# A Trust-Region Algorithm for Global Optimization [*]

Bernardetta Addis[†] and Sven Leyffer[‡]

August 6, 2004

**Abstract**

We consider the global minimization of a bound-constrained function with a so-called funnel structure. We develop a two-phase procedure that uses sampling, local optimization, and Gaussian smoothing to construct a smooth model of the underlying funnel. The procedure is embedded in a trust-region framework that avoids the pitfalls of a fixed sampling radius. We present a numerical comparison to three popular methods and show that the new algorithm is robust and uses up to 20 times fewer local minimizations steps.

**Keywords:** Global optimization, smoothing, trust region.
**AMS-MSC2000:** 90C26, 90C59.

## 1 Introduction

We consider the global optimization problem

$$\left\{ \begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & x \in S \subset \mathbb{R}^n, \end{array} \right. \tag{1.1}$$

where $f$ is sufficiently smooth and $S \subset \mathbb{R}^n$ is a compact set with simple structure, such as a bounded box. We require $S$ to be simple for two reasons. The first reason is that we then can sample a uniform point in $S$ without too much computational effort. The second reason is that we can use bound-constrained solvers and avoid possible difficulties caused by the local solver converging to an infeasible point.

Problems of type (1.1) arise in diverse fields, in particular, well-known conformational problems such as protein folding and atomic/cluster problems. In these applications we are interested in finding the lower free energy conformation in three-dimensional space. A box can be defined that eventually will contain all "interesting" molecular conformations.

[*]Preprint ANL/MCS-P1190-0804

[†]Dipartimento Sistemi e Informatica, Università di Firenze, via di S. Marta 3, Firenze 50129, Italy (b.addis@ing.unifi.it).

[‡]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA (leyffer@mcs.anl.gov).

If the problem allows the use of a sufficiently efficient local optimization algorithm, a two-phase procedure is a good candidate for global optimization [16]. Such a procedure involves sampling coupled with local searches started from the sampled points. We define the local minimization operator as

$$L(x) := \begin{cases} \underset{y}{\text{minimize}} & f(y) \quad \text{starting from} \quad x \\ \text{subject to} & y \in S. \end{cases} \tag{1.2}$$

We note that this operator is implicitly defined and depends on the local minimizer used. In general, $L(x)$ is a piecewise constant function whose pieces correspond to the basins of attraction of the local minimizers of $f(x)$.

Clearly, the global optimization problem (1.1) has the same optimal objective value as the following problem:

$$\begin{cases} \underset{x}{\text{minimize}} & L(x) \\ \text{subject to} & x \in S. \end{cases} \tag{1.3}$$

We note that the piecewise constant nature of $L(x)$ implies that the minimizers of (1.1) and (1.3) need not agree. In fact, any global minimizer of (1.1) is also a global minimizer of (1.3), but not vice versa. Because $L(x)$ is implicitly defined, however, we can simply record

$$x_{min} := \mathcal{LS}(x) := \begin{cases} \underset{y}{\text{argmin}} & f(y) \quad \text{starting from} \quad x \\ \text{subject to} & y \in S. \end{cases} \tag{1.4}$$

It follows that $x_{min}$ is also a local minimizer of $f(x)$, and we can recover a global minimizer of $f(x)$ by solving (1.3) in this way.

Multistart is an elementary example of a two-phase method aimed at minimizing $L(x)$; in practice, it reduces to a purely random (uniform) sampling applied to $L(x)$. It is in principle possible to apply any known global optimization method to solve the transformed problem (1.3), but many difficulties arise. First, function evaluation becomes much more expensive: we have to perform a local search on the original problem in order to observe the function $L(x)$ at a single point. Second, the analytical form of $L(x)$ is not available, and it is a discontinuous, piecewise constant function.

Given these difficulties, most two-phase methods have been designed without exploiting the fact that the true objective function to be minimized is $L(x)$ instead of $f(x)$. Indeed, many promising two-phase methods (e.g., multilevel single-linkage [14] or simple-linkage clustering approaches [11, 15]) neglect in some sense the piecewise constant shape of $L(x)$ and concentrate most of their effort on improvements over the multistart method. In particular, for clustering methods, improvements over multistart are obtained through a sequential decision procedure that chooses starting points it deems worthwhile for a local search. Such strategies are doomed to fail, however, when either the number of variables is high or the number of local optima is huge, situations that are both extremely common (e.g., in most molecular conformation problems [5]). It is widely believed that in these cases the local optimum points are not randomly displaced but that the objective function $f(x)$ displays a so-called funnel structure. A univariate example of such a function is given in Figure 1, where the function to be minimized is represented by the solid line and the underlying funnel structure is given by the dotted line. In general, we say $f(x)$

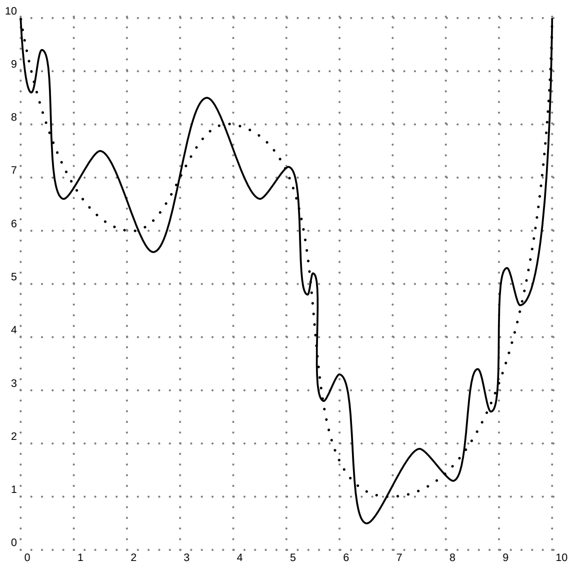

Figure 1: Example of a funnel function

has funnel structure if it is a perturbation of an underlying function with a low number of local minima. Motivated by examples of this kind, some authors [12, 13, 18] have proposed filtering approaches: if one can filter the high frequencies that perturb the funnel structure, then one can recover the underlying funnel structure and use a standard global optimization method on the filtered function (which is much easier to globally optimize) in order to reach the global optimum.

