J Glob Optim (2008) 41:447–464
DOI 10.1007/s10898-007-9256-8

An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization

Kenneth Holmström

Received: 4 January 2006 / Accepted: 2 October 2007 / Published online: 7 November 2007

Abstract Powerful response surface methods based on kriging and radial basis function (RBF) interpolation have been developed for expensive, i.e. computationally costly, global nonconvex optimization. We have implemented some of these methods in the solvers *rbfSolve* and *EGO* in the TOMLAB Optimization Environment (http://www.tomopt.com/tomlab/). In this paper we study algorithms based on RBF interpolation. The practical performance of the RBF algorithm is sensitive to the initial experimental design, and to the static choice of target values. A new adaptive radial basis interpolation (ARBF) algorithm, suitable for parallel implementation, is presented. The algorithm is described in detail and its efficiency is analyzed on the standard test problem set of Dixon–Szegö. Results show that it outperforms the published results of *rbfSolve* and several other solvers.

Keywords Global optimization · Expensive function · CPU-intensive · Costly function · Mixed-integer · Nonconvex · Software · Black-box · Derivative-free · Response surface · Radial basis functions · Surrogate model · Response surface · Splines

1 Introduction

Global optimization of continuous black-box functions that are costly (CPU-intensive, computationally expensive) to evaluate is a challenging problem. Several approaches based on response surface techniques, most of which need to utilize every computed function value, have been developed over the years. In his excellent paper [11], Jones reviews the most important developments. Many methods have been developed based on statistical approaches, called kriging, see e.g. the Efficient Global Optimization (EGO) method in Jones et al. [13]. A second approach is found in the *MCS* solver described in [10]. In this paper we consider methods based on radial basis function interpolation, RBF methods, first discussed in [5,15]. Gutmann gave an interesting presentation of his algorithmic developments in 1999

K. Holmström (✉)
Department of Mathematics and Physics, Mälardalen University, P.O. Box 883, 721 23 Västerås, Sweden
e-mail: kenneth.holmstrom@mdh.se

[3], and since then we have been implementing RBF and kriging based algorithms. In [1], we describe a careful numerical implementation of the RBF algorithm in the solver *rbfSolve* that has been commercially available in the TOMLAB/CGO toolbox [7] for expensive problems since 2000. Originally aimed at simply bounded black-box problems, *rbfSolve* and the solver *EGO* has been improved to handle mixed-integer constrained expensive problems. Regis and Shoemaker discuss another way to utilize RBF interpolation in [16] and have developed a new algorithm called CORS-RBF. Our objective in this paper is to formulate a new adaptive RBF algorithm to overcome some of the weaknesses found in the RBF algorithm.

Problems that are costly to evaluate are commonly found in engineering design, industrial and financial applications. The function value could be the result of a complex computer program, an advanced simulation, e.g. computational fluid dynamics (CFD), tuning of financial trading strategies, or design optimization. One function value might require the solution of a large system of partial differential equations, and hence consume anything from a few minutes to many hours. In the application areas discussed, derivatives are most often hard to obtain and the algorithms make no use of such information. The practical functions involved are often noisy and nonsmooth, however the approximation methods used commonly assume smoothness. Our goal is to develop global optimization algorithms that work in practice and produce reasonably good solutions with a very limited number of function evaluations.

1.1 The costly global black-box nonconvex problem

$$\begin{aligned} &\min_x \ f(x) \\ &s/t \quad -\infty < x_L \le x \le x_U < \infty \end{aligned} \tag{1}$$

where $f(x) \in \mathbb{R}$, $x_L,\ x,\ x_U \in \mathbb{R}^d$. Let $\Omega \in \mathbb{R}^d$ be the compact set defined by the simple bounds in (1). The task of global optimization is to find the set of parameters x in the feasible region $\Omega \subset \mathbb{R}^d$ for which the objective function $f(x)$ obtains its lowest value. In other words, a point x^* is a *global optimizer* to $f(x)$ on Ω, if $f(x^*) \le f(x)$ for all $x \in \Omega$. On the other hand, a point $\hat{x}$ is a *local optimizer* to $f(x)$, if $f(\hat{x}) \le f(x)$ for all x in some neighborhood around $\hat{x}$. Obviously, when the objective function has several local minima, there could be solutions that are locally optimal but not globally optimal and standard local optimization techniques are likely to get stuck before the global minimum is reached. Therefore, some kind of global search is needed to find the global minimum with some reliability.

From an application perspective there are often restrictions on the variables besides the lower and upper bounds, such as linear, nonlinear or even integer constraints. Henceforth, we seek to solve a more complicated problem formulated as follows:

1.2 The mixed-integer costly (expensive) global black-box nonconvex problem

$$\begin{aligned} &\min_x \ f(x) \\ &\quad -\infty < x_L \le \ x \ \le x_U \ < \infty \\ &s/t \quad\quad b_L \le \ Ax \ \le b_U \\ &\quad\quad c_L \le c(x) \le c_U, \quad x_j \in \mathbb{N} \ \ \forall j \in \mathbb{I}, \end{aligned} \tag{2}$$

where $f(x) \in \mathbb{R}, x_L,\ x,\ x_U \in \mathbb{R}^d$, $A \in \mathbb{R}^{m_1 \times d}, b_L,\ b_U \in \mathbb{R}^{m_1}$ and $c_L,\ c(x),\ c_U \in \mathbb{R}^{m_2}$. The variables x_I are restricted to be integers, where $\mathbb{I}$ is an index subset of $\{1, \ldots, d\}$. Let

$\Omega_C \in \mathbb{R}^d$ be the feasible set defined by the constraints in (2). We assume that the function $f(x)$ is continuous with respect to all variables, even though we demand that some variables only take integer values. Otherwise it would not make sense to do surrogate modeling of $f(x)$. Another assumption is that the nonlinear constraints are cheap to compute compared to the costly $f(x)$. All costly constraints can be treated by adding penalty terms to the objective function in the following way:

$$\min_x \; p(x) = f(x) + \sum_i w_i \max\left(0,\; c^i(x) - c^i_U, c^i_L - c^i(x)\right), \tag{3}$$

where weighting parameters w_i have been added. As we have shown in [1] this strategy works in practice for an industrial train set design problem.

The idea of the RBF algorithm by Powell and Gutmann [5] is to use radial basis function interpolation to build an approximating surrogate model and define three utility functions. The next point, where the original objective function should be evaluated, is determined by optimizing one or more of these utility functions. The utility functions are derived from the definition of target values. The RBF methods described so far are based on static schemes in the selection of these values. However, the result of the algorithm is very sensitive to the choice of these. Therefore we propose a more general adaptive approach to set target values. The standard RBF algorithm commonly results in points sampled on the boundary, which leads to poor performance and non-convergence. In order to deal with this, we propose a one-dimensional search for suitable target values f_n^* to improve convergence. This leads to a sequence of global optimization problems to be solved in each iteration.

In Sect. 2 the RBF interpolation and algorithm are described. A detailed presentation of the new Adaptive RBF algorithm is given in Sect. 3. The convergence of ARBF is discussed in Sect. 4. The efficiency of the ARBF algorithm is analyzed on the standard test problem set of Dixon-Szegö [2] in Sect. 5. Repeated tests show that the adaptive implementation of the RBF algorithm is very robust and accurate on the problems compared to the standard RBF algorithm. It also outperforms several other solver options according to previously published results. Finally, in Sect. 6, we give some concluding remarks and summarize the new features. The generalization of the ARBF algorithm to solve the more complicated problem in (2) together with test results on standard mixed-integer nonlinear (MINLP) and nonconvex constrained problems are presented in [8]. A large part of the added complexity from linear, nonlinear and integer constraints is normally handed by the subsolvers used in conjunction with ARBF. This simplifies the generalization of the ARBF algorithm to handle MINLP.

