Optimization Letters
https://doi.org/10.1007/s11590-024-02109-w

ORIGINAL PAPER

Extending OSCARS-II to generally constrained global optimization

C. J. Price[1] · B. L. Robertson[1] · M. Reale[1]

Received: 27 July 2023 / Accepted: 5 March 2024

Abstract
A derivative free method for generally constrained global optimization is described. A non-smooth merit function with one parameter is used. When this parameter equals the optimal objective function value f^*, the merit function becomes an exact penalty function. The method estimates f^*, avoiding the need for it to be supplied. The method randomly samples the region satisfying the simple bounds from time to time, ensuring convergence almost surely. Other samples are drawn randomly from smaller regions considered promising. Numerical testing is done using a variety of bound constrained problems and generally constrained problems from the G-suite and elsewhere. Results show the method is competitive in practice. They also show that the method performs better when it estimates the optimal objective function value than when the actual value is used.

Keywords Derivative free · Direct search optimisation · OSCARS · Numerical results

1 Introduction

The generally constrained global optimization problem addressed here has the form

$$\min_{x\in\Omega} f(x) \quad \text{subject to} \quad g_I(x) \le 0 \quad \text{and} \quad g_E(x) = 0 \tag{1}$$

where Ω is a finite box defined by upper (U) and lower (L) bounds, as follows

✉ C. J. Price
chrisj.price@canterbury.ac.nz

B. L. Robertson
blair.robertson@canterbury.ac.nz

M. Reale
marco.reale@canterbury.ac.nz

[1] School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand

Published online: 31 March 2024

$$\Omega = \{x \in \mathbb{R}^n : L_i \leq x_i \leq U_i \quad \forall i = 1, \ldots, n\}.$$

The objective function f maps $\mathbb{R}^n$ into $\mathbb{R}$. The equality constraint function $g_E(x)$ and inequality constraint function $g_I(x)$ map $\mathbb{R}^n$ into $\mathbb{R}^q$ and $\mathbb{R}^{m-2q}$, respectively. It is assumed that $L < U$ and that f, g_E and g_I are continuous functions of x on Ω.

For simplicity, the equality constraints are replaced by the pair of inequality constraints $g_E(x) \leq 0$ and $-g_E(x) \leq 0$. This puts all constraints in the convenient form $g(x) \leq 0$ where $g(x)$ maps $\mathbb{R}^n$ into $\mathbb{R}^m$, yielding a simpler expression of problem (1), viz.

$$\min_{x \in \Omega} f(x) \quad \text{subject to} \quad g(x) \leq 0 \tag{2}$$

From now on, we work with (2) rather than (1).

A wide variety of algorithms have been proposed for this problem. The majority of these methods are stochastic, although deterministic methods also exist [9].

Many stochastic methods are population based, such as particle swarm [4, 14], differential evolution [17], fish swarm [24] artificial immune system [3], and electromagnetism-like methods [1]. Stochastic methods are often paired with local searches to refine identified minimizers. For example, Hedar and Fukushima [7] pair the derivative free Nelder–Mead method with simulated annealing. Sequential quadratic programming is paired with multistart-clustering in DEFT-FUNNEL [27], and with particle swarm methods [11].

Methods can also be differentiated on how expensive each function value is to compute. For expensive objectives methods using surrogates [15, 25] are effective, with radial basis functions being a common means of generating the surrogates. The surrogate is designed to be much cheaper to evaluate than the objective and constraint functions. The surrogate can be minimized by a subsidiary global optimization method designed for cheaper to evaluate functions giving a new sample point for (2). A more sophisticated fusion combining accelerated random search [2] with surrogates is given by Nuñez et al. [19] and Regis [23].

These methods employ a variety of strategies to adjudicate between changes in objective function and constraint violations. They include filters [7, 15, 21, 24], penalty functions [1, 9, 26], interval arithmetic [14] and other techniques such as adaptive trade-off models [30]. In addition, sample points can be biased towards feasibility by using constraint gradient information [29] or other processes [31]. A thorough survey of constraint handling techniques is given by Mezura-Montes and Coello [16].

In the next section, we describe a stochastic penalty function method for problems with objective and constraint functions that are cheap to evaluate. Convergence is discussed in Sect. 3, with numerical results given in Sect. 4. The final section concludes the paper.

2 Algorithm development

The feasible region of problem (2) is

$$\mathcal{F} = \{x \in \Omega : \ g_i(x) \leq 0 \quad \forall i = 1, \ldots, m\}.$$

The proposed algorithm extends the OSCARS-II algorithm [22] for bound constrained global optimization to problems of the form (1). OSCARS-II and the new method generate random sample points in Ω and various subregions of Ω. Both methods retain two points at each iteration: the best known point $b \in \Omega$ and a control point $c \in \Omega$ used to direct the construction of the sampled subregions of Ω. The basic structure of each iteration is to randomly choose an iterate from the current subregion of Ω, calculate f and g at that iterate, and then update b, c and the sampling subregion.

With OSCARS-II general constraints are absent, and b is just the iterate with the least known value of f. From time to time the control point is reset to b or a random point in Ω. Between resets c is the point with the least f value from or after the most recent reset.

The presence of general constraints means b and c must be chosen differently. At each iteration the new method chooses b as the best known feasible point, or if no feasible point is found, the least infeasible infeasible point. In contrast, c is chosen to minimize a merit function J over the iterates generated since the most recent reset. The merit function contains a parameter which is adjusted occasionally to obtain an acceptable convergence rate.

If $\mathcal{F}$ has measure zero, the algorithm will almost surely fail to find a feasible point with any finite number of sample points. To circumvent this issue, violations of the general constraints $g \leq 0$ up to a specified tolerance $\tau_c \geq 0$ are permitted. The subset of Ω satisfying the general constraints within this tolerance is

$$\mathcal{F}_{\text{tol}} = \left\{x \in \Omega : \ g_i(x) \leq \tau_c \quad \forall i = 1, \ldots, m\right\}.$$

Provided $\mathcal{F}$ is non-empty, continuity of g implies $\mathcal{F}_{\text{tol}}$ has positive measure for all $\tau_c > 0$. Depending on the nature of the constraints, $\tau_c > 0$ might be necessary, but if $\mathcal{F}$ has a positive measure, then $\tau_c = 0$ suffices.

