J Glob Optim (2010) 48:73–85
DOI 10.1007/s10898-009-9519-7

A local search method for continuous global optimization

M. Gaviano · D. Lera · A. M. Steri

Received: 14 December 2009 / Accepted: 18 December 2009 / Published online: 10 January 2010

Abstract A large number of algorithms introduced in the literature to find the global minimum of a real function rely on iterative executions of searches of a local minimum. Multistart, tunneling and some versions of simulated annealing are methods that produce well-known procedures. A crucial point of these algorithms is to decide whether to perform or not a new local search. In this paper we look for the optimal probability value to be set at each iteration so that by moving from a local minimum to a new one, the average number of function evaluations *evals* is minimal. We find that this probability has to be 0 or 1 depending on the number of function evaluations required by the local search and by the size of the level set at the current point. An implementation based on the above result is introduced. The values required to calculate *evals* are estimated from the history of the algorithm at running time. The algorithm has been tested both for sample problems constructed by the *GKLS* package and for problems often used in the literature. The outcome is compared with recent results.

Keywords Random search · Global optimization

1 Introduction

In this paper we consider the following global optimization problem

Problem 1.1

$$\text{find } x^* \in S, \quad \text{such that } f(x^*) \leq f(x), \quad \forall x \in S,$$

where $f : S \to \mathbf{R}$ is a function defined on a set $S \subseteq \mathbf{R}^n$.

M. Gaviano · D. Lera (✉) · A. M. Steri
Dipartimento di Matematica e Informatica, Università di Cagliari, Cagliari, Italy
e-mail: lera@unica.it

M. Gaviano
e-mail: gaviano@unica.it

A. M. Steri
e-mail: marissteri@yahoo.it

Many applied problems in engineering and economics can be solved by the above mathematical formulation; that is, it is required to find a global optimizer of a non linear function defined on a given set of points.

In the literature, there exist many methods for resolution of the Problem 1.1: deterministic approaches based on the partition of the domain (see [1–3]), stochastic approaches (see [4]), or techniques based on the reduction of the dimension (see Strongin and Sergeyev [3]). The algorithms based on local search strategies to solve 1.1, exploit local minimization algorithms whose up to date procedures are very efficient in finding a local minimum. A local solver algorithm **A** usually constructs, by starting from a point x_0, a sequence $\{x_j\}$, $f(x_j) > f(x_{j+1})$, that converges to a local minimum x^*. The starting point x_0 can be chosen uniformly at random in S or according to a probability distribution based on the algorithm running time history. The value $f(x^*)$ is the global minimum in the so-called *region of attraction* of x^* for **A**; that is, the *region of attraction* is the set of all points that chosen as starting points of **A**, enable us to find x^*. Clearly if we start from different points in S we are able to find all the local minima of $f(\cdot)$ and then to get its global minimum. The researchers have designed and investigated different strategies for choosing the starting points of the local searches; see the papers by Boender et al. [5], Cetin et al. [6], Desai et al. [7], Hedar et al. [8], Levy et al.[9], Lucidi et al. [10], Oblow [11].

The main point to tackle whenever local search strategies are used to find a global minimum, is to avoid finding the same local minima. One can choose the starting point x_0 of a new local search such that the function value $f(x_0)$ is less than the value of the last local minimum found. In such a way the local searches guarantee that a new local minimum point with function value less than the previous ones can be found. On the other hand, by proceeding in this way, we reduce the size of the region that could take us to the global minimum.

In this paper we first consider an algorithm scheme that given a local minimum x_i, chooses uniformly at random in S a point x_0 and then, starting from it, carries on a new local search both if $f(x_0)$ is less than $f(x_i)$ or if $f(x_0)$ is greater than $f(x_i)$, but in the latter case according to a probability d_i. Then, we investigate the problem of finding the value d_i such that the average number $evals(d_i)$ of function evaluations necessary to find a new local optimizer is minimal. We proceed as follows: provided that the sizes of all the regions of attraction of the function and the number of function evaluations to get a local minimum are known, we find the expression of $evals(d_i)$ by assuming that the algorithm can run both for an infinite (i) and finite number of iterations (ii). We show that the value of the probability d_i that minimizes $evals(d_i)$ can be only at the endpoints of the interval [0, 1]. We get the same result both in case (i) and case (ii). Finally an algorithm that follows from this investigation is presented; for it estimates of the sizes of the regions of attraction and of the number of function evaluations to get a local minimum are derived from the history of the algorithm behavior.

