Optim Lett
DOI 10.1007/s11590-013-0631-8

ORIGINAL PAPER

# One side cut accelerated random search

## A direct search method for bound constrained global optimization

**C. J. Price · M. Reale · B. L. Robertson**

Received: 18 September 2012 / Accepted: 2 March 2013

**Abstract** A new algorithm for bound constrained global optimization is presented. At each iteration the method uses a single feasible box containing a control point. A single random sample point is generated inside the box. If this new point is better it replaces the control point and the box is reset to the feasible region. Otherwise the box is shrunk so that it no longer contains the random sample point. If a minimal box size is reached the box is also reset to the feasible region. The method is shown to converge almost surely to an essential global minimizer. The method is very simple to implement. Numerical testing shows that the method is viable in practice. A simple modification to accelerated random search is also discussed.

**Keywords** Direct search · Bound constrained global optimization · OSCARS

## 1 Introduction

The bound constrained global optimization problem is

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

where $\Omega \subset \mathbb{R}^n$ is a finite box of the form

C. J. Price (✉) · M. Reale · B. L. Robertson
Mathematics and Statistics, University of Canterbury,
Private Bag 4800, Christchurch, New Zealand
e-mail: C.Price@math.canterbury.ac.nz

M. Reale
e-mail: marco.reale@canterbury.ac.nz

B. L. Robertson
e-mail: blair.robertson@canterbury.ac.nz

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

Here $x_i$ is the $i$th element of $x$, and all upper and lower bounds $U_i$ and $L_i$ are finite. Minimum requirements on the objective function $f$ are that it be Lebesgue measurable, bounded below, and lower semi-continuous at points of interest.

The simplest stochastic method for solving (1) is Pure Random Search (PRS), which calculates $f$ at $N$ random points in $\Omega$ and returns the lowest as an approximate global minimizer. Numerous improvements on PRS have been proposed. One strategy is to partition $\Omega$ into sub-regions and calculate $f$ at random sample points in each sub-region. Early sample points give information which can be used by sampling more often in sub-regions in which the lower initial sample points lie [9,15]. Alternatively the probability distribution used to generate the samples can be altered to incorporate information from earlier sample points [10].

Many stochastic and deterministic methods subdivide the partitions from time to time. The deterministic algorithm DIRECT [7] is an extreme example of this approach. It uses one sample point in the centre of each box shaped sub-region. DIRECT iteratively selects promising boxes and divides each into three congruent boxes with two cuts orthogonal to a coordinate axis. The centre of each original box is also the centre of one of its three smaller boxes and thus can be 're-used.' The function values at all remaining boxes' centres are calculated, and DIRECT proceeds to the next iteration. A related stochastic algorithm is TILECUTTER [11] which has one random sample point per box (or 'tile'). At each iteration TILECUTTER picks a number of tiles, and cuts each into two tiles using a hyperplane orthogonal to some coordinate axis. One sample point is placed in each empty tile. Herein we reverse this strategy by placing a second random point in a tile, and then cutting between the two points.

An alternative to partitioning is multistart [16]. The simplest version of this generates $N$ sample points randomly in $\Omega$ and performs $N$ local searches, using each sample point as one local search's initial point. Efficiency can be improved by grouping the sample points into clusters [2,13,14] with the intent that all local searches starting from points in the same cluster will yield the same local minimizer. Hence only one local search per cluster need be done. Only the lowest quantile (say the lowest 20 %) of sample points are clustered. This helps separate clusters, and avoids wasting time processing points which are clearly far from optimal. One local search per cluster is performed, starting from the lowest point in each cluster. In [13,14] a Newton method is used for the local search and the optimal Hessian used in clustering. In contrast [4] use a derivative free local search.

Recursive Random Search (RRS) [17] computationally simplifies this approach by generating a number of random sample points in $\Omega$, and using the lowest to define a level. It continues with PRS until a point below this level is found. A derivative free stochastic local search is then executed. This local search applies PRS to sub-regions of $\Omega$ placed around the current point until progress becomes insignificant. The algorithm reverts to PRS on $\Omega$ until another point below the defined level is found. This approach avoids retaining and processing all previous sample points, but risks doing local searches from initial points which are very close together.

Accelerated Random Search (ARS) is introduced in [3]. It samples randomly once within each member of a finite sequence of nested boxes. The first box in the sequence is $\Omega$, and each subsequent box is the intersection of $\Omega$ with a hyperrectangle centred on the current best point. The $m$th of these hyperrectangles is a copy of $\Omega$ centred on the current best point, and scaled by a factor $(1/\sqrt{2})^m$. If the new sample is better than the best known point, or if the finite sequence of boxes is exhausted, then the algorithm reverts to the start of the sequence of boxes. A variation of ARS with an exponential rate of convergence is described in [12].

