J Glob Optim (2009) 43:67–82
DOI 10.1007/s10898-008-9293-y

A reassessment of the Snyman–Fatti dynamic search trajectory method for unconstrained global optimization

J. A. Snyman · S. Kok

Received: 14 August 2007 / Accepted: 5 March 2008 / Published online: 22 March 2008

Abstract The aim of this paper is to present a thorough reassessment of the *Snyman–Fatti (SF) Multi-start Global Minimization Algorithm with Dynamic Search Trajectories*, first published twenty years ago. The reassessment is done with reference to a slightly modified version of the original method, the essentials of which are summarized here. Results of the performance of the current code on an extensive set of standard test problems commonly in use today, are presented. This allows for a fair assessment to be made of the performance of the SF algorithm relative to that of the popular Differential Evolution (DE) method, for which test results on the same standard set of test problems used here for the SF algorithm, are also given. The tests show that the SF algorithm, that requires relatively few parameter settings, is a reliably robust and competitive method compared to the DE method. The results also indicate that the SF trajectory algorithm is particularly promising to solve minimum potential energy problems to determine the structure of atomic and molecular clusters.

Keywords Global optimization algorithm · Dynamic search trajectories · Random multi-start optimization · Lennard-Jones clusters

1 Introduction

When first published twenty years ago, the *Snyman–Fatti (SF) Multi-start Global Minimization Algorithm with Dynamic Search Trajectories* [1] for global continuous unconstrained optimization, promised to have great potential impact because its performance, on the limited number of test problems considered then, appeared to be better or at least competitive with

J. A. Snyman (✉) · S. Kok
Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria 0002, South Africa
e-mail: jan.snyman@up.ac.za

S. Kok
e-mail: schalk.kok@up.ac.za

that of the established global optimization methods known and used at the time. During the intervening years, however, this promise was not fulfilled.

Although a fair number of authors have referred to the SF method in the literature, and most notably it was extensively used to solve laminate and composite structural problems [2–6], it has certainly not enjoyed extensive theoretical interest and wide practical application initially hoped for. We believe there are three main reasons for this disappointing response. Firstly, although the SF algorithm has similar stochastic features, it was overshadowed by the emergence of more exotic new stochastic methods, such as simulated annealing (SA) and genetic algorithms (GAs), that came to the fore and created extensive interest and associated research activity at the time the SF algorithm was published. Since then further attention to the SF method was also undermined by the prominence enjoyed by new population based global optimization algorithms such as the differential evolution (DE) and particle swarm optimization (PSO) algorithms that were published in the mid 1990s. These evolutionary methods are conceptually simple and relatively easy to implement and have been extensively studied and tested. Currently variants of these methods, as well as related techniques, continue to be the subjects of much international research. Of particular interest here is the recent paper by Laskari et al. [7] in which they propose a hybrid algorithm that combines the SF and DE algorithms.

A second reason that inhibited the use of the SF method is that, although the physical idea on which it is based is fairly simple, the implementation is relatively complicated and also requires gradient information, compared to the function-value-only evolutionary techniques. Thirdly, at a time when extensive tests results are available for the other methods, the sparse and outdated test results presented for the SF algorithm in the original paper do not represent a sufficient incentive to create renewed interest in the SF method. This is a pity since, as the authors will attempt to show in this paper, a careful computer implementation of the SF algorithm yields a code which is very robust and gives results that are competitive with that of one of the best evolutionary algorithms currently available.

With the above in mind, this paper focusses on the thorough numerical testing of the performance of the current SF code on an extensive set of standard test problems commonly in use today. These results allow for a fair assessment to be made of the performance of the SF algorithm relative to that of other popular evolutionary global optimization methods by comparing, in particular, its performance with that of the Differential Evolution (DE) method. The DE algorithm is selected for comparison because it is currently considered as arguably one of the best global optimization methods available. In the next section, a description of the basic methodology employed in the SF algorithm is presented. This is followed by the presentation of computational results for both the SF and DE algorithms applied to: (i) an extensive set of standard test problems; and (ii) the determination of lowest energy structures of Lennard-Jones clusters. The paper concludes with a reassessment of the SF algorithm in the light of the results of the new computational experiments.

Before continuing, however, it is of interest to briefly relate the SF algorithm to similar ideas in molecular dynamics that pre-date the publication of the SF method in 1987, and of which at the time Snyman and Fatti were unaware. Basic to the SF method is the search for relatively low local minima of an objective function through energy conserving search trajectories of a particle of unit mass. These trajectories are generated by the solution of a second order dynamical differential equation involving the gradient of the function, where the function is taken to represent the potential energy of the particle in space. For the numerical integration use is made of the approximately energy-conserving leap-frog method [8]. Methods similar in spirit to the SF algorithm, although not specifically aimed at searching for the global minimum, have been proposed in the molecular simulation area. For example,

the intrinsic states method of Stillinger and Weber [9] uses molecular dynamic trajectories to search the space of importance, with regular switches after a number of integration steps to a local minimization method like conjugate gradient to find a local minimum. The SF algorithm is also similar to the standard practice in molecular dynamics for describing the dynamics of a particle system with fixed volume so that the total energy is conserved. This approach, in which a second order dynamical equation identical to that of the SF algorithm is used, has been around since the late 1950s (see [10]). Also of interest is the use, since 1970, of leap-frog algorithms in molecular dynamics to perform the numerical integration (see [11]), and are actually an outgrowth of what is known as the Verlet algorithm [12].

2 A description of the SF global minimization methodology using dynamic search trajectories

For a detailed presentation and discussion of the motivation and theorems on which the SF algorithm is based, the reader is referred to the original paper of Snyman and Fatti [1]. Here we restrict ourselves to a summary giving the essentials of the multi-start global optimization methodology using dynamic search trajectories.

We consider the unconstrained global optimization problem that can be stated: for a continuously differentiable objective function f: $X \subset \Re^n \rightarrow \Re$, find a point $\mathbf{x}^* \in X$, such that

$$f^* = f(\mathbf{x}^*) = \text{minimum}_{\mathbf{x}\in X}\{f(\mathbf{x})\} \tag{1}$$

The SF algorithm applied to this problem, is basically a multi-start technique in which several starting points are sampled in the domain of interest X (usually defined by a box in $\Re^n$), and a local search procedure is applied to each sample point. The method is heuristic in essence with the lowest minimum found after a finite number of searches being taken as an estimate of f^*.

