Soft Comput (2017) 21:6331–6350
DOI 10.1007/s00500-016-2187-x

METHODOLOGIES AND APPLICATION

Gradient subspace approximation: a direct search method for memetic computing

Oliver Schütze[1] · **Sergio Alvarado**[1] · **Carlos Segura**[2] · **Ricardo Landa**[3]

Published online: 25 May 2016

Abstract The hybridization of evolutionary algorithms and local search techniques as, e.g., mathematical programming techniques, also referred to as memetic algorithms, has caught the interest of many researchers in the recent past. Reasons for this include that the resulting algorithms are typically robust and reliable since they take the best of both worlds. However, one crucial drawback of such hybrids is the relatively high cost of the local search techniques since many of them require the gradient or even the Hessian at each candidate solution. Here, we propose an alternative way to compute search directions by exploiting the neighborhood information. That is, for a given point within a population $\mathcal{P}$, the neighboring solutions in $\mathcal{P}$ are used to compute the most greedy search direction out of the given data. The method is hence particularly interesting for the usage within population-based search strategies since the search directions come ideally for free in terms of additional function evaluations. In this study, we analyze the novel method first as a stand-alone algorithm and show further on its benefit as a local searcher within differential evolution.

Communicated by V. Loia.

✉ Oliver Schütze
schuetze@cs.cinvestav.mx

[1] Computer Science Department, Cinvestav-IPN, Mexico City, Mexico

[2] Centro de Investigación en Matemáticas, Área Computación Callejón Jalisco s/n, Mineral de Valenciana, 36240 Guanajuato, Guanajuato, Mexico

[3] Information Technology Laboratory, Cinvestav-IPN Tamaulipas, Victoria, Tamaulipas, Mexico

Keywords Scalar optimization · Direct search · Memetic computing · Evolutionary algorithms · Differential evolution

1 Introduction

Set-based heuristics such as evolutionary algorithms (EAs) have been widely used for the numerical treatment of scalar optimization problems in the last decades (Schwefel 1993; Beyer and Schwefel 2002; Bäck and Schwefel March 1993; Eiben and Smith 2003; Osyczka and Krenich 2006). Reasons for this include the general applicability and robustness of such methods and that they are in many cases able to reach the global solution of a given problem. Several different variants of EAs have been proposed. Among them, differential evolution (DE) (Storn and Price 1995; Kukkonen and Lampinen 2006) is an efficient variant initially designed for continuous optimization. From its inception, it has yielded remarkable results in several optimization competitions (Das and Suganthan 2011; Qin and Suganthan 2005; LaTorre et al. 2011) and in practical demanding applications (Junhua et al. 2014; Shiwen and Anyong 2005), which is why we decided to work with this specific variant of EA.

In spite of the numerous advantages of EAs (Eiben and Smith 2003), one drawback is that in many cases, they can suffer from slow convergence rates (Gong et al. 2006; Sivanandam and Deepa 2007). As a remedy, researchers have considered memetic strategies (Moscato 1989; Neri et al. 2012), i.e., EAs (or other set-based heuristics) coupled with local search strategies coming, e.g., from (but not limited to) mathematical programming (Caraffini et al. 2013, 2014). The latter ensures that every non-local optimal solution x_0 can be further improved in each step due to the existence of descent directions (i.e., a direction in which the objective can be improved from x_0) but comes with a certain additional cost. If, for instance, the gradient (those negative is the most greedy descent direction) is computed via forward differences (FD), the cost to obtain this vector is given by $O(n)$ additional function evaluations, where n is the dimension of the problem. Several different ways of integrating strategies have been devised. For this reason, different taxonomies to classify them have been proposed. Among them, the one presented by Talbi (2002) is probably one of the most popular one. In this taxonomy, the high-level teamwork hybrid (HTH) term refers to schemes where several self-contained algorithms perform a search in parallel, whereas the high-level relay hybrid (HRH) term refers to schemes where several self-contained algorithms are executed in a sequence. In this work, we have adopted the multiple offspring sampling (MOS) framework (LaTorre 2009). The MOS framework allows the development of both HTH and HRH schemes.

Here, we present a novel method of the most greedy search direction that can be computed via exploiting the existing neighborhood information. More precisely, given points x_i, $i = 1, \ldots, r$, that are near x_0, whose function values are known, the idea is not only to perform a search into one of the discrete directions $\nu_i := (x_i - x_0)$ as done, e.g., in DE or particle swarm optimization (PSO), but to compute the most promising (here: most greedy) direction within $W := span\{\nu_1, \ldots, \nu_r\}$, i.e., within the subspace spanned by the ν_i's. As we will see further on, this direction is equal to the orthogonal projection of the gradient onto W, whereby the gradient is not explicitly computed. Thus, already for $r \geq 1$ neighboring solutions, one typically obtains a descent direction and for $r = n$ an approximation for the gradient. The latter is as for FD; however, the advantage of the gradient subspace approximation (GSA) is that there is a higher degree of freedom for the choice of the neighboring solutions x_i, whereas for FD, the difference vectors $x_i - x_0$ have to be coordinate directions. Another advantage is that the GSA can be directly adapted to constrained problems as we will show in this paper.

The new approach to obtain greedy search directions is particularly advantageous when integrating it into set-based heuristics since such methods provide the required neighborhood information, and thus, an approximation of the descent direction comes ideally 'for free' in terms of additional function evaluations. Figure 1 shows the averaged number of neighboring solutions r of the ten best found individuals within a run of DE/rand/1/bin (averaged over 30 independent runs). In the top, we see the number of evaluations versus r for the Rosenbrock function, whereas in the bottom, the Ackley function (both for $n = 10$) is considered. In both cases, we show the values for the neighborhood sizes $\delta \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$ (here, we have used the 2-norm to define the neighborhood). Since the number of neighbors quickly increases, it is hence possible to obtain for both problems approximations of the gradients of the best found individuals via GSA without additional function evaluations after only a few generations.

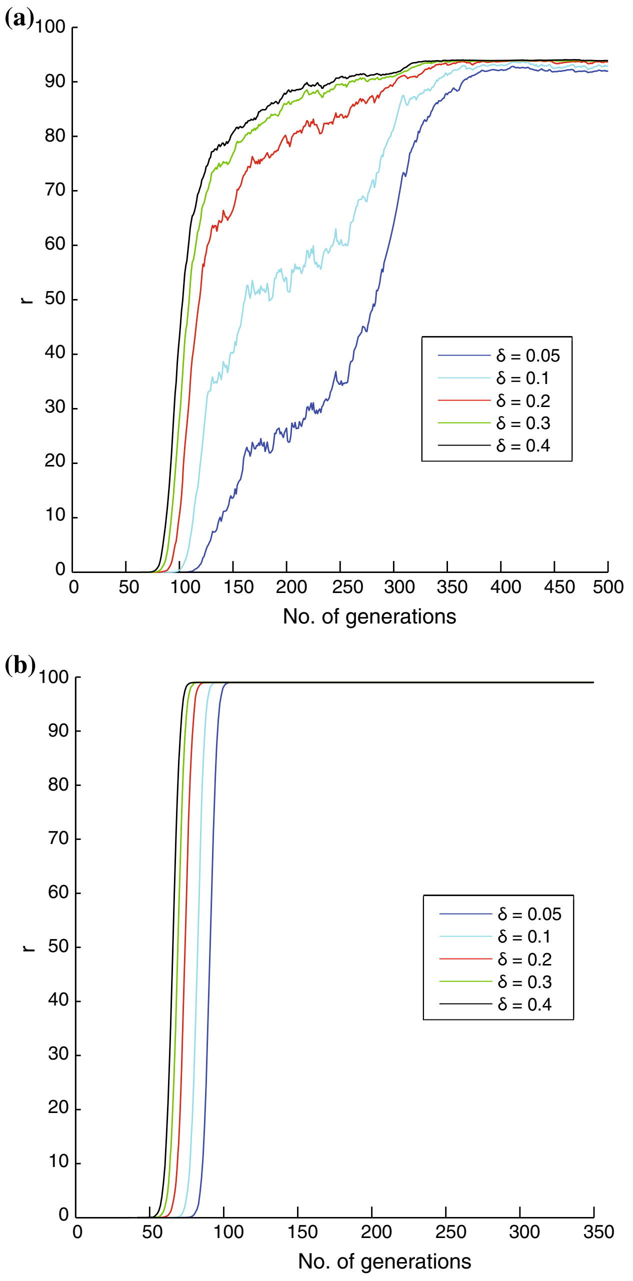

Fig. 1 Number r of neighbors of the ten best individuals within a DE run for different neighborhood sizes on two benchmark problems with dimension $n = 10$ of the decision space. Results are averaged over 30 independent runs. **a** Rosenbrock **b** Ackley

The focus of this work is (a) to introduce GSA and (b) to show its potential as a local search engine within a memetic algorithm. As demonstrator for the latter, we have made a first attempt to integrate GSA into DE. The resulting memetic strategy already shows significant improvements over its base algorithm.

The remainder of this paper is organized as follows: In Sect. 2, we briefly state the required background and give an overview of the related work. In Sect. 3, we present the GSA for both unconstrained and constrained problems, and the related stand-alone algorithm. In Sect. 4, we present a memetic variant of DE, called DE/GSA. In Sect. 5, we present some numerical results on the GSA stand-alone algorithm and DE/GSA. Finally, we conclude and give paths for future research in Sect. 6.

2 Background and related work

In the following, we consider scalar optimization problems of the form

$$\begin{aligned} &\min_x \; f(x) \\ &\text{s.t. } g_i(x) \leq 0, \; i = 1, \ldots, p \\ &h_j(x) = 0, \; i = 1, \ldots, m, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is called the objective function, the functions $g_i : \mathbb{R}^n \to \mathbb{R}$ are inequality constraints, and the functions $h_j : \mathbb{R}^n \to \mathbb{R}$ are equality constraints. A feasible point x (i.e., x satisfies all inequality and equality constraints) is called a solution to (1) if there exists no other feasible point y that has a lower objective value.

Throughout this paper, we will consider the gradients of the objective as well as of the constraint functions, e.g.,

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right)^T \in \mathbb{R}^n \tag{2}$$

denotes the gradient of f at x. However, we stress that this is for purpose of the analysis of the approach. The resulting algorithms are gradient-free, and it is not required that the gradients of the objective or the constraint functions are given in analytic form.

A vector $v \in \mathbb{R}^n$ is called a *descent direction* for f at x_0 if $\langle \nabla f(x_0), v \rangle < 0$. In that case, it holds for sufficiently small step sizes $t > 0$ that $f(x_0 + tv) < f(x_0)$.

The idea to compute/approximate the gradient—which is the most greedy direction among all search directions in $\mathbb{R}^n$—is clearly not new. The most prominent way to approximate the gradient via sampling is the finite difference (FD) method (e.g., Nocedal and Wright 2006). GSA also uses a certain finite difference approach, but the difference is that GSA accepts in principle samples in all directions, whereas 'classical' FD requires samples in coordinate directions. Thus, GSA is more suited to population-based algorithms since neighboring individuals are typically not aligned in coordinate directions.

There exist some works where these difference vectors do not have to point in orthogonal directions to estimate the gradients. In the response surface methodology (RSM), for instance, the objective f is replaced by low-order polynomials $\tilde{f}$ (typically of degree one and two); those gradients are approximated using least-squares techniques (Kleijnen 2015). If a first-order model is chosen, the match of the gradients $\nabla f(x)$ and $\nabla \tilde{f}(x)$ is typically quite good for a nonlinear function f only if x is sufficiently far away from the optimum. For second-order models, the match is in general much better, and however, it comes with the cost that n^2 parameters have to be fitted at every point x. Further works that can utilize scattered samples can be found in Hazen and Gupta (2006), Segovia Domínguez et al. (2014). In Hazen and Gupta (2006), a least-squares regression is performed, while in Segovia Domínguez et al. (2014), statistical expectation is used. In both works, the authors restrict themselves to unconstrained problems. The same approximation idea as for GSA can be found in Brown and Smith (2005), where the authors use the technique to approximate the Jacobian of the objective map of a multi-objective optimization problem. This work is also restricted to the unconstrained case, and no discussion is given on details as step size control and integration into global search heuristics. Further, neighborhood exploration similar to GSA is used in Lara et al. (2010), Schütze et al. (2015), again in the context of multi-objective optimization. In these works, the method is not used to approximate the Jacobian of the objective map, but to compute a search direction with a certain steering property. Next, there is the possibility to use automatic differentiation (AD) to numerically evaluate the gradient if the function is specified by a computer program (Griewank 2000). These methods, however, can not be applied if the objective is only provided in form of binary computer code.

