Local Optima Smoothing for Global Optimization

Bernardetta Addis[*] Marco Locatelli[†] Fabio Schoen[‡]

October 30, 2003

Abstract

It is widely believed that in order to solve large scale global optimization problems an appropriate mixture of local approximation and global exploration is necessary. Local approximation, if first order information on the objective function is available, is efficiently performed by means of local optimization methods. Unfortunately, global exploration, in absence of some kind of global information on the problem, is a "blind" procedure, aimed at placing observations as evenly as possible in the search domain. Often this procedure reduces to uniform random sampling (like in Multistart algorithms, or in techniques based on clustering).

In this paper we propose a new framework for global exploration which tries to guide random exploration towards the region of attraction of low-level local optima. The main idea originated by the use of smoothing techniques (based on gaussian convolutions): the possibility of applying a smoothing transformation not to the objective function but to the result of local searches seems to have never been explored yet. Although an exact smoothing of the results of local searches is impossible to implement, in this paper we propose a computational approximation scheme which has proven to be very efficient and (maybe more important) extremely robust in solving large scale global optimization problems with huge numbers of local optima.

[*]Email: b.addis@ing.unifi.it - Dip. Sistemi e Informatica - Università di Firenze
[†]Email: locatell@di.unito.it - Dip. Informatica - Università di Torino
[‡]Email: schoen@ing.unifi.it - Dip. Sistemi e Informatica - Università di Firenze

Keywords: global optimization, local search, basin-hopping, Multistart, random sampling, smoothing transformation

1 Introduction: two-phase and smoothing methods for global optimization

Let us consider a general, essentially unconstrained, global optimization problem of the form

$$f^\star = \min_{x \in S \subseteq \mathbb{R}^n} f(x)$$

where S is a "simple" subset of $\mathbb{R}^n$, typically an n–dimensional box, and f is a sufficiently smooth real-valued function. Here the feasible set is defined in order to have the possibility of sampling a uniform point in it without too much computational burden. In other words, it is expected that the global minimum $x^\star$ is attained in the interior of S (and most likely many, if not all, local minima are also in the interior): the feasible set is simply a search space wide enough to contain all of the "interesting solutions" to the problem. Also, by "sufficiently smooth" we mean that f should be at least as regular as required by an available local search algorithm.

Many problems of practical interest can be posed in these general terms: one of the most well-known and hard essentially unconstrained global optimization is the problem of folding a protein in such a way as to globally minimize its free energy: in this problem, atoms are relatively free to move in $\mathbb{R}^3$, even if a box can be defined which eventually will contain "interesting" molecular conformations.

If the problem allows for the use of a sufficiently efficient local optimization algorithm, it is well known [Schoen, 2002] that a two-phase procedure is a good candidate for global optimization. A two-phase approach is composed of sampling coupled with local searches started from sampled points. Let us denote by

$$\mathcal{LS}(x)$$

a mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$ such that $\bar{x} = \mathcal{LS}(x)$ is a local optimum point; then a two-phase global optimization method can be seen as a (random) sampling technique applied to the problem

$$\min_{x \in S} f(\mathcal{LS}(x)).$$

Ideally, if $\bar{x} = \mathcal{LS}(x)$ than x should belong to the basin of attraction of $\bar{x}$, in the sense that there should be a continuous non-increasing path connecting x with $\bar{x}$. When using any practical algorithm in place of $\mathcal{LS}$ this might not be true anymore – sometimes local searches perform very large steps, so that the set of points leading to the same local optimum may be very different from its region of attraction. But let us, for now, assume that an "ideal" local search is available. The composed function

$$L(x) = f(\mathcal{LS}(x))$$

is piecewise constant, with constant "plateaux" corresponding to the regions of attraction of different local optima. It should be evident that globally minimizing $f(x)$ is equivalent to globally minimizing the piecewise constant function $f(\mathcal{LS}(x))$. Multistart is an elementary example of a two-phase method aimed at minimizing $L(x)$ - in practice it reduces to a pure random (uniform) sampling applied to $L(\cdot) = f(\mathcal{LS}(\cdot))$. It could in principle be possible to apply any known global optimization to the transformed problem $\min L(x)$, but many problems arise. First of all, function evaluation becomes much more expensive – we have to perform a local search on the original problem to observe the transformed function L in a single point. Secondly, the analytical form of L is not available; moreover it is a discontinuous, piecewise constant function: even performing a local search on L is a difficult task, let alone finding a global minimum! Given this difficulty, most two-phase methods have been built without exploiting the fact that the true objective function to be minimized is L instead of f; some of the best known examples of two-phase methods, like e.g. multi-level single-linkage [Rinnooy Kan and Timmer, 1987], or simple-linkage clustering approaches [Locatelli and Schoen, 1999, Schoen, 2001] neglect in some sense the piecewise constant shape of L and concentrate most of the effort on improving with respect to Multistart. In particular, for clustering methods, improvement over Multistart is obtained through a sequential decision procedure which is used to choose from which points it seems worthwhile starting a local search. This way much of the effort is placed in sampling $f(x)$, while L is sampled in a relatively small number of points. This approach is sensible, and indeed produced very good results, only for problems with relatively few variables (a few tens) and with relatively few local minima whose regions of attraction are not too small: in these cases clustering methods are superior to Multistart, as they succeed in avoiding to start many local searches which eventually lead to the discovery of the same local minimum. Such kind of strategies are doomed

