Heuristic Pattern Search and Its Hybridization with Simulated Annealing for Nonlinear Global Optimization*

Abdel-Rahman Hedar and Masao Fukushima
Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

January 31, 2003, Revised November 5, 2003

Abstract

In this paper, we present a new approach of hybrid simulated annealing method for minimizing multimodel functions called the simulated annealing heuristic pattern search (SAHPS) method. Two subsidiary methods are proposed to achieve the final form of the global search method SAHPS. First, we introduce the approximate descent direction (ADD) method, which is a derivative-free procedure with high ability of producing a descent direction. Then, the ADD method is combined with a pattern search method with direction pruning to construct the heuristic pattern search (HPS) method. The last method is hybridized with simulated annealing to obtain the SAHPS method. The experimental results through well-known test functions are shown to demonstrate the efficiency of the proposed method SAHPS.

Keywords: Unconstrained global optimization, Descent direction, Pattern search, Metaheuristics, Simulated annealing

1 Introduction

Optimizing multimodal functions has been fascinating over the two decades coincidental with the great interest of the metaheuristics. Metaheuristics have primarily been applied to combinatorial optimization problems [16, 18]. However, during the past decade metaheuristics have extended their coverage to continuous global optimization problems. Global optimization refers to finding the extreme value of a given nonconvex function in a certain feasible region [7, 17]. In this paper, we will focus on the case of unconstrained global minimization, i.e., our problem is

$$\min_{x \epsilon R^n} \ f(x),$$

where f is a real valued function defined on R^n.

Simulated annealing (SA) [1, 10, 13, 14] is one of the most effective metaheuristics not only for combinatorial optimization but also for continuous global optimization. The SA algorithm successively generates a trial point in a neighborhood of the current solution and determines whether or not the current solution is replaced by the trial point based on a probability depending on the difference between their function values. Convergence to an

*This research was supported in part by a Grant-in-Aid for Scientific Research from the Ministry of Education, Science, Sports and Culture of Japan.

optimal solution can theoretically be guaranteed only after an infinite number of iterations controlled by the procedure so-called cooling schedule. A proper cooling schedule is needed in the finite-time implementation to simulate the asymptotic convergence behavior of the SA. For that reason, SA suffers from slow convergence and also it may wander around the optimal solution if high accuracy is needed.

In continuous optimization, combining SA with direct search methods is a practical remedy to overcome the slow convergence of SA. Recently, there have been a number of attempts in the combined use of SA and direct search methods. Nelder-Mead method [15], which is a popular simplex-based direct search method, has received much attention in designing such hybrid methods [3, 4, 6, 12]. In this paper, we present a hybrid method that combines SA with a new pattern search method. Pattern search (PS) methods constitute a subclass of direct search methods, in which exploratory moves from the current solution to trial points are made along pattern directions with a certain step size. If these exploratory moves give no improvement, then the step size is decreased to refine the search [11, 20].

We will make use of two new ideas to form the main parts of the hybrid algorithm. We first introduce a derivative-free heuristic method to produce an approximate descent direction at the current solution, which we call the Approximate Descent Direction (ADD) method. Some preliminary numerical results show that the ADD method has a high ability to obtain a descent direction. Next, we use the ADD method to design a new PS method called the Heuristic Pattern Search (HPS) method. In the HPS method, the ADD method is recalled to obtain an approximate descent direction v at the current iterate. If no improvement is obtained along the vector v, then we use v to prune the set of pattern search directions to generate other exploratory moves. Finally, we hybridize SA and HPS to construct a global search method, called the Simulated Annealing Heuristic Pattern Search (SAHPS) method. The SAHPS tries to get better movements through the SA acceptance procedure or by using the HPS procedure. More specifically, we first introduce a new exploring neighborhood search to generate a number of SA trial points. If some of these trail points can be accepted by the SA acceptance procedure, this means the search can go further and there is no need to use a local search method. Otherwise, we apply some iterations of the HPS method to generate more local exploratory trial points. In the final stage of the search, we apply a direct search method to refine the best solution obtained so far. Numerical results with 19 well-known test functions indicate that the SAHPS exhibits a very promising performance to obtain global minima of multimodal functions.

The paper is organized as follows. We introduce the ADD and the HPS methods with some numerical results to show their performance in Section 2 and Section 3, respectively. The description of the main SAHPS method is given in Section 4. In Section 5, we discuss the experimental results along with the initialization of some parameters and the setting of the control parameters of the SAHPS method. Finally, the conclusion makes up Section 6.

2 Approximate Descent Direction

In this section, we present the ADD method in which we use m close exploring points to generate an approximate descent direction. Given a point $p \in R^n$, we want to obtain an approximate descent direction $v \in R^n$ of f at p. We randomly generate m points $\{y_i\}_{i=1}^m$ close to p and compute the direction v at p as follows:

$$v = \sum_{i=1}^{m} w_i e_i, \tag{1}$$

where

$$\begin{aligned} w_i &= \frac{\Delta f_i}{\sum_{j=1}^{m} |\Delta f_j|}, \quad i = 1, 2, \ldots, m, \\ e_i &= -\frac{(y_i - p)}{\|y_i - p\|}, \quad i = 1, 2, \ldots, m, \\ \Delta f_i &= f(y_i) - f(p), \quad i = 1, 2, \ldots, m. \end{aligned} \tag{2}$$

We can show some theoretical results concerning the descent property of the direction v in the following two special cases.

The linear case. If f is a linear function, i.e., $f(x) = c^T x + b$, $c \in R^n, b \in R$, then the vector v in (1) can be written as

$$\begin{aligned} v &= \frac{-1}{\sum_{j=1}^{m} |\Delta f_j|} \sum_{i=1}^{m} \Delta f_i \frac{(y_i - p)}{\|y_i - p\|} \\ &= \frac{-1}{\sum_{j=1}^{m} |\Delta f_j|} \sum_{i=1}^{m} c^T (y_i - p) \frac{(y_i - p)}{\|y_i - p\|} \\ &= \frac{-1}{\sum_{j=1}^{m} |\Delta f_j|} \left(\sum_{i=1}^{m} \frac{(y_i - p)(y_i - p)^T}{\|y_i - p\|} \right) c \\ &= -\gamma A c, \end{aligned}$$

where $\gamma = 1/\sum_{j=1}^{m} |\Delta f_j|$ and $A = \sum_{i=1}^{m} (y_i - p)(y_i - p)^T / \|y_i - p\|$. Note that matrix A is positive semidefinite, since $x^T A x = \sum_{i=1}^{m} \left((y_i - p)^T x \right)^2 / \|y_i - p\| \geq 0$ for any $x \in R^n$. Therefore, it holds that $\nabla f(p)^T v = -\gamma c^T A c \leq 0$, i.e., v is a descent direction.

