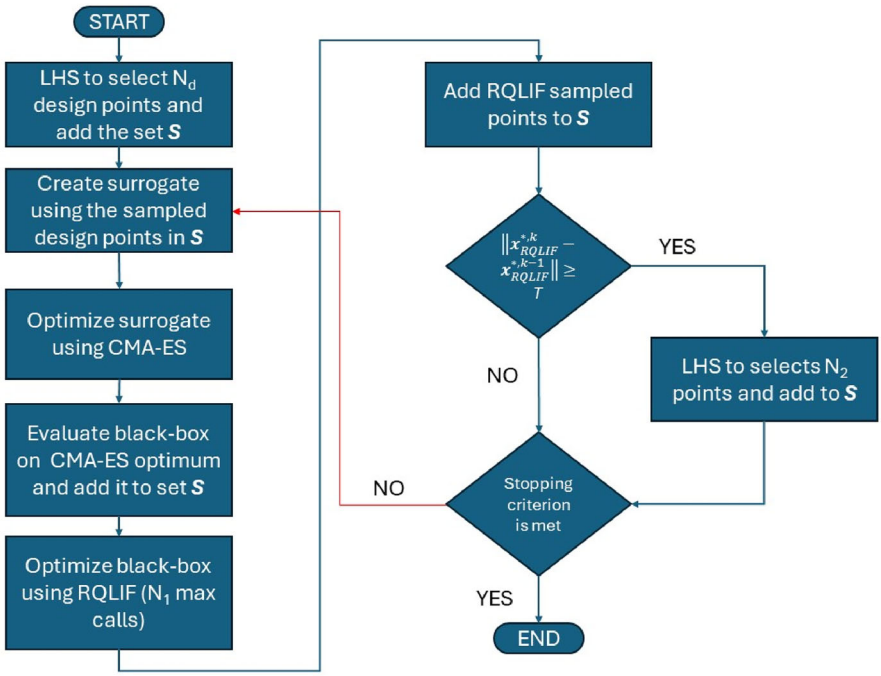

Fig. 1 Block flow diagram of *SCR* algorithm

hypercube sampling is used to generate N_d random input points (notice that N_d must be greater or equal than the minimum number of test sites required by the *DACE* toolbox to create the surrogate model, $(n+1) * (n+2)/2$, where n is the number of optimization variables). The black box involved functions are sampled at these set of input points and the samples are stored. By using these samples, *SCR* creates separate Kriging based surrogate models of the objective function and both its nonlinear and linear constraints using the *DACE* toolbox (Lophaven et al. 2002). *SCR* then calls *CMA-ES* to find the global minimum of the constrained surrogate model. The surrogate models of the constraints and objective function are combined in *CMA-ES* using the quadratic penalty approach. If the optima found by *CMA-ES* in two consecutive iterations are close to each other then it raises a need of performing a local search in that region. Then the local search algorithm *RQLIF* is called with a limit of N_1 black-box function evaluations. According to our computational tests (reported in Sect. 4), good performance can be achieved by setting $N_1 = 3n$. After the *RQLIF* run is terminated, the surrogate models for the constraints and the objective function are updated by adding the points tested by *RQLIF*. If the new best point found by *RQLIF* ($\boldsymbol{x}^{*,k}_{RQLIF}$) is distant from the one found in the previous iteration (i.e., RQLIF is exploring a new zone of the feasible region, indicating that the global search is still ongoing), N_2 additional points are generated by the Latin hypercube and sampled. Our tests (shown in Sect. 4) show good results with $N_2 = 3n$. The *SCR* algorithm stops whenever one of the following stopping criteria is met (i) *RQLIF* reaches the convergence tolerance on the search step length, (ii) the maximum number of black-box evaluations is reached. If the stopping criteria is not met the algorithm begins a new iteration by calling *CMA-ES* again to find the global minimum of the constrained surrogate model.

3.1 Latin hypercube sampling

Latin hypercube sampling (*LHS*) is a statistical sampling method which is used to produce a random sample of parameter values from a multidimensional dataset. *LHS* is generally used for Monte Carlo Simulations. A Latin square is a square grid where there is only one sample that is selected for each row and each column. Whereas a cube which has more than three dimensions is called a Hypercube. Thus, if we generalize the concept of a Latin square on a hypercube we get a Latin Hypercube Sampling where there exists only one sample in each axis-aligned hyperplane containing it.

3.2 Kriging surrogate model

Kriging is a widely used method for creating deterministic computer models. It is a form of Gaussian process regression which originated in the area of geostatistics. This method is used to interpolate data based on Gaussian process governed by prior covariances. In short Kriging is a generalized linear regression model which is able to provide a Best Linear Unbiased Prediction at unsampled points by taking into account the relation between the residuals of the regression model and the known data. The

unique characteristic of Kriging is its ability to combine both global and local modeling to provide results.

The Kriging model can be expressed as follows:

$$\hat{y}(\boldsymbol{x}) = \sum_{j=1}^{p} \beta_j f_j(\boldsymbol{x}) + z(\boldsymbol{x}) = \mathbf{f}^T(\boldsymbol{x})\boldsymbol{\beta} + \boldsymbol{z}(\boldsymbol{x})$$

This equation can be divided into two parts. The first part is a regression model which is a combination of p chosen functions $\mathbf{f} = [f_1, f_2, \ldots, f_p]$. This helps to model the drift of the process mean. Here the coefficients $\boldsymbol{\beta}$ are the regression parameters. The second part of the equation $z(\boldsymbol{x})$ is a random process (stochastic process) which is assumed to have zero mean and the following covariance:

$$Cov\left[z(\boldsymbol{x}_i), z\left(\boldsymbol{x}_j\right)\right] = \sigma^2 \mathcal{R}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$$

where σ^2 is the process variance and $\mathcal{R}$ is the spatial correlation function. The spatial correlation function is responsible for the smoothness of the model based on the nearby points. The Gaussian function is generally used as the spatial correlation function for most cases.

Consider a black box function $y = b(\boldsymbol{x})$ where for n design sites $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n)$ the output of the function are known, thus $\boldsymbol{y} = (y_1, y_2, y_3 \ldots, y_n) = [b(\boldsymbol{x}_1), b(\boldsymbol{x}_2), b(\boldsymbol{x}_3) \ldots, b(\boldsymbol{x}_n)]$. Here $\boldsymbol{x}$ is a vector which contains the input parameters for the black box function. The best linear unbiased predictor of this black box function at an unknown input location $\boldsymbol{x}^*$ is provided by the following equation:

$$\hat{y}\left(\boldsymbol{x}^*\right) = \mathbf{f}^T\left(\boldsymbol{x}^*\right)\hat{\boldsymbol{\beta}} + \boldsymbol{r}^T\left(\boldsymbol{x}^*\right)\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right)$$

where $\mathbf{f}\ (\boldsymbol{x}^*)$ is a vector of all the linear regression functions computed at $\boldsymbol{x}^*$

$$\mathbf{f}(\boldsymbol{x}^*) = [f_1(\boldsymbol{x}^*),\ f_2(\boldsymbol{x}^*), \ldots,\ f_p(\boldsymbol{x}^*)]^T$$

$\boldsymbol{r}(\boldsymbol{x}^*)$ is the correlation of $\boldsymbol{x}^*$ with the design sites $\boldsymbol{X}$

$$\boldsymbol{r}\left(\boldsymbol{x}^*\right) = \left[\mathcal{R}\left(\boldsymbol{x}^*, \boldsymbol{x_1}\right), \mathcal{R}\left(\boldsymbol{x}^*, \boldsymbol{x}_2\right), \ldots, \mathcal{R}\left(\boldsymbol{x}^*, \boldsymbol{x_n}\right)\right]^T$$

$\boldsymbol{R}$ is the correlation matrix of design sites $\boldsymbol{X}$

$$\boldsymbol{R} = \begin{bmatrix} \mathcal{R}(\boldsymbol{x_1}, \boldsymbol{x_1}) & \mathcal{R}(\boldsymbol{x_1}, \boldsymbol{x_2}) & \cdots & \mathcal{R}(\boldsymbol{x_1}, \boldsymbol{x_n}) \\ \mathcal{R}(\boldsymbol{x_2}, \boldsymbol{x_1}) & \mathcal{R}(\boldsymbol{x_2}, \boldsymbol{x_2}) & \ldots & \mathcal{R}(\boldsymbol{x_2}, \boldsymbol{x_n}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{R}(\boldsymbol{x_n}, \boldsymbol{x_1}) & \mathcal{R}(\boldsymbol{x_n}, \boldsymbol{x_2}) & \cdots & \mathcal{R}(\boldsymbol{x_n}, \boldsymbol{x_n}) \end{bmatrix}$$

