PGS-COM: A hybrid method for constrained non-smooth black-box optimization problems
Brief review, novel algorithm and comparative evaluation

Emanuele Martelli [a],[*], Edoardo Amaldi [b]

[a] *Dipartimento di Energia, Politecnico di Milano, via Lambruschini 4, 20156 Milano, Italy*
[b] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy*

Article history:
Received 9 February 2013
Received in revised form
24 December 2013
Accepted 27 December 2013
Available online 14 January 2014

1. Introduction

Many engineering optimization problems are formulated as black-box problems, in the sense that the analytical expressions of the objective function and/or the constraints are not explicitly given. For example, the objective function value could be the result of a process simulation, a finite element method study, a computational fluid-dynamic analysis, or the value returned by the solver of a lower level optimization problem (i.e., in bilevel programming, the lower level optimization problem is solved multiple times for fixed values of the upper level variables). Non-smooth black-box problems with non-differentiable and/or discontinuous objective functions typically arise in chemical and energy engineering when the process is simulated by means of a sequential flowsheet solver, as in Luus and Jaakola (1973a), Gaines and Gaddy (1976), Banga and Seider (1996), Gross and Roosen (1998), and Morin, Wahl, and Molnvik (2011). On the other hand, Gassner and Marechal (2009) have shown that total site optimization problems, in which the process is optimized together with the heat exchanger network and the utility systems, can be successfully decomposed into bilevel programs with a non-smooth black-box problem at the upper level, and a Mixed Integer Linear Problem (MILP) at the lower level. A similar bilevel decomposition is typically adopted to tackle steam cycle optimization problems, as in Colmenares and Seider (1989) and Martelli, Amaldi, and Consonni (2011). Also dynamic process optimization problems are often converted into black-box problems in which the optimization algorithm selects the values of the independent decision variables, and the dependent state variables as well as the objective function value are determined by the so-called black-box, i.e., by solving a set of differential-algebraic equations with a dedicated numerical method (see, e.g., Adam & Seider, 2008; Egea, Rodrigues-Fernandez, Banga, & Marti, 2007; Moles, Mendes, & Banga, 2003; Rodrigues-Fernandez, Egea, & Banga, 2006).

Depending on the black-box structure, the objective function may be noisy (e.g., see Plot A of Fig. 1), non differentiable (e.g., see Plot B), discontinuous (e.g., see Plot C), and even not defined in some solution $\boldsymbol{x}$ of the feasible region determined by the constraints (e.g., see Plot D). The black-box may indeed fail to return a value for some values of the variable vector $\boldsymbol{x}$ because either a physical implicit constraint is violated and does not allow the evaluation of the objective function (e.g., in chemical process optimization, a tray of an absorption column of the plant dries up or floods), or the black-box algorithm does not reach convergence for some numerical issues. Moreover it may be impossible to identify a priori a constraint function which describes where the objective function evaluation fails. Such constraints, which cannot be explicitly introduced in the problem formulation, are typically referred to as “hidden constraints” (Conn, Scheinberg, & Vicente, 2009).

* Corresponding author. Tel.: +39 0523356894; fax: +39 0523 623097.
E-mail address: emanuele.martelli@polimi.it (E. Martelli).

Nomenclature

Acronyms

CMAES	Covariance Matrix Adaptation Evolution Strategy
CPSO	Constrained Particle Swarm Optimizer
CRS	Controlled Random Search method
DIRECT	Dividing Rectangle method
eSS	enhanced Scatter Search algorithm
GA	Genetic Algorithms
GENOCOP III	Genetic Algorithm for Numerical Optimization of Constrained Problems
GPS	Generalized Pattern Search
GSS	Generating Set Search
HRSC	Heat recovery steam cycle
IEEE	Institute of Electrical and Electronic Engineers
IGCC	Integrated gasification combined cycle
LJRS	Luus–Jaakola random search
LP	Linear program
LTMADS	Lower Triangular Mesh Adaptive Direct Search
MADS	Mesh Adaptive Direct Search
MILP	Mixed Integer Linear Program
MINLP	Mixed integer non linear program
NLP	Nonlinear program
NPV	Net Present Value
ORC	Organic Rankine Cycle
OrthoMADS	Mesh adaptive direct-search with orthogonal poll directions
PGS-COM	Particle Generating Set – Complex algorithm
PSO	Particle Swarm Optimizer
PSwarm	Particle swarm – pattern search hybrid algorithm
RS	Random Search method
SA	Simulated Annealing method
SS	Scatter Search methodology
SSm	Scatter Search algorithm for Matlab
TCPSO	Truncated Constrained Particle Swarm Optimizer
VNS	Variable Neighborhood Search
VNS-MADS	Hybrid Variable Neighborhood Search – Mesh Adaptive Direct Search Algorithm
εCDE	εConstrained Differential Evolution method

Symbols

$\boldsymbol{A}$	matrix of coefficients of the linear inequality constraints
$\boldsymbol{a}$	vector denoting the rows of the matrix $\boldsymbol{A}$
$\boldsymbol{b}$	right-hand side vector of the linear inequality constraints
c	constant adopted to generate discontinuous test problems
C	set of neighboring particles in the particle array
c_0, c_1, c_2	constant parameters of the particle velocity update formula
$\boldsymbol{d}$	poll direction considered by GPS and GSS methods
D	set of coordinate poll directions
D_{COM}	set of poll directions generated by the last Complex step
D_{TCPSO}	set of poll directions generated by the last TCPSO step
$\boldsymbol{e}$	unit vector of the canonical basis of $\Re^n$
$f(\boldsymbol{x})$	objective function
$f_{\mathrm{eb}}(\boldsymbol{x})$	extreme barrier objective function
$f_m(\boldsymbol{x})$	modified discontinuous objective function
$f_n(\boldsymbol{x})$	modified noisy objective function
g	index denoting the particle with the best solution value found so far among those of the swarm
$\boldsymbol{g}(\boldsymbol{x})$	vector of functions of nonlinear inequality constraints
G	set of core poll directions (used by GSS methods)
$h_k(\boldsymbol{x})$	discontinuous penalty function used to generate discontinuous test problems
H	set of additional poll directions (used by GSS methods)
i	index denoting the ith particle of the swarm
$I(\boldsymbol{x}, \varepsilon)$	set of linear constraints which are ε-active in $\boldsymbol{x}$
Ini	counter of the iterations of the initialization algorithm
Ite_{C}	counter of the contraction updates carried out by the Complex algorithm
Iun_{GSS}	counter of consecutive unsuccessful GSS steps within PGS-COM
Iun_{TCPSO}	counter of consecutive unsuccessful TCPSO steps within PGS-COM
j	index denoting the jth component (decision variable) of the solution vector $\boldsymbol{x}$
k	index denoting the kth linear constraint of the optimization problem
$\boldsymbol{l}$	lower bound vector of the optimization problem
L	set of indices of linear constraints which could be violated by a particle move in TCPSO
m	number of linear inequality constraints
n	number of optimization variables
$N(\boldsymbol{x}, \varepsilon)$	cone positively generated by the normals to the ε-active linear constraints
N_{COM}	number of consecutive reflections executed within Complex step of PGS-COM
N_{f}	number of function evaluations
N_{GSS}	threshold number of consecutive TCPSO iterations of the PGS-COM method
N_{INI}	maximum number of iterations of the initialization algorithm
N_{p}	number of particles of the swarm
N_{np}	number of neighboring particles
N_{s}	number of solutions contained in the set S
N_{TCPSO}	threshold number of consecutive TCPSO iterations of the PGS-COM method
p	number of linear inequality constraints
P	set of solutions polled in GPS and GSS methods
q	index denoting the particle with the best solution value found so far among a particle neighborhood
$\boldsymbol{Q}$	orthogonal matrix of the QR decomposition of $\boldsymbol{Y}^{\mathrm{T}}$
t	iteration counter
S	set of starting solutions used by the Complex algorithm
T	temperature parameter (parameter of simulated annealing methods)
$T(\boldsymbol{x}, \varepsilon)$	tangent cone to the ε-active linear constraints
$\boldsymbol{u}$	upper bound vector of the optimization problem
U	set of outward pointing normals to the ε-active linear constraints
$\boldsymbol{v}$	particle velocity vector
V	subset of feasible solutions (considered within the Controlled Random Search method)
W	set of core poll directions to be considered by the whole algorithm (for GPS and GSS methods)
$\boldsymbol{x}$	$n \times 1$ dimensional vector of decision variables of the optimization problem
$\boldsymbol{y}$	$n \times 1$ dimensional vector denoting the best solution found so far by the swarm particle

$\boldsymbol{Y}$	matrix whose columns are the rows of $\boldsymbol{A}$ corresponding to the ε-active linear constraints
Z	set of solutions returned by the Complex step of PGS-COM
α	step size parameter (used by GPS and GSS methods)
β	mesh size parameter of MADS algorithms
γ	reduction coefficient of the particle velocity computed with respect to the linear constraints
δ	parameter of the initialization algorithm
$\delta_n(\boldsymbol{x})$	noise generator function
ε	tolerance parameter to identify the ε-active linear constraints
ε_n	parameter of the noise generator function $\delta_n(\boldsymbol{x})$
λ	parameter of the Complex algorithm
ρ	reflection coefficient considered in the Complex algorithm
ϕ	pseudo-random number uniformly distributed between 0 and 1 considered in the Complex algorithm
$\phi_n(\boldsymbol{x})$	component of the noise generator function $\delta_n(\boldsymbol{x})$
$\varphi_n(\boldsymbol{x})$	component of the noise generator function $\delta_n(\boldsymbol{x})$
τ	tolerance parameter to define the "data profiles"
χ	reduction coefficient of the particle velocity computed with respect to bound constraints
$\varnothing$	empty set
$\|\|\boldsymbol{a}\|\|_1$	1-norm of vector $\boldsymbol{a}$
$\|\|\boldsymbol{a}\|\|_2$	Euclidean norm of vector $\boldsymbol{a}$
$\|\|\boldsymbol{a}\|\|_\infty$	infinity norm of vector $\boldsymbol{a}$

Many non-smooth problems with black-box objective function, which arise in process engineering, can be formulated as follows:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{g}(\boldsymbol{x}) \le \boldsymbol{0} \\ & \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u} \\ & \boldsymbol{x} \in \Re^n, \end{aligned} \tag{1}$$

where f is the objective function, $\boldsymbol{x}$ the vector of decision variables, $\boldsymbol{u}$ and $\boldsymbol{l}$ the upper and lower bound vectors, $\boldsymbol{A}$ and $\boldsymbol{b}$ are respectively the $m \times n$ matrix of coefficients and the m-dimensional right hand side vector of the linear inequality constraints, and $\boldsymbol{g}(\boldsymbol{x})$ denotes the p-dimensional vector of functions of nonlinear inequality constraints.

The set of constraints is usually subdivided into two subsets: relaxable and unrelaxable constraints. Unrelaxable constraints cannot be violated by any considered solution because they guarantee either the successful evaluation of the black-box function (i.e., if they were violated, the black-box solver would fail to return an objective function value) or the physical/structural feasibility of the solution. For instance, a verification procedure which evaluates the structural feasibility of a distillation column is an unrelaxable constraint since its output is a binary variable (equal to 1 if the column structure satisfies all the verification criteria, and 0 otherwise). Relaxable constraints may instead be violated as the objective function evaluation is still successful. Typically, constraints related to performance and costs are relaxable while those related to physical laws or technological limits (e.g., energy and mass balances, maximum/minimum pressures, temperatures, and pressure/temperatures differences, etc.) are unrelaxable.

Due to the presence of discontinuities in the objective function, gradient or Hessian-based methods are not appropriate for tackling Problem (1). Since discontinuous functions cannot be closely represented by surrogate models, also model-based derivative-free methods (where the descent direction is estimated on the basis of an objective function surrogate) seem not to be appropriate, as discussed in Audet and Dennis (2006), Audet, Bechard, and Chaouki (2008a), Audet, Bechard, and Le Digabel (2008b), Audet, Dennis, and Le Digabel (2010), Custódio, Dennis, and Vicente (2008) and Vicente and Custodio (2012). To develop efficient methods for this type of problems, it is thus reasonable to focus on the class of direct-search methods in which search directions are determined by directly evaluating the objective function (Conn et al., 2009). See Section 2 for a brief description of the main direct-search methods. In the last decade a growing attention has been dedicated to non-smooth problems. On one hand, the mathematical programming community has been investigating the global convergence properties of methods like Mesh Adaptive Direct Search of Audet and Dennis (2006) (see, e.g., Vicente & Custodio, 2012) and, on the other hand, engineers have been using population-based algorithms, such as particle swarm, Genetic Algorithms and scatter search also on challenging real-world non-smooth problems (see some examples Section 3). Thus, there is a need for comparing the available direct-search methods applicable to Problem (1), and for the development of more efficient ad hoc methods.

In this work we propose a novel hybrid direct-search method for the global optimization of black-box problems with non-differentiable and discontinuous objective function subject to relaxable, unrelaxable as well as hidden inequality constraints. From now on, to be concise, we refer to this class of problems as non-smooth black-box problems. The idea is to appropriately adapt and combine three search strategies, namely Constrained Particle Swarm of Hu and Eberhart (2002), Generating Set Search of Lewis, Shepherd, and Torczon (2007) and the Complex of Box (1965), so as to better exploit their potentialities while reducing their individual weaknesses. The resulting algorithm is called "Particle Generating Set – Complex" and referred to as "PGS-COM".

This novel method is part of an ongoing research project of the Department of Energy of Politecnico di Milano and Laboratorio Energia Ambiente Piacenza (LEAP) which aims at developing a modular simulation software for energy systems. This software, called "GS", has been successfully applied to the analysis of complex energy systems, see for instance Consonni (1992), Chiesa and Lozza (2005) and Chiesa, Consonni, Kreutz, and Williams (2005).

This article is organized as follows. In Sections 2 and 3, we briefly review the available direct-search methods for non-smooth black-box optimization, and some typical applications in process and energy engineering. Section 4 is devoted to the detailed description of our PGS-COM algorithm. In Section 5, we summarize the twenty two test problems with challenging non-smooth objective functions and unrelaxable constraints used in the computational experiments. In Section 6, we specify the eleven direct-search algorithms used for comparison purposes. In Sections 7 and 8, we report and compare the results for the twenty two test problems, and respectively, for two challenging process design problems, namely, the optimization of a styrene production plant (Audet, Bechard, & Chaouki, 2008a; Audet, Bechard, & Le Digabel, 2008b) and the design of a complex heat recovery steam cycle (Martelli, Amaldi, et al., 2011).