The nonlinear case. If f is a differentiable nonlinear function, we can approximate f around point p as $f(x) \cong f(p) + \nabla f(p)^T (x - p), \forall x \in N(p)$, where $N(p)$ is a small neighborhood of p. Therefore, if points y_i, $i = 1, \ldots, m$, are chosen from the neighborhood $N(p)$, then the vector v in (1) can be represented approximately as

$$\begin{aligned} v &= \frac{-1}{\sum_{j=1}^{m} |\Delta f_j|} \sum_{i=1}^{m} \Delta f_i \frac{(y_i - p)}{\|y_i - p\|} \\ &\cong \frac{-1}{\sum_{j=1}^{m} |\Delta f_j|} \sum_{i=1}^{m} \nabla f(p)^T (y_i - p) \frac{(y_i - p)}{\|y_i - p\|} \\ &= \frac{-1}{\sum_{j=1}^{m} |\Delta f_j|} \left(\sum_{i=1}^{m} \frac{(y_i - p)(y_i - p)^T}{\|y_i - p\|} \right) \nabla f(p) \qquad (3) \\ &= -\gamma A \nabla f(p), \qquad (4) \end{aligned}$$

where A and γ are defined as before. Since A is positive semidefinite, we obtain $\nabla f(p)^T v \cong -\gamma \nabla f(p)^T A \nabla f(p) \leq 0$, i.e., v is expected to be a descent direction.

Remark 1 *The vector $-\nabla f(p)$, which is referred to as the steepest descent direction of f at p, provides the direction along which the function f decreases most rapidly. Since our aim is to minimize f, it is therefore plausible to try to obtain a direction v that imitates $-\nabla f(p)$. Actually, we can show that the vector v in (1) can simulate the steepest descent direction $-\nabla f(p)$ under some conditions. Specifically, the vector v becomes approximately proportional to $-\nabla f(p)$ by setting $m = n$ and choosing the points $\{y_i\}_{i=1}^{n}$ so as to meet the following conditions:*

- *The points* $\{y_i\}_{i=1}^n$ *are in equal distance from* p*, i.e.,* $\|y_i - p\| = \epsilon,\ i = 1, 2, \ldots, n$*, for some* $\epsilon > 0$*;*
- *the vectors* $\{(y_i - p)\}_{i=1}^n$ *are orthogonal to each other.*

In fact, by letting $u_i = (y_i - p) / \|y_i - p\|$ *for* $i = 1, \ldots, n$*, we may rewrite the formula (3) as follows:*

$$\begin{aligned} v &\cong \frac{-\epsilon}{\sum_{j=1}^n |\Delta f_j|} \left(\sum_{i=1}^n u_i u_i^T \right) \nabla f(p) \\ &= \frac{-\epsilon}{\sum_{j=1}^n |\Delta f_j|} Q\ \nabla f(p), \end{aligned}$$

where $Q = \sum_{i=1}^n u_i u_i^T$*. Since* $Qu_i = u_i, i = 1, \ldots, n$*, we can readily see* $Q = I_n$*, and this shows that* v *is approximately proportional to* $-\nabla f(p)$*. This result provides a controlled way to generate the exploring points* $\{y_i\}_{i=1}^m$ *rather than a complete random choice of them. Both of these two ways of generating the points* $\{y_i\}_{i=1}^m$ *are tested numerically at the end of this section.*

Remark 2 *In general, it is not easy to know how small the neighborhood* $N(p)$ *should be in order to ensure the validity of approximation (4). Let* $N(p) = \{x : \|x - p\| \le \epsilon\}$ *and* $M = \sup\{\nabla^2 f(\zeta) : \zeta \in N(p)\}$*. Then we have for any* $x \in N(p)$

$$\left| f(x) - f(p) - \nabla f(p)^T (x - p) \right| = \left| \frac{1}{2} (x - p)^T \nabla^2 f(p + \theta (x - p))(x - p) \right| \le \frac{1}{2} M \epsilon^2$$

where $\theta \in (0, 1)$*. This estimate may suggest a proper choice of radius* ϵ *of the neighborhood. However, a priori knowledge of* M *is not available except for some special cases. Our numerical experiments reported below suggest that the choice* $\epsilon = 10^{-3}$ *practically works well.*

The previous theoretical analysis uses an approximation of f in the nonlinear case. Here we give some numerical results to show the effectiveness of the ADD method in obtaining a descent direction. We test this procedure using Easom (ES), Goldstein and Price (GP), Griewank (GR) and Rosenbrock ($R_n, n = 2, 4, 10, 20, 50$) functions, as shown in Table 1. See the Appendix for the analytical formulae and search domains for these test functions. For each test function, three different test points $p_j, j = 1, 2, 3$, are randomly chosen from its search domains. In addition, three test points $p_j, j = 4, 5, 6$, are chosen to be close to the global minimum x^* for each test function such that $p_4 = x^* - 0.1e$, $p_5 = x^* - 0.01e$ and $p_6 = x^* - 0.001e$, where $e \in R^n$ is the vector of ones. An approximate descent direction v is computed 100 times for each point using different exploring points $\{y_i\}_{i=1}^m$ in each trial. The success rate for obtaining a descent direction in these 100 trials are reported in Table 1. The following two methods are used to generate the exploring points $\{y_i\}_{i=1}^m$ close to each point $p = p_j, j = 1, \ldots, 6$:

1. *Random:* Let $m = 2$ and choose points $\{y_i\}_{i=1}^2$ randomly from the neighborhood $N(p, \epsilon) = \{x \in R^n : \|p - x\| \le \epsilon\}$.
2. *Orthogonal:* Let $m = n$ and choose points $\{y_i\}_{i=1}^n$ such that $\{(y_i - p)\}_{i=1}^n$ are parallel to the coordinate axes and $\|y_i - p\| = \epsilon,\ i = 1, \ldots, n$, for some $\epsilon > 0$.

Table 1: Success rates of obtaining descent direction for the test functions

f	ϵ	p_1	p_2	p_3	p_4	p_5	p_6
	10^{-1}	100%	100%	100%	100%	55%/49%	45%/46%
R_2	10^{-3}	100%	100%	100%	100%	100%	100%
	10^{-5}	100%	100%	100%	100%	100%	100%
	10^{-1}	100%	100%	100%	96%/100%	54%/63%	42%/35%
R_4	10^{-3}	100%	100%	100%	100%	100%	96%/100%
	10^{-5}	100%	100%	100%	100%	100%	100%
	10^{-1}	100%	100%	100%	94%/100%	55%/47%	51%/55%
R_{10}	10^{-3}	100%	100%	100%	100%	100%	95%/100%
	10^{-5}	100%	100%	100%	100%	100%	100%
	10^{-1}	100%	100%	100%	95%/100%	52%/52%	57%/38%
R_{20}	10^{-3}	100%	100%	100%	100%	99%/100%	98%/100%
	10^{-5}	100%	100%	100%	100%	100%	100%
	10^{-1}	100%	100%	100%	92%/100%	56%/57%	48%/31%
R_{50}	10^{-3}	100%	100%	100%	100%	100%	97%/100%
	10^{-5}	100%	100%	100%	100%	100%	100%
	10^{-1}	100%	100%	100%	99%/100%	90%/70%	58%/76%
ES	10^{-3}	100%	100%	100%	100%	100%	99%/100%
	10^{-5}	100%	100%	100%	100%	100%	100%
	10^{-1}	100%	100%	87%/48%	100%	52%/45%	52%/48%
GP	10^{-3}	100%	100%	100%	100%	100%	99%/100%
	10^{-5}	100%	100%	100%	100%	100%	100%
	10^{-1}	93%/100%	96%/100%	87%/100%	89%/100%	80%/65%	51%/48%
GR	10^{-3}	94%/100%	96%/100%	83%/100%	92%/100%	91%/100%	89%/100%
	10^{-5}	95%/100%	97%/100%	89%/100%	95%/100%	87%/100%	90%/100%

As to the neighborhood radius ϵ, smaller value of ϵ is expected to yield higher possibility of obtaining a descent direction. To examine how small ϵ is enough to achieve this goal, we have tested three values of ϵ, which are 10^{-1}, $10^{-3}$and 10^{-5}. If two percentages are reported in the same space in Table 1, the first one is related to *Random* and the second one is related to *Orthogonal*. If only one percentage is reported, this means both of them have this percentage.