In contrast, we believe that it is better to filter the piecewise linear function $L(x)$ because it is less oscillatory than $f(x)$; Figure 2 shows $L(x)$ for the simple funnel function previously presented. This follows the approach of [2], and much of the analysis in [2] also applies here.

In this paper we make two important contributions to global optimization. First, we remove the need for the arbitrary parameters in [2] by interpreting these parameters as a trust-region radius. We embed the algorithm from [2], called ALSO, in a trust-region framework and show that our new algorithm is more robust than other methods. Second, we introduce the concept of global quality. This concept is motivated by the fact that the trust-region framework is essentially a local optimization scheme and therefore requires modifications to be effective as a global method.

The remainder of the paper is organized as follows. In Section 2 we introduce the smoothing scheme, in Section 3 we introduce the trust-region framework for global optimization, and in Section 4 we present numerical results for test problems having a funnel structure.

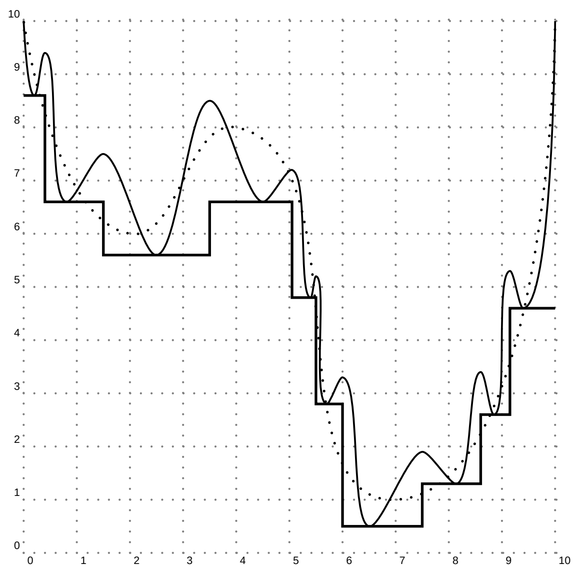

Figure 2: Example of the effect of local minimization

## 2 Gaussian Smoothing for Global Minimization

In this section we introduce a smoothing scheme to solve (1.3). The material of this section largely follows [2], although we give a different emphasis. We apply smoothing to $L(x)$ for two reasons. First, $L(x)$ is a piecewise constant function for which descent directions are difficult to define (first-order derivatives, when defined, are always zero). Second, we expect the smoothing to provide a more global view of the function.

Given a real-valued function $L : \mathbb{R}^n \to \mathbb{R}$ and a smoothing kernel $g : \mathbb{R} \to \mathbb{R}$, which is a continuous, bounded, nonnegative, symmetric function whose integral is one, we define the $g$–transform of $L(x)$ as

$$\langle L \rangle_g(x) = \int_{\mathbb{R}^n} L(y) g(\|y - x\|) \, \mathrm{d}y. \tag{2.5}$$

The value $\langle L \rangle_g(x)$ is an average of the values of $L(x)$ in all the domain; in particular, the closer the points are to $x$, the higher is the contribution to the resulting value. Another important property is that $\langle L \rangle_g(x)$ is a differentiable function. Hence we can use standard smooth optimization methods to minimize it.

The most widely used kernel in the literature is the Gaussian kernel

$$g(z) \propto \exp\left(-z^2/(2\sigma^2)\right), \tag{2.6}$$

where we use the symbol $\propto$ to avoid writing a multiplicative constant that plays no role in the methods we present here. In Figure 3 we present an example of the resulting smoothing for different values of the parameter $\sigma$. In particular, the smoothing effect is more evident if the parameter is larger.

We refer to [2] for some theoretical properties of the smoothing function in one dimension. In the one-dimensional case we can assume without loss of generality that the

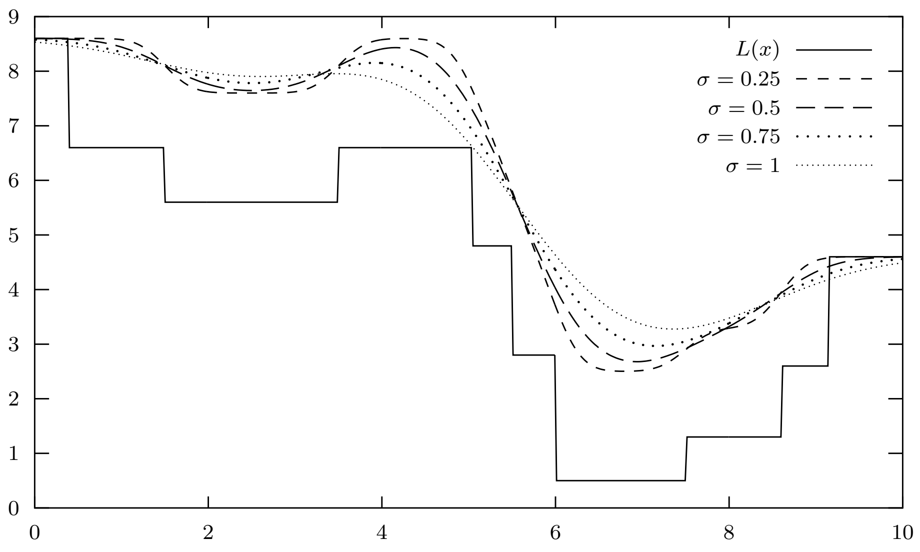

Figure 3: Gaussian filtering of $L(x)$

function $L(x)$ associated to a generic funnel function can be written as

$$L(x) = \sum_{i=1}^{N} V_i \, \mathbf{1}_{x \in [a_{i-1}, a_i)}, \tag{2.7}$$

where $a_0 = -\infty < a_1 < \cdots < a_{N-1} < a_N = +\infty$, and $V_i \in \mathbb{R}, i = 1, \ldots, N$, with the condition that

$$\begin{array}{ll} V_{i+1} < V_i & i = 1, \ldots, \ell - 1 \\ V_{i+1} > V_i & i = \ell, \ldots, N-1 \end{array} \tag{2.8}$$

for an index $\ell \in \{1, \ldots, N\}$. Notice that the global minimum value is $V_\ell$. Here $\mathbf{1}_{x \in [a_{i-1}, a_i)}$ is the indicator function for the interval $[a_{i-1}, a_i)$.