2 The RBF method

First, the surrogate model used in the RBF method is defined. Given n distinct points $x_1, \ldots, x_n \in \Omega$ with known function values $F_i = f(x_i), i = 1, \ldots, n$, the radial basis function interpolant s_n has the form

$$s_n(x) = \sum_{i=1}^{n} \lambda_i \phi\left(\|x - x_i\|_2\right) + p(x), \tag{4}$$

where $\|\cdot\|$ is the Euclidean norm, $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ and p is in Π_m^d (the space of polynomials in d variables of degree less than or equal to m). Common choices of radial basis functions ϕ and the corresponding polynomial $p(x)$ and minimal polynomial degree m_ϕ are given in Table 1. When ϕ is either cubic with $\phi(r) = r^3$ or thin plate spline with $\phi(r) = r^2 \log r$, the

Table 1 Different choices of radial basis functions

RBF	$\phi(r) > 0$	$p(x)$	$m_\phi = \text{degree}(p(x))$
Cubic	r^3	$a^T \cdot x + b$	1
Thin plate spline	$r^2 \log r$	$a^T \cdot x + \text{b}$	1
Linear	r	b	0
Multiquadric	$\sqrt{(r^2 + \gamma^2)}, \gamma > 0$	b	0
Gaussian	$\exp(-\gamma r^2), \gamma > 0$	$\{0\}$	-1

radial basis function interpolant s_n has the form

$$s_n(x) = \sum_{i=1}^{n} \lambda_i \phi\left(\|x - x_i\|_2\right) + b^T x + a, \tag{5}$$

with $\lambda_1, \ldots, \lambda_n \in \mathbb{R}, b \in \mathbb{R}^d, a \in \mathbb{R}$. The unknown parameters λ_i, b, a are obtained as the solution of the linear equations

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}, \tag{6}$$

where Φ is the $n \times n$ matrix with $\Phi_{ij} = \phi\left(\|x_i - x_j\|_2\right)$ and

$$P = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_n^T & 1 \end{pmatrix}, \quad \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}, \quad c = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \\ a \end{pmatrix}, \quad F = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}. \tag{7}$$

If $\text{rank}(P) = d+1$, the matrix $\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix}$ is nonsingular and system (6) has a unique solution [14]. Thus a unique radial basis function interpolant to f at the points $x_1, \ldots, x_n$ is obtained.

After this, one has to consider the question of choosing the next point x_{n+1} to evaluate the objective function for. The idea of the RBF algorithm is to use radial basis function interpolation and a measure of 'bumpiness' of a radial function, σ. A target value f_n^* is chosen as an estimate of the global minimum of f. For each $y \notin \{x_1, \ldots, x_n\}$ there exists a radial basis function $s_y(x)$ that satisfies the interpolation conditions

$$\begin{aligned} s_y(x_i) &= f(x_i), \quad i = 1, \ldots, n, \\ s_y(y) &= f_n^*. \end{aligned} \tag{8}$$

The next point x_{n+1} is then calculated as the value of y in the feasible region that minimizes $\sigma(s_y)$. As a surrogate model is used, the function $y \mapsto \sigma(s_y)$ is much cheaper to compute than the original function. In [4], a 'bumpiness' measure $\sigma(s_n)$ is defined and it is shown that minimizing $\sigma(s_y)$ subject to the interpolation conditions (8) is equivalent to minimizing a utility function $g_n(y)$ defined as

$$g_n(y) = (-1)^{m_\phi + 1} \mu_n(y) \left[s_n(y) - f_n^*\right]^2, \quad y \in \Omega \setminus \{x_1, \ldots, x_n\}. \tag{9}$$

Writing the radial basis function solution to the target value interpolation problem (8) as

$$s_y(x) = s_n(x) + \left[f_n^* - s_n(y)\right] l_n(y, x), \quad x \in \mathbb{R}^d, \tag{10}$$

$\mu_n(y)$ is the coefficient corresponding to y of the radial basis interpolation function solution $l_n(y, x)$ that satisfies $l_n(y, x_i) = 0, i = 1, \ldots, n$ and $l_n(y, y) = 1$. $\mu_n(y)$ can be computed as follows. Φ is extended to

$$\Phi_y = \begin{pmatrix} \Phi & \phi_y \\ \phi_y^T & 0 \end{pmatrix}, \tag{11}$$

where $(\phi_y)_i = \phi(\|y - x_i\|_2), i = 1, \ldots, n$, and P is extended to

$$P_y = \begin{pmatrix} P \\ y^T \; 1 \end{pmatrix}. \tag{12}$$

Then $\mu_n(y)$ is the $(n + 1)$-th component of $v \in \mathbb{R}^{n+d+2}$ that solves the system

$$\begin{pmatrix} \Phi_y & P_y \\ P_y^T & 0 \end{pmatrix} v = \begin{pmatrix} 0_n \\ 1 \\ 0_{d+1} \end{pmatrix}. \tag{13}$$

The notations 0_n and 0_{d+1} are used for column vectors with all entries equal to zero and with dimension n and $(d + 1)$, respectively. The computation of $\mu_n(y)$ is done for many different y when minimizing $g_n(y)$. This requires $O(n^3)$ operations if not exploiting the structure of Φ_y and P_y. Hence, it does not make sense to solve the full system each time. A better alternative is to factorize the interpolation matrix and update the factorization for each y. An algorithm that requires $O(n^2)$ operations is described in [1].

Note that μ_n and g_n are not defined at $x_1, \ldots, x_n$ and

$$\lim_{y \to x_i} \mu_n(y) = \infty, \quad i = 1, \ldots, n. \tag{14}$$

This will cause problems when μ_n is evaluated at a point close to one of the known points. The function $h_n(x)$ defined by

$$h_n(x) = \begin{cases} \frac{1}{g_n(x)}, & x \notin \{x_1, \ldots, x_n\} \\ 0, & x \in \{x_1, \ldots, x_n\} \end{cases} \tag{15}$$

is differentiable everywhere on Ω, and is thus a better choice as an objective function. Instead of minimizing $g_n(y)$ in (9), Gutmann [5] suggests to minimize $-h_n(y)$. In [1] instead we propose to minimize $-\log(h_n(y))$. By this we avoid a flat minimum and numerical trouble when $h_n(y)$ is very small.

When there are large differences between function values, the interpolant has a tendency to oscillate strongly. It might also happen that $\min s_n(y)$ is much lower than the best known function value, which leads to a choice of f_n^* that overemphasizes global search. To handle these problems, large function values are in each iteration replaced by the median of all computed function values.

We are now ready to formulate a general description for the basic RBF algorithm, which has been discussed in [6] and [1].

Algorithm RBF:

- Find initial set of $n \geq d + 1$ sample points x_i using experimental design.
- Compute the n costly function values $f(x_i), i = 1, \ldots, n$.
- Find point with lowest function value $(x_{\text{Min}}, f_{\text{Min}})$ by computing $f_{\text{Min}}(x_{\text{Min}}) = \min_{i=1,\ldots,n} f(x_i)$.