2. Brief review of direct-search algorithms

As described in Conn et al. (2009), derivative-free methods can be classified into "direct" or "model-based" algorithms. In direct ones, search directions are determined by directly evaluating the objective function. In model-based ones, a surrogate model of the objective function (e.g., a quadratic best fit function) is used to guide the search process. Although a review and performance

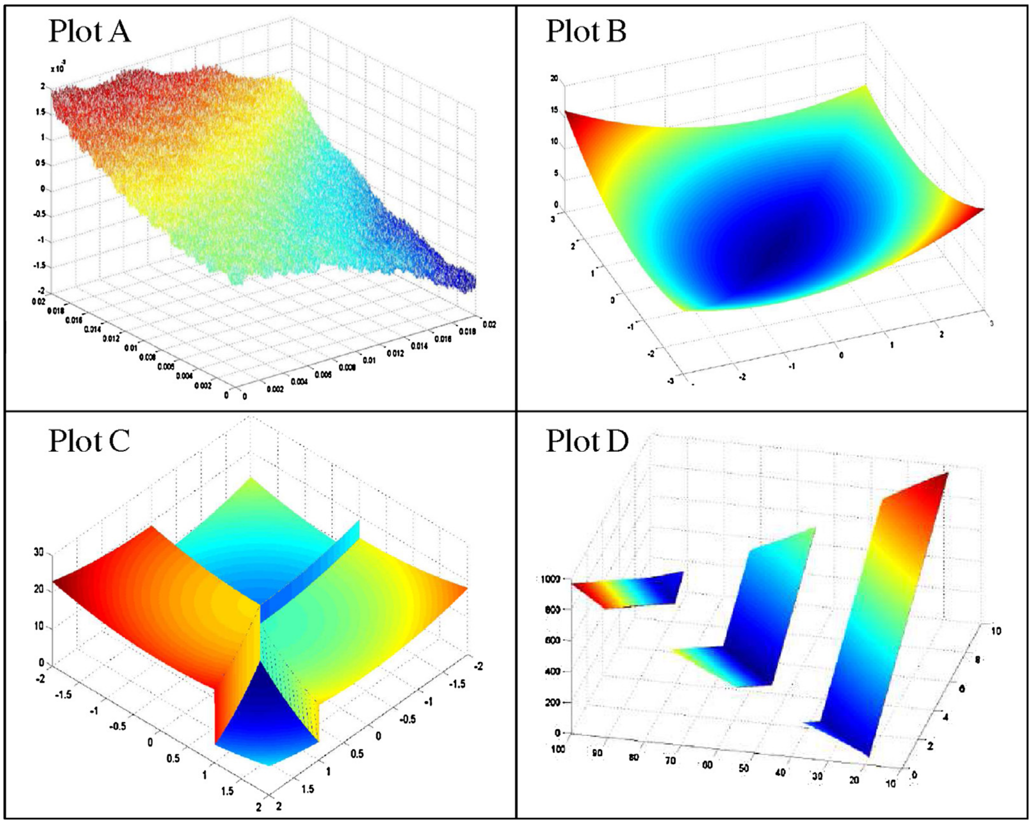

Fig. 1. (Plot A) Plot of a noisy function representative of simulations that are defined by iterative processes. (Plot B) Tridimensional plot of the non-differentiable Dennis-Wood-type function as described in Conn et al. (2009). (Plot C) Plot of the function with step type discontinuities presented in Vicente and Custodio (2012). (Plot D) Plot of the non-smooth function not defined in some regions considered in Buzzi-Ferraris and Manenti (2010).

comparison of the existing derivative-free codes is presented by Rios and Sahinidis (2013), attention is restricted to smooth and continuous non-differentiable problems with bound constraints, like the piecewise linear test problems of Richtárik (2009) and some of the non-differentiable problems of the collection by Luksan and Vicek (2000). Instead, the main focus of this paper is on non-smooth (non-differentiable and discontinuous) problems with unrelaxable and hidden constraints. In this section, we briefly describe the existing local and global direct-search methods applicable to black-box problems with general inequality constraints. Particular attention is devoted to the capability of handling unrelaxable constraints and function discontinuities. For the sake of presentation, we subdivide such methods into six main classes:

1. Complex methods.
2. Random Search (RS) and Simulated Annealing (SA) methods.
3. Population-based methods: Genetic Algorithms, Particle Swarm, Scatter Search.
4. Dividing Rectangle (DIRECT) methods.
5. Directional direct-search methods: Generalized Pattern Search (GPS), Generating Set Search (GSS), and Mesh Adaptive Direct Search (MADS).
6. Hybrid methods.

2.1. *The Complex method*

In the Complex method proposed by Box (1965), starting from a set S of at least $n+1$ randomly generated feasible solutions (where n denotes the number of variables), at each iteration the worst solution $\boldsymbol{x}_{\mathrm{w}}$ of the set is updated by considering the solution $\boldsymbol{x}_{\mathrm{r}}$ obtained by reflecting it through the centroid of the other n solutions. If this solution satisfies the constraints and has better objective function value than $\boldsymbol{x}_{\mathrm{w}}$, then $\boldsymbol{x}_{\mathrm{r}}$ replaces $\boldsymbol{x}_{\mathrm{w}}$. Otherwise, we iteratively move $\boldsymbol{x}_{\mathrm{r}}$ toward the centroid by a random amount. If the feasible region is convex, this allows to reach feasibility. This procedure is repeated until it stops improving the worst solution or the set S collapses into a solution (i.e., the solutions in S get so close to each other that no progress can be made with the reflection steps) or the maximum number of function evaluations is reached. Guin (1968) and Andersson (2001) proposed some improvements in the updating formula (see Section 4.1) to avoid premature convergence. A sequence consisting of two Complex iterations is represented in Fig. 2.

According to our computational experiments described in Section 7, the Complex version of Andersson (2001) is quite effective to tackle problems with non-smooth objective functions and unrelaxable constraints. However, the method often fails to find the global minimum of functions with many valleys or numerical noise: as pointed out by Conn et al. (2009), numerical noise in the objective function affects simplicial direct-search methods like the Simplex of Nelder and Mead (1965) and the Complex. For instance, when trying to replace the worst solution of the solution set S, the Complex algorithm may carry out many contraction iterations which prematurely shrink the set S and limit the method exploration capability.

Moreover, since at each iteration the algorithm generates and evaluates a single solution at time, the procedure cannot be easily parallelized. Given the recent diffusion of multiple-core processors, most evolutionary methods as well as pattern search methods are implemented so as to allow the simultaneous evaluation of the considered solutions on the available processors (e.g., by assigning the evaluation of one solution to each core). This feature is particularly advantageous for problems with computationally expensive black-box function. Thus, despite its effectiveness, the Complex method

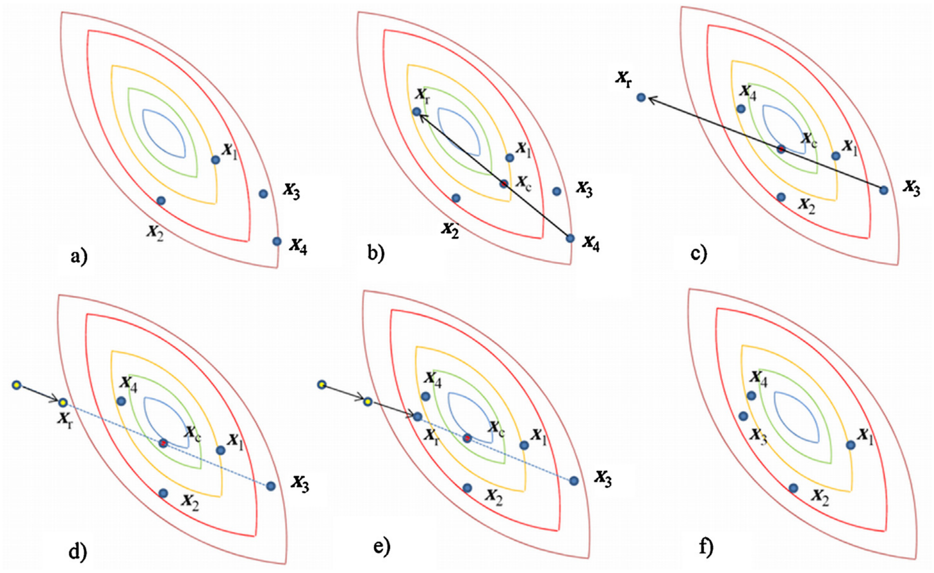

Fig. 2. Sequence of two complex iterations according to the updating formula of Box (1965). Frame (a) shows the initial set of solutions. Frame (b) shows the first iteration: the worst solution $\boldsymbol{x}_4$ is reflected with respect to the centroid of the other solutions $\boldsymbol{x}_c$. Then, since the new reflected solution $\boldsymbol{x}_r$ is better than $\boldsymbol{x}_4$, it replaces $\boldsymbol{x}_4$ in the simplex. Frame (c) corresponds to the second iteration: the worst solution $\boldsymbol{x}_3$ is reflected with respect to the centroid. However the new solution $\boldsymbol{x}_r$ is worse than $\boldsymbol{x}_3$ and then, it is progressively moved toward $\boldsymbol{x}_c$ up to find a solution better than $\boldsymbol{x}_3$, as shown in frames (d) and (e). Frame (f) shows the set of solutions at the end of the second iteration.

is not well suited for problems with computationally expensive objective functions.

2.2. Random Search and Simulated Annealing methods

Random Search methods are stochastic methods that rely solely on the random sampling of a sequence of solutions in the feasible region of the problem. These methods have been attracting attention for global optimization from the work of Brooks (1958) because of their ability to tackle problems whose mathematical structure is difficult to analyze (e.g., black-box functions), and the existence of asymptotic convergence results (if the method satisfies a certain set of conditions, the sequence of iterates convergences to the a global optimum with probability one). The basic idea is to generate a sequence of independent feasible solutions in the feasible region while keeping track of the best solution that is found. For instance, Luus and Jaakola (1973b) proposed the "LJ Random Search" (LJRS) algorithm based on the idea of progressively reducing the search domain around the best solution. At each iteration, a set of solutions is randomly generated by summing random components to the current best solution. The range of the random components is progressively reduced in order to intensify the exploration around the best current solution. The LJRS algorithm has been successfully applied to many chemical engineering problems by Luus and Jaakola (1974) and Spaans and Luus (1992), however it requires a large number of function evaluations. A different approach, called Hit and Run, is proposed by Goulcher and Casares (1978), and lately improved by Banga and Seider (1996), Smith (1984) and Zabinsky et al. (1993). The search moves over the feasible region by generating a direction vector that is uniformly distributed over a hypersphere, and a step length that is uniformly distributed in the range corresponding to the current best solution and the boundary of the feasible region. Such approach turns out to be very effective on multimodal functions as well as chemical engineering problems, as shown by Banga and Seider (1996). Price (1983) developed the Controlled Random Search (CRS) algorithm, a robust method suitable for nonlinear constrained problems. A set of solutions, sufficiently larger than the number of variables n, is generated and evaluated. At each iteration, a subset V of $n+1$ solutions is randomly selected and a new solution is generated as the reflection of one, randomly chosen, solution of V through the centroid. If the new solution does not satisfy all the constraints or does not improve the worst solution, it is rejected and a new solution is generated by performing a new iteration with a different subset of solutions. More recently Kaelo and Ali (2006) proposed a CRS algorithm that combines different solution generation schemes on a probabilistic basis. The reflection scheme of Price (1983) can be probabilistically replaced by either a new global solution generation scheme based on linear and quadratic interpolation, or a local mutation procedure (corresponding to a local search step around the current best solution). According to their numerical experiments, this new scheme considerably improves the method robustness and effectiveness in terms of number of function evaluations. Therefore we will use it for comparison purposes (see Sections 7 and 8).

Simulated Annealing algorithms are essentially RS methods in which the new solutions, generated according to a sequence of probability distributions (e.g., the Boltzmann distribution) or a random procedure (e.g., a Hit and Run algorithm), may be accepted even if do not lead to an improvement of the objective function. The candidate solution is accepted with a probability equal to $\min\{1, e^{(f_{\text{best}}-f_{\text{new}})/T}\}$, where T is called the "temperature" parameter, f_{best} denotes the best value found so far and f_{new} the function value of the new solution. The temperature parameter is decreased monotonically with f_{best} in order to generate a probability distribution which progressively accepts only improving solutions. The most known methods for constrained real variable problems are those proposed by Romeijn and Smith (1994), Wah and Wang (1999), and Hedar and Fukushima (2004).

Romeijn and Smith (1994) proposed the Hide and Seek algorithm, essentially a Hit and Run RS algorithm accepting also deteriorations of the objective functions according to a probability distribution. The approach is capable of handling both linear and nonlinear unrelaxable constraints. Linear constraints are treated at the algorithmic level by computing the maximum allowed step length, while nonlinear constraints are guaranteed by rejecting infeasible solutions. The authors prove probabilistic convergence to stationary points for continuous (with no condition on differentiability) functions. However, all the test problems reported in

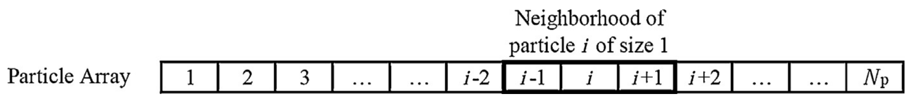

Fig. 3. Particle array: array in which the particles are ordered during the PSO execution and which determines the particle neighborhood. The particle neighborhood for the *i*th particle is indicated in bold.

the paper are smooth, and then, the performances of the algorithm on non-smooth problems are not known. The approach of Wah and Wang (1999) is based on the necessary and sufficient conditions for discrete constrained local minima in the theory of discrete Lagrange multipliers. The method looks for the saddle point of the discrete Lagrangian function (a point which is maximum with respect to the Lagrangian multipliers and minimum with respect to the problem variables). Instead Hedar and Fukushima (2004) reformulate the problem as a multi-objective optimization problem so as to take the form of optimizing two functions: the objective function and a constraint violation function.

Even if the performance of these simulate annealing methods on discontinuous problems and problems with hidden and unrelaxable constraints has not been evaluated, it is worth noting that only the Hide and Seek method is capable of handling nonlinear unrelaxable constraints. The other two methods allow the violation of nonlinear constraints.

2.3. *Population-based algorithms*

This broad class contains all the meta-heuristic search methods which make use of a population of solutions to orient the exploration. Such class comprises not only the well-known Genetic Algorithms but also other evolutionary methods, such as Differential Evolution, Memetic Algorithms, Ant Colony, Particle Swarm, and Scatter Search. We focus the attention on Genetic and Differential Evolution algorithms, Particle Swarm, and Scatter Search which have been successfully applied to the global optimization of non-smooth black-box problems with continuous (real) variables.