The algorithm has been tested both for solving $GKLS$ problems [12] of different features and for 16 sample problems quoted in [13]; our results are compared to those by Hedar [8].

2 Preliminary results

For Problem 1.1 we consider the following assumption

Assumption 2.1 i) $f(\cdot)$ has m local minimum points l_i, $i = 1, \ldots, m$ *and* $f(l_i) > f(l_{i+1})$;
ii) $meas\,(S) = 1$.
with $meas\,(S)$ denoting the measure of S.

The main idea of our algorithm is that of carrying out a sequence of local searches. Once a local search has been completed and a new local minimum l_j found, a point x_0 at random uniformly on S is chosen. Whenever $f(x_0)$ is less than $f(l_j)$ a new search is performed from x_0; otherwise a local search is performed with probability d_i. The value d_i is evaluated so that the number of function evaluations required to reach a new local minimum is minimized. The general scheme of our algorithm is

Algorithm 2.1 (**Algorithm Glob**)
Choose x_0 uniformly on S;
$i \leftarrow 1;\ j \leftarrow 1;$
$(x_1, fx_1) \leftarrow local\_search(x_0);$
$l_i = x_1;\ fl_i = fx_1;$
repeat
$j \leftarrow j + 1;$
choose x_0 uniformly on S;
if $f(x_0) \le fl_i$ or $(f(x_0) > fl_i$ and $rand(1) < d_i)$
$(x_1, fx_1) \leftarrow local\_search(x_0);$
if $fx_1 < fl_i$
$i \leftarrow i + 1;$
$l_i \leftarrow x_1;\ fl_i \leftarrow fx_1;$
end if
end if
until a stop rule is met;
end

In algorithm *Glob* we denote by $local\_search(x_0)$ any procedure that starting from a point x_0 returns a local minimum l_i of Problem 1.1 and its function value. We assume that Problem 1.1 satisfies all the conditions required to make $local\_search(x_0)$ convergent. The function $rand(1)$ denotes a generator of random numbers in the interval [0, 1]. We have the following proposition

Proposition 2.1 *Let assumption 2.1 hold and consider a run of algorithm Glob. Then the probability that l_i is a global minimum of Problem 1.1 tends to one as $j \to \infty$.*

First we settle the following notation

Definition 2.1 – $A_{0,j} \equiv \{x \in S \mid$ starting from x, $local\_search(\cdot)$ returns local minimum $l_j\}$;
– $A_{i,j} \equiv \{x \in S \mid f(x) \le f(l_i)$; starting from x, $local\_search(\cdot)$ returns local minimum $l_j\}$;
– $p_{0,j} = meas(A_{0,j})$;
– $p_{i,j} = meas(A_{i,j})$.

We have

$$\sum_{i=1}^{m} p_{0,i} = meas(S) = 1.$$

Clearly a local minimum l_i with $meas(A_{0,i})$ close to zero will be difficult to find, while the case $meas(A_{0,i}) \approx 1$ is easy to handle. We consider the following definitions for algorithm *Glob*.

Definition 2.2 – $t_i \equiv$ the probability that having found the local minimum l_i, in a subsequent iteration no new local minimum is detected;
– $Prob_{i,j}(d_i) \equiv$ the probability that the algorithm, having found the local minimum l_i, can find the local minimum l_j in a subsequent iteration.

3 Infinite number of iterations

We calculate the average number of function evaluations so that algorithm *Glob* having found a local minimum finds any new one. We assume that algorithm *Glob* can run an infinite numbers of iterations. Further it is assumed to know the values $p_{0,j}$ and $p_{i,j}$, $i = 1, \ldots, m-1$ e $j = 1, \ldots, m$, and that the number of function evaluations required by $local\_search$ is $k =$ constant.

We prove first

Lemma 3.1

$$t_i = \sum_{j=1}^{i} p_{0,j} + \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) (1 - d_i), \tag{3.1}$$

$$Prob_{i,j}(d_i) = p_{i,j} + d_i (p_{0,j} - p_{i,j}). \tag{3.2}$$

Proof We get the first statement by noting that whenever x_0 is chosen in $A_{i,j}$ as $j = 1, \ldots, i$, then no new local minimum can be found; while whenever x_0 is chosen in $A_{0,j} - A_{i,j}$, $j = i+1, \ldots, m$, then the probability of not moving from l_i is $(1 - d_i)$.