2.1 Local convergence of search trajectories

In the local search the SF algorithm explores the variable space using search trajectories derived from the differential equation:

$$\ddot{\mathbf{x}} = -\nabla f(\mathbf{x}(t)) \tag{2}$$

where $\nabla f(\mathbf{x}(t))$ is the gradient vector of $f(\mathbf{x}(t))$. The gradient vector may also be denoted by $\mathbf{g}(\mathbf{x}(t))$. Equation 2 describes the motion of a particle of *unit mass* in an n-dimensional conservative force field, where $f(\mathbf{x}(t))$ represents the potential energy of the particle. The search trajectories generated here are similar to those used in Snyman's dynamic method for local minimization [13,14]. In the SF global method, however, the trajectories are modified in a manner that ensures, in the case of multiple local minima, a higher probability of convergence to a lower local minimum than would have been achieved had conventional gradient local search methods, including the local dynamic method of Snyman, been used. The specific modifications employed results in an increase in the *regions of convergence* of the lower minima including, in particular, that of the global minimum. A stopping rule, derived from a Bayesian probability argument, is used to decide when to end the global sampling and take the current overall minimum value of f, taken over all sampling points (*iterations*) to date, as the global minimum f^*.

For initial conditions, position $\mathbf{x}(0) = \mathbf{x}^0$ and velocity $\mathbf{v}(0) = \dot{\mathbf{x}}(0) = \mathbf{0}$, integrating (2) from time 0 to t, implies the energy conservation relationship:

$$\frac{1}{2}\|\mathbf{v}(t)\|^2 + f(\mathbf{x}(t)) = \frac{1}{2}\|\mathbf{v}(0)\|^2 + f(\mathbf{x}(0)) = f(\mathbf{x}(0)) \tag{3}$$

The first term on the left-hand side of (3) represents the kinetic energy, whereas the second term represents the potential energy of the particle of unit mass, at any instant t. Obviously the particle will start moving in the direction of steepest descent and its kinetic energy will increase, and thus f will decrease, as long as

$$-\nabla f \cdot \mathbf{v} > 0 \tag{4}$$

where $\cdot$ denotes the scalar product.

If descent is not met along the generated path then the magnitude of the velocity $\mathbf{v}$ decreases as it moves uphill and its direction changes towards a local minimizer. If the possibility of more than one local minimizer exists and we are interested in finding the global minimum, a realistic global strategy is to monitor the trajectory and record the point $\mathbf{x}^m$, and corresponding velocity $\mathbf{v}^m = \dot{\mathbf{x}}^m$ and function value f^m, at which the minimum along the path occurs, letting the particle continue uninterrupted along its path with conserved energy. This is done in the hope that it may surmount a ridge of height f^r, $f^m < f^r < f^0 = f(\mathbf{x}^0)$, continuing further along a path that may lead to an even lower value of f beyond the ridge. On the other hand it is necessary to terminate the trajectory before it retraces itself or approximately retraces itself in indefinite periodic or ergodic (space-filling) motion. A proper termination condition, and that employed in the SF algorithm, is to stop the first trajectory once it reaches a point with a function value close to its starting value $f^s = f^0 = f(\mathbf{x}^0)$ while still moving uphill, i.e. while $\mathbf{g} \cdot \mathbf{v} > 0$. At this point, once termination has occurred and after setting the best point $\mathbf{x}^b := \mathbf{x}^m$, it is proposed that a further auxiliary or inner trajectory be started from a new inner starting point $\mathbf{x}^s := \frac{1}{2}(\mathbf{x}^0 + \mathbf{x}^b)$ with initial velocity $\frac{1}{2}\mathbf{v}^m$ and associated starting function value $f^s = f(\mathbf{x}^s)$. Again for this new auxiliary or inner trajectory the function value is monitored, and for this new trajectory its $\mathbf{x}^m$ and associated $\mathbf{v}^m$ are recorded anew. On its termination, again once the function value approaches f^s sufficiently closely while moving uphill, the starting point for the next inner trajectory is taken as $\mathbf{x}^s := \frac{1}{2}(\mathbf{x}^s + \mathbf{x}^b)$ with initial velocity $\frac{1}{2}\mathbf{v}^m$, where $\mathbf{x}^b$ again corresponds to the overall best point for the current sampling point. This generation of successive inner trajectories is continued until $\mathbf{x}^b$ converges or the gradient vector $\nabla f(\mathbf{x}^b) = \mathbf{g}(\mathbf{x}^b)$ is effectively zero. (In the original paper a theorem (Theorem 2.1) is given that guarantees this convergence if the level set $\{\mathbf{x}\colon f(\mathbf{x}) \le f(\mathbf{x}^0)\}$ is bounded and certain conditions concerning the continuity and differentiability of $f(\mathbf{x})$ are satisfied.)

2.2 Numerical considerations in terminating trajectories

Of course, the above strategy assumes that the trajectory obtained from the solution of (2) is exactly known at all time instances. In practice this is not possible, and the generation of the trajectories is done numerically by means of the leap-frog scheme [13]: Given initial position $\mathbf{x}^0$ and initial velocity $\mathbf{v}^0 = \dot{\mathbf{x}}^0$ and a time step Δt, compute for $k = 0, 1, 2, \ldots$

$$\begin{aligned} \mathbf{x}^{k+1} &:= \mathbf{x}^k + \mathbf{v}^k \Delta t \\ \mathbf{v}^{k+1} &:= \mathbf{v}^k - \mathbf{g}(\mathbf{x}^{k+1}) \Delta t \end{aligned} \tag{5}$$

The initial velocity over the first ($k = 0$) step (or on a restart) is taken as $\mathbf{v}^k := -\frac{1}{2}\mathbf{g}^k \Delta t$. The leap-frog scheme (5) requires the selection of an integration time step Δt. In practice, for the first starting point (first iteration $it = 1$), Δt is set at $\Delta t = 1.0$. Thereafter, also for $it = 1$, the following tests are performed: If $\Delta t < \Delta t_{\min} = [D/(50\|\mathbf{g}(\mathbf{x}^0)\|)]^{\frac{1}{2}}$, set $\Delta t := \Delta t_{\min}$, where D is the diameter of the box defining the region of interest X. This guarantees a first step no smaller than one hundredth of the diameter (D) of the box. On the other hand, if $\Delta t > \Delta t_{\max} = \sqrt{50}\Delta t_{\min}$, set $\Delta t := \Delta t_{\max}$, which guarantees a step no larger than one half of the diameter (D) of the region of interest. If necessary, the time step Δt is adjusted further so as to ensure descent on the first step.

Because of inaccuracies introduced by the numerical leap-frog integration scheme it computes only approximately energy conserving trajectories [13]. Additional test are therefore introduced in the actual implementation to ensure termination of the inner trajectories and convergence of $\mathbf{x}^b$, updated after each inner trajectory, to a local minimum for each sample starting point $\mathbf{x}^0$. In particular an inner trajectory is terminated if the particle moves uphill and $f(\mathbf{x}^k) - \breve{f} > \alpha(f(\mathbf{x}^s) - \breve{f})$ or its kinetic energy $= \frac{1}{2}\|\mathbf{v}^k\|^2 < (1-\alpha)(f(\mathbf{x}^s) - \breve{f})$, where $\breve{f}$ is the lowest available value of $f(\mathbf{x})$, and α a parameter chosen to be almost equal to but less than one. Initially $\breve{f} := \infty$ and updated as new values of f^k are computed, and a typical choice for α is $\alpha := 0.95$. An inner trajectory is also terminated if the particle starts zig–zagging, i.e. if the scalar product $\mathbf{v}^{k+1} \cdot \mathbf{v}^k < 0$, and if the number of inner steps exceeds a prescribed number ($inkmax$).