- As an approximation of the function $f(x), x \in \Omega$, use the n sampled points to build a smooth RBF interpolation model $s_n(x)$ (surrogate model, response surface model) with chosen ϕ and $m \geq m_\phi$ from Table 1.
- Iterate until $n \geq n_{\text{Max}}$, or a prescribed maximal CPU time (or f_{Goal}, known goal for $f(x)$, achieved with a certain relative tolerance).
  - Find global minimum of the RBF surface, $s_n(x_{s_n}) = \min_{x\in\Omega} s_n(x)$.
  - In every iteration in sequence pick one of the $N + 2$ cycle step choices.
    1. **Cycle step** -1 **(InfStep).** Set target value $f_n^* = -\infty$, i.e. solve the global optimization problem
    $$g_n^\infty(x_{g_n}^\infty) = \min_{x\in\Omega\setminus\{x_1,\ldots,x_n\}} \mu_n(x), \tag{16}$$
    where $\mu_n(x)$ is computed as described in equation (13). Set $x_{n+1} = x_{g_n}^\infty$.
    2. **Cycle step** $k = 0, 1, \ldots, N-1$ **(Global search).** Define target value $f_n^* \in \left(-\infty, s_n(x_{s_n})\right]$ as $f_n^*(k) = s_n(x_{s_n}) - w_k \cdot \left(\max_i f(x_i) - s_n(x_{s_n})\right)$, with $w_k = (1 - k/N)^2$ or $w_k = 1 - k/N$. Solve the global optimization problem
    $$g_n(x_{g_n}^k) = \min_{x\in\Omega\setminus\{x_1,\ldots,x_n\}} (-1)^{m_\phi+1}\mu_n(x)\left[s_n(x) - f_n^*(k)\right]^2 \tag{17}$$
    and set $x_{n+1} = x_{g_n}^k$.
    3. **Cycle step** N **(Local search).**
    If $s_n(x_{s_n}) < f_{\text{Min}} - 10^{-6}|f_{\text{Min}}|$, accept x_{s_n} as the new search point x_{n+1}.
    Otherwise set $f_n^*(k) = f_{\text{Min}} - 10^{-2}|f_{\text{Min}}|$, solve (17) and set $x_{n+1} = x_{g_n}^k$.
  - If x_{n+1} is not too close to $x_1, \ldots, x_n$, accept x_{n+1} as search point and evaluate $f(x_{n+1})$.
  - Update the point with lowest function value $(x_{\text{Min}}, f_{\text{Min}})$, if $f(x_{n+1}) < f_{\text{Min}}$.
  - Increase n and compute new RBF surface.

The InfStep is optional, since for most problems, it does not improve the convergence to the global optimum. However, the coefficient $\mu_n(x)$ is always needed in the Global search step, and sometimes in the Local search step as well. Note that Gutmann [5] only considers one special case of the algorithm in which InfStep among others are not included. The range $\max_i f(x_i) - s_n(x_{s_n})$ may for many problems become too big. Gutmann suggests replacing $f(x_i) > median_i f(x_i)$ with $median_i f(x_i)$ both when computing the range and in the RBF interpolation. In practice one commonly needs to use some strategy to reduce the range. When large values are replaced by the median in the RBF interpolation, many numerical interpolation problems are avoided, but when additional points are sampled close to a stationary point, the function approximation gets less and less accurate in other parts of the space.

The RBF algorithm was implemented by Gutmann in a C program *RBFGLOB* described in his thesis [6]. The release of *rbfSolve* in TOMLAB has enabled users to solve many new practical problems, for example in [1], we describe the solution of a train set design problem. The first version was entirely written in Matlab, and too slow for practical use. Over the years the solver has been greatly improved with major speedups achieved from performing all time-consuming operations in Fortran. It is also possible to warm start the TOMLAB/CGO solvers from a previous run with any /CGO solver.

Based on [1], Regis and Shoemaker has made a research implementation of the RBF algorithm, and tried several algorithmic enhancements, some described in [16].

The RBF algorithm in practice is very sensitive to the choice of initial experimental design, especially when using stochastic designs. If the initial steps of the algorithm fail to find some

point in the basin of the global optimum, it often starts iterating repeatedly with sample points on the boundaries in the Global search, and only refines a local minima in the Local search.

Define the number of active variables $\alpha(x)$ as the number of elements of x that have components close to the bounds in the box, i.e.

$$\alpha(x) = |\{j \in 1, \ldots, d : |x^j - x_L^j| \leq \epsilon_x \text{ or } |x^j - x_U^j| \leq \epsilon_x\}|. \tag{18}$$

If a point is in the interior, then obviously $\alpha(x) = 0$. If studying $\alpha(x)$ during the iterations when running *rbfSolve* for many problems, the algorithm frequently generates points with some components on their bounds, $\alpha(x) > 0$. Doing a systematic study of the solution of (17) for many $f_n^* \in \left(-\infty, s_n(x_{s_n})\right]$ on different subproblems during the RBF iterations confirmed that the solution is typically not interior, and hence a careful choice of target value is needed. Solving a large set of problems (17) for different target values should generally be much less time-consuming than computing the costly $f(x)$. In addition computations for different target values are independent and could be done in parallel on different CPUs. By examining solutions for a large set of target values, it should be possible to find good search points in most iteration steps, and only evaluate the costly $f(x)$ for these points. In the next section a new adaptive RBF algorithm suitable for parallel implementation is formulated.

3 The adaptive radial basis algorithm (ARBF)

In this section the main ideas of the new Adaptive Radial Basis Algorithm are discussed and a formalized description is provided. To overcome the limitations of the RBF algorithm, the choice of target values must be made more flexible. The objective function class is very wide and a robust algorithm must adapt to the particular behavior of a function. A few choices of target values based on the function value range as in the RBF algorithm only works for nice well-behaved problems. This observation has been confirmed by practical experience with the RBF algorithm for a large set of real-life user problems over the past 6 years. Instead, a more adaptive algorithm is proposed, based on evaluating a large set of target values in each iteration. The approach is similar to two of the algorithms proposed by Jones in [11] to solve kriging problems, named the Enhanced Method 4 and Method 7.

Jones considers several kriging algorithms, e.g. Method 4, where the problem in each iteration is to maximize the probability of improvement after setting a target value. The optimal solution found is used as the new search point, and the costly $f(x)$ is evaluated for this search point and a new surrogate model of kriging type is computed. As in the RBF algorithm, it is a major difficulty to set the target value properly in each iteration. To overcome this problem, Jones proposes a new method called the Enhanced Method 4 that uses a range of target values in each iteration, corresponding to low, medium and high desired improvement. In the ARBF algorithm, target values are selected from a range, but the global optimization problem defined by (17) is solved instead.

Evaluating a large set of target values leads to many candidate points. If all were used, it would lead to several costly function evaluations in each iteration. Jones shows on one-dimensional examples that the optimal solutions tend to cluster in different areas of the parameter space. Similar behavior has been observed for the solutions of (17) with different target values. It is hence natural to apply a clustering algorithm to the set of optimal points $\{\hat{x}_j\}_{j=1}^M$ obtained by solving (17) with different target values. After this only one or a few points from each group found need to be used. The clustering process is executed with values mapped to the unit cube $[0, 1]^d$. As described below, Jones suggests applying a tailor-made

Table 2 Weight factors w_j used in the global grid search

0.0	10^{-4}	10^{-3}	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10	0.11
0.12	0.13	0.15	0.20	0.25	0.30	0.40	0.50	0.75	1.00	1.50	2.00	3.00	
100	∞												

clustering algorithm to the sequence of optimal points in decreasing target value order. The algorithm has been modified by adding steps 6 and 7. Motivation for these additional steps are given later on in this section. In the test in step 7 the number of components on bounds is used, defined as in (18).