2.3.1. *Genetic and Differential Evolution Algorithms*

Genetic Algorithms (GA) and Differential Evolution techniques have been attracting attention from the first publication of Holland (1975). Many authors have extended their use to continuous variables problems. This class of methods seems to work well on multimodal black-box functions for their exceptional exploration capability. However, similarly to the Particle Swarm, GAs are capable of quickly finding promising regions of the search space but may take a relatively long time to reach the optimal solution. Despite the large number of GA codes currently available, only few of them are designed to handle constraints and, more in detail, unrelaxable constraints. For example, the Covariance Matrix Adaptation Evolution Strategy (CMAES) of Hansen, Muller, and Koumoutsakos (2003) and the ε Constrained Differential Evolution (εCDE) method of Takahama and Sakai (2006), best codes respectively in the 2005 and 2006 IEEE (Institute of Electrical and Electronic Engineers) Congress on Evolutionary Computation Benchmark, are not designed to handle unrelaxable constraints. CMAES, originally developed for bound constrained problems, treats linear as well as nonlinear constraints with a penalty approach while εCDE uses a constraint violation approach to progressively move the unfeasible solutions into the feasible region.

To the best of our knowledge, one of the few GA which is designed to handle general unrelaxable constraints is GENOCOP III (Genetic Algorithm for Numerical Optimization of Constrained Problems) of Michalewicz and Nazhiyath (1995). The population is divided into two groups, search solutions and reference solutions. The reference solutions satisfy all the constraints (both linear and nonlinear) while the search solutions may be unfeasible with respect to the nonlinear constraints. Unfeasible search solutions are repaired by moving them progressively toward one (randomly selected) of the reference solutions. Additionally, if a repaired search solution $\boldsymbol{x}_s$ is better than that of the reference solutions $\boldsymbol{x}_r$ (with respect to the objective function), then $\boldsymbol{x}_s$ replaces $\boldsymbol{x}_r$ becoming a reference solution.

2.3.2. *Particle Swarm Optimizer*

In the Particle Swarm Optimizer (PSO) proposed by Kennedy and Eberhart (1995), starting from a population of random solutions (particles) initialized over the feasible region, the particles move across the variables space attracted by both the current best solution found by the swarm (or subset of the swarm) particles and the best solution found so far by the particle itself. In Kennedy and Eberhart (1995), two versions of the particle swarm optimizer are presented: the *g-best* and the *l-best* versions. In the *g-best* version (which stands for global best version), the particles are attracted by the best solution found so far by the complete swarm and the best solution found so far by the particle itself. This strategy may lead to premature convergence or the swarm may get trapped into local minima. In the *l-best* (local best version), the particles are stored in an array and each particle moves toward a linear combination of the best solution found so far by the particle itself and the best solution found by the N_{np} $(1, 2, \ldots)$ closest particles in the particle array (see Fig. 3).

Although this evolution strategy slows down the swarm convergence rate, it improves its exploration ability, and thus increases the probability of finding a global optimum.

Hu and Eberhart (2002) and Hu, Eberhart, and Shi (2003) have successfully extended PSO, originally developed for bound constrained problems, to the case with any type of constraints, and even black-box functions. In the constrained PSO (CPSO) method, PSO is applied to the extreme barrier function, i.e., the objective function is set equal to infinite if any of the constraints is violated. Thus, particles which fall out of the feasible region are forced back into the feasible region by the attraction of the best (feasible) solutions found so far.

Unlike the Complex method, CPSO can exploit parallelization in the objective function evaluations. The *g-best* CPSO version has also been successfully applied to the optimization of discontinuous steam cycle design problems by Martelli (2010), Martelli, Amaldi, et al. (2011), Martelli, Nord, and Bolland (2012) and Martelli, Kreutz, Gatti, Chiesa, and Consonni (2013). Nevertheless, it has the two following shortcomings: (1) if the function is multimodal or has many steps, it may fail (as already pointed out by Eberhart and Kennedy, 1995) to locate the global optimum, (2) due to the lack of a systematic search strategy, particles inefficiently oscillate around minima, as described in Fan and Zahara (2007) and Fu and Tong (2010).

2.3.3. *Scatter Search algorithms*

Scatter Search (SS) was proposed by Glover (1977) for integer programming and further developed and extended to real variable problems by Laguna and Martì (2003). In contrast to evolutionary algorithms which generate new solutions making large use of randomization, scatter search methods rely on deterministic solution combination criteria with minor random components.

SS algorithms are based on the following five methods (the so-called "five-method template"):

1. A "Diversification Generation Method" which generates a relatively large set of diverse new solutions (not necessarily feasible), using an arbitrary solution as an input.
2. An "Improvement Method" which starts from the new solutions and attempts to improve them (also these improved solutions may not be feasible). Potentially, such method can be any local search algorithm.
3. A "Reference Set Update Method" to determine at each iteration a limited set containing the "best" solutions found. Solutions are evaluated on the basis of their objective function value and their "diversity".
4. A "Subset Generation Method" to select a few solutions of the reference set as candidates for the "Solution Combination Method".
5. A "Solution Combination Method" to transform the candidate solutions selected by the Subset Generation Method into one or more combined solutions. Such solutions can be enhanced by applying the "Improvement Method".

After generating the reference set, each SS iteration includes a subset generation step, a solution combination step, a solution improvement step and a reference set update step. The algorithm stops when none of the enhanced solutions generated by the solution combination step is admitted to the reference set by the "Reference Set Update Method". In SS algorithms, to make the search flexible, constraints are typically handled with the penalty function approach.

Starting from the above described template, Egea et al. (2007) proposed a SS algorithm for constrained black-box process optimization problems. In particular, the authors tailored the five basic methods of the SS scheme for process engineering problems taking into account the typical problem features (noise, flat areas, and discontinuities). The solution improvement method can be chosen among a broad set comprising several gradient-based algorithms as well as Pattern Search algorithms. The SSm code, is implemented in Matlab and can be downloaded from http://www.iim.csic.es/~gingproc/ssmGO.html.

2.4. The Dividing Rectangle method

The Dividing Rectangle (DIRECT) algorithm of Jones, Perttunen, and Stuckman (1993) is global optimization method which mimics a sampling technique. The feasible region is divided into a number of hyper-rectangels whose centers are the DIRECT's sample solutions. In each iteration, new hyper-rectangles are formed by dividing old ones, and then the function is sampled at the centers of the new hyper-rectangles. In the first phase, DIRECT identifies hyper-rectangles that show the most potential to contain good, unsampled solutions. Potentially optimal hyper-rectangles either have low function values at the centers or are large enough to be good targets for global search. The second or sampling phase is to sample the objective function at the centers of the newly created hyper-rectangles. DIRECT typically terminates when a user-supplied budget of function evaluations is exhausted. The original version of Jones et al. (1993) was developed for bound constrained problems and can be extended to instances with general relaxable linear and nonlinear constraints by means of a quadratic penalty approach. Carter, Gablonsky, Patrick, Kelley and Eslinger (2001) develop a DIRECT version which assigns an artificial value to infeasible solutions with the neighborhood assignment strategy of Gablonsky (2001). The penalty approach does not guarantee the satisfaction of the unrelaxable constraints, while the neighborhood assignment strategy may not be appropriate for discontinuous problems. Indeed, our preliminary computational experiments with the DIRECT implementation of Finkel (2003) show that the algorithm is not well suited for non-smooth problems subject to unrelaxable constraints.

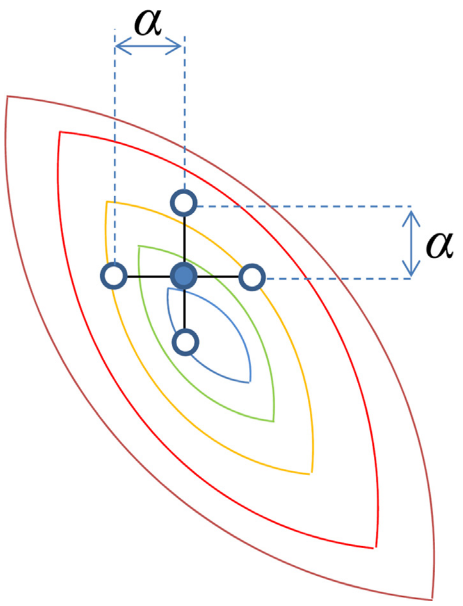

Fig. 4. Example of GPS poll step adopting the coordinate directions (i.e., the canonical basis and its negative counterpart) as poll directions. The filled dot denotes the current solution while the empty dots denote the poll solutions and α is the step size parameter.

2.5. Directional direct-search methods

In directional direct-search methods at each iteration the search around the current solution is carried out along a set of directions which is rich enough to guarantee convergence to stationary points. Depending on the set of directions, such methods can be further subdivided into three main types: Generalized Pattern Search, Generating Set Search and Mesh Adaptive Direct Search.

2.5.1. Generalized Pattern Search

Lewis and Torczon (1999, 2000, 2002), and Audet and Dennis (2003) generalized the early Pattern Search algorithm of Hooke and Jeeves (1961). They introduced the class of Generalized Pattern Search (GPS) methods which have been attracting great attention for both unconstrained and constrained black-box problems. According to Audet and Dennis (2003), GPS methods generate a sequence of iterates with non-increasing objective function values where each iteration is either an optional so-called "search step" or a so-called "poll step". In the search step, the objective function is evaluated at a finite number of solutions lying on a mesh (a discrete subset of $\Re^n$ whose fineness is parameterized by the parameter α, as further specified below) to try to find one that yields a lower objective function value than the current solution. Any strategy may be used to search for a better solution, such as an analytical method applied to a surrogate model of the objective function. The search step is optional and not essential for the convergence guarantees. The poll step consists in the exploration of the neighboring mesh points around the current best solution $\boldsymbol{x}^t$. Such poll solutions are of the form $\boldsymbol{x}^t + \alpha^t \boldsymbol{d}^t$ where α is called "step size" or "mesh size" parameter (since in GPS methods the step size is kept equal to the mesh size) and the direction $\boldsymbol{d}$ is called "poll direction". According to the rationale of GPS methods, the set of poll directions must positively span the solution space. Typically the coordinate directions (i.e., the vectors of the canonical basis and their negative counterpart) are adopted (see Fig. 4).

The objective function is evaluated in the poll solutions and if none of the poll solutions has a better function value than the current iterate, the step size parameter is decreased by an appropriate amount. Otherwise, the best poll solution (in the complete polling approach) or the first improving polled solution (in the opportunistic polling approach) is taken as the next iterate $\boldsymbol{x}^{t+1}$.

Although convergence to stationary points is proved for continuously differentiable problems under different assumptions by Yu (1979), Torczon (1997) and Audet and Dennis (2003), for non-smooth problems GPS methods may not converge to a stationary point (minimum), as pointed out by Kolda, Lewis, and Torczon (2003). As illustrated in Fig. 5, if at the current $\boldsymbol{x}^{t}$ the function is non-differentiable or discontinuous, none of the poll directions may be a descent direction, even if $\boldsymbol{x}^{t}$ is not a local minimum. In such cases, the step size parameter may be decreased down to the convergence tolerance without any progress toward the minimum.

Lewis and Torczon (2000) extend GPS methods to linearly constrained problems and prove global convergence to stationary points for smooth functions. For problems with nonlinear constraints, they apply the above GPS algorithm to an augmented Lagrangian function which only includes the nonlinear constraints. The algorithm is globally convergent to stationary points even if the problem is characterized by derivative-free constraints. Since their analysis assumes that nonlinear constraints are relaxable, it is not clear how to adapt this augmented Lagrangian approach to problems with nonlinear unrelaxable (and hidden) constraints.

2.5.2. *Generating Set Search*

In the Generating Set Search (GSS) methods proposed by Kolda, Lewis, and Torczon (2006a) and Lewis et al. (2007), the considered solutions around the current iterate $\boldsymbol{x}^{t}$ are of the form $\boldsymbol{x}^{t}+\alpha^{t}\boldsymbol{d}^{t}$, where the directions $\boldsymbol{d}^{t}$ belong to two different sets, G^{t} and H^{t}. The set G^{t} must contain the directions needed to positively span the variable space and then ensure global convergence with smooth functions. The (possibly empty) set H^{t} contains a finite number of additional search directions that are selected according to some heuristic or specific rules so as to accelerate the progress toward the closest minimum. Among the different GSS variants for linearly constrained problems, the most effective one is that of Lewis et al. (2007). Kolda, Lewis, and Torczon (2006b) also propose a GSS algorithm for problems with general nonlinear constraints based on the augmented Lagrangian function.

It is worth noting that, despite the lack of theoretical guarantees for non-smooth problems, the fact that GSS methods search in multiple directions reduces the probability of getting stuck in situations like those of Fig. 5.

2.5.3. *Mesh Adaptive Direct Search*

For non-smooth problems and problems with unrelaxable constraints, Audet and Dennis (2006) propose the Mesh Adaptive Direct Search (MADS) method. While in GPS and GSS the set W of core polling directions to be considered by the whole algorithm is finite and defined a priori, in MADS W is progressively enriched generating new different polling directions according to a systematic construction procedure. The basic idea is that all the evaluation solutions must lie on an integer lattice, i.e., a mesh defined by all possible integer combinations of the n coordinate directions for a given mesh size parameter β. Since the step size parameter α is not equal to β (as in GPS and GSS methods) but to $n\sqrt{\beta}$, the MADS algorithm can select the poll directions from a set larger than that of the coordinate directions. As shown in Fig. 6, all the solutions of the mesh $\boldsymbol{x}_{k}$ satisfying the condition $||\boldsymbol{x}_{k}-\boldsymbol{x}^{t}||_{\infty}\leq\alpha$ (where $\boldsymbol{x}^{t}$ denotes the current solution and $||\cdot||_{\infty}$ the infinity norm) can be selected as poll solutions.