The sequence of inner (auxiliary) trajectories is terminated, i.e. the final inner trajectory ends if at any point $\mathbf{x}^{k+1}$, for given tolerances ε_x and ε_g, either $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| < \varepsilon_x$ or $\|\mathbf{g}(\mathbf{x}^{k+1})\| < \varepsilon_g$ (whilst $f^{k+1} \leq f^m$ and $f^{k+1} < f^b$), or if the overall allowable number of steps ($kmax$) per sample point is exceeded. At this point the current values $\mathbf{x}^{k+1}$ and associated function value f^{k+1} (the local minimum for the current iteration (i.e. current starting point)) are passed to a global evaluation procedure.

2.3 Global termination of SF algorithm

Once the sequence of inner (auxiliary) trajectories for the current *iteration* (i.e. current random starting point) is terminated, the local minimum ($\mathbf{x}^{k+1}$ with function value f^{k+1}) obtained at that point, is evaluated for its probability of being the global minimum after it iterations. This global component of the algorithm involves a stochastic criterion that reports the probability of the lowest obtained minimum to be the global one. To this end, let R_j denote the region of convergence of a local minimum $\hat{f}_j$ in the search space, and α_j denote the probability that a randomly selected point falls within R_j. Let R^* and α^* denote the corresponding quantities for the global minimum f^*. Snyman and Fatti [1] then argue that for the local search methodology described above, because of its special characteristic of seeking a low local minimum, one may assume that for a large class of problems of practical and scientific importance:

$$\alpha^* = \text{maximum}_j\{\alpha_j\} \tag{6}$$

Accordingly they made use of the following theorem to terminate the multi-start algorithm.

Theorem *Let ir be the number of sample (starting) points falling within the region of convergence of the current overall minimum f^{opt} after it random starting points have been sampled. Then under the assumption given in (6) and a noninformative prior distribution, the probability that f^{opt} be equal to f^*, $Pr[f^{opt} = f^*]$, satisfies the following relationship:*

$$Pr \geq q(it, ir) = 1 - (it+1)!(2 \times it - ir)!/[(2 \times it + 1)!(it - ir)!] \tag{7}$$

In practice a tolerance ε_f is prescribed in order to determine whether a newly obtained local minimum also corresponds to the current overall minimum f^{opt}. Thus if, at the end of the final inner trajectory for a given starting point, $|f^{k+1} - f^{opt}| < \epsilon_f$ then the number of successes ir is incremented by one. Also a prescribed target value q^* is set for $q(it, ir)$ so that once $q(it, ir) > q^*$ the global procedure terminates with $f^* := f^{opt}$.

In the above subsections we have attempted to present a thorough but readable account of the essentials of the SF methodology. Of course, in practice much more detailed bookkeeping is done to record, for example, the exact lowest f^{opt} value and corresponding overall

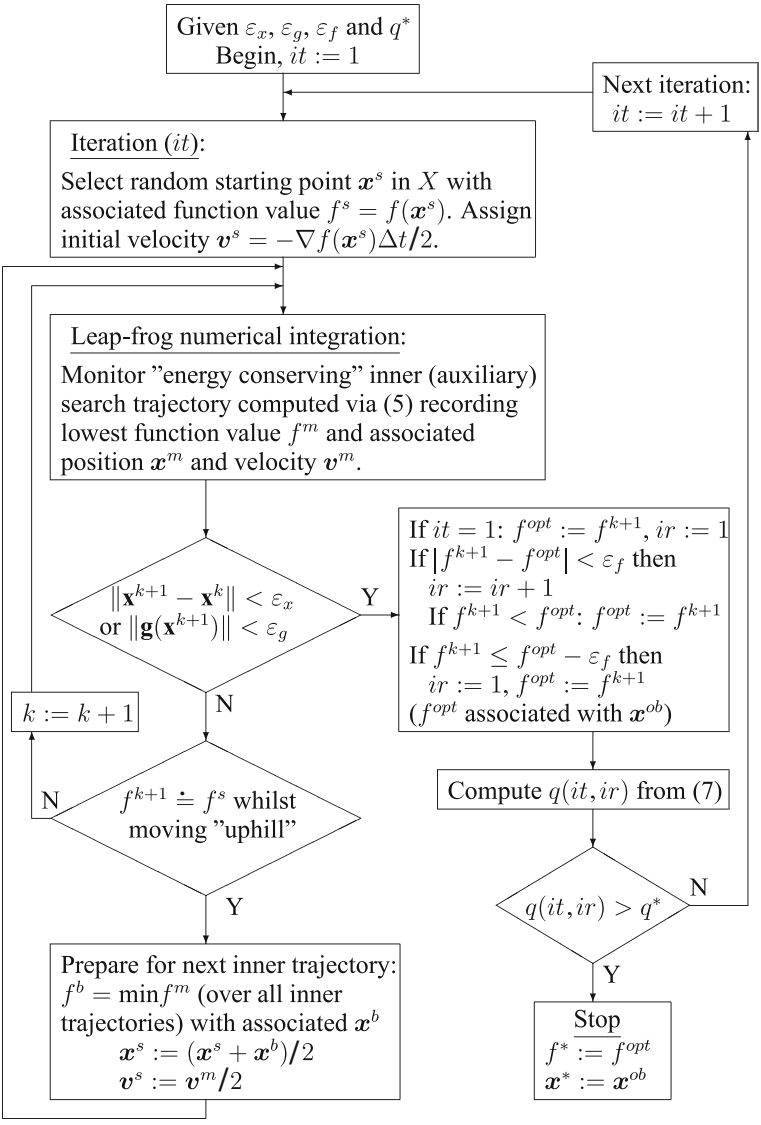

Fig. 1 Schematic overview of SF algorithm

best minimum point $\mathbf{x}^{ob}$. This, and a few other refinements to ensure a logical and robust implementation have, for the sake of clarity, not been detailed above.

Figure 1 provides a schematic overview of the structure and essential elements of the SF multi-start global optimization algorithm. A detailed pseudo-code, in exact correspondence with the actual FORTRAN program currently in use by the authors and employed in the tests of the SF algorithm reported here, as well as the FORTRAN code itself, are available from the authors on request.