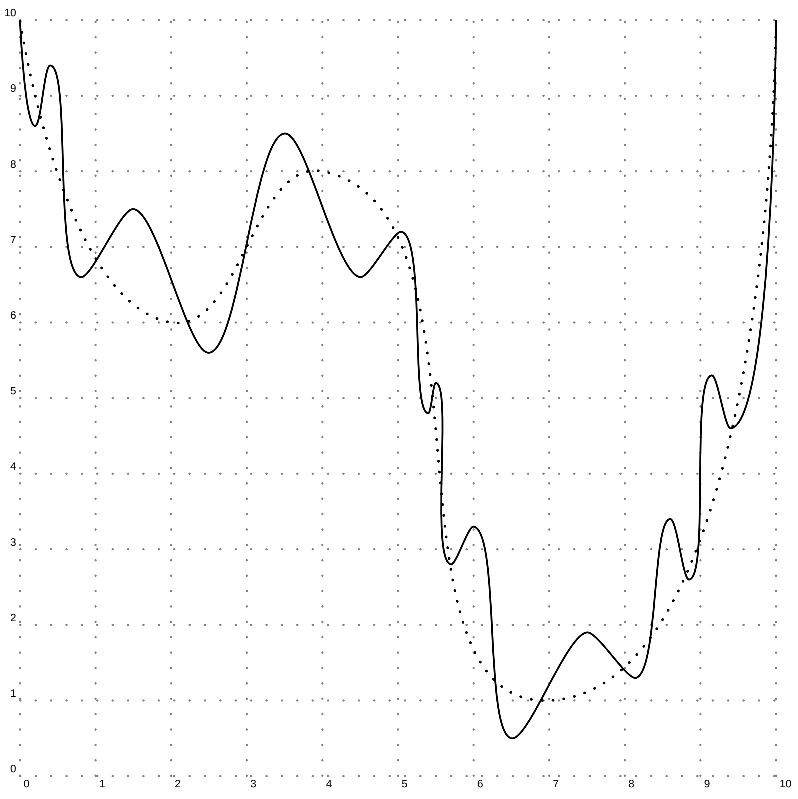

Figure 1: An example of a multimodal function

to fail when either the number of variables is high (and thus uniform sampling becomes excessively expensive) or when the number of local optima is huge, a situation which is extremely common, e.g., in most molecular conformation problems [Doye, 2002]. When this is the case, it frequently happens that local optima are not randomly displaced, but quite often they can be seen as perturbations of an underlying function which possesses quite a low number of "easy" local optima – in biology this kind of functions are referred to as "funnel"-like structures. An univariate example of such kind of functions is given in Figure 1, where the function to be minimized is represented by the solid line, while it can be seen that the underlying funnel structure is given by the dotted line. Motivated by examples of this kind, quite naturally, some authors [Moré and Wu, 1996, Moré and Wu., 1997, Shao et al., 1997] proposed filtering approaches: if we could filter the high frequencies which perturb the funnel structure, then it will be possible to 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 the authors' opinion, however, one of the main drawbacks of smoothing methods is the fact that they are based on the desire of directly smoothing $f(x)$, and, as it frequently happens, if this function is extremely oscillating, it may happen that either the smoothed function is very different from the original one, or, if the smoothing factor is small, the smoothed function is again multimodal, and optimizing it is as difficult and error-prone as optimizing the original one. In this paper we will try to exploit the interesting characteristics of smoothing methods for the optimization of $L(x)$.

2 Smoothing local searches

A quite elementary analysis of the above discussion led to the main contribution of this paper: in order to build a more reliable method for large-scale, funnel-type global optimization problems, sampling L coupled with smoothing L might prove a good strategy. If we look at Figure 2, it can be immediately noticed that a very good smoothing effect has already been achieved by simply observing L, the results of local searches, in place of f. However, in order to fully exploit the funnel structure, a smoothing method should be applied to L: this way the piecewise constant structure of L will be replaced by a smooth function which contains information of descent directions. It is in fact evident that in a piecewise constant function no local information is able to provide guidelines on descent steps; however, if smoothing were possible, the smoothed function could help in finding appropriate descent directions and thus guide the search towards the global optimum.

2.1 Theoretical framework

Given a real valued function $f : \mathbb{R}^n \to \mathbb{R}$ and a smoothing kernel $g : \mathbb{R} \to \mathbb{R}$, which is a continuous, bounded, non-negative, symmetric function whose integral is one, the g–transform of f is defined as

$$\langle f \rangle_g(x) = \int_{\mathbb{R}^n} f(y) g(\|y - x\|)\, dy.$$

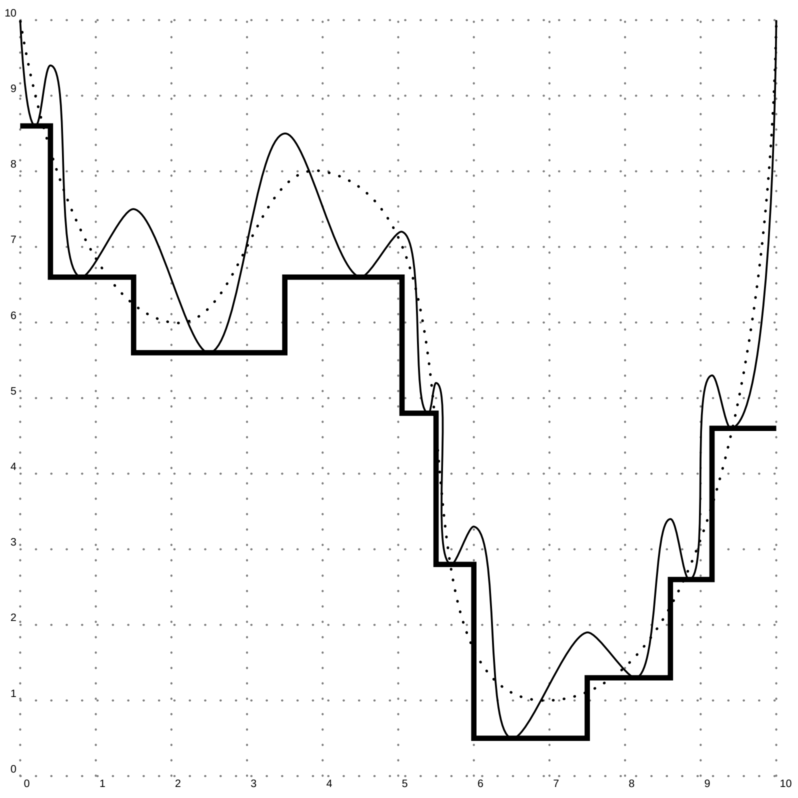

Figure 2: The effect of local optimization

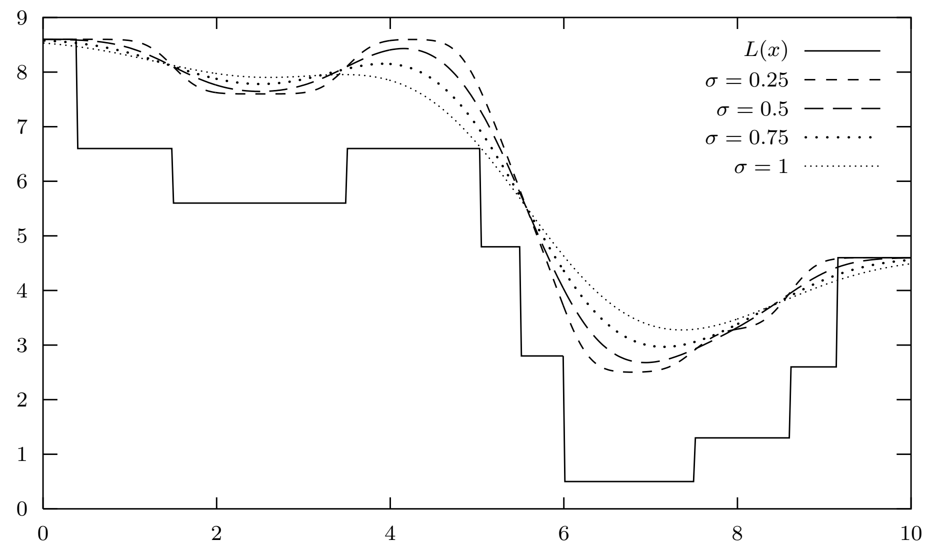

Figure 3: Gaussian filtering of $L(x)$

The most widely used kernel in the literature is of course the gaussian kernel

$$g(z) \propto \exp\left(-z^2/(2\sigma^2)\right)$$

but by no means this is the only one which has been used in practice (we used the symbol $\propto$ to avoid writing a multiplicative constant which plays no rôle in the methods we are presenting here).