The second statement follows from the remark: we get a new local minimum both whenever the starting point x_0 is chosen in $A_{i,j}$ and, with probability d_i, whenever x_0 is chosen in $A_{0,j} - A_{i,j}$. □

The following holds.

Theorem 3.1 *The average number of function evaluations so that algorithm Glob, having found a local minimum l_i, finds any new one is given by*

$$evals_1(d_i) = f_i \frac{1}{Prob_{i,*}}, \quad i = 1, \ldots, m-1, \tag{3.3}$$

with

$$Prob_{i,*} = \sum_{j=i+1}^{m} p_{i,j} + d_i \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right), \tag{3.4}$$

$$f_i = k \sum_{j=i+1}^{m} p_{i,j} + k d_i (1 - \sum_{j=i+1}^{m} p_{i,j}) + (1 - d_i)(1 - \sum_{j=i+1}^{m} p_{i,j}). \tag{3.5}$$

Proof Let $Prob_{i,*}$ be the probability of finding a new local minimum having already found l_i. $Prob_{i,*}$ is calculated by noting that whenever x_0 is chosen, a new minimum is found or in $A_{i,j}$, as $j = i+1, \ldots, m$, or with probability d_i in $A_{0,j} - A_{i,j}\, j = i+1, \ldots, m$. Hence

$$Prob_{i,*} = \sum_{j=i+1}^{m} Prob_{ij}(d_i).$$

Combining the latter with (3.2) we get (3.4).

Let f_i be the average number of function evaluations carried out in a single choice of x_0. We get (3.5) by noting that at each iteration we can have or $x_0 \in A_{i,j}$, as $j = i+1, \ldots, m$ or $x_0 \in S - A_{i,j}$ as $j = i+1, \ldots, m$. In the first case k function evaluations are needed; in the second k or one function evaluations are needed with probability d_i or $1 - d_i$ respectively. By taking into account that the number n_p of iterations needed to find a new minimum is the inverse of the value of the probability of finding a new minimum, the product of n_p with the number of function evaluations par iteration gives (3.3). □

Problem 3.1 Let us consider Problem 1.1 and let the values k, $p_{0,j}$ and $p_{i,j}$ be given. Find value d_i^* such that

$$evals_1(d_i^*) = \min_{d_i} evals_1(d_i).$$

The investigation of Problem 3.1 is easy to carry out. We calculate which value of d_i gives the minimum of such a function. We have as $i = 1, \ldots, m-1$,

$$evals_1(d_i) = \frac{(k-1)\sum_{j=i+1}^{m} p_{i,j} + d_i\left(1 - \sum_{j=i+1}^{m} p_{i,j}\right)(k-1) + 1}{\sum_{j=i+1}^{m} p_{i,j} + d_i\left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j}\right)}.$$

The derivative with respect to d_i gives

$$evals_1(d_i)' = \frac{(k-1)\left(\sum_{j=i+1}^{m} p_{i,j}\right)\left(1 - \sum_{j=i+1}^{m} p_{0,j}\right) - \sum_{j=i+1}^{m}(p_{0,j} - p_{i,j})}{\left(\sum_{j=i+1}^{m} p_{i,j} + d_i\left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j}\right)\right)^2},$$

from which it can be stated that the derivative sign does not change for $d_i \in [0, 1]$. Specifically, we find by simple calculations that the derivative is greater than or equal to zero for

$$k \geq \frac{\left(\sum_{j=i+1}^{m} p_{0,j}\right)\left(1 - \sum_{j=i+1}^{m} p_{i,j}\right)}{\left(\sum_{j=i+1}^{m} p_{i,j}\right)\left(1 - \sum_{j=i+1}^{m} p_{0,j}\right)}. \tag{3.6}$$

The condition (3.6) is important in that it links the probability $p_{i,j}$ with the number k of function evaluations performed at each local search in order to choose the most convenient value of d_i: if the condition is met, we must take $d_i = 0$ otherwise $d_i = 1$.