3 Numerical results for set of standard test problems

In this section we test the performance of the SF method using a selected set of 48 standard test problems. These problems range from 2 to 20 in dimension and have a variety of inherent difficulty. All problems have continuous variables. A detailed description of each problem in this set can be found in the recent paper by Ali et al. [15]. Note that we excluded the Odd Square Problem (OSP) and Price's Transistor Modelling Problem (PTM) from the test set proposed by Ali et al. [15], since none of the algorithms could ever solve these problems. We compare the results obtained for the SF algorithm with those obtained by the classical differential evolution (DE) method [16] (denoted DE/rand/1/bin in [17]). The results are for the algorithms run 100 times on each of the 48 test problems to determine the success rate ($sr \equiv$ percentage success). All computations were done on a personal computer with an AMD Athlon 64 X2 Dual Core 4400+ CPU with 4Gb RAM, using a single processor.

For the standard implementation of the SF algorithm the minimum number of iterations was set at $minit := 0$ unless otherwise indicated. Other standard settings are $maxit := 5000$, $kmax := 5000$, $inkmax := 40$. The parameter α was set to $\alpha := 0.95$, and the tolerances $\varepsilon_f := 10^{-2}$, $\varepsilon_g := 10^{-3}$ and $\varepsilon_x := 10^{-4}$. For termination on satisfaction of (7), the target probability was set at $q^* := 0.99$.

In Table 1 we compare the performance of the standard SF algorithm with that of the classical DE algorithm, for each n-dimensional problem ($Problem(n)$) in the standard set. In our implementation of the classical DE algorithm, the mutation operation is repeated until the trial point falls within the allowed search space, as described by Ali and Fatti [18]. For the SF algorithm we record the average number of function-cum-gradient-vector evaluations ($No.f\mathbf{g}$) required for each problem, as well as the average relative absolute function value error, $r_f := |f^c - f^*|/(1 + |f^*|)$ at termination, where f^c is the approximation to f^* obtained. The termination is determined here by the satisfaction of (7) for the specified q^*. For all the problems, the components of the gradient vector ∇f were computed by forward finite differences with variable perturbations of 10^{-6}. For the DE algorithm, which uses only function values, the average number of required function evaluations ($No.f$) is listed. We used a population size $N := 10n$, maximum number of iterations (generations) 10,000 and terminated the algorithm whenever $|f_{\max} - f_{\min}| < 10^{-4}$. The reported DE parameters (cross-over setting Cr and mutation scaling factor F) were determined by running the DE algorithm for values from 0.2 to 0.9, in increments of 0.1, for all possible combinations of Cr and F. The reported settings produce the best success rate. If different parameter values produce the same success rate, the settings that require the least number of function evaluations are reported. For both algorithms the percentage success rate (sr) is also indicated for each problem. A success was counted when the absolute error was less than 0.01. The average values listed are taken over successful runs only. The last column in Table 1 contains the relative CPU time, denoted r_{cpu}, defined as the CPU time required by the SF algorithm, divided by the CPU time required by the DE algorithm.

Table 1 Comparison of SF (target probability termination) with DE (termination when $|f_{\max} - f_{\min}| < 10^{-4}$)

Problem	Snyman–Fatti			Differential evolution					r_{cpu}
	No. fg	sr	r_f	C_r	F	No. f	sr	r_f	
ACK(10)	–	0	–	0.8	0.2	15, 485	100	7.8e − 05	–
AP(2)	755	100	6.4e − 05	0.9	0.4	623	100	6.3e − 05	0.66
BL(2)	62	100	5.9e − 10	0.9	0.3	636	100	1.3e − 06	0.27
B1(2)	1,239	98	2.9e − 08	0.9	0.4	863	100	1.4e − 06	1.52
B2(2)	907	98	4.1e − 09	0.8	0.3	871	100	1.2e − 06	1.11
BR(2)	371	100	6.9e − 08	0.8	0.3	760	100	1.2e − 06	0.46
CB3(2)	854	96	4.2e − 08	0.9	0.3	564	100	6.4e − 05	0.88
CB6(2)	794	99	1.4e − 05	0.9	0.3	680	100	1.4e − 05	0.74
CM(2)	665	95	1.4e − 09	0.9	0.2	501	100	3.1e − 05	1.32
DA(2)	999	100	2.4e − 08	0.9	0.4	1, 108	100	2.3e − 08	0.50
EP(2)	722	30	4.6e − 09	0.7	0.8	1, 144	78	4.4e − 07	0.98
EM(10)	–	0	–	0.4	0.4	147, 139	100	1.5e − 05	–
EXP(10)	179	100	8.1e − 08	0.8	0.2	6, 168	100	1.0e − 05	0.07
GP(2)	1,198	100	1.4e − 06	0.9	0.4	738	100	2.9e − 07	0.97
GW(10)	12,542	95	2.2e − 03	0.4	0.2	31, 839	100	3.1e − 05	1.74
GRP(3)	4,362	100	5.0e − 04	0.9	0.4	1, 368	100	6.4e − 05	12.84
H3(3)	967	100	4.0e − 08	0.8	0.3	1, 026	100	1.1e − 05	2.14
H6(6)	949	99	1.1e − 08	0.6	0.4	5, 205	100	1.3e − 06	0.63
HV(3)	838	100	9.8e − 08	0.9	0.5	2, 165	100	3.0e − 06	0.42
HSK(2)	321	98	2.6e − 06	0.9	0.3	474	100	2.6e − 06	0.81
KL(4)	405	100	8.6e − 06	0.9	0.2	730	100	2.1e − 05	1.01
LM1(3)	2,041	100	2.9e − 07	0.9	0.3	1, 132	100	4.0e − 06	2.70
LM2(10)	78,314	98	1.3e − 06	0.7	0.2	7, 641	100	1.6e − 05	41.35
MC(2)	346	99	2.6e − 05	0.9	0.3	502	100	2.9e − 05	0.66
MR(3)	3,355	100	1.1e − 03	0.9	0.3	1, 193	100	1.2e − 04	3.03
MCP(4)	768	100	1.3e − 05	0.9	0.4	1, 015	100	1.7e − 06	1.21
ML(5)	5,171	41	5.9e − 08	0.8	0.8	13, 936	100	3.7e − 06	1.09
MRP(2)	620	100	3.8e − 04	0.9	0.5	1, 033	100	3.5e − 03	0.35
MGP(2)	6,962	9	1.8e − 06	0.2	0.7	5, 818	96	1.7e − 06	1.81
NF2(4)	11,019	99	1.3e − 03	0.8	0.4	14, 582	100	4.1e − 04	2.27
NF3(10)	748	100	7.4e − 10	0.9	0.4	21, 596	100	8.4e − 08	0.05
PP(10)	342	100	1.0e − 05	0.7	0.2	9, 386	100	1.0e − 05	0.49
PRD(2)	7,074	8	2.2e − 08	0.8	0.5	1, 351	100	5.9e − 05	5.87
PQ(4)	1,519	100	5.5e − 06	0.9	0.4	2, 481	100	1.7e − 06	0.35
RG(10)	–	0	–	0.3	0.2	27, 938	100	6.5e − 06	–
RB(10)	6,062	100	1.5e − 04	0.9	0.6	71, 222	100	1.9e − 05	0.12
SAL(5)	13,828	95	5.6e − 07	0.2	0.2	500, 050	15	3.9e − 04	0.03
SF1(2)	9,224	93	6.3e − 04	0.9	0.3	1, 709	100	8.6e − 03	4.02
SF2(2)	49,718	98	3.4e − 03	0.9	0.4	2, 389	100	1.9e − 04	30.13
SBT(2)	5,344	100	6.8e − 08	0.8	0.4	2, 244	100	6.4e − 08	3.98
SWF(10)	–	0	–	0.5	0.2	17, 986	100	5.7e − 08	–