The resulting GSA stand-alone algorithm—obtaining a greedy search direction via sampling together with a line search in that direction—can be seen as a variant of the well-known pattern search method (Hooke and Jeeves 1961). The difference is also here the choice of the sampling points which makes GSA more flexible and cheaper (in terms of function evaluations) as local searcher within population-based search algorithms.

Finally, pattern search has already been used as local searchers within MA. For instance, in Zapotecas Martínez and Coello Coello (2012) in the context of memetic multi-objective optimization and in Bao et al. (2013) for the parameter tuning of support vector machines.

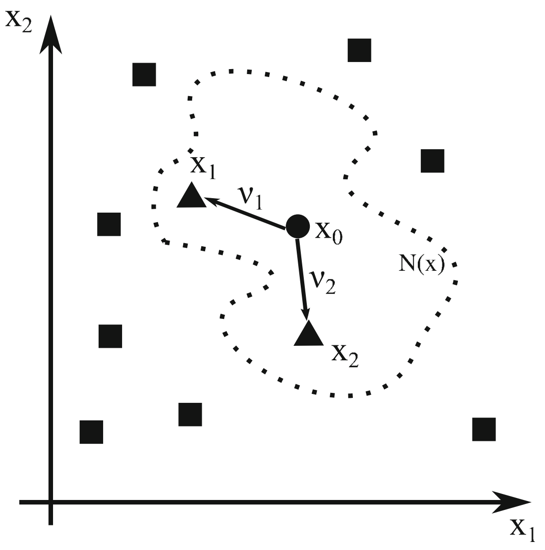

Fig. 2 Basic idea of the GSA: to exploit the neighborhood information out of a given population

3 Gradient subspace approximation

In the following, we present the GSA for both unconstrained and constrained SOPs. The general idea behind the method is as follows (compare to Fig. 2): Given a point x_0 that is designated for local search as well as a further point x_i in the current population (i.e., its objective value is known) that is in the vicinity of x_0, the given information can be used to approximate the directional derivative in direction

$$\nu_i := \frac{x_i - x_0}{\|x_i - x_0\|_2} \tag{3}$$

without any additional cost (in terms of function evaluations). To be more precise, it holds

$$f'_{\nu_i}(x_0) = \langle \nabla f(x_0), \nu_i \rangle = \frac{f(x_i) - f(x_0)}{\|x_i - x_0\|_2} + O(\|x_i - x_0\|_2), \tag{4}$$

where O denotes the Landau symbol. The above estimation can be seen by considering the forward difference quotient on the one-dimensional line search function $f_{\nu_i}(t) = f(x_0 + t\nu_i)$.

3.1 Unconstrained problems

Assume we are given a problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{5}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, and a point $x_0 \in \mathbb{R}^n$ with $\nabla f(x_0) \neq 0$. It is known that the greedy search direction at x_0 is given by

$$g := -\frac{\nabla f(x_0)}{\|\nabla f(x_0)\|_2}, \tag{6}$$

i.e., the direction $\nu \in \mathbb{R}^n$ in which the strongest decay with respect to f can be obtained. For our purpose, it is advantageous to write g as the solution of the following problem:

$$\begin{aligned} &\min_{\nu \in \mathbb{R}^n} \langle \nabla f(x_0), \nu \rangle \\ &\text{s.t. } \|\nu\|_2^2 = 1. \end{aligned} \tag{7}$$

Now, we examine the situation that we wish to compute the most greedy direction by exploiting the neighborhood information. For this, we will first consider the idealized case. That is, assume we are given in addition to x_0 the directions $\nu_1, \ldots, \nu_r \in \mathbb{R}^n$, $r \in \mathbb{N}$, and the directional derivatives $\langle \nabla f(x_0), \nu_i \rangle$, $i = 1, \ldots, r$. The most greedy search direction in the span of the given directions,

$$\nu \in span\{\nu_1, \ldots, \nu_r\}, \tag{8}$$

can be obtained by solving the following problem:

$$\begin{aligned} &\min_{\lambda \in \mathbb{R}^r} \left(\langle \nabla f(x_0), \sum_{i=1}^r \lambda_i \nu_i \rangle = \sum_{i=1}^r \lambda_i \langle \nabla f(x_0), \nu_i \rangle \right) \\ &\text{s.t. } \left\| \sum_{i=1}^r \lambda_i \nu_i \right\|_2^2 - 1 = \lambda^T V^T V \lambda - 1 = 0, \end{aligned} \tag{9}$$

where

$$V = (\nu_1, \ldots, \nu_r) \in \mathbb{R}^{n \times r}. \tag{10}$$

If $\lambda^* \in \mathbb{R}^r$ is a solution of (9), then we set

$$\nu^* := \sum_{i=1}^r \lambda_i^* \nu_i = V\lambda^* \tag{11}$$

as associated search direction.
The Karush–Kuhn–Tucker (KKT) system of (9) reads as

$$\nabla_\lambda L(\lambda, \mu) = \begin{pmatrix} \langle \nabla f(x_0), \nu_1 \rangle \\ \vdots \\ \langle \nabla f(x_0), \nu_r \rangle \end{pmatrix} + 2\mu V^T V \lambda = 0 \tag{12}$$

$$h(\lambda) = \lambda^T V^T V \lambda - 1 = 0. \tag{13}$$

Apparently, Eq. (13) is only used for normalization. If we omit this equation and the factor $2\,\mu$ in (12), we can rewrite (12) as the following normal equation system

$$V^T V \lambda = -V^T \nabla f(x_0), \tag{14}$$

and directly obtain the following result.

Proposition 1 *Let $\nu_1, \ldots, \nu_r \in \mathbb{R}^n$, $r \leq n$, be linearly independent and*

$$\tilde{\lambda}^* := -(V^TV)^{-1}V^T\nabla f(x_0). \tag{15}$$

Then,

$$\lambda^* := \frac{\tilde{\lambda}^*}{\|V\lambda^*\|_2^2} \tag{16}$$

is the unique solution of (9)*, and thus,*

$$\nu^* = \frac{-1}{\|V\lambda^*\|_2^2}V(V^TV)^{-1}V^T\nabla f(x_0) \tag{17}$$

is the most greedy search direction in $span\{\nu_i, \ldots, \nu_r\}$.

Proof Follows by the above discussion setting $2\mu = \|V\lambda^*\|_2^2$ and since the Hessian of the Lagrangian

$$\nabla^2_{\lambda\lambda}L(\lambda, \mu) = V^TV \tag{18}$$

is positive definite. □

Before we come to a realization of (9) without explicitly using $\nabla f(x_0)$, we first discuss some properties of (9).

- Let λ^* be a solution of (9) and

$$\xi(\lambda) := \sum_{i=1}^r \lambda_i\langle\nabla f(x_0), \nu_i\rangle, \tag{19}$$

  then $\xi(\lambda^*) \leq 0$. To see this, assume that for a solution λ^* it holds $\xi(\lambda^*) > 0$. Then, $\tilde{\lambda} := -\lambda^*$ is also feasible, and it holds $\xi(\tilde{\lambda}) = -\xi(\lambda^*) < 0$.
  Thus, if there exists a direction ν_i, $i \in \{1, \ldots, r\}$, such that $\langle\nabla f(x_0), \nu_i\rangle \neq 0$, then $\xi(\lambda^*) < 0$, and hence, the related direction ν^* is a descent direction. Further, note that for a randomly chosen direction $\nu_i \in \mathbb{R}^n$ the probability is zero that $\langle\nabla f(x_0), \nu_i\rangle = 0$. Hence, the probability is one to obtain a descent direction via (9) using randomly chosen directions.
- The more directions are incorporated into (9), the better (i.e., more greedy) the resulting search direction gets. To be more precise, let λ_r^* be the solution of (9) using the directions $\nu_1, \ldots, \nu_r$, and λ_{r+l}^* be the solution of (9) where the search directions $\nu_1, \ldots, \nu_r, \nu_{r+1}, \ldots, \nu_{r+l}$ are used, then $\xi(\lambda_{r+l}^*) \leq \xi(\lambda_r^*)$. For $r = n$ and if the ν_i's are linearly independent, then ν^* coincides with g.
- The normal equation system (14) helps to understand the numerics of the problem. For the condition number of the matrix in (14), it holds if the rank of V is maximal that $\kappa_2(V^TV) = \kappa(V)^2$. That is, when choosing (or selecting) directions ν_i, the condition number has to be checked. In particular, directions that point nearly in the same direction have to be avoided.
- The above consideration shows that orthogonal directions are strongly preferred. In that case, we obtain

$$\nu^* = \frac{-1}{\|\lambda^*\|_2^2}VV^T\nabla f(x_0), \tag{20}$$

  i.e., the orthogonal projection of $\nabla f(x_0)$ onto $span\{\nu_1, \ldots, \nu_r\}$. Hence, ν^* can be seen as the best approximation of the most greedy search direction g in the subspace $span\{\nu_1, \ldots, \nu_r\}$.

Now, we come to the gradient-free realization of (9). Assume we are given next to x_0 further sample points $x_1, \ldots, x_r$ in a neighborhood of x_0 whose objective values $f(x_i)$, $i = 1, \ldots, r$, are known. Following the above discussion, we can thus use

$$\tilde{\nu}_i := \frac{x_i - x_0}{\|x_i - x_0\|_2}, \quad d_i := \frac{f(x_i) - f(x_0)}{\|x_i - x_0\|_2}, \quad i = 1, \ldots, r. \tag{21}$$

Hence, instead of (9), we can solve

$$\begin{aligned}&\min_{\lambda\in\mathbb{R}^r} \lambda^Td\\ &\text{s.t. } \lambda^T\tilde{V}^T\tilde{V}\lambda - 1 = 0,\end{aligned} \tag{22}$$

where $\tilde{V} = (\tilde{\nu}_i, \ldots, \tilde{\nu}_r)$. The normal equation system (14) gets

$$\tilde{V}^T\tilde{V}\tilde{\lambda} = -d, \tag{23}$$

and the most greedy search direction can be approximated as

$$\tilde{\nu}^* = \frac{-1}{\|\tilde{V}\tilde{\lambda}\|_2^2}\tilde{V}(\tilde{V}^T\tilde{V})^{-1}d. \tag{24}$$

For the particular case that distinct coordinate directions are chosen, i.e.,

$$x_i = x_0 + t_ie_{j_i}, \; i = 1, \ldots, r, \tag{25}$$

where e_j denotes the j-th unit vector, we obtain for the j_i-th entry of $\tilde{\nu}^*$ (without normalization)

$$\tilde{\nu}_{j_i}^* = \frac{f(x_0 + t_ie_{j_i}) - f(x_0)}{|t_i|}. \tag{26}$$

That is, for $x_i = x_0 + t_ie_i$, $i = 1, \ldots, n$, the search direction coincides with the result of the forward difference method.

Example 1 Consider the problem $f : \mathbb{R}^5 \to \mathbb{R}$,

$$f(x) = \sum_{i=1}^{5} x_i^2 \tag{27}$$

and the point $x_0 = (1, 1, 1, 1, 1)^T$. It is $\nabla f(x_0) = (2, 2, 2, 2, 2)^T$ and $g = \frac{-1}{\sqrt{20}}(2, 2, 2, 2, 2)^T$ with $\langle \nabla f(x_0), g\rangle = -4, 4721$. Choosing the three orthogonal search directions

$$V^T = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{28}$$

we obtain $\langle \nabla f(x_0), \frac{\nu_1}{\|\nu_1\|}\rangle = 2.683$, $\langle \nabla f(x_0), \frac{\nu_2}{\|\nu_2\|}\rangle = 2.683$, and $\langle \nabla f(x_0), \frac{\nu_3}{\|\nu_3\|}\rangle = 2$. When solving problem (9) via (17), we obtain $\nu^* = (-0, 2798, -0, 5595, -0, 2798, -0, 4663, -0, 5595)^T$ with $\langle \nabla f(x_0), \nu^*\rangle = -4.2895$. When discretizing the above setting via choosing

$$x_i = x_0 + 0.1\nu_i, \quad i = 1, 2, 3, \tag{29}$$

we obtain via (23) the search direction $\tilde{\nu}^* = (-0, 2816, -0, 5632, -0, 2816, -0, 4549, -0, 5632)^T$ leading to $\langle \nabla f(x_0), \tilde{\nu}^*\rangle = -4.2892$ which is very close to the value for ν^*. Next, we consider the non-orthogonal search directions

$$V^T = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 0 \\ 0 & 1 & 2 & 2 & 1 \end{pmatrix} \tag{30}$$

with $\langle \nabla f(x_0), \frac{\nu_1}{\|\nu_1\|}\rangle = 2.683$, $\langle \nabla f(x_0), \frac{\nu_2}{\|\nu_2\|}\rangle = 3.795$, and $\langle \nabla f(x_0), \frac{\nu_3}{\|\nu_3\|}\rangle = 3.795$. These search directions lead to $\tilde{\nu}^* = (-0, 1640, -0, 6558, -0, 5902, -0, 2951, -0, 3279)^T$ with $\langle \nabla f(x_0), \nu^*\rangle = -4.0661$ for the idealized problem and to $\tilde{\nu}^* = (-0, 1757, -0, 6571, -0, 5960, -0, 2980, -0, 3056)^T$ with $\langle \nabla f(x_0), \tilde{\nu}^*\rangle = -4.0647$ for the discretized problem.