$\boldsymbol{F}$ is the matrix with the set of regression function that have been calculated at n design sites

$$\boldsymbol{F} = \begin{bmatrix} f_1(\boldsymbol{x_1}) & f_2(\boldsymbol{x_1}) & \cdots & f_p(\boldsymbol{x_1}) \\ f_1(\boldsymbol{x_2}) & f_2(\boldsymbol{x_2}) & \ldots & f_p(\boldsymbol{x_2}) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\boldsymbol{x_n}) & f_1(\boldsymbol{x_n}) & \cdots & f_p(\boldsymbol{x_n}) \end{bmatrix}$$

And $\widehat{\boldsymbol{\beta}}$ is the least square estimate of $\boldsymbol{\beta}$

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{y}$$

The surrogate models of *SCR* are created using *DACE* kriging toolbox (Lophaven et al. 2002). This toolbox constructs a kriging approximation surrogate for a given set of design sites (inputs) and respective responses (outputs) of a model. These kriging models are deterministic therefore for each repeated run with the same input parameters these models would provide the same results. The pseudo codes for creating a kriging surrogate model and prediction of unknown point are given below:

The Kriging Surrogate Model Algorithm

1: **Input:** Design site $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, .. \boldsymbol{x}_n)$ and their responses by the black box function $\boldsymbol{y} = (y_1, y_2, y_3 ..., y_n)$

2: Initialize $\boldsymbol{R}$, $\boldsymbol{F}$, $\mathbf{f}$, $\boldsymbol{r}$, $\widehat{y}$

3: Evaluate: $\boldsymbol{R} = \begin{bmatrix} \mathcal{R}(\boldsymbol{x_1}, \boldsymbol{x_1}) & \mathcal{R}(\boldsymbol{x}_{varvec1}, \boldsymbol{x_2}) & \cdots & \mathcal{R}(\boldsymbol{x_1}, \boldsymbol{x_n}) \\ \mathcal{R}(\boldsymbol{x_2}, \boldsymbol{x_1}) & \mathcal{R}(\boldsymbol{x_2}, \boldsymbol{x_2}) & \ldots & \mathcal{R}(\boldsymbol{x_2}, \boldsymbol{x_n}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{R}(\boldsymbol{x_n}, \boldsymbol{x_1}) & \mathcal{R}(\boldsymbol{x_n}, \boldsymbol{x_2}) & \cdots & \mathcal{R}(\boldsymbol{x_n}, \boldsymbol{x_n}) \end{bmatrix}$

4: Evaluate: $\boldsymbol{F} = \begin{bmatrix} f_1(\boldsymbol{x_1}) & f_2(\boldsymbol{x_1}) & \cdots & f_p(\boldsymbol{x_1}) \\ f_1(\boldsymbol{x_2}) & f_2(\boldsymbol{x_2}) & \ldots & f_p(\boldsymbol{x_2}) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\boldsymbol{x_n}) & f_1(\boldsymbol{x_n}) & \cdots & f_p(\boldsymbol{x_n}) \end{bmatrix}$

5: **Output:** A struct called dmodel with $\boldsymbol{R}$ and $\boldsymbol{F}$ as elements

	The Kriging Predictor Algorithm
1:	**Input:** Unknown design site location **x*** and the surrogate model struct dmodel
2:	Evaluate: $\mathbf{f}(\boldsymbol{x}^*) = [f_1(\boldsymbol{x}^*), f_2(\boldsymbol{x}^*), ..., f_p(\boldsymbol{x}^*)]^{\mathrm{T}}$
3:	Evaluate:$\boldsymbol{r}(\boldsymbol{x}^*) = [\mathcal{R}(\boldsymbol{x}^*, \boldsymbol{x_1}), \mathcal{R}(\boldsymbol{x}^*, \boldsymbol{x_2}), \ldots, \mathcal{R}(\boldsymbol{x}^*, \boldsymbol{x_n})]^{\mathrm{T}}$
4:	Evaluate:$\widehat{y}(\boldsymbol{x}^*) = \mathbf{f}^{\boldsymbol{T}}(\boldsymbol{x}^*)\widehat{\boldsymbol{\beta}} + \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{x}^*)\boldsymbol{R}^{-\mathbf{1}}(\boldsymbol{y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}})$
5:	**Output:** $\widehat{y}(\boldsymbol{x}^*)$ the kriging prediction at unknown point $\boldsymbol{x}^*$

These two algorithms are separated as the first one requires a lot of computing power. Thus, it is wise to just evaluate the $\boldsymbol{R}$ and $\boldsymbol{F}$ of the black box function once and then just use these values when a prediction of an unknown design point is required.

3.3 CMA-ES

CMA-ES is a derivative free numerical optimization strategy for non-linear non-convex black-box optimization which is based on gaussian mutation (Hansen and Kern 2004). In evolution strategies (*ES*) new candidates are generated after the mutation of previous generation. In *CMA-ES* the new candidates for the next generation $(g+1)$ are sampled as:

$$\boldsymbol{x}_k^{(g+1)} \sim \boldsymbol{m} + \sigma^{(g)}\mathcal{N}\left(\mathbf{0}, \boldsymbol{C}^{(g)}\right) k = 1, \ldots \lambda.$$

where $\mathcal{N}$ is a multivariate normal distribution, $\boldsymbol{m} \epsilon \mathbb{R}^n$ is the mean, σ is the step-size and $\boldsymbol{C} \epsilon \mathbb{R}^{nxn}$ is the covariance matrix which is symmetric and positive definite. In the *ES* $\boldsymbol{m}$, $\boldsymbol{C}$ and σ are very important parameters as the mean is the center of the distribution, the covariance matrix dictates the direction in which the sampling should take place while the step size controls the sampling radius of the algorithm. *CMA-ES* ranks λ candidates in ascending order based on their fitness and then selects a μ best candidates. Here the mean $\boldsymbol{m}$ is updated based on weighted recombination of the selected candidates as shown below:

$$\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} \omega_i \boldsymbol{x}_{\boldsymbol{i}:\lambda}^{(g+1)}$$

where $\boldsymbol{x}_{\boldsymbol{i}:\lambda}^{(g+1)}$ represents the i^{th} ranked individual of the λ candidates of the generation $g+1$. It must be noted that all the weight are positive, and their combined sum is 1 as shown below:

$$\sum_{i=1}^{\mu} \omega_i = 1, \omega_1 \geq \omega_2 \geq \ldots \geq \omega_\mu \geq 0$$

Furthermore,

$$\omega_i = \log\left(\frac{\lambda - 1}{2} + 1\right) - \log(i)$$

The step size σ is updates with the following formula:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left|\left|\boldsymbol{p}_\sigma^{(g+1)}\right|\right|}{\mathbb{E}[||\mathcal{N}(\mathbf{0}, \boldsymbol{I})||]}\right) - \sqrt{\gamma_\sigma^{(g+1)}}\right)$$

here d_σ is the damping factor of the step size update, c_σ is the cumulation factor, $\mathbb{E}[||\mathcal{N}(\mathbf{0}, \boldsymbol{I})||]$ is the expected norm of the n-variate standard normal distribution, $\boldsymbol{p}_\sigma$ is the evolution path for the step size adaptation and γ_σ is the normalizing factor for the step size evolution path. An extensive discussion on all these factors can be found in Hansen (2007).

The equation of the covariance matrix for the next generation $(g + 1)$ is shown below, where both the rank-one and rank-μ update are combined.

$$\boldsymbol{C}^{(g+1)} = (1 - c_{\text{cov}})\,\boldsymbol{C}^{(g)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}}\boldsymbol{p}_{\boldsymbol{c}}^{(g+1)}\boldsymbol{p}_{\boldsymbol{c}}^{(g+1)^T} + c_{\text{cov}}\left(1 - \frac{1}{\mu_{\text{cov}}}\right)\sum_{i=1}^{\mu}\omega_i \boldsymbol{y}_{\boldsymbol{i}:\lambda}^{(g+1)}\left(\boldsymbol{y}_{\boldsymbol{i}:\lambda}^{(g+1)}\right)^T$$

The equation above is the sum of three parts. The first part is the update of old covariance matrix with the learning rate c_{cov} of the covariance matrix update (where $c_{\text{cov}} \in [0, 1]$). The second part is the rank-one update which used the evolution path $\boldsymbol{p_c}$ to compute the change of the mean overtime. The last part is the rank-μ update which focuses on the good variations of the last generation. The pseudo code for the *CMA-ES* algorithm used in *SCR* is given below:

	The *CMA-ES* Algorithm
1:	**Input:** Objective function Θ, maximum evaluations Ev
2:	Initialize $\mathbf{m}$, σ, λ, μ, $\mathbf{C} = \mathbf{I}$, $\mathbf{p_\sigma} = 0$, $\mathbf{p_c} = 0$
3:	**While** total surrogate function evaluations < Ev
4:	Updating the population:$\boldsymbol{x_k} = \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \boldsymbol{C}), k = 1, \ldots \lambda$
5:	Evaluating objective function: $\Theta\,(x_k), k = 1, \ldots \lambda$
6:	Mean update:$\boldsymbol{m} = \sum_{i=1}^{\mu}\omega_i \boldsymbol{x}_{\boldsymbol{i}:\lambda}$
7:	Step size update:$\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathbb{E}[\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|]}\right) - \sqrt{\gamma_\sigma}\right)$