In Lower Triangular based Mesh Adaptive Direct Search (LTMADS) of Audet and Dennis (2006), a simple algebraic

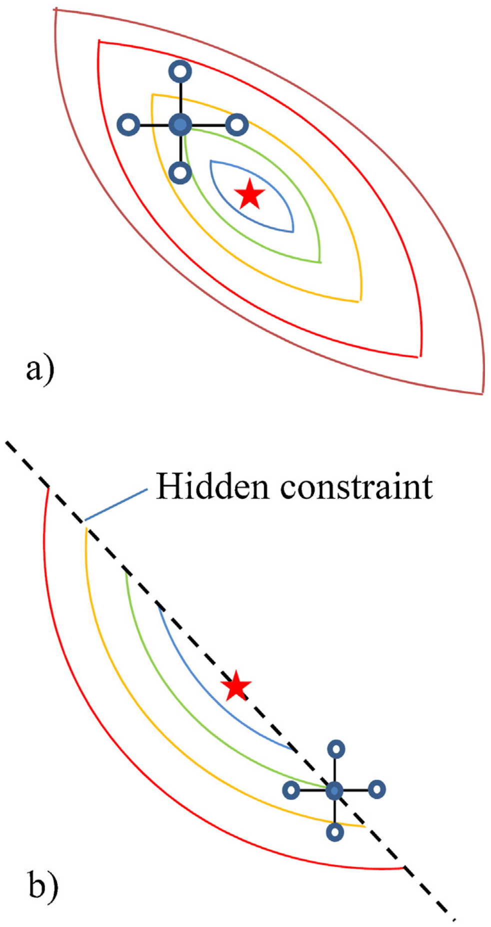

Fig. 5. Two situations in which GPS does not converge to a local minimum (red star). (Case a) Since the objective function is non-differentiable in the current iterate, none of the poll directions is a descent direction for any arbitrary small step size parameter. As a consequence, GPS is stuck in the current solution (filled dot). (Case b) The objective function is smooth but subject to a hidden constraint which does not allow its evaluation. Also in this situation, GPS gets stuck in the current solution (filled dot) because none of the poll directions is a descent direction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

calculation is proposed to produce, as the mesh size approaches zero, a set of polling directions with union that is asymptotically dense in the space of the optimization variables with probability one. This feature allows them to prove that, if the objective function is Lipschitz continuous near the limit solution $\boldsymbol{x}^{*}$ of unsuccessful poll steps, then $\boldsymbol{x}^{*}$ is a Clarke stationary point (i.e., the Clarke generalized directional derivative is nonnegative for any direction of the problem space) with probability one. This result can be extended to constrained problems: if constraints satisfy a certain qualification assumption and are handled with the extreme barrier approach, LTMADS is guaranteed to converge to a Clarke-KKT stationary point with probability one. The reader is referred to Audet and Dennis (2006) for a hierarchy of convergence results. Vicente and Custodio (2012) generalize such convergence results to discontinuous functions using Rockafellar generalized directional derivatives.

Since in the original version of the method (LTMADS) the polling directions are not orthogonal, Abramson, Audet, Dennis and Le Digabel (2009) propose an improved version with orthogonal directions (OrthoMADS). These methods have been successfully applied to challenging constrained non-smooth black-box problems, such as the design of a spent potliners treatment process (Audet, Bechard, & Chaouki, 2008a) and a styrene production process (Audet, Bechard, & Le Digabel, 2008b; Audet et al., 2010).

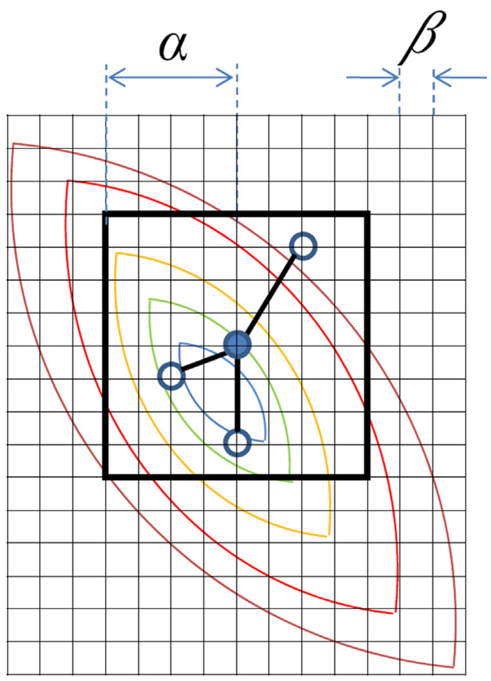

Fig. 6. Example of LTMADS poll step in two dimensions with mesh size parameter $\beta = 1/4$ and step size parameter $\alpha = 1$. The selected poll solutions must be inside the bold square and the related poll directions must positively span $\Re^2$.

2.5.4. *Globalization Strategies for GPS, GSS and MADS algorithms*

Unlike the above-mentioned global optimization methods, GPS, GSS and MADS are local algorithms, in the sense that they tend to converge to a local minimum close to the starting solution. Three globalization strategies can be adopted:

1. "Multi-start": the idea is to carry-out a large number of runs of the algorithms with different starting solutions so as to reasonably sample the feasible region.
2. "Global search step": before each poll step or each specified number of poll steps, an optional global search step is performed by applying either a population-based method (e.g., GA, PSO or DE) or a sampling technique (e.g., the Latin Hypercube method of Stein, 1987). If the global search step returns a solution with an improved objective function value, that solution is used at the next iteration. Note that the algorithm used for the global search step and the poll step do not exchange information, except for the best solution and its value. This strategy is implemented in the Matlab® Global Optimization Toolbox by MathWorks (2013a).
3. "Hybrid methods": the poll step and the global search method interact in such a way that the current parameters (for GSS: step size parameter, poll directions, outcome of the poll step) and the solutions considered by each algorithm are shared and can be used by both of them. The methods described in the next subsection as well as our PGS-COM fall within this class.

2.6. *Hybrid methods*

Several hybrid strategies have been recently developed by coupling the above mentioned algorithms. For instance, Audet, Bechard, & Le Digabel (2008b) have coupled the MADS with the Variable Neighborhood Search (VNS) strategy proposed by Hansen and Mladenovi′c (2001) for combinatorial problems and extended by Drazi′c, Lavor, Maculan, and Mladenovi′c (2004) to problems with real variables. The VNS method is included into the MADS algorithm as a diversification search step. The method, here abbreviated as VNS-MADS, has been tested on several real-world constrained non-smooth test problems, such as the styrene process design problem reported in Section 8.1.

Vaz and Vicente (2007) have proposed a hybrid Particle swarm – generalized pattern search method (PSwarm) which turns out to be very effective on smooth problems with bounds constraints. The key idea is to apply a GPS strategy where each search step consists of a single update iteration of the Particle Swarm method. In Vaz and Vicente (2009) PSwarm is extended to handle general linear inequality constraints. The poll step includes essentially the set of core polling directions used by Lewis and Torczon (2000) in their GPS method for linearly constrained problems. In the search step, the Particle Swarm update formula is adapted so as to account for the maximum feasible displacement of the particles. Moreover, in order to save function evaluations, PSwarm drops particles which become too close to each other. Thus, when particles get close to an optimal solution, most of the particles of the swarm are killed and the search proceeds with GPS poll steps.

Other hybrid methods have been developed by combining the Particle Swarm paradigm with local search algorithms. It is worth mentioning the Hybrid Simplex Search – Particle Swarm for unconstrained and constrained problems proposed respectively by Fan and Zahara (2007) and Zahara and Hu (2008), and the Complex based PSO for generally constrained problems of Fu and Tong (2010). In the one of Fan and Zahara the Simplex Search of Nelder and Mead (1965) is applied to a subset of the best swarm particles. Although this algorithm turns out to be very effective on challenging smooth unconstrained global test problems, it has not been tested on non-smooth objective functions. In the version for constrained problems, particles which violate the constraints are moved toward the feasible region with the gradient repair method proposed by Chootinan and Chen (2006), while the Nelder–Mead search step is modified to handle constraints with the constraint fitness priority-based ranking method of Dong, Tang, Xu, and Wang (2005). The method is successfully tested on the global optimization problems published by Koziel and Michalewicz (1999) and a real engineering problem. However, none of the test functions is non-smooth and the two approaches adopted to handle constraints are not suitable for unrelaxable and hidden constraints.

In the Complex based PSO, the CPSO is applied until the best particles are sufficiently close (differences in function value are sufficiently small) and then the particle update is replaced by iterations of the Complex of Box (1965) applied to those best particles. Unfortunately, the emphasis is on an application and results for a single black-box problem is reported.

Egea, Marti, and Banga (2010) proposed a hybrid evolutionary method, called eSS, inheriting features from scatter search and path-relinking (Glover, Laguna, & Martì, 2004). Compared to SSm (Egea et al., 2007), eSS makes use of a different solution combination method based on path-relinking, a different population update procedure including an intensification step (which replaces the local optimizers used by SSm as solution improvement methods), and a search diversification mechanism. The algorithm, coded in Matlab and downloadable from http://www.iim.csic.es/~gingproc/ssmGO.html, has been successfully applied to a variety of black-box process optimization problems.

2.7. *Remarks*

As previously mentioned, for non-smooth problems with unrelaxable constraints, only Random Search algorithms and MADS variants have some asymptotic convergence guarantees, and to the best of our knowledge only RS, PSO, GA, SS, MADS and GPS algorithms have so far been tested on this kind of discontinuous problems. In Section 7, a detailed evaluation of 11 of the above-mentioned methods, namely Complex, CRS, CMAES, CPSO, SSm, GSS, LTMADS, OrthoMADS, PSwarm, VNS-MADS and eSS, is carried out on constrained non-smooth problems. Their results are compared with those provided by PGS-COM, the new direct-search method that we propose and describe in Section 4.

3. Applications of direct-search methods

Black-box optimization has been applied to the design of chemical processes with modular flowsheet simulation codes from the seventies. For instance, Adelman and Stevens (1972) show the effectiveness of the Complex method of Box (1965) to find the optimal design of a complex plant. Friedman and Pinder (1972), and Gaines and Gaddy (1976) consider as case study a gasoline polymerization process and optimize the plant net earnings with different direct-search methods, namely the Complex of Box (1965), the Pattern Search of Hooke and Jeeves (1961) and a Random Search algorithm. Luus and Jaakola (1973a, 1973b, 1974) apply their "LJ Random Search" algorithm to the design of a variety of chemical processes. In all these works the attention is focused on direct-search methods instead of the more efficient gradient/Hessian based methods, such as the very efficient sequential quadratic programming method, for multiple reasons. First of all, in the black-box approach, the objective function and constraints derivatives cannot be calculated. Moreover, even if derivatives are estimated in terms of finite differences, they may be meaningless because the objective function and the constraints returned by a flowsheet simulator may be noisy, non-differentiable, discontinuous and not defined in some points of the feasible region determined by the constraints. Numerical noise derives from the iterative methods embedded into the process units and used by the flowsheet solver. Of course, in presence of noise, gradient and Hessian of the objective function are not reliable indicators of the descent direction. Non-differentiability typically is caused by "max", "min" operators appearing in the models of process units while step type discontinuities are determined by "if-then-else" statements or process units with discrete parameters which are automatically adjusted by the flowsheet solver. The flowsheet model may even fail to return a value because either a hidden physical constraint is violated and does not allow the evaluation of the objective function (e.g., a tray of an absorption column of the plant dries up or floods), or the black-box algorithm does not reach convergence for some numerical issues. As already mentioned in Section 1, this type of constraints, which do not allow the evaluation of the objective function even if the problem constraints are satisfied, is referred to as "hidden constraints" (Conn et al., 2009).

Although the positive results obtained by Biegler, Grossmann and Westerberg (1997) and Biegler and Cuthrell (1985) with the "equation oriented" approach (in which process optimization and simulation are simultaneous) have attracted considerable attention, the black-box approach is still widely used in process optimization, as confirmed by the works of Banga and Seider (1996), Gross and Roosen (1998), Rodríguez, Mussati, and Scenna (2011) and Morin et al. (2011). The main advantages of the black-box approach are: (1) legacy but still widely used codes and efficient solution methods (specifically customized for each single process unit) can be easily included into the flowsheet model, (2) the flowsheet model is easier to debug in case of input errors or failed convergence, (3) the optimization level considers only the independent decision variables and design specification constraints, since all the stream variables and unit equations are hidden into the flowsheet model. Moreover, the large disadvantage of such black-box approach, that is the computational time required to solve the flowsheet up to convergence for each considered solution, has been diminished by the commercialization of multiple-core processors and the development of parallel versions of direct-search optimization codes, which simultaneously evaluate multiple solutions (i.e., flowsheet designs).

Gassner and Marechal (2009) show how large total-site optimization problems, in which the process is optimized together with the heat exchanger network and the utility system, can be successfully decomposed into bilevel programs with a Mixed Integer Linear Program (MILP) at the lower level and a non-smooth black-box problem at the upper level. The upper level problem optimizes both integer and nonlinear design variables of the process as well as the nonlinear variables (i.e., pressures/temperatures) of the utility systems. The lower level MILP optimizes the process heat recovery network and the utility synthesis/design variables. Due to the presence of discrete variables of the lower level MILP and the possible convergence failures of the process simulation, the objective function of the upper level problem turns out to be non-differentiable, discontinuous and not defined in some points. For instance, the authors use a robust multi-objective evolutionary algorithm to tackle the upper level problem.

A similar bilevel decomposition is at the basis of the two-stage algorithms proposed by Colmenares and Seider (1989), Martelli, Amaldi, et al. (2011), and Martelli et al. (2012, 2013) for the optimization of steam generators, steam cycles and steam networks. In the model of Colmenares and Seider (1989), since some of the constraints may have discontinuous derivatives with respect to the steam pressures and temperatures, the non-linear program (NLP) is decomposed in two-stages: steam pressures and temperatures are optimized in the upper level, while the steam/water mass flow rates and the other design variables are optimized in the lower level. The smooth lower problem is solved with classical derivative-based optimization methods, while the upper level problem, which is often non-smooth, is solved with the Complex method of Box (1965). In Martelli, Amaldi, et al. (2011), the lower level problem is a linear program (LP) and it is solved with the Simplex algorithm, while the upper level program, which is non-differentiable as well as discontinuous, is tackled with the CPSO of Hu and Eberhart (2002). In Section 8.2 we will solve the same heat recovery steam cycle (HRSC) optimization problem for an Integrated gasification combined cycle plant with PGS-COM.

The use of direct-search methods for optimizing cycle pressures and temperatures is not limited to the analysis of steam cycles but is being extended to the study of Organic Rankine Cycles (ORC). For instance, Astolfi, Romano, Bombarda, and Macchi (2013) optimize the efficiency and the economic performance of ORC based plants for low temperature heat recovery by decomposing the problem into two levels, and tackling the upper level problem with the Pattern Search method of Lewis and Torczon (2002). Following a similar approach, De Servi, Tizzanini, Pietra, and Campanari (2013) optimize an ORC plant integrated with a fuel cell using Aspen Plus V. 7.3 (2012) and a Matlab® implementation of the Complex method of Andersson (2001). Santarelli and Pellegrino (2005) apply the Simplex method of Nelder and Mead (1965) to the optimization of a stand-alone energy system based only on renewable sources integrated with a hydrogen production unit. Han, Kokkolaras, and Papalambros (2008) maximize the fuel savings of hybrid fuel cell vehicles by varying the design and control parameters. Since the objective function and the constraints exhibit considerable numerical noise, derivative-free methods are adopted. In particular, both the Divided Rectangles (DIRECT) algorithm developed by Jones et al. (1993) and the MADS method of Audet and Dennis (2006) are used and compared.