3.2 Constrained problems

One advantage of the approach in (9) is that it can be extended to constrained problems. In the following, we will separately consider equality and inequality constrained problems.

First, we assume we are given p equality constraints, i.e., the problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \; f(x) \\ &\text{s.t. } h_i(x) = 0, \; i = 1, \ldots, p, \end{aligned} \tag{31}$$

where each constraint $h_i : \mathbb{R}^n \to \mathbb{R}$ is differentiable. Hence, the greedy direction at x_0 can be expressed as

$$\begin{aligned} &\min_{\nu\in\mathbb{R}^n} \; \langle \nabla f(x_0), \nu\rangle \\ &\text{s.t. } \|\nu\|_2^2 = 1 \\ &\langle \nabla h_i(x), \nu\rangle = 0, \; i = 1, \ldots, p, \end{aligned} \tag{32}$$

and given the directions ν_i, $i = 1, \ldots, r$, the subspace optimization problem can be written as

$$\begin{aligned} &\min_{\lambda\in\mathbb{R}^r} \; \sum_{i=1}^{r} \lambda_i \langle \nabla f(x_0), \nu_i\rangle \\ &\text{s.t. } \lambda^T V^T V\lambda - 1 = 0 \\ &\sum_{i=1}^{r} \lambda_i \langle \nabla h_j(x_0), \nu_i\rangle = 0, \; j = 1, \ldots, p. \end{aligned} \tag{33}$$

The KKT system of (33) reads as follows

$$V^T \nabla f(x_0) + 2\mu_0 V^T V\lambda + (HV)^T \mu = 0 \tag{34}$$
$$HV\lambda = 0 \tag{35}$$
$$\lambda^T V^T V\lambda - 1 = 0, \tag{36}$$

where

$$H = \begin{pmatrix} \nabla h_1(x_0)^T \\ \vdots \\ \nabla h_p(x_0)^T \end{pmatrix} \in \mathbb{R}^{p\times n}. \tag{37}$$

If we apply the same 'normalization trick' as for (14), we can transform the KKT equations into

$$\begin{pmatrix} V^T V & H^T V^T \\ HV & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} -V^T \nabla f(x_0) \\ 0 \end{pmatrix} \tag{38}$$

which leads to the following result.

Proposition 2 *Let $\nu_1, \ldots, \nu_r \in \mathbb{R}^n$ be linearly independent where $p \le r \le n$, let rank$(H) = p$, and*

$$\begin{pmatrix} \tilde{\lambda}^* \\ \tilde{\mu}^* \end{pmatrix} = \begin{pmatrix} V^T V & H^T V^T \\ HV & 0 \end{pmatrix}^{-1} \begin{pmatrix} -V^T \nabla f(x_0) \\ 0 \end{pmatrix}, \tag{39}$$

then

$$\lambda^* := \frac{\tilde{\lambda}^*}{\|V\lambda^*\|_2^2} \tag{40}$$

is the unique solution of (33)*, and thus,*

$$\nu^* = \frac{-1}{\|V\lambda^*\|_2^2} V(V^T V)^{-1} V^T \nabla f(x_0) \tag{41}$$

is the most greedy search direction in span$\{\nu_i, \ldots, \nu_r\}$.

Proof To show that the matrix in (39) is regular, let $y \in \mathbb{R}^r$ and $z \in \mathbb{R}^p$ such that

$$\begin{pmatrix} V^T V & H^T V^T \\ HV & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = 0. \tag{42}$$

It follows that $HVy = 0$ and hence that

$$0 = \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} V^T V & H^T V^T \\ HV & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = yV^T Vy. \tag{43}$$

Thus, it is $y = 0$ since $V^T V$ is positive definite. Further, by (42), it follows that $V^T H^T z = 0$. Since $V^T \in \mathbb{R}^{n\times r}$ has rank $r \geq p$, it follows that $V^T H^T$ has rank p. This implies that also $z = 0$, and thus, that the matrix in (39) is regular.
The rest follows by the discussion above setting $2\mu_0 = \|\sum_{i=1}^r \tilde{\lambda}_i^* \nu_i\|_2^2$ and since the Hessian of the Lagrangian $\nabla^2_{\lambda\lambda} L(\lambda, \mu) = V^T V$ is positive definite. □

Now, we come to the gradient-free realization: Assume we are given next to x_0 the sample points $x_1, \ldots, x_r$ together with their objective values. Define $\tilde{\nu}_i$, $i = 1, \ldots, r$, and $\tilde{V}$ as above and

$$m_{ij} := \frac{h_i(x_j) - h_i(x_0)}{\|x_j - x_i\|_2}, \quad i = 1, \ldots, p, \; j = 1, \ldots, r. \tag{44}$$

Note that m_{ij} is an approximation of the entry $(HV)_{ij}$. Hence, by defining $\tilde{M} := (m_{ij}) \in \mathbb{R}^{p\times r}$, the KKT equation (38) gets

$$\begin{pmatrix} \tilde{V}^T \tilde{V} & \tilde{M}^T \\ \tilde{M} & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} -d \\ 0 \end{pmatrix} \tag{45}$$

which has to be solved.

Next, we consider inequality constrained problems of the form

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \; f(x) \\ &\text{s.t. } g_i(x) \leq 0, \; i = 1, \ldots, l, \end{aligned} \tag{46}$$

where $m \leq l$ constraints are active at x_0. The greedy search direction of this problem is—after a re-enumeration of the constraints—hence given by

$$\begin{aligned} &\min_{\nu\in\mathbb{R}^n} \; \langle \nabla f(x_0), \nu\rangle \\ &\text{s.t. } \|\nu\|_2^2 = 1 \\ &\langle \nabla g_i(x), \nu\rangle \leq 0, \; i = 1, \ldots, m. \end{aligned} \tag{47}$$

Thus, given the directions ν_i, $i = 1, \ldots, r$, the subspace optimization problem can be written as

$$\begin{aligned} &\min_{\lambda\in\mathbb{R}^r} \sum_{i=1}^r \lambda_i \langle \nabla f(x_0), \nu_i\rangle \\ &\text{s.t. } \lambda^T V^T V\lambda - 1 = 0 \\ &\sum_{i=1}^r \lambda_i \langle \nabla g_j(x_0), \nu_i\rangle \leq 0, \; j = 1, \ldots, m. \end{aligned} \tag{48}$$

One way to solve (48) numerically is to use active set methods (e.g., Nocedal and Wright 2006) together with the results on equality constrained problems. Another possible alternative would be to introduce slack variables. Doing so, Equation (48) can be re-formulated as

$$\begin{aligned} &\min_{(\lambda,s)\in\mathbb{R}^{r+m}} \sum_{i=1}^r \lambda_i \langle \nabla f(x_0), \nu_i\rangle \\ &\text{s.t. } \lambda^T V^T V\lambda - 1 = 0 \\ &\sum_{i=1}^r \lambda_i \langle \nabla g_j(x_0), \nu_i\rangle + s_j^2 = 0, \quad j = 1, \ldots, m. \end{aligned} \tag{49}$$

The resulting KKT equations read as

$$V^T \nabla f(x_0) + 2\mu_0 V^T V\lambda + V^T G^T \nu = 0 \tag{50}$$
$$S\mu = 0 \tag{51}$$
$$\lambda^T V^T V\lambda - 1 = 0 \tag{52}$$
$$GV\lambda + S^2 e = 0, \tag{53}$$

where

$$G = \begin{pmatrix} \nabla g_1(x_0)^T \\ \vdots \\ \nabla g_m(x_0)^T \end{pmatrix} \in \mathbb{R}^{m\times n}, \tag{54}$$
$$S = \text{diag}(s_1, \ldots, s_m) \in \mathbb{R}^{m\times m}, \text{ and} \tag{55}$$
$$e = (1, \ldots, 1)^T \in \mathbb{R}^m. \tag{56}$$

Hence, one can try to solve the above equation system of dimension $2r + 2m$ using, e.g., a Newton method.
Another possibility to find a solution to (48) is to utilize gradient projection methods. This works in particularly well if m is small and $r \gg m$. We first assume that we are given one inequality constraint (i.e., $m = 1$) and will consider the general case later. The classical approach is to perform an orthogonal projection of the solution ν^* of the unconstrained problem (9) to the orthogonal space of $\nabla g(x_0)$ as follows (compare to Fig. 3): given a QR decomposition of $\nabla g(x_0)$, i.e.,

$$\nabla g(x_0) = QR = (q_1, \ldots, q_n)R, \tag{57}$$

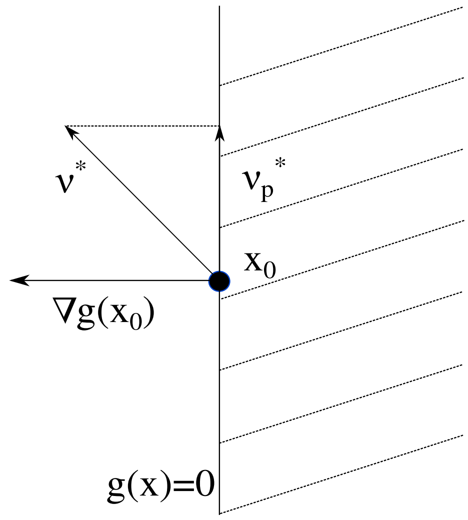

Fig. 3 Gradient projection in case one inequality constraint g is active at x_0

then the vectors $q_2, \ldots, q_n$ build an orthonormal basis of $\nabla g(x_0)^\perp$. Defining $Q_g = (q_2, \ldots, q_n)$, the projected vector is hence given by

$$\nu_{\text{new}} = Q_g Q_g^T \nu^*. \tag{58}$$

The problem that we have, however, is that $\nabla g(x_0)$ is not at hand. Instead, we propose to perform the following steps: define

$$M{:=}\nabla g(x_0)^T V = (\langle \nabla g(x_0), \nu_1\rangle, \ldots, \langle \nabla g(x_0), \nu_r\rangle) \in \mathbb{R}^{1\times r}. \tag{59}$$

Thus, if a vector w is in the kernel of M, this is equivalent that Vw is orthogonal to $\nabla g(x_0)$. Hence, we can compute the matrix

$$K = (k_1, \ldots, k_{r-1}) \in \mathbb{R}^{r\times(r-1)}, \tag{60}$$

where its column vectors build an orthonormal basis of the kernel of M. If the search directions ν_i are orthogonal, then also the vectors $Vk_1, \ldots, Vk_{r-1}$ are orthogonal which are the column vectors of $VK \in \mathbb{R}^{n\times(r-1)}$ (else VK has to be orthogonalized via another QR decomposition). Doing so, the projected vector to the kernel of M is given by

$$\tilde{\nu}_{\text{new}} = VK(VK)^T \nu^* = VKK^TV^T\nu^*. \tag{61}$$

The above method can be extended to general values of m as follows: define

$$M = GV = (\langle \nabla g_i(x_0), \nu_j\rangle)_{i=1,\ldots,m\ j=1,\ldots,r}, \tag{62}$$

then

1. compute an orthonormal basis $K \in \mathbb{R}^{r\times(r-m)}$ of the kernel of M
2. compute $VK = QR = (q_1, \ldots, q_{r-m}, \ldots, q_n)R$ and set $O := (q_1, \ldots, q_{r-m}) \in \mathbb{R}^{n\times(r-m)}$
3. $\tilde{\nu}_{\text{new}} = OO^T\nu^*$

The gradient-free realization is as above, using the approximations

$$\langle \nabla g_i(x_0), \nu_j\rangle := \frac{g_i(x_j) - g_i(x_0)}{\|x_j - x_i\|_2}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, r, \tag{63}$$

which can be obtained by the samples.