Let $g(x)$ be a kernel (which, in particular, is a probability density function). In [2] the following theorem is proved.

**Theorem 1** *Let $g(x)$ be a continuously differentiable probability density function whose support is $\mathbb{R}$. If $g$ is logarithmically concave (i.e., if $\log g(x)$ is concave) and if the step function $L$ defined in (2.7) satisfies (2.8), then $\langle L \rangle_g(x)$ is either monotonic or unimodal. If $g(x)$ is* strictly *log-concave, then the transform has at most a single minimum point.*

From this theorem it immediately follows that, for example, if $g$ is a Gaussian kernel, then the transform always has one and only one minimum point. In [2] it is also shown that if the variance of the density function $g$ is sufficiently small, then the minimum point occurs inside the interval corresponding to the bottom step of the objective function. Being restricted to the one-dimensional case, all these results are of limited usefulness. For the multidimensional case, however, they motivate the notion of a "path of descending steps down to the global minimum," which leads to results similar to those obtained for the one-dimensional case.

Clearly, one cannot explicitly apply the smoothing operator as in (2.5) because this approach requires the approximation of an $n$-dimensional integral. Instead, we restrict our attention to a ball of radius $\Delta$ around the current point $x$ ($B(x, \Delta)$) and define

$$\langle L \rangle_g^B(x) = \int_{B(x,\Delta)} L(y) \frac{g(\|x - y\|)}{\int_{B(x,\Delta)} g(\|x - t\|)\, \mathrm{d}t} \, \mathrm{d}y. \tag{2.9}$$

We cannot obtain an analytical expression of (2.9) because of the integral and the fact that the expression of $L(x)$ is unknown. Nor is a numerical estimate of the integral practical because we would need to evaluate $L(x)$ at a large number of points.

To solve these difficulties, we construct a discretized version of the smoothing of $L(x)$,

$$\hat{L}_g^B(x) = \sum_{i=1}^{K} L(y_i) \frac{g(\|y_i - x\|)}{\sum_{i=1}^{K} g(\|y_i - x\|)}, \tag{2.10}$$

where $y_i$, $i = 1...K$, are samples in $B(x, \Delta)$. Under mild assumptions, for a sufficiently large number of samples, $\hat{L}_g^B(x)$ is a good approximation of the original smoothed function. Even for an arbitrary number of samples, it has interesting properties: it is a continuous function and a convex sum of the values of the samples. In particular, the weight associated with each sample is larger if we are closer to the sample point. In other words, the more confident we are in the sample value, the greater is the weight associated with it. In [2], the model (2.10) is used to choose the new candidate point $x_+$, starting from a point (say, $x_k$). The model in the subset of given radius $\Delta$, $B(x_k, \Delta)$, is solved by using a constrained optimization procedure. The algorithm, called ALSO, is described in Algorithm 1.

When a candidate point $x_+$ is found, an unconstrained local optimization of the original objective function $f(x)$ is performed because, in general, $x_+$ is not a local minimum of $f(x)$. This procedure is equivalent to evaluating $L(x_+)$. If we obtain an improvement, the local minimum is taken as a new point, the center of the new region $B$. Otherwise the new center is $x_+$.

In contrast to other procedures such as monotonic basin hopping (MBH), see [6] this model allows us to identify a search direction even when all the samples assume a value larger than the current record.

The use of a fixed value for $\Delta$ is restrictive for the length of the steps we are allowed to take. In addition, it is not clear a priori what value $\Delta$ should take. We do know that a small radius results in small steps and, hence, in slow convergence, but large values of $\Delta$ can result in poor agreement between the model and the function and, hence, in useless candidate points. Therefore, we propose to embed ALSO within a trust-region framework that adjusts the radius $\Delta$ automatically. The details of this new framework are discussed in the next section.

## 3 Trust-Region Framework for Global Optimization

In this section we show how to embed ALSO in a trust-region framework that adaptively updates the radius $\Delta$. The motivation for this approach is that it avoids the pitfalls

```
Data : $\Delta$, $K$, $N$
$n = 0$, $k = 0$, $M = \emptyset$
$x =$ random uniform point in $S$
$x_0 = x^\star = \mathcal{LS}(x)$
$record = L(x^\star)$
while $n < N$ do
    repeat
        Collect a sample in $B(x_k, \Delta)$ and add it to $\mathcal{K}$
    until a new record is found or $| \mathcal{K} | < K$
    if $\min_{y \in \mathcal{K}} L(y) < record$ then
        $n = 0$
        $x^\star = x_{k+1} = \arg\min_{y \in \mathcal{K}} L(y)$
    else
        $n = n + K$
        Add the samples to $M$
        Construct the model (using samples in $M$)
        Solve $x_+ = \arg\min_{x \in B(x_k, \Delta)} \hat{L}_g^B(x)$
        $x = \mathcal{LS}(x_+)$
        if $L(x) < record$ then
            $n = 0$
            $x^\star = x_{k+1} = x$
        else
            $x_{k+1} = x_+$
    $k = k + 1$
    Set $M = \emptyset$
```

**Algorithm 1**: ALSO

of unsuitable trust-region radii. Trust-region methods are traditionally used in local optimization to force convergence to local minima from remote starting points. To extend the trust-region framework to global optimization, we introduce the concept of global quality, and we use a model improvement step.

The use of trust-region methods for local optimization dates back to [7]; see the comprehensive book [3]. We review the basic idea of a trust-region method for the unconstrained local optimization of a smooth function $f(x)$. At a given iterate $x_k$, we construct a (second order) Taylor series model $m_k(x)$. This model is then minimized in the trust region, usually a ball of radius $\Delta_k$ around $x_k$. If the new point, $x_+$, improves the objective, we move to it and construct a new model. Otherwise, we improve the agreement between $f(x)$ and the model $m_k(x)$ by reducing the trust-region radius $\Delta_k$. Convergence follows from the fact that $f(x)$ and $m_k(x)$ agree up to first order.

To extend the trust-region framework to global optimization of $L(x)$, we need to modify the smooth local trust-region algorithm. Specifically, we have to account for the fact that $L(x)$ is nonsmooth and that $\hat{L}_g^B(x)$ does not agree with $L(x)$ to first order. More important, we wish to avoid getting trapped in local minima. Clearly, it is not sufficient to simply reduce the trust-region radius if we reject the step.