The results in Table 1 show that using the neighborhood radius $\epsilon = 10^{-3}$ or 10^{-5} is very effective in obtaining a descent direction even in a vicinity of the global minimum. Moreover, there is no significant difference between the results obtained using these two values of ϵ. It is noteworthy that although the *Random* method uses only two random exploring points, it succeeds to obtain a descent direction with a high rate even for higher dimensional functions.

3 Heuristic Pattern Search

In this section, we describe the details of the new pattern search method HPS. At the iteration k with iterate $x_k \in R^n$, the HPS uses the ADD method to generate a direction v at x_k. If we could obtain a better movement along direction v with a certain step size, then we proceed to the next iteration by updating the current iterate. Otherwise, the HPS, like conventional

pattern search (PS) algorithms [20], uses a finite set D of positive spanning directions in R^n to generate a mesh of points. To avoid searching randomly in all these direction, we prune the positive spanning direction set D, by using a control parameter $\beta \in (-1, 1)$, to select only those directions which lie within the angle $\cos^{-1}(\beta)$ from vector v or $-v$, depending on whether v is a descent direction or not, respectively. Thus, we have the following two cases:

1. If v is a descent direction, we prune the positive spanning direction set D to obtain the pruned direction set D_k^p as

$$D_k^p = \left\{ d \in D : d^T v \geq \beta \|d\| \|v\| \right\}. \tag{5}$$

2. If v is not a descent direction, the pruned direction set D_k^p is obtained as

$$D_k^p = \left\{ d \in D : d^T v \leq -\beta \|d\| \|v\| \right\}. \tag{6}$$

Since we do not want to evaluate the (computationally expensive) gradient of f, we judge whether or not v is a descent direction by using a sufficiently small step size $\alpha > 0$. That is, if $f(x_k + \alpha v) < f(x_k)$, we consider v a descent direction. Otherwise, we do not consider v a descent direction. It is noteworthy that Abramson et al. [2] use the gradient to prune the positive spanning direction set D by means of (5) with $v = -\nabla f(x_k)$ and $\beta = 0$. The use of gradients, however, may not be appropriate in the case where they are computationally so expensive that a derivative-free method such as a PS method becomes a method of choice.

Algorithm 3.1 below describes the steps of the HPS method. In the ADD step, we may use either the *Random* method or the *Orthogonal* method described in the previous section. In practice, we prefer to use the *Random* method since the *Orthogonal* method is computationally more expensive. Moreover, the positive spanning direction set D used in the PS step can be set either $\{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$ or $\{e_1, \ldots, e_n, -e\}$, where $e_i \in R^n$ is the ith unit vector in R^n and $e \in R^n$ is the vector of ones.

Algorithm 3.1. **HPS**$(f, x_0, \Delta_0, \alpha, \sigma)$

1. Initialization. Choose an initial solution x_0, fix an initial mesh size $\Delta_0 > 0$, choose the shrinkage coefficient σ of the mesh size from $(0, 1)$, fix a sufficiently small step size $\alpha > 0$, set the pruning control parameter $\beta \in (-1, 1)$, and set the iteration counter $k := 0$.

2. ADD. Calculate the vector v at x_k as in (1). If $f(x_k + \Delta_k v) < f(x_k)$, then set $x_{k+1} := x_k + \Delta_k v$, and go to Step 5.

3. PS. If $f(x_k + \alpha v) < f(x_k)$, then use (5) to obtain D_k^p. Otherwise, use (6) to obtain D_k^p. Evaluate f on the trial points

$\{p_j = x_k + \Delta_k d_j : d_j \in D_k^p,\ j = 1, \ldots, |D_k^p|\}$.

4. Parameter Update. If $\min_{1 \leq j \leq |D_k^p|} f(p_j) < f(x_k)$, then set $x_{k+1} := \arg\min_{1 \leq j \leq |D_k^p|} f(p_j)$. Otherwise, decrease Δ_k through the rule $\Delta_{k+1} := \sigma \Delta_k$.

5. If the stopping condition is satisfied, then terminate. Otherwise, let $k := k+1$ and return to step 2.

To implement Algorithm 3.1, we have to determine a proper value of the pruning control parameter β. We use the standard $2n$ directions, $D = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n, \}$ as a positive spanning direction set. In this case, a proper value for β can be chosen from $(-1, \frac{1}{\sqrt{n}})$

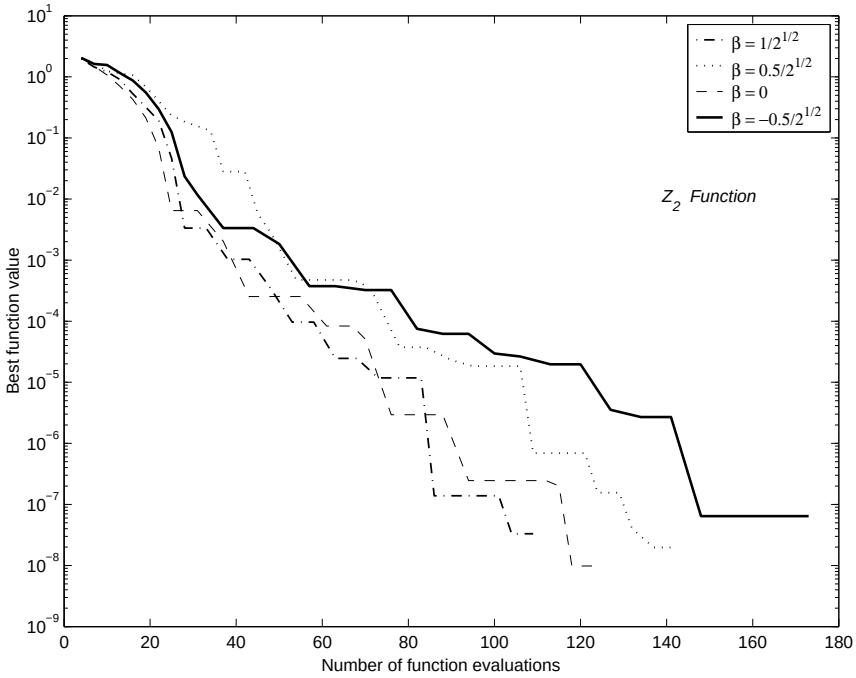

Figure 1: The HPS performance for Zakharov function Z_2.

to guarantee that the pruned direction set D_k^p contains at least one direction. Since the idea of using the parameter β is first presented in this paper and there is no similar computational result we can refer to it, we study the tuning of parameter β through some numerical experiments. Four values $\beta = \frac{1}{\sqrt{n}}, \frac{1}{2\sqrt{n}}, 0, \frac{-1}{2\sqrt{n}}$ have been chosen to make some numerical simulations using Rosenbrock function R_2, De Joung function DJ, and Zakharov functions $Z_n, n = 2, 4, 10, 20$, see the Appendix for the analytical formulae of these test functions. Note that the global minimum values of all these functions are 0. Figures 1–6 show that $\beta = \frac{1}{\sqrt{n}}$ generally gives faster convergence toward the global minima than the other values of β. It is notable from these figures that the performance of the HPS method for the function R_2 is different from that for other functions. Figure 2 shows that the HPS method with $\beta = \frac{1}{\sqrt{n}}$ works well in the early stage of the search, while it suffers from slow convergence in the later stage compared with the method using other values. However, this difference in performance is expected since the HPS method uses the ADD method and descent-type methods usually suffer from slow convergence when applied to R_2.