Direct-search methods are also applied in the field of dynamic process optimization. According to Biegler (2010), the optimization of dynamic systems includes for instance: the design of distributed systems of reactors and separators, real-time optimization in process control, trajectory optimization for transitions between operating conditions, optimum batch process operation, parameter estimation and inverse problems arising in model building applications, and integrated process design and control problems. Such problems can be formulated as nonlinear programming problems with both differential and algebraic constraints. Although their mathematical properties and optimality conditions are well established and dedicated derivative-based numerical methods have been developed to tackle them directly (see, e.g., Chapter 8 in Biegler, 2010), several authors prefer to convert them into black-box problems and use global direct-search methods. Indeed, many real-world problems present one or more of the following issues:

- lack of an explicit expression of the objective function because it is computed by a simulation code (Egea et al., 2007),
- non-smoothness of the objective function or constraints due to either discontinuous data (e.g., a rectangular pulse perturbation function like that considered in the fourth test problem of Banga and Seider, 1996) or non-smooth changes in the physical regime of the system (such as flow transitions, phase changes, irregularities in the geometry of vessels, the action of non-return valves; see, e.g., Khan et al., 2011 and Schittkowski, 2002),
- over-determined or badly scaled model (Schittkowski, 2002),
- regions in which the solution does not exist,
- multiple local minima as well as regions where the objective function is flat (Rodrigues-Fernandez et al., 2006),
- tight constraints with non-convex feasible region necessary to ensure safe operation of equipment units in multiple operation modes, such as the constraints set by Adam and Seider (2008) to prevent weeping or flooding in a distillation column.

For problems with such issues, it may be necessary to use robust black-box approaches instead of gradient-based techniques. Successful applications of such strategy in dynamic optimization and parameter estimation problems are reported in Moles et al. (2003), Rodrigues-Fernandez et al. (2006), Egea et al. (2007, 2010) and Adam and Seider (2008).

Finally it is worth pointing out that there are also several mechanical and aerospace engineering applications that give rise to non-smooth black-box problems. As shown in Cramer, Dennis, Frank, Lewis, and Shubin (1994), several engineering applications require the use of coupled system simulations. For example, in aeroelastic optimization problems of aircraft wings and helicopter rotor blades, the computational fluid-dynamic software must be coupled with the finite element method software (Booker, Dennis, Frank, Moore, & Serafini, 1998) in order to determine the fluid-dynamic performance (i.e., lift and drag forces) of the inflected blades. In computational engineering, such instances are often referred to as "multidisciplinary design optimization" problems. Another relevant area where direct derivative-free methods are typically used is the optimization of molecular geometries (Meza and Martinez, 1994).

4. The PGS-COM algorithm

Our new hybrid direct-search method is specifically designed for non-differentiable and discontinuous black-box problems with linear and nonlinear relaxable and unrelaxable constraints, like Problem (1). The general idea is to combine the positive features of Constrained Particle Swarm, Generating Set Search and Complex, thus we refer to the algorithm as PGS-COM. Each iteration consists of three steps:

1. a search step corresponding to a population update of a modified Constrained PSO,
2. an optional iteration of the GSS based on an enriched set of poll directions around the best solution found so far,
3. a few optional iterations of the Complex variant of Andersson (2001).

The rationale is to exploit the effectiveness of the population-based CPSO algorithm to rapidly identify promising regions of the set of the feasible solutions, and then take advantage of the effectiveness of the Complex search for non-smooth problems to intensify the search in selected regions. However, since the Complex iterations are computationally expensive (because the objective function evaluations cannot be parallelized) and they may not be effective to tackle noisy problems (see Section 2.2), we first use the GSS local search step with a relatively large step size and apply just a few iterations of the Complex when the GSS fails. The integration between GSS and Complex is aimed at:

- avoiding premature convergence of the GSS poll step to suboptimal solutions of the type illustrated in Fig. 5,
- limiting the number of Complex iterations which cannot be easily parallelized,
- making the local search step more robust with respect to numerical noise in the objective function.

From the computational point of view, it is worth pointing out that in both Particle Swarm and Generating Set Search steps the objective function evaluations involved can be easily parallelized.

4.1. Preliminaries

Before describing the algorithm, we need to summarize in some detail the main aspects of Particle Swarm, Generating Set Search and Complex, which are combined into PGS-COM.

4.1.1. Particle Swarm

In the first paper on Particle Swarm, Kennedy and Eberhart (1995) propose two different formula for updating the particle positions and "velocities" (i.e., displacements): the *l-best* and the *g-best* versions. As already mentioned in Section 2.3, in the *l-best* version each particle is attracted by the best solution that it has found so far and by the best solution found by the $N_{\rm np}$ closest particles in the particle array (see the particle array representation in Fig. 3). In the *l-best* PSO, given the total number $N_{\rm p}$ of particles in the swarm, the neighborhood size $N_{\rm np}$, the iteration index t, the current solution $\boldsymbol{x}_i^{\rm t}$ and best found solution $\boldsymbol{y}_i$ of each particle of index i, one determines the set of neighboring particles of each particle i, $C_i = \{i - N_{\rm np}, i - N_{\rm np} + 1, \ldots, i, \ldots, i + 1, i + N_{\rm np}\}$, and the index q of the particle in C_i with the best solution value $\boldsymbol{y}_{\rm q}$. Then the particle positions are updated according to the formula,

$$\boldsymbol{x}_i^{\rm t+1} = \boldsymbol{x}_i^{\rm t} + \boldsymbol{v}_i^{\rm t+1}, \qquad (2)$$

where the velocity components are computed as follows,

$$v_{i,j}^{\rm t+1} = c_0 \cdot v_{i,j}^{\rm t} + c_1 \cdot r_1 \cdot (x_{i,j}^{\rm t} - y_{i,j}) + c_2 \cdot r_2 \cdot (x_{i,j}^{\rm t} - y_{q,j}), \qquad (3)$$

where the index j denotes the component of the velocity vector and c_0, c_1 and c_2 are three constant parameters. The authors suggest to adopt $c_0 = 0.729$, and $c_1 = c_2 = 1.49445$. Usually the velocity components are limited so as to satisfy the bound constraints.

In the *g-best* version of PSO, the neighborhood size $N_{\rm np}$ is set equal to the size of the swarm, and the swarm updating procedure is the same as in the *l-best* version.

4.1.2. Generating Set Search

At each iteration t of a GSS algorithm, a local poll around the current best solution $\boldsymbol{x}^{\rm t}$ is carried out by exploring a set of solutions defined by the step size parameter $\alpha^{\rm t}$, and by two sets of

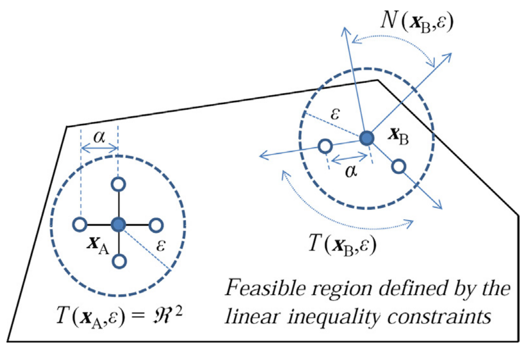
$N(\boldsymbol{x}_B,\varepsilon)$
α
ε
$\boldsymbol{x}_B$
α
$\boldsymbol{x}_A$
ε
$T(\boldsymbol{x}_B,\varepsilon)$
Feasible region defined by the linear inequality constraints
$T(\boldsymbol{x}_A,\varepsilon) = \Re^2$

Fig. 7. Two-dimensional representation of the poll directions used by GSS. In $\boldsymbol{x}_A$, since no linear constraint is ε-active, then GSS uses the coordinate directions which span $\Re^2$. In $\boldsymbol{x}_B$, since two of the five linear constraints are ε-active, GSS computes the generators of $N(\boldsymbol{x}_B,\varepsilon)$, the cone positively generated by the normals to the ε-active constraints, and its polar, $T(\boldsymbol{x}_B,\varepsilon)$, and uses the generators of T as poll directions.

directions G^t and H^t. As previously mentioned, the set G^t must contain the directions needed to positively span the problem space so as to ensure global convergence for smooth functions. The set H^t (possibly empty) contains a finite number of additional search directions that are selected according to some heuristic or specific rules so as to accelerate the progress toward a local minimum. In the unconstrained case, the set G^t can be any positive basis or positively independent spanning set of $\Re^n$. The typical choice is the canonical basis and its negative counterpart, which make the set D of coordinate directions (see Fig. 4). The set of solutions P^t in which the objective function is evaluated at the iteration t, can thus be expressed as:

$$P^t = \{\boldsymbol{x}^t + \alpha^t \boldsymbol{d} : \boldsymbol{d} \in G^t \cup H^t\}. \tag{4}$$

For problems with linear inequality constraints, Lewis and Torczon (2000) show that, if the current solution $\boldsymbol{x}^t$ is close to the border of the feasible region, the set of polling directions must conform the local geometry of the feasible region. More precisely, they prove that in the linearly constrained case GPS algorithms converge to a stationary point if the polling set contains the generators of the tangent cone T relative to the "ε-active constraints" (see Fig. 7). The set of ε-active constraints is defined as

$$I(\boldsymbol{x}^t, \varepsilon) = \{k \in \{1, 2, \ldots, m\} : b_k - \boldsymbol{a}_k^T \boldsymbol{x}^t \le \varepsilon\}, \tag{5}$$

where $\boldsymbol{a}_k^T$ denotes the kth row of the $m \times n$ dimensional matrix $\boldsymbol{A}$ and b_k is the kth element of the m-dimensional vector $\boldsymbol{b}$. The tangent cone T relative to $\boldsymbol{x}^t$ is defined as the polar of the cone N that is positively generated by the normals to the ε-active constraints. Thus

$$T(\boldsymbol{x}^t, \varepsilon) = \{\boldsymbol{v} \in \Re^n : \boldsymbol{w}^T \boldsymbol{v} \le 0 \quad \forall \boldsymbol{w} \in N(\boldsymbol{x}^t, \varepsilon)\} \tag{6}$$

with

$$N(\boldsymbol{x}^t, \varepsilon) = \left\{ \sum_{k \in \mathbf{I}(x^t,\varepsilon)} \lambda_k \boldsymbol{a}_k : \lambda_k \ge 0,\ k \in I(\boldsymbol{x}^t, \varepsilon) \right\}. \tag{7}$$

Moreover, the authors show how the LU factorization of the matrix defined by the ε active constraints can be used to find rational positive generators for the cone $T(\boldsymbol{x}^t,\varepsilon)$ for any $\varepsilon \in [0, \varepsilon^*]$, where ε^* is a positive value independent of the iteration counter t. The fact that the positive generators are rational is important for proving the global convergence of the method to a stationary point. The above mentioned convergence result of Lewis and Torczon (2000) is valid not only for GPS but also for GSS methods. Indeed, the global convergence of both GPS and GSS methods is ensured if set of poll directions positively span the cone $T(\boldsymbol{x}^t,\varepsilon)$, and either the mesh of poll solutions generated by the algorithm lies on an integer lattice (e.g., the mesh consist of positive integer combinations of vectors in G^t) or a sufficient decrease in the value of the objective function is required (for further details see page 134 of Conn et al., 2009).

In the last decade different variants of the approach of Lewis and Torczon (2000) have been proposed for GPS and GSS methods. Lewis et al. (2007) propose one of the most effective GSS variant for handling general (both equality and inequality) linear constraints. Unlike in Lewis and Torczon (2000), the authors yoke the constraint tolerance ε^t with the step size parameter α^t by setting $\varepsilon^t = \alpha^t$. As proved in Kolda et al. (2006a), this choice still ensures global convergence for smooth problems. Moreover, G^t contains the positive spanning set for $T(\boldsymbol{x}^t, \varepsilon^t)$ (which is computed by means of a QR matrix decomposition procedure) and H^t the outward-pointing normals to the set of ε-active constraints (i.e., the generators of N). Since the generators of $T(\boldsymbol{x}^t, \varepsilon)$ necessarily point into the relative interior of the feasible region, the addition of the generators of N allows to move toward the boundary. Based on the numerical tests reported in Lewis et al. (2007), the addition of the augmented directions in H^t significantly improves the convergence rate to an optimal solution.

4.1.3. *Complex algorithm*

In the Complex algorithm of Box (1965), starting from a set of feasible solutions S of cardinality N_S, the worst solution $\boldsymbol{x}_w$ of S is reflected through the centroid of the remaining solutions $\boldsymbol{x}_c$, defined as

$$\boldsymbol{x}_c = \frac{1}{N_S - 1} \sum_{i \in S, i \neq w} \boldsymbol{x}_i. \tag{8}$$

The reflected solution $\boldsymbol{x}_r$ is generated according as follows:

$$\boldsymbol{x}_r = \boldsymbol{x}_c + \rho(\boldsymbol{x}_c - \boldsymbol{x}_r), \tag{9}$$

where ρ is the reflection coefficient. Selecting $\rho = 1.3$ as in Box (1965) gives good results for a large number of problems. If $\boldsymbol{x}_r$ is feasible, it replaces $\boldsymbol{x}_w$ unless it is still the worst one of the set S (including $\boldsymbol{x}_w$). If $\boldsymbol{x}_r$ is infeasible or still the worst of the set S, it is moved progressively toward the centroid $\boldsymbol{x}_c$. The reflection/contraction sequence is repeated as long as the worst solution is improved or the set of solutions S does not collapse into a solution (i.e., the solutions in S get so close to each other that no progress can be achieved with the reflection steps).

Since this updating strategy may lead to premature convergence when there is a local maximum at the centroid, Guin (1968) proposed to gradually move $\boldsymbol{x}_r$ toward the best value $\boldsymbol{x}_b$, if it continues to be the worst value. To avoid that $\boldsymbol{x}_r$ is moved too close to $\boldsymbol{x}_b$ and the set of solutions prematurely collapses, Andersson (2001) introduced a random component in the contraction formula of the new solution:

$$\boldsymbol{x}_r = \frac{1}{2}[\boldsymbol{x}_r + \lambda \boldsymbol{x}_c + (1 - \lambda)\boldsymbol{x}_b] + (\boldsymbol{x}_c - \boldsymbol{x}_b)(1 - \lambda)(2 \cdot \phi - 1), \tag{10}$$

where

$$\lambda = \left(\frac{4}{4 + Ite_C - 1}\right)^{4 + Ite_C - 1/4}. \tag{11}$$

Ite_C is the number of times the solution has repeated itself as lowest value, and ϕ is a uniformly distributed pseudo-random number in the interval [0,1]. This allows the algorithm to explore a wider region of the neighborhood around the best current solution.