	The *CMA-ES* Algorithm
8:	Covariance matrix update: $\boldsymbol{C} \leftarrow (1 - c_{\text{cov}})\boldsymbol{C} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} \boldsymbol{p_c} \boldsymbol{p_c^T} + c_{\text{cov}}\left(1 - \frac{1}{\mu_{\text{cov}}}\right) \sum_{i=1}^{\mu} \omega_i \boldsymbol{y_{i:\lambda}} \boldsymbol{y_{i:\lambda}^T}$
9:	**Select** best candidate $\boldsymbol{x^*_{CMAES}}$ amongst $\boldsymbol{x_k}$
10:	**End While**
11:	**Output**: Best found solution $\boldsymbol{x^*_{CMAES}}$ and $\Theta\left(\boldsymbol{x^*_{CMAES}}\right)$

3.4 RQLIF

RQLIF (Manno et al. 2020) is a hybrid direct search model-based method, which exploits a regularized quadratic model and an implicit filtering linear model to accelerate convergence and to cope with possible noise. *RQLIF* consists of the alternation of three main steps:

1. A direct step based on multiple local samples of the objective function f in a neighborhood of the current iterate $\boldsymbol{x^k}$,
2. A quadratic step in which the function evaluation is performed at the minimizer of an appropriate quadratic model of f,
3. A linear step in which f is evaluated at points located on an approximated steepest descent direction.

	The *RQLIF* Algorithm
1:	**Input:** starting point $\boldsymbol{x}^0_{RQLIF}$, maximum evaluations E, tolerance on minimum size of step length ϵ, black box function f
2:	Initialize: $F^*_{RQLIF} = f(\boldsymbol{x}^0_{RQLIF})$
3:	**While** total black-box function evaluations $< E$ *and* $\alpha^k > \epsilon$
4:	Step1: Perform direct search along set of directions and return the best solution found $\boldsymbol{x}^D_{RQLIF}$ and F^D_{RQLIF} and the step length α^k
5:	Step2: Optimize a quadratic model of f created from sampled points and return $\boldsymbol{x}^Q_{RQLIF}$ and F^Q_{RQLIF}
6:	Update: $F^*_{RQLIF} = \min(F^*_{RQLIF}, F^D_{RQLIF}, F^Q_{RQLIF})$ and update $\boldsymbol{x}^*_{RQLIF}$ and α^k
7:	**If** Step1 and Step2 fail to improve F^*_{RQLIF}

	The *RQLIF* Algorithm
8:	Step3: Perform black box function evaluation along approximated steepest descent direction and return $\boldsymbol{x}^{L}_{RQLIF}$ and F^{L}_{RQLIF}
9:	Update: F^{*}_{RQLIF} =min(F^{*}_{RQLIF},F^{L}_{RQLIF})
10:	**End If**
11:	**End While**
12:	**Output**: Best solution found $\boldsymbol{x}^{*}_{RQLIF}$ and F^{*}_{RQLIF} All evaluated sampled points $\mathbf{S_{RQLIF}}$ and their responses (obj. function and constraints) $\mathbf{R_{RQLIF}}$

RQLIF is designed to be as parsimonious as possible. Indeed, the quadratic step, which is the most effective one in accelerating convergence towards the solution, requires no further function evaluation to build the model. Moreover, in such step the regularization is exploited not only to cope with noise, but also to allow the construction of the quadratic model when the number of sampled points is much lower than the number of free parameters. Even if not specified in the pseudocode for simplicity, only sampled points located in a neighborhood of the current solution are considered when building the quadratic model. The radius of such neighborhood is adaptively updated based on the value of the current step length α^{k}. Notice that also the linear step requires no further function evaluations to construct the model, and it is performed only in case of quadratic step failure.

3.5 Pseudo code of SCR

The Pseudo Code of *SCR* is given below:

	The *SCR* Algorithm
1:	**Input**: Black box function f, black-box and algebraic constraints $\boldsymbol{h}$ and $\boldsymbol{g}$, maximum evaluations E, number of variables n, tolerance T, N_d initial number of sites to be sampled, Maximum RQLIF evaluations N_1, Additional points to be sampled N_2
2:	Initialize: $eval = 0, k = 0$
3:	**Call**: *LHS* with $N_d \geq (n + 1) * (n + 2) / 2$ and number of variables $= n$
4:	**Return:** *LHS* matrix **S**
5:	**For** each point $\mathbf{x_i}$ of **S**
6:	**Call:** black box function f
7:	**Store:** Values of objective function and constraints

The *SCR* Algorithm

Increment: *eval* by 1

End For

Initialize objective function surrogate model Θ and Constraints models $\boldsymbol{\beta}$

While $eval < E$

Increment: k by 1

Call: Kriging Surrogate Model Algorithm with design sites $= \boldsymbol{S}$ and response $= \mathbf{R}$

Store: objective function surrogate model Θ

Call: Kriging Surrogate Model Algorithm with design sites $= \boldsymbol{S}$ and response $= \mathbf{C}$

Store: Constraints models $\boldsymbol{\beta}$

Call: *CMA-ES* with black box function being the penalized surrogate model

Return: Optimized solution $\boldsymbol{x}^{*,k}_{CMAES}$

Compute: black box function f and the constraints $\boldsymbol{h}(\boldsymbol{x})$ and $\mathbf{g}(\boldsymbol{x})$ at $\boldsymbol{x}^{*}_{CMAES}$

Increment: *eval* by 1

Add: $\boldsymbol{x}^{*}_{CMAES}$ in the set of surrogate design points $\mathbf{S}$

Call: RQLIF with starting point $\boldsymbol{x}^{*,k}_{CMAES}$, maximum evaluations $= N_1$, black box function

Return: Optimized solution $\boldsymbol{x}^{*,k}_{RQLIF}$ and $F^{*,k}_{RQLIF}$, all sites that *RQLIF* sampled $\boldsymbol{X}^{k}_{RQLIF}$, number of sampled sites *R_eval* and the values of the black-box function and its constraints at sampled sites

Add: $\boldsymbol{X}^{k}_{RQLIF}$ to the set $\mathbf{S}$

Increment: *eval* by *R_eval*

If $\|\boldsymbol{x}^{*,k}_{RQLIF} - \boldsymbol{x}^{*,k-1}_{RQLIF}\| \geq T$

Call: *LHS* with Sampling Size $= N_2$ and number of variables $= n$

Return: *LHS* selected points $\mathbf{S_{new}}$

For each row $\mathbf{x}_i$ of $\mathbf{S_{new}}$

Add: $\mathbf{x}_i$ to the set of design points $\mathbf{S}$

e **Call:** black box function f

Store: Values of Objective function and constraints

Increment: *eval* by 1

	The *SCR* Algorithm
35:	**End For**
36:	**End If**
37:	**End While**
38:	**Output**: Best found solution $\boldsymbol{x}^*_{RQLIF}$ and F^*_{RQLIF} and all points sampled by *SCR*

4 SCR benchmarking

In this section the performances of *SCR* are compared with other well-known derivative free algorithms. A similar short version of this benchmarking procedure for *SCR* has been reported in Zaryab et al. (2022) but in this paper not only we will go through the benchmarking procedure in greater depth, but we have also added an extra algorithm to compare with *SCR*. For the benchmarking, we selected three global-search algorithms belonging to different classes: *CMA-ES* (Hansen and Kern 2004) (the MATLAB implementation (Heris 2021)) which is one of the most effective evolutionary algorithms on nonconvex problems (including non-smooth functions) (Rios and Sahinidis 2013), *RBFOPT* (Costa and Nannicini 2018) (*RBFOPT* V4.2.0 software (Nannicini 2021)) which is a well-known global surrogate based algorithm, and *NOMAD* (Digabel 2011) (*NOMAD* V3.9.1 software (Digabel et al. 2021)), which implements a very efficient hybrid algorithm (*NOMAD* uses, in addition to orthoMADS, a quadratic surrogate model both as search strategy and as ordering criterion for the search directions). For all algorithms we used the default options, except for *NOMAD* for which we activated the VNS option (Variable Neighborhood Search) to improve its global search strategy. As for SCR, after preliminary tests, the parameters adopted the SCR algorithm are:

- $N_{\mathrm{d}} = 6n$
- $N_1 = N_2 = 3n$
- $T = 10^{-3}$ (distance measured in the normalized variable space)

Firstly, *SCR* was compared with the other algorithms on 25 nonlinear unconstrained test problems taken from ref. (Jamil and Yang 2013). A list of these test problems along with their domain is given in Table 1. Then these algorithms were tested on 21 constrained test problems which have 2–13 real variables and up to 17 constraints taken from (GLOBALLib 2020). A list of these test problems along with their variables and constraints are shown in Table 2.

