Journal of Global Optimization (2021) 79:39–57
https://doi.org/10.1007/s10898-020-00928-6

OSCARS-II: an algorithm for bound constrained global optimization

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

Received: 23 July 2019 / Accepted: 13 July 2020 / Published online: 8 August 2020

Abstract
An adaptation of the OSCARS algorithm for bound constrained global optimization is presented, and numerically tested. The algorithm is a stochastic direct search method, and has low overheads which are constant per sample point. Some sample points are drawn randomly in the feasible region from time to time, ensuring global convergence almost surely under mild conditions. Additional sample points are preferentially placed near previous good sample points to improve the rate of convergence. Connections with partitioning strategies are explored for OSCARS and the new method, showing these methods have a reduced risk of sample point redundancy. Numerical testing shows that the method is viable in practice, and is substantially faster than OSCARS in 4 or more dimensions. Comparison with other methods shows good performance in moderately high dimensions. A power law test for identifying and avoiding proper local minima is presented and shown to give modest improvement.

Keywords Direct search · Numerical results · Accelerated random search · Power law test

1 Introduction

This paper looks at the bound constrained global optimization problem

$$\min_{x\in\Omega} f(x) \tag{1}$$

where $\Omega \subset \mathbb{R}^n$ is the box shaped feasible region aligned with the coordinate axes. The objective function $f(x)$ is assumed to be 'black-box' in the sense that, if provided with a point x, it will return $f(x)$ and no other information. The feasible region Ω is defined by simple bounds via

$$\Omega = \left\{x \in \mathbb{R}^n : L_i \le x_i \le U_i \quad \forall i = 1, 2, \ldots, n\right\}.$$

✉ C. J. Price
C.Price@math.canterbury.ac.nz

[1] Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand

The upper (U_i) and lower (L_i) bounds on each element x_i of x are necessarily finite, where x_i is the i^{th} element of x, etc. It is assumed that $L_i < U_i$ for all $i = 1, \ldots, n$, without loss of generality.

This problem has been the subject of a plethora of approaches in many different papers, a selection of which can be found in [8,10,15,19,31,32] and the references therein. In spite of its apparently simple form, there are many different variations on this problem, and each in turn has many different ways it can be addressed. Variations include the the dimension n of the problem, the level of differentiability of the objective function f, the number of derivatives which are available to the algorithm, and the cost of calculating function values, and derivatives, where applicable. Additionally, finding one global minimizer might suffice, or all global minimizers might be required. The latter is closely related to approximating a near optimal level set [13].

A very simple method for solving (1) is Pure Random Search (PRS), which calculates f at N random points in Ω and returns the lowest as an estimate of a global minimizer. This method has very low overheads, does not require derivatives, and can be shown to converge almost surely. Nevertheless it is seldom used as it is intolerably slow in practice.

An obvious strategy for improving PRS is to sample preferentially in areas where the objective appears to be relatively low. An example of this is Accelerated Random Search (ARS) [3] which uniformly samples in box shaped sub-regions of Ω. A finite sequence of successively smaller boxes containing the current best known point are polled until either a new best point is found, or the sequence is exhausted. The first box in each such sequence is always Ω. Each subsequent box is formed by taking a scaled version of Ω, translating it so that it is centred on the best known point, and intersecting it with the feasible region. If an improving point is found, a new sequence is constructed centred on the new best point. Otherwise, if the sequence of boxes is exhausted without locating a new best point, that sequence of boxes is repeatedly re-used until a new best point is found or the algorithm halts. A version of ARS with an exponential rate of convergence is described in [27]. However this result applies only when close to the global minimizer, and the tendency of ARS to sample largely in the vicinity of the best known point can mean locating a neighbourhood of the global optimizer might not happen quickly.

The way the sequence of sampling boxes is formed for ARS is very simple, but can have undesirable properties at times. For instance, let ARS generate a non-improving sample point s with the best known point being b. Since a point better than b has not been found, the next sampling box is a subset of its predecessor. If s is far from b, s will not be part of the new sampling box. However points on the opposite side of b to s will also be excluded from the new sampling box, and s provides no evidence that these points are worse than b. Moreover, if s is sufficiently near b, ARS will retain the area around s where f is known to be worse.

A more sophisticated approach [25] is to shrink the sampling box along one coordinate axis only, so that s lies outside the new box, but b does not. More precisely, one face of the box is moved inwards to lie between s and b, and the remaining $2n - 1$ faces are left in their original positions. The face that is shifted is the one orthogonal to the largest component of $s - b$. This forms the nub of the One Side Cut Accelerated Random Search (OSCARS) algorithm [25].

An important feature of the way OSCARS shrinks the polling box at each iteration is that every face which is closer to b than to s remains unmoved. Clearly $f(s) \geq f(b)$ does not constitute evidence that points on the opposite side of b to s are also higher than b. In fact, if the sample box is sufficiently small, and f is C^1, then f will be approximately linear on the sample box. In this case $f(s) \geq f(b)$ provides weak evidence that points on the opposite

side of b to s are likely to be better. This is in stark contrast to ARS which moves all faces inwards towards b after a non-improving sample point has been generated.

Herein we present a variation on OSCARS. The first significant change is that the box can be cut by several hyperplanes aligned with different coordinate axes in the same iteration. Like OSCARS, each of these hyperplanes separates b from s. The purpose of this change is to reduce the number of iterations needed to generate a small box around the control point when n is large. The second change is that the lengths of the cycles are directly limited. A new examination of the ability of both algorithms to avoid redundancy in the sample points chosen is also presented. The new method is tested both with and without a power law test for identifying and avoiding suspected proper local minima.

Almost sure convergence is shown in exact arithmetic when f is bounded below, Lebesgue measurable, and lower semi-continuous over Ω. If these conditions do not hold, the algorithm can be executed but no guarantees of convergence are established. In addition, connections with partitioning strategies, and the implications for the non-asymptotic convergence behaviour of these methods are explored.

Both OSCARS and the new method have low overheads which are constant per function evaluation. This is in contrast to methods such as DIRECT [14] which store all function evaluations and have overheads per function evaluation which increase as the number of function evaluations already performed increases. This is especially advantageous on functions which are cheap to evaluate, as the low overheads allow many more function evaluations to be performed in the same length of time [25]. Methods with low overheads can also be usefully employed for minimizing surrogates on methods which select iterates by, for example, minimizing a radial basis function approximation of the objective [28].

Partitioning methods [1,7,17,18,22,30] typically subdivide Ω into sub-regions which overlap only at their boundaries, and then calculate f at various sample points in each sub-region. The partition is then refined, or updated in some way to reflect the information gained from the sample points, and the process repeated as necessary. Adapting the partition from time to time allows areas found to be lower to be explored more heavily. One advantage of partitioning is that it reduces the risk of a sample point being placed unintentionally very close to a previous sample point, resulting in a wasted function evaluation. Generally speaking, the fewer sample points that are placed in each sub-region, the less the risk of wasted function evaluations. This requires that the partition be refined as the total number of sample points increases. Hence, for many of these methods the storage requirements and overheads per function evaluation tend to rise as the number of function evaluations increases. A number of methods, both stochastic [23] and deterministic [14] push this approach to its limit, by having one sample point per sub-region. These methods store and use all previous sample points when generating further sample points. Consequently as the number of iterations increases, the overheads per iterate of these methods climb steadily, and eventually dominate the cost of computing the objective function [25].

The general structure of the paper is as follows. The new algorithm is described in detail in the next section. Section 3 examines the convergence properties. Numerical results are presented and discussed in Sect. 4, and concluding remarks are in Sect. 5.