Example 2 We reconsider problem (27) from Example 1, but impose the constraints

$$\begin{aligned} g_1(x) &= -x_1 - 1 \le 0\\ g_2(x) &= -x_2 + 1 \le 0\\ g_3(x) &= -x_3 + 1 \le 0 \end{aligned} \tag{64}$$

Choosing $x_0 = (-1, 1, 1, 2, 2)^T$, we have $\nabla f(x_0) = (-2, 2, 2, 4, 4)^T$ and all three inequality constraints are active. We select the orthogonal search directions

$$V^T = \begin{pmatrix} 1 & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 1 & 1\\ -1 & 1 & 0 & 0 & 0\\ 0 & 0 & -1 & 0 & 1 \end{pmatrix}. \tag{65}$$

When solving (47) using the idealized data, we obtain $\nu_1 = (0.35, 0, 0, -0.42, -0.84)^T$ and the value $\langle \nabla f(x_0), \nu_{\text{new}}\rangle = -5.72$ for the directional derivative in direction ν_1. Using the discretized problem via choosing the points x_i as in (29), we obtain $\tilde{\nu}_1 = (0.37, 0, -0.09, -0.45, -0.81)^T$ with $\langle \nabla f(x_0), \nu_2\rangle = -5.94$ (i.e., even a slightly better value which is, however, luck).
When using the gradient projection as described above, we obtain for the idealized problem the search direction $\nu_2 = (0, 0, 0, -0.45, -0.89)^T$ with a corresponding directional derivative value of $\langle \nabla f(x_0), \nu_{\text{new}}\rangle = -5.36$. For the discretized problem, we obtain
$\tilde{\nu}_2 = (0, 0, -0.09, -0.48, -0.87)^T$ and $\langle \nabla f(x_0), \tilde{\nu}_2\rangle = -5.58$.

4 GSA as stand-alone algorithm

In the following, we discuss some aspects for the efficient numerical realization of line search methods that use a search direction obtained via GSA. Line search methods generate new candidate solutions $x_{(i+1)}$ via

$$x_{(i+1)} = x_i + t_{(i)} \nu_{(i)}, \tag{66}$$

where $x_{(i)}$ is the current iterate, $t_{(i)}$ the step size, and $\nu_{(i)}$ the search direction (here obtained via an application of GSA). Further on, we present a possible GSA stand-alone algorithm that shares some characteristics with the widely used pattern search algorithm (Hooke and Jeeves 1961).

Sampling further test points Crucial for the approximation of the gradient is the choice of the test points x_i. If the function values of points in a neighborhood $N(x_0)$ are already known, it seems to be wise to include them to build the matrix V. Nevertheless, it might be the case that further test points have to be sampled to obtain a better search direction. More precisely, assume that we are given $x_0 \in \mathbb{R}^n$ as well as l neighboring solutions $x_1, \ldots, x_l \in N(x_0)$. According to the discussion above, it is desired that all further search directions are both orthogonal to each other and orthogonal to the previous ones. In order to compute the new search directions $\nu_{l+1}, \ldots, \nu_r$, $r > l$, one can proceed as follows: compute a QR-factorization of $V = (\nu_1, \ldots, \nu_l)$, i.e.,

$$V = QR = (q_1, \ldots, q_l, q_{l+1}, \ldots, q_n)R, \tag{67}$$

where $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $R \in \mathbb{R}^{n\times r}$ is a right upper triangular matrix with nonvanishing diagonal elements, i.e., $r_{ii} \neq 0$ for all $i = 1, \ldots, r$. Then, it is by construction $\nu_i \in span\{q_1, \ldots, q_i\}$ for $i = 1, \ldots, l$, and hence

$$\langle \nu_i, q_j\rangle = 0, \quad \forall i \in \{1, \ldots, l\},\; j \in \{l+1, \ldots, r\}. \tag{68}$$

One can thus, e.g., set

$$\begin{aligned} \nu_{l+i} &= q_{l+i}, \quad i = 1, \ldots, r-l \\ x_{l+i} &= x_0 + \nu_{l+1}, \quad i = 1, \ldots, r-l. \end{aligned} \tag{69}$$

Since the cost for the QR-factorization is $O(n^3)$ in terms of flops, one may alternatively use the Gram–Schmidt procedure (e.g., Nocedal and Wright 2006) to obtain the remaining sample points (e.g., if $r - l$ is small and n is large). This leads to a cost of $O((r-l)^2 n)$ flops. For the special case that $\nu_1 = x_1 - x_0$ and $\tilde{\nu}_2 = x_2 - x_0$ are given such that $\{\nu_1, \tilde{\nu}_2\}$ are linearly independent, the second search vector ν_2 can be computed by

$$\nu_2 = \tilde{\nu}_2 - \langle \nu_1, \tilde{\nu}_2\rangle \nu_1. \tag{70}$$

Step size control It is important to note that even if no sample in direction ν^* exists, the directional derivative in this direction can—by construction of the approach—be approximated as

$$\langle \nabla f(x_0), \nu^*\rangle = \sum_{i=1}^{r} \lambda_i^* \langle \nabla f(x_0), \tilde{\nu}_i\rangle \approx \sum_{i=1}^{r} \lambda_i^* d_i. \tag{71}$$

This might be advantageous for the step size control, and in particular, backtracking methods can be used (e.g., Nocedal and Wright 2006; Dennis andSchnabel 1983) which is not the case for most direct search methods [e.g., the ones reviewed in Brent (1973)].

In order to handle equality constraints, we proceed as in Polak and Mayne Chao and Detong (2011), and use the secant method to backtrack to a solution x with $|h_j(x)| < \epsilon$ for $j = 1, \ldots, m$ and for a predescribed tolerance $\epsilon > 0$.

Stopping criterion If gradient information is at hand, one can simply stop when $\|g\|_2 \leq tol$ for a given small tolerance value $tol > 0$. We have adapted this to our context. Note, however, that there is a certain risk that the process is stopped by mistake if simply $\nu_{(i)}$ is monitored for one iteration step: If all search directions ν_i in V are orthogonal to $\nabla f(x_{(i)})$, then $\nu_{(i)}$ obtained via GSA is zero regardless of the value of $\|g\|_2$. To avoid this potential problem, we test whether the condition is satisfied for several iteration steps. That is, we stop the process if

$$\|\nu_{(i)}\|_2 \leq tol, \quad i = i_0, i_0+1, \ldots, i_0+s, \tag{72}$$

for a number s. In our computations, we have taken $s = 2$.

GSA stand-alone algorithm Now, we are in the position to state the GSA-based line search algorithm. The pseudocode of the procedure for unconstrained problems can be found in Algorithm 1. Note that the resulting algorithm shares some characteristics with the pattern search algorithm: First, a sampling is performed around a given point, and then, a line search is performed in direction of the most promising (greedy) search direction ν computed from the given data.

Algorithm 1 Pseudocode of the stand-alone GSA for unconstrained problems.

```
Require: Initial point $x_{(0)}$
Ensure: Sequence $x_{(k)}$ of candidate solutions
1: $k := 0$
2: while Stopping criteria is not met do
3:     Select neighbors $x_{(k),1}, \dots, x_{(k),r}$ of $x_{(k),0} := x_{(k)}$
4:     $d_i := \frac{f(x_{(k),i}) - f(x_{(k),0})}{\|x_{(k),i} - x_{(k),0}\|_2} \quad i = 1, \dots, r$
5:     $d := (d_1, \dots, d_r)^T$
6:     $\tilde{v}_i := \frac{x_{(k),i} - x_{(k),0}}{\|x_{(k),i} - x_{(k),0}\|_2} \quad i = 1, \dots, r$
7:     $\tilde{V} := (\tilde{v}_1, \dots, \tilde{v}_r)$
8:     Solve $\tilde{V}^T \tilde{V} \tilde{\lambda} = -d$ to obtain $\tilde{\lambda}$
9:     $\tilde{v}^* := \frac{-1}{\|\tilde{V}\tilde{\lambda}\|_2^2} \tilde{V}\tilde{\lambda}$
10:    Compute step size $t_k \in \mathbb{R}_+$
11:    $x_{(i+1)} := x_{(i)} + t_k \tilde{v}^*$
12:    Set $k := k + 1$
13: end while
```

Under mild assumptions, we can expect global convergence toward a local solution of the given SOP: If the angle

$$\cos \Theta_k = \frac{-\nabla f(x_k)^T \nu_{(k)}}{\|\nabla f(x_k)\| \|\nu_{(k)}\|} \tag{73}$$

between the search direction $\nu_{(k)}$ and the steepest descent direction $-\nabla f(x_k)$ is bounded away from 90°, i.e., there exists a $d > 0$ s.t.

$$\cos \Theta_k \geq d, \quad \forall k, \tag{74}$$

and the step size satisfies the Wolfe conditions, then it holds by the theorem of Zoutendijk (e.g., Nocedal and Wright 2006) that $\lim_{k \to \infty} \|\nabla f(x_k)\| = 0$. Condition (74) is quite likely satisfied for $\nu_{(k)}$ which is obtained via GSA at least if the ν_i's are chosen uniformly at random.

Algorithms 2 and 3 show the GSA for constrained problems which incorporate the adaptions discussed above. In order to handle constraint violations, we adopt a penalization function to incorporate the objective and the restrictions into a single value. In particular, we use

$$pen(x) = f(x) + C \sum_{i=1}^{m} H_i(x) + C \sum_{q=1}^{p} g_i(x), \tag{75}$$

where $g_i(x)$ denotes the inequality constraints and $H_i(x)$ denotes the equality constraints transformed into inequalities (i.e., the i-th function is given by $max(0, |h_i(x)| - \epsilon)$), where $C = 1e7$ and $\epsilon = 1e - 4$.

Algorithm 2 Pseudocode of the stand-alone GSA for constrained problems.

```
Require: Initial point $x_{(0)}$
Ensure: Sequence $x_{(k)}$ of candidate solutions
1: $k := 0$
2: while Stopping criteria is not met do
3:     Select the neighbors $x_{(k),1}, \dots, x_{(k),r}$ of $x_{(k),0} := x_{(k)}$
4:     $\tilde{v}_i := \frac{x_{(k),i} - x_{(k),0}}{\|x_{(k),i} - x_{(k),0}\|_2}, \ i = 1, \cdots, r$
5:     $\tilde{V} := (\tilde{v}_1, \cdots, \tilde{v}_r)$
6:     if $\exists j \in \{1, \cdots, p\} : |h_j(x_{(k),0})| > \epsilon$ then
7:         Let $j_1, \cdots, j_l$ s.t. $|h_{j_s}(x_{(k),0})| > \epsilon, \ j_s \in \{1, \cdots, p\}$
8:         $d_i := \frac{f(x_{(k),i}) - f(x_{(k),0})}{\|x_{(k),i} - x_{(k),0}\|_2}, \ i = 1, \cdots, r$
9:         $\tilde{m}_{is} := \frac{h_{j_s}(x_{(k),i}) - h_{j_s}(x_{(k),0})}{\|x_{(k),i} - x_{(k),0}\|_2}, \ i = 1, \cdots, r, \ s = 1, \cdots, l$
10:        $d := (d_1, \cdots, d_r)^T$
11:        Solve $\begin{pmatrix} \tilde{V}^T \tilde{V} & \tilde{M}^T \\ \tilde{M} & 0 \end{pmatrix} \begin{pmatrix} \tilde{\lambda} \\ \tilde{\mu} \end{pmatrix} = \begin{pmatrix} -d \\ 0 \end{pmatrix}$ to obtain $\tilde{\lambda}$
12:    else
13:        $d_i := \frac{pen(x_{(k),i}) - pen(x_{(k),0})}{\|x_{(k),i} - x_{(k),0}\|_2}, \ i = 1, \cdots, r$
14:        $d := (d_1, \cdots, d_r)^T$
15:        Solve $\tilde{V}^T \tilde{V} \tilde{\lambda} = -d$ to obtain $\tilde{\lambda}$
16:    end if
17:    $\tilde{v}^* := \frac{-1}{\|\tilde{V}\tilde{\lambda}\|_2^2} \tilde{V}\tilde{\lambda}$
18:    Compute step size $t_k \in \mathbb{R}_+$
19:    $x_{(k+1)} := x_{(i)} + t_k \tilde{v}^*$
20:    if $\exists j \in \{1, \cdots, m\} : g_j(x_{(k+1)}) > 0$ then
21:        Let $j_1, \cdots, j_l$ s.t. $g_{j_l}(x_{(k+1)}) > 0, \ j_l \in 1, \cdots, m$
22:        $GV_{i j_l} := \frac{g_{j_l}(x_{(k),i}) - g_{j_l}(x_{(k+1)})}{\|x_{(k),i} - x_{(k+1)}\|_2}, \ i = 1, \cdots, r, \ j = 1, \cdots, m$
23:        Calculate kernel $O$ from $(GV)^T$
24:        $\tilde{v}_{new} := O O^T \tilde{v}^*$
25:        Compute step size $t_k \in \mathbb{R}_+$
26:        $x_{(k+1)} := x_{(k)} + t_k \tilde{v}_{new}$
27:    end if
28:    $x_{(k+1)}$ = correctionStep($x_{(k+1)}, t_k$)
29:    Set $k := k + 1$
30: end while
```