The results of these tests are shown in Figs. 2 and 3 where the performance parameter shows the fraction of problems solved within a certain tolerance "τ"(as defined by Dolan and J. Moré, (2002)) and is calculated as follows:

$$Performance Parameter = \frac{1}{|N|} size\left\{n \in N : f\left(x^k\right) \leq f^* + \tau\left(f^o - f^*\right)\right\}$$

Table 1 List of bound-constrained test functions along with their variables and domain

Name	Variables	Domain	Name	Variables	Domain
Booth	2	[−10,10]	Price 3	2	[−5,5]
Camel Six Hump	2	[−3,2]	Price 4	2	[−5,5]
Camel Three Hump	2	[−5,5]	Quadratic	2	[−10,10]
Chung Reynolds	4	[−10,10]	Rosenbrock	4	[−5,10]
Colville	4	[−10,10]	Rotated Ellipse	2	[−5,5]
Cube	2	[−10,10]	Rotated Ellipse 2	2	[−50,50]
Deckkers-Aarts	2	[−20,20]	Schumer	4	[−10,10]
El-Attar-Vidyasagar-Dutta	2	[−5,5]	Schwefel	4	[−10,10]
Goldstien Price	2	[−2,2]	Schwefel 236	2	[0,100]
Himmelblau	2	[−5,5]	Sphere function	4	[−5.12,5.12]
Hosaki	2	[0,10]	Styblinski-Tang	4	[−5,5]
Leon	2	[−1.2,1.2]	Trid	4	[−16,16]
Powell Singular 2	4	[−4,5]			

Table 2 List of constrained test functions along with their variables and constraints

Name	Variable	Constraints	Name	Variable	Constraints
Chance	4	3	Ex9_2_8	6	5
Chem	11	4	Ex14_1_1	3	4
Circle	3	10	Ex14_1_2	6	9
Ex2_1_1	5	1	Ex14_1_6	9	15
Ex2_1_3	13	9	Ex14_1_7	10	17
Ex3_1_1	8	6	Ex14_2_3	6	9
Ex5_4_2	8	6	House	8	8
Ex6_1_1	8	6	Nemhaus	5	5
Ex6_2_7	9	3	Sample	4	2
Ex7_2_3	8	6	Wall	6	6
Ex8_1_8	6	5			

where N is the set of all test functions n, f^o is the value of the objective function found at the starting point, f^* is the optimum value of the function (or the best value obtained by any algorithm) and $f(\boldsymbol{x}^k)$ is the optimum value of the objective function found by the algorithm at iteration k. Each algorithm was used to optimize each test function ten times and the average results of the ten runs were then used to create the performance profiles. For each run the same starting point was given to the algorithms using Latin Hypercube sampling.

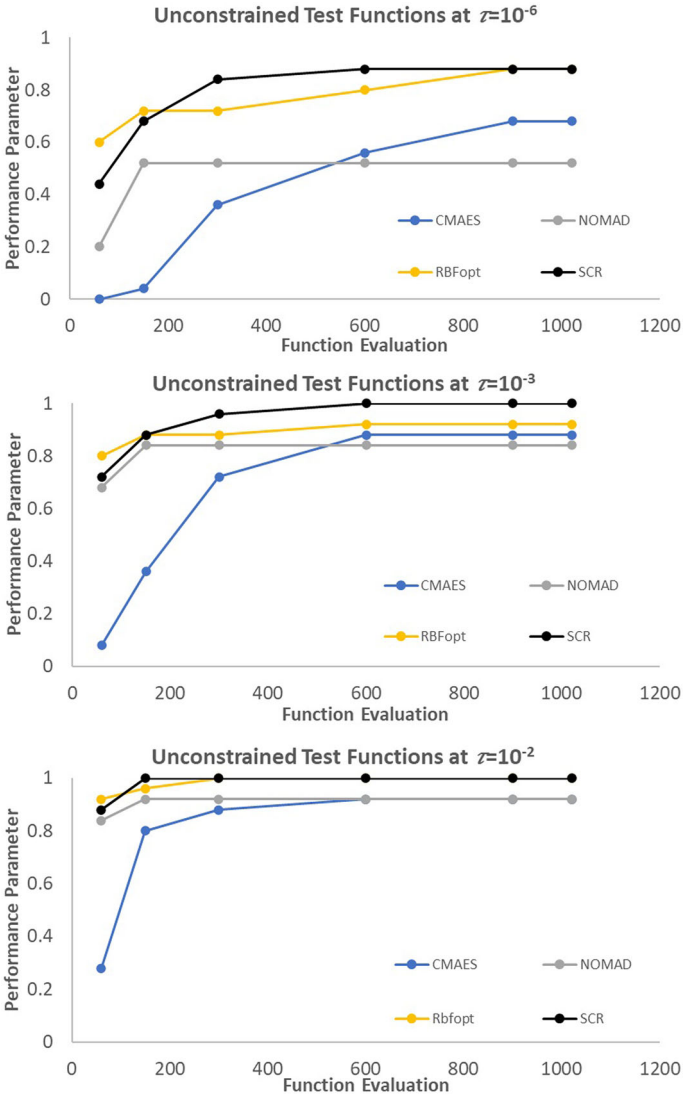

Fig. 2 Performance profiles of *SCR* and benchmark algorithms for unconstrained test problems

The performance profiles of *SCR* along with other benchmark algorithms for unconstrained test problems are shown in Fig. 2. In the figure the results are reported for three different accuracy levels which are expressed by the values of "τ", namely, 10^{-2}, 10^{-3}, 10^{-6}. The figure shows that *SCR* is the best performing algorithm compared to the rest. *SCR* also performs quite well at low function evaluations (< 300) in terms of fraction of problems solved and solution quality. Amongst the rest of the algorithms which were tested, *RBFOPT* has the second-best performance profile and in the case where the tolerance "τ" is large (10^{-2}) its performance profile matches that of *SCR*.

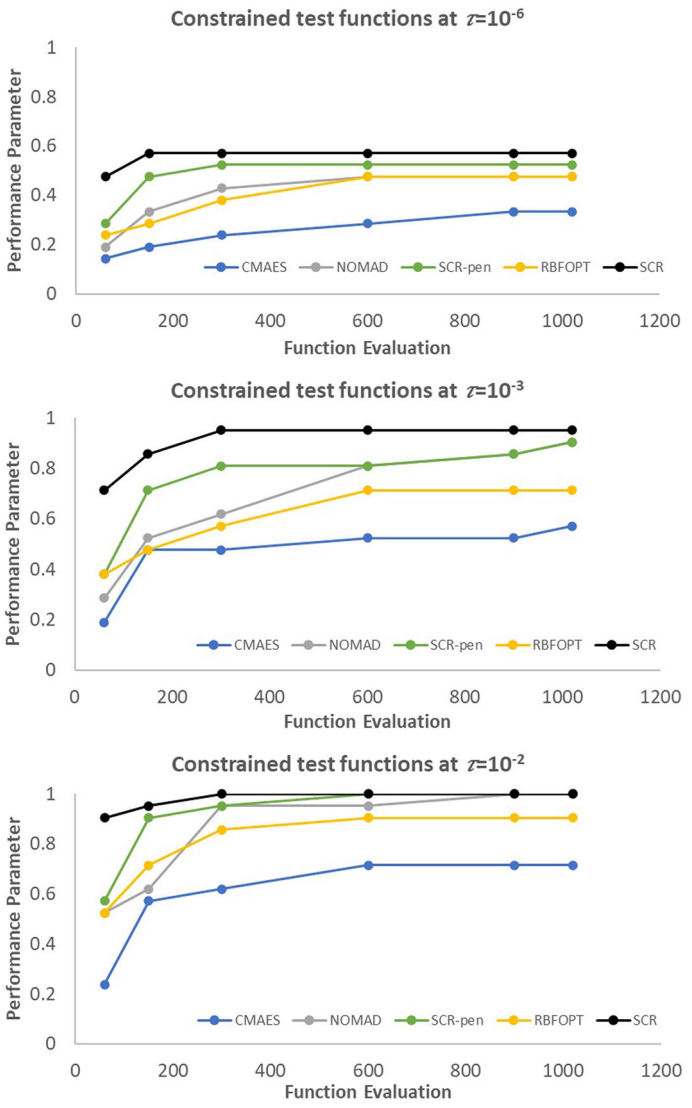

Fig. 3 Performance profiles of *SCR* and benchmark algorithms for constrained test problems

For the case where "τ" is low (10^{-6}) *RBFOPT* shows good results up to 75 function evaluations but after that it fails to reach a high-performance parameter as compared to *SCR*.