2 The OSCARS-II algorithm

The structure of the algorithm consists of a series of alternating cycles, each of limited length. Each cycle is in turn made up of a finite number of passes. Loosely speaking, a pass

is performed by constructing a sequence of nested sample boxes and drawing one sample point randomly from each box in turn. The act of drawing one sample point s randomly from one box, calculating $f(s)$, and constructing the next sample box constitutes one iteration of the algorithm.

Each pass begins with a control point c where $f(c)$ is known. The aim of a pass is to find a point which is better than the control point. If this aim is achieved, the pass is called successful, and ends immediately. Otherwise the pass ends unsuccessfully when, for example, the sample box is too small, or the cycle length limit is reached. If the pass ends successfully, the improving point becomes the new control point, and another pass is begun. If the pass ends unsuccessfully, then the cycle containing that pass also ends.

The cycles alternate in the sense that odd numbered cycles begin with a random control point, and even numbered cycles use the best known point as the initial control point. This is referred to as the SORC strategy, where SORC stands for 'start odd cycles with a random control.' The purpose of this is to avoid focusing effort too heavily in the vicinity of the current best known point.

At each iteration the algorithm uses a control point c_k and a current sample box Ω_k, where $c^{(k)} \in \Omega_k$, and $\Omega_k \subseteq \Omega$. The counter k is the iteration number, and it is attached either as a subscript, or as a superscript (k) on various quantities to refer to the values taken in iteration k. A superscript (k) is used on vector quantities where subscripts refer to individual elements of that vector. Iteration k's sample box Ω_k is defined by the upper and lower bounds $u^{(k)}$ and $\ell^{(k)}$ via

$$\Omega_k = \{x \in \Omega : \ell^{(k)} \leq x \leq u^{(k)}\}.$$

A random sample point $x^{(k)} \in \Omega_k$ is chosen, and if $x^{(k)}$ is not a better point than $c^{(k)}$ (i.e. $f(x^{(k)}) \geq f(c^{(k)})$) the box Ω_k is shrunk, and the process repeated. This 'sample and shrink' continues until one of the following four reset conditions is met:

(a) IMPROVING RESET: An improving point is found. That is to say $f(x^{(k)}) < f(c^{(k)})$;
(b) BOX SIZE RESET: The box Ω_k falls below a minimum size $h_{\min}$ as measured by the infinity norm of its main diagonal;
(c) CYCLE LENGTH RESET: The number of iterations j since the cycle began reaches the cycle length limit $J(m)$, where m is the current cycle number; or
(d) POWER LAW FIT RESET: A power law test indicates convergence to a proper local minimizer. This can only be used on odd numbered cycles.

Each of the above four outcomes is called a reset, and marks the completion of one pass of the method. If an improving reset occurs, the method sets the new control point $c^{(k+1)} = x^{(k)}$ and resets the new sampling box Ω_{k+1} to Ω. All non-improving resets mark not only the end of a pass, but also the end of a cycle. The power law fit reset is optional, and the method is tested both with and without it.

The process of shrinking (or cutting) a box is as follows. Let c be the current control point, and let x be a sample point generated in the box Ω. The component of the vector $x - c$ which has the largest magnitude is identified. Let this component be along the ith coordinate axis with magnitude M. A cut orthogonal to the ith axis is put through Ω_k, where this cut passes through the point $(1 - A)x + Ac$, where $0 < A < 1$. The part of Ω_k that is retained is the part containing c. At this stage, this cut is exactly that used in the OSCARS algorithm. An illustration of this cut appears in Figure 3 of [26].

OSCARS- II also performs an additional cut for each other coordinate axis along which the component of $x - c$ is at least βM in magnitude, where $0 < \beta < 1$. This cut passes through the point $(1 - \gamma_r A)x + \gamma_r Ac$ where $\gamma_r = |x_r - c_r|/M$ is the relative magnitude of

the component along coordinate axis r. The factor γ_r means the cut passes relatively closer to x for faces orthogonal to coordinate directions along which $x - c$ has smaller magnitude components.

Algorithm 1 OSCARS- II

1. *Choose a random point* $x^{(0)} \in \Omega$. *Calculate* $f_s = f_c = f_b = f(x^{(0)})$. *Set* $k = m = 1$, *set* $b = c = x^{(0)}$, *and set* $u = U$ *and* $\ell = L$. *Choose* $A, \beta \in (0, 1)$. *Select* $h_{\min} > 0$.
2. *If stopping conditions hold, then exit. Set* $j = 0$. *If* $f_c < f_b$ *set* $b = c$ *and* $f_b = f_c$.
3. *Randomly choose* $x^{(k)}$ *subject to* $\ell \le x^{(k)} \le u$. *Calculate* $f^{(k)} = f(x^{(k)})$.
4. *If* $f^{(k)} < f_c$ *then set* $c = x^{(k)}$, *set* $f_c = f^{(k)}$, *set* $u = U$, *set* $\ell = L$, *increment* k, *and go to step 2.*
5. *Set* $M = \left\| x^{(k)} - c \right\|_\infty$ *and set* $\gamma_i = |x_i^{(k)} - c_i|/M$ *for* $i = 1, \ldots, n$.
6. *For each* $i \in 1, \ldots, n$ *perform the following steps:*
   (a) *Calculate the cut position* $K_i = (1 - \gamma_i A)x_i^{(k)} + \gamma_i A c_i$.
   (b) *If* $x_i^{(k)} - c_i \ge \beta M$, *set* $u_i = K_i$, *otherwise if* $x_i^{(k)} - c_i \le -\beta M$, *set* $\ell_i = K_i$.
7. *Increment* j *and* k. *If* $\|u - \ell\|_\infty \le h_{\min}$ *or* $j \ge J(m)$, *go to the next step. Otherwise, if the power law test does not indicate convergence to a local minimizer, go to step 3.*
8. *Set* $u = U$ *and* $\ell = L$. *If* $f_c < f_b$ *set* $b = c$ *and* $f_b = f_c$. *Increment* m.
9. *If* m *is odd, randomly choose* $c \in \Omega$ *and set* $f_c = f(c)$; *otherwise set* $c = b$ *and* $f_c = f_b$. *Set* $f_s = f_c$ *and go to step 2.*

Here f_b, f_c and f_s are the objective function values at the current best known point b, the current control point c, and the initial control point used at the beginning of the current cycle. The quantities c, b, ℓ, and u are not explicitly indexed by k in the algorithm description, however the sequences of these quantities are needed to analyse the algorithm. To this end a superscript (k) is affixed to these quantities to identify the values they take in step 3 of iteration k. This yields the sequences of best known points $\{b^{(k)}\}_{k=1}^{\infty}$ and control points $\{c^{(k)}\}_{k=1}^{\infty}$. Subscripts on a vector refer to a specific element of that vector.

2.1 Cycles, passes, and iterations

Cycles and passes are made up of a number of consecutive iterations. Iterations are indexed by the iteration number k. Each iteration executes step 3 once, generating a random sample point $x^{(k)} \in \Omega_k$ and calculating $f^{(k)}$. One or more of the steps following step 3 are performed. The iteration ends when execution transfers back to either step 2 or step 3.

Each pass consists of a number of consecutive iterations (iterations $k + 1$ to $k + p$ say). For these iterations to form a pass three conditions must hold:

1. Either $k = 0$ or there is a reset in iteration k;
2. There is a reset in iteration $k + p$; and
3. There are no resets of any sort in iterations $k + 1$ to $k + p - 1$.

The situation for cycles is similar. If a cycle consists of iterations $k + 1$ to $k + p$, then firstly either $k = 0$ or there was a cycle ending reset in iteration k. Secondly, there must be a cycle ending reset in iteration $k + p$, with no cycle ending resets in iterations $k + 1$ to $k + p - 1$. In contrast to a pass, a cycle is permitted to have improving resets in iterations $k + 1$ to $k + p - 1$. Hence a cycle consists of one or more complete passes, whereas each pass is contained entirely within one cycle. Cycles are indexed by m, passes are not indexed.