At each iteration we construct the model $\hat{L}_g^B(x)$ around our current iterate $x_k$. Specifically, we choose $K$ samples (uniform) inside the current trust-region $B(x_k, \Delta_k)$ and perform a local minimization of $f(x)$ from each sample. If we find a new best point during this phase, we simply move to the new point and construct a new model. Otherwise, we apply a local minimization to the model inside the trust region and obtain

$$x_+ = \operatorname*{argmin}_{x \in B(x_k, \Delta_k)} \hat{L}_g^{B_k(x)}. \tag{3.11}$$

To decide whether to accept a step, we compute the ratio of the actual to the predicted reduction, namely,

$$\rho = \frac{L(x_k) - L(x_+)}{\hat{L}_g^{B_k}(x_k) - \hat{L}_g^{B_k}(x_+)}, \tag{3.12}$$

noting that the predicted reduction $\hat{L}_g^{B_k}(x_k) - \hat{L}_g^{B_k}(x_+)$ is always nonnegative. We accept the new point $x_+$ if we observe sufficient decrease, that is $\rho \geq \eta_1 > 0$. If the step is very successful, $\rho \geq \eta_2 > \eta_1$, and the trust-region is active, $\|x_k - x_+\| \geq \Delta$, then we increase the trust region radius for the next iteration. As long as $\rho \geq \eta_1$, we refer to the iteration as successful, otherwise ($\rho < \eta_1$) the iteration is referred to as unsuccessful. Unsuccessful trust-region iterations require special attention in the global optimization setting. In smooth local optimization, reducing $\Delta$ is guaranteed to improve the agreement between the model and the objective function. The same is not true in the global optimization context. Hence, we introduce a measure for the global quality of our model $\hat{L}_g^{B_k(x)}$,

$$q(\hat{L}_g^{B_k(x)}) = \frac{\max_{i \in M} \mid \{y_j : L(y_j) = L(y_i)\} \mid}{M}, \tag{3.13}$$

where $M$ is the set of collected samples, that is, the largest number of samples with the same objective value, divided by the total number of samples. Clearly, $0 \leq q(\hat{L}_g^{B_k(x)}) \leq 1$, and a value close to 1 means that a large number of samples have the same function value and stem from the same "flat region" of $L(x)$. A smaller value of $q(\hat{L}_g^{B_k(x)})$ implies that the samples represent the global nature of the function $L(x)$ better.

In our algorithm, we compute $q(\hat{L}_g^{B_k(x)})$ at every unsuccessful iteration. If it is larger than a fixed value $\bar{q}$, we remove all but one sample from the largest set, increase the trust-region radius, and obtain new uniform samples in $B(x_k, \Delta_{k+1}) \backslash B(x_k, \Delta_k)$. The motivation for this step is twofold: it improves the global nature of the model $\hat{L}_g^{B_k(x)}$, and it increases $\sigma$, thus smoothing the model.

The increase of $\sigma$ arises because we have adopted the following formula for calculating the smoothing parameter, depending on the trust-region radius $\Delta$ and the number of samples $K$:

$$\sigma = \frac{\Delta}{K^{1/n}}, \tag{3.14}$$

where $n$ is the dimension of the problem. The aim of this choice of $\sigma$ is to cover the trust-region volume with the Gaussian weights. The largest part of the volume (more than 60%) of a Gaussian is a ball of radius $\sigma$ and center zero. If we divide the volume of our trust region, which is proportional to $\Delta^n$, by the number of samples, we get the volume that has to be covered by the Gaussians. Thus, to obtain equal coverage for different trust-region values, we need $K = \frac{\Delta^n}{\sigma^n}$, which gives (3.14).

We can now state the complete global optimization trust-region algorithm. Let $0 \leq \bar{q} \leq 1$ be a bound on global quality. Let the trust-region parameters $0 < \eta_1 < \eta_2 \leq 1$ be fixed. Let $N$ be a given upper bound for the number of samples, $\beta_1, \beta_2 > 1$ and $0 < m \leq 1$. The algorithm is presented in Algorithm 2. Typical values for the parameters are provided in Section 4.

```
Data : $\Delta$, $K$, $N$
$n = 0$, $k = 0$, $M = \emptyset$, $\Delta_k = \Delta$
$x$ = random uniform point in $S$
$x_k$=$x^\star = \mathcal{LS}(x)$
$record = L(x^\star)$
while $n < N$ do
    repeat
        Collect a sample in $B(x_k, \Delta_k)$ and add it to $\mathcal{K}$
    until a new record is found or $\mid \mathcal{K} \mid < K$
    if $\min_{y \in \mathcal{K}} L(y) < record$ then
        $n = 0$ and set $M = \emptyset$
        $x^\star = x_{k+1} = \arg\min_{y \in \mathcal{K}} L(y)$
    else
        $n = n + K$
        Add the samples to $M$
        Construct the model (using samples in $M$)
        Solve $x_+ = \text{argmin}_{x \in B(x_k, \Delta_k)} \hat{L}_g^{B_k(x)}$
        Evaluate $\rho$
        $x = \mathcal{LS}(x_+)$
        if $\rho \geq \eta_1$ then
            $n = 0$ and set $M = \emptyset$
            $x^\star = x_{k+1} = x$
            if $\rho > \eta_2$ and StepLenght $\geq \Delta$ then
                $\Delta_{k+1} = \Delta_k \beta_1$
        else
            $x_{k+1} = x_k$
            if $q(\hat{L}_g^{B_k(x)}) \leq \bar{q}$ then
                $\Delta_{k+1}$=$\Delta_k / \beta_2$ (only every two consecutive steps)
            else
                Throw away all but one sample with same value
                $\Delta_{k+1} = \Delta_k \beta_1$
    $k = k + 1$
```

**Algorithm 2**: TRF

We conclude this section with a few remarks on our trust region. We have a pool of samples $M$ that is used to collect measures. If we do not achieve good agreement, we collect new samples, and we add them to the set $M$ (with the old ones). We use the following strategy for improving model agreement. If the global quality is low, we increase the trust-region radius and throw away part of the samples; in particular, we delete all but one of the samples with the same value that determined the poor global quality. The aim of this strategy is to increase the diversity between samples and avoid the region around the center $x_k$ from becoming flat. We decrease the radius only if the agreement is poor and the global quality is sufficient. We note that reducing the trust-region radius can be considered dangerous for a global optimization strategy because it restricts the method to a local search. Hence, we take a conservative approach: we reduce the trust region only every two consecutive steps of poor agreement with good global quality.