In this paper we propose to apply a smoothing transformation to the piecewise constant function L and, thus, we would like to estimate the transformed function

$$\langle L \rangle_g(x) \quad = \quad \int_{\mathbb{R}^n} L(y) g(\|y - x\|)\, dy. \tag{1}$$

In Figure 3 we plot the piecewise constant result of local searches of Figure 2 with a few examples of gaussian transforms for different values of σ.

This task, apart from trivial examples, is usually an impossible one, as the analytical expression of L is not available. Nevertheless, it seems worthwhile to explore some of the main characteristics of this transform before proceeding with an approximation which eventually leads to the definition of a practical algorithm. Even if it is perfectly clear that an explicit computation of (1) is impossible, in this section we will analyze its behavior in the simplest possible case, i.e. that of a function of a single variable which possesses a single funnel.

In the one-dimensional case it may be assumed without loss of generality that the regions of attraction of different local optima are contiguous segments and thus

$$L(x) = \sum_{i=1}^{N} V_i \, \mathbf{1}_{x \in [a_{i-1}, a_i)} \tag{2}$$

where $a_0 = -\infty < a_1 < \cdots < a_{N-1} < a_N = +\infty$ and $V_i \in \mathbb{R}, i = 1, \ldots, N$. We observe that in the definition of $L(x)$ some care has to be taken for the value this function attains at points which are on the boundary between two different regions of attraction. Usually these are stationary points and a typical local algorithm will not make any step if initialized at such a point. In general it is not clear to which region of attraction these points should be assigned, if any. In the one-dimensional case we arbitrarily choose to leave the right hand side of each region of attraction open, but in the multidimensional case the question remains to be considered.

A single-funnel function is characterized by the fact that there exists an index $\ell \in \{1, \ldots, N\}$ such that

$$\begin{array}{ll} V_{i+1} < V_i & i = 1, \ldots, \ell - 1 \\ V_{i+1} > V_i & i = \ell, \ldots, N - 1. \end{array} \tag{3}$$

Notice that the global minimum value is V_ℓ. Observe also that (3) states the property that a descending (more precisely, not ascending) path down to the global minimum exists from any starting point. We can also assume, without loss of generality, that the origin $x = 0$ is the midpoint of the bottom step, i.e.

$$a_{\ell-1} = -a_\ell \tag{4}$$

(this can always be obtained through a translation).

Let $g(x)$ be a kernel (which, in particular, is a probability density function) and let $F(x) = \int_{-\infty}^{x} g(y)\, dy$ be the corresponding probability distribution function. In the following theorem we prove that if g satisfies quite general conditions (enjoyed, among others, by gaussian densities), then the transform L of a single–funnel step function is unimodal.

Theorem 1 *Let $g(x)$ be a continuously differentiable probability density function whose support is $\mathbb{R}$; then, if g is logarithmically concave, i.e. if*

$\log g(x)$ is concave, and if the step function L defined in (2) satisfies (3), then either $\langle L\rangle_g(x)$ is monotonic or it is unimodal.

Proof By definition of the transform, the value of $\langle L\rangle_g(x)$ is given by:

$$\langle L\rangle_g(x) = \sum_{i=1}^{N} V_i \int_{a_{i-1}}^{a_i} g(y-x)dy.$$

After the change of variable

$$t = y - x$$

we get

$$\langle L\rangle_g(x) = \sum_{i=1}^{N} V_i \int_{a_{i-1}-x}^{a_i-x} g(t)\,dt$$

and, after easy computations,

$$\langle L\rangle_g(x) = V_N - \sum_{i=1}^{N-1} (V_{i+1} - V_i) F(a_i - x).$$

It is immediate to proof that:

$$\lim_{x\to-\infty} \langle L\rangle_g(x) = V_1,\ \lim_{x\to+\infty} \langle L\rangle_g(x) = V_N$$

and that

$$\min_{i=1,N} V_i \le \langle L\rangle_g(x) \le \max_{i=1,N} V_i = \max\{V_1; V_N\}$$

so that the transform might be monotonic (increasing from V_1 to V_N if $V_1 < V_N$, or decreasing if $V_1 > V_N$), otherwise it must have at least a minimum point. We would like to show that stationary points are minima.

Taking the first derivative we obtain:

$$L'(x) := \frac{d}{dx} \langle L\rangle_g(x) = \sum_{i=1}^{N-1} (V_{i+1} - V_i) g(x - a_i). \tag{5}$$

We wish now to prove that $\langle L\rangle_g(x)$ is monotonic or unimodal by looking at the zeros of $L'(x)$. In fact

$$L'(x) = g(x) \sum_{i=1}^{N-1} (V_{i+1} - V_i) \frac{g(x - a_i)}{g(x)}$$

and its sign, as $g(x) > 0 \,\forall\, x \in \mathbb{R}$, is the same as that of

$$\sum_{i=1}^{N-1} (V_{i+1} - V_i) \frac{g(x-a_i)}{g(x)}.$$

We have:

$$\frac{d}{dx} \frac{g(x-a_i)}{g(x)} = \frac{g'(x-a_i)g(x) - g'(x)g(x-a_i)}{g^2(x)} \tag{6}$$

The denominator is always positive. The numerator is nonnegative if and only if

$$\frac{g'(x-a_i)}{g(x-a_i)} \geq \frac{g'(x)}{g(x)}.$$

If, as it has been assumed, g is logarithmically concave, then the first derivative of its logarithm

$$\frac{d}{dx} \log g(x) = \frac{g'(x)}{g(x)}$$

is nonincreasing; thus the numerator in (6) is nonnegative if and only if $a_i > 0$ and so

$$\frac{g(x-a_i)}{g(x)}$$

is nondecreasing if and only if $a_i > 0$; however, also $V_{i+1} - V_i$ is positive if and only if $a_i > 0$, so $L'(x)$ is the product of a nondecreasing function and a positive one. Thus either it is non zero for all x, or it can be zero either at a single point or it might be zero in a segment. In any case, as the transform either has no stationary points or it must have a minimum, the theorem is proven. $\square$

With slightly stronger assumptions we can obtain a transform which has one and only one minimum point.