2.2 The cycle length reset

The original OSCARS algorithm had only two reset mechanisms: an improving reset (a), and a box size reset (b). The inclusion of the cycle length reset means that at least m points are chosen randomly from Ω after $\sum_{i=1}^{m} J(i)$ iterations, which guarantees almost sure convergence to an essential global minimizer. The maximum cycle length $J(m)$ is required to be a positive, strictly increasing function of the cycle counter m with the property that $J(m) \to \infty$ as $m \to \infty$. The optional power law reset operates similarly to the cycle length reset when the current cycle appears to be converging to a proper local minimizer.

The cycle length resets are used to improve the numerical performance of the algorithm by eliminating undesirable behaviour that can occasionally occur with the original OSCARS algorithm. This behaviour occurs most prominently on the Hartmann 6 problem. This behaviour can occur if the control point is near a stationary point which is not a global minimizer. In such a case OSCARS can generate a large number of passes, each of which ends in an improving reset. This means all of these passes belong to a single cycle. Each such improving reset yields a negligible improvement in the objective function f, and the improving iterate is only marginally different from the control point. Since all of this occurs in a single cycle, the SORC strategy is not invoked, and the search is strongly focused in the vicinity of the non-optimal stationary point. If the basin of the global minimizer(s) is distant from that stationary point, OSCARS will have only a very low chance of detecting it in each pass. Using a cycle reset curtails such cycles. The next odd numbered cycle will start with a randomly generated control point, and will usually have a much better chance of locating the distant basin more quickly.

2.3 The power law fit reset

Stopping rules based on power law distributions for near minimal values of f have been explored extensively [9,11], including for local optima via the Kolmogorov–Smirnov test [24]. Herein we use these ideas to test for convergence to a proper local minimizer. If this convergence is detected, the current cycle is ended to prevent the method from wasting effort by refining a proper local minimizer to high accuracy. A record of the B best function values $\phi_1, \ldots, \phi_B$ seen in the current cycle is kept, where these values are assumed to be in increasing order for simplicity. This data is used to estimate the lowest function value that cycle is likely to reach. If this value is significantly above the best known function value, the empirical cumulative distribution function (ECDF) of the data is compared against two relevant power law distributions using a Kolmogorov–Smirnov test. If either fit is accepted, the cycle halts.

The power law fit test automatically fails if less than B data values are held, or if $\phi_1 > f_s - \tau_{\text{ks}}$ where f_s is the value of f at the initial control point for that cycle. The latter ensures that f is not so nearly flat compared to the power law fit tolerance τ_{ks} that the test would be misleading. The initial estimate of the lowest value the cycle is likely to reach is formed by assuming that the data are drawn randomly from a power law distribution of the form

$$\rho_{cdf}(\phi) = \left(\frac{\phi - \phi^\sharp}{\phi_B - \phi^\sharp} \right)^n$$

where $\phi^\sharp$ is the lowest value for which the probability that $B - 2$ points drawn randomly from this distribution will have at least a 1% chance of all lying in the interval $[\phi_1, \phi_B]$. If $\phi^\sharp > f_b + 10\tau_{\text{ks}}$, where τ_{ks} is the power law fit tolerance in f, then Kolmogorov–Smirnov goodness of fit tests are performed at the 5% level against two power law distributions:

$$\rho_{cdf}(\phi) = \left(\frac{\phi - \phi_1^\sharp}{\phi_B - \phi_1^\sharp}\right)^n \quad \text{and} \quad \rho_{cdf}(\phi) = \left(\frac{\phi - \phi_2^\sharp}{\phi_B - \phi_2^\sharp}\right)^{(n/2)}$$

where $\phi_1^\sharp$ and $\phi_2^\sharp$ are calculated as per $\phi^\sharp$, but with a 50% chance of $B-2$ random points lying in the interval $[\phi_1, \phi_B]$. If either power law is accepted by the goodness of fit test, the current cycle is ended. The first power law distributions includes many local minimizers of non-smooth locally Lipschitz functions, and the second includes local minimizers with positive definite Hessians when f is twice continuously differentiable [9].

The cost of this test can be significant, especially in low dimensions. This cost can be reduced by, say, performing this test only every 5^{th} or 10^{th} iteration, and automatically rejecting both power law fits in all other iterations. Even if the power law test is not used at all, the cycle length resets will limit the number of function evaluations wasted in accurately finding proper local minimizers, but to a lesser extent.

The difference between the algorithm with (hereafter OSCARS- II(P)) and without (hereafter OSCARS- II(N)) the power law test is slight. Algorithm 1 lists it with the power law test implemented. To remove it, the second sentence of step 7 is replaced with "Otherwise, go to step 3." Recording the initial random control point's function value f_s in steps 1 and 9 is no longer needed, and can be removed.

2.4 Avoiding the neighbourhood of disinterest

Lower semi-continuity of f on Ω implies all level sets of the form $\{x \in \Omega : f(x) \leq \eta\}$ are closed. Hence, for all s such that $f(s) > f(c)$, there is a 'neighbourhood of disinterest' $N_s(f(c))$ centred on s such that $f(x) > f(c)$ for all $x \in N_s(f(c))$.

The main justification in [25] of the 'one side cut' method of shrinking the sampling box, is that in contrast to ARS, it guaranteed that a neighbourhood of a failed sample point would not be resampled in the same pass. This reduces the risk of sampling in the neighbourhood of disinterest of a previous sample point in the same pass. This is now extended to sample points from different passes. Clearly the box cutting process can not be used directly to separate sample points from different paths. An alternative approach which guarantees separation of sample points is partitioning [1,22,30]. The essence of this tactic is simple: different sample points are drawn from different non-overlapping subsets of Ω, reducing the risk that two points are extremely close to each other.

Herein we show that similar sampling boxes in different passes have a strong tendency to diverge from one another. A full analysis of the box cutting strategy is extremely messy due to the myriad of different cases that arise from the relative positions of the control and sample points, and so is not attempted.

For simplicity, we look at one step of each of two passes, where both passes have the same control point c and same sampling box $\Omega_c = \{x \in \Omega : \ell \leq x \leq u\}$. Let the two sample points from the two passes be s and t. After cutting, let the new sample boxes be Ω_s and Ω_t respectively, and let s_{new} and t_{new} be the new sample points chosen randomly from Ω_s and Ω_t. A measure M_{overlap} of the separation between Ω_s and Ω_t is the probability that both s_{new} and t_{new} will lie in $\Omega_s \cap \Omega_t$. For ARS this probability is one.

For the original OSCARS algorithm [25] the situation is better. Without loss of generality, we assume that the dimensions have been ordered, and if necessary reversed, so that $c \leq s$, and the components of $s - c$ are in decreasing order. Then OSCARS forms Ω_s by imposing

an upper bound on Ω_c, viz.

$$\Omega_s = \{x \in \Omega_c : x_1 \leq c_1 + (1 - A)(s_1 - c_1)\}.$$

Let C, S and T be the vectors of edge lengths of the boxes Ω_c, Ω_s and Ω_t. Hence $C = u - \ell$. When Ω_t is formed by imposing an upper or lower bound on x_1, the measure of overlap is

$$M_{\text{overlap}} = \frac{\min(S_1, T_1)}{\max(S_1, T_1)} \quad \text{or} \quad M_{\text{overlap}} = \frac{(S_1 + T_1 - C_1)^2}{S_1 T_1}$$

respectively. The bound when only Ω_s is cut along x_1 can be found by setting $T_1 = C_1$ in either formula. If cuts are made orthogonal to more than one coordinate axis, the measure of overlap is simply the product of the measures for each dimension. Hence if a bound is imposed on x_i, $i \neq 1$ to form Ω_t, then $M_{\text{overlap}} = S_1 T_i (C_1 C_i)^{-1}$.