Table 1 continued

Problem	Snyman–Fatti			Differential evolution					r_{cpu}
	No. $f\boldsymbol{g}$	sr	r_f	C_r	F	No. f	sr	r_f	
S5(4)	855	100	3.0e − 04	0.8	0.6	4, 580	100	3.0e − 04	0.27
S7(4)	983	100	2.7e − 04	0.8	0.5	3, 640	100	2.7e − 04	0.46
S10(4)	1,181	100	3.9e − 04	0.9	0.6	4, 048	100	3.9e − 04	0.62
FX(5)	14,811	84	1.5e − 06	0.2	0.9	367, 796	48	1.5e − 06	0.15
SIN(20)	1,223	51	6.3e − 07	0.8	0.2	28, 936	100	8.8e − 06	0.52
ST(9)	–	0	–	0.9	0.5	36, 129	100	0.0e + 00	–
WP(4)	2,483	100	1.4e − 05	0.9	0.6	7, 220	100	5.3e − 06	0.24

Note that the SF algorithm could never solve 5 of the 48 test problems (ACK(10), EM(10), RG(10), SWF(10) and ST(9)), and that for a further 20 problems it did not achieve a 100% success rate. In those cases where the success rate wasn't 100%, better success can be achieved by increasing the minimum number of iterations ($minit$). This is especially necessary for functions that contain many identical near global local minima, with the effect that all these identical local minima have a large effective basin of convergence. Premature convergence to this near global minimum is prevented by increasing the minimum number of iterations. Table 2 contains these results. Note that the 10D Ackley problem ACK(10) can now be solved 46% of the time, while all 20 other problems now have a 100% success rate. A total of 43 of the 48 test problems can thus be solved with a 100% success rate, one problem with a 46% success rate, while only four problems (EM(10), RG(10), SWF(10) and ST(9)) are never solved. It is worth noticing that for the selected tolerances the convergence, as indicated in the column listing the relative function value error r_f, is remarkably sharp.

It is difficult to make a fair comparison of the efficiency of the two algorithms with regard to number of function evaluations required for convergence because of the different natures of the respective algorithms. The DE algorithm only required function values, while the SF algorithm uses both function and gradient information. If the gradients are computed by forward finite differences, as was done for all the problems in Tables 1 and 2, the effective number of function evaluations required is equal to $(n + 1)\times$ the $No.f\mathbf{g}$ listed in the tables. Thus we conclude that in the cases where the gradient information cannot be obtained cheaply, that the DE algorithm may be significantly more efficient. (Of course in practical engineering cases, such as in the field of structural design and CFD studies, where the function values are computed by simulation, the gradients are often obtained very cheaply as a byproduct in the computation of the function value.) However, recall that significant effort is required to find the optimal DE parameter settings for each problem. If one set of DE parameters is used for all problems, we found the best total success rate over all problems for $Cr = 0.6$ and $F = 0.6$. If we compare the performance at this setting with the optimal settings for each problem, we noted a reduction in success rate on 3 of the 48 problems, while on the remaining 45 problems the same success rate was recorded, but on average requiring a factor 2.8 more function evaluations. This includes problem ST(9), where the average DE setting requires a factor 24.9 more function evaluations than the optimal DE setting for this problem. If this outlier is excluded, the average DE settings still require a factor 2.3 more function evaluations as compared to the optimal DE settings per problem.

In additional numerical experiments (for which the results are not detailed here) the computations for the SF algorithm were repeated for a selection of the 48 problems, using

Table 2 Effect of changing the minimum number of iterations on the SF algorithm

Problem	Minit	No. fg	sr	r_f	r_{cpu}
ACK(10)	20	311,260	46	4.7e − 05	99.62
B1(2)	10	1,486	100	2.5e − 08	1.68
B2(2)	10	1,131	100	3.3e − 09	1.26
CB3(2)	10	1,177	100	3.9e − 08	1.04
CB6(2)	10	1,153	100	1.4e − 05	1.01
CM(2)	10	774	100	1.5e − 09	1.51
EP(2)	60	903	100	3.9e − 09	1.18
GW(10)	60	17,366	100	8.9e − 04	2.42
H6(6)	10	1,392	100	7.7e − 09	0.93
HSK(2)	10	493	100	2.6e − 06	1.14
LM2(10)	80	82,156	100	1.1e − 06	42.83
MC(2)	10	543	100	2.6e − 05	1.27
ML(5)	60	5,356	100	5.5e − 08	1.13
MGP(2)	200	12,638	100	1.8e − 06	3.30
NF2(4)	40	13,524	100	9.5e − 04	2.75
PRD(2)	200	7,492	100	1.6e − 08	6.19
SAL(5)	200	16,294	100	3.5e − 07	0.04
SF1(2)	30	9,371	100	1.1e − 03	4.08
SF2(2)	1,500	71,505	100	2.8e − 03	43.33
FX(5)	150	16,582	100	1.5e − 06	0.17
SIN(20)	80	1,453	100	6.0e − 07	0.56

analytical expressions for the gradients. The structure of these selected test problems allows for very efficient computation of the analytical gradients. As example, the percentage reduction in CPU time, only due to using analytical gradients rather than finite differences, for the problems ACK(10), EP(2), GW(10), H3(3), H6(6), RB(10) and SAL(5) are 84%, 53%, 77%, 25%, 44%, 33% and 18% respectively. In some cases, an increase in problem dimension further increases the benefit of analytical gradients, e.g. a CPU time reduction of 78% and 69% is achieved for the 20D RB and 20D SAL test problems respectively. However, sometimes there is no benefit in using analytical gradients, typically if the structure of the cost function is such that multiple chain rule applications are required to compute the analytical gradients.

Nevertheless, even in the worst case of obtaining gradient information using finite differences, there are eight problems where the SF algorithm requires less function evaluations. These are problems BL(2), EXP(10), NF3(10), PP(10), RB(10), SAL(5), S5(4) and FX(5). On the other hand the DE algorithm is vastly more economic in terms of total function evaluations in solving the eight problems, GRP(3), LM1(3), LM2(10), MR(3), PRD(2), SF1(2), SF2(2) and SBT(2). In the remaining cases the performance of the algorithms are, within an order of magnitude, comparable to each other.