The performance profiles of the benchmark algorithms for constrained test problems are shown in Fig. 3. For the three different tolerance cases *SCR* outperforms all other benchmarking algorithms. In Fig. 3 the performance profile of a modified version of *SCR* called *SCR-pen* is also shown. Instead of generating separate surrogate models for the black box function ($f(\boldsymbol{x})$) and the constraints ($\boldsymbol{g}$ and $\boldsymbol{h}$), *SCR-pen* generates a

single surrogate model of the penalized black box function. The black box function is penalized by adding a penalty term to the objective function, this penalty term is proportional to the quadratic violation of constraints. In the benchmarking activity carried out for the constrained test problems *SCR-pen* is the second-best performing algorithm before *SCR*. In our opinion, the reason why *SCR* performs better than *SCR-pen* is because creating surrogates for a black box function with quadratic penalty terms added to it can result in ill conditioning problems generating steep valleys and high curvatures.

Apart from *SCR* and its modified version *SCR-pen* there are two algorithms which give good results at high tolerance these are *NOMAD* and *RBFOPT*. *NOMAD* performs a bit better than *RBFOPT* as it is able to provide a higher performance profile parameter at lower functional evaluations. Both *NOMAD* and *RBFOPT* have been selected for testing and comparison with *SCR* in the optimization of the two considered practical case-studies reported in the next sections, i.e. CPU and VPSA. Tables 6 and 7 in the Appendix report the list of test functions which were not optimized with the high level of accuracy, $\tau = 10^{-6}$.

To assess the influence of the number of sampling points used to build and update the surrogate (N_d, N_1 and N_2). A comparative analysis was also conducted. First, N_d was varied considering the minimum value required by the *DACE* toolbox, $N_d = (n + 1)*(n + 2)/2$ and $N_d = 6n$ (which leads to a larger number of points for $n \leq 8$). The computational results on the above described test functions showed only a minor change of performance. As for N_1 and N_2, three values were considered:

- **SCR 2n:** $N_1 = N_2 = 2n$
- **SCR 3n:** $N_1 = N_2 = 3n$ (Base case)
- **SCR 4n:** $N_1 = N_2 = 4n$

The results, presented in Fig. 4, demonstrate that the three cases have similar performance and SCR $3n$ performs (on average) better than SCR $2n$ and SCR $4n$.

5 Optimization of a CO_2 purification unit

In this section we apply *SCR* to a very challenging optimization problem, concerning a techno-economic optimization of a CO_2 Purification Unit (CPU) which is used to purify the CO_2 captured from a cement plant (Voldsund et al. 2019). The results obtained by *SCR* were compared with those obtained *NOMAD* and *RBFOPT*. The CO_2 Purification Unit used for this optimization is based on a patent by Air Products (White and Allam 2008) and its scheme is provided in Fig. 5. The CPU was modeled in Aspen Plus. It is assumed that the CPU takes a medium purity captured CO_2 coming from a VPSA cycle which is capturing 95% of the CO_2 coming from the flue gas of a cement plant (as in Voldsund et al. (2019)) and provides a high purity CO_2 gas.

This simulation code takes about 10–20 s to converge based on the input conditions. However, for some input conditions, the simulation has convergence issues: it runs for several minutes and then it crashes without returning a solution. The issue is caused by the failure of convergence of the flash calculation for some streams.

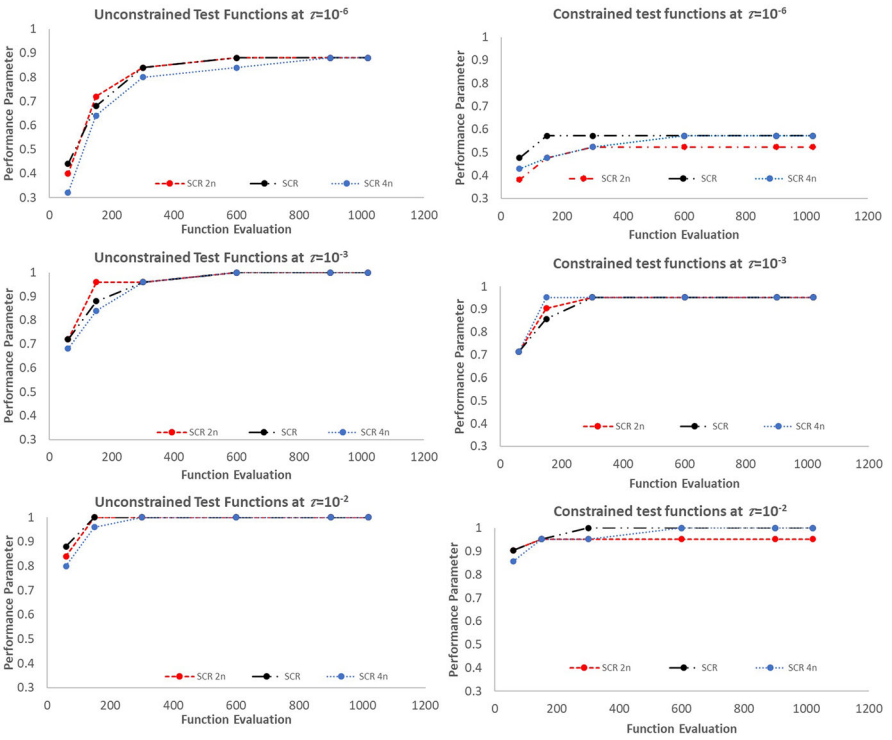

Fig. 4 Performance Comparison of *SCR* with different values of N_1 and N_2

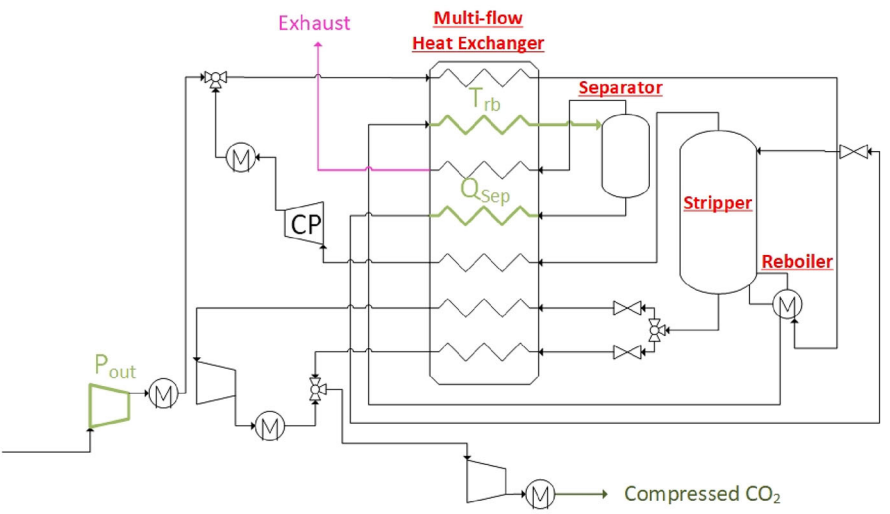

Fig. 5 Scheme of CO_2 purification unit (CPU)

The goal of the optimization is to minimize the total annual cost. The process variables that are optimized for the CPU are (1) the compressor outlet pressure (P_{out}), (2) the heat absorbed by the liquid stream coming from the separator and going to the stripper (Q_{sep}), and (3) the exit temperature of the stream entering the heat exchanger from the reboiler and going to separator (T_{rb}). There are 4 inequality constraints for the optimization: (i) the recovery of CO_2 should be greater than 95%, (ii) the purity of CO_2 should be greater than 98%, (iii) the oxygen in the outlet stream should be less than 75 ppm, and (iv) the mole fraction of nitrogen and argon in the outlet stream should be less than 3%.

Figure 6 shows the value of the objective function found by each optimizer as a function of number of black box function evaluations. It must be noted that unlike (Zaryab et al. 2022) in this paper a total of ten optimization runs were carried out for each optimizer and the average result from these runs are shown in the Fig. 6. Here *SCR* was able to find a good solution in only a limited number of black box function evaluations (50), which was further improved until the optimum was found at 200 function calls. Both *NOMAD* and *RBFOPT* were able to find the optimum of the simulation code but it took twice the number of function evaluation (400 calls) as compared to *SCR*. The comparison between the optimum found at different function evaluation of the three algorithms is provided in Table 3.

The main challenges of the CPU optimization problem are due to the highly non-linear and non-convex black-box constraints and the high prevalence of convergence failures of the flowsheet simulation. The Fig. 7 shows that 53.2% of the simulations failed to reach convergence (those denoted by an objective function value of zero in the figure). Thus, even though the required simulation time is not excessive, we reported such a test case because it demonstrates the algorithm's robustness to convergence failures and nonlinear-nonconvex black-box constraints.