A question that arises when we test the performance of the HPS method is to what extent the ADD used in HPS helps the PS to get better results. To examine this issue, we compare the HPS method with the plain PS method on Zakharov functions $Z_n, n = 2, 4, 10, 20$. We use the standard $2n$ directions, $D = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$, to generate the pattern search directions in each method. Table 2 shows the best function value (Best f value) and the number of function evaluations (No. f-evals.) achieved by each method. We use the same starting points for all methods. Since there is no random step in the PS method, it was run only once for each problem. On the other hand, the HPS method was run 100 times and the Best f value and the No. f-evals. are the average of these 100 trials. The pruning control parameter β was set equal to $\frac{1}{\sqrt{n}}$. Moreover, the shrinkage coefficient σ is set equal to 0.5, which is the standard value of the shrinkage coefficient in direct search methods. The initial mesh size should be chosen big enough for more efficient local search, so that we set Δ_0

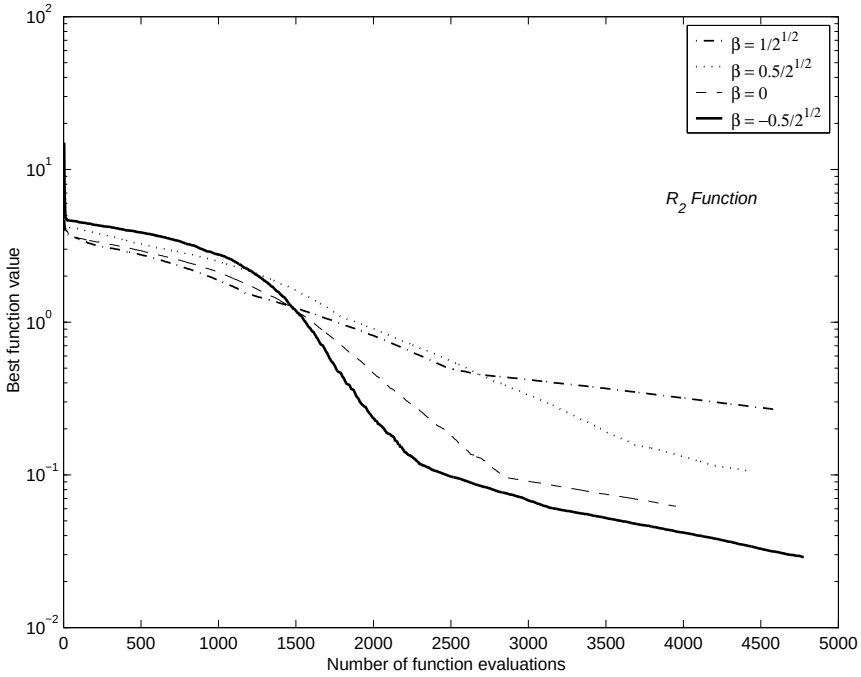

Figure 2: The HPS performance for Rosenbrock function R_2.

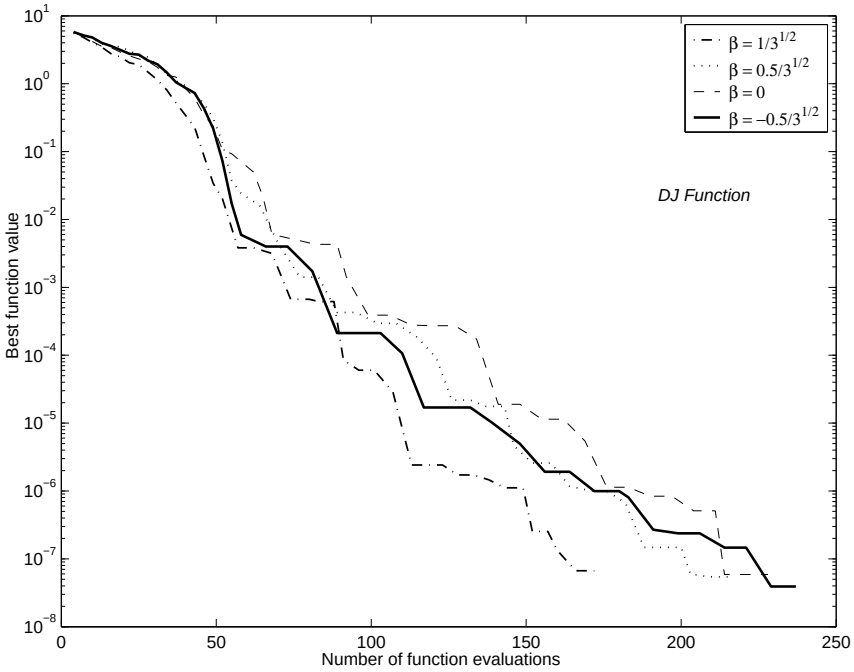

Figure 3: The HPS performance for De Joung function.

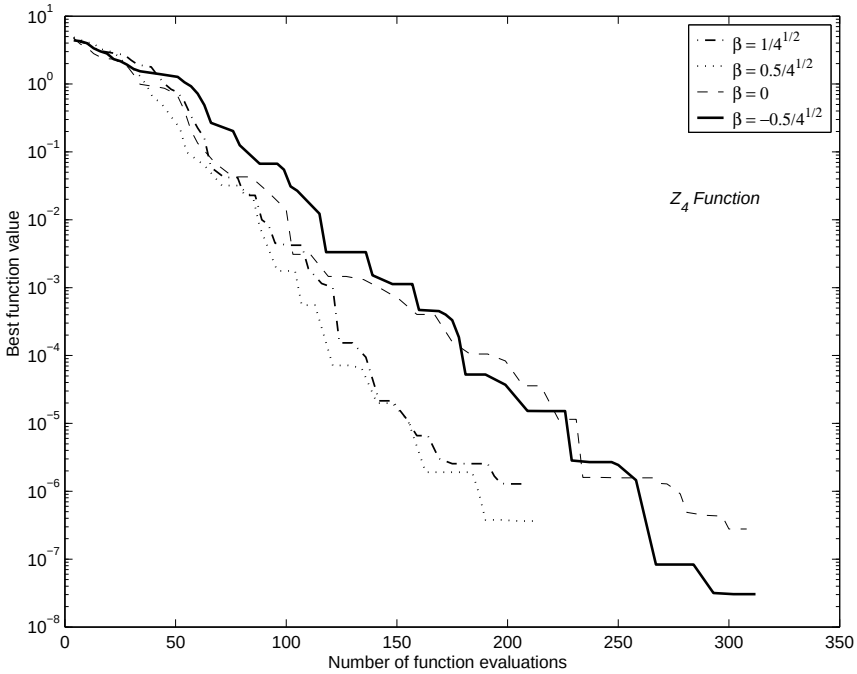

Figure 4: The HPS performance for Zakharov function Z_4.