These methods differ in a number of ways including storage requirements, overheads per function evaluation, and how these change with increasing number of sample points and problem dimension. For example DIRECT and TILECUTTER retain and process all points, and hence face increasing computational overheads per sample point as the number of sample points increases. The clustering algorithms in [2,4,13,14] avoid this at the expense of a process that is exponential in the dimension $n$ of $\Omega$. In contrast, ARS and RRS are essentially memoryless.

Herein we present a variation on ARS called one side cut ARS (or OSCARS). The algorithm uses only a single tile, and one point $c$ inside that tile called the control point. OSCARS generates a second point in the tile, and, if worse than $c$, cuts between these two points. The part containing $c$ becomes the new tile. The cut is orthogonal to some coordinate axis to ensure all tiles are hyperrectangles, permitting efficient random sampling of each tile. This 'sample then cut' approach reduces the tile only in the direction of a known worse point, and implicitly partitions $\Omega$ about $c$.

## 2 The OSCARS algorithm

The algorithm consists of two nested loops. The inner loop generates random sample points in a nested sequence of successively smaller tiles, each of which contains a control point $c$. At each iteration a sample point is generated randomly in the tile. If this new point is better than $c$, it replaces $c$ and the tile is reset to $\Omega$, exactly as in ARS [3]. Otherwise the tile is cut between the new point and $c$. The cut is orthogonal to the coordinate axis along which the difference between the two points is greatest. The part of the tile containing $c$ becomes the tile for the inner loop's next iteration.

This continues until a minimum tile size $h_{\min}$ is reached, which completes one iteration of the outer loop. The search then returns to $\Omega$, but the control point $c$ is selected differently. If the number $m$ of completed outer iterations is even, $c$ is chosen randomly from $\Omega$, otherwise $c$ is set equal to the best known point $b$. This means odd outer iterations start with a random control point. We call this approach SORC for 'Starting Odd iterations with a Random Control point.' The logic behind SORC is simple. If $c$ is always picked randomly, no information is passed on to subsequent iterations. On the other hand if the control point is always set to the best known point, the search becomes strongly focused around this point, and the method may become prone to being trapped by a local minimizer. A precise statement of OSCARS is given in Fig. 1, where the inner loop is step 2 and the outer loop steps 2–5.

The bounds $\ell$ and $u$ define the current tile $\Omega_k$ via

$$\Omega_k = \{x \in \mathbb{R}^n : \ell^{(k)} \le x \le u^{(k)}\}$$

```
1. Choose a random point $x^{(0)} \in \Omega$. Calculate $f_c = f_b = f(x^{(0)})$. Set $k = m = 1$, set
   $b = c = x^{(0)}$, and set $u = U$ and $\ell = L$. Choose $A \in (0,1)$. Select $h_{\min} > 0$.
2. While $\|u - \ell\|_\infty > h_{\min}$ and stopping conditions are not satisfied do
   (a) Choose a random point $x^{(k)}$ satisfying $\ell \le x^{(k)} \le u$.
   (b) If $f(x^{(k)}) < f_c$ then set $c = x^{(k)}$, set $f_c = f(x^{(k)})$, set $u = U$, set $\ell = L$, and go
       to step 2(d).
   (c) Let $i$ maximize $|x_i^{(k)} - c_i|$.
       If $x_i^{(k)} < c_i$, set $\ell_i = (1-A)x_i^{(k)} + Ac_i$, otherwise set $u_i = (1-A)x_i^{(k)} + Ac_i$.
   (d) Increment $k$.
   end of while.
3. Set $u = U$ and $\ell = L$. If $f_c < f_b$ set $b = c$ and $f_b = f_c$.
4. If $m$ is even, randomly choose $c \in \Omega$ and calculate $f_c$; otherwise set $c = b$ and set
   $f_c = f_b$.
5. Increment $m$. If stopping conditions are not satisfied, go to step 2.
```

**Fig. 1** The one side cut accelerated random search algorithm

where $\ell^{(k)}$ and $u^{(k)}$ are the values at step 2(a) at iteration $k$. In step 2(c) the coordinate direction along which $x^{(k)} - c^{(k)}$ has its longest component is identified. The tile is then cut with a hyperplane orthogonal to this coordinate, where the hyperplane passes between $x^{(k)}$ and $c^{(k)}$ at the point $Ac^{(k)} + (1 - A)x^{(k)}$. The next tile $\Omega_{k+1}$ is the part of $\Omega_k$ containing $c^{(k)}$. Here $A$ gives the fraction of the cut's displacement from $x^{(k)}$ to $c$.