Theorem 2 *If in addition to the assumptions of Theorem 1 $g(x)$ is* strictly *log-concave, then the transform has at most a single minimum point.*

Proof In fact strict log-concavity for a continuously differentiable function means that

$$\frac{d}{dx} \log g(x)$$

is decreasing; thus

$$\frac{g'(x)}{g(x)}$$

is decreasing and, as a consequence, $L'(x)$ is the product of a positive function and of an increasing one. Thus it can have at most one zero. $\square$

Theorem 3 *If, in addition to the assumptions of Theorem 1, g is such that*

$$\lim_{x\to-\infty}\frac{g'(x)}{g(x)}=+\infty,\ \lim_{x\to+\infty}\frac{g'(x)}{g(x)}=-\infty \tag{7}$$

then $L(x)$ has a minimum.

Proof We simply need to show that $L'(x)$ changes sign. Given the assumptions, thanks to the log-concavity of g, then

$$\log g(x-a)-\log g(x)\le -a\frac{g'(x)}{g(x)}$$

Thus, if $a>0$ the right hand side tends to $-\infty$ and we obtain that

$$\lim_{x\to-\infty}\frac{g(x-a)}{g(x)}=0.$$

Now,

$$\begin{aligned}
\frac{L'(x)}{g(x)} &= \sum_{i=1}^{N}(V_{i+1}-V_i)\frac{g(x-a_i)}{g(x)}\\
&= \sum_{i:a_i<0}(V_{i+1}-V_i)\frac{g(x-a_i)}{g(x)}+\sum_{i:a_i>0}(V_{i+1}-V_i)\frac{g(x-a_i)}{g(x)}\\
&\le \sum_{i:a_i<0}(V_{i+1}-V_i)\frac{g(a_N-a_i)}{g(a_N)}+\sum_{i:a_i>0}(V_{i+1}-V_i)\frac{g(x-a_i)}{g(x)}\\
&\to \sum_{i:a_i<0}(V_{i+1}-V_i)\frac{g(a_N-a_i)}{g(a_N)}\quad \text{for } x\to-\infty\\
&< 0
\end{aligned}$$

A similar reasoning can be followed to show that if $x\to+\infty$ then $L'(x)/g(x)$ tends to a strictly positive quantity. $\square$

From the above theorems, it immediately follows that, for example, if g is a gaussian kernel, then the transform has always one and only one minimum point. It is also easy to show that if the variance of the density function g is sufficiently small, then the minimum point occurs, as expected, inside the interval corresponding to the bottom step of the objective function:

Theorem 4 *There exists a $\bar{\sigma} > 0$ such that if*

$$Var(g) = \int_{-\infty}^{\infty} x^2 g(x)\, dx \leq \bar{\sigma}$$

then the minimum of $\langle L \rangle_g(x)$ belongs to $[a_{\ell-1}, a_\ell]$

Proof As the variance goes to zero, by Tchebychev's inequality, the probability tends to be concentrated at zero. So $\lim_{\bar{\sigma}\to 0} g(x) = 0$ for all $x \neq 0$. Thus

$$\lim_{\bar{\sigma}\to 0} L'(a_{\ell-1}) = (V_\ell - V_{\ell-1}) g(0) < 0$$

and

$$\lim_{\bar{\sigma}\to 0} L'(a_\ell) = (V_{\ell+1} - V_\ell) g(0) > 0.$$

$\square$

Being restricted to the one-dimensional case, all these results are of limited usefulness. However, they at least suggest the existence also for the multidimensional case of some 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.

The second requirement we need to relax is that of finding an analytic expression for $\langle L \rangle_g(x)$. This is usually not possible also because we do not even have an analytic expression for L. However, $\langle L \rangle_g(x)$ can be approximated and the local search over $\langle L \rangle_g(x)$ can be substituted by the local search over its approximation. Next section will be dedicated to the definition of an approximation and to the resulting global optimization algorithm.

3 An approximation scheme based on smoothing

Of course it is impossible to obtain an analytical expression of the transformed function $\langle L \rangle_g$; it is even impossible to obtain a numerical estimate, as the transform depends on values of L in all the domain: of course, apart from the impossibility of sampling L everywhere, even if this would be possible there would be no point in using a smoothing filter, as, at that point, the global optimum of the original would have been already discovered. So a complete description, or even a numerical estimate of $\langle L\rangle_g(x)$ at a single point is out of question. Considering the fact that only finite samples of L will be available, the approach we followed in this paper was that of sampling in a spherical neighborhood of prefixed radius r centered at the current point, i.e. in $B(x_0, r) = \{x : \|x - x_0\| \leq r\}$; after sampling, we build an approximation to the restriction of $\langle L\rangle_g(x)$ on $B(x_0; r)$ which should capture some local information of the function L. In other words, in some sense similar to what is done in trust-region methods, we use information and models in a neighborhood of the current point in order to find a descent direction. Assuming the ball is contained in the feasible set, the restriction of the transform on B can be defined as follows:

$$\langle L\rangle_g^B(x) = \frac{\int_{B(x_0,r)} L(y)g(\|y - x\|)\, dy}{\int_{B(x_0,r)} g(\|y - x\|)\, dy}. \tag{8}$$

We notice that the normalization factor is necessary in order to satisfy the requirements of a kernel function. In order to roughly estimate $\langle L\rangle_g^B(x)$ we choose to draw a uniform sample on $S(x_0, r)$ of a prefixed cardinality K. Let $y_1, \dots, y_K$ denote the points sampled in $S(x_0, r)$. Then a local estimate of the restricted transform is given by

$$\hat{L}_g^B(x) = \frac{\sum_{i=1}^K L(y_i)g(\|y_i - x\|)}{\sum_{i=1}^K g(\|y_i - x\|)}. \tag{9}$$

This function shares with (8) some important properties:

1. it is continuous
2. for each x it is a convex combination of the values obtained by applying local searches from points inside $B(x_0, r)$ (or, equivalently, obtained through observations of L)
3. the nearer x is to a sampled point y_i, the stronger the influence of the observed value is at that point.

Under mild regularity assumptions it can be seen that this estimate is consistent with (8), in the sense that the estimate converges to the restricted transform as K grows to infinity.

4 A numerical algorithm based upon smoothed local searches

At the beginning of this paper we observed that one of the disadvantages of working with $L(x)$ was the fact that, due to its piecewise constant shape, no information could be obtained on descent directions. Here we propose to use the local approximation of the smoothed transform of L given in (9) as a local model useful in guiding the search towards the global optimum. In other words, the local approximation is used as a model to predict a descent direction of $L(x)$.