Throughout this paper x^* and x^*_{tol} denote arbitrary global minimizers of f over $\mathcal{F}$ and $\mathcal{F}_{\text{tol}}$ respectively. Also $f^* = f(x^*)$ and $f^*_{\text{tol}} = f(x^*_{\text{tol}})$ are used.

2.1 The merit function

Jones [9] introduced the auxiliary function

$$J_{\text{orig}} = [f(x) - \phi]_+ + \sum_{j=1}^{m} w_j [g_j(x)]_+$$

to extend the DIRECT algorithm [10] to problems of the form (1). Here ϕ is a possible value of f^*, w_j are positive weights and $[g_j]_+ = \max(g_j, 0)$. DIRECT subdivides Ω into ever finer sets of rectangles, where each such set covers Ω. At each iteration some rectangles are divided, yielding the next cover. The rectangles which are selected are those which could contain a global minimizer of J_{orig} for some value of ϕ and some value of a Lipschitz constant for J_{orig}. In order to do this, DIRECT calculates and retains the objective and constraint function values at the centre of every rectangle. In constrast OSCARS-II and the new method retain only b and c.

The auxiliary function J_{orig} is modified cosmetically to yield the merit function J. Specifically ϕ is added to the first term, and a different measure of infeasibility is used for the second term giving

$$J(x,\phi) = \max\{f(x),\phi\} + \frac{(v(x))^2}{1+v(x)}$$

Here $v(x)$ is the unweighted 2–norm of the constraint violations

$$v(x) = \left\|[g(x)]_+\right\|_2,$$

and the parameter ϕ is an estimate of f^* that is updated from time to time. The second term in J behaves like v^2 when v is small, and like v for $v \gg 1$. This gives J some of the characteristics of a rounded ℓ_2 exact penalty function.

When $\phi = f^*$, J has the property that its global minimizer(s) over Ω are precisely the solutions of (2). To see this first note that $J(x^*,\phi) = f^*$ when $\phi = f^*$ and x^* solves (2). For any $x \in \Omega$ we have

$$J(x,f^*) = \max\{f(x),f^*\} + \frac{(v(x))^2}{1+v(x)} \geq f^* + 0 = J(x^*,f^*)$$

showing that x^* is a global minimizer of $J(\cdot,f^*)$ over Ω. Conversely $J(x,f^*) = f^*$ can only hold if $f(x) \leq f^*$ and $v(x) = 0$. However $v(x) = 0$ implies $x \in \mathcal{F}$, which means that $f(x) \geq f^*$. Hence the set of global minimizers of (2) is the same as the set of global minimizers of $J(\cdot,f^*)$ over Ω, as required.

2.2 An iteration

The algorithm uses a sequence of iterations indexed by k. At iteration k, the algorithm randomly draws one sample point x_k from a sampling region $\Omega_k \subseteq \Omega$ and calculates f and g at x_k. The algorithm also updates two points at each iteration: the best known point b_k and the current control point c_k. A subscript k refers to a quantity's value at iteration k.

Each Ω_k is box-shaped and aligned with the coordinate axes, which makes randomly sampling Ω_k straightforward. Specifically, the sampling region has the form

$$\Omega_k = \{x \in \mathbb{R}^n : \ell_k \leq x \leq u_k\}$$

where $\ell_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^n$ are the vectors of lower and upper bounds satisfying $L \leq \ell_k$ and $u_k \leq U$. The bounds ℓ_k and u_k are adjusted at each iteration so that $c_k \in \Omega_k$ always holds.

If at least one point in $\mathcal{F}_{\text{tol}}$ has been found, b_k is the sample point in $\mathcal{F}_{\text{tol}}$ with the least value of f. Otherwise, b_k is the least infeasible sample point in the sense that it has the smallest value of $v(x)$. In contrast, the control point is the 'recently generated' sample point with the least J value.

From time to time, the control point is reset, allowing the method to alternately search widely across Ω, and focus attention in the vicinity of the currently best known point.

2.3 The control point and selecting Ω_k

The control point c_k is used to direct how each Ω_{k+1} is formed from its predecessor Ω_k. After selecting the current sample point x_k, if $J(x_k, \phi) \geq J(c_k, \phi)$, then x_k is rejected and Ω_{k+1} is chosen so that $\Omega_{k+1} \subset \Omega_k$ with $c_{k+1} = c_k \in \Omega_{k+1}$ and $x_k \notin \Omega_{k+1}$. If this proposed Ω_{k+1} is too small along all coordinate directions, it is reset via $\Omega_{k+1} = \Omega$. In any case, the current control point is retained via $c_{k+1} = c_k$.

Alternatively, if $J(x_k, \phi) < J(c_k, \phi)$, then x_k is judged to be superior to c_k by the merit function, and $c_{k+1} = x_k$ and $\Omega_{k+1} = \Omega$ are used. Hence, if a better point is found, this becomes the new control and the sampling box resets to Ω.

2.4 Passes and cycles

Iterations in which the sampling box Ω_k is reset to Ω can be used to group iterations into passes. Each pass starts at an iteration with $\Omega_k = \Omega$, and ends on the iteration before $\Omega_k = \Omega$ next occurs. Similarly, iterations in which c is reset can be used to group iterations (and passes) into cycles. Each cycle starts at an iteration with in which c is reset, and ends on the iteration before the next reset of c occurs. Since c is only ever reset when $\Omega_k = \Omega$, each cycle consists of a whole number of passes. This is described in more detail now.

The process used to generate each Ω_k produces contiguous subsequences of nested boxes, bracketed by iterations where $\Omega_k = \Omega$. A sequence of such iterations forms a pass. For example, if iterations $k, k+1, \ldots, k+p$ form a pass then $\Omega_k \supset \Omega_{k+1} \supset \cdots \supset \Omega_{k+p-1} \supset \Omega_{k+p}$ where $\Omega_k = \Omega_{k+p+1} = \Omega$.