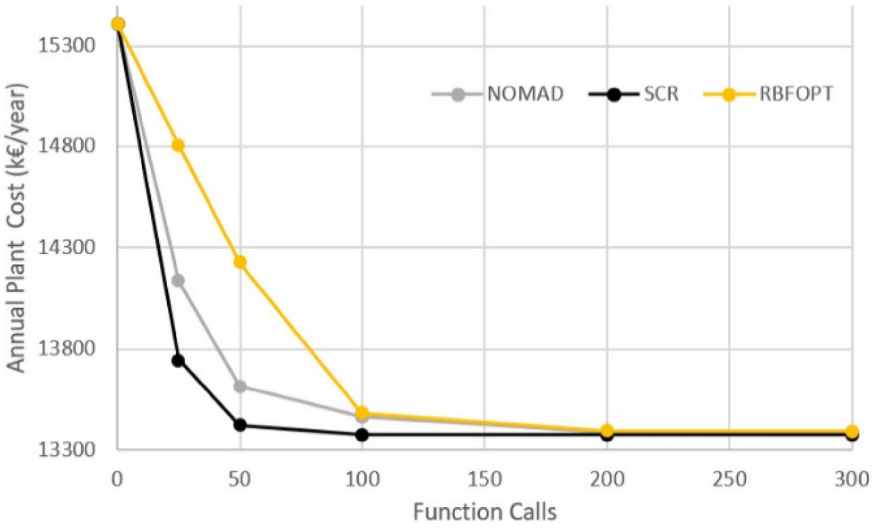

Fig. 6 Comparison of SCR with RBFOPT and NOMAD for the CPU case study

Table 3 Comparison of the results of *SCR* with that of *NOMAD* and *RBFOPT* for CPU optimization

Function calls	*SCR*	*NOMAD*	*RBFOPT*
1	15,406.67	15,406.67	15,406.67
25	13,748.74	14,137.93	14,806.76
50	13,422.39	13,617.11	14,227.18
100	13,377.41	13,463.54	13,480.76
200	13,377.41	13,387.77	13,395.69
300	13,377.40	13,385.15	13,390.99
400	13,377.38	13,377.77	13,390.59
500	13,377.38	13,377.40	13,377.51

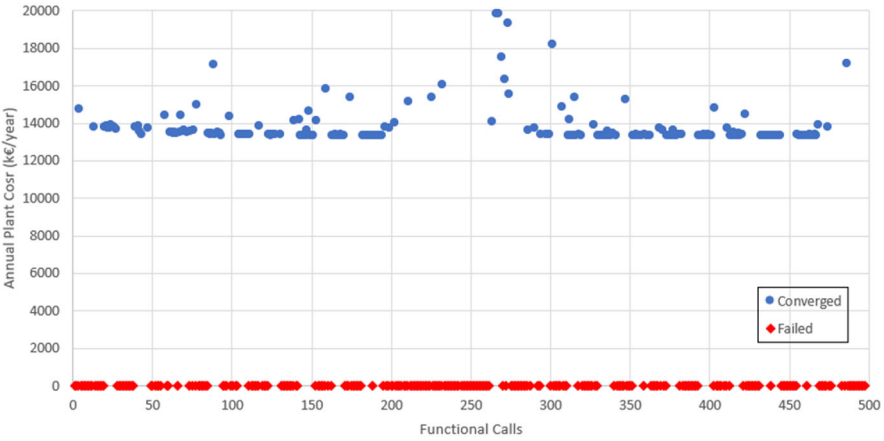

Fig. 7 Functional Evaluations versus Annual Plant Cost of *SCR* algorithm for CPU where 0 specific cost means that the simulation code failed to converge and provide results

6 Optimization of a vacuum pressure swing adsorption cycle (VPSA)

In this section *SCR* is used to optimize the specific cost (k€/ton$_{CO2}$) of a 6 step 3 bed VPSA. This VPSA will also be optimized using *NOMAD* and *RBFOPT* and the results of the three optimization algorithms would then be compared. The VPSA has been modeled in Aspen Adsorption, the simulation code of the VPSA takes from 30 min to 2 h to converge based on the input conditions. We are assuming that the flue gas coming to the VPSA has 15% CO_2 and 85% N_2. The 6 stages of the VPSA cycle are provided below:

Adsorption: The flue gas (CO_2/N_2 mixture) is fed into the column to adsorb the CO_2. A N_2 rich stream is obtained at the outlet of the column.

Rinse: A part of the CO_2 retrieved from the capture process is reused to rinse the bed at the end of the adsorption step. The rinse step aims to increase the purity of the CO_2 recovered by flushing the nitrogen out of the column.

Light blowdown: The pressure of the bed is decreased to a middle pressure, the stream recovered during this step is not collected and sent to the atmosphere. The purpose of this step is to remove the nitrogen from the column without desorbing the CO_2.

Blowdown: The pressure is decreased to a lower value to desorb the CO_2 and retrieve it.

Purge: A fraction of the N_2 rich stream collected during the adsorption step is used to flush the bed during the blowdown to push the desorbed CO_2 allowing to increase the recovery.

Pressurization: The pressure of the bed is increased to the adsorption pressure by a part of the nitrogen stream from the adsorption step.

The goal of the optimization is to minimize the specific cost of CO_2 captured (k€/ton_{CO2}). The process variables are adsorption time, purge time, Light blowdown time, Middle pressure, Feed flowrate, Purge flowrate and the Adsorption pressure. There are 2 inequality constraints for the optimization: (i) the recovery of CO_2 should be greater than 95% and, (ii) the purity of CO_2 should be greater than 85%.

Since the simulation model of the VPSA is very noisy and within the upper bound and the lower bound of the optimization variables there are many regions where no solution exists. In such a case where there is no solution (non-feasible point) the VPSA model gives a very high value of the objective function f (10^9). If a non-feasible starting point is given to an optimization algorithm, it can happen that for many iterations the optimization algorithm is not able to find any good solution.

Figure 8 shows the value of the objective function obtained by each optimizer as a function of number of black box function evaluations for the VPSA when a

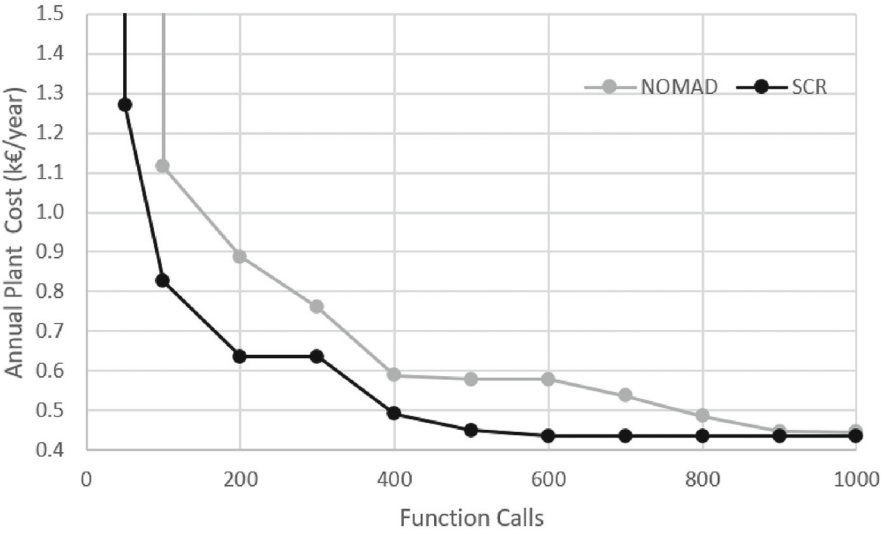

Fig. 8 Comparison of *SCR* and *NOMAD* for the VPSA optimization case where a non-feasible starting point is given to the optimizer

Table 4 Comparison of the results of *SCR* with *NOMAD* for VPSA optimization where a non-feasible starting point was given to the algorithms

Function Calls	*SCR*	*NOMAD*
10	1.00E + 09	1.00E + 09
100	1.00E + 09	1.00E + 09
200	1.270	1.00E + 09
300	0.826	1.115
400	0.636	0.887
500	0.636	0.760
600	0.492	0.589
700	0.449	0.578
800	0.435	0.578
900	0.435	0.537
1000	0.435	0.485

non-feasible starting point is chosen. The value of the optimum found at each iteration by the algorithms is shown in Table 4.

For the case where non-feasible starting point was given to the algorithms *SCR* was able to optimize the system within a limited number of function calls (around 600). While *NOMAD* took 900 iterations to reach the close to optimum value. *RBFOPT* was also used to optimize the VPSA model, but it was unable to find any solution at all. The optimizer kept trying to find solutions in the region of the optimization variables where no solution existed, therefore in figure *RBFOPT* results are not shown.