If, however, CPU time is used as comparative measure, the SF algorithm is more efficient ($r_{\text{cpu}} < 1$) than the DE algorithm for 20 of the test problems. Since both algorithms were implemented in FORTRAN, using the same compiler, this indicates that the SF algorithm has less overhead than the DE algorithm for the test problems considered. This could be due to

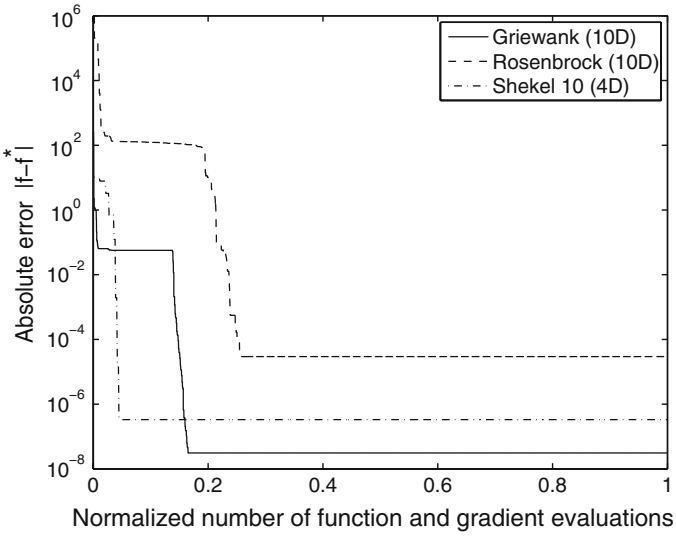

Fig. 2 Convergence history for the SF algorithm on three selected problems

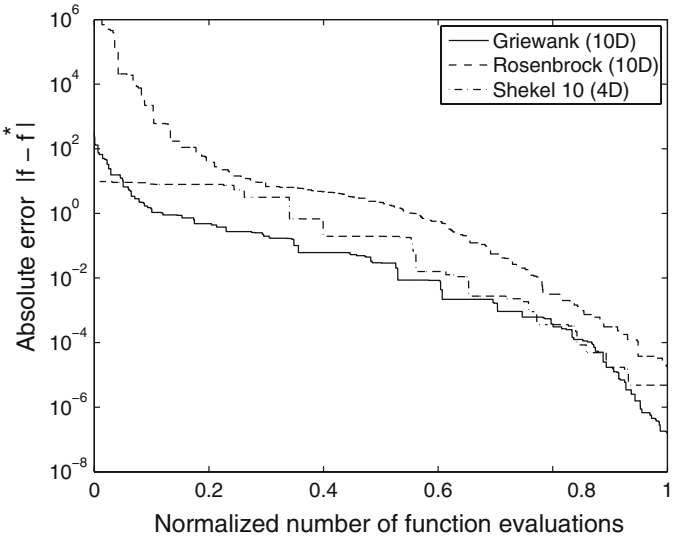

Fig. 3 Convergence history for the DE algorithm on three selected problems

the DE algorithm that requires random number generation in each iteration, and the repeated mutation operation to ensure that the trial point falls within the allowed search space.

Further insight into the different nature of the two methods may be obtained by inspection of typical global convergence histories of the respective methods for different problems. As representative, we selected the GW(10), RB(10) and S10(4) problems. Convergence histories for typical runs of the methods for each of these problems are plotted in Figs. 2 and 3 respectively. The number of function evaluations axis was normalized in these figures, to highlight the similarity of the convergence histories between different problems. Figure 2

illustrates that the SF algorithm rapidly obtains a relatively low function value, and then the remainder of computational effort is spent to confirm that what has been found to date is the global minimum (within probability q^*). The DE algorithm on the other hand gradually improves on the global best function value until the convergence criterion is met.

In a second and perhaps fairer comparison of the efficiency of the trajectory method to obtain a low local minimum relative to the performance of the DE algorithm, we evaluate the performance of the algorithms when we shoot for the known global target value of f^*. Such a test can be considered realistic in the sense that in many practical cases the target objective function value may be known, e.g. $f^* = 0$, or it is required to obtain a solution which corresponds to some low target function value. In the implementations we terminate an algorithm once it reaches a point where the absolute error $|f^c - f^*| < 9 \times 10^{-4}$, i.e. when a function value is obtained which is identical to the target optimal solution to at least three decimal digits. For the trajectory method this test is done at convergence for a given random starting point (at the end of an iteration). Otherwise the parameters specified are the same as before, except that no q^* need be specified. Again we record the average, over 100 runs, of the number of function-cum-gradient-vector evaluations ($No.f\mathbf{g}$) required to reach the prescribed target for each problem in the standard set of test problems. These values together with the success rates are listed in Table 3. For the trajectory algorithm we again also record the average relative absolute function value error, $r_f := |f^c - f^*|/(1 + |f^*|)$ at termination.

In Table 3 we compare the results for the SF algorithm with the average number of function evaluations ($No.f$) required for the DE algorithm. As before, we use a population size $N := 10n$, and the optimal DE parameters F and Cr are again determined by running the DE for all possible combinations between 0.2 and 0.9, using intervals of 0.1. The average values listed in the table are taken over successful runs only.

Table 3 shows that the SF algorithm is extremely robust in successfully obtaining the global optimum for 44 of the 48 problems. Again, for the tolerances specified, the convergence to the global optimum is sharp as shown in the column listing the relative function value errors, r_f. Using CPU time as comparative measure, the SF algorithm outperforms the DE method in 34 of the problems. For seven of these 34 problems, the SF algorithm requires more than a factor 10 less CPU time. Admittedly, the DE algorithm is also more than a factor 10 faster than the SF algorithm for 5 of the test problems.

4 Potential energy minimization

The physics upon which the SF algorithm is based would suggest that the SF algorithm is well suited to problems concerned with potential energy minimization. To illustrate this, we use the SF algorithm to obtain the lowest energy structures of Lennard-Jones clusters [19]. The problem was solved for the number of atoms N_a ranging from 3 to 30, and for $N_a = 38$. To suppress rigid body motion, the first atom is fixed at the origin, the second atom is constrained along the x-axis and the third atom is constrained to lie in the x–y plane. Therefore, the total number of variables is $3N_a - 6$. For this problem, the required gradients are computed analytically, which is significantly more efficient than finite difference computations. Exactly the same SF parameter settings were used as before, except the parameter $maxit$ (maximum number of random starting points) was increased from 5,000 to 1 million. The random initial positions are generated within a cube of side lengths four, centered around the origin. Whenever any coordinate magnitude exceeds two, a quadratic penalty is added to the cost function. This penalty is necessary to prevent the optimization algorithm from locating low energy solutions that consist of a number of smaller clusters.