4.2. *Problem formulation and unrelaxable constraints*

The non-smooth and noisy black-box optimization problems with non-differentiable and/or discontinuous objective function subject to both linear and nonlinear inequality constraints that

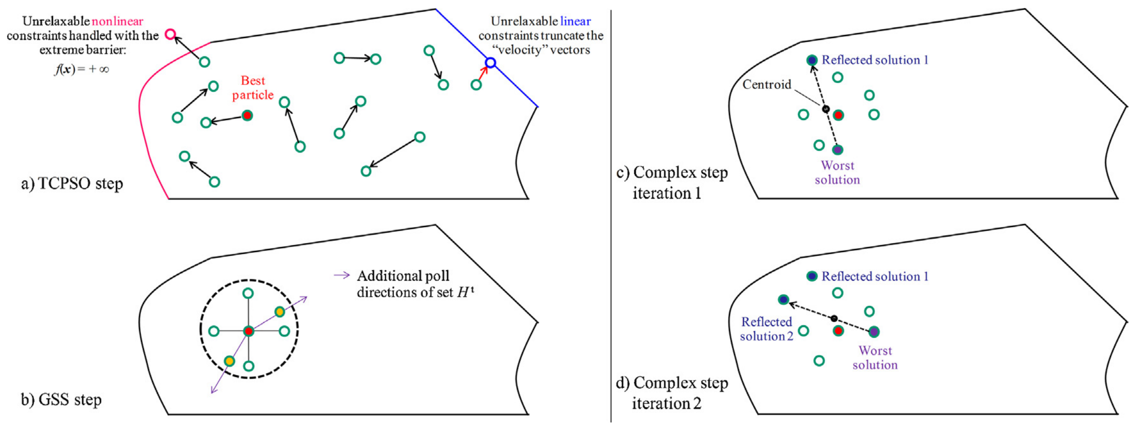
Unrelaxable nonlinear constraints handled with the extreme barrier: $f(\mathbf{x}) = +\infty$
Unrelaxable linear constraints truncate the "velocity" vectors
Best particle
a) TCPSO step
Reflected solution 1
Centroid
Worst solution
c) Complex step iteration 1
Additional poll directions of set H^t
b) GSS step
Reflected solution 1
Reflected solution 2
Worst solution
d) Complex step iteration 2

Fig. 8. Example of a PGS-COM iteration in which the three steps are applied. Frame (a) shows the swarm of particles together with their velocity vectors as well as the two strategies adopted to handle linear and nonlinear constraints. Frame (b) represents the set of solutions polled in the GSS poll step, when there are no ε-active linear constraints. If linear constraints are ε-active, the poll directions are computed so as to conform to the local geometry, as shown in Fig. 7. Frames (c) and (d) show two consecutive reflections of the Complex step.

we address in this paper can be formulated as Problem (1) (see Section 1).

Since relaxable constraints can be easily handled with a penalty approach or a repair method, as done by Zahara and Hu (2008), in this work we focus on unrelaxable constraints. As well known, unrelaxable constraints introduce infinite-type function discontinuities in the sense that, if violated, the objective function value cannot be computed and is set to infinity (or to a very large value). For this reason, optimization problems with unrelaxable constraints are much more challenging than those with relaxable ones. Thus, in Problem (1) we assume that all the constraints (i.e., bound, linear and the nonlinear ones) are unrelaxable and f may be non-differentiable, discontinuous, noisy and subject to hidden constraints.

Within the three PGS-COM steps (i.e., Particle Swarm, GSS and Complex) the nonlinear unrelaxable as well as the hidden constraints are taken into account with the extreme barrier function f_{eb}, defined as

$$f_{eb}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) < +\infty & \text{if the hidden and unrelaxable constraints are satisfied} \\ +\infty & \text{otherwise.} \end{cases} \tag{12}$$

Instead, the unrelaxable bound and linear constraints are accounted for by computing the maximum feasible displacement of the particles in the Particle Swarm step (as further described in Section 4.6), and by adjusting the poll directions to the local geometry defined by the ε-active linear constraints in the GSS step (see next subsection). In the Complex step, as in the original version of Andersson (2001), linear constraints are handled with the extreme barrier function while bound constraints are taken into account by limiting the values of the variables.

4.3. The PGS-COM algorithm

Starting from a set of N_p feasible solutions randomly generated according to the procedure described in Section 4.4, at each iteration the PGS-COM algorithm applies the following three steps:

1. A search step consists in a single update of all the particles (more precisely a single update of the N_p current solutions) according to the "Truncated" CPSO (TCPSO) described in Section 4.6.
2. If the TCPSO search step is not successful (the best objective function value is not improved) for N_{TCPSO} iterations, an iteration of the GSS is carried out around the best solution $\mathbf{y}_g$ found so far (among all the particles) by the swarm for a given step size α^t with the set G^t of core poll directions and the set H^t of additional poll directions. The set G^t contains the generators of the tangent cone relative to the "ε-active constraints", which are computed as described in Section 4.5. To accelerate the progress toward a minimum and avoid premature convergence on non-smooth suboptimal solutions, the set H^t contains the directions of the last successful moves made in the TCPSO and Complex steps (see the pseudo-code), the outward-pointing normals to the ε-active constraints and their sum, which is expected to approximate the direction toward the vertex or edge of the polyhedron defined by the ε-active linear constraints. It is worth pointing out that while the computation of the exact vertex or edge position can be quite straightforward if the set of ε-active linear constraints is consistent, it can be impossible or very expensive computationally when this set is inconsistent or linearly dependent.
3. If the TCPSO search step is not successful for N_{TCPSO} iterations and the GSS step is not successful for N_{GSS} iterations, a few (N_{COM}) reflection iterations of the Complex algorithm of Andersson (2001) are carried out starting from the set of feasible solutions considered in the last GSS step. In case both the TCPSO and GSS steps keep failing while Complex finds an improving solution, the Complex algorithm restarts from its last set S of solutions.

Fig. 8 depicts an iteration of the algorithm in which the three steps are applied. Frame (c) and (d) emphasize how the Complex step, thanks to its reflection scheme, explores promising regions of the solution space which were not polled by the GSS step.

The algorithm stops when the swarm size, i.e., the maximum distance between the best particle and the remaining particles, the step size parameter of the GSS step, and the size of the simplex used by the Complex step, i.e., the maximum distance between the best solution and the solutions used by the Complex search, fall below given threshold values. A maximum number of function evaluations is also considered as stopping criterion.

4.3.1. PGS-COM pseudo-code

4.3.1.1. PGS-COM parameters. Set values of: number of particles N_p, size of the particle neighborhoods N_{pn}, threshold number of consecutive unsuccessful TCPSO and GSS steps, respectively N_{TCPSO} and N_{GSS}, number of reflections N_{COM} executed in each Complex step, initial, minimum and maximum step size parameter for the GSS step, α^0, α_{min}, and α_{max}.

4.3.1.2. Initialization.

- *Particle Swarm*: Generate N_p feasible particles $\mathbf{x}_i{}^0$ with initial velocities $\mathbf{v}_i^0$ by applying the initialization algorithm described

in Section 4.4. Initialize the best solution found so far by each particle $\boldsymbol{y}_i$ equal to $\boldsymbol{x}_i^0$, and the index g of the best particle equal to 1. Initialize the counter Iun_{TCPSO} of consecutive unsuccessful TCPSO iterations.
- *GSS step:* Initialize as empty sets the set of GSS solutions P^0, the set D_{TCPSO} of the successful solution updates (moves) of the last TCPSO step, and the set D_{COM} of promising poll directions determined by the Complex step. Initialize the counter Iun_{GSS} of consecutive unsuccessful GSS iterations.
- *Complex step:* Initialize the set of solutions of the Complex step S^0 as an empty set.
- Initialize the counter t of PGS-COM iterations.

4.3.1.3. PGS-COM iterations. Repeat the following steps until the stopping criterion is met

1. TCPSO step
   - Evaluate the extreme barrier function f_{eb} on the current solutions (particle positions) $\boldsymbol{x}_i^{\mathrm{t}}$
   - For each particle $i = 1, 2, \ldots, N_{\mathrm{p}}$ do:
     - Update the best solution found so far by each particle: If $f_{\mathrm{eb}}(\boldsymbol{x}_i^{\mathrm{t}}) < f_{eb}(\boldsymbol{y}_i)$, then $\boldsymbol{y}_i := \boldsymbol{x}_i^{\mathrm{t}}$
     - Let g be the index of particle with the best solution found so far, and $\boldsymbol{y}_{\mathrm{g}}$ the best solution found so far:
       If $f_{\mathrm{eb}}(\boldsymbol{x}_i^{\mathrm{t}}) < f_{\mathrm{eb}}(\boldsymbol{y}_g)$ /* TCPSO search step “successful”
     - $Iun_{\mathrm{TCPSO}} := 0$;
     - Compute the direction from the previous and the new best solution as $(\boldsymbol{x}_i^{\mathrm{t}} - \boldsymbol{y}_g)/||\boldsymbol{x}_i^{\mathrm{t}} - \boldsymbol{y}_g||$ and store it in D_{TCPSO}
     - Update the step size parameter: $\alpha^{\mathrm{t}+1} := \min\{\alpha_{\max}, \max\{\alpha^{\mathrm{t}}, ||\boldsymbol{x}_i^{\mathrm{t}} - \boldsymbol{y}_g||\}\}$
     - Empty the set S^{t} to be used in the Complex step: $S^{\mathrm{t}} = \Leftrightarrow$
     - Update the best particle index and best solution found so far: $g := i$ and $\boldsymbol{y}_g := \boldsymbol{x}_i^{\mathrm{t}}$.
       else /* TCPSO search step “unsuccessful”
     - $Iun_{\mathrm{TCPSO}} := Iun_{\mathrm{TCPSO}} + 1$;
     - $D_{\mathrm{TCPSO}} := \varnothing$.
2. *GSS step*: executed if TCPSO fails for N_{TCPSO} iterations (if $Iun_{\mathrm{TCPSO}} \geq N_{\mathrm{TCPSO}}$) and $\alpha^{\mathrm{t}} \geq \alpha_{\min}$
   - Execute the GSS poll step around $\boldsymbol{y}_{\mathrm{g}}$ (the best solution found so far) with step size α^{t} and the sets G^{t} of core directions, and H^{t} (comprising D_{TCPSO} and D_{COM}) of additional directions, as described in Section 4.5. Let $\boldsymbol{x}_{\mathrm{GSS}}$ the best solution among the set P^{t} of polled solutions.
     If $f_{\mathrm{eb}}(\boldsymbol{x}_{\mathrm{GSS}}) < f_{\mathrm{eb}}(\boldsymbol{y}_{\mathrm{g}})$ /* GSS step “successful”
     - Update the best solution found so far: $\boldsymbol{y}_{\mathrm{g}} := \boldsymbol{x}_{\mathrm{GSS}}$.
     - $Iun_{\mathrm{GSS}} := 0$
     - $S^{\mathrm{t}} := \varnothing$
     - Increase the step size parameter α: $\alpha^{\mathrm{t}+1} := \min\{2\alpha^{\mathrm{t}}, \alpha_{\max}\}$
       else /* GSS step “unsuccessful”
     - $Iun_{\mathrm{GSS}} := Iun_{\mathrm{GSS}} + 1$.
     - Decrease the step size α: $\alpha^{\mathrm{t}+1} := \max\{\alpha^{\mathrm{t}}/2, \alpha_{\min}\}$
3. *Complex step*: executed if TCPSO fails for N_{TCPSO} iterations (if $Iun_{\mathrm{TCPSO}} \geq N_{\mathrm{TCPSO}}$) and either GSS fails for N_{GSS} iterations ($Iun_{\mathrm{GSS}} \geq N_{\mathrm{GSS}}$) or $\alpha^{\mathrm{t}} \leq \alpha_{\min}$
   - Initialize the set S^{t} of feasible solutions:
     If $S^{\mathrm{t}} = \varnothing$ /* TCPSO or GSS was successful at the previous iteration
     - $S^{\mathrm{t}} := P^{\mathrm{t}} \cup \{\boldsymbol{y}_{\mathrm{g}}\}$
     - If the number N_{S} of solutions in S^{t} is smaller than $2n$, add to S^{t} the best $2n - N_{\mathrm{S}}$ solutions found so far by the TCPSO.
       else /* both TCPSO and GSS were unsuccessful
     - Set $S^{\mathrm{t}} := Z^{\mathrm{t}-1}$, where $Z^{\mathrm{t}-1}$ is the set of solutions returned by the previous Complex step.
   - Starting from S^{t} apply N_{COM} iterations (reflections) of the Complex algorithm of Andersson (2001) to Problem (12) with unlimited contractions (Eq. (10)). Let Z^{t} denote the set of solutions returned by the Complex step, $\boldsymbol{x}_{\mathrm{r}}$ the solution generated by the Complex algorithm in the last reflection, $\boldsymbol{x}_{\mathrm{COM}}$ and $\boldsymbol{x}_{\mathrm{w}}$ the best and worst solutions in Z^{t}.
   - Add to D_{COM} the normalized direction $\boldsymbol{x}_{\mathrm{COM}} - \boldsymbol{x}_{\mathrm{w}}$ (as an estimate of the steepest descent direction), $\boldsymbol{x}_{\mathrm{COM}} - \boldsymbol{x}_{\mathrm{r}}$, and, if $f_{\mathrm{eb}}(\boldsymbol{x}_{\mathrm{COM}}) < f_{\mathrm{eb}}(\boldsymbol{y}_{\mathrm{g}})$, also $\boldsymbol{x}_{\mathrm{COM}} - \boldsymbol{y}_{\mathrm{g}}$.
   - If $f_{\mathrm{eb}}(\boldsymbol{x}_{\mathrm{COM}}) < f_{\mathrm{eb}}(\boldsymbol{y}_{\mathrm{g}})$ then update the best solution and the step size parameter:
     - Update the step size parameter: $\alpha^{\mathrm{t}+1} := \min\{||\boldsymbol{x}_{\mathrm{COM}} - \boldsymbol{y}_{\mathrm{g}}||, \max\{\alpha^{\mathrm{t}}, \alpha_{\min}\}\}$
     - Update the best solution found so far: $\boldsymbol{y}_{\mathrm{g}} := \boldsymbol{x}_{\mathrm{COM}}$
4. Update particle velocities and current solutions:
   - For each particle $i = 1, 2, \ldots, N_{\mathrm{p}}$ generate the new velocity $v_i^{\mathrm{t}+1}$, and update its current solution (its position) $\boldsymbol{x}_i^{\mathrm{t}+1}$ as described in Section 4.6.
   - Set $t := t + 1$ and go back to Step 1.

4.3.2. PGS-COM features

Note that, when the TCPSO step is successful at each iteration, the GSS and Complex steps are skipped and the PGS-COM method behaves similarly to the CPSO of Hu and Eberhart (2002) (except for the fact that the swarm update formula is modified according to the calculation described in Section 4.6). In case the TCPSO is unsuccessful and the GSS is successful, the algorithm becomes a GSS method which ensures good performance on linearly constrained smooth problems.