3.1 The Jones Cluster Algorithm

- Transform the set of optimal points $\{x_j^*\}$ to the unit cube $[0, 1]^d$, and compute the distance between two successive optimal points as $\Delta_j = \sqrt{\sum_{l=1}^{d}(x_{jl}^* - x_{j+1,l}^*)^2/d}$.
- Assign point 1, x_1^*, to group 1.
- Sequentially consider point 2 to M. For each point, if a criterion $C > 12$, a new group is started. The criterion C is computed as follows:

  1. If $\Delta_j > 0.1$ and $\Delta_{j-1} > 0.1$ then set $C = 100$, i.e. start a new group.
  2. Otherwise, if $\Delta_j > 0.0005$, then set $C = \Delta_{j-1}/\Delta_j$.
  3. Otherwise, if $j \geq 3$ and $\Delta_j > 0.0005$, then set $C = \Delta_{j-1}/\max(\Delta_{j-2}, 0.0005)$.
  4. Otherwise, if $j = 2$ and $\Delta_1 > 0.1$ and $\Delta_2 < 0.0005$, then set $C = 100$ to signal the need for a new group.
  5. If none of the above conditions is satisfied, set $C = 0$, i.e. no need to start a new group unless any of the following two criteria are fulfilled.
  6. If $j = M$ and $C < 12$ and $\Delta_{M-1} > 0.1$, set $C = 100$, i.e. start a new group for the last point.
  7. If $C < 12$ and $\Delta_{j-1} > 0.1$ and $\alpha(x_j^*) = \alpha(x_{j-1}^*)$, check if any of the components on the bounds for point x_j^* have at least a 10% difference in the corresponding components in x_{j-1}^*. Also test if any of the components on the bounds for point x_{j-1}^* have at least a 10% change in the corresponding components in x_j^*. If any of the tests are true, start a new group by setting $C = C + 200$.

Let $f_{\text{Min}} = \min_i f(x_i)$ and $f_{\text{Max}} = \max_i f(x_i)$. Jones suggests setting the target values using a fixed grid as $f_n^*(j) = s_n(x_{s_n}) - w_j \cdot f_\Delta$, where the range is set to $f_\Delta = f_{\text{Max}} - f_{\text{Min}}$. The two first rows in Table 2 show the choice of target value factors w_j. In the new algorithm, two extreme values shown in row three have been added. It is then easier to detect if the range of target values is sufficient.

The above choice of f_Δ might become too big if the function varies over a large range. In such cases, as a fixed grid is used, there may be failure to sample important target values that would lead to the region of the global minimum. Since the range can only get larger as the iterations proceed, the algorithm is unlikely to sample these target values in later iterations. To be more flexible and adaptive, the target values are set as $f_n^*(j) = s_n(x_{s_n}) - \beta \cdot w_j \cdot f_\Delta$, where β is an adaptive factor and the range is

$$f_\Delta = \begin{cases} \min(\max(1, f_{\text{Min}}), f_{\text{Max}} - f_{\text{Min}}), & \text{if } f_{\text{Min}} > 0 \\ \min(10 \cdot \max(1, |f_{\text{Min}}|), f_{\text{Max}} - f_{\text{Min}}), & \text{if } f_{\text{Min}} \leq 0. \end{cases} \tag{19}$$

If several optimal solutions for different target values are equal or very close, it might be a sign that the target values are too close, so β is increased. If the optimal point found for the second target value is far from the solution of the first target value, i.e. the minimum of the RBF surface, it is a sign that the target values are too spread out, and β is decreased. In most cases this happens close to a stationary point.

As in the original RBF algorithm, every iteration of the ARBF algorithm starts by finding the global minimum of the RBF surface by solving

$$\hat{s}_n := s_n(x_{s_n}) = \min_{x \in \Omega} s_n(x).$$

If $\hat{s}_n \ll f_{\text{Min}}$ the RBF surface is fluctuating wildly and there is no point in applying a target value strategy. The target values need to have values even lower than $\hat{s}_n$, so they might be set much lower than the actual global minimum. Applying the target value strategy in such cases generally produces garbage solution points. Instead, the minimum of the RBF surface is added as a new point repeatedly until the interpolation stabilizes and more reasonable target values can be set. If $\hat{s}_n$ is closer to f_{Min} and the oscillation of the surface is less pronounced, the question is when to switch to the target value strategy. The RBF surface is still considered wildly fluctuating, i.e. the same strategy is applied, outside a relative difference of 10%. This works well in tests so far, but other values could be tried. Currently the following test is used: If $\hat{s}_n < f_{\text{Min}} - 0.1|f_{\text{Min}}|$ when $f_{\text{Min}} \neq 0$, or $\hat{s}_n < f_{\text{Min}} - 10v$ when $f_{\text{Min}} = 0$, then $\hat{s}_n \ll f_{\text{Min}}$ is considered true. v is computed as $v = \min(f(x)), x \in \{x : f(x) > 10^{-7}\}$. For the special case when the set is empty, $v = 10^{-7}$ is used.

For every ARBF iteration, the algorithm is in one of three modes: the wild mode described in the previous paragraph, a global grid search mode, or a local grid search mode. In the global grid search the aim is to sample one or more points from every region of interest. In the local grid search the aim is to find a better approximation of any stationary points close to the best point found so far. Ideally one of these stationary points is also the global minimum. Note that the global grid search adds sample points in the vicinity of the currently best known point, but to a lesser accuracy than the local grid search. In the wild mode the aim is to proceed with surface minimum points until the interpolation is stable enough to make a global target value grid give reasonable results. The wild mode is entered automatically when the surface is fluctuating wildly, but the switch between global and local grid mode is determined by the algorithm in the following way: Start with global grid mode, and as long as this gives function value reductions, stay in that mode. Both the global and local grid mode always end by adding the minimum point of the surface. When no reduction is achieved in the global mode (or possibly only in the final surface minimum sampling) the algorithm switches to local mode. The same logic applies for local mode: it continues the local grid search until no reductions are achieved, and then switches to global mode. In both the local and global grid mode, one or more points might be selected using the cluster algorithm and some heuristic rules (given later in this section). The formal ARBF algorithm description can now be given.

Algorithm ARBF:

- Find initial set of $n \geq d + 1$ sample points x_i using experimental design.
- Compute the n costly function values $f(x_i), i = 1, \ldots, n$.
- Find point with lowest function value $(x_{\text{Min}}, f_{\text{Min}})$ by computing $f_{\text{Min}}(x_{\text{Min}}) = \min_{i=1,\ldots,n} f(x_i)$.
- Use the n sampled points to build a smooth RBF interpolation model $s_n(x)$ with chosen ϕ and $m \geq m_\phi$ from Table 1.