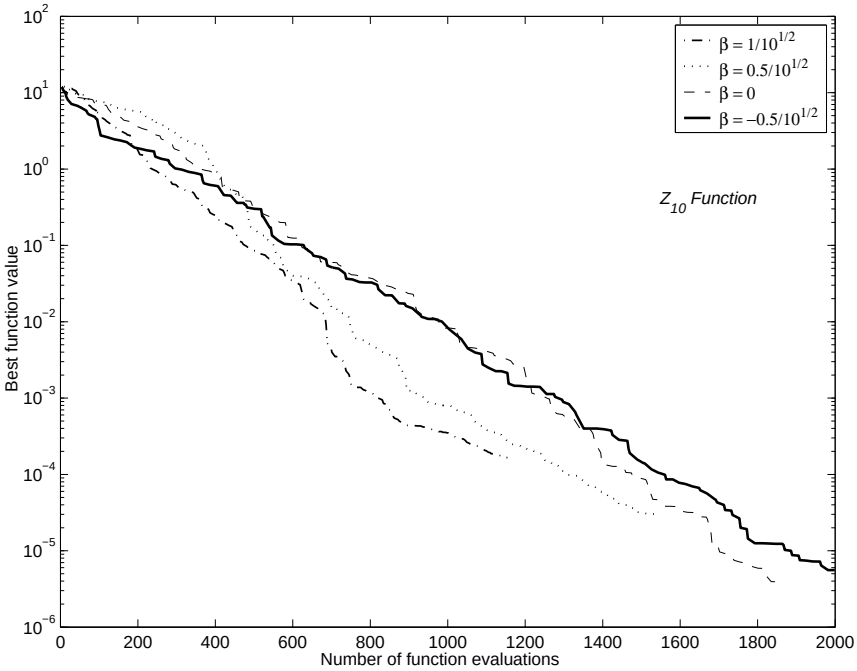

Figure 5: The HPS performance for Zakharov function Z_{10}.

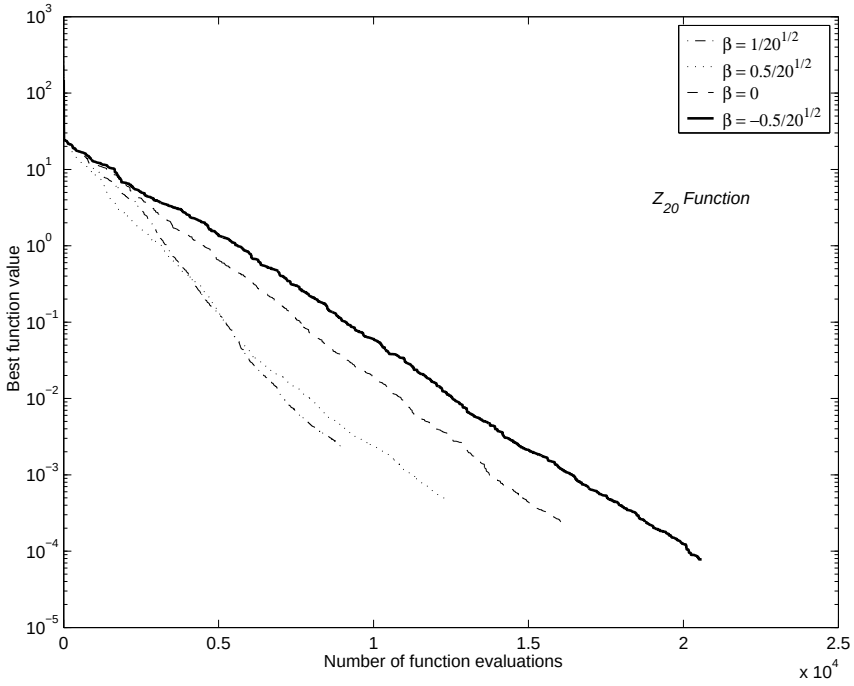

Figure 6: The HPS performance for Zakharov function Z_{20}.

Table 2: Results of PS and HPS for Zakharov functions

	No. f-evals.		Best f value	
f	PS	HPS	PS	HPS
Z_2	97	132	3.0E–9	2.6E–8
Z_4	353	234	9.1E–9	3.1E–7
Z_{10}	7821	1547	2.6E–6	8.4E–5
Z_{20}	50000+	11140	4.4E–2	1.7E–3

equal to 1. The step size α is set equal to 10^{-3}, which is small enough to avoid misleading the search especially in the vicinity of a local minimum. The iteration was terminated in Step 5 when the mesh size became smaller than 10^{-4}, or the number of function evaluations exceeded $50,000$.

From the results shown in Table 2, we may observe that using the ADD in the HPS method can reduce the number of function evaluations in the plain PS method especially for higher dimensional problems.

4 Simulated Annealing HPS

In this section we give the details of our main hybrid pattern search method SAHPS. The SA approach is combined with the HPS to form the hybrid method SAHPS, which is expected to have a higher ability to detect global minima. At each major iteration of the SAHPS method, we first repeat the simulated annealing acceptance trials m_1 times. In each time, a trial point is generated by using an exploring point to guide the SA search along a promising

direction and to avoid making a blind random search. Specifically, we generate an exploring point z_k close to the current iterate x_k and a SA trial is generated along the direction $sign(f(x_k) - f(z_k))(z_k - x_k)$, with a certain step size. If more than m_{ac} out of m_1 trials are accepted, then we immediately proceed to the next major iteration of SAHPS. Otherwise, within the same major iteration, we repeat the HPS iterations m_2 times. In the early stage of the search, the diversification is more needed than the intensification, however, the converse is needed in the final stage of the search. Since the HPS represents the intensification part of the SAHPS, it is better to initialize the value of m_2 at a moderate value and increase it while the search is going on. In the end of the search, we complete the algorithm by applying a fast local search method to refine the best point obtained by the search so far. We prefer to use the Kelley's modification [8, 9] of the Nelder-Mead method [15] in this final step.

More detailed and formal description of the SAHPS method is shown in the following Algorithm 4.1. The setting of parameters used in this algorithm will be discussed later in Section 5.

Algorithm 4.1. **SAHPS**$(f, x_0, \Delta_0, r, \alpha, \epsilon)$

1. Initialization. Choose an initial solution x_0, fix an initial mesh size $\Delta_0 > 0$, choose the shrinkage coefficient σ of the mesh size from $(0, 1)$, fix the SA trial point radius r, fix a sufficiently small step size $\alpha > 0$, and fix a sufficiently small neighborhood radius $\epsilon > 0$. Fix the cooling schedule parameters; initial temperature $T_{\max}$, epoch length M, cooling reduction ratio $\lambda \in (0.5, 0.99)$, and minimum temperature $T_{\min}$. Set the temperature $T := T_{\max}$.