The sequence of passes is divided up into contiguous subsequences of passes called cycles. The event which distinguishes the start of a cycle (and hence the end of the previous cycle) is the standard way of choosing $c_k = c_{k-1}$ or $c_k = x_{k-1}$ is suspended for one iteration. Instead, at the start of cycle number N_c, if N_c is odd, then c_k is chosen randomly from Ω. If N_c is an even number, c_k is set equal to the current best known point b_k. In both cases $\Omega_k = \Omega$ is used.

The motivation for alternately starting cycles with random controls and the best known point is that the former aids exploration of unexplored areas of Ω whereas the latter focuses the search in the most promising area found so far. To ensure the method alternates between these two cases on a regular basis, the maximum number of sample points in each cycle is limited, with this limit increasing with increasing N_c. Cycles are also ended if they repeatedly fail a 'stall test' which assesses whether or not the current cycle is likely to improve the best known point. The stall test is only performed at the end of each pass. If a cycle is ended for any reason, the current pass is also ended.

In summary, passes and cycles end for the following four reasons

1. An improved control point has been found (pass ends only).
2. The sample box size falls below h_{min} along all axes (pass ends only).
3. T_{stall} consecutive stall test failures occur (pass and cycle end).
4. The maximum permitted number of sample points in the current cycle is reached (pass and cycle end). Herein this maximum is $30(3 + N_c)$ as per Price et al. [22].

In practice, case 4 is checked first. If case 4 does not hold, cases 1 and 2 are checked. If either case 1 or 2 holds, then case 3 is checked for the end of a cycle, otherwise case 3 is skipped.

At the start of each even numbered cycle, the algorithm may take one uphill step that increases J by at most G. This step aids the algorithm in moving along constraint boundaries, and is described in more detail later.

Next, the main algorithm is listed. It counts the number of iterates in the current cycle using j. Also c_{old} and c_{older} denote the control points at the ends of the previous two passes in the cycle. At the start of each cycle c_{old} and c_{older} are set equal to that cycle's initial control point. These two quantities are used in the stall test.

Algorithm 1 The main algorithm

1. Randomly pick the initial control $c \in \Omega$. Set $N_c = k = 1$, set $G = j = 0$, and set $\Omega_1 = \Omega$. Calculate $f(c)$ and $g(c)$. Set $b = c_{\mathsf{old}} = c_{\mathsf{older}} = c$.
2. Choose $x_k \in \Omega_k$ randomly, calculate $f(x_k)$ and $g(x_k)$. Update the best known point and increment j.
3. Set $\phi = f(b) + v(b)$. Set `NewPass = false` and set `NewCycle = false`.
4. If $j > 30(3 + N_c)$ set `NewCycle = true` and go to step 7.
5. If $J(x_k, \phi) < J(c, \phi) + G$ then do (a); otherwise do (b).
   (a) Set `NewPass = true`. Set $c_{\mathsf{older}} = c_{\mathsf{old}}$ and then set $c_{\mathsf{old}} = c$. Set $G = 0$.
   (b) Perform the box cutting sub-algorithm (Algorithm 2).
6. If `NewPass = true` set $\Omega_{k+1} = \Omega$ and perform a stall check.
7. If `NewCycle = true` then
   (a) If N_c is even, set $c = b$ and set $G = \frac{1}{100k} \sum_{i=1}^{k} \min\{v(x_i), 100\}$;
   (b) If N_c is odd select c randomly from Ω. Calculate $f(c)$ and $g(c)$ and update the best known point; and
   (c) Increment N_c. Set $j = 0$, set $\Omega_{k+1} = \Omega$ and set $c_{\mathsf{old}} = c_{\mathsf{older}} = c$.
8. Increment k. Go to step 2.

2.5 The uphill step

When $G > 0$, step 5 will accept an iterate x_k up to G worse than c. Normally $G = 0$ is used except for cycles which start with the best known point as the initial control point. These cycles can use $G > 0$ until the control point is first updated; after that $G = 0$ is used. At the start of each such cycle, G is set to is one percent of the average constraint violation (capped at 100) seen by the algorithm so far. This choice permits an uphill step of at most 1 when all sample points generated so far are highly

infeasible. As less infeasible or feasible sample points are found, this maximum uphill step seamlessly reduces to its minimum possible value of $G = 0$. The latter is achieved when only feasible sample points are encountered, irrespective of whether general constraints are present or not.

This allows a single worse step off the best known point at the start of each evenly numbered cycle. The purpose of this is to enhance the algorithm's ability to move along constraint boundaries.

Allowing uphill steps has pros and cons. If the best point is wedged in a vee, then an uphill step can escape the notch, and subsequent steps might be able to locate an improved point more rapidly. The risk is that the algorithm will waste time undoing the uphill step without any gain. Numerical experiments with test problems which have no general constraints show that the latter is the dominant effect on such problems. When general constraints are present, numerical results indicate the uphill step is beneficial.

2.6 Box cutting procedure

At each iteration, a sample point x_k is generated in Ω_k. If x_k is not better than the current control point c_k (specifically $J(x_k, \phi_k) \geq J(c_k, \phi_k)$), then part of the region Ω_k is cut off, yielding the next sample box $\Omega_{k+1} \subset \Omega_k$. This cutting process is done by shifting some of the bounds defining Ω_k inwards towards c_k. Each such bound shift is equivalent to cutting off part of Ω_k with a hyperplane orthogonal to some coordinate axis. Each of these cuts is selected so that c_k lies in the part of Ω_k that is retained, and x_k lies in the part that is cut off. Since the current control point is retained when Ω is cut, this yields $c_{k+1} = c_k \in \Omega_{k+1}$.

In the rest of this subsection, the iteration number k is dropped from the subscripts of all quantities. In some places a subscript i appears. It denotes the i^{th} component of the relevant quantity at iteration k.

The cutting process (listed in Algorithm 2) is governed by two parameters: A and β. The former governs how close each cut is to c_k and β affects which faces of Ω_k are cut off. A cut is performed perpendicular to each coordinate axis for which the magnitude of the corresponding component of the trial step $s = x - c$ is at least $\beta \|x - c\|_\infty$, where $0 < \beta < 1$. For each such dimension i, the cut passes a fraction $A|s_i|/\|s\|_\infty$ along the line segment from x to c, where $0 < A < 1$. Hence, for dimensions with maximal $|s_i|$, the cut is a fraction A of the distance from x to c.