- Set $GlobalProgress = 1$ and $LocalProgress = 0$, to make the initial search mode global.
- Iterate until $n \geq n_{\text{Max}}$, or a prescribed maximal CPU time (or f_{Goal}, known goal for $f(x)$, achieved with a certain relative tolerance).
  - Find the global minimum of the RBF surface, $s_n(x_{s_n}) = \min_{x \in \Omega} s_n(x)$.
  - Find a set of new search points $X = \{\hat{x}_j,\ j = 1, \ldots, k\}$ by applying one of the following three types of search procedures dependent on logical conditions given for each procedure.
    1. **Wild Mode (S-step).** If $s_n(x_{s_n}) \ll f_{\text{Min}}$ or $EndGridMode$ is set, accept the RBF surface minimum x_{s_n} as the new search point, i.e. $X = x_{s_n}$.
       Set $EndGridMode = 0$.
    2. **Global Grid Mode. (G-step).** If *GlobalProgress=1*, define M target values $f_n^* \in (-\infty, s_n(x_{sn})]$ as $f_n^*(j) = s_n(x_{s_n}) - \beta \cdot w_j \cdot f_\Delta$ with $w_j,\ j = 1, \ldots, M$ a vector of predefined factors in the range $[0, \infty]$, and β an adaptive weight factor. The function range f_Δ is determined in each step as described in (19). For each of the M target values, solve the global optimization problem

$$g_n(x_j^*) = \min_{x \in \Omega \setminus \{x_1, \ldots, x_n\}} (-1)^{m_\phi + 1} \mu_n(x) \left[s_n(x) - f_n^*(j)\right]^2. \tag{20}$$

       Then use the Jones Clustering Algorithm on the M optimal solution points x_j^*.
       Apply heuristic rules to determine which of the clustered groups to consider, and in each selected group, which of the points to include in the new set of search points X; see the *Point Selection Algorithm* later in this section.
       Set $EndGridMode = 1$.
    3. **Local Grid Mode. (L-step).** If $LocalProgress = 1$, define M_L target values using the same factors w_j as in the G-step together with some additional small factors.
       Solve (20) and apply the Jones Clustering Algorithm to the M_L optimal solutions x_j^*.
       Apply the heuristic rules described in the *Point Selection Algorithm* to determine which points in the first cluster group should be included in set X.
       Set $EndGridMode = 1$.
  - Check the set of new search points $X = \{\hat{x}_j,\ j = 1, \ldots, k\}$, deleting any point too close to the sampled points $x_1, \ldots, x_n$ or to one of the other points in X.
  - Set $x_{n+j} = \hat{x}_j,\ j = 1, \ldots, k$ and evaluate $f(x_{n+j}),\ j = 1, \ldots, k$.
  - If $\min_{j=1,\ldots,k} f(x_{n+j}) < f_{\text{Min}}$
    * Update the point with lowest function value (x_{Min}, f_{Min}).
    * Set $LocalProgress = 1$ (if L-step).
    * Set $GlobalProgress = 1$ (if G-step).

    else
    * Set $LocalProgress = 0$ (if L-step).
    * Set $GlobalProgress = 0$ (if G-step).
  - Increase n by k and compute new RBF surface.

The selection of trial points utilizing the result of the clustering process applied to the set of optimal solutions computed from the target values is one of the heuristics. For kriging algorithms, Jones suggests picking the last member of each of the groups formed by the clustering algorithm as a new candidate point, i.e. the one with the smallest target value in each group. This selection criteria has been found a bit crude when applied to RBF algorithms. As a practical example refer to the standard test problems (the Shekel test problems from

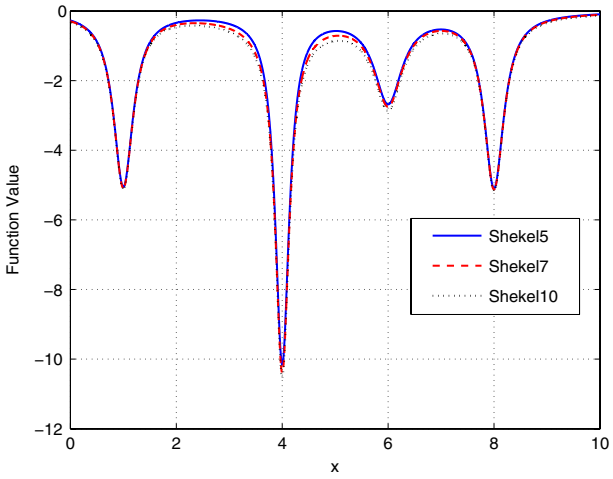

Fig. 1 Plot of the Shekel 5, Shekel 7 and Shekel 10, functions along the line segment from (0, 0, 0, 0) to (10, 10, 10, 10)

the Dixon–Szegö [2] set) that have been very troublesome for the RBF algorithm. These problems have very steep minima, i.e. local minimizers that lie at the bottom of steep and narrow basins. In Fig. 1 a cut in the four-dimensional space for the three problems are shown. Shekel5 has five local minima in $[0, 10]^4$, four of which lie along the diagonal segment from (0, 0, 0, 0) to (10, 10, 10, 10). These four local minima corresponds to the local minima in Fig. 1 and the steepest one is the global minimum point. Shekel 7 and Shekel 10 has 7 and 10 local minima, respectively, and as can be seen in Fig. 1 have a very similar behavior along the line segment as Shekel 5.

Using the $2^4 = 16$ corner points as the initial experimental design and applying the ARBF algorithm, the clustering results as shown by Table 3 are obtained, where the actual Shekel 5 function values $f(x_j^*)$ are displayed in the last column. Δ_j is the distance measure and C_j is the criterion in the Jones Cluster Algorithm. Grp is the number of the group that the jth element belongs to. $Dist$ is the distance from the first group member to the group member in the jth row.

As seen in the table, the clustering results in six groups ($n_{\text{Group}} = 6$). Looking at the target values $f_n^*(j)$ and the corresponding optimal solutions x_j^* it is evident that many components in the solutions are on the bounds. This has been observed as a common pattern for many of the solutions of (20). The column with $\alpha(x_j^*)$ values shows the number of components in x_j^* that are on the bounds using a loose tolerance, $\epsilon_x = 10^{-3}$. Group 5 has interior solutions, with no components on the bounds. This is an interesting set of solutions with points in the basin of three local minima, of which one is the global minimum, see Fig. 1. Picking only the last component would slow down the convergence to the global minima, or even in some cases make the algorithm only reach a local minima using the maximal number of function evaluations (e.g. 250).

A *Point Selection Algorithm* has been developed to deal with this issue. It generates new trial points primarily based on the results from the Jones Cluster Algorithm. For the Shekel 5 problem, the ARBF algorithm, using the Point Selection Algorithm, finds the global minimum with 4 digits of accuracy in only 43 function evaluations, whereas picking only the last group component results in 67 function evaluations.

Table 3 The initial grid search and the result of the Jones clustering algorithm on the Shekel 5 ($x \in [0,\ 10]^d$, but scaled to $[0,\ 1]^d$ below) test problem from the Dixon–Szegö set of test functions