The behavior of $evals_1$ as d_i varies, it is illustrated in Fig.1, where we have taken

$$\sum_{j=i+1}^{m} p_{0,j} = 0.1, \quad \sum_{j=i+1}^{m} p_{i,j} = 0.01,$$

and where to k are given the values 10, 11 e 12, respectively. We get the constant straight line for $k = 11$, which is the value of the right hand side of (3.6) too. In such a case any choice of d_i gives the same number of function evaluations.

4 Finite number of iterations

We proceed in much the same way as for the algorithm running an infinite number of iterations. We prove

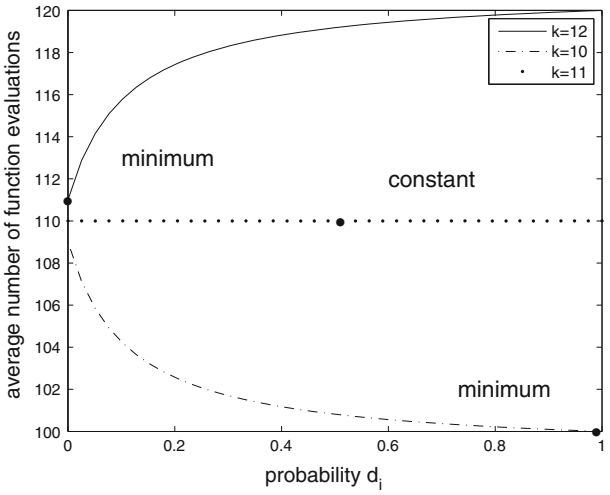

Fig. 1 Average number of function evaluations needed for $d_i \in [0, 1]$, once having found a local minimum, to find a new one

Lemma 4.1 *Assume that algorithm Glob has found a local minimum l_i. Then the probability $Prob_i^{(n)}$ to find, starting from l_i, any new local minimum within at most n iterations, is*

$$Prob_i^{(n)} = 1 - t_i^n, \tag{4.1}$$

where t_i is defined in definition 2.2.

Proof The probability that not moving from l_i, no new local minimum is found in n iterations, is

$$\overline{Prob}_i^{(n)} = t_i^n,$$

Clearly

$$Prob_i^{(n)} + \overline{Prob}_i^{(n)} = 1;$$

that is we get (4.1). □

We have

Theorem 4.1 *Assume that algorithm Glob, starting from l_i, runs for at most n iterations and we stop when a new local minimum is found. Then the average number of function evaluations is given by*

$$evals_2(n, d_i) = k(1 - t_i^n) + \frac{t_i(1 - t_i^n)(1 + d_i(k - 1))}{1 - t_i}. \tag{4.2}$$

where t_i is given by (3.1).

Proof In running algorithm *Glob* for n iterations once we have reached the local minimum l_i, $n + 1$ events e_l, $l = 1, \ldots, n + 1$ can take place. The first n events denote that the algorithm finds a new minimum at the l-th iteration and for the remaining $n - l$ iterations no

move from l_i takes place. The last event denotes that no new local minimum is found in n iterations. The probability of each event e_l is given by

$$t_i^{l-1}\left(\sum_{j=i+1}^{m} p_{i,j} + d_i\left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j}\right)\right),$$

and the the average number of function evaluations for each event e_l is

$$t_i^{l-1}\left(\sum_{j=i+1}^{m} p_{i,j} + d_i\left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j}\right)\right)(k + (kd_i + (1-d_i))(l-1)). \tag{4.3}$$

The first two factors of the product in (4.3) refer to the probabilities to find or not to find a new local minimum in any iteration. The third factor gives the average number of function evaluations that are carried out in the event e_l. The probability of the event e_{n+1} is t_i^n. Hence the average number of function evaluations for e_{n+1} is

$$t_i^n\,((kd_i + (1-d_i))n).$$

Then the overall average number of function evaluations is

$$(1-t_i)\sum_{l=1}^{n} t_i^{l-1}\,(k + (kd_i + (1-d_i))(l-1)) + t_i^n((kd_i + (1-d_i))n),$$

which can be written as

$$k(1-t_i)\sum_{l=0}^{n-1} t_i^l + (1-t_i)\sum_{l=0}^{n-1} t_i^l(kd_i + (1-d_i))l + t_i^n(kd + (1-d_i))n.$$