For OSCARS the risk of a large overlap is greatest when two passes impose the same type of bound (upper or lower) on the same variable. Since OSCARS can impose $2n$ different types of bounds when c is near the centre of Ω_c, eventually some passes must impose the same type of bound on the same variable. However, this situation is much improved over that of ARS.

For OSCARS- II a similar analysis yields

$$M_{\text{overlap}} = \left(\prod_{i \notin I_{\text{lower}}} \frac{\min(S_i, T_i)}{\max(S_i, T_i)} \right) \left(\prod_{i \in I_{\text{lower}}} \frac{(S_i + T_i - C_i)^2}{S_i T_i} \right)$$

where the index set I_{lower} contains the values of i for which a lower bound on x_i is imposed when forming Ω_t from Ω_c. Clearly OSCARS- II will usually have a lower overlap than OSCARS. The number of different combinations of bounds OSCARS- II can impose when c is near the centre of Ω_c is $3^n - 1$. If c is a corner of Ω_c, this falls to $2^n - 1$, which is still well above that for OSCARS except when n is very small.

This observation is useful in choosing β. If the largest component (the ith say) of $t - c$ has magnitude M, then the other components of $t - c$ are uniformly distributed across the interval $[-M, M]$, assuming for simplicity that the boundary of Ω_c is remote and can be neglected. A choice of $\beta = 1/3$ gives a $1/3$ probability that each coordinate direction (other than the ith) will have a new upper bound, a new lower bound, or no new bound. Similar reasoning when c is a corner point of Ω_c suggests $\beta = 1/2$.

3 Convergence

OSCARS- II uses only function values, and so can be executed even when the objective function is not smooth, or even discontinuous. We show OSCARS- II almost surely finds the essential global minimum, which is defined as follows.

Definition 1 The essential global minimum f^*_{egm} of f over Ω is defined as

$$f^*_{egm} = \sup \{\phi : \mu (\{x \in \Omega : f(x) < \phi\}) = 0\}$$

where $\mu(\cdot)$ denotes the Lebesgue measure in $\mathbb{R}^n$.

Points in the level set $\{x \in \Omega : f(x) \leq f^*_{egm}\}$ are referred to as essential global minimizers of f over Ω. The following assumption guarantees the sequence $\{b^{(k)}\}_{k=1}^{\infty}$ of best known points converges almost surely to one or more essential global minimizers.

Table 1 An overview of the performances of 5 methods on Test Set 1

Method	All 50 problems				Problems with $n \geq 4$			
	nf	Best on	F	Norm nf	nf	Best on	F	Norm nf
OSCARS	5897	5	4	3.05	8890	1	3	3.19
PSO- A	9225	7	60	5.65	15,161	3	49	5.93
PSO- I	7499	10	42	4.26	13,227	4	42	4.90
OSCARS- II(N)	4241	8	5	1.69	5312	7	–	1.32
OSCARS- II(P)	4158	20	2	1.71	5105	9	1	1.30

The Table's legend is given in the first paragraph of Sect. 4.2

Assumption 2 The objective function f is lower semi-continuous and bounded below on Ω.

Theorem 3 *Given Assumption 2, if* OSCARS- II *is run forever in exact arithmetic, the sequence* $\{b^{(k)}\}_{k=1}^{\infty}$ *of best known points converges to one or more essential global minimizer(s), almost surely.*

Proof The feasible region Ω is closed and bounded, and thus compact. The absence of stopping conditions means $\{b^{(k)}\}$ is an infinite sequence. Since $\{b^{(k)}\} \subset \Omega$, this sequence must have limit points. Let b_* be a point to which $\{b^{(k)}\}$ converges (possibly in subsequence) but is otherwise arbitrary.

The sequence of best function values $\{f(b_k)\}$ is monotonically decreasing and bounded below, so its limit f_∞ exists. If $f_\infty > f^*_{egm}$, then $\mu(\{x \in \Omega : f(x) < f_\infty\}) > 0$. Now OSCARS- II generates one random sample in Ω at the start of each cycle. Noting $k \leq \sum_{i=1}^{m} J(i)$ it follows that $k \to \infty$ implies $m \to \infty$. The probability P_m that a point with a lower function value than f_∞ is located within m cycles has the lower bound

$$P_m = P\left(f(b^{(k)}) < f_\infty \text{ after } m \text{ cycles}\right) \geq 1 - \left(1 - \frac{\mu(\{x \in \Omega : f(x) < f_\infty\})}{\mu(\Omega)}\right)^m.$$

Now $k \to \infty$ implies $m \to \infty$ which implies $P_m \to 1$ as $k \to \infty$. Hence $\lim_{k\to\infty} f(b^{(k)}) \leq f^*_{\text{egm}}$ almost surely. Finally, lower semi-continuity of f means

$$f(b_*) \leq \lim_{k\to\infty} f(b^{(k)}) \leq f^*_{\text{egm}}$$

and so b_* is an essential global minimizer of f over Ω almost surely, as required. □

In practice a weaker form of Assumption 2 is sufficent to prove this result: lower semi-continuity at the limits of $\{b^{(k)}\}$ is sufficient. Of course these limits are not known in advance.

4 Numerical results

The new method is compared with the original OSCARS algorithm [25], with a variant of DIRECT [14], and with two particle swarm methods [6,16]. Test Set 1 consists of the 50 problems which form the main test set for OSCARS [25]. These problems are listed in Table 1 of [25], and can be found in [2,20,23,25]. An extra 21 problems form Test Set 2, and are used to explore the algorithm's behaviour in moderate dimensions and above, along with the

Table 2 A comparison of four methods on Test Set 2, averaged over 30 runs. Columns marked 'F' list the number of runs which halted after reaching the function evaluation cap of 250,000

	Function	n	PSO- I		OSCARS- II(N)		OSCARS- II(P)		OSCARS	
			nf	F	nf	F	nf	F	nf	F
51	Chebyquad	9	8350	–	**2470**	–	3014	–	2671	–
52	Weka 2	9	**25,105**	–	42,187	–	28,069	–	165,162	16
53	Weka 2	10	**80,198**	7	120,538	5	129,754	10	239,417	27
54	Neumaier 3	10	81,397	–	24,945	–	24,414	–	**19,564**	–
55	Paviani	10	3648	–	2022	–	**1941**	–	2377	–
56	Mod. Beckers-Lago	10	**6988**	–	15,483	–	17,345	–	13,772	–
57	Mod. Langerman	10	241,408	28	110,691	4	**85,662**	2	101,829	2
58	Brown almost linear	10	52,251	3	33,919	–	**32,897**	–	66,161	3
59	Chained crescent II	12	**960**	–	35,235	–	28,536	–	180,762	15
60	Himmelblau ring	10	222,001	25	**153,031**	14	157,516	12	196,448	16
61	Dixon	10	231,060	10	21,296	–	**18,218**	–	114,741	–
62	Reflected Dixon	10	187,640	–	36,080	–	**30,960**	–	96,831	–
63	Modified Qing	20	**54,663**	–	95,230	2	91,441	2	195,130	16
64	Sinusoidal	20	101,970	9	28,078	–	**16,523**	–	216,099	22
65	Linked exponential	25	250,000	30	**151,381**	–	151,781	–	181,532	–
66	Extended Easom	30	75,053	–	**16,229**	–	16,715	–	47,157	–
67	Trigonometric	30	**6158**	–	36,966	–	82,098	–	64,068	–
68	Variably dimension	30	136,816	3	55,695	–	**54,298**	–	63,007	–
69	Zakharov	30	95,527	–	50,859	–	**49,990**	–	70,431	–
70	Mod. Bohachevsky 2	60	250,000	30	52,959	–	**52,725**	–	246,553	21
71	Schaffer 2	60	250,000	30	76,922	–	**79,799**	–	250,000	30

Here (P) and (N) identify results generated with and without the power law

Schoen family of test functions [29]. Test Set 2 is listed in Table 2, where problems 54, 57, 64, 66, and 71 are from [2]; problems 51, 55, 58, 67 and 68 are from [20]; problem 59 (with $\Omega = [-10, 10]^n$) is from [4]; problem 56 is from [25] and problem 69 is from [12].