Algorithm 3 x_{new} =correctionStep(x_0, t)

```
Require: Initial point $x_{(0)}$, Initial step size $t$
Ensure: Candidate solution $x_{new}$
1: Select the neighbors $x_{N_1}, \dots, x_{N_r}$ of $x_{(0)}$
2: Let $j_1, \cdots, j_l$ s.t. $|h_{j_s}(x_{(k+1)})| > \epsilon, \;\; j_s \in \{1, \cdots, p\}$
3: $\nu_H := 0 \in \mathbb{R}^n$
4: $m_{ij} := \frac{h_{j_s}(x_{N_i}) - h_{j_s}(x_0)}{\|x_{x_{N_i}} - x_0\|_2} \;\; s = 1, \dots, l, i = 1, \dots, r$
5: $\tilde{v}_i := \frac{x_{N_i} - x_0}{\|x_{N_i} - x_{(0)}\|_2}, \;\; i = 1, \cdots, r$
6: $\tilde{V} := (\tilde{v}_1, \cdots, \tilde{v}_r)$
7: $M := (m_1, \cdots, m_l)$
8: for j = 1,..., l do
9:     $d := m_j$
10:    Solve $\tilde{V}^T \tilde{V} \tilde{\lambda} = -d$ to obtain $\tilde{\lambda}$
11:    $\tilde{\nu}_H := \frac{-1}{\|\tilde{V}\tilde{\lambda}\|_2^2} \tilde{V} \tilde{\lambda}$
12:    if $h_j(x_0) < 0$ then
13:        $\tilde{\nu}_H := -\tilde{\nu}_H$
14:    else
15:        $\tilde{\nu}_H := \tilde{\nu}_H$
16:    end if
17:    $\nu_H := \nu_H + \tilde{\nu}_H$
18: end for
19: $\nu_H = \frac{\nu_H}{\|\nu_H\|_2}$
20: $t_{min} := t$
21: for j = 1,..., l do
22:    $t_a := 0$
23:    $h_a := h_j(x_0)$
24:    $t_b := t$
25:    $h_b := h_j(x_0 + t\nu_H)$
26:    $t_c := t_a - \frac{h_a(t_b - t_a)}{h_b - h_a}$
27:    if $t_c < t_{min}$ then
28:        $t_{min} := t_c$
29:    end if
30: end for
31: $x_{new} := x_0 + t_{min}\nu_H$
```

5 GSA within DE

In this section, we detail a novel memetic strategy called DE/GSA that combines DE and GSA. Since the two combined strategies are so different—with one being more explorative and the other more exploitative—the most promising approach might depend both on the optimization stage and on the problem at hand. As a result, we decided to apply a framework that allows adaptively granting more resources to the more promising scheme. Given the high-quality results obtained with the MOS framework (LaTorre 2009; LaTorre et al. 2011), we decided to apply this procedure. In MOS, the improvement achieved by each of the combined strategies is measured at each stage of the optimization. Then, the resources given to the schemes are adapted by taking these measurements of quality into account.

The pseudocode of our proposal is given in Algorithm 4. This scheme belongs to the HTH class, and is thus based on sharing the resources among the different strategies involved (DE and GSA in our case) in each generation. Specifically, a participation ratio is evolved for each of the schemes. The participation ratio is used to set the number of offspring created by each strategy. In the pseudocode, O_{DE} and O_{GSA} represent the offspring created by DE and GSA, Q_{DE} and Q_{GSA} represent the quality of each set of individuals, and Π_{DE} and Π_{GSA} refer to the participation ratios of each technique. The way used to calculate the quality of each scheme was adopted from LaTorre et al. (2011). Specifically, the quality of each improved individual new is defined by

$$Q(new) := \left| \frac{pen(new) - pen(old)}{pen(old)} \right|, \tag{76}$$

where old denotes the old individual. The quality of each method is calculated using its offspring. For instance, if the offspring of GSA is O, it is calculated as

$$Q_{\text{GSA}} := \frac{1}{|O|} \sum_{o \in O} Q(o). \tag{77}$$

Algorithm 4 Pseudocode of the DE/GSA.

```
1: Randomly create initial population $P_0$.
2: Set $\Pi_{DE} := \Pi_{GSA} := 0.5$.
3: Set $k := 0$.
4: while Stopping criteria is not met do
5:     Select $|P_i|\,\Pi_{GSA}$ random individuals from $P_i$.
6:     Create subpopulation $O_{GSA}$ by using GSA.
7:     Calculate quality $Q_{GSA}$ from $O_{GSA}$.
8:     Select $|P_i|\,\Pi_{DE}$ random individuals from $P_i$.
9:     Create subpopulation $O_{DE}$ by using DE.
10:    Calculate quality $Q_{DE}$ from $O_{DE}$.
11:    Update the participation ratios $\Pi_{GSA}$ and $\Pi_{DE}$ using the qualities of each technique.
12:    Select the best individuals from $P_i \cup O_{DE} \cup O_{GSA}$ and save them into $P_{i+1}$
13:    Set $k := k + 1$
14: end while
```

Algorithm 5 describes the way to calculate the participation ratios. This procedure is similar to the one used in LaTorre et al. (2011). The basic principle is to update the participation ratio by taking into account the relationship between the quality of the schemes considered. Two different parameters must be set: the reduction factor (r_f), which is used to tune the velocity of adaptation, and the minimum participation ratio ($r_{\min}$) that can be assigned to any of the strategies used. In our study, we set both r_f and $r_{\min}$ to 0.05.

Algorithm 5 Pseudocode of the participation ratio calculation.

```
Require: Initial ratios $\Pi_{GSA}$, $\Pi_{DE}$, qualities $Q_{GSA}$, $Q_{DE}$
Ensure: New ratios $\Pi_{GSAn}$, $\Pi_{DEn}$
1: if $Q_{GSA} > Q_{DE}$ then
2:     $r_a := r_f \left(\frac{Q_{GSA}-Q_{DE}}{Q_{GSA}}\right) \Pi_{GSA}$.
3:     if $(\Pi_{DE} - r_1) < r_{min}$ then
4:         $r_a := \Pi_{DE} - r_{min}$.
5:     end if
6:     Set $\Pi_{GSAn} := \Pi_{GSA} + r_a$.
7:     Set $\Pi_{DEn} := \Pi_{DE} - r_a$.
8: else if $Q_{DE} > Q_{GSA}$ then
9:     $r_a := r_f \left(\frac{Q_{DE}-Q_{GSA}}{Q_{DE}}\right) \Pi_{DE}$.
10:    if $(\Pi_{GSA} - r_a) < r_{min}$ then
11:        $r_a := \Pi_{GSA} - r_{min}$.
12:    end if
13:    Set $\Pi_{DEn} := \Pi_{DE} + r_a$.
14:    Set $\Pi_{GSAn} := \Pi_{GSA} - r_a$.
15: else
16:    Set $\Pi_{GSAn} := \Pi_{GSA}$.
17:    Set $\Pi_{DEn} := \Pi_{DE}$.
18: end if
```

6 Numerical results

6.1 GSA as stand-alone algorithm

Here, we make a small comparison of the GSA stand-alone algorithm against the classical pattern search method and the Nelder–Mead downhill simplex method (e.g., Nocedal and Wright 2006) on two illustrative examples. For the pattern search method, we have taken the routine `patternsearch` from MATLAB[1], and `fminsearch` (using penalty search to handle constraints) for the Nelder–Mead algorithm.

First, we consider the following box-constrained problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{2} x_i^2 \\ \text{s.t.} \quad & -11 \le x_1 \le 7 \\ & 2 \le x_2 \le 10 \end{aligned}. \tag{78}$$

The initial point for all algorithms has been chosen as $x_0 = (6, 6)^T$. For the realization of GSA, we have chosen $r = 2$. The first search direction has been chosen at random using an initial point in the neighborhood defined by the 2-norm and a radius of 0.2. The second search direction was obtained via (67). In all the experiments, we measure the number of function calls required to achieve an error , which is given by $|pen(x_i) - pen(x^*)|$, less or equal than $5e - 3$. Table 1 presents the total cost of the algorithms in terms of function calls, and Fig. 4 shows the trajectories for each algorithm.

Second, we consider again problem (78), but replace the box constraints by the linear constraint

[1] http://mathworks.com.

Table 1 Number of function calls required for each algorithm on problem (78), compare to Fig. 4

Algorithm	Function calls
Pattern search	219
Nelder–Mead with penalty search	149
GSA	**38**

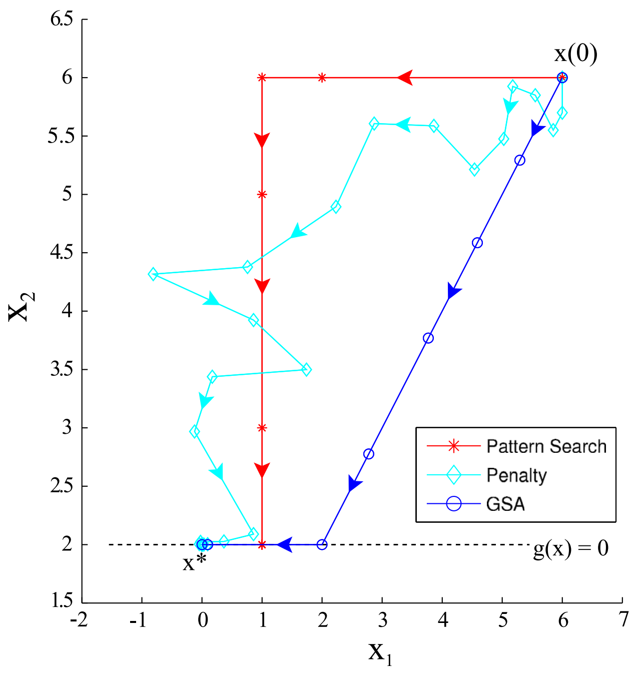

Fig. 4 Trajectories obtained by the different algorithms on problem (78)

Table 2 Number of function calls required for each algorithm on objective (78) subject to the constraint (79), compare to Fig. 5

Algorithm	Function calls
Pattern search	182
Nelder–Mead with penalty search	182
GSA	**47**

$$x_1 + x_2 \ge 1. \tag{79}$$

As initial solution, we have chosen $x_0 = (6, -1)^T$. Table 2 presents the total cost, and Fig. 5, the trajectories for each algorithm. In both cases, the GSA stand-alone algorithm clearly outperforms the other methods.

The above impression gets confirmed when solving both problems taking 1000 randomly chosen feasible starting points. Table 3 shows the averaged number of function calls required for each algorithm on both problems.