We note that, in case of poor agreement, we update the trust-region parameters and keep the old points only if we do not move to a new point. If we do move, the new point can be far away from the old point, and the information on the latter—in particular, the samples collected—may be useless for the former.

## 4 Numerical Results

To evaluate the computational performance of our algorithm, we tested it on several functions from the literature. The test problems we chose have the following characteristics: they are box unconstrained, their dimension can be chosen, the global optimum is known, they possess a very high number of local optima, and they have a funnel structure.

As a reference, we give the test problems with their relative boxes and the value of the global minimum:

**Rastrigin** [19]

$$Ras(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i) \tag{4.15}$$

with $x_i \in [-5.12, 5.12]$ , $f^\star = 0.0$

**Levy** [8]

$$Levy(x) = 10\sin^2(\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1})) + (x_n - 1)^2 \tag{4.16}$$

with $x_i \in [-10, 10]$ , $f^\star = 0.0$

**Ackley** [1]

$$Ack(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(2\pi x_i)) \tag{4.17}$$

with $x_i \in [-32.768, 32.768]$ , $f^\star = -20.0 - e$

**Schwefel** [17]

$$Sch(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}) \tag{4.18}$$

with $x_i \in [-500, 500]$ , $f^\star = -418.9829n$

We also tested our method on

**Scaled Rastrigin**

$$ScaledRas(x) = 10n + \sum_{i=1}^{n} (\alpha_i x_i)^2 - 10\cos(2\pi(\alpha_i x_i)) \tag{4.19}$$

with $x_i \in [-5.12, 5.12]$ , $f^\star = 0.0$

where $\alpha_i = 1$ if $\lfloor i/10 \rfloor \equiv 0 (\text{mod } 2)$ and otherwise $\alpha_i = 2$ (i.e., $\alpha_i = 1$ for the first ten variables, then 2 for the next ten variables, and so on). This test function was introduced in order to check the behavior of the method in presence of asymmetric level sets.

The trust-region algorithm is run with the following parameter values. The initial trust-region radius $\Delta$ is taken from results in the literature and is different for every run. The decrease factor for $\Delta$ is $\beta_1 = 1.11$ and the increase factor is $\beta_2 = 1.2$. The global quality threshold is $\overline{q} = 0.6$. The parameters for step acceptance, and trust-region increase are $\eta_1 = 0.001$, $\eta_2 = 0.75$, respectively.

We compare our results with the ones obtained with ALSO. We report as reference the results of the tests using MBH, even if in the majority of the cases ALSO outperforms it. We test all algorithms for a range of initial trust-region radii. The initial radii were chosen to optimize the performance of MBH. In other words, our new algorithm competes against "optimal $\Delta$ values" of the other methods. In practice, it is unlikely that a user knows a good value for $\Delta$, and in our examples the "optimal" $\Delta$ is the result of numerous tests.

Both reference methods, ALSO and MBH, have no strategy to adapt the radius $\Delta$. To our knowledge the only adaptive version of MBH is proposed in [10]. We describe the strategy for updating the radius using the same notation used in the description of MBH method (see Algorithm 3). Let $\bar{N}$ be a positive integer, $l \in (0, 1)$ and set the radius $\tilde{\Delta} = \Delta$. Then, every $\bar{N}$ steps, evaluate the fraction $p$ of iterations for which $z_k \neq x_k$ (that is the candidate point is different from the center of sampling ball), and update $\tilde{\Delta}$ using the following strategy:

$$\text{if} \quad p = 1.0 \begin{cases} \tilde{\Delta} > \Delta, & \text{then} \quad \tilde{\Delta} = \tilde{\Delta} - \Delta \\ \tilde{\Delta} \leq \Delta, & \text{then} \quad \tilde{\Delta} = \tilde{\Delta}/2 \end{cases} \tag{4.20}$$

$$\text{if} \quad p < 1.0 \begin{cases} \tilde{\Delta} \geq \Delta, & \text{then} \quad \tilde{\Delta} = \tilde{\Delta} + \Delta \\ \tilde{\Delta} < \Delta, & \text{then} \quad \tilde{\Delta} = 2\tilde{\Delta}. \end{cases} \tag{4.21}$$

We have introduced this radius update in our MBH procedure and refer to the resulting method as AMBH (adaptive MBH). In this way we can use the same local search procedure for all the methods presented and perform tests with the same randomly generated starting points. The parameters $\bar{N}$ and $l$ are chosen as in [10]; that is, $\bar{N} = 10$ and $l = 0.8$. We test AMBH for all the radius values used for the other methods.

For each test function (different dimensions are considered as different tests) $1,000$ trials from randomly generated points inside the domain are performed. The stopping criterion is the same for all the methods presented. We consider a local search unsuccessful if there is no global improvement for the objective function. After $1,000$ consecutive unsuccessful local searches we stop the algorithm. Every time a global improvement is

```
Data : $\Delta$, $N$
$n = 0$, $k = 0$
$x$ = random uniform point in $S$
$x_0 = x^\star = \mathcal{LS}(x)$
$record = L(x^\star)$
while $n < N$ do
    $y_k$ = random uniform point in $B(x_k, \Delta)$
    $z_k = \mathcal{LS}(y_k)$
    if $L(y_k) < record$ then
        $n = 0$
        $x^\star = x_{k+1} = z_k$
    else
        $n = n + 1$
    $k = k + 1$
```

**Algorithm 3**: MBH

obtained, we restart the count of unsuccessful local searches. The local searches are performed with the L-BFGS algorithm [9].

The results are given in Appendix A. Every test function is presented in a separate table. For each method we report the percentage of successes and the average number of local searches for each success. For trials without success the number of local searches for success is set to $\infty$. The average number of local searches for success is a measure of the computational effort needed for finding the global optimum.