The remaining problems, and problems modified for use herein are listed below along with any bounds not listed elsewhere. These problems all have $f^* = 0$. Weka 2 is

$$f_{\text{weka2}} = \sum_{i=1}^{n} x_i(1 - x_i) + \sum_{i=1}^{n} (k_i \lfloor p_i x_i \rfloor \mod p_i) \qquad x \in [0, 1]^n,$$

where the floor function $\lfloor s \rfloor$ denotes the greatest integer not larger than s. The parameter values used by Weka 2 in n dimensions are the first n entries of the lists $p_i = 2, 3, 5, 7, 11, 13, 17, 19, 23, 29$ and $k_i = 1, 1, 2, 2, 5, 5, 7, 11, 13, 17$.

The Himmelblau ring problem is defined via the two dimensional Himmelblau function

$$H(x_1, x_2) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

which has four global minimizers at approximately $(3, 2)$, $(3.5844, -1.8481)$, $(-2.8051, 3.1313)$ and $(-3.7793, -3.2832)$. The Himmelblau ring function for n even is

$$f_{60} = \sum_{i=1}^{n/2} H(x_{2i-1}, x_{2i}) + \sum_{i=2}^{n/2} (x_{2i-1} - x_{2i-3})^2 + (x_{2n-1} - x_1)^2 \qquad x \in [-5, 5]^n.$$

The first summation term alone has 2^n global minimizers and no local minimizers. The remaining terms are minimized when all odd numbered variables are identical. This means that the Himmelblau ring function has four global minimizers. There are another $2n$ local minimizers that are nearly global, and many other less deep local minimizers.

Dixon's function is

$$f_{61} = (1 - x_1)^2 + (1 - x_n)^2 + \sum_{i=2}^{n} \left(x_{i-1}^2 - x_i\right)^2 \qquad x \in [-2, 2]^n.$$

Dixon's function is separable, indeed many test functions are partially or totally separable. To test the effect of loss of separability, Dixon's function was reflected through the hyperplane orthogonal to $w = (1, 2, \ldots, n)$ which passes through the origin, with Ω held constant. This leaves the solution inside Ω but destroys any separability. Results for both forms of Dixon's function are presented.

The modified Qing problem is to minimize

$$f_{63} = \sum_{i=1}^{n} \left(x_i^2 - i\right)^2 \qquad \text{subject to} \qquad -1 \le x_i \le \sqrt{n} \qquad \forall i = 1, \ldots, n$$

where the bounds have been changed from $-50 \le x \le 50$ to $-1 \le x \le \sqrt{n}$. With the original bounds, f_{63} has 2^n global minima at $(\pm 1, \pm\sqrt{2}, \ldots, \pm\sqrt{n})$, and no local minima. Restricting the feasible region eliminates all but two of these global minimizers, and yields $2^n - 2$ local minimizers on the boundary of the feasible region. In spite of the feasible region being smaller, this makes the problem much harder.

A problem with similar characteristics to f_{63} is the linked exponential function, which is

$$f_{65} = x_1^2 e^{x_1} + x_n^2 e^{x_n} + \sum_{i=1}^{n-1} e^{-x_{i+1}} (x_{i+1} + x_i - 1)^2 \qquad x \in [0, 4]^n$$

where n must be odd. The modified Bohachevsky 2 problem with $x \in [-50, 50]^n$ is

$$f_{70} = \sum_{i=1}^{n} v_i x_i^2 + 0.3 \left(1 - \prod_{i=1}^{n} \cos\left(\pi x_i (2 + v_i)\right)\right) \qquad \text{where} \qquad v_i = 1 + \frac{i-1}{n-1} \quad \forall i.$$

4.1 The testing process

On all problems in all test sets except the Trigonometric function, the tolerance in f was set to $\tau = 10^{-3}$, and all runs halted immediately after locating a point with a function value less than $f^* + \tau$, where f^* is the known optimal function value for the problem. For the Trigonometric function (with $n = 5$, 10, and 30) $\tau = 10^{-5}$ was used in order to prevent termination at a local (but non-global) minimizer. Other parameter values were as follows: $A = 0.9$, $h_{min} = 10^{-6}$ and $\beta = 1/3$. The cycle length limit J was chosen as $J(m) = 90 + 30m$. This choice was motivated by a rough estimate of 62 iterations as the length of an unsuccessful pass when Ω is the unit hypercube. Hence the κ^{th} odd cycle is

typically expected to make at least $\kappa + 1$ improving steps before being halted by a cycle length reset. The power law fit test used $\tau_{\text{ks}} = 10^{-3}$ and $B = 100$.

The numerical experiments were performed in accordance with [5], with solution quality measured by a fixed target solve time: specifically the number of function evaluations needed to be within τ of the optimum. Reliability is assessed via the number of runs attaining an accuracy of τ before striking the function evaluation cap, and the efficiency is measured by the function evaluation count, which, in light of the low and constant per unit overheads, is a reasonable measure. The fixed target stopping condition is clearly only viable when the optimum is known in advance. If it is not, other stopping conditions must be used.

Data profiles are presented for the Schoen test functions. In each case computational effort is measured along the horizontal axis, and is simply the number of function evaluations N taken to reach the fixed target accuracy τ. The vertical axis plots the number of functions requiring at most N function evaluations to solve, on average where appropriate.

Comparisons are made with the original OSCARS algorithm, with a variant of DIRECT, and with two particle swarm methods. These last three are described now.

Two versions of the SPSO particle swarm were used, where SPSO is as described in [6] with $\max(5n, 50)$ particles and the global best topology. The parameter values recommended by [16] were used. The two methods differed in their reset strategies. The first (PSO- A) resets whenever all particles were within 10^{-5} of the global best known point, as measured by the infinity norm. This resetting retained the best known point, but placed all particles in random locations in Ω with zero initial velocities. The best known points for each particle were reset to their new random locations. Numerical testing showed that this resetting improved the performance of PSO, reducing the number of fails on Test Set 1 by about 25%.

The second particle swarm method (PSO- I) reset each particle individually whenever it was within a distance of 10^{-6} of the global best known point, and its velocity's magnitude was at most 10^{-6}, both measured using the infinity norm. Each reset particle was placed at a random point in Ω with zero initial velocity. Its best known position was reset to this new random location.

A slightly modified version of DIRECT is used herein and in [25] which we describe now. First, Ω is normalized to the unit hypercube. At each iteration DIRECT has a list of boxes aligned with the coordinate axes, where f is known at the centre point of each box. Initially the list contains one box: Ω. The size of each box is measured by the length of any main diagonal of the box, and the height of the box is equal to the value of f at the centre point of the box. At each iteration, each box which is either lower or larger than every other box is subdivided. Subdivision consists of cutting a box into three sub-boxes of identical size and shape. The two cuts are made using hyperplanes orthogonal to one coordinate axis along which the length of the uncut box is maximal. The centre point of the uncut box is also the centre of the middle sub-box, and so f does not have to be calculated there. The values f takes at the centres of the other two sub-boxes are calculated, which completes the subdivision. This process differs somewhat from the original algorithm in [14] in that all Pareto optimal (i.e. larger or lower) boxes are cut, whereas [14] subdivides only those on the efficient frontier. Numerical testing showed that there was little difference between the two in terms of function evaluations, but that the Pareto version took significantly less time.