Another optimization run was carried out for all three optimizers where a feasible solution was given as a starting point. The optimum obtained by each optimizer as a function of number of black box function evaluations for this case is shown in Fig. 9. It can be seen that in 300 iteration both *SCR* and *NOMAD* were able to reach the optimum while *RBFOPT* was unable to reach it. Furthermore, at each iteration *SCR* provides a lower value of optimum compared to *NOMAD*. The results of this comparison are shown in Table 5.

7 Conclusions

SCR is a surrogate based derivative free algorithm which has been developed specifically for black-box optimization problems with expensive function evaluations and general constraints (including black-box constraints). *SCR* creates separate surrogate models of the black-box objective function and black-box constraints. The search procedure combines the use and update of Kriging as surrogate model, *CMA-ES* as global search algorithm and *RQLIF* for local search.

Performance profiles of *SCR* compared with three well-known optimization algorithms (*CMA-ES*, *RBFOPT* & *NOMAD*) indicate that *SCR* provides good quality solutions in limited number of black box function evaluations on a large set of benchmarking test problems. Furthermore, when applied to real engineering black-box

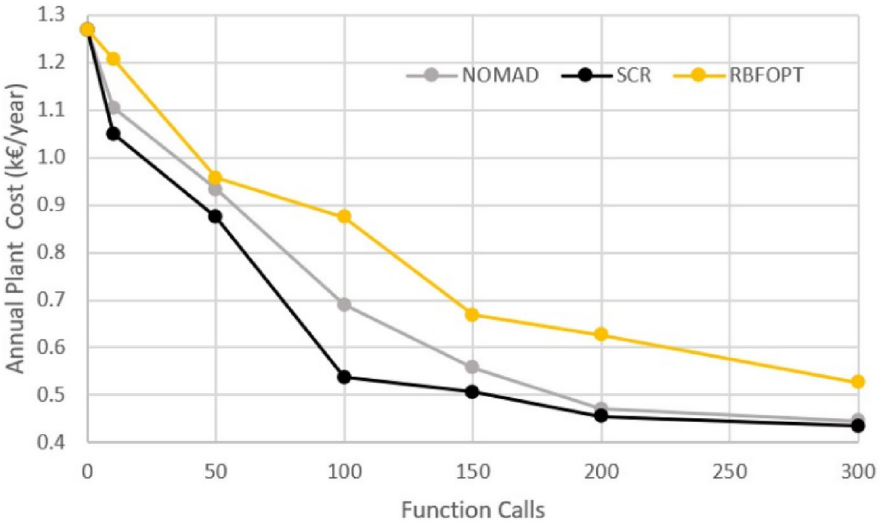

Fig. 9 Comparison of *SCR* with the results of *RBFOPT* and *NOMAD* for the VPSA optimization case where a feasible starting point is given to the optimizers

Table 5 Comparison of the results of *SCR* with *NOMAD* and *RBFOPT* for VPSA optimization where a feasible starting point was given to the optimizer

Function Calls	*SCR*	*NOMAD*	*RBFOPT*
0	1.270	1.270	1.270
10	1.050	1.105	1.208
50	0.874	0.933	0.957
100	0.537	0.690	0.874
150	0.507	0.557	0.668
200	0.455	0.470	0.627
300	0.435	0.445	0.527

optimization problem, the optimization of a CO_2 purification process and the optimization of a VPSA process, *SCR* needs about half simulation runs compared to *NOMAD* & *RBFOPT* to find the same optimal solution. Tests indicate also that generating separate surrogates for the black-box constraints improves the convergence of the algorithm compared to generating surrogates directly for the penalized objective function.

In conclusion, *SCR* looks as a promising algorithm to tackle the global optimization of computationally expensive black-box problems subject to black-box constraints.

Appendix

See Tables 6 and 7.

Table 6 List of unconstrained test problems which are not optimized within the high level of accuracy $\tau = 10–6$

SCR	*NOMAD*	*CMA-ES*	*RBFOPT*
Camel	Camel 6	Camel 6	Camel 6
Goldstien	Deck	Deck	Deck
Hosaki	Himmel	Himmel	Styblinski
	Leon	Leon	
	Schwefel 236	Schwefel 236	
	Colville	Colville	
	Rosenback	Rosenback	
	Styblinski	Styblinski	
	Attar		
	Goldstien		
	Hosaki		
	Price 3		

Table 7 List of constrained test problems which are not optimized within the high level of accuracy $\tau = 10–6$

SCR	*NOMAD*	*CMA-ES*	*RBFOPT*
Chem	Chem	Chance	Chance
Wall	Ex6-1-1	Chem	Chem
Ex2-1-3	Wall	Ex6-1-1	Ex6-1-1
Ex6-2-7	Ex2-1-3	Wall	Wall
Ex8-1-8	Ex6-2-7	Ex2-1-3	Ex2-1-3
Ex9-2-8	Ex8-1-8	Ex6-2-7	Ex8-1-8
Ex14-1-6	Ex14-1-6	Ex7-2-3	Ex9-2-8
Ex14-2-3	Ex14-2-3	Ex8-1-8	Ex14-1-6
		Ex9-2-8	Ex14-2-3
		Ex14-1-6	
		Ex14-1-7	
		Ex14-2-3	
		nemhaus	

Acknowledgements The authors gratefully acknowledge Professor Guy De Weireld and Arnauld Henrotin from University of Mons for helping in setting up the VPSA simulation model used as a test case in this paper.

Author's contribution S.A.Z: Methodology, Data curation, Software, Writing- Original draft preparation, Visualization. A.M.: Methodology, Software, Writing- Reviewing and Editing, Supervision E.M.: Conceptualization, Methodology, Software, Supervision, Writing- Reviewing and Editing, Resources.

Funding This research has received funding from the European Union's Horizon 2020 research and innovation program under the MOF4AIR project, grant agreement No. 831975. This output reflects only the author's view and the European Union cannot be held responsible for any use that may be made of the information contained therein.

Data availability The benchmark optimization algorithms can be downloaded from the websites of the original developers (NOMAD [12] downloadable from https://www.gerad.ca/en/software/nomad/, CMA-ES [10] downloadable from https://yarpiz.com/235/ypea108-cma-esand RBFOPT [13] downloadable from ttps://github.com/coin-or/rbfopt) and the literature constrained and unconstrained test problems described in Sect. 4 can be downloaded from the repository https://github.com/alizaryab/Test-functions-for-SCR. The Matlab implementation of the SCR algorithm can be made available on request while the simulation codes for the two real-world case studies described in Sects. 5 and 6 cannot be shared because based on commercial software (Aspen Plus and Apen Adsorption) and contain confidential data.

Declarations

Conflict of interest The authors declare no competing interests.

Ethical approval and consent to participate The work does not concern human/animal studies. Thus, this declaration is not applicable.

Consent for publication The manuscript does not contain any individual person's data in any form (including any individual details, images or videos). Thus, this declaration is not applicable.

References

Abramson M, Audet C (2006) Convergence of mesh adaptive direct search to second-order stationary points. SIAM J Optim 17:606–609

Abramson M, Asaki T, Dennis JJ, O'Reilly K, Pingel R (2008) Quantitative object reconstruction via Abel-based X-ray tomography and mixed variable optimization. SIAM J Imaging Sci 1:322–342

Abramson M, Audet C, Dennis JJ, Le Digabel S (2009) OrthoMADS: a deterministic MADS instance with orthogonal directions. SIAM J Optim 20:948–966

Abramson M, Audet C, Couture G, Dennis JJ LeDigabel S (2024) The Nomad project, [Online]. Available: http://www.gerad.ca/nomad/.

Audet C, Dennis JJ (2006) Mesh adaptive direct search algorithms for constrained optimization. SIAM J Optim 17:188–217

Audet C, Hare W (2017) Derivative-free and blackbox optimization. Springer, New York