The algorithm implicitly partitions $\mathbb{R}^n$ into $2n$ congruent cones emanating from the current control point $c^{(k)}$. These cones are

$$\mathscr{K}(c^{(k)}, w) = \{x \in \mathbb{R}^n : w^T(x - c^{(k)}) = \|x - c^{(k)}\|_\infty\}, \quad \forall w \in \mathscr{W},$$

where $\mathscr{W} = \{\pm e_1, \ldots, \pm e_n\}$, and $e_i$ is the $i$th unit coordinate vector. The intersections of these cones with $\Omega$ then partition $\Omega$ into $2n$ truncated cones. Each such cone either contains a nearby sample point $x$ satisfying $f(x) \ge f(c^{(k)})$, or it contains nearby infeasible points. Here 'nearby' means either a specific point $x$ satisfying $Ac^{(k)} + (1 - A)x \in \Omega_k$, or a sequence of points converging to $c^{(k)}$.

To see why this is so, let $\mathscr{F}$ be a face of $\Omega_k$ where the outward normal to $\mathscr{F}$ for $\Omega_k$ is $w \in \mathscr{W}$. The face $\mathscr{F}$ either has been created by a cut between a sample point $x$ and $c^{(k)}$, or $\mathscr{F}$ is a subset of the corresponding face of $\Omega$. In the former case $x$ must lie in the cone $\mathscr{K}(c^{(k)}, w)$ otherwise the cut would not have been orthogonal to $w$. Additionally $x$ must also satisfy $f(x) \ge f(c^{(k)})$, otherwise $x$ replaces $c^{(k)}$ as the control point. The position the cut is made in step 2(c) of the algorithm means $Ac^{(k)} + (1 - A)x \in \mathscr{F}$ and so $x$ is a nearby point.

When $\mathscr{F}$ is part of the boundary of $\Omega$ and $c^{(k)} \notin \mathscr{F}$, it is clear that any $x$ satisfying $Ac^{(k)} + (1 - A)x \in \mathscr{F}$ must lie outside $\Omega$. If $c^{(k)} \in \mathscr{F}$ then every point on the ray $c^{(k)} + \alpha w$, $\alpha > 0$, is infeasible. Either way there are nearby infeasible points in the cone $\mathscr{K}(c^{(k)}, w)$.

## 3 Convergence

OSCARS is applicable to non-smooth and many discontinuous functions. Hence we define the essential global minimum $f^*_{\text{egm}}$ of $f$ over $\Omega$ as the greatest value of $f^*$ for which the open level set $\{x \in \Omega : f(x) < f^*\}$ has Lebesgue measure zero. Any point $x^*$ with a function value $f(x^*)$ satisfying $f(x^*) \leq f^*_{\text{egm}}$ is referred to as an essential global minimizer. We show the sequence of best known points $\{b_k\}_{k=1}^{\infty}$ converges to one or more essential global minimizers. This requires the following assumption.

**Assumption 1** The function $f$ is lower semi-continuous and bounded below on $\Omega$.

Next we show OSCARS performs an infinite number of restarts with probability 1.

**Theorem 2** *The sequence of iterates $\{x^{(k)}\}_{k=1}^{\infty}$ contains an infinite number of sample points drawn randomly from $\Omega$ with probability one.*

*Proof* Each iteration of the while loop that ends with resetting $\Omega_{k+1}$ to $\Omega$ results in a random sample being drawn from $\Omega$ in the following iteration. This resetting happens infinitely often unless both $\|\Omega_k\|_\infty > h_{\min}$ and $f(x^{(k)}) \geq f_c$ for all $k$ sufficiently large. We show that the probability that both of these conditions hold for all $k$ sufficiently large is zero.

Assume that $\Omega_k$ is reset to $\Omega$ only a finite number of times, the last of which occurs at iteration $K$. That means the current point $c^{(k)}$ must be constant for all $k \geq K$. For each $w \in \mathscr{W}$ define the reach $\rho_k(w)$ of $\Omega_k$ in the direction $w$ as

$$\rho_k(w) = \max_{x \in \Omega_k} w^T\left(x - c^{(k)}\right).$$

For fixed $w$ each sequence $\{\rho_k(w)\}_{k=K}^{\infty}$ is a decreasing sequence bounded below by zero. Hence we can define

$$\rho_\infty(w) = \lim_{k \to \infty} \rho_k(w) \quad \forall w \in \mathscr{W}$$

and $\rho_{\max} = \max\{\rho_\infty(w) : w \in \mathscr{W}\}$. Noting that $c^{(k)} \in \Omega_k$ for all $k$, if $\|\Omega_k\|_\infty > h_{\min}$ for all $k \geq K$, it follows that $\rho_{\max} \geq h_{\min}/2$. Define the set $\mathscr{M} \subseteq \mathscr{W}$ of outward normals $w \in \mathscr{W}$ such that $\rho_\infty(w) = \rho_{\max}$. Choose $J \geq K$ such that for each $w \in \mathscr{W}$ either

$$\rho_k(w) < \rho_{\max} \quad \text{or both} \quad \rho_k(w) < \frac{\rho_{\max}}{(1-A)} \quad \text{and} \quad w \in \mathscr{M}.$$