Table 3 Comparison of SF with DE (global function value termination)

Problem	Snyman–Fatti			Differential evolution					r_{cpu}
	No. fg	sr	r_f	C_r	F	No. f	sr	r_f	
ACK(10)	111,654	55	6.8e − 05	0.7	0.2	12,399	100	8.0e − 04	43.86
AP(2)	154	100	6.3e − 05	0.8	0.3	336	99	2.6e − 04	0.38
BL(2)	5	100	3.6e − 10	0.9	0.3	356	100	4.1e − 04	0.26
B1(2)	283	100	3.7e − 07	0.9	0.4	609	100	4.1e − 04	0.51
B2(2)	190	100	1.5e − 07	0.9	0.4	589	100	4.1e − 04	0.40
BR(2)	94	100	2.9e − 07	0.9	0.4	438	100	3.0e − 04	0.27
CB3(2)	173	100	2.5e − 07	0.9	0.4	320	100	4.0e − 04	0.39
CB6(2)	159	100	1.4e − 05	0.7	0.2	367	100	2.1e − 04	0.36
CM(2)	138	100	7.6e − 07	0.9	0.3	258	100	3.7e − 04	0.63
DA(2)	375	100	2.7e − 08	0.8	0.4	886	100	0.0e + 00	0.24
EP(2)	133	100	1.7e − 06	0.8	0.3	441	100	2.0e − 04	0.47
EM(10)	–	0	–	0.2	0.4	149,485	100	5.3e − 05	–
EXP(10)	41	100	1.3e − 07	0.8	0.2	3,855	100	3.5e − 04	0.03
GP(2)	259	100	1.8e − 05	0.9	0.5	507	100	1.0e − 04	0.38
GW(10)	9,073	100	2.0e − 06	0.2	0.2	26,566	100	6.8e − 04	1.51
GRP(3)	2,960	100	3.4e − 04	0.9	0.5	723	100	5.7e − 04	15.74
H3(3)	220	100	2.8e − 07	0.8	0.4	543	100	1.0e − 04	0.90
H6(6)	212	100	4.0e − 07	0.3	0.3	3,988	100	1.5e − 04	0.19
HV(3)	217	100	5.0e − 06	0.9	0.5	1,597	100	4.6e − 04	0.16
HSK(2)	69	100	2.6e − 06	0.8	0.3	239	100	1.2e − 04	0.42
KL(4)	119	100	3.1e − 05	0.6	0.6	100	100	4.3e − 04	1.25
LM1(3)	472	100	1.8e − 06	0.7	0.2	668	100	5.4e − 04	1.02
LM2(10)	19,075	100	2.1e − 06	0.7	0.2	5,368	100	7.3e − 04	15.77
MC(2)	84	100	2.7e − 05	0.9	0.4	262	100	1.6e − 04	0.67
MR(3)	6,122	100	4.7e − 04	0.9	0.4	429	100	5.1e − 04	12.35
MCP(4)	202	100	3.2e − 05	0.9	0.4	340	100	4.4e − 04	0.88
ML(5)	919	100	1.9e − 07	0.9	0.8	10,920	100	3.2e − 04	0.24
MRP(2)	296	100	3.2e − 05	0.3	0.9	4,374	81	4.6e − 04	0.05
MGP(2)	1,800	100	3.0e − 06	0.2	0.9	3,363	97	2.0e − 04	0.82
NF2(4)	14,645	100	4.1e − 04	0.7	0.5	28,390	100	6.6e − 04	1.47
NF3(10)	198	100	6.9e − 09	0.9	0.4	17,284	100	3.0e − 06	0.01
PP(10)	96	100	4.5e − 04	0.6	0.2	6,811	100	1.3e − 05	0.17
PRD(2)	1,061	100	5.1e − 08	0.7	0.5	863	100	2.5e − 04	1.40
PQ(4)	396	100	5.8e − 05	0.9	0.4	1,615	100	5.2e − 04	0.18
RG(10)	–	0	–	0.2	0.2	23,328	100	7.3e − 04	–
RB(10)	2,917	100	1.2e − 04	0.9	0.6	61,063	100	7.5e − 04	0.07
SAL(5)	2,229	100	5.1e − 06	0.2	0.3	297,559	27	7.0e − 04	0.01
SF1(2)	5,132	100	4.1e − 07	0.2	0.5	13,458	93	4.6e − 04	0.30
SF2(2)	85,025	58	8.0e − 04	0.8	0.5	2,297	100	8.1e − 04	55.24
SBT(2)	882	100	1.3e − 07	0.9	0.5	1,854	100	2.0e − 06	0.80
SWF(10)	–	0	–	0.4	0.2	15,528	100	0.0e + 00	–

Table 3 continued

Problem	Snyman–Fatti			Differential evolution					r_{cpu}
	No. fg	sr	r_f	C_r	F	No. f	sr	r_f	
S5(4)	154	100	2.9e − 04	0.8	0.7	3,881	100	8.3e − 05	0.07
S7(4)	173	100	2.7e − 04	0.8	0.5	2,563	100	7.1e − 05	0.12
S10(4)	211	100	3.9e − 04	0.7	0.4	2,477	100	1.1e − 04	0.18
FX(5)	2,538	100	2.7e − 06	0.2	0.8	155,652	49	4.9e − 05	0.06
SIN(20)	208	100	4.2e − 06	0.8	0.2	23,044	100	1.7e − 04	0.10
ST(9)	–	0	–	0.9	0.5	29,074	100	4.3e − 04	–
WP(4)	683	100	6.3e − 05	0.9	0.6	5,461	100	6.0e − 04	0.09

The numerical results are summarized in Table 4, where the number of atoms (N_a), the average CPU time, average relative absolute function value error (r_f), the average required number of function-cum-gradient-vector evaluations (fg) and the average number of starting points required are listed for each cluster. Note that a 100% success rate was achieved on all these problems, where the global function value termination criterion was used. A run terminated as soon as a local minimum was located with a function value within 0.01 of the known published minimum. Because the final trajectory is allowed to converge to the local minimum with great accuracy the actual difference found in practise is significantly smaller (see listings of r_f values in Table 4). Especially noteworthy is the location of the global minimum for the 38 atom cluster. This structure is a face-centered-cubic (fcc) truncated octahedron, and according to Wales and Doye [19] presents the "first hurdle" for any global optimization technique in potential energy surface (PES) minimization. On average more than 200 thousand starting points are required before a starting point is generated that converges to the global minimum. This seems large, but compared to the estimated number of local minima (10^{11}) it is insignificant. Note that no tuning was performed on the SF algorithm to solve these problems; the standard settings were used throughout.

Limited testing was performed using the target probability termination criterion, since this criterion requires significantly more computational resources before convergence is obtained. As an example, the 30 atom cluster requires 29.3 million function-cum-gradient evaluations on average to terminate based on a 0.99 target probability. The success rate was 95 out of 100 runs. This is significantly more demanding than the 4.9 million function-cum-gradient evaluations required on average when termination is based on a known global minimum.