The proposed algorithm consists of an approximation phase coupled with a displacement phase: in the approximation phase we are given a current point x_h and the value of $L(x_h)$; a sample of cardinality K of points in a neighborhood $S(x_h, r)$ of x_h is drawn and L is observed in each point in the sample. The values thus obtained are used to build an approximate filtered function (9) which is numerically minimized in $S(x_h, r)$. The global minimum of $\hat{L}_g^B(x)$ in $B(x_h, r)$ is taken as the next current point and the procedure is iterated (displacement phase). Of course, as local searches might produce very good local minima and, possibly, local minima which are very far from the current point, in order to speed up the procedure each time a record (i.e., the best local optimum observed so far) is obtained, the procedure interrupts and sets the current point to the record. The whole procedure is stopped when no improvement of the record has been observed since a prefixed number of steps. In the following we describe in pseudo-code the whole procedure:

```
**Procedure** LocSmoothGlobOpt($r$, MaxNoImprove, $K$)
    // *MaxNoImprove: stopping criterion*
    // *$K$: number of observations to perform in the current sphere*
    // *$r$: radius used in local perturbation of the current point*
    NoImprove = 0;
    $x$ = random uniform point in $S$;
    $x^\star = \mathcal{LS}(x)$;
    current = $f(x^\star) = L(x)$;
    record = current;
    **while** (NoImprove < MaxNoImprove)
        $i = 0$;
        while ($i < K$ and record $\leq$ current)
            $i = i + 1$;
            $y_i$ = random uniform point in $S(x^\star, r)$;
            $y_i^\star = \mathcal{LS}(y_i)$;
            current = $L(y_i)$;
        end while
        if (current < record)
            // a new record has been found while sampling in $S(x^\star, r)$
            record = current
            $x^\star = y_i^\star$
            NoImprove = 0;
        else
            NoImprove = NoImprove + $K$;
            // Major iteration: optimization of the
            //approximate smoothing based upon the observations placed in $y_1, \ldots, y_K$
            $x = \arg\min_{x \in S(x^\star, r)} \hat{L}_g^B(x)$;
            // The current point is moved
            $y = \mathcal{LS}(x)$
            current = $L(x)$;
            if (current < record)
                // a new record has been found
                record = current;
                NoImprove = 0;
                $x^\star = y$
            else
                $x^\star = x$
            end if
        end if
    end while
end Procedure
```

As it can be seen from the above scheme, in order to speed up the algorithm, the center of the ball where sampling is performed is moved as soon as a new record is observed.

5 Numerical results

In order to test the numerical performance of the proposed algorithm both a set of hard test problems and an alternative algorithm have to be chosen. We decided to compare our method with Monotonic basin-hopping (MBH) [Leary, 2000], [Wales and Doye, 1997], a widely used stochastic method which proved to be extremely efficient when applied to problems possessing a funnel-like structure. Using a terminology similar to that of the algorithm proposed in this paper, MBH can be described as follows:

```
Procedure MBH($r$, MaxNoImprove)
    // MaxNoImprove: stopping criterion
    // $r$: radius used in local perturbation of the current point
    NoImprove = 0;
    $x$ = random uniform point in $S$;
    $x^\star = \mathcal{LS}(x)$
    current = $f(x^\star) = L(x)$
    record = current;
    while (NoImprove < MaxNoImprove)
        $y$ = random uniform point in $S(x^\star, r)$
        $y^\star = \mathcal{LS}(y)$
        current = $L(y_i)$
        if (current < record)
            // a new record has been found while sampling in $S(x^\star, r)$
            record = current
            $x^\star = y^\star$
            NoImprove = 0;
        else
            NoImprove = NoImprove + 1
            end if
        end while
    end procedure
```

As it can be seen, MBH performs local optimization in a neighborhood of the current record, moving as soon as an improvement is observed; MBH has been very successful in many hard test problems of global optimization like, e.g., the minimization of Lennard-Jones or Morse potential energy functions. It seems to be the best algorithm to compare with our approach in that it shares with our method some parameters (and thus a comparison is more meaningful) and it performs extremely well on some test problems which are generally recognized as extremely difficult ones. We do not explore in this paper the possibility of applying our method to molecular conformation problems, as this subject deserves a deeper analysis which would excessively lengthen this paper.

As a set of test problems we choose a few objective functions which share the following characteristics:

- they are essentially unconstrained (defined in a box)
- the dimension of the problems can be chosen
- the global optimum is known
- they possess a very high number of local optima

Unfortunately a reliable and stable set of test problems is still lacking for unconstrained large scale global optimization. Even in [Floudas and Pardalos, 1999] the chapter on continuously differentiable unconstrained problems contains very few (and quite easy) general tests. The problems we solved were the following

Rastrigin [Törn and Žilinskas, 1989]

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

with $x_i \in [-5.12, 5.12]$

Levy [Levy and Montalvo, 1985]

$$Levy(x) = 10\sin^2(\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2(1 + 10\sin^2(\pi x_{i+1})) + (x_n - 1)^2 \tag{11}$$

with $x_i \in [-10, 10]$

Ackley [Ackley, 1987]

$$Ack(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(2\pi x_i)) \tag{12}$$

with $x_i \in [-32.768, 32.768]$

Schwefel [Schwefel, 1981]

$$Sch(x) = \sum_{i=1}^{n} -x_i \sin(|x_i|) \tag{13}$$

with $x_i \in [-500, 500]$

Moreover we performed tests on an amplified version of the Rastrigin function

$$AmplRas(x) = 10n + \sum_{i=1}^{n} x_i^2 - A\cos(2\pi x_i) \tag{14}$$

with $A = 100$ and with $A = 1000$; this function was chosen in order to test the sensitivity of the method to larger oscillations.

Finally we choose to test our method also on a scaled Rastrigin function:

$$ScaledRas(x) = 10n + \sum_{i=1}^{n} (\alpha_i x_i)^2 - 10\cos(2\pi(\alpha_i x_i)) \tag{15}$$

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

In the following tables we report computational results obtained with our algorithm and with MBH with different choices of the parameters. For each test we performed 1000 independent runs of each algorithm; for each test performed either with our method or with MBH on the same function but with different parameters, we used the same seeds in random number generation, so that each of the 1000 runs was started in the same point for all the methods tested. In both methods we used the value 1000 for MaxNoImprove parameter. We choose a gaussian kernel for the smoothing and we let its standard deviation be defined as

$$\sigma = rK^{-1/n}$$

This choice for σ originates from the desire, in some sense, to have enough information from the observations in all the neighborhood $B(x; r)$. In other words, assuming that, using a gaussian kernel, enough weight is placed in a ball of radius σ centered in each observation, the above formula is obtained from the requirement that the volume covered by K balls of radius σ should suffice to cover the ball with radius r.

This way the only parameters which remains to be chosen are r, the radius of the sphere used both for generating new observations of L and for the local approximation based upon smoothing, and K, the number of observation used in order to build an approximation to the smoothed function.

For what concerns MBH, we tested several values of r in order to find the best value for that algorithm. Our main aim in these experiments has been that of showing that even when a very good algorithm, like MBH, is placed in the best possible conditions, our method still outperforms it.