For each $k \geq J$ choose $w_k \in \mathscr{M}$ such that $\rho_k(w_k) \geq \rho_k(w)$ for all $w \in \mathscr{W}$. Let $\mathscr{F}_k$ be the face of $\Omega_k$ for which $w_k$ is the outwards normal. Define the pyramid $\mathscr{R}_k$ with apex $c^{(k)}$ and base $\mathscr{F}_k$ via

$$\mathscr{R}_k = \{c^{(k)} + \alpha(x - c^{(k)}) : x \in \mathscr{F}_k \quad \text{and} \quad \alpha \in [0,1]\} \subset \mathscr{K}(c^{(k)}, w) \tag{2}$$

for all $k > J$. The fact that $\mathscr{F}_k$ is a face furthest away from $c^{(k)}$ means the largest of the components $w^T(x - c^{(k)})$, $w \in \mathscr{W}$, is achieved when $w = w_k$. Hence $\mathscr{F}_k$ is contained in $\mathscr{K}(c^{(k)}, w_k)$. The form of all other points in $\mathscr{R}_k$ means $\mathscr{R}_k$ is a subset of $\mathscr{K}(c^{(k)}, w_k)$.

Now $\Omega_k$ can be viewed as a rectangular prism with base $\mathscr{F}_k$ and height equal to the length of its edge parallel to $w_k$. The pyramid $\mathscr{R}_k$ has the same base $\mathscr{F}_k$ and height at least half that of $\Omega_k$ because $\mathscr{F}_k$ is a face furthest away from $c^{(k)}$. Thus the ratio $m(\mathscr{R}_k)/m(\Omega_k)$ is at least $1/(2n)$, where $m(.)$ is the Lebesgue measure. Hence the probability that $x^{(k)} \in \mathscr{R}_k$ is at least $1/(2n)$ for all $k \geq J$. If $x^{(k)}$ is placed in $\mathscr{R}_k$, it must satisfy $f(x^{(k)}) \geq f(c^{(k)})$ (otherwise $\Omega_k$ is reset to $\Omega$) and this results in a cut orthogonal to $w_k$ at a distance not more than $(1 - A)\rho_k(w_k)$ from $c^{(k)}$. Hence $\rho_{k+1}(w_k) \leq (1 - A)\rho_k(w_k) < \rho_{\max}$, contradicting the fact that $\{\rho_j(w_k)\}_{j=J}^{\infty}$ is a decreasing sequence converging to $\rho_{\max}$. Hence if $\Omega_k$ is reset to $\Omega$ only a finite number of times, then $x^{(k)} \notin \mathscr{R}_k$ for all $k \geq J$, an event which occurs with probability zero. □

**Theorem 3** *The sequence of best known points $\{b_k\}$ converges to one or more essential global minimizer(s), with probability 1.*

*Proof* The sequence of best function values $\{f(b_k)\}$ is monotonically decreasing and bounded below, so its limit exists. The fact that $\Omega$ is sampled randomly infinitely often almost surely, means that the sequence of best function values $\{f(b_k)\}$ converges to a value not more than the essential global minimum $f^*_{\text{egm}}$, almost surely.

The set $\Omega$ is compact, hence $\{b_k\}$ has cluster points. Let one such cluster point be $b_*$. 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 $\Omega$, as required. □

## 4 Numerical results

Tests were performed on 50 problems, as listed in Table 1. Problems 1–25 are from [1], 26–37 from [8], and 38–42 from [11]. Problems 43–50 are as follows. The Beckers–Lago problem was modified to $f_{43} = \sum_{i=1}^{n}((|x_i| - 5)^2 + 5|x_i - 5|)$, so that not all minimizers are global. The Rastrigin and offset Rastrigin functions are respectively

$$f_{44} = \sum_{i=1}^{n}\left(x_i^2 - \cos(18x_i)\right) \quad \text{and}$$

$$f_{45}(x) = \sum_{i=1}^{n}\left(\left(x_i - \frac{1}{i}\right)^2 - \cos\left(18\left(x_i - \frac{1}{i}\right)\right)\right)$$