Audet C, Kokkolaras M (2016) Blackbox and derivative-free optimization:theory, algorithms and applications. Optim Eng 177:1–2
Audet C, Béchard V, Chaouki J (2008a) Spent potliner treatment process optimization using a MADS algorithm. Optim Eng 9:143–160
Audet C, Béchard V, Le Digabel S (2008b) Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. J Glob Optim 41:299–318
Bartholomew-Biggs M, Parkhurst S, Wilson S (2002) Using DIRECT to solve an aircraft routing problem. Comput Optim Appl 21:311–323
Barton R, Metamodeling: a state of the art review," in *Proceedings of the 1994 Winter Simulation Conference*, 1994.
Biegler LT, Grossmann IE, Westerberg AW (1997) Systematic methods of chemical process design, 1st ed., Pearson College Div.
Boston J, Britt H (1978) A radically different formulation and solution of the single-stage flash problem. Comput Chem Eng 2(2–3):109–122
Boukouvala F, Floudas CA (2016) ARGONAUT: algorithms for global optimization of constrained grey-box computational problems. Optim Lett 11:895–913
Boneh A, Golan A (1979) Constraints' redundancy and feasible region boundedness by random feasible point generator (RFPG). In: *3rd European Congress on Operations Research (EURO III)*, Amsterdam.
Caballero A, Grossmann IE (2008) An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE J 54(10):2633–2650
Conn A, Scheinberg K, Toint P (1997) Recent progress in unconstrained nonlinear optimization without derivatives. Math Program 79:397–414
Conn A, Scheinberg K, Toint P (1998) A derivative free optimization algorithm in practice. In: *Proceedings of AIAA St Louis Conference*, St Louis.
Conn A, Scheinberg K, Vicente L (2009) Introduction to derivative-free optimization. SIAM, Philadelphia
Costa A, Nannicini G (2018) RBFOpt: an open-source library for black-box optimization with costly function evaluations. Math Program Comput 10:597–629
Currie J, Download OPTI toolbox, 2020. [Online]. Available: https://www.controlengineering.co.nz/Wikis/OPTI/pmwiki.php/DL/DownloadOPTI. [Accessed 15 3 2021].
Deming S, Parker LJ, Denton M (1974) A review of simplex optimization in analytical chemistry. Crit Rev Anal Chem 7:187–202
Digabel SL, Tribes C, Montplaisir VR, Audet C, NOMAD - a blackbox optimization software, GERAD, [Online]. Available: https://www.gerad.ca/en/software/nomad/. [Accessed 04 05 2021].
Dolan E, Moré J (2002) Benchmarking optimization software with performance profiles. Math Program 91(2):201–213
Eldred Mab, Gay D, Swiler L, Haskell K, Bohnhoff W, Eddy J, Hart W, Watson J-P, Hough P, Kolda T, Williams P, Martinez-Canales M, Dakota A (2008) Multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 4.2 User's Manual," Sandia National Laboratories, Albuquerque.
Fowler K, Reese J, Kees C, Dennis JJ, Kelley C, Miller C, Audet C, Booker A, Couture G, Darwin R, Farthing M, Finkel D, Gablonsky J, Gray G, Kolda T (2008) A comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. Adv Water Resour 31:743–757
GLOBALLib and Floudas' Collection, [Online]. Available: https://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Library1_new_v1.html. [Accessed 14 August 2020].
Gray G, Kolda T, Sale K, Young M (2004) Optimizing an empirical scoring function for transmembrane protein structure determination. Inform J Comput 16:406–418
Gutmann H (2001) A radial basis function method for global optimization. J G Opti 19:201–227
Hooke R, Jeeves TA (1961) Direct search solution of numerical and statistical problems. J ACM 2(8):212–229
Hansen N (2007) The CMA evolution strategy: a tutorial.
Hansen N, Kern S (2004) Evaluating the CMA evolution strategy on multimodal test functions. Eighth Int Conf Parallel Problem Solv Nat PPSN 8:282–291
Hansen N. The CMA evolution strategy: a tutorial. [Online]. Available: http://www.lri.fr/hansen/cmaesintro.html.
Heris MK, CMA-ES in MATLAB. yarzip, [Online]. Available: https://yarpiz.com/235/ypea108-cma-es. [Accessed 12 02 2021].

Holland J (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
Holmström K, Quttineh N-H, Edvall M (2008) An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Optim Eng 9:311–339
Hooke R, Jeeves T (1961) "Direct search" solution of numerical and statistical problems. J ACM 2(8):21–229
Huyer W, Neumaier A (1999) Global optimization by multilevel coordinate search. J Glob Optim 14:331–355
Jamil M, Yang X-S (2013) A literature survey of benchmark functions for global optimization problems. Int J Math Model Numeric Optim 4(2):150–194
Jones D (2001) A taxonomy of global optimization methods based on response surfaces. J Glob Optim 21:345–383
Jones D, Perttunen C, Stuckman B (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 79:157–181
Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Global Optim 13:455–492
Kelley C (1999) Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. SIAM J Optim 3(45):43–55
Kolda T, Lewis R, Torczon V (2003) Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev 3(45):385–482
Le Digabel S (2011) Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. ACM Trans Math Softw (TOMS) 37(4):1–15
Le Besnerais J, Fasquelle A, Lanfranchi V, Hecquet M, Brochet P (2011) Mixed-variable optimal design of induction motors including efficiency, noise and thermal criteria. Optim Eng 1–2(12):55–72
Liepins G, Hilliard M (1989) Genetic algorithms: foundations and applications. Ann Oper Res 21:31–58
Lophaven SN, Nielsen HB, Sondergaard J (2002) Dace: a MATLAB Kriging toolbox. Technical University of Denmark: Technical Report No. IMM-TR-2002–12., Kongens Lyngby.
Lucidi S, Sciandrone M (2002) A derivative-free algorithm for bound constrained optimization. Comput Optim Appl 21:119–142
Madsen J, Langthjem M (2001) Multifidelity response surface approximations for the optimum design of diffuser flows. Optim Eng 4(2):453–468
Manno A, Amaldi E, Casella F, Martelli E (2020) A local search method for costly black-box problems and its application to CSP plant start-up optimization refinement. Optim Eng 21(4):1563–1598
Marsden A, Feinstein J, Taylor C (2008) A computational framework for derivative-free optimization of cardiovascular geometries. Comput Method Appl Mech Eng 197:1890–1905
Martelli E, Amaldi E (2014) PGS-COM: A hybrid method for constrained non-smooth black-box optimization problems: brief review, novel algorithm and comparitive evaluation. Comput Chem Eng 63:108–139
Matheron G (1967) Principles of geostatistics. Econ Geol 58:1246–1266
Meo M, Zumpano G (2008) Damage assessment on plate-like structures using a global–local optimization approach. Optim Eng 2(9):161–177
Nannicini G, "rbfopt," [Online]. Available: https://github.com/coin-or/rbfopt. [Accessed 10 02 2021].
Nelder J, Mead R (1965) A simplex method for function minimization. Comput J 7:308–313
Neumaier A, MCS: global optimization by multilevel coordinate search. [Online]. Available: http://www.mat.univie.ac.at/neum/software/mcs/.
Oeuvray R (2005) Trust-region methods based on radial basis functions with application to biomedical imaging. *PhD thesis, Institute of Mathematics, Swiss Federal Institute of Technology, Lausanne (March)*.
Olivero DPGRM (2014) Three-dimensional turbulent optimization of vaned diffusers for centrifugal compressors based on metamodel-assisted genetic algorithms. Optim Eng 4(15):973–992
Peeters J, Louarroudi E, Bogaerts B, Sels S, Dirckx J, Steenackers G (2018) Active thermography setup updating for nde: a comparative study of regression techniques and optimisation routines with high contrast parameter influences for thermal problems. Optim Eng 1(19):163–185
Powell MJD (1987) Radial basis functions approximations to polynomials. *Proceedings 12th Biennial Numerical Analysis Conference*
Powell M (2006) The NEWUOA software for unconstrained optimization without derivatives. In *Nonconvex Optimization and Its Applications*, Boston, MA, Springer.
Regis R, Shoemaker C (2005) Constrained global optimization of expensive black box functions using radial basis functions. J Glob Optim 31:153–171

Regis R, Shoemaker C (2007) Improved strategies for radial basis function methods for global optimization. J Glob Optim 37:113–135

Rios L, Sahinidis N (2013) Derivative-free optimization: a review of algorithms and comparison of software implementations. J Global Optim 56(3):1247–1293

Scheinberg K (2003) Manual for fortran software package DFO v2.0.

Smith R (1984) Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. Oper Res 32:1296–1308

Spendley W, Hext G, Himsworth F (1962) Sequential application for simplex designs in optimisation and evolutionary operation. Technometrics 4:441–461

Tseng P (1999) Fortified-descent simplicial search method: a general approach. SIAM J Optim 1(10):269–288

Vaz A, Vicente L (2007) A particle swarm pattern search method for bound constrained global optimization. J Glob Optim 39:197–219

Vaz A, PSwarm Home Page. [Online]. Available: http://www.norg.uminho.pt/aivaz/pswarm/.

Voldsund M, Anantharaman R, Berstad D, Lena ED, Fu C, Gardarsdottir OS, Jamali A, Pérez-Calvo J-F, Romano M, Roussanaly S, Ruppert J, Stallmann O, Sutter D (2019) D4.6 CEMCAP comparative techno-economic analysis of CO2 capture in cement plants. https://doi.org/10.5281/zenodo.2597090

White V, Allam R (2008) Purification of carbon dioxide. Patent US 2008/0173584A1.

Zaryab SA, Scaccabarozzi R, Martelli E (2020) Advanced part-load control strategies for the Allam cycle. Appl Therm Eng 168:114822

Zaryab SA, Manno A, Martelli E (2022) SCR: a novel surrogate-based global optimization algorithm for constrained black-box problems. In *32nd European Symposium on Computer Aided Process Engineering*, Toulouse.