Algorithm 2 The box cutting sub-algorithm

1. For $i = 1, \ldots, n$ do
   (a) calculate
   $$\mu = x_i + A \frac{|c_i - x_i|}{\|x - c\|_\infty} (c_i - x_i)$$
   (b) If $c_i < x_i$ and $c_i - x_i > \beta \|x - c\|_\infty$ set $u_i = \mu$.
   (c) If $c_i \geq x_i$ and $c_i - x_i > \beta \|x - c\|_\infty$ set $\ell_i = \mu$.
2. If $\|u - \ell\|_\infty < h_{\mathsf{min}}$, set `NewPass = true`, set $c_{\mathsf{older}} = c_{\mathsf{old}}$, then set $c_{\mathsf{old}} = c$.

2.7 Stall test

Oscars-ii [22] uses a Kolmogorov–Smirnov (KS) statistic to terminate unprofitable cycles on problems with no general constraints. This test uses the 100 best objective function values in the current cycle, sorted in increasing order. This prevents the KS test from easily being transported over to the generally constrained case. The obvious approach is to apply the same test with the values of J in place of the f values. However J depends on ϕ, and ϕ can be updated at any iteration. This means all points in the current cycle must be stored, and re-sorted after every change in ϕ. This makes the KS test significantly more expensive to implement. Thus we replace it with a simpler, cheaper to implement stall test which is described now.

A stall test is done at the end of each pass which is not also the end of the current cycle. This situation occurs whenever a new point with a lower J value is found or the minimum box size is reached. If the stall test indicates that progress is poor for T_{stall} consecutive passes, the cycle is considered to have stalled. Poor progress can occur in two ways: Firstly, if the pass does not improve the current control point c (no progress), or progress is made, but it is insignificant.

For the latter case, let J^* minimize $J(\cdot, \phi_k)$ over Ω. We make the simplifying assumption that the measure of the level sets for J just greater than J^* can be adequately approximated by a power law of the form $K(J - J^*)^p$ for some values of K and p. This often occurs in practice: for example an unconstrained minimizer of a C^2 function with a positive definite Hessian has this characteristic. Under this assumption, it is easily seen that the expected reduction in J from an improving step is proportional to $J(c) - J^*$, provided c is sufficiently close to x^* for any sample x satisfying $J(x) < J(c)$ to be drawn randomly from the level set $\{x \in \Omega : J(x) < J(c)\}$.

The expected value of each reduction in J is unknown. In its place, the actual changes in J are used. This allows a rough estimate J_{est} of the limit of J for the current cycle that can be formed after each improvement in J via

$$J_{\mathsf{est}} = J(c) - \lambda \frac{J(c_{\mathsf{old}}) - J(c)}{1 - \lambda} \quad \text{where} \quad \lambda = \frac{J(c_{\mathsf{old}}) - J(c)}{J(c_{\mathsf{older}}) - J(c_{\mathsf{old}})}$$

If either

(a) no improvement in J has been made in this pass; or
(b) both $J(c_{\text{old}}) - J(c) < J(c_{\text{older}}) - J(c_{\text{old}})$ and $J_{\text{est}} \geq J(b) - \tau_{\text{stall}}$

hold, then the number of consecutive stalled passes N_s is incremented, otherwise N_s is set to zero. Noting that $J(c) \leq J(c_{\text{old}}) \leq J(c_{\text{older}})$, the first condition in (b) guarantees that λ is defined and $\lambda < 1$. If $\lambda \geq 1$, progress does not appear to be decaying and the method is assumed to not be stalling. Here τ_{stall} is the smallest decrease in J which is considered significant.

Once T_{stall} consecutive passes have each yielded poor progress the cycle is ended, and a new cycle starts from either a random point in Ω, or the best known point.

3 Convergence properties

This section looks at the convergence properties of the method when run in exact arithmetic without halting. Since there is no guarantee that the feasible region has positive measure, it is necessary to frame the results in terms of the essential global minimum, which is as follows.

Definition 1 The essential global minimum $f^\sharp$ of f over a set $S \subseteq \Omega$ is

$$f^\sharp(S) = \inf\{\eta \in \mathbb{R} : m(\{x \in S : f(x) \leq \eta\}) > 0\}$$

where $m(\cdot)$ denotes the Lebesgue measure. If $m(S) = 0$, we set $f^\sharp = \infty$.

The main convergence result shows that the algorithm locates an essential global minimizer of f over $\mathcal{F}_{\text{tol}}$ almost surely.

Theorem 1 *Let $\mathcal{F}$ be non-empty and let b_∞ be an arbitrary limit of $\{b_k\}$. Then firstly*

(a) $\tau_c > 0$ *implies* $\mathcal{F}_{\text{tol}}$ *has positive measure; and secondly*
(b) $m(\mathcal{F}_{\text{tol}}) > 0$ *implies both* $b_\infty \in \mathcal{F}_{\text{tol}}$ *and* $f(b_\infty) \leq f^\sharp(\mathcal{F}_{\text{tol}})$ *almost surely.*

Proof For part (a), let $z \in \mathcal{F}$ and define the neighborhood

$$\mathcal{N}_\epsilon^* = \{x \in \Omega : \|x - z\|_\infty < \min(\delta, \epsilon)\} \quad \text{where} \quad \delta = \min_{i \in 1,\ldots,n} (U_i - L_i)/2 > 0.$$

$\mathcal{N}_\epsilon^*$ has Lebesgue measure of at least $\min(\delta^n, \epsilon^n)$, which is positive for all $\epsilon > 0$. This is easily seen on noting that at least one orthant of the uniform norm ball $\{x \in \mathbb{R}^n : \|x - z\|_\infty < \min(\delta, \epsilon)\}$ lies entirely within Ω. Continuity of g implies $\exists \epsilon > 0$ such that $\mathcal{N}_\epsilon^* \subset \mathcal{F}_{\text{tol}}$, as required.

For part (b), step 4 of the main algorithm ensures the number of cycles $N_c \to \infty$ as the number of points $k \to \infty$. At the start of each odd numbered cycle, the control point is drawn randomly from Ω, hence the number of sample points drawn randomly from Ω becomes arbitrarily large as $k \to \infty$.