**Table 1** A comparison of OSCARS, DIRECT, and ARS on test functions from [1,8,11]

| | Function | $n$ | $L$ | $U$ | DIRECT | ARS | | OSCARS | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | $nf$ | fails | $nf$ | fails |
| 1 | Ackley | 2 | −30 | 30 | – | 4,074 | – | **2,211** | – |
| 2 | Bohachevsky 1 | 2 | −50 | 50 | – | 615 | – | **237** | – |
| 3 | Bohachevsky 2 | 2 | −50 | 50 | – | 481 | – | **250** | – |
| 4 | Branin | 2 | – | – | 107 | 139 | – | **105** | – |
| 5 | Camel 6 | 2 | −5 | 5 | 159 | 120 | – | **102** | – |
| 6 | Goldstein–Price | 2 | −2 | 2 | **123** | 225 | – | 151 | – |
| 7 | Hartmann 3 | 3 | 0 | 1 | **187** | 963 | – | 252 | – |
| 8 | Hartmann 6 | 6 | 0 | 1 | **671** | 25,207 | 5 | 22,671 | 1 |
| 9 | Dekkers–Aarts | 2 | −20 | 20 | 413 | **250** | – | 685 | – |
| 10 | Levy–Montalvo 1 | 3 | −10 | 10 | **293** | 639 | – | 907 | – |
| 11 | Levy–Montalvo 2 | 3 | −5 | 5 | **305** | 490 | – | 607 | – |
| 12 | Levy–Montalvo 2 | 6 | −5 | 5 | 2,057 | 8,972 | – | **1,133** | – |
| 13 | mod. Rosenbrock | 2 | −5 | 5 | **407** | 19,213 | 3 | 1,253 | – |
| 14 | Neumaier 2 | 4 | 0 | 4 | fail | **14,439** | – | 16,152 | – |
| 15 | Paviani | 10 | 2 | 10 | 2,807 | **1,540** | – | 2,379 | – |
| 16 | Periodic price | 2 | −10 | 10 | – | **9,506** | – | 9,723 | – |
| 17 | Salomon | 2 | −100 | 100 | – | **3,991** | – | 10,107 | – |
| 18 | Schaffer 1 | 2 | −100 | 100 | – | 18,980 | – | **16,867** | – |
| 19 | Schaffer 2 | 2 | −100 | 100 | – | 562 | – | **365** | – |
| 20 | Schaffer 2 | 6 | −100 | 100 | – | 45,587 | 7 | **3,254** | – |
| 21 | Schubert | 2 | −10 | 10 | 2,393 | **320** | – | 330 | – |
| 22 | Shekel 5 | 4 | 0 | 10 | **463** | 29,062 | 4 | 5,290 | – |
| 23 | Shekel 7 | 4 | 0 | 10 | **459** | 20,741 | 4 | 5,761 | – |
| 24 | Shekel 10 | 4 | 0 | 10 | **707** | 21,500 | 3 | 6,052 | – |
| 25 | Woods | 4 | −1 | 1 | **457** | 25,580 | 4 | 1,285 | – |
| 26 | Beale | 2 | 0 | 5 | 331 | 174 | – | **118** | – |
| 27 | Freudenstein–Roth | 2 | −15 | 15 | 5,889 | 2,067 | – | **1,886** | – |
| 28 | Jennrich–Sampson | 2 | −10 | 10 | **541** | 6,766 | – | 2,167 | – |
| 29 | Biggs 6 | 6 | 0 | 10 | **1,051** | 10,429 | – | 9,626 | – |
| 30 | Gulf | 3 | – | – | fail | **9,937** | – | 21,961 | 1 |
| 31 | Trigonometric | 5 | 0 | 1 | 1,271 | 9,251 | 1 | **809** | – |
| 32 | Trigonometric | 10 | 0 | 1 | **2,365** | 35,483 | 7 | 14,090 | – |
| 33 | Variably dimension | 10 | 0 | $e$ | fail | **2,759** | – | 4,098 | – |
| 34 | Variably dimension | 20 | 0 | $e$ | fail | **12,467** | – | 22,157 | – |
| 35 | Broyden 3D | 4 | −3 | 3 | 9,439 | 30,238 | 6 | **8,055** | – |
| 36 | Brown almost linear | 5 | −5 | 25 | 30,703 | **2,780** | – | 6,255 | – |
| 37 | Chebyquad | 9 | −1 | 1 | fail | 2,639 | – | **2,314** | – |
| 38 | Weka 1 | 10 | −1 | 1 | fail | **105** | – | 186 | – |
| 39 | Weka 2 | 2 | 0 | 1 | 1,565 | 2,796 | – | **524** | – |