2. The main iteration. Repeat the following *Global SA Search* (Step 2.1) m_1 times. If more than m_{ac} out of m_1 trial points are accepted, then skip the *Local HPS* (Step 2.2) and proceed to Step 3.

2.1 Global SA search. Given the current iterate x_k, generate an exploring point z_k randomly in the neighborhood of x_k with radius ϵ. Generate a trial point x_{SA} in the neighborhood of the current solution x_k by

$$x_{SA} = \begin{cases} x_k + \eta r \left(z_k - x_k\right) / \left\|z_k - x_k\right\|, \text{ if } f\left(z_k\right) \le f\left(x_k\right), \\ x_k - \eta r \left(z_k - x_k\right) / \left\|z_k - x_k\right\|, \text{ otherwise,} \end{cases}$$

where η is a random number in $(0.1, 1)$. Evaluate f on the trial point x_{SA}, and accept it, i.e. $x_{k+1} := x_{SA}$, if

i. $\Delta f := f(x_{SA}) - f(x_k) < 0$, or

ii. $\Delta f \ge 0$, and $p = \exp\left(\frac{-\Delta f}{T}\right) \ge u$, where u is a random number in $(0, 1)$.

2.2. Local HPS. Repeat the following procedure m_2 times.

2.2.a. ADD. Calculate the vector v at x_k as in (1). If $f\left(x_k + \Delta_k v\right) < f\left(x_k\right)$, then set $x_{k+1} := x_k + \Delta_k v$, and proceed to the next iteration of the *Local HPS* loop.

2.2.b. PS. If $f\left(x_k + \alpha v\right) < f\left(x_k\right)$, then use (5) to obtain D_k^p. Otherwise, use (6) to obtain D_k^p. Evaluate f on the trial points $\left\{p_j := x_k + \Delta_k d_j : d_j \in D_k^p,\ j = 1, \ldots, \left|D_k^p\right|\right\}$.

2.2.c. Parameter update. If $\min_{1 \le j \le \left|D_k^p\right|} f(p_j) < f\left(x_k\right)$, then set $x_{k+1} := \arg\min_{1 \le j \le \left|D_k^p\right|} f(p_j)$. Otherwise, decrease Δ_k through the following rule:

$$\Delta_{k+1} := \sigma \Delta_k. \tag{7}$$

3. If the epoch length, which corresponds to M iterations of *Global SA Search*, is not achieved, then go to Step 2.

4. If the cooling schedule is completed ($T \leq T_{\min}$) or the function values of two consecutive improvement trials become close to each other or the number of iterations exceeds $50n$, then go to Step 5. Otherwise, decrease the temperature by setting $T := \lambda T$, increase m_2 slightly, decrease r slightly, and go to Step 2.

5. From the best point found, apply the modified Nelder-Mead method [8, 9].

5 Experimental Results

Below we elaborate on the implementation of Algorithm 4.1.

Initial trial. The initial point x_0 is chosen randomly from the predetermined range $[L, U]$ of the initial points for each test function, where

$$[L, U] = \{x \in R^n : l_i \leq x_i \leq u_i,\ i = 1, \ldots, n\}.$$

Cooling schedule. Generally, choosing a proper cooling schedule is not a trivial task. Our cooling schedule is designed based on some common choice of parameters suggested in the literature, or according to our observation gained through some preliminary numerical experiments. First, the initial temperature $T_{\max}$ is set large enough to make the initial probability of accepting transition close to 1. Beside the initial point x_0, another point $\widetilde{x}_0$ is generated in a neighborhood of x_0 to calculate $T_{\max}$ as

$$T_{\max} := -\frac{1}{\ln(0.9)} \left| f(\widetilde{x}_0) - f(x_0) \right|.$$

The cooling ratio λ is normally chosen to be between 0.9 and 0.99 [13]. Actually, in the original SA method, Kirkpatrick et al. [10] suggested $\lambda = 0.95$, which has become a common choice. However, in our experiments, we observed that the results obtained with $\lambda = 0.95$ were not significantly different from those with $\lambda = 0.9$. The main reason for this insignificant difference is due to the use of refining local search method at the final stage. Since setting λ equal to 0.95 is more computationally costly, we set λ equal to 0.9. A common choice of the number of trials allowed at each temperature, which is called epoch length M, is to let it depend on the size of the problem [14]. Although Kirkpatrick et al. [10] set the value of M equal to n, our preliminary experiments have revealed that setting M equal to $2n$ fits the SAHPS algorithm well. Therefore, we set M equal to $2n$. In the implementation of SA, the cooling schedule is terminated when the temperature reaches a fixed minimum value $T_{\min}$ [14]. We observed that setting $T_{\min}$ equal to $\min\left(10^{-3}, 10^{-3}T_{\max}\right)$ can give a complete cooling schedule in the sense that the acceptance probability at the end is almost zero.

Neighborhood radius. The neighborhood radius ϵ, which is used in generating the exploring points z_k in the *Global SA Search* and in generating the exploring points used to compute vector v in the ADD step, is set equal to 10^{-3}.

SA trial point radius. The radius r, which is used in generating the SA trial points, is initialized as $r_0 := \min_{1 \leq i \leq n} (u_i - l_i)/5$ to fit the search domain of each test function, and then r is reduced in parallel to the reduction of temperature T by setting $r := \max\{0.95r, 0.02r_0\}$.

HPS parameters. The mesh size is initialized as $\Delta_0 := \min_{1 \leq i \leq n} (u_i - l_i)/10$, and when no improvement is achieved, its shrinkage factor σ in (7) is set equal to 0.7. Actually, the standard value of the shrinkage coefficient in direct search methods is 0.5 but, in our

experiments, we observed that using the value 0.7 gives more ability of efficient exploration than the value 0.5, because the latter may decrease the step size prematurely before reaching a proper exploration process. We set the step size α used in Step 2.2.a equal to 10^{-3}. The pattern search strategy described in Section 3 is adopted in the *Local HPS* step.

Loop repetitions. The repetition numbers of the *Global SA Search* and *Local HPS* steps, m_1 and m_2, are both set equal to n initially, which is equal to the half of the epoch length M. However, m_2 is updated as $m_2 := \min\{5n, 1.05m_2\}$ in Step 4 at every major iteration. The control parameter m_{ac}, the desired number of accepted points in the *Global SA Search* step, is set equal to 1.

Termination conditions. Beside the completeness of the cooling schedule, Algorithm 4.1 may be terminated in Step 4, if the difference between the function values of two consecutive improvement trials becomes less than $Tol = 10^{-8}$, or the number of iterations exceeds $It_{max} = 50n$.

In Table 3, we summarize all parameters used in the SAHPS algorithm with their assigned values.