The most notable difference between the trust-region scheme and ALSO is the number of local searches for success. For ALSO, this number is very sensitive to the initial trust-region choice and typically varies by one or two orders of magnitude. For instance, for Ackley ($n = 50$) it varies from 288 to $48,433$; for Levy ($n = 50$) from 47 to $1,412$; and for Schwefel ($n = 5$) from $4,056$ to $2,235$. The variance of the number of local searches for our method is much more modest. Clearly, the new trust-region method is far less sensitive to the choice of the initial radius.

We also observe that the new trust-region algorithm uses far fewer local searches than does ALSO. In some cases, ALSO uses up to 20 times the number of local searches used by our trust-region framework. This is an important improvement: it relates directly to the amount of time a user needs to wait for the global minimum.

At the same time, the total number of successes does not deteriorate with the new trust-region algorithm compared to ALSO. We suspect the reason is that ALSO is a nonmonotone method that continues to move the center of the trust-region even during unsuccessful iterations. In this way it appears to be more suitable for global exploration. We are experimenting with a new nonmonotone version of our method.

Comparing our results with AMBH, we conclude that, in general, we are more successful, but our algorithm seems to be less effective on Ackley and Levy. In particular, the number of local searches is in general larger. One reason for the better performances of AMBH is that the trust-region radius is chosen in such a way as to optimize the performances of MBH. Another reason is probably related to the different way of updating the radius: in AMBH the radius is updated by adding/subtracting a constant, whereas our algorithm multiplies/divides the trust-region radius. AMBH can find the optimal value of the trust-region radius for MBH more easily.

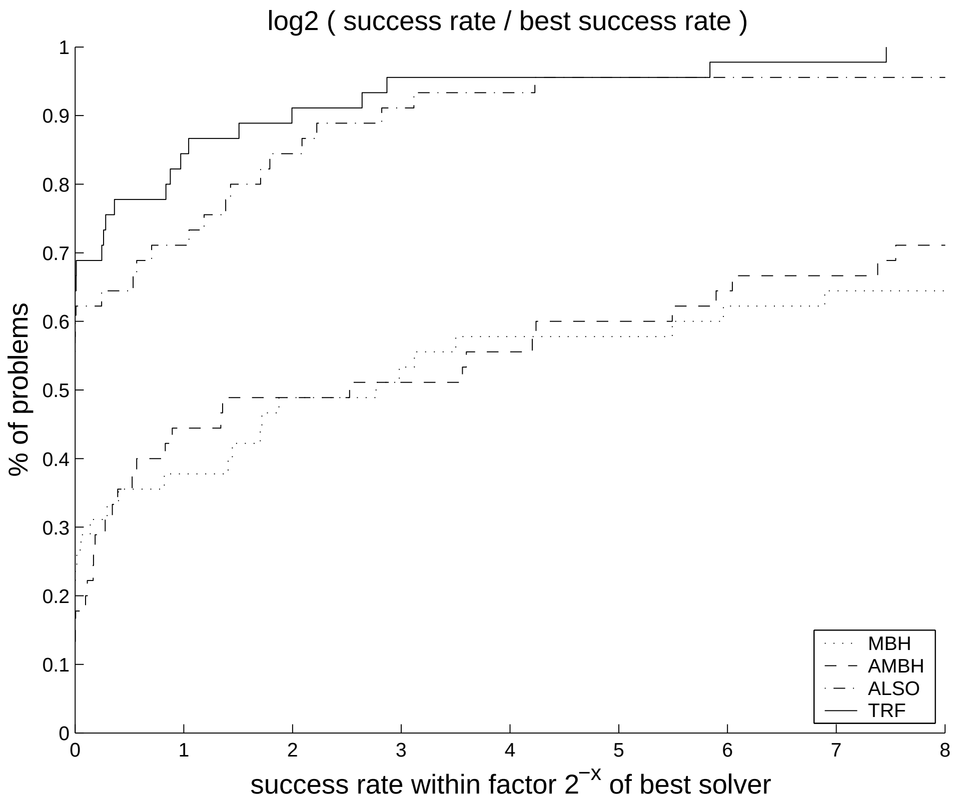

Figure 4: Performance profile of the success rate

The detailed results of Tables 1–10 are summarized in the performance profiles (see [4]) in Figures 4 and 5. The plots are generated by regarding every initial trust-region radius and every problem as a separate run. We form the ratio of the performance measure perf$(s, r)$ of solver $s$ on run $r$ divided by the best performance for problem $s$, and take its base 2 logarithm:

$$\log_2 \left( \frac{\text{perf}(s, r)}{\min\limits_s \text{perf}(s, r)} \right).$$

By sorting these ratios in ascending order for every solver, the resulting plot can be interpreted as the probability distribution that solver $s$ performs within a given multiple of the best solver.

Figure 4 compares the success rates of the four solvers. To apply the performance profile, we take the reciprocal of the success rates in Tables 1–10 and set 1/0 to $10^8$. In this way, the data is consistent with the performance profile idea that smaller values are better. We can interpret the plots in Figure 4 as a probability distribution that a solver has a success rate at worst a factor $2^{-x}$ of the best solver. We observe that ALSO and TRF are more robust and have better success rates than do AMBH and MBH. TRF is