Next, we consider dimension $n = 10$ and consider the following five SOPs:

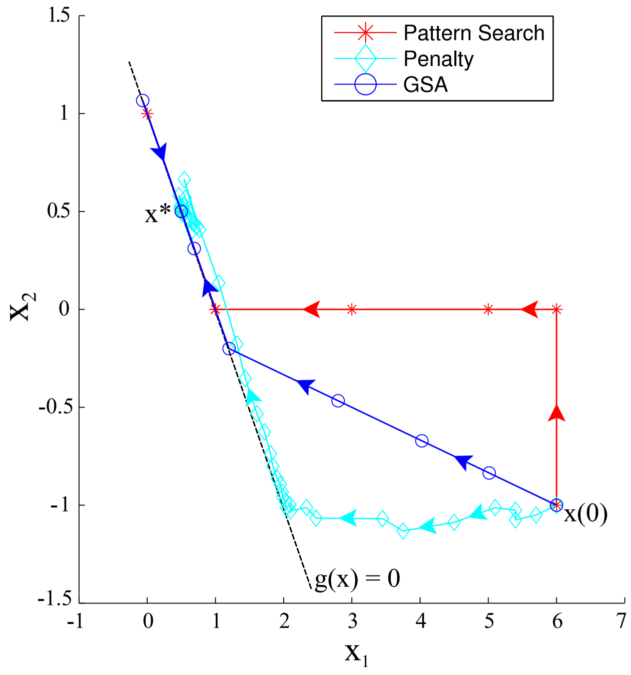

Fig. 5 Trajectories obtained by the different algorithms on objective (78) subject to the constraint (79)

$$\min \sum_{i=1}^{10} x_i^2 \quad \text{s.t.} \quad -1 \le x_1 \le 7,\; 2 \le x_2 \le 10,\; 1 \le x_3 \le 9. \tag{80}$$

$$\min \sum_{i=1}^{10} x_i^2 \quad \text{s.t.} \quad x_1 + x_2 + x_3 \ge 1. \tag{81}$$

$$\min \sum_{i=1}^{10} x_i^2 \quad \text{s.t.} \quad \sum_{i=1}^{10} x_i \ge 1. \tag{82}$$

$$\min \sum_{i=1}^{10} x_i^2 \quad \text{s.t.} \quad \sum_{i=1}^{10} x_i = 1. \tag{83}$$

$$\min \sum_{i=1}^{10} x_i^2 \quad \text{s.t.} \quad (x_1+1)^2 + \sum_{i=2}^{10} x_i^2 = 4. \tag{84}$$

Table 3 Required number of function calls ($\#FC$) for each solver

Problem	Algorithm	$\#FC$
(78)	Pattern search	263.73
	Nelder–Mead w. penalty search	173.46
	GSA	**35.33**
(78) + (79)	Pattern search	302.36
	Nelder–Mead w. penalty search	161.73
	GSA	**38.36**

The values are averaged over 1000 runs coming from different starting points

Table 4 Required number of function calls ($\#FC$) for each solver

Problem	Algorithm	$\#FC$
(80)	Pattern search	4218.9
	Nelder–Mead w. penalty search	1285.9
	GSA	**377.7**
(81)	Pattern search	3461.5
	Nelder–Mead w. penalty search	1558.3
	GSA	**339.8**
(82)	Pattern search	4015.8
	Nelder–Mead w. penalty search	–
	GSA	**310.3**
(83)	Pattern search	3818.3
	Nelder–Mead w. penalty search	–
	GSA	**271.0**
(84)	Pattern search	–
	Nelder–Mead w. penalty search	–
	GSA	**1277.8**

The values are averaged over 1000 runs coming from different starting points

For the realization of the GSA, we have proceeded as above, albeit using $r = 5$. Table 4 shows the result of all three algorithms on these five problems. The values are again averaged over 1000 runs coming from different feasible starting points. In all cases, the GSA line searcher requires about one order of magnitude less function evaluations to reach its goal.

Finally, Fig. 6 presents the graphical results of the comparison between the three algorithms. On this, it is presented the number of function calls against the error which is defined as the 2-norm of the best found solution to the solution, $\|x_{\text{best}} - x^*\|$. By this, it becomes clear that the GSA algorithm (presented in black) overpasses the convergence rate of the two other algorithms.

6.2 GSA within DE

In this section, the experiments conducted with the newly designed hybrid DE scheme are described. First, we consider

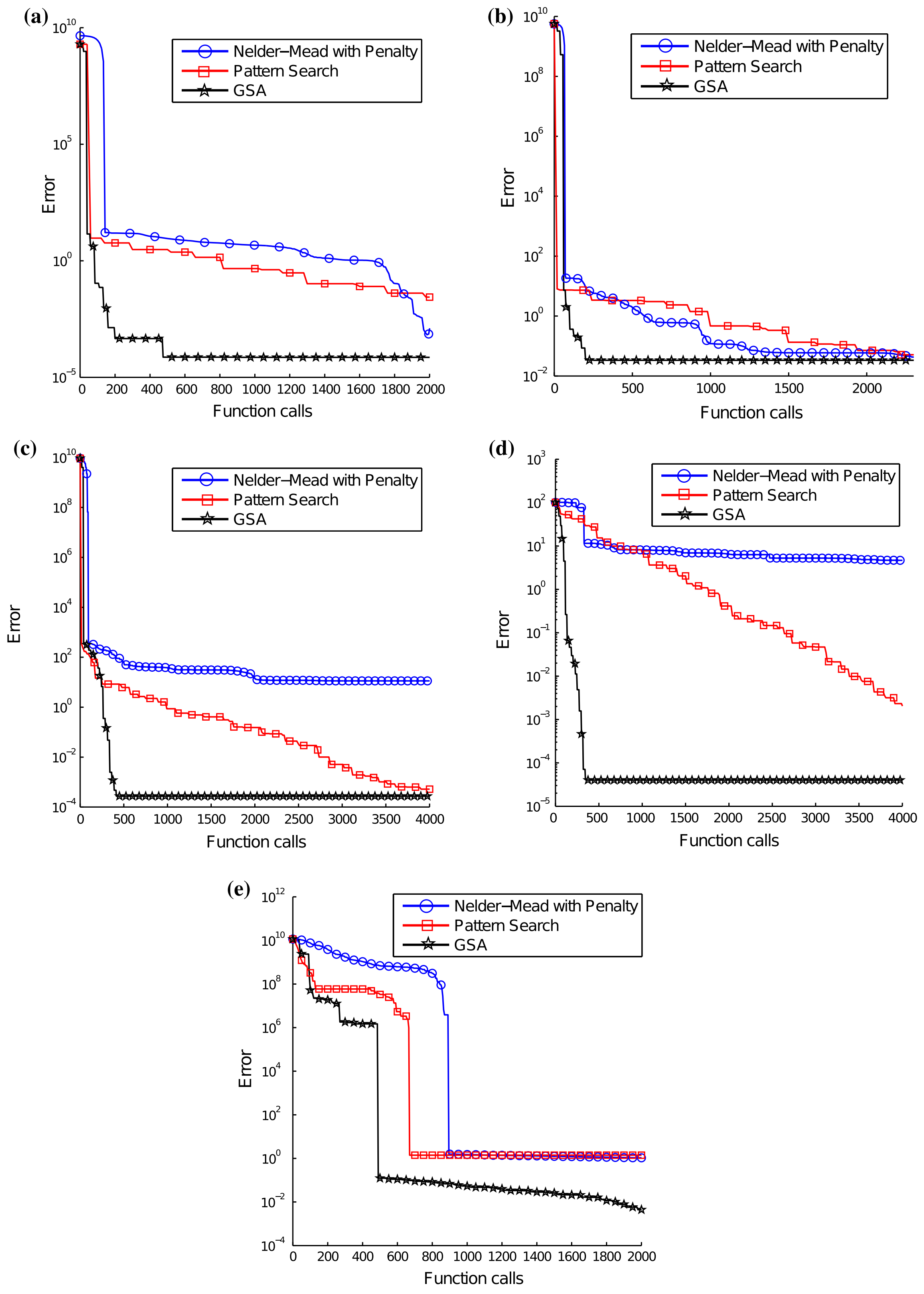

Fig. 6 Convergence plot for the stand-alone algorithms for SOPs (80) to (84). **a** SOP (80) **b** SOP (81) **c** SOP (82) **d** SOP (83) **e** SOP (84)

Table 5 Average function calls needed to find the optimum

Problem	Algorithm	FC
SOP (80)	DE	29315.1
	DE/GSA	**16123.3**
SOP (81)	DE	46692.3
	DE/GSA	**22315.2**
SOP (82)	DE	51214.2
	DE/GSA	**20031.1**
SOP (83)	DE	43100.5
	DE/GSA	**12266.7**
SOP (84)	DE	68166.7
	DE/GSA	**27200.2**

the SOPs (80) to (84) from the previous section and compared the averaged performance between the stand-alone DE and the algorithm coupled with the GSA. Table 5 presents the number of function calls that both algorithms required to archive an error of $5e-3$.

Moreover, Fig. 7 presents the convergence plot for both algorithms. From these plots, it is possible to compare the algorithms involved. The memetic strategy yields the best convergence behavior for all the problems. As the objective is quadratic in all cases, inferiority of DE can be expected (Auger et al. 2009).

Next, we analyzed the performance of the DE/GSA using the benchmarks devised for the competition on constrained real-parameter optimization (Liang et al. 2006) held at the congress on evolutionary computation. They comprise 24 problems of varying complexity which have different features involving modality and separability, which is why we selected this set of problems [2].

In addition, in order to test our proposal on unconstrained problems, the happy cat function (HC, Beyer and Finck 2012) was also used. In this case, two different dimensionalities ($D = 10$ and $D = 20$) were considered. This last function was selected because most current EAs fail at solving it. In order to analyze the schemes, two different sets of experiments were carried out. In both experiments, the fitness function defined in Eq. 75 was used to compare the results. In every case, three different DE variants were considered. The first one (DE- 1) is the well-known DE/target-to-best/1 variant Das and Suganthan (2011). The second one (DE/LS1) incorporates the use of the LS1 strategy that was defined in Tseng and Chen (2008). The basic principle of this strategy is to search along each of the dimensions and adapt the step sizes depending on the requirements of the problem at hand. It was parameterized as recommended by the authors. Note that LS1 had previously been successfully combined with DE (LaTorre et al. 2011), yielding high-quality results for a large number of unconstrained optimization problems. However, to the best of our knowledge, it had not been tested previously with constrained optimization problems. The local search LS- 1 was incorporated into DE- 1 following the guidelines described in Sect. 5. Finally, the last scheme (DE/GSA) incorporates the GSA approach. The same parameterization was used in every DE variant. Specifically, NP, F and CR were set to 100, 0.8 and 0.95, respectively.

Since stochastic algorithms were considered in this study, each execution was repeated 30 times for every case. Moreover, comparisons were made by applying a set of statistical tests. A similar guideline to the one applied in Durillo et al. (2010) was considered. Specifically, the following tests were applied, assuming a significance level of 5 %. First, a *Shapiro–Wilk test* was performed to check whether or not the values of the results followed a Gaussian distribution. If so, the *Levene test* was used to check for the homogeneity of the variances. If samples had equal variance, an ANOVA *test* was done; if not, a *Welch test* was performed. For non-Gaussian distributions, the nonparametric *Kruskal–Wallis* test was used to determine whether samples were drawn from the same distribution.

6.2.1 First set of experiments: quality of results

The aim of the first experiment was to analyze the benefits contributed by DE/GSA in terms of the quality of the solutions obtained. Since the relative behavior among the different schemes might depend on the stopping criterion established, the schemes were analyzed for both short and long periods. For the short period, the stopping criterion was set to 50,000 function evaluations. For the long period, it was set to 500,000 function evaluations.

Table 6 shows the mean and median obtained by the three models in question for 50,000 function evaluations. Moreover, in the columns corresponding to DE/LS1 and DE/GSA, we show the results of the statistical tests when compared with the results provided by DE- 1. The $\uparrow$ symbol indicates the superiority of the model given in the column, whereas the $\downarrow$ symbol indicates its inferiority. In those cases where the differences were not statistically significant, the $\leftrightarrow$ symbol is used. Among the models considered, the superiority of DE/GSA is clear. In fact, the statistical tests show that DE/GSA was superior to DE1 in 13 out of the 25 cases, whereas it was inferior in only 2 cases. In the case of DE/LS1, its benefits are not as clear. DE/LS1 was superior to DE1 in 7 cases; however, in another 7 cases, it was inferior. In light of these results, we may state that using a local search, such as GSA, that profits from the content of the population to estimate the gradient seems more promising than applying some of the simpler schemes that have been successfully used for unconstrained problems.

[2] Problem g12 was discarded since it is not a continuous problem.