OSCARS- II was tested extensively both with and without the power law fit test. In order to distinguish between them OSCARS- II(P) will denote results generated with the power law test implemented, and OSCARS- II(N) will mark results generated without the power law test.

4.2 Results and discussion for test sets 1 and 2

A comparison of both variants of OSCARS- II with OSCARS, and the two particle swarm methods on Test Set 1 appears in Table 1. The testing regime from [25] is used, which forms ten run averages of the number of function evaluations needed to get within τ of f^* for each problem, with the number of function evaluations capped at 50,000. If this cap is reached the run is classed as a fail and the number of function evaluations is set at the cap's value. The first column in Table 1 lists the method used to generate the results. Columns 2–5 list the average function evaluations taken across all problems, the number of problems on which each method used the fewest function evaluations, the number of failed runs, and average normalized function evaluation counts. Columns 6–9 list the same quantities as columns 2–5, but for the subset of problems in Test Set 1 with dimension at least four. Function evaluation counts for each problem are normalized by dividing by the smallest of the average function counts of each method for that problem. Using the cap value on failed runs biases the figures in columns 2, 5, 6, and 9 in favour of methods with more fails. These results show that the two variants of OSCARS- II are similar, with OSCARS and PSO- I coming in third and fourth places respectively.

Test Set 2 looks at four algorithms over a range of problems in nine or more dimensions. For this test set the cap on the number of function evaluations was increased to 250,000. Results on this test set using 30 run averages on each problem are listed in Table 2. Results for OSCARS and PSO- I are listed, along with those for the new method with and without the power law test. Bolded entries mark the best method on each problem.

The data in Table 2 shows that OSCARS- II outperforms OSCARS by a significant margin, on average. Noting the runs which strike the cap are set to the cap's value when evaluating the average function counts, the high number of times OSCARS struck the cap (168 from 630 runs) on Test Set 2 suggests the margin between the two methods could be significantly greater than the table suggests. The performance of PSO- I is similar to OSCARS, with 175 fails.

OSCARS- II(P) and OSCARS- II(N) used an average of 54,938 and 55,344 function evaluations per run respectively, where this average is across all 21 problems in Test Set 2. Considering these two methods in isolation, the average normalized function counts are 1.083 for OSCARS- II(P) and 1.107 for OSCARS- II(N) respectively.

In comparison to PSO- I, all methods did particularly poorly on problem 59, which is unimodal, but non-smooth. This problem requires a method to follow a curved 'V' shaped valley to the solution. The ability of particle swarm methods to take uphill steps is very useful in navigating such valleys. On the other non-smooth problems (52, 53, and 56) PSO- I did comparatively better than the other methods. This suggests non-smoothness causes more difficulty for an OSCARS type method than a particle swarm method. OSCARS- II largely has the approach of retaining a good control point until a better point is found. The only exception to this being at the start of odd iterations when the SORC strategy is applied. In contrast, particle swarm methods routinely move from a good point to a worse one, and this is a well known tactic for trying to deal with nonsmooth and ill conditioned valleys.

PSO- I also did very well on problem 67. This problem has a deep local minimizer on the opposite side of Ω to the global minimizer. If OSCARS- II focuses on the deep local minimizer, it might take several cycles to locate the global minimizer on the far side of Ω. In constrast, PSO- I's use of multiple particles seems to give it a better chance of spotting the global basin quickly. Problems 52 and 53 also have similar features, which partly explains OSCARS- II's comparatively poor performance on these problems.

Comparing PSO- I against OSCARS- II(N) alone, there are 10 problems where one method was at least 3 times as fast as the other. On eight of these 10 problems the faster method

Table 3 Results for OSCARS- II(N) on sets of 100 Schoen test functions of the form (2) with a function evaluation cap of 250,000

n	S = 15			S = 30			S = 60			S = 90		
	nf	SD	F	nf	SD	F	nf	SD	F	nf	SD	F
10	2245	2703	–	8122	24,024	–	16,847	47,874	3	32,990	70,957	7
20	4681	3701	–	14,106	35,333	2	23,174	45,765	2	41,063	69,290	7
30	12,886	33,797	1	15,896	15,875	–	38,509	58,122	5	57,909	71,254	10
40	20,407	30,122	1	22,277	24,280	–	50,567	65,601	6	92,049	87,724	15
50	19,254	17,664	–	56,784	63,881	3	70,484	72,878	7	77,945	81,740	12
60	31,876	40,801	1	61,391	68,674	4	91,886	81,926	12	118,105	97,357	27

Mean and standard deviation of number of function evaluations needed to get within 10^{-3} of the optimal function value are listed, along with the number of runs that reached the function evaluation cap (columns marked 'F')

was OSCARS- II(N), and the margin of OSCARS- II(N) over PSO- I is greater than it looks in Table 2. This is because OSCARS- II(N) did not strike the function evaluation cap of 250,000 at all on those problems, whereas PSO- I struck that cap 79 times. If the cap were removed, OSCARS- II(N)'s results would be unchanged, but those of PSO- I would be substantially worse.

Results for Test Set 2 were also generated for DIRECT. Problems 70 and 71 were omitted as their solutions are at the centre of Ω, which is the first point used by DIRECT. It solved eight of the remaining 19 problems, and was the fastest of all methods on two: Dixon (nf = 2005) and modified Qing (nf = 30,667). Both of these problems are separable. It also solved the reflected Dixon problem, taking 165,631 function evaluations. Reflected Dixon is non-separable. The disparity between DIRECT's results on the two Dixon problems suggests that separability is advantageous for DIRECT.

4.3 Results and discussion for the Schoen test set

Test functions from the Schoen class [29] used herein have the form

$$f(x) = \frac{\sum_{i=1}^{S} \theta_i \prod_{j \neq i} \|x - z_j\|^2}{\sum_{i=1}^{S} \prod_{j \neq i} \|x - z_j\|^2} \tag{2}$$

where the indices in each product run from $j = 1$ to S, excluding $j = i$. Each $z_i \in \Omega$ is a stationary point of f, and is chosen randomly from Ω. The values θ_i give the heights of these stationary points, and are chosen randomly from a unit normal distribution, with one exception. After choosing all θ_i in this way, the lowest (θ^* say) is reduced by 10^{-3}, ensuring that any point with a function value less than $\theta^* + 10^{-3}$ is in the vicinity of the global minimizer. Let the point z^* be the global minimizer of f. In general, f will have many other local minimizers, some of which will be other $z_i \neq z^*$, and others will lie on the boundary of Ω.