Algorithm 4.1 was programmed in Matlab and applied to 19 well-known test functions [5, 6]. For each function, this Matlab code was run 100 times with different initial points. To judge the success of a trial, we used the condition

$$|f^* - \widehat{f}| < \epsilon_1 |f^*| + \epsilon_2, \tag{8}$$

where $\widehat{f}$ refers to the best function value obtained by SAHPS, f^* refers to the known exact global minimum value, and ϵ_1 and ϵ_2 are small positive numbers. We set ϵ_1 and ϵ_2 equal to 10^{-4} and 10^{-6}, respectively. The average number of function evaluations (Av. f-evals.) and the average errors (Av. Error) reported in Table 4 are those for the successful trials. It is noteworthy that, for some of the functions that fail to achieve the 100% success rate, the success rate can be improved by relaxing the maximum number of iterations or by slowing down the cooling schedule. For example, the success rate for SH function can be improved to 95% with Av. f-evals. of 822 and Av. Error of 9E–6, if we set the maximum number of iterations equal to $100n$ instead of $50n$.

To complete the testing of the SAHPS method, we compare it with other SA-based methods, Enhanced Simulated Annealing (ESA) [19] and Direct Search Simulated Annealing (DSSA) [6]. The results of ESA are taken from its original paper [19], as well as [5]. The results of DSSA are taken from its original paper [6].

The results shown in Table 4 indicate that SAHPS generally outperforms the ESA. Since ESA is a plain SA method without any combination with a local search method, we may conclude that hybridizing HPS with SA significantly improves the performance of SA. On the other hand, the comparison between SAHPS and DSSA does not seem to yield a definitive answer. The performance of DSSA is better for the problems with $n < 5$ but SAHPS outperforms DSSA in higher dimensional problems, i.e., $n \geq 5$, in terms of the number of function evaluations.

6 Conclusion

In this paper, we have presented a new hybrid global search method in which a direct search method is combined with the SA procedure to remedy the slow convergence of the latter

Table 3: SAHPS Parameters

Parameter	Definition	Value
$T_{\max}$	maximum (initial) temperature	$-\frac{1}{\ln(0.9)}\left\|f(\widetilde{x}_0)-f(x_0)\right\|$
$T_{\min}$	minimum temperature	$\min\left(10^{-3}, 10^{-3}T_{\max}\right)$
λ	cooling ratio	0.9
M	epoch length	$2n$
ϵ	neighborhood radius	10^{-3}
r_0	initial radius for generating SA trial points	$\min_{1\le i\le n}\left(u_i-l_i\right)/5$
Δ_0	initial mesh size	$\min_{1\le i\le n}\left(u_i-l_i\right)/10$
σ	reduction factor of mesh size	0.7
α	step size for checking descent directions	10^{-3}
m_1	*Global SA Search* repetition number	n
m_2	*Local HPS* repetition number	$\begin{cases}\text{initial: } n\\ \text{update: } \min\{5n, 1.05m_2\}\end{cases}$
m_{ac}	number of accepted points in *Global SA Search*	1
It_{max}	maximum number of iterations	$50n$
Tol	termination tolerance	10^{-8}

Table 4: Results of SAHPS and other SA methods

	Av. f-evals.			Av. Error		
f	SAHPS	ESA	DSSA	SAHPS	ESA	DSSA
RC	318	–	118	4E–7	–	4E–7
ES	432(96%)	–	1442(93%)	5E–9	–	3E–9
GP	311	783	261	5E–9	9E–3	4E–9
BH	346	–	252	8E–9	–	5E–9
HM	278	–	225	5E–8	–	5E–8
SH	450(86%)	–	457(94%)	9E–6	–	9E–6
Z_2	276	15820	186	7E–9	–	4E–9
R_2	357	796	306	6E–9	–	4E-9
DJ	398	–	273	6E–9	–	5E–9
$H_{3,4}$	517(95%)	698	572	2E–6	5E–4	2E–6
$S_{4,5}$	1073(48%)	1137(54%)	993(81%)	3E–7	4E–3	2E–6
$S_{4,7}$	1059(57%)	1223(54%)	932(84%)	4E–5	8E–3	6E–7
$S_{4,10}$	1031(48%)	1189(50%)	992(77%)	1E–5	4E–2	1E–5
Z_5	716	96799	914	8E–9	–	5E–9
R_5	1104(91%)	5364	2685	7E–9	–	3E-9
$H_{6,4}$	997(72%)	1638	1737(92%)	2E–6	6E–2	2E–6
GR	795	–	1830(90%)	8E–9	–	5E–9
Z_{10}	2284	15820	12501	3E–8	2E–3	7E–9
R_{10}	4603(87%)	12403	16785	2E-8	4E-2	7E-9

method. Two new methods have been introduced to design the SAHPS method; one is the ADD method that produces an approximate descent direction, the other is the HPS method that is used to make a local exploratory search in the main SAHPS method. The latter has turned out to be particularly effective because the HPS method shows a superior performance in reducing the computational expense of the plain PS method.

References

[1] E. Aarts and J. Korst (2002). Selected topics in simulated annealing. In C.C. Ribeiro and P. Hansen (Eds.), *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers, Boston, MA.

[2] M. A. Abramson, C. Audet, and J. E. Dennis Jr. (2002). Generalized pattern searches with derivative information. Technical Report TR02-10, Department of Computational and Applied Mathematics, Rice University, Houston, Texas.

[3] M. F. Cardoso, R. L. Salcedo and S. F. de Azevedo (1996). The simplex-simulated annealing approach to continuous non-linear optimization. *Comput. Chem. Eng.*, **20**, 1065–1080.

[4] M. F. Cardoso, R. L. Salcedo, S. F. de Azevedo and D. Barbosa (1997). A simulated annealing approach to the solution of minlp problems. *Comput. Chem. Eng.*, **21**, 1349–1364.

[5] R. Chelouah and P. Siarry (2000). Tabu search applied to global optimization. *European J. of Operational Reasearch*, **123**, 256–270.

[6] A. Hedar and M. Fukushima (2002). Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. *Optimization Methods and Software*, **17**, 891–912.

[7] R. Horst and P. M. Pardalos (Eds.) (1995). Handbook of Global Optimization, Kluwer Academic Publishers, Boston, MA.

[8] C. T. Kelley (1999). Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. *SIAM J. Optim.*, **10**, 43–55.

[9] C. T. Kelley (1999). Iterative Methods for Optimization, Frontiers Appl. Math. 18, SIAM, Philadelphia, PA.

[10] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi (1983). Optimisation by simulated annealing. *Science*, **220**, 671–680.

[11] T.G. Kolda, R.M. Lewis and V. Torczon (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, **45**, 385–482.

[12] V. Kvasnicka and J. Pospichal (1997). A hybrid of simplex method and simulated annealing. *Chemometrics and Intelligent Laboratory Systems*, **39**, 161–173.

[13] P. J. Laarhoven (1988). Theoretical and Computational Aspects of Simulated Annealing, Stichting Mathematisch Centrum, Amsterdam.

[14] P. J. Laarhoven and E. H. Aarts (1987). Simulated Annealing: Theory and Applications, D. Reidel Publishing Company, Dordrecht, Holland.

[15] J. A. Nelder and R. Mead (1965). A simplex method for function minimization. *Comput. J.*, **7**, 308–313.

[16] I. H. Osman and J. P. Kelly (1996). Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Boston, MA.

[17] P.M. Pardalos and M.G.C. Resende (Eds.) (2002). Handbook of Applied Optimization. Oxford University Press, Oxford.

[18] C.C. Ribeiro and P. Hansen (Eds.) (2002). Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, Boston, MA.