By using the formulae

$$\sum_{s=0}^{n} t^s = \frac{1-t^{n+1}}{1-t},$$
$$\sum_{s=0}^{n} st^s = \frac{-t^{n+1}(1+n(1-t))+t}{(1-t)^2},$$

we get

$$k(1-t_i^n) + \frac{(kd_i + (1-d_i))(-t_i^n(1+(n-1)(1-t_i))+t_i)}{1-t_i} + t_i^n(kd_i + (1-d_i))n.$$

From the latter by trivial steps we get (4.2). □

We consider the problem

Problem 4.1 Suppose algorithm *Glob* has found the local minimum l_i. Given values k and $p_{i,j}$, and fixed a real positive number acc, as accuracy value in the computation, find n^* and

d_i^* such that

$$evals_2(n^*, d_i^*) = \min_{n,d_i} \; k \cdot (1 - t_i^n) + \frac{t_i(1 - t_i^n)(1 + d_i(k-1))}{1 - t_i},$$

$$\text{under the constraint} \quad 1 - t_i^n \geq 1 - acc. \tag{4.4}$$

In Fig. 2 we draw the graph of $evals_2(n, d_i)$ with

$$acc = 0.001, \quad \sum_{j=1}^{i} p_{0,j} = 0.9, \quad \sum_{j=i+1}^{m} p_{0,j} = 0.1, \quad \sum_{j=i+1}^{m} p_{i,j} = 0.01,$$

and n chosen such that the constraint (4.4) is met. The graphs show again that the minimum is reached for $d_i = 0$ or for $d_i = 1$. We now investigate Problem 4.1. By assuming that the constraint (4.4) is an equality (the optimal solution n^* is the smallest positive integer such that the constraint is met), we have $1 - t_i^n = 1 - acc$, and the function to minimize is

$$g(d_i) = k \cdot (1 - acc) + \frac{t_i(1 - acc)(1 + d_i(k-1))}{1 - t_i}, \tag{4.5}$$

where t_i depends on d_i. The derivative of (4.5) with respect to d_i, following trivial steps, becomes a second degree polynomial in d_i

$$g(d_i)' = \frac{-(p_2 - p_3)^2(k-1)d_i^2 - 2p_3(k-1)(p_2 - p_3)d_i - p_2 + kp_3 - p_3^2(k-1)}{(1-t)^2} \tag{4.6}$$

where p_2 and p_3 are given by

$$p_2 = \sum_{j=i+1}^{m} p_{0,j}, \quad p_3 = \sum_{j=i+1}^{m} p_{i,j}.$$

Since $p_2 > p_3$, the signs of the coefficients of (4.6) allow us to state there does not exist a local minimum in the interior of the interval [0, 1] and as a consequence the minimum of $g(d_i)$ is attained at the endpoint, that is, $d_i = 0$ or $d_i = 1$. We get

$$g(0) = k(1 - acc) + \frac{(1 - p_3)(1 - acc)}{p_3}, \quad g(1) = k(1 - acc) + \frac{k(1 - p_2)(1 - acc)}{p_2}.$$

The inequality $g(0) \leq g(1)$ (i.e. the assumption that g is increasing) gives

$$\frac{1 - p_3}{p_3} \leq \frac{k(1 - p_2)}{p_2}; \tag{4.7}$$

hence we find again the relation we got in (3.6).

5 The new algorithm and numerical results

We present a new algorithm that fits the general scheme 2.1 and exploits (3.6). In real problems we usually do not know the values of $p_{0,j}$ and $p_{i,j}$; hence the probabilities $d_1, d_2, \ldots, d_m$