j	Grp	Δ_j	C_j	Dist	$x^*_{j1}, \ldots, x^*_{j4}$	$f^*_n(j)$	$\alpha(x^*_j)$	$f(x^*_j)$
1	1	0.00121	NaN	0.000	[0.000, 0.000, 0.000, 0.000]	−0.2731	4	−0.27312
2	1	0.00403	0.2998	0.001	[0.000, 0.000, 0.002, 0.000]	−0.2731	3	−0.27593
3	1	0.02337	0.1723	0.005	[0.000, 0.000, 0.010, 0.000]	−0.2734	3	−0.28526
4	1	0.01169	1.9996	0.023	[0.000, 0.000, 0.000, 0.046]	−0.2755	3	−0.32277
5	1	0.05563	0.2101	0.034	[0.000, 0.000, 0.000, 0.069]	−0.2778	3	−0.34100
6	1	0.00780	7.1290	0.044	[0.000, 0.087, 0.000, 0.000]	−0.2802	3	−0.34929
7	1	0.07731	0.1010	0.051	[0.000, 0.103, 0.000, 0.000]	−0.2825	3	−0.35135
8	2	0.00610	12.6650	0.000	[0.000, 0.000, 0.115, 0.000]	−0.2849	3	−0.34879
9	2	0.09477	0.0644	0.006	[0.000, 0.000, 0.128, 0.000]	−0.2872	3	−0.34418
10	3	0.00515	18.4139	0.000	[0.000, 0.000, 0.000, 0.140]	−0.2896	3	−0.33782
11	3	0.00476	1.0801	0.005	[0.000, 0.000, 0.000, 0.150]	−0.2919	3	−0.33016
12	3	0.00444	1.0738	0.010	[0.000, 0.000, 0.000, 0.160]	−0.2943	3	−0.32194
13	3	0.00415	1.0687	0.014	[0.000, 0.000, 0.000, 0.169]	−0.2967	3	−0.31352
14	3	0.00390	1.0644	0.019	[0.000, 0.000, 0.000, 0.177]	−0.2990	3	−0.30511
15	3	0.00368	1.0609	0.022	[0.000, 0.000, 0.000, 0.185]	−0.3014	3	−0.29689
16	3	0.14087	0.0261	0.026	[0.000, 0.000, 0.000, 0.192]	−0.3037	3	−0.28894
17	3	0.01425	9.8820	0.125	[0.000, 0.206, 0.000, 0.000]	−0.3084	3	−0.27407
18	3	0.01147	1.2429	0.137	[0.000, 0.234, 0.000, 0.000]	−0.3202	3	−0.24333
19	3	0.18879	0.0608	0.147	[0.000, 0.257, 0.000, 0.000]	−0.3320	3	−0.22049
20	4	0.17739	100.0000	0.000	[0.000, 0.000, 0.000, 0.276]	−0.3437	3	−0.20335
21	5	0.03510	205.0534	0.000	[0.193, 0.220, 0.192, 0.220]	−0.3673	0	−0.31805
22	5	0.08462	0.4148	0.035	[0.231, 0.252, 0.231, 0.252]	−0.3908	0	−0.27072
23	5	0.06126	1.3813	0.120	[0.318, 0.334, 0.318, 0.334]	−0.4497	0	−0.55948
24	5	0.04936	1.2412	0.181	[0.380, 0.394, 0.380, 0.394]	−0.5085	0	−5.54053
25	5	0.01899	2.5988	0.230	[0.431, 0.442, 0.431, 0.442]	−0.6262	0	−1.74012
26	5	0.01648	1.1526	0.249	[0.450, 0.460, 0.451, 0.460]	−0.7439	0	−0.96267
27	5	0.02759	0.5973	0.266	[0.468, 0.476, 0.468, 0.476]	−0.9793	0	−0.70704
28	6	0.00079	34.7723	0.000	[0.500, 0.499, 0.499, 0.499]	−23.814	0	−0.57592
29	6	0.00000	0.0288	0.000	[0.500, 0.500, 0.500, 0.500]	$-\infty$	0	−0.57535

The Point Selection Algorithm tries to find a smaller subset of optimal solutions that are of interest. There are three distinct parts in the selection procedure.

In any group there may be solutions that have components on the bounds. If a solution has more components on the bounds than any other solutions within the cluster group it is rejected. Furthermore, if the solutions in a cluster group with the lowest number of components on bounds have more components on bounds than any solution in other clusters, the whole cluster group is rejected.

The algorithm treats the first and last cluster group differently. The cluster group size, i.e. the distance between the first and the last optimal solution in the group, is used to determine how many trial solutions to select.

The individual distances between optimal solutions are used for the selected cluster groups in between. If solutions are very close, only one solution is selected, otherwise every solution is included in the set X. The motivation is that these solutions far away from each other may be in the basin of different local minima, as seen in the Shekel example discussed above. Following is a formal description of the algorithm:

3.2 Point selection algorithm

- Compute the minimum and maximum number of components on the bounds in the cluster groups,

$$\alpha^G_{\text{Min}} = \min_{j \in \text{Group } G} \alpha(x^*_j),\ G = 1, \ldots, n_{\text{Group}}$$

and

$$\alpha^G_{\text{Max}} = \max_{j \in \text{Group } G} \alpha(x^*_j),\ G = 1, \ldots, n_{\text{Group}}.$$

- Compute the minimum over all groups of both the minimal number of components on the bounds in the group, and the maximal number of components on the bounds in the group, i.e. compute

$$\alpha_{\text{Min}} = \min_{G = 1,\ldots,n_{\text{Group}}} \alpha^G_{\text{Min}}$$

and

$$\alpha_{\text{Max}} = \min_{G = 1,\ldots,n_{\text{Group}}} \alpha^G_{\text{Max}}.$$

- Only the points in each group G that has the minimal number of components on the bounds, α^G_{Min}, are considered for inclusion in the set X of trial points. Define

$$\bar{x}^G = \{x^*_j : j \in \text{Group } G,\ \alpha(x^*_j) = \alpha^G_{\text{Min}}\}$$

as the set of acceptable points for each group G and let $|\bar{G}|$ denote the number of acceptable points in the shrinked group.
- Compute the normalized distance from the first point in each shrinked group to every other point in the group, and denote the result $\bar{\delta}^G_i$, $i = 1, \ldots, |\bar{G}|$. The normalized distance is computed using the same formula as Δ_i in the Jones Cluster algorithm. Note that $\bar{\delta}^G_1 = 0$ and $\bar{\delta}^G_{|\bar{G}|}$ is the cluster group size. Column Dist in Table 3 shows a numerical example.
- Set $\epsilon_B = 0.1$. Used to test if the cluster group size is large or not.
- Set $\epsilon_G = 0.03$. Used to test if the distance between individual solutions is large or not.
- Different rules are applied for the first group close to the RBF surface minimum ($G = 1$), the last group including the infinite target value ($G = n_{\text{Group}}$) or any group in between ($G = 1, \ldots, n_{\text{Group}} - 1$):

1. If $G = 1$
   - *If* $\alpha^1_{\text{Min}} = \alpha_{\text{Min}}$ or L-step, one of the first elements in group 1 is included in X based on a heuristic set of rules. For the L-step another element is often added. The rules are dependent on the relative error in the last sampled point, $\frac{|s_n(x_n) - f(x_n)|}{|f(x_n)|}$, which gives a measure of how accurate the RBF interpolation is and how close the currently best point is to a stationary point.
   - *If* $\bar{\delta}^1_{|\bar{G}|} > \epsilon_B$, the last element in the first group, $\bar{x}^1_{|\bar{G}|}$, is included in X. Compute the element closest to half the size of the first group, i.e.

$$i^* = \text{argmin} \min_i ||\bar{\delta}^1_i - \bar{\delta}^1_{|\bar{G}|}/2||.$$

   The solution $\bar{x}^1_{i^*}$ is included in X.

- *elseif* either $\bar{\delta}^1_{|\bar{G}|} > \frac{\epsilon_B}{2}$ or ($\bar{\delta}^1_{|\bar{G}|} > \frac{\epsilon_B}{5}$ and $|\bar{G}| \geq \frac{2}{3} \cdot M$), then the last element in the first group, $\bar{x}^G_{|\bar{G}|}$, is included in X. Note that the second logical expression is true if the number of elements in the first group is very large and the group size is not too small.