[19] P. Siarry, G. Berthiau, F. Durbin and J. Haussy (1997). Enhanced simulated annealing for globally minimizing functions of many continuous variables. *ACM Transactions on Mathematical Software*, **23**, 209–228.

[20] V. Torczon (1997). On the convergence of pattern search algorithms. *SIAM J. Optim.*, **7**, 1–25.

A List of test functions

A.1 Branin RCOS function (RC)

Number of variables: $n = 2$.

Definition: $RC(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$.

Range of initial points: $-5 < x_1 < 10,\ 0 < x_2 < 15$.

Number of local minima: no local minimum[1].

Global minima: $(x_1, x_2)^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$;

$RC((x_1, x_2)^*) = 0.397887$.

A.2 Easom function (ES)

Number of variables: $n = 2$.

Definition: $ES(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$.

Range of initial points: $-10 < x_j < 10,\ j = 1, 2$.

Number of local minima: several local minima.

The global minimum: $(x_1, x_2)^* = (\pi, \pi);\ ES((x_1, x_2)^*) = -1$.

[1]Throughout the Appendix, 'local minimum' means a mere local minimum that is not a global minimum

A.3 Goldstein and Price function (GP)

Number of variables: $n = 2$.

Definition: $GP(x_1, x_2) = u * v$,

where $u = 1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)$,

and $v = 30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)$.

Range of initial points: $-2 < x_j < 2,\ j = 1, 2$.

Number of local minima: 4 local minima.

The global minimum: $(x_1, x_2)^* = (0, -1);\ GP((x_1, x_2)^*) = 3$.

A.4 Bohachevsky functions (BH)

Number of variables: $n = 2$.

Three functions were considered: B_1, B_2 and B_3.

Definition: $BH(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$.

Range of initial points: $-10 \leq x_j \leq 10,\ j = 1, 2$.

Number of local minima: many local minima.

Global minima: $(x_1, x_2)^* = (0, 0);\ BH((x_1, x_2)^*) = 0$.

A.5 Hump function (HM)

Number of variables: $n = 2$.

Definition: $HM(x_1, x_2) = 1.0316285 + 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$.

Range of initial points: $-5 < x_j < 5,\ j = 1, 2$.

Number of local minima: no local minima.

Global minima: $(x_1, x_2)^* = (0.0898, -0.7126),\ (-0.0898, 0.7126); HM((x_1, x_2)^*) = 0$.

A.6 Shubert function (SH)

Number of variables: $n = 2$.

Definition: $SH(x_1, x_2) = \left(\sum_{j=1}^{5} j\cos\left((j+1)\,x_1 + j\right)\right)\left(\sum_{j=1}^{5} j\cos\left((j+1)\,x_2 + j\right)\right)$.

Range of initial points: $-10 < x_j < 10,\ j = 1, 2$.

Number of local minima: 760 local minima.

Global minima: 18 global minima and $SH((x_1, x_2)^*) = -186.7309$.

A.7 De Joung function (DJ)

Number of variables: $n = 3$.

Definition: $DJ(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$.

Range of initial points: $-5 < x_j < 5,\ j = 1, 2, 3$.

Number of local minima: no local minima.

The global minimum: $(x_1, x_2, x_3)^* = (0, 0, 0);\ DJ((x_1, x_2, x_3)^*) = 0$.

A.8 Hartmann function ($H_{3,4}$)

Number of variables: $n = 3$.

Definition: $H_{3,4}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$.

Range of initial points: $0 < x_j < 1,\ j = 1, 2, 3$.

Number of local minima: 4 local minima.

The global minimum: $\mathbf{x}^* = (0.114614, 0.555649, 0.852547)$; $H_{3,4}(\mathbf{x}^*) = -3.86278$.

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.0	30.0	1.0	0.689	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.4699	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.1091	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.0381	0.5743	0.8828

A.9 Shekel functions ($S_{4,m}$)

Number of variables: $n = 4$.

Definition: $S_{4,m}(\mathbf{x}) = -\sum_{i=1}^{m}\left[\sum_{i=1}^{4}\left(x_i - a_i\right)^2 + c_i\right]^{-1}$.

Three functions were considered: $S_{4,5}, S_{4,7}$ and $S_{4,10}$.

Range of initial points: $0 < x_j < 10,\ j = 1, \ldots, 4$.

Number of local minima: m local minima.

Same global minimum for three functions $S_{4,5},\ S_{4,7}$ and $S_{4,10}$: $\mathbf{x}^* = (4, 4, 4, 4)$;
$S_{4,5}(\mathbf{x}^*) = -10.1532,\ S_{4,7}(\mathbf{x}^*) = -10.4029$ and $S_{4,10}(\mathbf{x}^*) = -10.5364$.

i	a_{ij}				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.3
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

A.10 Hartmann function ($H_{6,4}$)

Number of variables: $n = 6$.

Definition: $H_{6,4}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$.

Range of initial points: $0 < x_j < 1,\ j = 1, \ldots, 6$.

Number of local minima: 6 local minima.

The global minimum:

$\mathbf{x}^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300)$; $H_{6,4}(\mathbf{x}^*) = -3.32237$.

i	a_{ij}						c_i
1	10.00	3.00	17.00	3.50	1.70	8.00	1.0
2	0.05	10.00	17.00	0.10	8.00	14.00	1.2
3	3.00	3.50	1.70	10.0	17.00	8.00	3.0
4	17.00	8.00	0.05	10.00	0.10	14.00	3.2
i	p_{ij}						
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886	
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991	
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650	
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381	

A.11 Griewank function (GR)

Number of variables: $n = 6$.

Definition: $GR(\mathbf{x}) = \sum_{j=1}^{6} \frac{x_j^2}{4000} - \prod_{j=1}^{6} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1$.

Range of initial points: $-10 < x_j < 10,\ j = 1, 2, \ldots, 6$.

Number of local minima: many local minima.

The global minimum: $\mathbf{x}^* = (0, \ldots, 0)$, $GR(\mathbf{x}^*) = 0$.

A.12 Rosenbrock functions (R_n)

Number of variables: $n = 2, 5, 10$.

Definition: $R_n(\mathbf{x}) = \sum_{j=1}^{n-1} \left[100\left(x_j^2 - x_{j+1}\right)^2 + (x_j - 1)^2\right]$.

Range of initial points: $-5 < x_j < 10,\ j = 1, 2, \ldots, n$.

Number of local minima: no local minimum.

The global minimum: $\mathbf{x}^* = (1, \ldots, 1)$, $R_n(\mathbf{x}^*) = 0$.

A.13 Zakharov functions (Z_n)

Number of variables: $n = 2, 4, 5, 10, 20$.

Definition: $Z_n(\mathbf{x}) = \sum_{j=1}^{n} x_j^2 + \left(\sum_{j=1}^{n} 0.5 j x_j\right)^2 + \left(\sum_{j=1}^{n} 0.5 j x_j\right)^4$.

Range of initial points: $-5 < x_j < 10,\ j = 1, 2, \ldots, n$.

Number of local minima: no local minimum.

The global minimum: $\mathbf{x}^* = (0, \ldots, 0)$, $Z_n(\mathbf{x}^*) = 0$.