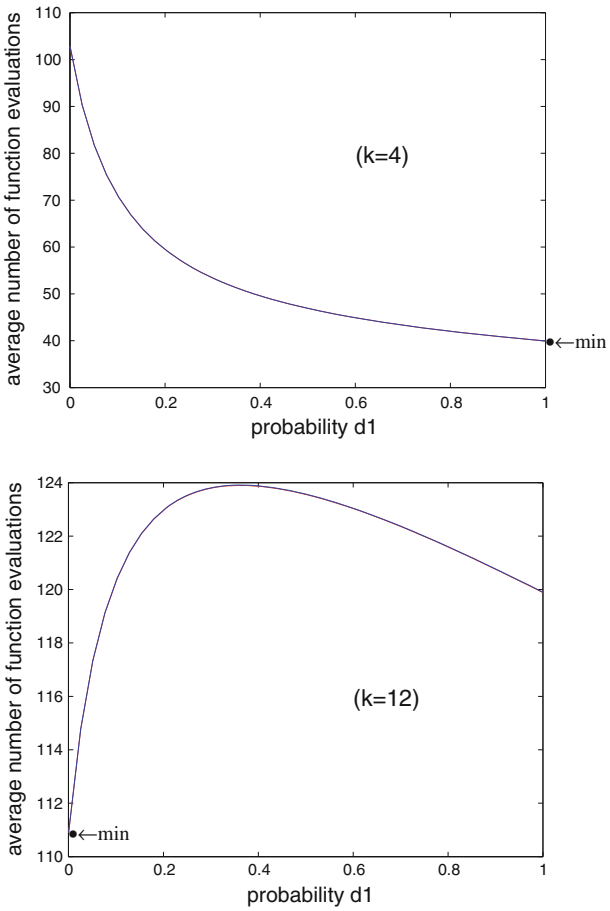

Fig. 2 Average number of function evaluations for $k = 4$ and $k = 12$

in the optimization of the functions in Problems 3.1 and 4.1 cannot be calculated exactly. By making the following approximation of the values

$$p_2 = \sum_{j=i+1}^{m} p_{0,j}, \quad p_3 = \sum_{j=i+1}^{m} p_{i,j},$$

which appear both in the definitions of $evals_1$ and $evals_2$ in Problems 3.1 and 4.1, respectively, we can design a rule for choosing the values d_i, $i = 1, \ldots, m$, in algorithm *Glob*. Specifically, from (3.6) and (4.7) we get:

$$d_i = \begin{cases} 0 & \text{if } k > (p_2 \cdot (1 - p_3))/(p_3 \cdot (1 - p_2)), \\ 1 & \text{otherwise,} \end{cases}$$

where p_2, p_3 and k are approximated as follows:

$$p_2 = 1/(\text{number of searches carried out}),$$
$$p_3 = 1/(\text{number of iterations already carried out}),$$
$$k = (\text{number of function evaluations in the local searches}).$$

Hence the line

$$\textbf{if } f(x_0) \leq fl_i \ \textbf{ or } \ (f(x_0) > fl_i \ \textbf{and} \ rand(1) < d_i)$$

of algorithm $Glob$ is replaced by the line

$$\textbf{if } f(x_0) \leq fl_i \ \textbf{ or } \text{yes\_box}() = 1$$

where $yes\_box()$ is a procedure that returns zero or 1 and is defined by

Procedure 5.1 (yes_box())

$p_2 = 1/$(number of searches already carried out);
$p_3 = 1/$(number of iterations already carried out);
$k =$ (number of function evaluations in the local searches);

```
if p2 = 1
  p2 = 2 · p3;
end if
if p2 < 1
  ratio=(p2 · (1 − p3))/(p3 · (1 − p2));
else
  ratio=inf;
end if
if k>ratio
  yes=0;
else
  yes=1;
end if
end
```

In the sequel we shall denote by $Glob_{\text{new}}$ the algorithm $Glob$ completed with procedure 5.1. We tested the performance of the new algorithm by minimizing two sets of functions: 6 functions constructed by the GKLS package [12] and 16 test problems used in [8] and reported in [13]. The GKLS functions are the first two elements of the classes of dimensions 2, 4 and 6 constructed with the default parameters. Each function is continuously differentiable and has 9 local minima and one global minimum. In Table 1 we report the approximate values, calculated by performing 10^6 runs, of the $ratio = (p_2 \cdot (1 - p_3))/(p_3 \cdot (1 - p_2))$ for each local minimum and for each function. The parameter k which refers to the computational cost of a local search has been evaluated as the sum of function and gradient evaluations; it has been found that the local minimizations have approximatively the same computational cost for any local minimum of a given GKLS function. The local minimization has been carried out by calls to the code *cgtrust* for the first set of functions and to the code *bfgswopt* for the second set. Both codes are by Kelley [14]. The *cgtrust* code is a $trust\ region$ algorithm while the *bfgswopt* code makes use of a *quasi-Newton* method with the *BFGS* formula to update the inverse of the hessian of the function to minimize. Further, a *polynomial* formula is used to carry out the search along a line.