It is worth noting that the value of the step size parameter α of the GSS step depends on the outcome of both the TCPSO and Complex steps. More precisely, in the TCPSO search step α can be directly increased because it is yoked to the distance between the previous and the new best solution found by the whole swarm, $\alpha^{\mathrm{t}+1} = \min(\alpha_{\max}, \max(\alpha^{\mathrm{t}}, ||\boldsymbol{x}_i^{\mathrm{t}} - \boldsymbol{y}_g||_2))$. When a new best solution is found by the particles, this update formula allows to quickly switch from a fine local search step around the previous best solution to an exploratory poll step around the new one. In the Complex step, when a better solution $\boldsymbol{x}_{\mathrm{COM}}$ is found very close to the initial best solution $\boldsymbol{y}_{\mathrm{g}}$, α is reduced according to $\alpha^{\mathrm{t}+1} = \min(||\boldsymbol{x}_{\mathrm{COM}} - \boldsymbol{y}_{\mathrm{g}}||_2, \alpha^{\mathrm{t}})$ so as to intensify the search in the neighborhood. Whenever α is smaller than the threshold value $\alpha_{\min}$, the GSS step is skipped and the TCPSO step is directly followed by the Complex step. This is to avoid GSS search steps with very small step size parameters which often lead to a negligible progress toward the optimum.

Concerning the GSS scheme, we have enriched the set of poll directions by including not only the outward pointing normals to the ε-active linear constraints (as done by Lewis et al., 2007) but also an estimate of the direction toward the edge/corner of the ε-active linear constraints, the direction of the last successful TCPSO best solution update, and the set of descent directions identified by the Complex step as detailed in the pseudo-code. The purpose of these additional search directions is to increase the probability of finding a descent direction in problems with non-smooth objective function and/or unrelaxable nonlinear constraints. The beneficial impact of this choice for non-smooth and nonlinearly constrained problems is clearly shown by the computational results reported in Section 7.

4.4. Initialization of the swarm

The set of initial feasible solutions (swarm of particles) is generated with three different schemes depending on the type of constraints:

- Uniform random initialization: If only bound constraints are present, the initial particles are randomly initialized within the bounds (i.e., each component of the vectors is a random variable with uniform distribution between its upper and lower bound).

- Maximum volume ellipsoid method: If bound and linear constraints are present, the maximum volume ellipsoid method of Vaz and Vicente (2009) is used. The rationale is to try to distribute as uniformly as possible the solutions over the feasible polyhedron defined by the linear constraints. After computing the maximum volume ellipsoid inscribed into the feasible polyhedron by using the algorithm of Zhang and Gao (2003), the ellipsoid center and scaling matrix are used to randomly generate initial solutions (particle positions) that are feasible with respect to linear constraints.
- Iterative replacement procedure: If also nonlinear unrelaxable and hidden constraints are present, an iterative replacement algorithm is implemented. A first set of solutions is randomly generated by either the uniform random initialization procedure or the maximum volume ellipsoid method, depending on the presence of linear constraints. Clearly these solutions may be infeasible with respect to the nonlinear unrelaxable and hidden constraints. Thus, infeasible solutions need to be iteratively replaced. To reduce the number of repetitions when the feasible region defined by the nonlinear unrelaxable constraints is narrow with respect to the feasible polyhedron, we progressively ensure feasibility as follows. Whenever an infeasible solution is found, it is replaced by the convex combination of a randomly selected (previously found) solution $\boldsymbol{x}_{\text{feas}}$ and a randomly generated solution $\boldsymbol{x}_{\text{gen}}$ (that may be infeasible) according to the formula.

$$\boldsymbol{x}_{\text{i}}^0 = (1-\delta)\boldsymbol{x}_{\text{gen}} + \delta\boldsymbol{x}_{\text{feas}}, \tag{13}$$

where δ is an increasing function of the number of iterations and that $\in [0, 1]$. The idea is to consider solutions which are progressively closer to feasibility. In particular we take

$$\delta = \frac{Ini^2}{N_{\text{INI}}^2}, \tag{14}$$

where Ini is the iteration counter and N_{INI} the maximum number of iterations of the initialization procedure. Thanks to the properties of Eq. (14), the first solutions are practically randomly generated (since δ is close to zero), while the influence of $\boldsymbol{x}_{\text{feas}}$ becomes considerable ($\delta > 0.5$) for $Ini > 0.7\ N_{\text{INI}}$.

It is important to ensure the feasibility of the starting solutions because, as pointed out by Hu and Eberhart (2002), the performance of particle swarm algorithms may be penalized when starting from several infeasible solutions.

As far as the initial velocities are concerned, we consider randomly initialized vectors with components limited by the bound constraints. Our computational results show that this choice ensures good performance.

4.5. *Computation of the poll directions*

The set of core poll directions G^{t} is computed according to the procedure described by Lewis et al. (2007). More precisely, G^{t} contains the positive generators for the cone $T(\boldsymbol{y}_{\text{g}}, \varepsilon^{\text{t}})$ defined in Eq. (6), with $\varepsilon^{\text{t}} = \alpha^{\text{t}}$, and such positive generators are computed in the following way.

- If no general linear constraint is ε-active in $\boldsymbol{y}_{\text{g}}$ (bound constraints may be ε-active), $G^{\text{t}} := D$, the set of coordinate directions (i.e., the canonical basis vectors and their negative counterparts).
- If only general linear constraints are ε-active in $\boldsymbol{y}_{\text{g}}$, the directions in G^{t} are computed as follows. Let $\boldsymbol{Y}$ be the matrix whose columns are the rows of $\boldsymbol{A}$ corresponding to the ε-active linear constraints. Compute the vectors spanning the null space of $\boldsymbol{Y}^{\text{T}}$ as the columns of the orthogonal matrix $\boldsymbol{Q}$ of the QR decomposition of $\boldsymbol{Y}^{\text{T}}$, and the right inverse $\boldsymbol{J}$ of $\boldsymbol{Y}^{\text{T}}$. As proved in Lewis et al. (2007), the columns of $\boldsymbol{Q}$ and $\boldsymbol{J}$ are positive generators of $T(\boldsymbol{y}_{\text{g}}, \varepsilon^{\text{t}})$ and thus form a proper set of poll directions. Since the matrix columns of $\boldsymbol{Q}$ are orthogonal, this is an advantage for the poll step.
- If general linear constraints as well as bounds are ε-active in $\boldsymbol{y}_{\text{g}}$, the positive generators corresponding to the linear constraints are computed according to the above mentioned scheme while those corresponding to the bound constraints are handled separately. Indeed, it is not necessary to compute the right inverse of $\boldsymbol{Y}^{\text{T}}$, and it suffices to add in G^{t} the corresponding positive or negative unit vectors $\boldsymbol{e}_j$ of the canonical basis (where $\boldsymbol{e}_j$ is defined such that $\boldsymbol{e}_{j,k} = 0$ for any component $k \neq j$). More precisely, for each jth variable, if it is verified that $y_{g,j} - l_j \le \varepsilon^{\text{t}}$, add $\boldsymbol{e}_j$ in G^{t}, vice versa, if $u_j - y_{g,j} \le \varepsilon^{\text{t}}$, add $-\boldsymbol{e}_j$.

In the degenerate case (i.e., when the vectors $\boldsymbol{a}_{\text{k}}$ in $I(\boldsymbol{y}_{\text{g}}, \varepsilon^{\text{t}})$ are linear dependent or, equivalently, the ε-normal cone $N(\boldsymbol{y}_{\text{g}}, \varepsilon^{\text{t}})$ is a cone with a degenerate vertex at the origin), the above described procedure cannot be applied because $\boldsymbol{Y}$ is rank deficient. In Lewis et al. (2007), the authors use advanced computational geometry algorithms capable of finding the vertices and extreme rays of the polyhedron of the linear constraints, and then determining the positive generators also in the degenerate case. Given the high computational complexity of such procedure and the fact that PGS-COM can compensate the possible lack of poll directions with those of the Complex search step, we have not followed the rigorous approach of Lewis et al. (2007) but the simpler approach of Vaz and Vicente (2009). When $\boldsymbol{Y}$ is rank deficient, ε is iteratively decreased so as to obtain a set of linearly independent ε-active constraints (i.e., full rank $\boldsymbol{Y}$) and, if no small enough ε is found, the set of coordinate directions D is used. Although this approach does not satisfy the basic requirements needed to guarantee global convergence for GSS algorithms, it proved to be very efficient for the PSwarm algorithm of Vaz and Vicente (2009) also on highly degenerate test cases.

Regarding the set H^{t} of additional poll directions, if no general linear constraint is ε-active, H^{t} is set equal to $D_{\text{TCPSO}} \cup D_{\text{COM}}$, and then it contains the direction of the last successful solution update of the TCPSO step and the (probably descent) directions determined by the solutions considered in the last Complex step. If some general linear constraint is ε-active, we consider in H^{t} not only $D_{\text{TCPSO}} \cup D_{\text{COM}}$ but also the set U^{t} of outward pointing normals to the ε-active linear constraints and their sum $\boldsymbol{d}_{\text{c}}$,

$$\boldsymbol{d}_{\text{c}} = \frac{\sum_{\boldsymbol{d} \in W^{\text{t}}} \boldsymbol{d}}{\|\sum_{\boldsymbol{d} \in W^{\text{t}}} \boldsymbol{d}\|_2}, \tag{15}$$

which estimates the direction pointing to the edge/corner defined by the ε-active linear constraints. Hence, we take $H^{\text{t}} := D_{\text{TCPSO}} \cup D_{\text{COM}} \cup U^{\text{t}} \cup \{\boldsymbol{d}_{\text{c}}\}$.

In summary, at each GSS step, the extreme barrier function f_{eb} is evaluated at the feasible solutions in P^{t}, where

$$\begin{aligned} P^{\text{t}} = \{&\boldsymbol{x} \in \Re^{\text{n}} : x = \boldsymbol{y}_{\text{g}} + \alpha^{\text{t}}\boldsymbol{d}, \quad \boldsymbol{d} \in G^{\text{t}} \cup H^{\text{t}}, \\ &\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b},\ \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u},\ \boldsymbol{g}(\boldsymbol{x}) \le \boldsymbol{0}\}. \end{aligned} \tag{16}$$

4.6. *Truncated Constrained Particle Swarm (TCPSO)*

The particle positions (solutions) and velocities (i.e., more rigorously defined as displacements of the particles) are updated as in the *l-best* version of Particle Swarm by Kennedy and Eberhart (1995), and the velocities are limited to the maximum extent allowed by the linear constraints as in Vaz and Vicente (2009).

Given the large impact of the size N_{np} of the particle neighborhood on the swarm behavior, in PGS-COM we consider N_{np} as a set-up parameter which can range from 1 to the complete swarm size.

4.6.1. *Truncated swarm update pseudo-code*

For each *i*th particle of the swarm execute the following steps:

1. Consider the cyclic set of neighboring particle indices in the particle array, $C_i = \{i - N_{\text{np}}, i - N_{\text{np}} + 1, \ldots, i, \ldots, i+1, i+N_{\text{np}}\}$, and determine the index q of the particle in C_i with the best solution found so far.
2. Compute the velocity vector $\boldsymbol{v}_i^{t+1}$ according to Eq. (3).
3. Compute the maximum allowed displacement which does not violate bound and linear constraints according to the standard truncation scheme.
   - Update the velocity according to

     $$v_{i,j}^{t+1} = v_{i,j}^{t+1} \cdot \chi_j \quad \forall j = 1, 2, \ldots, n \tag{17}$$

     where the reduction coefficient χ_j is computed according to

     $$\chi_j = \begin{cases} \min\left(1, \dfrac{l_j - x_{i,j}^t}{v_{i,j}^{t+1}}\right) & \text{if } v_{i,j}^{t+1} < 0, \\ \min\left(1, \dfrac{u_j - x_{i,j}^t}{v_{i,j}^{t+1}}\right) & \text{if } v_{i,j}^{t+1} > 0, \\ 1 \quad \text{if } v_{i,j}^{t+1} = 0. \end{cases} \tag{18}$$

   - Determine the set L of indices of linear constraints which could be violated by a displacement along the velocity vector $\boldsymbol{v}_i^{t+1}$:

     $$L = \{k \in \{1, 2, \ldots, m\} : \boldsymbol{a}_k^T \boldsymbol{v}_i^{t+1} > 0\}. \tag{19}$$

   - Compute the maximum step length γ along the velocity vector $\boldsymbol{v}_i^{t+1}$

     $$\gamma = \min_{k \in L}\left(1, \frac{b_k - \boldsymbol{a}_k^T \boldsymbol{x}_i^t}{\mathbf{a}_k^T \boldsymbol{v}_i^{t+1}}\right), \tag{20}$$

     and reduce the particle velocity vector guaranteeing feasibility with respect to the bound and linear constraints,

     $$\boldsymbol{v}_i^{t+1} = \gamma \boldsymbol{v}_i^{t+1}. \tag{21}$$

4. Update the current solution of the particle (particle position):

   $$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{22}$$

4.7. *Differences with respect to the PSwarm method*

Although both PSwarm of Vaz and Vicente (2009) and PGS-COM are based on the idea of following unsuccessful particle swarm iterations with a poll step, there are four main differences between the two methods that we now summarize.

PSwarm is based on a two-level structure including a particle swarm search and a poll step, while PGS-COM is based on a three level structure with a particle swarm search, a poll step, and a sequence of Complex iterations. The purpose of the Complex iterations is to find a descent direction even on problems with non-smooth functions or unrelaxable constraints, for which the poll step may lead to a premature convergence at suboptimal solutions.

The PSwarm poll step considers only the set of core search directions (i.e., the set of positive generators of the ε-tangent cone) and optionally the set of outward pointing normals of the ε-active constraints. In the GSS step of PGS-COM, not only the two above set of directions are always considered, but the set is further enriched by adding the descent directions determined by the TCPSO and Complex steps.

In PSwarm the step size parameter of the poll step is doubled or unchanged if the poll step is successful and halved if unsuccessful. In PGS-COM the step size parameter is also yoked to the distance between the best solution found at the previous PGS-COM iteration and the new best solution found at the end of the current iteration (which can be either a TCPSO step or a Complex step).

Finally, while in our algorithm the particles are updated according to the truncated *l-best* version of Kennedy and Eberhart (1995), in PSwarm the particles are updated according to a truncated *g-best* version.

As shown by the computational experiments reported in Sections 7 and 8, the above mentioned differences considerably change the behavior of PGS-COM with respect to that of PSwarm and make it much more effective than PSwarm on non-smooth constrained problems.