2. If $G = n_{\text{Group}}$
   - *If* $\alpha^{n_{\text{Group}}}_{\text{Min}} = \alpha_{\text{Min}}$ and $\bar{\delta}^{n_{\text{Group}}}_{|\bar{G}|} > \epsilon_B$, the last group is unusually large. Two points are selected, the first and the last, i.e. $\bar{x}^{n_{\text{Group}}}_1$ and $\bar{x}^{n_{\text{Group}}}_{|\bar{G}|}$ are included in X.
   - *elseif* $\alpha(x^*_{|G|}) = \alpha_{\text{Min}}$, then add the last point in the last group, $\bar{x}^{n_{\text{Group}}}_{|\bar{G}|}$ to X. This test is fulfilled if the optimal solution to (20) with infinite target value, $f^*_n = -\infty$, has a minimal number of components on the bounds.
3. *If* $G = 2$ and $\alpha^2_{\text{Min}} = \alpha_{\text{Min}}$ or for $G = 3, \ldots, n_{\text{Group}-1}$: if $\alpha^G_{\text{Min}} = \alpha_{\text{Min}}$ and (either $\alpha^G_{\text{Max}} = \alpha_{\text{Max}}$ or $\bar{\delta}^G_{|\bar{G}|} > \epsilon_B$), then select points from group G using the following procedure:
   - Add the first point $\bar{x}^G_1$ to X.
   - Set $D = 0$.
   - For $i = 2, \ldots, |\bar{G}|$: if $\bar{\delta}^G_i - D > \epsilon_G$, add $\bar{x}^G_i$ to set X and set $D = \bar{\delta}^G_i$.

Applying the Point Selection Algorithm on the result in Table 3, it can be seen that the size of the first group is sufficiently large for the inclusion of one point, $0.051 > \frac{\epsilon_B}{2} = 0.05$. The last point in group 1, number 7, is selected. The last group 6 consists of two interior points, and the first element, number 28, is selected. Note that the criterion value for number 21, $C_{21} > 200$. This element was hence selected using the additional rule 7 in the Jones Clustering Algorithm. For the RBF interpolation it is important whether or not the trial points are close to the bounds, as closeness to the bounds is a clear indication of failure. Such points are of no interest and only slows down the convergence of the algorithm to the global minimum. It is therefore important to distinguish between points on bounds and points which clearly have values far from the bounds. In rule 7 a new group is started if one of the components have separated at least the distance 0.1 from the neighboring bound component. In this case candidate 21 have three out of four components changing around 0.2 compared to the previous solution point, and it is quite natural to consider point 21–27 as a new group. The first element in group 5, number 21, is selected. The distance between each successive point in this group is larger than ϵ_G until point number 26, which is not included, while number 27 is far enough from number 25. In total six of the seven points in group 5 will be included in the trial set X, number 21, ..., 25, 27, together with number 7 and 28. No points are selected in group 2, 3 and 4 because all group members have too many components on the bounds. Note that the above only applies when implementing a G-step, as opposed to the L-step where only the first cluster group ($G = 1$) is considered in the *Point Selection Algorithm*.

4 The convergence of the ARBF method

The aim is to prove convergence of the method for any continuous function f. The theorem by Törn and Zilinskas [17] states that the sequence generated by the RBF and ARBF algorithms should be dense. Applied to these methods the theorem states

Theorem 1 *The algorithm converges for every continuous function f if and only if it generates a sequence of points that are dense in Ω.*

The convergence of the RBF method is proved and discussed in detail in Sect. 4 in Gutmann [5]. The new ARBF algorithm fits into the general framework of the RBF algorithm as formulated in [5], there referenced as *Algorithm 3*. The requirement for the chosen target values is that in every iteration step the target value is chosen in $f_n^* \in \left[-\infty, s_n(x_{s_n})\right]$. In every cycle of the ARBF algorithm we pick at least one point in this set. Gutmann suggests several static algorithms for the choice of target values, but in our ARBF algorithm the target value is dynamically chosen as described in the previous section. Therefore the ARBF algorithm will have the same convergence properties as is proved by Gutmann for the RBF method called *Algorithm 3*.

The convergence results does not allow a free choice of the target values. In order to achieve convergence we have to assure that the target values are low enough in an infinite number of steps. The following condition must hold:

$$s_n(x_{s_n}) - f_n^* > \tau \Delta_{n+1}^{\rho/2} ||s_n||_\infty, \tag{21}$$

where $\tau > 0$ and $\rho \geq 0$ are constants and $\rho < 1$ is the linear spline case, and $\rho < 2$ the cubic and thin plate spline cases. The $||.||_\infty$ denotes the maximum norm of a function on Ω, defined as

$$||g||_\infty := \max_{x \in \Omega} |g(x)|, \, g \in C(\Omega)$$

and we define the minimal distance to the new point as

$$\Delta_{n+1} = \min_{1 \leq j \leq n} ||x_{n+1} - x_j||.$$

Convergence to the global minimum has so far only been possible to establish for the surface spline type functions, presented in the following compact reformulation of Gutmann's major convergence theorem:

Theorem 2 *Let $\phi(r) = r, \phi(r) = r^2 \log r$ or $\phi(r) = r^3$. Further, choose an integer m, the polynomial degree, such that $0 \leq m \leq d$ in the linear spline case, $0 \leq m \leq d+1$ in the thin plate spline case and $0 \leq m \leq d+2$ in the cubic spline case. Let $(x_n)_{n \in \mathbb{N}}$ be the sequence generated by the ARBF algorithm, and s_n be the radial function that interpolates $(x_i, f(x_i)), i = 1, \ldots, n$. Assume that for infinitely many $n \in \mathbb{N}$, the choice of f_n^* satisfies (21). Then the sequence (x_n) is dense in Ω.*

With the assumption of smoothness of f, Gutmann is able to show that the right hand side of (21) can be replaced by $\tau \Delta_{n+1}^{\rho/2}$, hence this constraint on the target values f_n^* can be easily checked. We prefer to avoid any additional assumptions in our ARBF method and instead always add the target value $f_n^* = -\infty$ in every iteration. We then apply a corollary from Gutmann, which proves the convergence of our ARBF method:

Theorem 3 *Let the assumptions of Theorem 2 on ϕ and m hold. Further, let f be continuous, and, for infinitely many $n \in \mathbb{N}$, let $f_n^* = -\infty$. Then the ARBF method converges.*

5 Numerical results

In this section the results obtained for the standard test functions from Dixon and Szegö [2] are reported. Table 4 gives a compact description of the test functions, including the abbreviation used, the problem dimension d, the number of local and global minima and the simple bounds.