The strategy for updating the best point means once a point in $\mathcal{F}_{\text{tol}}$ is located, all future best points will lie in $\mathcal{F}_{\text{tol}}$. Additionally, each such best point in $\mathcal{F}_{\text{tol}}$ can only be replaced by another point in $\mathcal{F}_{\text{tol}}$ which has a lower f value. Let

$$\epsilon_\mu = m\big(\{x \in \mathcal{F}_{\text{tol}} : f(x) \le \mu\}\big)$$

Now $\epsilon_\mu > 0$ for all $\mu > f^\sharp\big(\mathcal{F}_{\text{tol}}\big)$ by Definition 1. Hence, after N_c cycles have been completed at the k^{th} iteration

$$\text{Prob}\big(f(b_k) > \mu \quad \text{or} \quad b_k \notin \mathcal{F}_{\text{tol}}\big) \le \big(1 - \epsilon_\mu\big)^{N_c/2}$$

The right hand side tends to zero as $N_c \to \infty$, yielding the result. □

When $\mathcal{F}_{\text{tol}}$ is the closure of its interior, the continuity of f implies the minimum of f over $\mathcal{F}_{\text{tol}}$ equals $f^\sharp(\mathcal{F}_{\text{tol}})$. When a non-empty $\mathcal{F}$ is the closure of its interior, the preferred choice is $\tau_{\text{c}} = 0$ can be made. This gives b_∞ as a global minimizer of (2) almost surely. The definition of ϕ means $\phi_k \to f^*$ almost surely, meaning $J(x, \phi_k)$ converges to the exact penalty function $J(x, f^*)$ almost surely.

The absence of equality constraints does not guarantee that $\mathcal{F}$ is the closure of its interior. The risk with $\tau_{\text{c}} > 0$ is that the best known point b which is returned by the algorithm satisfies the constraints within tolerance, but is infeasible and the distance between b and the feasible region is large.

4 Numerical testing

The new method was compared against its predecessors OSCARS [20] and OSCARS-II [22] on 50 bound constrained problems in 2–30 dimensions, and on 21 additional problems in 9–60 dimensions. Comparisons with other methods for generally constrained problems are also done using problems from the G-suite and elsewhere. Finally, some tests are done to explore the value of knowing f^*, and how performance varies with dimension and number of constraints on randomized test problems [28].

The numerical tests herein were all performed with $A = 0.9, \beta = 1/3, h_{\text{min}} = 10^{-6}$, $\tau_{\text{stall}} = 10^{-6}$ and $T_{\text{stall}} = 5$. For tests with the T-CELL method of Aragón et al. [3] and stochastic ranking evolutionary search (SRES) [26], $\tau_{\text{c}} = 10^{-4}$ is used.

4.1 Bound constrained only tests

When general constraints are absent, the algorithm minimizes f over Ω. This follows because $v \equiv 0$ and $J(x, \phi) \equiv \max\{f(x), \phi\}$. Since ϕ is always set equal to the best known f immediately, for any sample point x we have $f(x) \ge \phi$. Hence $f(x) = J(x, \phi)$ for all points x at which f has been calculated. This means the algorithm minimizes f when no general constraints are present. The stall test applied to J is identical to the stall test applied to f.

Table 1 Summary of results on problems without general constraints

Method	First 50 problems				21 Higher dim. problems			
	Fevals	Best	FR	Norm nf	Fevals	Best	FR	Norm nf
OSCARS	5897	14	4	2.7643	120,653	0	168	3.4729
OSCARS-II(P)	4158	16	2	1.3105	54,938	6	25	1.2889
CURRENT	3976	20	3	1.2874	48,659	15	17	1.0802

The legend for this table is described in Sect. 4.1

The method was tested on both problem sets used in Price et al. [22]. Test set 1 [20] contains 50 test problems and test set 2 [22] contains an additional 21 largely higher dimensional problems. Ten runs were performed for problems in test set 1, and 30 runs for problems in test set 2. Each run which located a best known point b satisfying

$$b \in \mathcal{F}_{\text{tol}} \quad \text{and} \quad f(b) \leq f^* + \tau_{\text{obj}} \max\{1, |f^*|\} \tag{3}$$

was deemed successful, and halted immediately on satisfying these two conditions. Here $\tau_{\text{obj}} = 10^{-3}$ gives the maximum permitted absolute error (when $|f^*| < 1$) or relative error (when $|f^*| \geq 1$) in f.

Runs which did not find a point satisfying (3) after 50,000 function evaluations (for test set 1) or 250,000 function evaluations (test set 2) were deemed unsuccessful, and halted at that point.

A summary of the results for both test sets is presented in Table 1. For each method, the number of function evaluations taken to find a solution was averaged across all runs for each problem. For each problem, each methods' averages were normalized by dividing by the least of the methods' averages for that problem. The normalized function evaluation counts are averaged across all problems, and listed in the column headed 'norm nf'. The non-normalized averages of the function counts for all runs of all problems are in the 'fevals' column. Columns headed 'best' and 'FR' list the number of problems on which each method had the lowest average function count, and the total number of runs of all problems which ended in failure. Failed runs are costed out at the maximum number of function evaluations when calculating the normalized and non-normalized averages. Doing so artificially reduces both average function counts, with more failed runs tending to yield greater reductions. The reason for listing both types of average function count is that the non-normalized averages are dominated by problems which take many function evaluations to solve. Some problems need more than 100 times as many function evaluations to solve as others.

These results show that the method is competitive with OSCARS-II [22] and superior to the original OSCARS [20] algorithm on bound constrained problems.