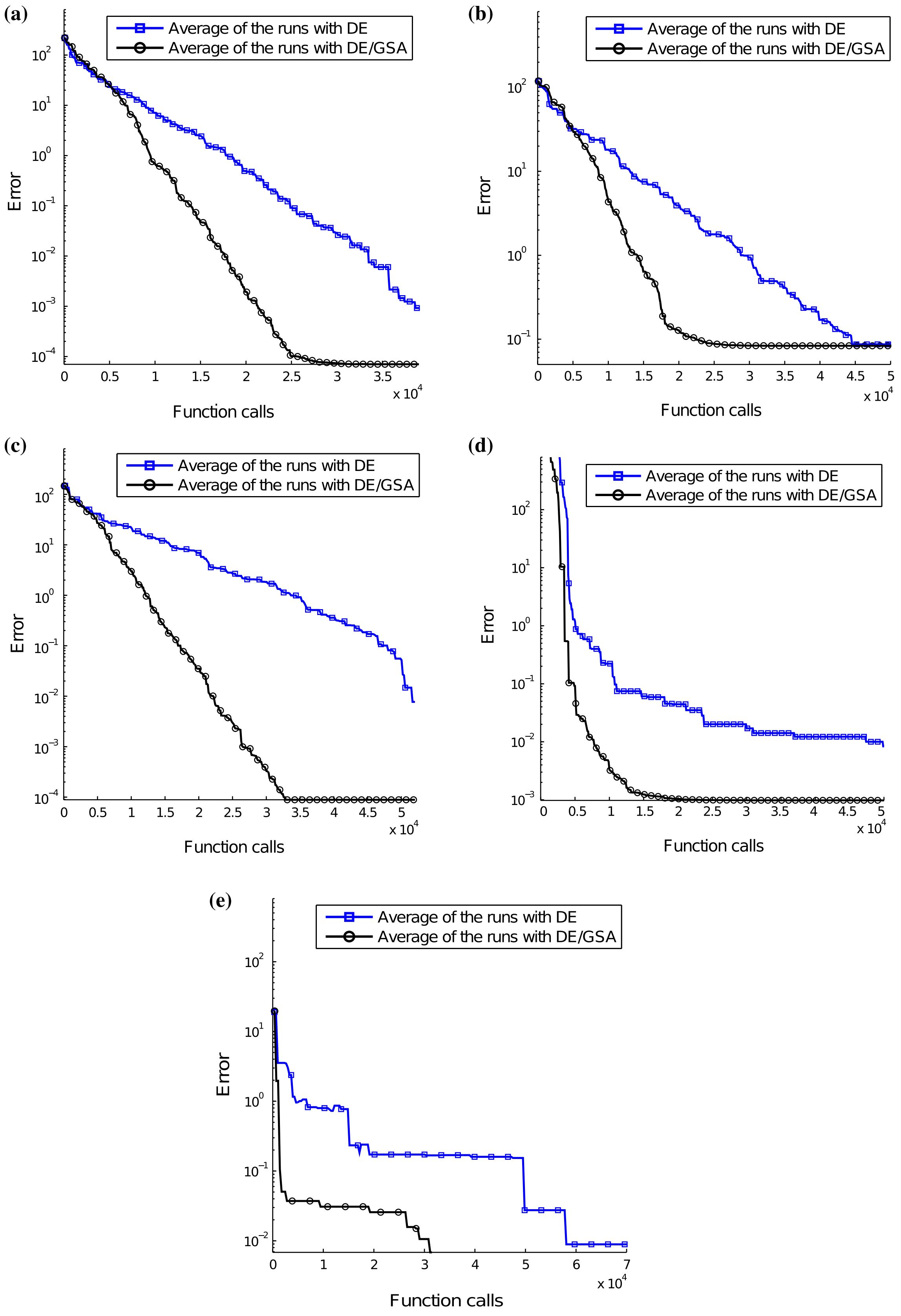

Fig. 7 Convergence plot for DE and DE/GSA for SOPs (80) to (84). **a** SOP (80) **b** SOP (81) **c** SOP (82) **d** SOP (83) **e** SOP (84)

Table 6 Solutions obtained by the different DE variants in 50,000 evaluations

	DE- 1		DE/LS1			DE/GSA		
	Mean	Median	Mean	Median	St.	Mean	Median	St.
g01	-1.49×10^1	-1.49×10^1	-1.49×10^1	-1.49×10^1	↓	-1.49×10^1	-1.49×10^1	↑
g02	-4.90×10^{-1}	-4.87×10^{-1}	-6.34×10^{-1}	-6.32×10^{-1}	↑	-4.75×10^{-1}	-4.79×10^{-1}	↓
g03	-4.84×10^{-2}	-1.90×10^{-2}	-2.00×10^{-1}	-1.84×10^{-1}	↑	-3.95×10^{-1}	-3.54×10^{-1}	↑
g04	-3.06×10^4	-3.06×10^4	-3.06×10^4	-3.06×10^4	↔	-3.06×10^4	-3.06×10^4	↔
g05	5.25×10^3	5.21×10^3	5.29×10^3	5.25×10^3	↔	5.12×10^3	5.12×10^3	↑
g06	-6.96×10^3	-6.96×10^3	-6.96×10^3	-6.96×10^3	↔	-6.96×10^3	-6.96×10^3	↔
g07	2.67×10^1	2.66×10^1	2.59×10^1	2.59×10^1	↑	2.44×10^1	2.44×10^1	↑
g08	-9.58×10^{-2}	-9.58×10^{-2}	-9.58×10^{-2}	-9.58×10^{-2}	↔	-9.58×10^{-2}	-9.58×10^{-2}	↔
g09	6.80×10^2	6.80×10^2	6.80×10^2	6.80×10^2	↓	6.80×10^2	6.80×10^2	↑
g10	2.10×10^3	2.10×10^3	2.10×10^3	2.10×10^3	↔	2.10×10^3	2.10×10^3	↔
g11	7.49×10^{-1}	7.49×10^{-1}	7.49×10^{-1}	7.49×10^{-1}	↔	7.49×10^{-1}	7.49×10^{-1}	↔
g13	9.90×10^{-1}	9.99×10^{-1}	7.77×10^{-2}	7.50×10^{-2}	↑	8.13×10^{-1}	9.90×10^{-1}	↔
g14	-4.27×10^1	-4.27×10^1	-4.25×10^1	-4.25×10^1	↔	-4.77×10^1	-4.77×10^1	↑
g15	9.62×10^2	9.61×10^2	9.63×10^2	9.62×10^2	↓	9.61×10^2	9.61×10^2	↑
g16	-1.90	-1.90	-1.90	-1.90	↓	-1.90	-1.90	↓
g17	8.94×10^3	8.94×10^3	8.95×10^3	8.94×10^3	↔	8.87×10^3	8.86×10^3	↑
g18	-7.89×10^{-1}	-7.90×10^{-1}	-8.23×10^{-1}	-8.24×10^{-1}	↑	-8.55×10^{-1}	-8.56×10^{-1}	↑
g19	5.07×10^1	5.11×10^1	5.69×10^1	5.73×10^1	↓	4.67×10^1	4.73×10^1	↑
g20	1.82×10^{-1}	1.84×10^{-1}	1.91×10^{-1}	1.83×10^{-1}	↔	1.70×10^{-1}	1.63×10^{-1}	↔
g21	3.14×10^2	3.26×10^2	4.50×10^2	3.42×10^2	↓	3.37×10^2	3.25×10^2	↔
g22	1.17×10^4	1.17×10^4	1.01×10^4	1.03×10^4	↔	7.53×10^3	5.94×10^3	↔
g23	5.05×10^1	-4.76	1.09×10^2	-6.26×10^{-2}	↓	-2.16×10^2	-2.23×10^2	↑
g24	-5.50	-5.50	-5.50	-5.50	↔	-5.50	-5.50	↔
HC-10	3.98×10^{-1}	4.04×10^{-1}	3.84×10^{-1}	3.76×10^{-1}	↑	2.88×10^{-1}	2.91×10^{-1}	↑
HC-20	5.67×10^1	5.54×10^1	2.64×10^0	2.91×10^0	↑	3.88×10^1	3.72×10^1	↑

Similar information is shown in Table 7 for 500,000 function evaluations. In this case, the differences are not as large. For instance, DE/GSA is superior to DE1 in only 5 cases. The reason is that in several test cases, every approach is able to reach solutions that are very close to optimal. Thus, when considering a very long period, the advantage of incorporating an effective local search, such as GSA, diminishes, showing that the real advantage of DE/GSA is a reduction in the resources required to obtain high-quality solutions. In any case, in some of the most complex problems, some advantages appear in both the short and long terms.

6.2.2 *Second set of experiments: saving resources*

The previous analyses show the benefits of the new scheme in terms of the quality of the results. In the short period, the results obtained by DE/GSA are clearly better than those obtained by the rest of the schemes. However, in the long term, the differences are not as large. Thus, it is very interesting to estimate the amount of resources that can be saved when using the DE/GSA approach. Performing a direct comparison between the final results obtained by the different models is not helpful; instead, the run-length distribution can be used.

Run-length distributions show the relationship between success ratios and number of evaluations, where the success ratio is defined as the probability of achieving a certain quality level. In order to establish said quality level, we considered, for each of the problems, the highest median obtained by any of the three models considered in 500,000 function evaluations. Since showing the run-length distributions for each problem would require too much space, we show in Table 8 the number of evaluations required by each of the models to attain a success ratio equal to 50 %. In addition, we show the percentage of evaluations that were saved by using DE/LS1 and DE/GSA with respect to DE1. Negative values indicate that the corresponding model required a larger number of function evaluations than DE1. Note that in those cases where DE/GSA required more function evaluations, the penalty was not very large. However, in several

Table 7 Solutions obtained by the different DE variants in 500,000 evaluations

	DE- 1		DE/LS1			DE/GSA		
	Mean	Median	Mean	Median	Stat.	Mean	Median	Stat.
g01	-1.5×10^{1}	-1.5×10^{1}	-1.5×10^{1}	-1.5×10^{1}	$\leftrightarrow$	-1.5×10^{1}	-1.5×10^{1}	$\leftrightarrow$
g02	-7.883×10^{-1}	-7.898×10^{-1}	-6.645×10^{-1}	-6.660×10^{-1}	$\downarrow$	-7.754×10^{-1}	-7.808×10^{-1}	$\downarrow$
g03	-1.791×10^{-1}	-1.569×10^{-1}	-2.039×10^{-1}	-1.845×10^{-1}	$\leftrightarrow$	-5.938×10^{-1}	-5.695×10^{-1}	$\uparrow$
g04	-3.066×10^{4}	-3.066×10^{4}	-3.066×10^{4}	-3.066×10^{4}	$\leftrightarrow$	-3.066×10^{4}	-3.066×10^{4}	$\leftrightarrow$
g05	5.126×10^{3}	5.126×10^{3}	5.127×10^{3}	5.126×10^{3}	$\leftrightarrow$	5.126×10^{3}	5.126×10^{3}	$\leftrightarrow$
g06	-6.961×10^{3}	-6.961×10^{3}	-6.961×10^{3}	-6.961×10^{3}	$\leftrightarrow$	-6.961×10^{3}	-6.961×10^{3}	$\leftrightarrow$
g07	2.430×10^{1}	2.430×10^{1}	2.434×10^{1}	2.433×10^{1}	$\downarrow$	2.430×10^{1}	2.430×10^{1}	$\uparrow$
g08	-9.582×10^{-2}	-9.582×10^{-2}	-9.582×10^{-2}	-9.582×10^{-2}	$\leftrightarrow$	-9.582×10^{-2}	-9.582×10^{-2}	$\leftrightarrow$
g09	6.806×10^{2}	6.806×10^{2}	6.806×10^{2}	6.806×10^{2}	$\leftrightarrow$	6.806×10^{2}	6.806×10^{2}	$\leftrightarrow$
g10	2.100×10^{3}	2.100×10^{3}	2.100×10^{3}	2.100×10^{3}	$\leftrightarrow$	2.100×10^{3}	2.100×10^{3}	$\leftrightarrow$
g11	7.499×10^{-1}	7.499×10^{-1}	7.499×10^{-1}	7.499×10^{-1}	$\leftrightarrow$	7.499×10^{-1}	7.499×10^{-1}	$\leftrightarrow$
g13	4.587×10^{-1}	4.609×10^{-1}	7.786×10^{-2}	7.506×10^{-2}	$\uparrow$	3.790×10^{-1}	4.388×10^{-1}	$\leftrightarrow$
g14	-4.776×10^{1}	-4.776×10^{1}	-4.776×10^{1}	-4.776×10^{1}	$\downarrow$	-4.776×10^{1}	-4.776×10^{1}	$\uparrow$
g15	9.617×10^{2}	9.617×10^{2}	9.617×10^{2}	9.617×10^{2}	$\leftrightarrow$	9.617×10^{2}	9.617×10^{2}	$\leftrightarrow$
g16	-1.905	-1.905	-1.905	-1.905	$\leftrightarrow$	-1.905	-1.905	$\leftrightarrow$
g17	8.907×10^{3}	8.930×10^{3}	8.923×10^{3}	8.941×10^{3}	$\leftrightarrow$	8.866×10^{3}	8.856×10^{3}	$\uparrow$
g18	-8.659×10^{-1}	-8.659×10^{-1}	-8.659×10^{-1}	-8.660×10^{-1}	$\uparrow$	-8.659×10^{-1}	-8.659×10^{-1}	$\downarrow$
g19	4.009×10^{1}	4.008×10^{1}	4.487×10^{1}	4.573×10^{1}	$\downarrow$	4.007×10^{1}	4.007×10^{1}	$\uparrow$
g20	1.987×10^{-1}	1.985×10^{-1}	1.736×10^{-1}	1.757×10^{-1}	$\uparrow$	2.016×10^{-1}	2.014×10^{-1}	$\downarrow$
g21	2.810×10^{2}	3.247×10^{2}	2.766×10^{2}	3.247×10^{2}	$\leftrightarrow$	2.592×10^{2}	2.592×10^{2}	$\leftrightarrow$
g22	1.060×10^{4}	1.124×10^{4}	3.653×10^{3}	3.995×10^{3}	$\uparrow$	6.901×10^{3}	6.797×10^{3}	$\leftrightarrow$
g23	-3.690×10^{2}	-3.989×10^{2}	-2.919×10^{2}	-2.743×10^{2}	$\downarrow$	-3.897×10^{2}	-3.950×10^{2}	$\leftrightarrow$
g24	-5.508	-5.508	-5.508	-5.508	$\leftrightarrow$	-5.508	-5.508	$\leftrightarrow$
HC-10	1.73×10^{-1}	1.81×10^{-1}	6.25×10^{-2}	6.26×10^{-2}	$\uparrow$	1.39×10^{-1}	1.32×10^{-1}	$\uparrow$
HC-20	5.08×10^{-1}	5.16×10^{-1}	3.50×10^{-1}	3.41×10^{-1}	$\uparrow$	3.15×10^{-1}	3.25×10^{-1}	$\uparrow$