The initial tests of OSCARS- II on the Schoen test functions did not use the power law fit test. These initial tests used $10 \leq n \leq 60$ in steps of 10, and various numbers S of interpolation points; specifically $S \in \{15, 30, 60, 90\}$. For each combination of S and n, 100 different Schoen functions were generated and minimized once each by OSCARS- II(N). Each run was capped at 250,000 function evaluations. Runs which struck that cap were costed at the cap value, and recorded as fails. These results are listed in Table 3 and show that

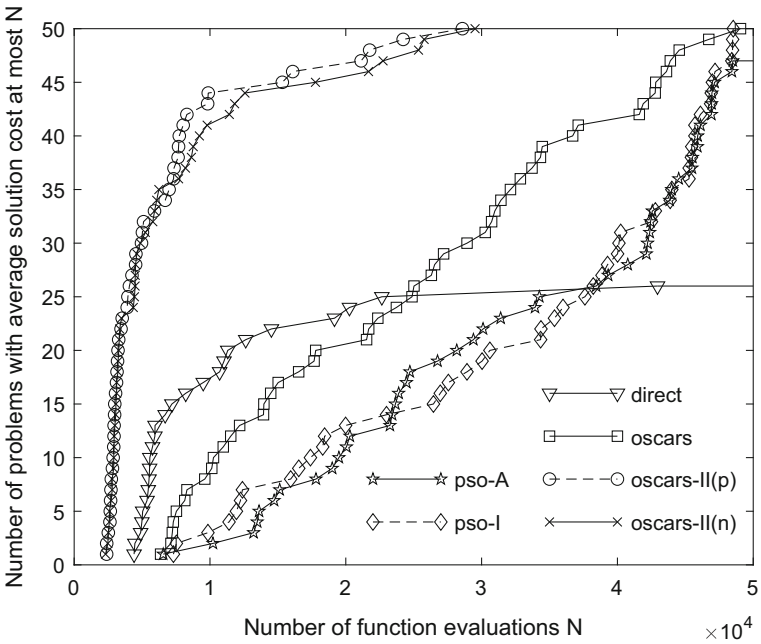

Fig. 1 Data profile for six methods on 50 randomly generated 15 dimensional Schoen test functions with the number of interpolation points varying randomly from 15 to 50. Each test problem was solved 30 times by each stochastic method, and once by DIRECT. The graph plots the number of problems (vertical axis) with an average solution cost not more than the function count given on the horizontal axis

OSCARS- II copes well with increasing dimension, provided the structure of f remains benign. Performance worsens with both increasing dimension, and increasing complexity in roughly equal measure, where the complexity here is measured by the number of interpolation points S. In almost all cases the standard deviation of the function evaluation cost exceeded its mean. This is because, on most runs, OSCARS- II solves the problem using far fewer function evaluations than the mean. Rarely, it takes many more function evaluations than the mean, yielding a high standard deviation.

This "usually quick, occasionally very slow" behaviour was also evident on problems in Test Sets 1 and 2. After some investigation it was found that the even numbered cycles were usually short and ended with a box size reset, whereas the odd numbered cycles were much longer. These odd cycles appeared to make fairly regular tiny reductions in f, with f significantly higher than the best known value. These tiny improvements yielded improving resets, allowing the cycle to continue until ended by a cycle length reset. The power law reset was included in an attempt to limit such behaviour.

The new method was compared against other algorithms using data profiles [5]. The first profile used 50 Schoen problems in 15 dimensions, where the number of interpolation points S for each problem was a randomly selected integer in the range $15 \leq S \leq 50$. Each test problem was solved 30 times by each of OSCARS, PSO- A, PSO- I, OSCARS- II with and without the power law fit, and once by the deterministic method DIRECT. All runs are capped at 50,000 function evaluations. The number of function evaluations N is the horizontal axis variable in Fig. 1, and the number of problems which, on average, take at most N function evaluations, is the vertical axis variable. The results show OSCARS- II outperforms the other methods.

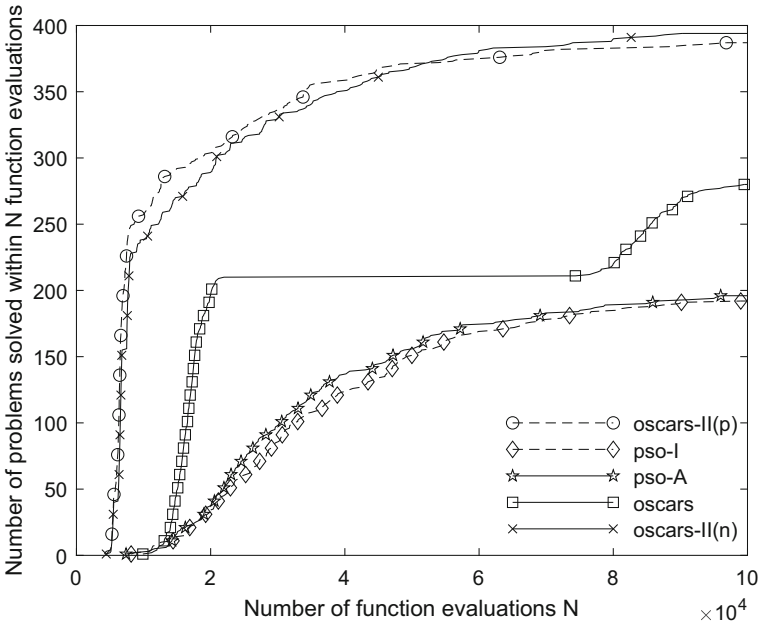

Fig. 2 Data profile for five methods on 400 randomly generated 30 dimensional Schoen test functions with the number of interpolation points varying randomly from 15 to 50. Each test problem was solved once by each method. The graph plots the number of problems (vertical axis) which were solved using at most the number of function evaluations given on the horizontal axis

DIRECT is initially a good performer, but lags as the number of required function evaluations increases. Both particle swarm methods are similar, with OSCARS having an edge.

The second profile appears in Fig. 2. Here 400 Schoen functions were generated with $n = 30$ and S was a randomly selected integer in [15, 50]. Each test problem was solved once by each method, with a maximum function evaluation cap of 100,000 function evaluations. Figure 2 plots the number of problems solved after a given number of function evaluations against the number of function evaluations. Once again, both variants of OSCARS- II are superior, and very similar to one another. Given that the number S of interpolation points is quite low, one would expect the particle swarm methods to perform well. However the results show they are outdone by the other methods.

The data profile for OSCARS in Fig. 2 has three distinct regions. The left most is a steep rise due to problems solved within the first two cycles. For the remaining problems these cycles continued until terminated by minimum box size resets. These cycles typically contain many improving resets which make negligible improving changes to the control point, and yield the central flat region of OSCARS' profile. The right hand rise appears to be due to problems solved in the third and subsequent cycles. All odd numbered cycles have random initial control points. Hence at the start of each odd cycle OSCARS' focus shifts to a different part of Ω, enabling it to solve additional problems. The flat region in OSCARS' profile is exactly what the extra cycle length resets in OSCARS- II are designed to prevent.

The final profile is given in Fig. 3, and compares OSCARS- II with and without the power law test in 60 dimensions with 120 interpolation points. In low dimensions the improvement from the power law test is most significant (see Table 1), and reduces somewhat with increasing dimension. Collectively, all results show the power law test makes a modest improvement,