4.2 Generally constrained problems: the G-suite

The method was tested on 17 problems from the G-suite [13, 17, 18, 30] and compared against Stochastic Ranking Evolutionary Search (SRES) [26] and the modified

Table 2 Comparison with T-CELL [3] and SRES [4] with all runs using 350,000 function evaluations

		T-CELL			This method			SRES results from [4]		
	f^*	Best	Mean	Worst	Best	Mean	Worst	Best	Mean	Worst
1	-15	-15.0	-15.0	-15.0	-14.99946	-14.99799	-14.99639	-15.0	-15.0	-15.0
2	-0.8036191	-0.801367	-0.752975	-0.687827	-0.628909	-0.530838	-0.430749	-0.803	-0.784	-0.734
3	-1	-1.0	-1.0	-1.0	-0.999611	-0.996873	-0.993609	-1.0	-1.0	-1.0
4	-30,665.539	-30,655.539	-30,655.538	-30,655.538	-30,655.66	-30,655.65	-30,655.65	-30,655.539	-30,655.48	-30,654.22
5	5126.4981	5126.6255	5378.2678	6112.1181	5043(inf)	5203(inf)	5675(inf)	5126.497	5130.792	5153.757
6	-6961.814	-6961.8139	-6961.8139	-6961.8139	-6962.046	-6962.040	-6962.021	-6961.814	-6863.645	-6267.787
7	24.306209	24.3209	24.6534	25.1347	24.31826	24.40095	24.54424	24.310	24.417	24.830
8	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095	-0.095	-0.095
9	680.63006	680.63	680.65	680.70	680.6311	680.6376	680.6496	680.63	680.646	680.697
10	7049.248	7050.8342	8020.7551	9054.2923	7049.638	7265.607	9423.380	7050.194	7423.434	8867.844
11	0.75	0.7499	0.7499	0.7499	0.749900	0.749900	0.749901	0.750	0.750	0.751
12	-1	-1.0	-1.0	-1.0	-1	-1	-1	-1	-1	-1
13	0.0539415	0.054638	0.458857	0.994983	0.0539428	0.0540931	0.0575916	0.053	0.061	0.128
14	-47.76489	-47.517100	-45.310800	-43.272382	-47.75237	-47.69656	-47.61821	-41.551	-41.551	-40.125
15	961.71502	961.71502	963.37482	970.59467	961.71502	961.71503	961.71504	961.715	961.731	962.008
16	-1.905155	-1.905155	-1.905155	-1.905155	-1.9051686	-1.905015	-1.904421	-1.905	-1.703	-1.587
18	-0.86602	-0.86596	-0.78455	-0.62977	-0.865944	-0.865521	-0.864334	-0.86600	-0.78600	-0.45700

The new method found feasible points on all runs of all problems except g5 and g10. All runs of the new method returned infeasible points on g5, and two runs did so on g10

T-CELL algorithm [3]. Results are listed in Table 2, where the best, worst and mean objective function values are listed for sets of 30 runs, each using 350,000 function evaluations and $\tau_c = 10^{-4}$. The stopping condition (3) was not used: all runs were halted at 350,000 function evaluations and the best point reported. These conditions match those used for T-CELL [3] and SRES [4], allowing a direct comparison with their results. Results for SRES and T-CELL listed in Table 2 are given to the same number of significant figures as listed by Aragón et al. [3] and Cagnina et al. [4].

We regard any solution as acceptable if it satisfies (3) with $\tau_c = 10^{-4}$ and $\tau_{obj} = 10^{-3}$. Some results from Aragón et al. [3] and Cagnina et al. [4] are not listed with sufficient accuracy to determine if this standard was met. In such cases, it is assumed that the required accuracy was achieved. Firstly, looking at the best points found by each method over the 30 runs SRES, T-CELL and the current method had 16, 15, and 15 acceptable solutions respectively. For the mean scores the new method was acceptable on 11 problems, and the other two methods each on 9. For the worst points found, T-CELL and this method were acceptable on 9, and SRES on 8. These results show the new method is competitive on generally constrained optimization problems.

The non-zero constraint tolerance permits points which are slightly better than optimal to be returned as the solution, and several such points feature in Table 2.

4.3 Other generally constrained tests

The method was also tested on a wider set of generally constrained problems. This set is the 17 G-suite problems used above, along with the Gomez3 problem [8], problems 3.3, 4.3, 4.4 and 4.5 from Floudas and Pardalos [6], problems 12.2.3 and 12.2.4 from Floudas et al. [5], the pentagon and equil problems from Lukšan and Vlček [12] and the cylinder-sphere problem (below).

The best known points for problems 4.4 and 4.5 [6] were updated to $f^* = -3.13363591$ at (0, 3, 0, 1) and $f^* = -13.401903555$ at (1/6, 2, 4, 1/2, 0, 2) respectively.

Problems 12.2.3 and 12.2.4 from Floudas et al. [5] contain a mix of real and binary variables. Each binary variable z_i was handled by using a real variable $x_i \in [-0.5, 1.5]$ and rounding x_i to the nearest member of $\{0, 1\}$ to get the binary value z_i before evaluating the objective and constraint functions.

The problems in Lukšan and Vlček [12] are nonsmooth local optimization test problems, and as such are not finitely bounded above and below in all dimensions. To rectify this $\Omega = [-2, 2]^6$ was used for the pentagon problem, and $\Omega = [0, 1]^8$ for equil. The last problem (cylinder-sphere) has similar characteristics. It is

$$\min x_2 \quad \text{subject to} \quad \|x\|_2^2 \le 1 \quad \text{and} \quad (x_1 + a)^2 + (x_2 + a)^2 \ge a^2 + (a - 1)^2$$

with $x \in [-2, 2]^{10}$ and $a = 0.25$. The solution is $x_2^* = -1$ and $x_i^* = 0$ for all $i \ne 2$ with an optimal objective function value of $f^* = -1$. There is one proper local minimizer with $f = 0$ at $x_1 = -1$ and $x_i = 0$ otherwise.

The 10 problems not from the G-suite were tested under the same conditions as the G-suite. Results appear in Table 3. The method found feasible points on all 300 runs,

Table 3 Results for generally constrained problems not in the G-suite

	n	q_i	q_e	f^*	best	mean	worst
Gomez3	2	1	–	-0.9711	-0.9711137	-0.9711134	-0.9711128
FP 3.3	6	6	–	-310	-310.0128	-307.74332	-297.24291
FP 4.3	4	2	1	-4.5142	-4.5148734	-4.5148357	-4.5148064
FP 4.4	4	2	1	-3.13363	-3.1337287	-3.1336766	3.1335644
FP 4.5	6	3	3	-13.4019	-13.402012	-13.402012	-13.402012
Pentagon	6	15	–	-1.859	-1.8597167	-1.859647	-1.8594908
Equil	8	–	1	0	0.00004187	0.14386	0.5575356
Cyl-sphere	10	2	–	-1	-0.9997767	-0.9994841	-0.9992009
CF12.2.3	7	2	1	4.5796	4.5792389	4.5792201	4.5792571
CF12.2.4	11	3	3	-0.94347	-0.9437524	-0.9421522	-0.9318465

and optimal points (as judged by (3)) on 274 runs. There were 7, 15 and 4 runs with non-optimal points on problem FP3.3, equil and CF12.2.4 respectively. These results show that the method can also be effective on problems that are nonsmooth or have some binary variables.