**Table 1** continued

| | Function | $n$ | $L$ | $U$ | DIRECT | ARS | | OSCARS | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | $nf$ | fails | $nf$ | fails |
| 40 | Weka 2 | 6 | 0 | 1 | fail | 45,777 | 9 | **2,171** | – |
| 41 | Weka 3 | 2 | 0 | 1 | **1,545** | 22,556 | 3 | 2,258 | – |
| 42 | Weka 3 | 4 | 0 | 1 | 1,411 | 38,147 | 7 | **1,129** | – |
| 43 | mod. Beckers–Lago | 10 | −10 | 10 | **12,393** | 50,000 | 10 | 18,829 | – |
| 44 | Rastrigin | 2 | −1 | 1 | – | 3,306 | – | **2,818** | – |
| 45 | offset Rastrigin | 3 | −1 | 1 | 7,321 | 31,504 | 5 | **4,974** | – |
| 46 | ext. Easom | 10 | −10 | 10 | fail | **1,587** | – | 3,173 | – |
| 47 | ext. Easom | 20 | −10 | 10 | fail | **4,196** | – | 14,341 | – |
| 48 | ext. Easom | 30 | −10 | 10 | fail | 45,704 | 9 | **42,153** | 2 |
| 49 | | 2 | – | – | 361 | 506 | – | **242** | – |
| 50 | | 2 | – | – | **327** | 1,512 | – | 396 | – |

Here 'nf' is the average number (over ten runs) of function evaluations needed to reduce $f$ to within $10^{-3}$ of the global minimum. Runs using more than 50,000 function evaluations were halted at that figure, and listed as fails in the columns marked 'fail.' DIRECT is deterministic, so its column lists either the number of function evaluations needed to reach the required accuracy, or 'fail' if this number exceeded 50,000. Figures in bold mark the fastest method for each problem

and the extended Easom problem is $f = -\prod_{i=1}^{n} e^{-(x_i-\pi)^2}\cos(x_i)$. Problems 49 (using $\gamma = 0$) and 50 (with $\gamma = 0.1$) are given by the following function and bounds:

$$\frac{\gamma x + y - 1}{1 + 200(y - 1 + x^2)^2} \quad \text{subject to} \quad -1 \le x \le 1 \quad \text{and} \quad 0 \le y \le 1.$$

Weka 1 was defined as $f = \min\,(1 + \|x\|_2, 4x_1 + 4)$ for $n > 2$. Weka 3 is

$$f_{\text{weka3}} = \sum_{i=1}^{n} x_i(1 - x_i) + \sum_{j=0}^{\infty} 3^{(-j)} \left[ \sum_{i=1}^{n} \left( k_i \lfloor 3^j p_i x_i \rfloor \mod p_i \right) \right],$$

where the floor function $\lfloor s \rfloor$ denotes the greatest integer not larger than $s$. In practice the summation on $j$ was terminated at $j = 40$ as $3^{-40}$ is far below machine precision. Weka 2 can be obtained from Weka 3 by omitting the summation on $j$ and setting $j \equiv 0$. The parameter values used by Weka 2 (with $n = 6$) and Weka 3 (with $n = 4$) are respectively the first six and four entries in the lists $p_i = 2, 3, 5, 7, 11, 13$ and $k_i = 1, 1, 2, 2, 5, 5$. Weka 2 and 3 with $n = 2$ both used $p_i = 17, 19$ and $k_i = 11, 12$.

For most problems the upper and lower bounds are the same for each dimension, and these bounds are listed in Table 1 under $U$ and $L$ respectively. The bounds are as for [1] for the first 24 problems. The Woods problem has tighter bounds which place the solution in one corner of the feasible region. The lower and upper bounds for the Gulf problem were (0.1,0,0) and (100,25.6,5) respectively.

Table 1 compares the performances of ARS, OSCARS, and DIRECT. OSCARS used the values $A = 3/4$ and $h_{\min} = 10^{-5}$. Each test run was performed with a maximum of

**Table 2** An aggregated view of ARS and OSCARS performances on problems 1–50

| Method | Total nf | Fastest on | Fails | Avg norm nf | Worst norm nf |
|---|---|---|---|---|---|
| ARS | 625,882 | 9 | 87 | 5.586 | 33.8 |
| ARS with SORC | 453,967 | 14 | 53 | 4.226 | 41.8 |
| OSCARS (with SORC) | 294,859 | 16 | 4 | 1.766 | 10.2 |
| OSCARS no SORC | 297,035 | 11 | 9 | 1.736 | 10.4 |

Each method is tested both with and without the ‘Start Odd iterations with a Random Control point’ (SORC) option. The columns list the sum of average function evaluation counts for all 50 problems, the number of problems on which each method was the fastest, the number of failed runs, and average and worst normalized function evaluation counts. Function evaluation counts for each problem are normalized by dividing by the fastest method’s function count for that problem. Fails are counted at 50,000 evaluations, which biases the figures in columns 2, 5, and 6 favour of methods with more fails. DIRECT failed ten times on 41 runs

50,000 function evaluations, and terminated when a function value within $10^{-3}$ of the global minimum was obtained. Runs for problem 32 required a higher accuracy of $10^{-5}$ to eliminate a known local minimum. Runs not reaching the required accuracy within 50,000 function evaluations were regarded as fails. DIRECT is deterministic, so results for one run of DIRECT on each problem is listed. DIRECT always selects $\Omega$’s centre point as its first point, and so comparisons with DIRECT on problems with a global minimizer at $\Omega$’s centre are meaningless. These problems have a dash in DIRECT’s column. For ARS and OSCARS the average number of function evaluations used over 10 runs is listed, along with the number of failed runs. Failed runs are included in the average function evaluation counts at 50,000 evaluations each.