The tables with numerical results can be found in the appendix; for each table we report the following columns: the radius r used, the number K of samples, the number of successes out of 1000 trials (by success we mean that the algorithm observed the global optimum at least once), the average number of local searches performed (obtained dividing the total number of local searches *except those used for stopping* by the number of trials), and the average number of local searches per success (obtained dividing the total number of local searches, again except the last ones, by the number of successes, if this number is positive). We choose not to count the last local searches as these are constant for both algorithms and, in some sense, they are just a waste, as they are just used only to stop the algorithm. Results relative to MBH are reported in the table with a dash "-" in column K.

5.1 Comments on numerical results

The results obtained with the Rastrigin function (see tables 1–3) show a clear improvement in our method with respect to MBH. Even when the best parameter r is chosen for MBH, our method generally outperforms it; however it is important to notice that not only our algorithm requires less local searches than MBH: it has also a significantly greater success rate and, what is particularly important, keeps its good behavior for a quite large set of choices of the parameter r: in other words not only the method is better than MBH, but it is also much more robust, so that the choice of parameters is far less critical. This greater robustness of our algorithm is shared by most of our tests, as it can be seen from the tables in the appendix.

Analyzing the behavior of the two methods with Levy's and Ackley's functions (Tables 4–6 and 7–9), again it is immediate to see that our method has a greater success rate, requires less local searches and is more robust than

MBH; differently from what happened with Rastrigin test functions, here increasing the cardinality K of the sample did not improve the performance of our method. This difference may be justified observing that increasing K has two opposite effects: from one side it gives us a better approximation of the smoothed function; from another side, however, as we choose to stop the algorithm after 1 000 local searches with no improvement, a greater value of K gives our algorithm less freedom to move. For example, with $K = 50$ the algorithm might stop after 20 "major" iterations (i.e. those iterations during which a new approximate smoothing is built and used) with no improvement; increasing K to 100, leaves to our method only 10 unsuccessful iterations before stopping. For easy functions, like Rastrigin, the advantages of better approximation compensate the small number of moves; for more challenging tests this is no more true. However, analyzing the output of our algorithm, we noticed that in any case the algorithm almost always stops with a record value which is significantly lower than that found by MBH even in the cases were failure occurs. Another observation should be made for these two classes of test functions: looking at the tables it is seen that when both MBH and our method display the best performance, their behavior is almost indistinguishable. In these cases our method is actually following the same decisions as MBH: in practice before a sample of K observations is taken a new record is almost always observed. In these cases, no approximation is performed and the method becomes the same as MBH (with no overhead). However, again, we observe that when the parameters of MBH are not optimal, our method still outperforms it. In other words, our method is equal to MBH when it is easy to follow descent steps towards the global optimum; instead, when MBH gets stuck in a local optimum, our algorithm often finds a descent path which improves the record of MBH and, quite often, ends up in the global optimum.

The Schwefel test function is particularly interesting as, differently from the previous ones, it is not a single-funnel one. So both methods might, and in fact do, end up in different optima. Again, however, it can be easily seen that our method is significantly more successful and efficient than MBH, the more so when the dimension increases. When r is sufficiently large, the identification of the funnel containing the global optimum becomes quite easy.

The modification introduced on the Amplified Rastrigin function (see the results in Tables 12–14), while not altering significantly the regions of attraction of local optima, changes their relative value; thus it is expected that global optimization methods will get into more trouble in finding a descent path towards the global optimum. In fact the results, both for MBH and for our method, are worse than those found on the original Rastrigin function. However, again, our method, for all choices of the parameters, with a single exception, outperforms MBH both in the number of successes and in the number of local searches required to observe each success.

Finally we analyze the scaled variant of the Rastrigin function (tables 15–16). The motivation for this modification is that we wished to check the ability of our method of finding descent paths even when the level sets near local optima are not spherical. Changing the scale of some of the variables has the effect of generating ellipsoidal level sets. It is thus quite evident that a method like MBH, which samples in spheres, will have great difficulties: in fact if the radius of the sphere is too small, MBH will not find an improvement and will get stuck in a local minimum. On the other hand, if the radius is too large, sampling will be performed in a region which is so large that the probability of finding a record will usually be very low. Quite surprisingly, however, our method is able to correctly identify descent directions, even when sampling in small spheres, as it is clearly displayed by the results in the tables. Of course the higher the dimension, the worse is the behavior of the algorithm; however the number of successes is quite high. We also performed a final test to check whether the number of successes could be improved by letting the algorithm run longer; we thus increased the MaxNoImprove parameter from 1000 to 2000 (see Table 17) and in fact observed a significant improvement: this reinforces our suspects that our algorithm successfully extracts information of good descent direction which enables it, if not stopped prematurely, to end up in the global minimum with high probability.

6 Conclusions

The numerical results we have presented support the evidence that our method is significantly efficient and robust. In fact, not only it is able to find the global optimum generally more often than MBH, with a much lower number of local searches and, what we think is a particularly interesting characteristics, with far less tuning of parameters. It can be seen from most of the tables that the behavior of our method is sensibly better than that of basin hopping for many choices of the two parameters of the algorithm, even when compared with the best choice for MBH. Of course in the tables presented we did not consider the overhead caused by the use and the optimization of the approximate transform. This is indeed a computational cost which other algorithms, and, in particular, MBH, do not possess; also we did not take into account the fact that while MBH is a memoryless method, in which only the current and the best observation have to be memorized, in our method $K + 1$ observations are to be kept in memory. Of course we could pretend that the method is applied to the optimization of computationally expensive objective functions, in which case the overhead will be negligible and surely counterbalanced by the increase in efficiency and robustness of the method. However, even in the case of relatively cheap objective functions, like the ones we used as test problems in this paper, the actual overhead is quite modest. It should in fact be observed that in the main loop of our algorithm a new approximation, with the necessity of storing K new observations and of optimizing the approximate transform, is generated every K iterations; but, as it can be seen from the tables dividing the number of local searches by $K+1$, the number of approximations required is generally extremely low. We should also remark that, although the approximate transform might be nonconvex in the neighborhood of the current point, we just performed a simple local search on it, based upon the assumption that a descent in the transform most likely corresponds to a descent in the original function. Although this is not true in general, in practice the results obtained tend to confirm that this is quite often the case.

In conclusion we can observe from the numerical experiments that our approach succeeds in extracting information on possible descent directions in the piecewise constant function which is obtained through local searches. The results display a significative improvement both in the precision (number of successes) and in the computational cost (number of local searches) with respect to basin-hopping, which is one of the best performing methods for problems with huge numbers of local optima. Moreover, the proposed approach is almost parameter free: we have proposed an automatic procedure for the choice of the σ parameter in the definition of the gaussian transform; all of our results (except one) were obtained with a fixed value for the MaxNoImprove parameter, and the size of the sample within each sphere was chosen either equal to n, the dimension of the problem, or to $2n$. So the only parameter which requires specification is r, but, as it can be seen from the tables, choosing r in our method is sensibly much easier than in basin hopping. Quite large ranges for r produce numerical results which are significantly better than those obtained with the best possible choice for basin hopping. We can conclude observing that in this paper a new approach has been proposed which tries to exploit the important information obtained through local searches by means of a smoothing technique aimed at discovering regions where it is likely to find better function values. The numerical results obtained are quite encouraging, as they support the fact that the proposed approach is robust, requires the setting of a very small number of parameters and is capable of finding the global optima of standard difficult test problems in quite an efficient way.