The method was also tested with $\tau_c = 0$ on the 15 generally constrained problems which do not have equality constraints. The testing regime was otherwise identical to that for the G-suite. With both $\tau_c = 0$ and $\tau_c = 10^{-4}$ the method found feasible points on 449 runs out of 450, with one fail each on problem G10. The number of successful runs were 358 and 356 respectively, showing the method is effective with zero constraint tolerance on inequality constrained problems.

4.4 Tests using generally constrained Schoen functions

Tests were also performed on a modified set of Schoen test problems [28] of the form

$$f(x) = \frac{\sum_{i=1}^{N} s_i \prod_{j=1, j\neq i}^{N} \|x - z_j\|_2^2}{\sum_{i=1}^{N} \prod_{j=1, j\neq i}^{N} \|x - z_j\|_2^2} - \sum_{j=1}^{m} \max(g_j(x), 0) \quad \Omega = [0,1]^n \tag{4}$$

subject to the constraints $g_j \leq 0,\ j = 1, \ldots, m$. Each z_j is chosen randomly from $[0,1]^n$ and its associated s_j is selected from a normal distribution with mean 5 and variance 1. Each constraint takes the form:

$$g_j(x) = \pm\left(\|x^* - y_j\|_2 - \|x - y_j\|_2\right) - 0.2\,\theta_j \|x^* - y_j\|_2$$

where each $\pm$ sign and $y_j \in \Omega$ are chosen randomly to yield either a convex or concave hyperspherical constraint centred on y_j. The $\theta_j \in \{0, 1\}$ are also chosen randomly, where $\theta_j = 0$ makes the constraint active at x^*; otherwise it is inactive at x^*.

The global minimizer x^* of the left hand term of f on Ω is the z_j with the least corresponding s_j [28]. This s_j is reduced by $10^{-3}\max(|s_j|, 1)$ to reduce the risk another minimum is within tolerance of the global minimum. Now $x^* \in \mathcal{F}$ and the right hand term in (4) is zero on $\mathcal{F}$, so x^* solves the generally constrained Schoen problem.

Tests were run on batches of 100 random problems in 5, 10 and 20 dimensions, with $N = 40$ and 0, 1, 3, 9 or 27 constraints. One run per problem was performed, with all runs using $\tau_c = 10^{-4}$ and $\tau_{\text{obj}} = 10^{-3}$. Each run halted successfully once (3) was satisfied, or as a fail if 350,000 function evaluations was reached.

Results appear in Table 4, and show steadily increasing difficulty with dimension. These results also show adding a small number of constraints dramatically increases the problem difficulty. This is because every constaint is active or nearly active at the solution, which significantly reduces the basin of attraction of x^* compared to other $z_j \in \mathcal{F}$. On such problems convergence to a proper local minimizer sometimes occurs. As more constraints are added, few or no z_j remain in $\mathcal{F}$ and the problem largely reduces to obtaining feasibility. Finding the basin of x^* is easier, but the constraints make estimating x^* accurately more computationally expensive.

Table 4 These generally constrained Schoen problem results list the average number of function evaluations over 100 problems, and number of failed runs (FR)

n	Unconstrained		1 Constraint		3 Constraints		9 Constraints		27 Constraints	
	FR	Fevals	FR	Fevals	FR	Fevals	FR	Fevals	FR	Fevals
5	0	4038	1	13,009	1	19,409	1	10,057	0	6393
10	0	5751	11	53,066	26	115,423	7	33,940	2	45,741
20	2	20,125	21	91,456	43	164,543	24	111,852	11	135,805

Failed runs are costed at the maximum number of function evaluations in the averages

Table 5 This table compares the new method when the optimal objective function value f^* is, and is not known

	Fevals	Norm nf	Best on	Successful runs	Feasible runs
f^* Unknown	126,056	1.263	17	619	781
f^* Known	161,044	2.142	8	525	711

Thirty runs were performed for each of the 27 problems giving 810 runs in total. On g2 and g10 neither method was best because both methods took 350,000 function evaluations on all runs

4.5 The value of knowing f^*

If there are no general constraints, setting $\phi = f^*$ and using the updating for ϕ both ensure $J(x) = f(x)$ holds at all sample points which have been used. This means both versions of the algorithm are functionally identical. Hence comparisons between estimating ϕ and using $\phi = f^*$ are made only on the 27 generally constrained problems given above. The Schoen problems were not used.

For each problem, 30 runs were performed with $\tau_c = 10^{-4}$ for two versions of the algorithm: one estimated ϕ as described above, and the other used $\phi = f^*$ at all times. Both versions halted when (3) was satisfied. This allows the relative speeds of both versions to be compared on the easier problems. Results are presented in Table 5. These show clearly that setting $\phi = f^*$ when f^* is known is detrimental to the algorithm's performance. The advantage of not setting $\phi = f^*$ is that as ϕ is adjusted, the induced kink in J from the $\max\{f, \phi\}$ term moves around. This movement enhances the algorithm's ability to traverse general constraint boundaries towards the solution (Table 5).

5 Conclusion

The performance of the new method on generally constrained problems is similar to that of the filter version F-OSCARS [21] of OSCARS in terms of function counts. In terms of overheads, the new method is almost twice as efficient. Moreover, on problems without general constraints, or where no general constraints are active in the vicinity of the solution, F-OSCARS becomes equivalent to OSCARS, and the latter is markedly inferior to the new method.

The new method has been shown to converge almost surely in exact arithmetic. Numerical results show that the method is effective on a wide range of bound and generally constrained problems, including nonsmooth problems. It compares well against other similar methods on problems from the G-suite. There is scope for improvement in the exploitation phase via a local search. This would be particularly beneficial on generally constrained problems as it would help the method traverse along any general constraint boundaries towards the solution.