Table 1 Approximate values of $ratio = p_2(1-p_3)/(p_3(1-p_2))$ and k for the GKLS functions

	$fun\_no.1$ $ratio$	$fun\_no.2$ $ratio$	$fun\_no.3$ $ratio$	$fun\_no.4$ $ratio$	$fun\_no.5$ $ratio$	$fun\_no.6$ $ratio$
$local\_min_1$	44	21.4	87.4	288	1,570	3,300
$local\_min_2$	12.1	11.9	140	1,070	2,330	4,350
$local\_min_3$	9.56	14.3	88.1	320	79,000	3,690
$local\_min_4$	8.16	10.2	90.5	399	90,800	18,000
$local\_min_5$	10.6	11.3	56.2	780	942	27,300
$local\_min_6$	10.7	15.8	21,300	49	1,710	234
$local\_min_7$	5.66	17.7	399	554	>1,00,000	5,850
$local\_min_8$	8.08	20.9	211	2,200	>1,00,000	4,990
$local\_min_9$	5.62	13.4	158	1,700	>1,00,000	>1,00,000
k	34	34	40	40	43	41

Table 2 Function and gradient evaluations of $Glob_{\text{new}}$ for GKLS functions

Problem	Dim	$Glob$ with $d_i = 0$			$Glob$ with $d_i = 1$			$Glob_{\text{new}}$		
		Succ	fun_ev	der_ev	Succ	fun_ev	der_ev	Succ	fun_ev	der_ev
fun_1	2	100	38	49	100	61	171	100	44	86
fun_2	2	100	63	56	100	51	143	100	60	100
fun_3	4	100	1,358	49	100	108	298	100	298	355
fun_4	4	100	26,875	65	100	157	438	100	423	490
fun_5	6	5	20,2849	53	100	9,038	25,122	100	14,124	19,815
fun_6	6	2	83,896	54	100	21,834	60,577	100	25,309	52,384

Since the $Glob_{\text{new}}$ algorithm makes use of random procedures and the results of few runs can be misleading, we execute 100 runs of the algorithm with each run bounded by 10^6 iterations for the GKLS set and by 2,000 iterations for the second set of functions but not for the Easom function whose maximum iteration number has been taken equal to 5,000. The values which refer to the numbers of function and gradient evaluations and those that refer to the number of local searches have been calculated as mean values of those collected in each successful run. To compare the algorithm performances we use, as a stop rule, the condition used in [8]:

$$|f^* - \bar{f}| < \varepsilon_1 |f^*| + \varepsilon_2, \quad \varepsilon_1 = 1.e-3, \quad \varepsilon_2 = 1.e-5$$

where f^* is the global minimum value and $\bar{f}$ is the function value of the last local minimum found.

In Table 2 we report the results of our experiments in minimizing the 6 GKLS functions. To test the effectiveness of $Glob_{\text{new}}$ we let algorithm $Glob$ run both with $d_i = 0$ and $d_i = 1$. That is, in the first case, at each iteration a local search is carried on only if the new chosen point x_0 has a function value $f(x_0)$ less than the lowest function value already found; in the second case we carry on a local search whatever $f(x_0)$ is. Columns 3, 4 and 5 of Table 2 refer to the three experiments. As it could be expected from the values in Table 1 for functions of dimension 2, the choice $d_i = 0$ is the optimal one while for the remaining functions $d_i = 1$

Table 3 Function and gradient evaluations of $Glob_{\text{new}}$ for standard test functions

Probl. no.	Probl. name	Dimension	Succ	*fun_eval*	*der_eval*	*tot_evals*
1	Branin	2	100	22	9	40
2	Easom	2	82	1,970	1,746	5,462
3	Goldstein	2	100	123	43	209
4	Shubert	2	100	140	28	196
5	Zakharov_2	2	100	22	10	42
6	Rosenbrok_2	2	100	123	54	231
7	Sphere	3	100	4	2	10
8	$\text{Hartmann}_{3,4}$	3	100	34	11	67
9	$\text{Shekel}_{4,5}$	4	100	185	68	457
10	$\text{Shekel}_{4,7}$	4	100	100	77	559
11	$\text{Shekel}_{4,10}$	4	100	253	74	549
12	Zakharov_5	5	100	104	49	349
13	Rosenbrok_5	5	100	452	151	1,207
14	$\text{Hartmann}_{6,4}$	6	100	68	19	182
15	Zakharov_{10}	10	100	307	137	1,677
16	Rosenbrok_{10}	10	82	1,386	351	4,896