Table 4 Compact description of the Dixon–Szegö test functions

Function	Abbrev.	Dim.	No. of loc. min	No. of glob. min	Domain
Shekel 5	S5	4	5	1	$[0, 10]^4$
Shekel 7	S7	4	7	1	$[0, 10]^4$
Shekel 10	S10	4	10	1	$[0, 10]^4$
Hartman 3	H3	3	4	1	$[0, 1]^3$
Hartman 6	H6	6	4	1	$[0, 1]^6$
Branin	BR	2	3	3	$[-5, 10] \times [0, 15]$
Goldstein–Price	GP	2	4	1	$[-2, 2]^2$

Table 5 Number of function evaluations needed to achieve a function value with relative error less than 1% for *ARBFMIP*, *rbfSolve*, *RBFGLOB*, *CORS-RBF*, *DIRECT*, *EGO*, *MCS* and *DE*

	ARBFMIP	*rbfSolve*	*RBFGLOB*	CORS-RBF	*DIRECT*	*EGO*	*MCS*	*DE*
S5	**34**	96	76	41	103	–	83	6400
S7	**31**	72	76	46	97	–	106	6194
S10	**25**	76	51	51	97	–	103	6251
H3	31	**22**	43	25	83	35	79	476
H6	**43**	87	112	108	213	121	74	7220
BR	**22**	26	44	34	63	28	30	1190
GP	**21**	27	63	49	101	32	40	1018

The definitions of problem S5, S7, S10, H3, H6 and GP are taken from [2] and problem BR from [9, p. 468]. The ARBF algorithm is implemented in the TOMLAB solver *ARBFMIP* and like *rbfSolve* available in the TOMLAB/CGO toolbox. Subsolvers are optional, but in our tests the global solver *glcDirect* and the local solver *SNOPT* have been used. The global subproblem solutions have been verified by running the slower *OQNLP* solver.

Table 5 shows the number of function evaluations needed to achieve a function value with relative error less than 1%. The relative error is defined as

$$E = \frac{f_{\text{Min}} - f_{\text{global}}}{|f_{\text{global}}|}, \tag{22}$$

where f_{Min} is the current best function value and f_{global} is the known global optimum (which is nonzero for all the problems in Table 4). For the results reported in Table 5, *ARBFMIP* was run with default values and $\phi(r) = r^3$ for problems H6 and GP, and with $\phi(r) = r^2 \log r$ for the other problems. *rbfSolve* was executed with $\phi(r) = r^3$, the search space being transformed to the unit hypercube and large function values replaced by the median. The results for the *RBFGLOB* and the *DE* solver were from [5] and the results for the CORS-RBF algorithm from [16]. The results for *DIRECT* were taken from [12] and the results for *EGO* from [13], where no results were presented for the S5, S7 and S10 problems. We tested *MCS* version 2.0 with parameter settings suggested by Dr. Waltrud Huyer; $smax = 5 \cdot d + 10, iinit = 0, local = 50$ and $gamma = eps \approx 2 \cdot 10^{-16}$ (the MATLAB floating point relative accuracy).

The best result for each test problem is marked bold. The results for the new ARBMIP solver is clearly very promising. It performs best for six out of seven problems. For one problem it pays a price for the additional robustness, and uses slightly more function evaluations than the RBF algorithm in *rbfSolve*.

Table 6 Number of function evaluations needed to achieve a function value with relative error less than 0.01% for *ARBFMIP*, *RBFGLOB*, *DIRECT*, *MCS* (best value and average value)

	ARBFMIP	*RBFGLOB*	*DIRECT*	*MCS* (Best)	*MCS* (Average)
S5	**43**	100	155	83	583
S7	**59**	125	145	129	633
S10	**47**	112	145	103	595
H3	82	158	199	**79**	131
H6	**64**	160	571	111	113
BR	**37**	64	195	41	51
GP	**26**	76	191	81	94

Table 6 shows the number of function evaluations needed to achieve a function value with relative error less than 0.01%. Even if the algorithm has achieved two digits of relative accuracy (1%), it might be a hard task to find two more, and many publications have not reported any such results. The results for the *MCS* solver are stochastic, and both the best and the average value are shown, taken from [10]. Looking at the results in Table 6, we draw the conclusion that the ARBF algorithm has very good convergence properties. Results were best for six out of seven problems. For the H3 problem the *MCS* (Best) result is slightly better.

All corner points were used in the initial experimental design for *ARBFMIP*, except for the H6 problem, where 7 corner points and the center point were used to avoid unnecessary sampling. A limited set of points in the initial design has proven more suitable for large-dimensional problems, In fact, the number of additional search points needed for convergence for the H6 problem with 8 initial points, were almost the same as compared to using all 64 corner points. This is very promising and shows that an initial strategy based on limited corner sampling is feasible for higher-dimensional problems. The performance for *rbfSolve* and *RBFGLOB* were worse than in the tables if started with these 8 initial points.

6 Conclusions

Methods based on radial basis interpolation are powerful tools in solving expensive black-box optimization problems. This paper has discussed the RBF algorithm, which has been used in practice for many years and seems to be the best method available. We have discussed some weak points with the algorithm and new methods to overcome the problems.

The RBF method uses a static selection of target values and is very dependent on the scaling of the problem. The global target value optimization problem often does not produce interior points, and the search points computed does not help the practical convergence of the algorithm.

We present the details of a new Adaptive RBF (ARBF) method. In every iteration, the method does an extensive search for target values to produce a suitable selection of search points. It is possible to parallelize the global target value optimization as well as the costly function evaluations. These two tasks are the most CPU intensive parts of the algorithm. We also show that the new ARBF algorithm have similar convergence properties as the RBF algorithm.

The results obtained for the Dixon and Szegö test functions show that ARBF is a very promising algorithm. Further development and tests are needed to show that the results are robust and that similar good performance can be achieved on real-life problems for different experimental designs.

References

1. Björkman, M., Holmström, K.: Global optimization of costly nonconvex functions using radial basis functions. Optim. Eng. **1**(4), 373–397 (2000)
2. Dixon, L.C.W., Szegö, G.P.: The global optimisation problem: an introduction. In: Dixon, L., Szego G. (eds.) Toward Global Optimization, vol. 2, pp. 1–15. New York (1978)
3. Gutmann, H.-M.: A Radial Basis for Function Method for Global Optimization. In: Talk at IFIP TC7 Conference, Cambridge (1999a)
4. Gutmann, H.-M.: A radial basis function method for global optimization. Technical Report DAMTP 1999/NA22. Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England (1999b)
5. Gutmann, H.-M.: A radial basis function method for global optimization. J. Global Optim. **19**, 201–227 (2001a)
6. Gutmann, H.-M.: Radial basis function methods for global optimization. Doctoral Thesis, Department of Numerical Analysis, Cambridge University, Cambridge, UK (2001b)
7. Holmström, K., Edvall, M.M.: Chapter 19: The TOMLAB optimization environment. In: Josef Kallrath, L.G. (ed.) Modeling Languages in Mathematical Optimization. Boston/Dordrecht/London (2004)
8. Holmström, K., Quttineh, N.-H., Edvall, M.M.: An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Optim. Eng. (Under review) (2007)
9. Horst, R., Pardalos, P.M.: Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht, Boston, London (1995)
10. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate Search. J. Global Optim. **14**, 331–355 (1999)
11. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Global Optim. **21**, 345–383 (2002)
12. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**(1), 157–181 (1993)
13. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**, 455–492 (1998)
14. Powell, M.J.D.: The theory of radial basis function approximation in 1990. In: Light, W. (ed.) Advances in Numerical Analysis, vol. 2: Wavelets, Subdivision Algorithms and Radial Basis Functions, pp. 105–210. Oxford University Press (1992)
15. Powell, M.J.D.: Recent research at Cambridge on radial basis functions. In: Buhmann, M.D., Felten, M., Mache, D., Müller, M.W. (eds.) New Developments in Approximation Theory, pp. 215–232. Birkhäuser, Basel (1999)
16. Regis, R.G., Shoemaker, C.A.: Constrained global optimization of expensive black box functions using radial basis functions. J. Global Optim. **31**(1), 153–171 (2005)
17. Törn, A., Zilinskas, A.: Lecture Notes in Computer Science, vol. 350. Berlin Heidelberg (1987)