An aggregated comparison between ARS and OSCARS, each both with and without the SORC strategy are listed in Table 2. Results for ARS and OSCARS are taken from Table 1. Results for the other two methods were generated under identical conditions to those in Table 1. Normalized function counts for each problem were formed by dividing all function counts by the function count of the fastest method for that problem. The results show SORC substantially improves ARS. This occurs because if ARS locks onto a local minimizer, escape requires a sample point in the part of the global minimizer’s basin which lies below the local minimizer. If these two minimizers are far apart this can only occur when ARS draws a random sample from $\Omega$, meaning escape could be a low probability event. With SORC, ARS has a much higher chance of escape when the control point is random. OSCARS’ implicit partitioning of $\mathbb{R}^n$ into $2n$ cones when the control point remains constant means OSCARS explores $\Omega$ more evenly than ARS. SORC diverts effort away from improving the best known point and can slow a method when the global minimizer’s basin is easily located. For these reasons SORC’s overall effect on OSCARS’ speed is minimal, but it improves reliability on harder problems, and thus is preferred.

Numerical results were generated for comparison with the GLOBAL method [4] listed in Table 4 of [4]. The stricter stopping conditions in [4] were used, and as a consequence $h_{\min} = 10^{-8}$ was used for OSCARS. These results are listed in Table 3, and show that OSCARS was faster than GLOBAL, the original CGRASP [5], and the new version of C- GRASP [6] on 8, 12, and 8 of the 14 problems respectively.

**Table 3** This table lists the number of function evaluations required to solve problems 51–64 for GLOBAL [4], two versions of C- GRASP [5,6], and OSCARS

| | Function | $n$ | GLOBAL | C-GRASP [5] | C-GRASP [6] | OSCARS |
|---|---|---|---|---|---|---|
| 51 | Shekel 5 | 4 | 1,489 | 5,545,982 | 9,274 | 14,455 |
| 52 | Shekel 7 | 4 | 1,684 | 4,052,800 | 11,766 | 5,047 |
| 53 | Shekel 10 | 4 | 1,815 | 4,701,358 | 17,612 | 21,749 |
| 54 | Hartmann 3 | 3 | 3,608 | 20,743 | 1,719 | 361 |
| 55 | Hartmann 6 | 6 | 16,933 | 79,685 | 29,894 | 55,350 |
| 56 | Goldstein–Price | 2 | 923 | 29 | 53 | 398 |
| 57 | Branin | 2 | 1,023 | 59,857 | 10,090 | 143 |
| 58 | Rosenbrock | 2 | 6,274 | 1,158,350 | 23,544 | 3,893 |
| 59 | chain. Rosenbrock | 5 | 374,685 | 6,205,503 | 182,520 | 466,849 |
| 60 | chain. Rosenbrock | 10 | 1,908,469 | 20,282,529 | 725,281 | 4,058,216 |
| 61 | Easom | 2 | 1,604 | 89,630 | 5,093 | 154 |
| 62 | Schubert | 2 | 1,399 | 82,363 | 18,608 | 246 |
| 63 | Zakharov | 5 | 8,227 | 959 | 12,467 | 1,882 |
| 64 | Zakharov | 10 | 47,288 | 3,607,653 | 2,297,937 | 9,562 |

All entries are averages over ten runs

**Table 4** A comparison of OSCARS, two versions of ARS and recursive random search (RRS) [17] on the version of Rastrigin's function in [17] in 200 dimensions

| Method | 500 | 1,000 | 1,500 | 2,000 |
|---|---|---|---|---|
| ARS | 3,341 | 2,548 | 2,001 | 1,680 |
| ARS with SORC | 3,290 | 2,527 | 1,988 | 1,632 |
| OSCARS (with SORC) | 5,899 | 5,561 | 5,261 | 4,970 |
| RRS | 6,100 | 5,450 | 5,050 | 4,850 |

Entries are averages over 50 runs of function values after 500, . . . , 2,000 function evaluations

The problems that challenged OSCARS most were problems 8 (=55), 22–24 (=51–53), 30, 59, and 60. Problems 8, 59, and 60 are ill conditioned, while 22–24 and 30 have sizeable basins or flat areas which initially mislead the algorithm. ARS did better on functions which lack deep local minima far from the global minimizer(s), such as problems 17, 30, 33, and 34. OSCARS performed well on the discontinuous problems (problems 39–42).