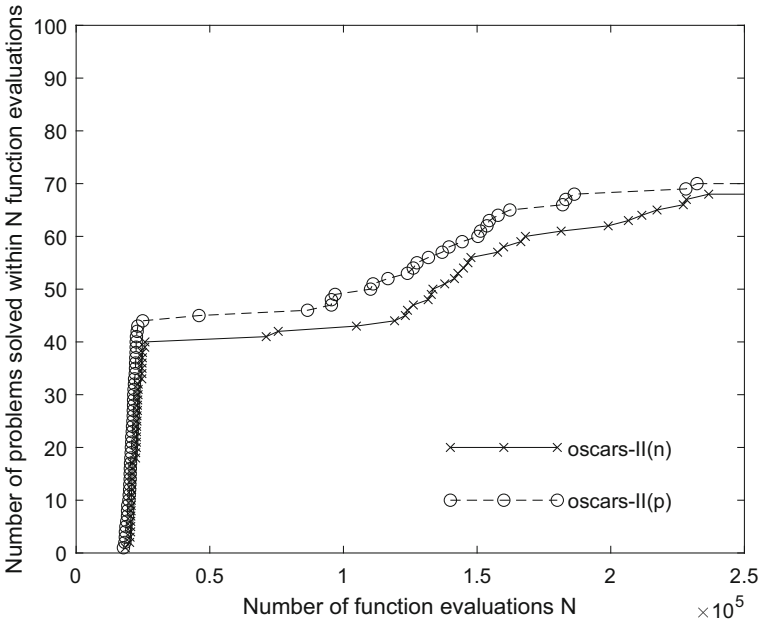

Fig. 3 Data profile for two methods on 100 randomly generated 60 dimensional Schoen test functions with 120 interpolation points. Each test problem was solved once by each method. The graph plots the number of problems (vertical axis) which were solved using at most the number of function evaluations given on the horizontal axis

at the expense of an increase in overheads, and a more complex algorithm. This is due to one simple fact. If a single cycle is focused on a proper local minimizer for any significant length of time, it must be making occasional improving resets to avoid a cycle ending box size reset. The iterations following an improving reset explore Ω widely. The alternative of ending the cycle through a power law reset also does this, but to a greater degree. Clearly the former is able to compensate partially for the latter over the time scale governed by the cycle length reset $J(m)$. The supposed independence of the algorithm's behaviour with regard to the control point's location can be tested more vigourously by using the best known point as the initial control point for all cycles, not just the even numbered ones. This leads to a marked deterioration in performance (far more so than for OSCARS [25]).

These data profiles and the results in Tables 2 and 3 show that OSCARS- II is effective on problems of moderate ($10 \leq n \leq 20$) dimension and up to the bottom of high dimensional ($n \geq 50$) problems [21], provided the objective function is reasonably benign. As shown in [21], providing guarantees for random search methods on less benign functions in high dimensions is very hard.

5 Conclusion

A variation on the OSCARS algorithm for global optimization has been presented. The principal changes to OSCARS are that each sampling box can be shrunk simultaneously along several dimensions, and that the lengths of each cycle are initially kept short, and only slowly allowed to grow. A power law test for abandoning proper local minimizers was also tested. This test

can be used with any algorithm in the accelerated random search class, and possibly with many other methods.

The new method was compared to DIRECT and two particle swarm methods and was found to be superior on average. Numerical results also show that the new method outperforms OSCARS above three dimensions. The new algorithm has constant overheads per function evaluation. Additional analysis shows that both OSCARS and the new method have point separation properties similar to partition methods, in contrast to ARS. This separation property enables OSCARS- II to outperform OSCARS above three dimensions in spite of a box shrinkage rate similar to ARS rather than OSCARS. OSCARS- II was compared with two particle swarm algorithms on Schoen test functions, and demonstrated a significant capability on reasonably benign problems of moderately high dimension.

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

References

1. Al Dujaili, A., Suresh, S., Sundararajan, N.: MSO: a framework for bound constrained black-box global optimization. J. Glob. Optim. **66**(4), 811–845 (2016)
2. Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization problems. J. Glob. Optim. **31**, 635–672 (2005)
3. Appel, M.J., Labarre, R., Radulović, D.: On accelerated random search. SIAM J. Opt. **14**, 708–731 (2003)
4. Bagirov, A.M., Ugon, J.: Piecewise partially separable functions and a derivative-free algorithm for large scale nonsmooth optimization. J. Glob. Optim. **35**, 163–195 (2006)
5. Beiranvand, V., Hare, W., Lucet, Y.: Best practices for comparing optimization algorithms. Optim. Eng. **18**, 815–848 (2017)
6. Bonyadi, M.R., Michalewicz, Z.: Particle swarm optimization for single objective continuous space problems: a review. Evolut. Comput. **25**, 1–54 (2017)
7. Calvin, J., Gimbutienė, G., Phillips, W.O., Žilinskas, A.: On the convergence rate of a rectangular, partition based global optimization algortihm. J. Glob. Optim. **71**, 165–191 (2018)
8. Csendes, T., Pál, L., Sendín, J.O.H., Banga, J.R.: The GLOBAL optimization method revisited. Optim. Lett. **2**, 445–454 (2008)
9. Dorea, C.C.Y.: Stopping rules for a random optimization method. SIAM J. Control Optim. **28**, 841–850 (1990)
10. Floudas, C.A., Gounanis, C.E.: A review of recent advances in global optimization. J. Glob. Optim. **45**, 3–38 (2009)
11. Hart, W.E.: Sequential stopping rules for random optimization methods with applications to multistart local search. SIAM J. Optim. **9**, 270–290 (1999)
12. Hirsch, M.J., Pardalos, P.M., Resende, M.G.C.: Speeding up continuous GRASP. Eur. J. Oper. Res. **205**, 507–521 (2010)
13. Huang, H., Zabinsky, Z.B.: Adaptive probabilistic branch and bound with confidence intervals for level set approximation. In: Pasupathy, R., Kim, S.H., Tolk, A., Hill, R., Kuhl, M.E. (eds.) Proceedings 2013 Winter Simulation Conference, pp. 980–991. IEEE, Washington DC (2013)
14. Jones, D., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**, 157–181 (1993)
15. Kawaguchi, K., Marayama, Y., Zheng, X.: Global continuous optimization with error bound and fast convergence. J. Artif. Intell. Res. **56**, 153–195 (2016)
16. Liu, Q.: Order-2 stability analysis of particle swarm optimization. Evolut. Comput. **23**, 187–216 (2014)
17. Liu, Q., Zeng, J.: Global optimization by multilevel partition. J. Glob. Optim. **61**, 47–69 (2015)
18. Liuzzi, G., Lucidi, S., Piccialli, V.: A partition based global optimization algorithm. J. Glob. Optim. **48**, 113–128 (2010)
19. Locatelli, M., Schoen, F.: Global Optimization. MOS-SIAM Series on Optimization 15. SIAM, Philadelphia (2013)
20. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**, 17–41 (1981)

21. Pepelyshev, A., Zhigljavsky, A., Žilinskas, A.: Performance of global random search algorithms for large dimensions. J. Glob. Optim. **71**, 57–71 (2018)
22. Pinter, J.: Convergence qualification of adaptive partition algorithms in global optimization. Math. Program. **56**, 343–360 (1992)
23. Price, C.J., Reale, M., Robertson, B.L.: A cover partitioning method for bound constrained global optimization. Optim. Methods Softw. **27**, 1059–1072 (2012)
24. Price, C.J., Reale, M., Robertson, B.L.: A CARTopt method for bound-constrained global optimization. ANZIAM J. **55**(2), 109–128 (2013)
25. Price, C.J., Reale, M., Robertson, B.L.: One side cut accelerated random search. Optim. Lett. **8**(3), 1137–1148 (2014)
26. Price, C.J., Reale, M., Robertson, B.L.: Stochastic filter methods for generally constrained global optimization. J. Glob. Optim. **65**, 441–456 (2016)
27. Radulović, D.: Pure random search with exponential rate of convergency. Optimization **59**, 289–303 (2010)
28. Regis, R.D., Shoemaker, C.A.: Constrained global optimization of expensive black box functions using radial basis functions. J. Glob. Optim. **31**, 153–171 (2005)
29. Schoen, F.: A wide class of test functions for global optimization. J. Glob. Optim. **3**, 133–137 (1993)
30. Tang, Z.B.: Adaptive partitioned random search to global optimization. IEEE Trans. Auto. Control **11**, 2235–2244 (1994)
31. Torn, A., Žilinskas, A.: Global Optimization. Lecture Notes in Computer Science, vol. 350. Springer, Berlin (1989)
32. Zhigljavsky, A., Žilinskas, A.: Stochastic Global Optimization. Springer, New York (2008)