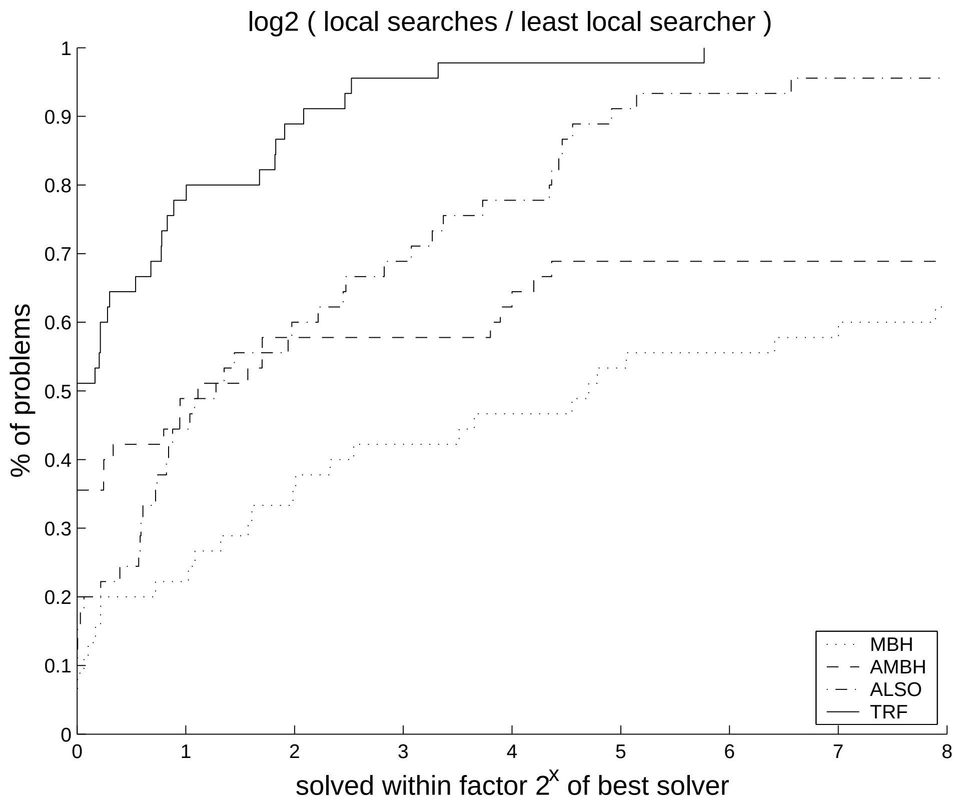

Figure 5: Performance profile of the average number of local searches

slightly more robust than ALSO, a result that shows that embedding ALSO within a trust-region framework did not deteriorate robustness of the solver.

In Figure 5 we compare the average number of local searches per successful solve. The plot represents a probability distribution that a solver solves a problem within a factor of at most $2^x$ of the fastest solve. The plot clearly shows that the new TRF outperforms all other solvers. We also observe that ALSO is faster than AMBH, which in turn is faster than MBH.

## 5 Conclusions

We presented a two-phase procedure for the global optimization of funnel functions. The approach builds on ALSO [2] and combines sampling with local searches. ALSO constructs a local smooth model from the samples by applying Gaussian smoothing. We demonstrated how to embed ALSO within a trust-region framework that adaptively updates the sample radius.

To extend the trust-region framework to global optimization, we introduced the concept of global quality, which triggers a model improvement step. Global quality measures the largest number of samples that have the same objective value and stem from the same basin of attraction. If global quality is large, then a model improvement step removes all but one sample from the largest set and generates a new set of uniform samples.

We compared our algorithm to ALSO and variants of monotone basin hopping (MBH). The new algorithm is more robust than ALSO and MBH on a range of test problems, and up to 20 times faster in terms of the number of local searches it requires per successful run.

## Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38. The first author was also supported by the Italian national research program "FIRB/RBNE01WBBBB Large Scale Nonlinear Optimization".

## References

[1] Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hillclimbing.* Kluwer Academic Publishers, Boston.

[2] Addis, B., Locatelli, M., and Schoen, F. (2003). *Local Optima Smoothing for Global Optimization.* Technical report DSI 5-2003, Dipartimento Sistemi e Informatica, Università di Firenze, Firenze, Italy.

[3] Conn, A. R., Gould N. and Toint, Ph. L. (2000). Trust-Region Methods. SIAM, Philadelphia.

[4] Dolan, E. D. and Moré, J. (2000). Benchmarking optimization software with COPS. Technical Report MCS-TM-246, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, November 2000.

[5] Doye, J. P. K. (2002). Physical perspectives on the global optimization of atomic clusters. In J. D. Pinter, editor, *Selected Case Studies in Global Optimization*, in press. Kluwer Academic Publishers, Dordrecht.

[6] Leary, R. H. (2000). Global optimization on funneling landscapes. *J. Global Optim.*, 18(4):367–383.

[7] Levenberg, K. (1944). A method for the solution of certain problems in least squares. Quart. Appl. Math., 2, 164–168.

[8] Levy, A., and Montalvo, A. (1985). The tunneling method for global optimization. *SIAM J. Sci. and Stat. Comp.*, 1:15–29.

[9] Liu, D., and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Math. Prog.*, B 45:503–528.

[10] Locatelli, M. (2003). On the multilevel structure of global optimization problems. To appear in *Computational Optimization and Applications.*

[11] Locatelli, M., and Schoen, F. (1999). Random linkage: A family of acceptance/rejection algorithms for global optimisation. *Math. Prog.*, 85(2):379–396.

[12] Moré, J. J., and Wu, Z. (1996). Smoothing techniques for macromolecular global optimization. In G. D. Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 297–312. Plenum Press, New York.

[13] Moré, J. J., and Wu., Z. (1997). Global continuation for distance geometry problems. *SIAM J. Optim.*, 7:814–836.

[14] Rinnooy Kan, A. H. G., and Timmer, G. (1987). Stochastic global optimization methods. Part II: Multilevel methods. *Math. Prog.*, 39:57–78.

[15] Schoen, F. (2001). Stochastic global optimization: Two phase methods. In C. Floudas and P. Pardalos, editors, *Encyclopedia of Optimization*, pages 301–305. Kluwer Academic Publishers, Dordrecht.

[16] Schoen, F. (2002). Two-phase methods for global optimization. In P. Pardalos and E. H. Romeijn, editors, *Handbook of Global Optimization Volume 2*, pages 151–178. Kluwer Academic Publishers, Dordrecht.

[17] Schwefel, H. P. (1981). *Numerical Optimization of Computer Models.* J. Wiley & Sons, Chichester.

[18] Shao, C. S., Byrd, R. H., Eskow, E., and Schnabel, R. B. (1997). Global optimization for molecular clusters using a new smoothing approach. In L. T. Biegler, T. F. Coleman, A. R. Conn, and F. N. Santosa, editors, *Large Scale Optimization with Applications: Part III: Molecular Structure and Optimization*, pages 163–199. Springer, New York.

[19] Törn, A., and Žilinskas, A. (1989). *Global Optimization.* Lecture Notes in Computer Sciences, vol. 350. Springer-Verlag, Berlin.