cases, the number of function evaluations that was saved by using DE/GSA was very significant. For instance, there were ten cases where the percentage of function evaluations saved by DE/GSA with respect to DE- 1 was larger than 50 %. In none of the cases, however, did the opposite happen. Moreover, the benefits are clear both in the constrained and unconstrained cases. Furthermore, taking into account the mean of the evaluations required by each problem, the percentage of evaluations saved was 43.27 %, demonstrating once again the benefits of the new scheme.

In the case of DE/LS1, the advantages are not as clear. There were several cases where DE/LS1 provided important benefits. However, in other cases, the negative impact of incorporating LS1 was also significant. Specifically, in the case of DE/LS1 there were three cases where the penalty was larger than 100 % and taking into account the mean of the evaluations required by each problem, the percentage of evaluations saved was −1.32 %; in other words, when all the problems were considered, DE/LS1 required a larger number of evaluations than DE1. This shows, once again, the importance of using a proper local search scheme. It is also worth noting that in the case of constrained problems, the benefits provided by DE/GSA are more significant.

7 Conclusions and future work

In this paper, we have proposed the gradient subspace approximation (GSA), a new method that computes the most greedy search direction at a point x out of a given set of samples which can be located (in principle) in any direction from x. This feature makes it an interesting candidate as a local searcher within set-based heuristics such as evolutionary algorithms since then the computation of the search direction comes ideally for free in terms of function evaluations. For unconstrained SOPs, a particular normal equation system has to be solved which makes the GSA for such problems very similar to existing gradient approximation approaches. The key difference, however, is that the ansatz of the GSA allows to directly incorporate constraints, i.e., to find the most

Table 8 Evaluations required by the different DE variants to obtain a fixed quality level and percentage of saved resources

	DE- 1	DE/LS1		DE/GSA	
	Eval.	Eval.	Saved (%)	Eval.	Saved (%)
g01	221250	281738	−27.33	274679	−24.18
g02	223200	438630	−96.51	185456	16.91
g03	378600	34205	90.96	7862	97.92
g04	66900	74409	−11.22	63274	5.42
g05	343000	417438	−21.70	143974	58.02
g06	35650	39905	−11.93	39424	−10.58
g07	188550	469315	−148.90	60750	67.78
g08	56550	49020	13.31	55824	1.28
g09	290400	343752	−18.37	235126	19.03
g10	32450	37468	−15.46	9062	72.07
g11	37700	42292	−12.18	32702	13.25
g13	494550	25001	94.94	265106	46.39
g14	327350	496336	−51.62	101196	69.08
g15	134700	222260	−65.00	98313	27.01
g16	147500	164214	−11.33	246482	−67.10
g17	226900	364715	−60.73	10194	95.50
g18	440850	96543	78.10	480413	−8.97
g19	75450	482225	−539.13	58110	22.98
g20	162550	143728	11.57	58079	64.27
g21	485950	497234	−2.32	286394	41.06
g22	256400	482546	−88.20	55090	78.51
g23	110300	366143	−231.95	133544	−21.07
g24	33150	36641	−10.53	43808	−32.15
HC-10	454200	77384	82.96	206230	54.59
HC-20	489900	106768	78.20	89969	81.63
Mean	228560	231596	−1.32	129642	43.27

greedy vector within the set of feasible directions which is still an issue in evolutionary computation.

Based on this idea, we have first developed and discussed the GSA as stand-alone algorithm which has some similarities with the pattern search method. Second, we have integrated GSA as local search engine within differential evolution. We have finally shown the strengths of both approaches on some numerical examples.

We think that the GSA opens a door for more sophisticated gradient-free memetic strategies. Though the variant proposed here already showed the potential of the novel method, we think that more powerful algorithms based on GSA can be designed that yield a more efficient interleaving of local and global search. Further, due to the relation of GSA and pattern search, an application to mixed-integer problems seems to be appealing which we also leave for future work. Another problem which we have to leave for future investigations is that GSA is only applicable to problems with moderate dimensions n of the decision space. In our computations, we have chosen $r = 5$ test points for problems with up to $n = 20$ decision variables. When considering that r grows linear proportional to n, it is apparent that for growing n the problems arise to (1) obtain the required neighboring solutions as well as (2) to solve the related normal equation systems which may get problematic for, say, $n \gg 100$. As a remedy, we conjecture that a combination of dimension reduction techniques as in Omidvar et al. (2014), Li and Yao (2012) with GSA may be effective.

Acknowledgements The first author acknowledges support from Conacyt Project No. 128554. The third author also acknowledges the financial support from CONCYTEG as part of the ‘Plan Investigadores Jóvenes - DPP-2014’.

Compliance with ethical standards

Conflicts of interest All authors declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

References

Auger A, Hansen N, Zerpa Perez J M, Ros R, Schoenauer M (2009) Experimental comparisons of derivative free optimization algorithms. In: Experimental algorithms, Springer, pp 3–15

Bäck T, Schwefel HP (1993) An overview of evolutionary algorithms for parameter optimization. Evol Comput 1(1):1–23

Bao Y, Hu Z, Xiong T (2013) A PSO and pattern search based memetic algorithm for SVMs parameters optimization. Neurocomputing 117:98–106

Beyer HG, Finck S (2012) Happycat a simple function class where well-known direct search algorithms do fail. In: Coello Coello CA, et al., (ed) Parallel problem solving from nature—PPSN XII, vol 7491 of Lecture Notes in Computer Science, Springer, Berlin, pp 367–376

Beyer HG, Schwefel HP (2002) Evol strategies: a comprehensive introduction Nat Comput 1(1):3–52

Brent RP (1973) Algorithms for minimization without derivatives, 1st edn. Prentice-Hall, Upper Saddle River, NJ

Brown M, Smith RE (2005) Directed multi-objective optimisation. Int J Comput Syst Signals 6(1):3–17

Caraffini F, Neri F, Iacca G (2013) Parallel memetic structures. Inf Sci 227:60–82

Caraffini F, Neri F, Picinali L (2014) An analysis on separability for memetic computing automatic design. Inf Sci 225:1–22

Chao G, Detong Z (2011) A secant algorithm with line search filter method for nonlinear optimization. Appl Math Model 35(2):879–894

Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31

Dennis JE, Schnabel RB (1983) Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Upper Saddle River

Domínguez IS, Aguirre AH, Valdez SI (2014) A new EDA by a gradient-driven density. In: Parallel problem solving from nature—PPSN XIII—13th international conference, pp 352–361

Durillo JJ, Nebro AJ, Coello Coello CA, Garcia-Nieto J, Luna F, Alba E (2010) A study of multiobjective metaheuristics when solving parameter scalable problems. IEEE Trans Evol Comput 14(4):618–635

Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Springer, NewYork

Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Natural computing series. Springer, NewYork

Gong W, Cai Z, Ling CX (2006) ODE: a fast and robust differential evolution based on orthogonal design. In: Sattar A, Kang BH (eds) AI 2006: advances in artificial intelligence, vol 4304. Lecture Notes in Computer Science, Springer, Berlin, pp 709–718

Griewank A (2000) Evaluating derivatives: principles and techniques of algorithmic differentiation. Number 19 in Frontiers in Applied Mathematics SIAM, Philadelphia, PA

Hazen M, Gupta MR (2006) A multiresolutional estimated gradient architecture for global optimization. In: IEEE international conference on evolutionary computation, CEC, pp 3013–3020

Hooke R, Jeeves TA (1961) Direct search solution of numerical and statistical problems. J ACM 8(2):212–229

Junhua Z, Yan X, Luo L, ZhaoYang D, Yaoyao P (2014) Power system fault diagnosis based on history driven differential evolution and stochastic time domain simulation. Inf Sci 275:13–29

Kleijnen J P C (2015) Response surface methodology. In: Fu MC (ed) Handbook of simulation optimization, vol 216 of international series in operations research & management science, Springer, New York, pp 81–104

Kukkonen S, Lampinen J (2006) Constrained real-parameter optimization with generalized differential evolution. In: Evolutionary computation, 2006. CEC 2006. IEEE Congress on IEEE, pp 207–214

Lara A, Sanchez G, Coello Coello CA, Schütze O (2010) HCS: a new local search strategy for memetic multiobjective evolutionary algorithms. IEEE Trans Evol Comput 14(1):112–132

LaTorre A (2009) A framework for hybrid dynamic evolutionary algorithms: multiple offspring sampling (MOS). Ph.D. thesis

LaTorre A, Muelas S, Pena JM (2011) A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. Soft Comput 15(11):2187–2199

Li X, Yao X (2012) Cooperatively coevolving particle swarms for large scale optimization. Evol Comput IEEE Trans 16(2):210–224

Liang JJ, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan PN, Coello Coello CA, Deb K (2006) Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore

Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report C3P Report 826, California Institute of Technology

Neri F, Cotta C, Moscato P (eds) (2012) Handbook of memetic algorithms, vol 379 of Studies in Computational Intelligence. Springer

Nocedal J, Wright S (2006) Numerical optimization. Springer series in operations research and financial engineering. Springer, NewYork

Omidvar MN, Mei Y, Li X (2014) Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms. In: Evolutionary computation (CEC), 2014 IEEE congress on IEEE, pp 1305–1312

Osyczka A, Krenich S (2006) Evolutionary algorithms for global optimization. In: Pintér JD (ed) Global optimization, vol 85. Springer, NewYork, pp 267–300

Polak E, Mayne DQA Robust secant method for optimization problems with inequality constraints. J Optim Theory Appl 33(4):463–477

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In IEEE congress on evolution computation 2005 (CEC'05), vol 2, pp 1785–1791

Schütze O, Martín A, Lara A, Alvarado S, Salinas E, Coello Coello CA (2015) The directed search method for multi-objective memetic algorithms. Comput Optim Appl 63(2):1–28

Schwefel HP (1993) Evolution and optimum seeking. Wiley, New York, NY

Shiwen Y, Anyong Q (2005) Design of high-power millimeter-wave TM_{01} - TE_{11} mode converters by the differential evolution algorithm. IEEE Trans Plasma Sci 33(4):1372–1376

Sivanandam SN, Deepa SN (2007) Introduction to genetic algorithms. Springer, NewYork

Storn R, Price K (1995) Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, Berkeley, Technical report TR95012

Talbi EG (2002) A taxonomy of hybrid metaheuristics. J. Heuristics 8(5):541–564

Tseng L-Y, Chen C (2008) Multiple trajectory search for large scale global optimization. In: Evolutionary computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress, pp 3052–3059

Zapotecas Martínez S, Coello Coello CA, (2012) A direct local search mechanism for decomposition-based multi-objective evolutionary algorithms. (2012). In: IEEE congress on evolutionary computation (CEC'2012). IEEE Press, Brisbane