References

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

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

[Floudas and Pardalos, 1999] Floudas, C. A. and Pardalos, P. M. (1999). *Handbook of Test Problems in Local and Global Optimization*, volume 33 of *Nonconvex Optimization and its Applications.* Kluwer Academic Publishers, Dordrecht.

[Leary, 2000] Leary, R. H. (2000). Global optimization on funneling landscapes. *Journal of Global Optimization*, 18(4):367–383. .

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

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

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

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

[Rinnooy Kan and Timmer, 1987] Rinnooy Kan, A. H. G. and Timmer, G. (1987). Stochastic global optimization methods. Part II: Multi level methods. *Mathematical Programming*, 39:57–78.

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

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

[Schwefel, 1981] Schwefel, H. P. (1981). *Numerical Optimization of Computer Models.* J. WIley & Sons, Chicester.

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

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

[Wales and Doye, 1997] Wales, D. J. and Doye, J. P. K. (1997). Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *Journal of Physical Chemistry A*, 101:5111–5116.

A Tables of numerical results

Table 1: Rastrigin, $n = 20$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.0	-	0	2330.770	∞
1.2	-	758	2289.410	3020.330
1.4	-	998	509.751	510.773
1.6	-	982	586.033	596.775
1.8	-	321	971.671	3027.012
1.0	20	1000	1776.600	1776.600
1.2	20	1000	1079.330	1079.330
1.4	20	1000	475.290	475.290
1.6	20	982	504.462	513.709
1.8	20	739	639.072	864.779
1.0	40	1000	3053.100	3053.100
1.2	40	1000	1432.960	1432.960
1.4	40	1000	468.052	468.052
1.6	40	1000	422.505	422.505
1.8	40	986	564.754	572.773

Table 2: Rastrigin, $n = 30$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.2	-	0	543.057	∞
1.4	-	89	5924.310	66565.281
1.6	-	972	1054.070	1084.434
1.8	-	983	747.692	760.623
2.0	-	289	1440.620	4984.844
1.2	30	1000	3515.280	3515.280
1.4	30	1000	2470.310	2470.310
1.6	30	1000	961.410	961.410
1.8	30	932	829.191	889.690
2.0	30	520	956.656	1839.723
1.2	60	907	6866.680	6866.680
1.4	60	1000	3779.340	3779.340
1.6	60	1000	984.847	984.847
1.8	60	1000	698.382	698.382
2.0	60	958	947.355	988.888

Table 3: Rastrigin, $n = 50$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.8	-	0	10116.300	∞
2.0	-	830	2028.960	2444.530
2.2	-	945	1156.560	1223.873
2.4	-	187	2245.920	12010.267
2.6	-	0	2187.590	∞
1.8	50	990	6000.190	6060.798
2.0	50	914	1987.440	2174.442
2.2	50	580	1304.740	2249.552
2.4	50	152	1529.020	10059.342
2.6	50	27	1598.350	59198.148
1.8	100	966	9876.560	10224.182
2.0	100	994	1999.290	2011.358
2.2	100	967	1238.560	1280.827
2.4	100	768	1734.090	2257.930
2.6	100	377	1804.170	4785.597

Table 4: Levy, $n = 20$

r	K	number of successes	Average number of LS	Average number of LS for each success
0.8	-	304	881.788	2900.618
1.0	-	989	1017.500	1028.817
1.2	-	1000	99.958	99.958
1.4	-	1000	33.629	33.629
0.8	20	1000	451.896	451.896
1.0	20	1000	320.191	320.191
1.2	20	1000	96.231	96.231
1.4	20	1000	33.621	33.621
0.8	40	957	1938.340	2025.434
1.0	40	1000	586.754	586.754
1.2	40	1000	99.717	99.717
1.4	40	1000	33.629	33.629

Table 5: Levy, $n = 30$

r	K	number of successes	Average number of LS	Average number of LS for each success
0.8	-	224	231.219	1032.228
1.0	-	346	801.464	2316.370
1.2	-	868	1552.890	1789.044
1.4	-	1000	163.874	163.874
0.8	30	885	2524.580	2852.633
1.0	30	987	1585.450	1606.332
1.2	30	1000	716.133	716.133
1.4	30	1000	158.480	158.480
0.8	60	330	719.213	2179.433
1.0	60	603	1192.370	1977.396
1.2	60	934	1242.220	1330.000
1.4	60	1000	162.173	162.173

Table 6: Levy, $n = 50$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.0	-	268	202.455	755.429
1.2	-	304	404.545	1330.740
1.4	-	499	941.807	1887.389
1.6	-	955	1184.410	1240.220
1.8	-	1000	146.909	146.909
2.0	-	1000	47.699	47.699
1.0	50	365	515.736	1412.975
1.2	50	479	737.640	1539.958
1.4	50	694	998.887	1439.318
1.6	50	986	957.039	970.628
1.8	50	1000	144.947	144.947
2.0	50	1000	47.669	47.669
1.0	100	346	287.712	831.538
1.2	100	451	572.906	1270.302
1.4	100	655	956.228	1459.890
1.6	100	969	1069.500	1103.715
1.8	100	1000	146.821	146.821
2.0	100	1000	47.699	47.699

Table 7: Ackley, $n = 20$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.0	-	0	452.045	∞
1.4	-	1000	793.112	793.112
1.8	-	1000	293.625	293.625
2.2	-	1000	274.034	274.034
3.5	-	115	911.591	7926.878
1.0	20	1000	7844.820	7844.820
1.4	20	1000	791.688	791.688
1.8	20	1000	293.625	293.625
2.2	20	1000	275.778	275.778
3.5	20	691	918.841	1329.726
1.0	40	1000	13044.100	13044.100
1.4	40	1000	793.143	793.143
1.8	40	1000	293.625	293.625
2.2	40	1000	274.077	274.077
3.5	40	890	814.533	915.206

Table 8: Ackley, $n = 30$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.0	-	0	1.843	∞
1.4	-	999	22459.200	22481.682
1.8	-	1000	505.210	505.210
2.2	-	1000	303.604	303.604
3.5	-	92	1042.250	11328.804
1.0	30	1000	20708.200	20708.200
1.4	30	1000	8385.500	8385.500
1.8	30	1000	505.210	505.210
2.2	30	1000	303.604	303.604
3.5	30	643	1038.170	1614.572
1.0	60	999	46343.900	46390.290
1.4	60	1000	13323.800	13323.800
1.8	60	1000	505.210	505.210
2.2	60	1000	303.604	303.604
3.5	60	912	920.429	1009.242