## A Tables of Numerical Results

Table 1: Rastrigin, $n = 20$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 1.0 | 0.0 | 0.6 | 100.0 | 77.8 | $\infty$ | 188833 | 1776 | 652 |
| 1.2 | 75.8 | 89.1 | 100.0 | 83.4 | 3020 | 1960 | 1079 | 602 |
| 1.4 | 99.8 | 92.5 | 100.0 | 84.3 | 510 | 916 | 475 | 577 |
| 1.6 | 98.2 | 87.3 | 98.2 | 80.8 | 596 | 987 | 513 | 591 |
| 1.8 | 32.1 | 15.2 | 73.9 | 87.5 | 3027 | 7711 | 864 | 519 |

Table 2: Rastrigin, $n = 50$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 1.8 | 0.0 | 0.0 | 99.0 | 48.0 | $\infty$ | $\infty$ | 6060 | 2952 |
| 2.0 | 83.0 | 49.2 | 91.4 | 46.6 | 2444 | 6459 | 2174 | 4363 |
| 2.2 | 94.5 | 74.5 | 58.0 | 51.5 | 1223 | 2644 | 2249 | 2173 |
| 2.4 | 18.7 | 4.2 | 15.2 | 49.6 | 12010 | 63048 | 10059 | 3940 |
| 2.6 | 0.0 | 0.0 | 2.7 | 50.6 | $\infty$ | $\infty$ | 59198 | 7026 |

Table 3: Levy, $n = 20$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 0.8 | 30.4 | 76.2 | 100.0 | 99.2 | 2900 | 34 | 451 | 120 |
| 1.0 | 98.9 | 82.5 | 100.0 | 99.3 | 1028 | 31 | 320 | 110 |
| 1.2 | 100.0 | 93.5 | 100.0 | 100.0 | 99 | 25 | 96 | 80 |
| 1.4 | 100.0 | 99.8 | 100.0 | 100.0 | 33 | 20 | 33 | 32 |

Table 4: Levy, $n = 50$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 1.0 | 26.8 | 38.4 | 36.5 | 98.3 | 755 | 60 | 1412 | 331 |
| 1.2 | 30.4 | 39.1 | 47.9 | 99.0 | 1330 | 51 | 1539 | 293 |
| 1.6 | 95.5 | 55.8 | 98.6 | 99.2 | 1240 | 45 | 970 | 169 |
| 2.0 | 100.0 | 97.1 | 100.0 | 100.0 | 47 | 22 | 47 | 45 |

Table 5: Ackley, $n = 20$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 1.0 | 0.0 | 88.0 | 100.0 | 100.0 | $\infty$ | 381 | 7844 | 654 |
| 1.4 | 100.0 | 99.6 | 100.0 | 100.0 | 793 | 374 | 791 | 693 |
| 1.8 | 100.0 | 100.0 | 100.0 | 100.0 | 293 | 346 | 293 | 292 |
| 2.2 | 100.0 | 100.0 | 100.0 | 100.0 | 274 | 345 | 275 | 274 |
| 3.5 | 11.5 | 100.0 | 69.1 | 100.0 | 7926 | 338 | 1329 | 578 |

Table 6: Ackley, $n = 50$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 1.4 | 0.0 | 67.5 | 99.8 | 100.0 | $\infty$ | 511 | 48433 | 2167 |
| 1.8 | 56.6 | 69.5 | 100.0 | 100.0 | 68965 | 538 | 19035 | 781 |
| 2.2 | 100.0 | 100.0 | 100.0 | 100.0 | 601 | 517 | 601 | 600 |
| 3.5 | 100.0 | 100.0 | 100.0 | 100.0 | 282 | 490 | 288 | 282 |
| 3.9 | 81.5 | 100.0 | 99.9 | 100.0 | 1013 | 498 | 654 | 613 |

Table 7: Schwefel, n=5

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 80 | 0.1 | 0.2 | 2.8 | 11.9 | 305582 | 3500 | 23714 | 1076 |
| 140 | 4.8 | 2.7 | 32.7 | 11.5 | 2688 | 667 | 1179 | 748 |
| 160 | 4.4 | 2.7 | 49.8 | 12.5 | 1953 | 778 | 981 | 656 |
| 220 | 9.8 | 4.1 | 77.4 | 10.6 | 1342 | 537 | 807 | 623 |

Table 8: Schwefel, n=10

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 160 | 0.1 | 0.1 | 1.3 | 4.5 | 751440 | 44000 | 22359 | 3156 |
| 220 | 0.0 | 0.1 | 6.6 | 3.7 | $\infty$ | 49000 | 5767 | 2378 |
| 280 | 0.3 | 0.1 | 18.7 | 3.0 | 30444 | 49000 | 4056 | 2667 |

Table 9: Scaled Rastrigin, $n = 20$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 0.6 | 0.0 | 0.0 | 57.2 | 1.0 | $\infty$ | $\infty$ | 7644 | 76400 |
| 0.8 | 0.0 | 0.0 | 52.8 | 0.3 | $\infty$ | $\infty$ | 4873 | 265333 |
| 1.0 | 0.0 | 0.0 | 0.3 | 1.4 | $\infty$ | $\infty$ | 444273 | 46143 |
| 1.2 | 0.0 | 0.0 | 0.6 | 5.2 | $\infty$ | $\infty$ | 202572 | 9981 |
| 1.4 | 0.0 | 0.0 | 3.1 | 8.1 | $\infty$ | $\infty$ | 34112 | 6148 |
| 1.6 | 0.0 | 0.0 | 5.8 | 13.2 | $\infty$ | $\infty$ | 20014 | 4303 |

Table 10: Scaled Rastrigin, $n = 50$

| | Percentage of Successes | | | | Average Local Searches | | | |
|---|---|---|---|---|---|---|---|---|
| $\Delta$ | MBH | AMBH | ALSO | TRF | MBH | AMBH | ALSO | TRF |
| 1.6 | 0.0 | 0.0 | 8.1 | 12.0 | $\infty$ | $\infty$ | 87546 | 59117 |
| 1.8 | 0.0 | 0.0 | 1.6 | 11.3 | $\infty$ | $\infty$ | 346525 | 63522 |
| 2.2 | 0.0 | 0.0 | 0.0 | 12.3 | $\infty$ | $\infty$ | $\infty$ | 44325 |
| 2.6 | 0.0 | 0.0 | 0.0 | 11.8 | $\infty$ | $\infty$ | $\infty$ | 49415 |