Comparing the DE algorithm to the SF algorithm for problems of this type is heavily biased against the DE algorithm. The update mechanisms in classical DE is not well suited to this type of problem. Nevertheless, results are available in the paper by Moloi and Ali [20], where more than 38 million function evaluations are reported to solve $N_a = 10$, where at worst the SF algorithm requires 228384 function evaluations (assuming finite difference gradient evaluation). Even a DE algorithm tailor-made for this type of problem (the topographical DE) requires more than 7 million function evaluations [21].

5 Conclusion

Based on numerical tests performed here, using an extensive set of standard test problems, the (slightly modified) SF global optimization algorithm is reassessed and found to be a reli-

Table 4 Performance of the SF algorithm on Lennard-Jones problems (global function value termination)

N_a	CPU(s)	r_f	No. fg	No. starts
3	0.0002	9.4e − 07	273	1.4
4	0.0003	4.0e − 06	432	1.3
5	0.0006	6.0e − 07	577	1.3
6	0.0057	3.2e − 06	5,408	10.8
7	0.0049	9.3e − 07	3,368	6.1
8	0.0035	1.6e − 06	1,881	3.0
9	0.0087	3.7e − 06	3,859	6.1
10	0.0259	2.5e − 06	9,516	14.8
11	0.0432	2.2e − 06	13,272	19.5
12	0.0432	4.4e − 06	11,061	15.6
13	0.0731	1.3e − 06	16,018	22.3
14	0.0362	1.5e − 06	6,809	8.9
15	0.0666	8.4e − 06	11,016	13.9
16	0.1246	5.1e − 06	18,387	23.2
17	0.4083	9.8e − 06	53,653	67.2
18	2.3705	1.5e − 05	279,742	346.9
19	1.2057	3.1e − 06	125,462	152.4
20	0.9144	4.4e − 06	88,066	103.6
21	4.5257	5.1e − 06	396,907	459.0
22	1.8587	2.4e − 06	147,852	165.4
23	5.6617	1.7e − 06	412,942	453.1
24	4.1844	2.5e − 06	280,793	298.8
25	5.7582	2.9e − 06	358,753	373.2
26	34.3503	2.1e − 06	1,980,050	2,012.1
27	25.2900	1.7e − 06	1,350,630	1,337.6
28	27.7834	1.5e − 06	1,372,260	1,327.4
29	77.1493	2.3e − 06	3,546,520	3,348.9
30	113.4188	2.6e − 06	4,921,010	4,527.8
38	10049.5545	5.1e − 07	275,062,479	214,730.5

ably robust and competitive method compared to the differential evolution (DE) algorithm. The latter algorithm is representative of popular evolutionary global optimization methods currently in wide usage. The implementation of the current SF code requires relatively few parameter settings and, without any tuning, the algorithm was successfully applied over a wide range of problems using standard parameter settings (mainly a set of three convergence tolerances). The results of further numerical experiments also indicate that the SF trajectory algorithm is particularly promising for solving minimum potential energy problems, to locate the structure of atomic and molecular clusters.

Although here we considered only the application of the SF algorithm to unconstrained problems, it may easily be adapted to handle constrained problems. This is done by transforming the constrained problem to an unconstrained penalty function minimization problem in the standard manner, and then applying the SF algorithm to determine the global unconstrained optimum of the penalty function. Once this point is determined it is used to determine the set of active constraints. With the unconstrained global minimizer of the penalty function

as single starting point, the intersection of the active constraints is obtained by applying the SF algorithm to the minimization of the sum of the squares of the active constraint function values. This point of intersection of the active constraints may be taken as the global optimum of the constrained problem. This application to practical constrained problems has [22] and is current successfully being applied by the authors, and the further results will be reported elsewhere.

References

1. Snyman, J.A., Fatti, L.P.: A multi-start global minimization algorithm with dynamic search trajectories. J. Optim. Theory Appl. **54**, 121–141 (1987)
2. Kam, T.Y., Snyman, J.A.: Optimal design of laminated composite plates using a global optimization technique. Compos. Struct. **19**, 351–370 (1991)
3. Kam, T.Y., Chang, R.R.: Optimal lay-up of thick laminated composite plates for maximum stiffness. J. Eng. Optim. **19**, 233–249 (1992)
4. Kam, T.Y., Lai, F.M., Liao, S.C.: Minimum weight design of laminated composite plates subject to strength constraint. AIAA J. **34**, 1699–1708 (1996)
5. Chang, R.R.: Optimal design of laminated composite torque wrench. Comput. Struct. **79**, 703–713 (2001)
6. Lee, C.R., Kam, T.Y.: System identification of partially restrained composite plates using measured natural frequencies. J. Eng. Mech. **132**, 841–850 (2006)
7. Laskari, E.C., Parsopoulos, K.E., Vrahatis, M.N.: Evolutionary operators in global optimization with dynamic search trajectories. Numer. Algorithms **34**, 393–403 (2003)
8. Greenspan, D.: Discrete Models. Addison-Welseley (1973)
9. Stillinger, F.H., Weber, T.A.: Computer simulation of local order in condensed phases of silicon. Phys. Rev. B **31**, 5262–5271 (1985)
10. Adler, B.J., Wainwright, T.E.: Phase transitions for a hard sphere system. J. Chem. Phys. **27**, 1208–1209 (1957)
11. Hockney, R.W.: The potential calculation and some applications. Methods Comput. Phys. **9**, 136–211 (1970)
12. Verlet, L.: Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys. Rev. **159**, 98–103 (1967)
13. Snyman, J.A.: A new and dynamic method for unconstrained minimization. Appl. Math. Model. **6**, 449–462 (1982)
14. Snyman, J.A.: An improved version of the original leap-frog dynamic method for unconstrained minimization LFOP1(b). Appl. Math. Model. **7**, 216–218 (1983)
15. Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J. Global Optim. **31**, 635–672 (2005)
16. Storn, R., Price, K.: Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
17. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Minimization. Springer, Berlin (2005)
18. Ali, M.M., Fatti, L.P.: A differential free point generation scheme in the differential evolution algorithm. J. Glob. Optim. **35**, 551–572 (2006)
19. Wales, D.J., Doye, J.P.K.: Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 100 atoms. J. Phys. Chem. A. **101**, 5111–5116 (1997)
20. Moloi, N.P., Ali, M.M.: An iterative global optimization algorithm for potential energy minimization. Comput. Optim. Appl. **30**, 119–132 (2005)
21. Ali, M.M., Smith, R., Hobday, S.: The structure of atomic and molecular clusters, optimised using classical potentials. Comput. Phys. Commun. **175**, 451–464 (2006)
22. Snyman, J.A., Geerthsen, K.A.: The practical application of a dynamic search-trajectory method for constrained global optimization. In: Bestle, D., Schielen, W. (eds.) Proceedings of the IUTAM Symposium on Optimization of Mechanical Systems, Kluwer, Dordrecht (1995)