Acknowledgements The authors would like to thank both anonymous referees for many helpful comments leading to an improved paper.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data Availability There are no data.

Declarations

Conflict of interest: The authors declare that there are no conflicts of interest.

References

1. Ali, M.M., Golakhani, C.M., Zhuang, J.: A computational study of different penalty approaches for solving constrained global optimization problems with the electromagnetism-like method. J. Optim. **63**, 403–419 (2014)
2. Appel, M.J., Labarre, R., Radulović, D.: On accelerated random search. SIAM J. Optim. **14**, 708–731 (2003)
3. Aragón, V.S., Esquivel, S.C., Coello, C.A.C.: A modified version of a T-cell algorithm for constrained optimization problems. Int. J. Numer. Methods Eng. **84**, 351–378 (2010)
4. Cagnina, L., Esquivel, S., Coello, C.A.C.: A bi-population PSO with a shake-mechanism for solving constrained numerical optimization. In: 2007 IEEE Congress on Evolutionary Computation (CEC'2007), Singapore 2007, pp. 670-676. IEEE Press, New York (2007)
5. Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gümüs, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A., Schweiger, C.A.: Handbook of test problems in local and global optimization. In: Nonconvex Optimization and its Applications, vol. 33. Kluwer, Dordrecht (1999)
6. Floudas, C.A., Pardalos, P.M.: A collection of test problems for constrained global optimization problems. In: Lecture notes in Computer Science, vol. 455. Springer, Berlin (1990)
7. Hedar, A.-R., Fukushima, M.: Derivative free filter simulated annealing method for constrained continuous global optimization. J. Glob. Optim. **35**, 521–549 (2006)
8. Gomez, S., Levy, A.: The tunneling method for solving the constrained global optimization problem with several non-connected feasible regions. In: Dold, A., Eckmann, B. (eds.) Lecture Notes in Mathematics, vol. 909, pp. 34–47. Springer (1982)
9. Jones, D.R.: The DIRECT global optimization algorithm. In: Floudas C.A., Pardalos P.M. (eds.) Encyclopaedia of Optimization, pp. 433–440. Springer, Boston (2001)
10. Jones, D.R., Pertunnen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**, 157–181 (1993)
11. Liu, Z., Li, Z., Zhu, P., Chen, E.: A parallel boundary search particle swarm optimization algorithm for constrained optimization problems. Struct. Multidiscip. Optim. **58**, 1505–1522 (2018)
12. Lukšan, L., Vlček, J.: Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report 798, Prague: Institute of Computer Science, Academy of Sciences of the Czech Republic (2000)
13. Koziel, S., Michalewicz, Z.: Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evol. Comput. **7**, 19–44 (1999)
14. Mazhoud, I., Hadj-Hamou, K., Bigen, J., Joyeux, P.: Particle swarm optimization for solving engineering problems: a new constraint handling mechanism. Eng. Appl. Artif. Intell. **26**, 1263–1273 (2013)
15. Macêdo, M.J.F.G., Karas, E.W., Costa, M.F.P., Rocha, A.M.A.C.: Filter-based stochastic algorithm for global optimization. J. Glob. Optim. **77**, 777–805 (2020)
16. Mezura-Montes, E., Coello, C.A.C.: Constraint handling in nature-inspired numerical optimization: past, present and future. Swarm Evol. Comput. **1**, 173–194 (2011)
17. Mezura-Montes, E., Miranda-Varela, M.E., del Carmen Gómez-Ramón, R.: Differential evolution in constrained numerical optimization: an empirical study. Inf. Sci. **180**, 4223–4262 (2010)
18. Michalewicz, Z.: Genetic algorithms, numerical optimization and constraints. In: Eshelman, L.J. (eds) Proceedings of the 6th International Conference on Genetic Algorithms, pp. 151–1580. Morgan Kaufman, San Mateo California (1995)
19. Nuñez, L., Regis, R.G., Varela, K.: Accelerated random search for constrained global optimization assisted by radial basis function surrogates. J. Comput. Appl. Math. **340**, 276–295 (2018)
20. Price, C.J., Reale, M., Robertson, B.L.: One side cut accelerated random search: a direct search method for bound constrained global optimization. Optim. Lett. **8**, 1137–1148 (2014)

21. Price, C.J., Reale, M., Robertson, B.L.: Stochastic filter methods for generally constrained global optimization. J. Glob. Optim. **65**, 441–456 (2016)
22. Price, C.J., Reale, M., Robertson, B.L.: OSCARS-II: an algorithm for bound constrained global optimization. J. Glob. Optim. **79**, 39–57 (2021)
23. Regis, R.G.: A hybrid surrogate assisted accelerated random search and trust region approach for constrained black-box optimization. In: Lecture Notes in Computer Science 13164, pp. 162–177. Springer, Cham (2022)
24. Rocha, A.M.A.C., Costa, M.F.P., Fernandes, E.M.G.P.: A filter-based fish swarm algorithm for constrained global optimization: theoretical and practical issues. J. Glob. Opt. **60**, 239–263 (2014)
25. Regis, R.G., Shoemaker, C.A.: Constrained global optimization of expensive black box functions using radial basis functions. J. Glob. Opt. **31**, 153–171 (2005)
26. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Comput. **4**, 284–294 (2000)
27. Sampaio, P.R.: DEFT-FUNNEL: an open-source global optimization solver for constrained grey-box and black-box problems. Comput. Appl. Math. **40**, 176–211 (2021)
28. Schoen, F.: A wide class of test functions for global optimization. J. Glob. Optim. **3**, 133–137 (1993)
29. Spettel, P., Beyer, H.-G.: Matrix adaptation evolution strategies for optimization under nonlinear equality constraints. Swarm Evol. Comput. **54**, 100653 (2020)
30. Wang, Y., Cai, Z., Zhou, Y.: Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. Int. J. Numer. Methods Eng. **77**, 1501–1534 (2009)
31. Xu, P., Luo, W., Lin, X., Qiao, Y.: Evolutionary continuous constrained optimization using random direction repair. Inf. Sci. **566**, 80–102 (2021)