Table 4 Comparison between $Glob_{\text{new}}$ and DTS_{APS}

Problem no.	Problem name	$Glob_{\text{new}}$		DTS_{APS}	
		Succ	Total evals	Succ	Total evals
1	Branin	100	40	100	212
2	Easom	82	5,462	82	223
3	Goldstein	100	209	100	230
4	Shubert	100	196	92	274
5	Zakharov_2	100	42	100	201
6	Rosenbrok_2	100	231	100	254
7	Sphere	100	10	100	446
8	$\text{Hartmann}_{3,4}$	100	67	100	438
9	$\text{Shekel}_{4,5}$	100	457	75	819
10	$\text{Shekel}_{4,7}$	100	559	65	812
11	$\text{Shekel}_{4,10}$	100	549	52	828
12	Zakharov_5	100	349	100	1,003
13	Rosenbrok_5	100	1,207	85	1,684
14	$\text{Hartmann}_{6,4}$	100	182	83	1,787
15	Zakharov_{10}	100	1,677	100	4,032
16	Rosenbrok_{10}	82	4,896	85	9,037

is optimal. Note that for the problems of dimension 6 the choice $d_i = 0$ is rarely successful. The $Glob_{\text{new}}$ algorithm shows good performance as it tends to the optimal case.

In Table 3 we report the detailed results of our experiments for the second set of test problems. In the column labeled *tot_evals* we assume that the computational cost of a gradient evaluation is equal to the problem dimension times the cost of a function evaluation. Finally,

in Table 4 we compare our results with those given in [8]. The given results allow us to state that the new technique to approximate the value of the optimal d_i is valuable and can lead to efficient algorithms.

References

1. Horst, R., Pardalos, P.: Handbook of Global Optimization. Kluwer, Dordrecht (1995)
2. Horst, R., Tuy, H.: Global Optimization, Deterministic Approaches. Springer, Berlin (1993)
3. Strongin, R., Sergeyev, Y.: Global Optimization with Non-convex Constraints: Sequential and Parallel Algorithms. Kluwer, Dordrecht (2000)
4. Zhigljavsky, A., Žilinskas, A.: Stochastic Global Optimization. Springer, New York (2008)
5. Boender, C.G.E., Rinnooy Kan, A.H.G.: Bayesian stopping rules for multistart global optimization methods. Math. Program. **37**, 59–80 (1987)
6. Cetin, B.C., Barhen, J., Burdick, J.W.: Terminal repeller unconstrained subenergy tunnelling (trust) for fast global optimization. J. Optim. Theory Appl. **77**(1), 97–126 (1993)
7. Desai, R., Patil, R.: Salo: combining simulated annealing and local optimization for efficient global optimization. In: Proceedings 9th Florida AI Research Simposium (FLAIRS-96'), pp. 233–237 (1996)
8. Hedar, A.R., Fukushima, M.: Tabu search directed by direct search methods for non linear global optimization. Eur. J. Oper. Res. **170**(2), 329–349 (2006)
9. Levy, A., Montalvo, A.: The tunneling algorithm for the global minimization of functions. SIAM J. Sci. Stat. Comp. **6**, 15–29 (1985)
10. Lucidi, S., Piccioni, M.: Random tunneling by means of acceptance-rejection sampling for global optimization. J. Optim. Theory Appl. **62**(2), 255–277 (1989)
11. Oblow, E.M.: Spt: a stochastic tunneling algorithm for global optimization. J. Global Optim. **20**, 195–212 (2001)
12. Gaviano, M., Kvasov, D., Lera, D., Sergeyev, Y.: Software for generation of test functions with known local and global minima for global optimization. ACM Trans. Math. Softw. **29**(4), 469–480 (2003)
13. Torn, A., Zilinskas, A.: Global Optimization. Lectures Notes in Computer Science, vol. 350, pp. 1–252. Springer (1989)
14. Kelley, C.T.: Iterative Methods for Optimization. SIAM, Philadelphia (1999)