Problems 11, 19, 31, 33, and 46 are solved in more than one choice of dimension $n$. These show that OSCARS copes at least as well with increasing dimension as ARS and DIRECT for $n \leq 30$. OSCARS reduces the tile size by shrinking along one coordinate axis each iteration with each shrink incurring one function evaluation. As $n$ increases, reducing the tile size in all directions becomes increasingly expensive. Hence one would expect OSCARS' performance to deteriorate as $n$ increases. To explore this, Table 4 compares OSCARS, ARS, and RRS [17] on [17]'s version of Rastrigin's function with $n = 200$. The superiority of both ARS versions is clear.

The relative costs of the overheads of OSCARS, ARS, and DIRECT were investigated by timing the methods on the offset Rastrigin function with $n = 30$ for various numbers of function evaluations. The times taken for overheads were constant for OSCARS and ARS at 1.7 and 1.3 times the time taken for the function evaluations alone. In contrast DIRECT's overheads rose steadily from 2.5 times at 10,000 function evaluations to 24.3 times at a million function evaluations.

Numerical results were also generated for $A = 0.1, 0.5$, and $0.9$, where $A$ is the fraction of each cut's displacement from $x^{(k)}$ to $c^{(k)}$. On average the results for $A = 1/2$ were very similar to $A = 3/4$, with $A = 0.9$ being only marginally worse. The choice $A = 0.1$ was clearly inferior, but still better than ARS and DIRECT. The choice of $A$ is not critical, with values in the region of $1/2$ to $3/4$ recommended.

## 5 Conclusion

A simple algorithm for global optimization inside a box shaped region has been presented, numerically tested, and shown to converge to an essential global minimizer with probability one. Numerical comparisons were made against accelerated random search and DIRECT. These showed the three methods were broadly similar when they worked well, but that OSCARS was more reliable than the other two methods. OSCARS is well suited to objective functions in low to moderate dimensions which can be evaluated quickly, including those needing very many function evaluations to optimize. In addition OSCARS' simple strategy of starting odd main iterations with random control points was shown to significantly improve accelerated random search.

**Acknowledgments** The authors would like to thank the two anonymous referees for many helpful comments leading to an improved paper.

## References

1. Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization problems. J. Global Optim. **31**, 635–672 (2005)
2. Ali, M.M., Storey, C.: Topographical multi-level single linkage. J. Global Optim. **5**, 349–358 (1994)
3. Appel, M.J., Labarre, R., Radulović, D.: On accelerated random search. SIAM J. Optim. **14**, 708–731 (2003)
4. Csendes, T., Pál, L., Sendín, J.O.H., Banga, J.R.: The GLOBAL optimization method revisited. Optim. Lett. **2**, 445–454 (2008)
5. Hirsch, M.J., Meneses, C.N., Pardalos, P.M., Resende, M.G.C.: Global optimization by continuous GRASP. Optim. Lett. **1**, 201–212 (2007)
6. Hirsch, M.J., Pardalos, P.M., Resende, M.G.C.: Speeding up continuous GRASP. Eur. J. Oper. Res. **205**, 507–521 (2010)
7. Jones, D., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**, 157–181 (1993)
8. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**, 17–41 (1981)
9. Pinter, J.: Convergence qualification of adaptive partition algorithms in global optimization. Math. Program. **56**, 343–360 (1992)
10. Pronzanto, L., Walter, E., Venot, A., Lebruchec, J.-F.: A general purpose global optimizer: implementation and applications. In: Mathematics and Computers in Simulation XXVI, pp. 412–422 (1984)

11. Price, C.J., Reale, M., Robertson, B.L.: A cover partitioning method for bound constrained global optimization. Optim. Methods Softw. **27**, 1059–1072 (2012)
12. Radulović, D.: Pure random search with exponential rate of convergency. Optimization **59**, 289–303 (2010)
13. Rinnooy Kan, A.H.G., Timmer, G.T.: Stochastic global optimization methods part I: clustering methods. Math. Program. **39**, 27–56 (1987)
14. Rinnooy Kan, A.H.G., Timmer, G.T.: Stochastic global optimization methods part II: multi-level methods. Math. Program. **39**, 57–78 (1987)
15. Tang, Z.Bo.: Adaptive partitioned random search to global optimization. IEEE Trans. Autom. Control **11**, 2235–2244 (1994)
16. Torn, A., Žilinskas, A.: Global optimization. Lecture Notes in Computer Science, vol. 350. Springer, Berlin (1989)
17. Ye, T., Kalyanaraman, S.: A recursive random search algorithm for large scale network parameter configuration. In: Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, pp. 196–205 (2003)