Table 9: Ackley, $n = 50$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.4	-	0	1	∞
1.8	-	566	39034.400	68965.371
2.2	-	1000	601.171	601.171
3.5	-	1000	282.960	282.960
3.9	-	815	826.145	1013.675
1.4	50	998	48336.400	48433.267
1.8	50	1000	19035.400	19035.400
2.2	50	1000	601.171	601.171
3.5	50	1000	288.402	288.402
3.9	50	999	654.000	654.655
1.4	100	0	61893.600	∞
1.8	100	1000	36007.000	36007.00
2.2	100	1000	601.171	601.171
3.5	100	1000	283.198	283.198
3.9	100	998	687.483	688.861

Table 10: Schwefel, $n = 5$

r	K	number of successes	Average number of LS	Average number of LS for each success
80	-	1	305.582	305582.000
100	-	4	417.661	104415.250
120	-	30	414.768	13825.600
140	-	48	129.044	2688.417
160	-	44	85.962	1953.682
180	-	64	92.103	1439.109
200	-	78	123.072	1577.846
220	-	98	131.597	1342.827
80	5	28	664.001	23714.321
100	5	46	303.967	6607.978
120	5	144	277.104	1924.333
140	5	327	385.778	1179.749
160	5	498	488.855	981.637
180	5	643	598.035	930.070
200	5	716	639.573	893.258
220	5	774	624.771	807.198
80	10	14	494.673	35333.786
100	10	39	314.438	8062.513
120	10	103	206.553	2005.369
140	10	213	252.764	1186.685
160	10	315	339.350	1077.302
180	10	393	390.877	994.598
200	10	476	428.535	900.284
220	10	591	513.847	869.453

Table 11: Schwefel, $n = 10$

r	K	number of successes	Average number of LS	Average number of LS for each success
160	-	1	751.440	751440.000
180	-	2	309.388	154694.000
200	-	0	122.818	∞
220	-	0	92.398	∞
240	-	0	90.421	∞
260	-	5	95.169	19033.800
280	-	3	91.332	30444.000
160	10	13	290.672	22359.385
180	10	25	287.336	11493.440
200	10	46	350.104	7610.957
220	10	66	380.625	5767.045
240	10	106	489.975	4622.406
260	10	149	650.252	4364.107
280	10	187	758.530	4056.310
160	20	4	257.841	64460.250
180	20	7	222.044	31720.571
200	20	18	189.065	10503.611
220	20	11	184.894	16808.545
240	20	36	231.097	6419.361
260	20	44	268.530	6102.955
280	20	63	341.827	5425.825

Table 12: AmpRas, $n = 20, A = 100$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.2	-	609	3928.250	6450.328
1.4	-	997	756.266	758.542
1.6	-	930	775.829	834.225
1.8	-	210	1094.700	5212.857
1.2	20	1000	1366.730	1366.730
1.4	20	1000	645.803	645.803
1.6	20	984	604.844	614.679
1.8	20	736	708.853	963.115
1.2	40	1000	1994.530	1994.530
1.4	40	1000	663.330	663.330
1.6	40	1000	508.582	508.582
1.8	40	974	628.643	645.424

Table 13: AmpRas, $n = 50, A = 100$

r	K	number of successes	Average number of LS	Average number of LS for each success
2.0	-	449	4213.260	9383.653
2.2	-	644	2135.190	3315.512
2.0	50	878	3514.180	4002.483
2.2	50	506	1925.200	3804.743
2.0	100	997	3827.090	3838.606
2.2	100	935	1962.490	2098.920

Table 14: AmpRas, $n = 50, A = 1000$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.8	-	0	8628.000	∞
2.0	-	407	2319.260	5698.428
2.2	-	587	4694.890	7998.109
2.4	-	19	2786.280	146646.316
1.8	50	981	8013.560	8168.767
2.0	50	869	3802.530	4375.754
2.2	50	530	2041.780	3852.415
2.4	50	131	1937.610	14790.916
1.8	100	980	15553.900	15871.327
2.0	100	994	4186.680	4211.952
2.2	100	929	2071.080	2229.365
2.4	100	668	2163.830	3239.266

Table 15: ScaledRas, $n = 20$

r	K	number of successes	Average number of LS	Average number of LS for each success
0.6	-	0	379.446	∞
0.8	-	0	204.054	∞
1.0	-	0	1536.690	∞
1.2	-	0	2633.660	∞
1.4	-	0	1349.760	∞
1.6	-	0	1076.680	∞
1.8	-	0	1004.370	∞
0.6	20	572	4372.460	7644.161
0.8	20	528	2572.770	4872.670
1.0	20	3	1332.820	444273.333
1.2	20	6	1215.430	202571.667
1.4	20	31	1057.460	34111.613
1.6	20	58	1160.840	20014.483
1.8	20	104	1325.110	12741.442
0.6	40	179	6358.290	35521.173
0.8	40	574	4063.510	7079.286
1.0	40	1	1747.470	1747470.000
1.2	40	1	1013.720	1013720.000
1.4	40	1	481.011	481011.000
1.6	40	3	555.833	185277.667
1.8	40	6	739.986	123331.000

Table 16: ScaledRas, $n = 50$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.6	-	0	297.031	∞
1.8	-	0	6470.100	∞
2.0	-	0	3456.610	∞
2.2	-	0	2256.010	∞
2.4	-	0	2135.750	∞
2.6	-	0	2267.390	∞
1.6	50	81	7091.190	87545.556
1.8	50	16	5544.400	346525.000
2.0	50	0	3287.120	∞
2.2	50	0	2271.110	∞
2.4	50	0	1875.970	∞
2.6	50	0	1647.570	∞
1.6	100	174	12380.300	71151.149
1.8	100	70	8883.980	126914.000
2.0	100	15	4586.400	305760.000
2.2	100	0	3002.430	∞
2.4	100	1	2392.600	2392600.000
2.6	100	0	2113.750	∞

Table 17: ScaledRas, $n = 50, \mathrm{MaxNoImprove} = 2\,000$

r	K	number of successes	Average number of LS	Average number of LS for each success
1.6	-	0	4004.450	∞
1.8	-	0	11155.900	∞
2.0	-	0	5282.770	∞
2.2	-	0	3686.780	∞
2.4	-	0	3644.540	∞
2.6	-	0	3788.660	∞
1.6	50	196	7673.810	39146.990
1.8	50	42	6174.080	146978.100
2.0	50	3	3947.490	1315497.000
2.2	50	0	2940.880	∞
2.4	50	0	2552.840	∞
2.6	50	0	2273.280	∞
1.6	100	416	13157.600	31626.440
1.8	100	154	9506.450	61723.700
2.0	100	34	5224.780	153640.600
2.2	100	1	3601.460	3600460.000
2.4	100	1	3024.360	3023360.000
2.6	100	0	2759.240	∞