5. Reference test problems

In order to evaluate the performance of PGS-COM with respect to that of the existing direct-search methods, we need to choose a set of challenging test problems. Compared to the analysis of Rios and Sahinidis (2013) which tests derivative-free methods on smooth and non-differentiable test problems subject only to bound constraints, we focus the attention on problems which share some important features with real process engineering problems, such as linear inequality constraints, unrelaxable nonlinear (or hidden) inequality constraints, step type discontinuities and numerical noise in the objective function. We consider two main groups of reference problems without (Group A) and with (Group B) numerical noise. Both groups include non-smooth (non-differentiable and/or discontinuous) problems subject to unrelaxable constraints (which, if violated, do not allow the evaluation of the objective function) with a number of variables ranging from 4 to 20. The focus of our computational experiments is on instances with up to 20 variables because, as well known and also shown in Rangaiah (1982), the number of function evaluations needed to find an optimum with direct-search methods tends to increase much faster with respect to the number of variables compared to gradient-based methods. However, the reader who is interested in larger scale problems can find in Section 7.5 numerical results on three challenging test problems with 30 variables.

The main features of the reference test problems are summarized in the following subsection, while their analytical formulations and Matlab implementations are included in the supplementary on-line material.

5.1. *Constrained non-smooth test problems*

Based on an extensive search of the literature, we have found challenging non-differentiable problems in Luksan and Vicek (2000), Audet and Dennis (2006), and Hu et al. (2003). The test collection of Luksan and Vicek (2000) is particularly relevant because it contains 25 unconstrained "minmax" (thus, non-differentiable) problems, 25 unconstrained general non-differentiable problems, and 25 linearly constrained non-differentiable problems with 2 up to 50 variables. We have selected the following four non-convex problems: "Colville 1", "Pentagon", "Wong 2", and "Shell Dual". In all of them, we have introduced upper and lower bounds on each variable, as specified in the problem formulation included in the supplementary on-line material.

As far as discontinuous test problems are concerned, since very few of them with more than 3 variables are available in the literature (those in Vicente & Custodio, 2012; Buzzi-Ferraris & Manenti, 2010; Hu et al., 2003; Michalewicz & Nazhiyath, 1995, have up to three variables), we have derived a number of analytical discontinuous problems with more than four variables by modifying the challenging constrained test problems reported by Koziel and

Michalewicz (1999). The idea is to add a discontinuous penalty term $h_k(\boldsymbol{x})$ to the original smooth objective function for each nonlinear constraint $g_k(\boldsymbol{x})$ which is violated. In particular, we obtain a modified objective function

$$f_m(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{k=1}^{p} h_k(\boldsymbol{x}), \tag{23}$$

with

$$h_k(\boldsymbol{x}) = \begin{cases} 0 & \text{if } g_k(\boldsymbol{x}) \le 0 \\ c > 0 & \text{if } g_k(\boldsymbol{x}) > 0 \end{cases} \quad \forall k \in \{1, 2, \ldots, p\}. \tag{24}$$

The modified objective function f_m is optimized over the feasible region defined by the bounds and the linear constraints. According to this construction, f_m has step type discontinuities (jumps) wherever one of the nonlinear constraint function $g_k(\boldsymbol{x})$ is equal to zero. To leave unchanged the optimal solution $\boldsymbol{x}^*$ of the original smooth problem, we choose c such that $c > f(\boldsymbol{x}^*) - f(\boldsymbol{x}_u)$, where $\boldsymbol{x}_u$ denotes the global minimum of the original smooth problem without nonlinear constraints. Since $f_m(\boldsymbol{x}) = f(\boldsymbol{x}) + c \ge f(\boldsymbol{x}^*)$ for any $\boldsymbol{x}$ and $f_m(\boldsymbol{x}^*) = f(\boldsymbol{x}^*)$ because no penalty is applied in $\boldsymbol{x}^*$, $\boldsymbol{x}^*$ is the global optimum also for the modified discontinuous problem. We have generated such step type discontinuities for the "G2", "G4", "G7", "G9", and "G10" test problems by Koziel and Michalewicz (1999), and the "Welded Beam" design problem by Hu et al. (2003). In addition, we have included the challenging constrained discontinuous "Discrete Pressure Vessel" design problem of Hu et al. (2003).

In summary, Group A comprises the following problems:

A.1 Discontinuous problem derived from the "Welded Beam" design problem of Hu et al. (2003) with 4 variables, only bound constraints, and global optimum value 2.218151.
A.2 The "Discrete Pressure Vessel" design problem of Hu et al. (2003), with 4 variables, 2 linear constraints, 2 nonlinear unrelaxable constraints and discontinuous objective function, and global optimum value 8796.86224.
A.3 The non-differentiable "Colville 1" problem of Luksan and Vicek (2000), with 5 variables, bound constraints, and global optimum value 32.348679.
A.4 The constrained discontinuous problem derived from test problem "G4" of Koziel and Michalewicz (1999), with 5 variables, 4 nonlinear unrelaxable constraints, and global optimum value 30,665.5
A.5 The linearly constrained non-differentiable "Pentagon" test problem of Luksan and Vicek (2000), with 6 variables, 15 unrelaxable linear inequality constraints, and global optimum value 1.8596187.
A.6 The constrained discontinuous problem derived from test problem "G9" of Koziel and Michalewicz (1999), with 7 variables, 2 nonlinear unrelaxable constraints, and global optimum value 680.6300573.
A.7 The constrained discontinuous problem derived from test problem "G10" of Koziel and Michalewicz (1999), with 8 variables, one linear and one nonlinear unrelaxable constraints, and global optimum value 7049.3307
A.8 The constrained discontinuous problem derived from the test problem "G7" of Koziel and Michalewicz (1999), with 10 variables, 2 linear unrelaxable constraints, global optimum value 24.306209 (see the 2-D plot in Fig. 9)
A.9 The linearly constrained non-differentiable "Wong 2" test problem of Luksan and Vicek (2000), with 10 variables, 3 linear unrelaxable inequality constraints, and global optimum value 24.306209.
A.10 The non-differentiable "Shell Dual" test problem in Luksan and Vicek (2000), with 15 variables, and global optimum value 32.348679.
A.11 The constrained discontinuous and non-differentiable problem derived from the test problem "G2"of in Koziel and Michalewicz (1999), with 20 variables, 1 linear and 1 nonlinear unrelaxable constraint, and best known solution value 0.8036.

Note that the linear constraints are unrelaxable and explicit (thus, they can be specified in the algorithm input), while the nonlinear constraints are unrelaxable and hidden (as they mimic evaluation failures of the black-box function and/or feasibility check procedures).

5.2. *Non-smooth test problems with noisy objective function*

Group B contains non-smooth optimization problems aimed at mimicking simulation-based numerical noise. For instance, if the objective function is a parameter returned by an iterative process solving a differential equation to a specified accuracy, the objective function value is generally affected by numerical noise. According to Moré and Wild (2009), numerical noise arising in this type of simulation is better modeled by a function with both high-frequency and low-frequency oscillations. In particular the noise generator function $\delta_n(\boldsymbol{x})$ is taken as

$$\delta_n(\boldsymbol{x}) = [1 + \varepsilon_n \phi_n(\boldsymbol{x})], \tag{25}$$

where

$$\phi_n(\boldsymbol{x}) = \varphi_n(\boldsymbol{x})[4\varphi_n(\boldsymbol{x})^2 - 3], \tag{26}$$

$$\varphi_n(\boldsymbol{x}) = 0.9\sin(100\|\boldsymbol{x}\|_1)\cos(100\|\boldsymbol{x}\|_\infty) + 0.1\cos(\|\boldsymbol{x}\|_2), \tag{27}$$

and ε_n denotes the noise level (e.g., 10^{-3}). Thanks to the function $\delta_n(\boldsymbol{x})$, it is possible to generate noisy functions $f_n(\boldsymbol{x})$ from any available test problem $f(\boldsymbol{x})$ by taking $f_n(\boldsymbol{x}) = f(\boldsymbol{x})\delta_n(\boldsymbol{x})$. Clearly, depending on the value of ε_n, the global minimum solution and optimal value of $f_n(\boldsymbol{x})$ may be slightly different from the ones of $f(\boldsymbol{x})$.

The noisy problems of Group B (referred to as B1, B2, . . ., B11) have been generated from those of Group A by applying the above described approach with $\varepsilon_n = 10^{-3}$. The plot of test problem B4 is reported in Fig. 10.

6. PGS-COM variants and benchmark algorithms

The PGS-COM algorithm presented in Section 4 has been implemented in Matlab®. To assess PGS-COM performance, we compare its results with those provided by several benchmark derivative-free methods. Unlike in Rios and Sahinidis (2013) where both model-based and direct-search derivative-free methods are considered, we focus on direct-search methods because we are interested in non-smooth black-box test problems with objective function discontinuities and unrelaxable (or hidden) constraints. Since it is unrealistic to consider all the existing direct-search algorithms, we have restricted the attention to eleven well known and widely used algorithms which are suited for constrained non-smooth black-box problems, and which are coded in Matlab®.

Benchmark methods:

1. PSwarm by Vaz and Vicente (2009),
2. Constrained Particle Swarm Optimizer (CPSO) of Hu and Eberhart (2002),
3. Complex method of Andersson (2001),
4. Generating Set Search of Lewis et al. (2007),
5. Lower Triangular Mesh Adaptive Direct Search (LTMADS) algorithm of Audet and Dennis (2006),

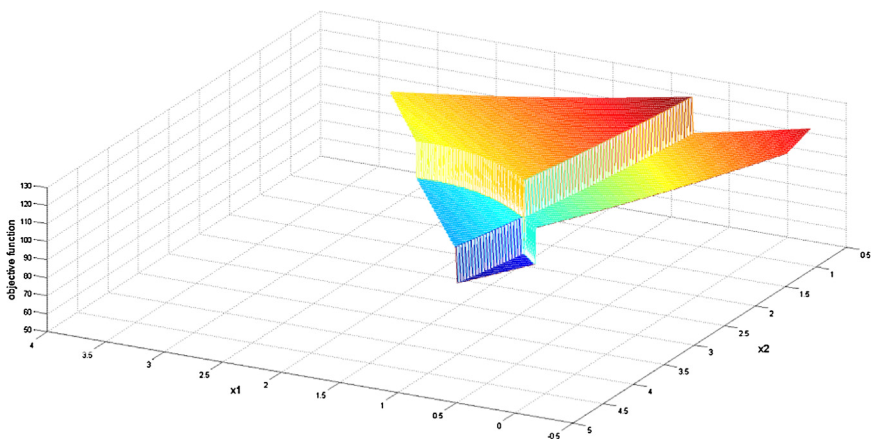

Fig. 9. Plot of the discontinuous objective function of test problem A7 with respect to variables x_1 and x_2.

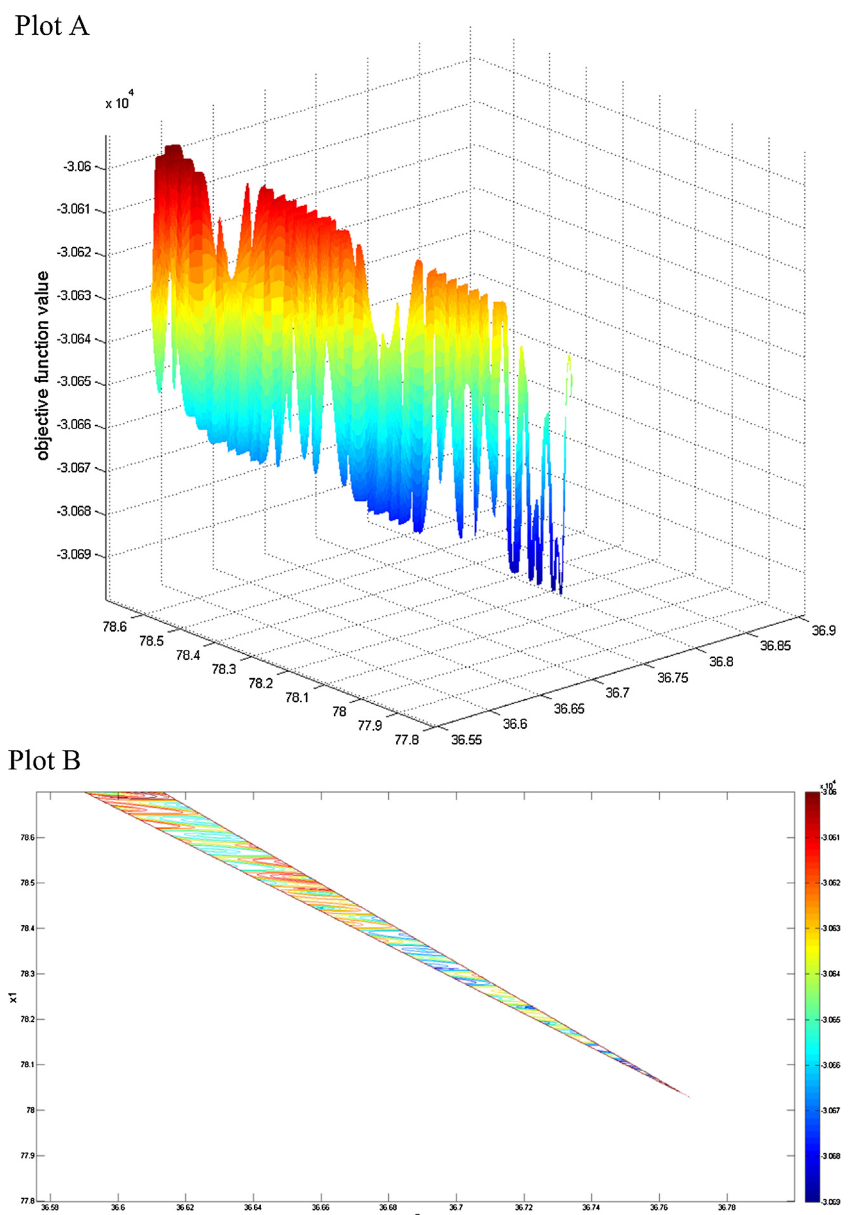

Fig. 10. 2-D plot (Plot A) and contour plot (Plot B) of the objective function of test problem B4 with respect to variables x_1 and x_5. Objective function values are defined only on a narrow region (see Plot B) which satisfies the unrelaxable constraints. Moreover the numerical noise generates multiple local minima (see Plot A).

6. Orthogonal directions Mesh Adaptive Direct Search (OrthoMADS) of Abramson et al. (2009),
7. Covariance Matrix Adaptation Evolution Strategy (CMAES) of Hansen et al. (2003),
8. Controlled Random Search algorithm of Kaelo and Ali (2006),
9. the Matlab Scatter Search code, SSm, of Egea et al. (2007),
10. the hybrid enhanced Scatter Search code, eSS, of Egea et al. (2010),
11. the Hybrid Variable Neighborhood Search-MADS algorithm, VNS-MADS, of Audet, Bechard, and Chaouki (2008a) and Audet, Bechard, and Le Digabel (2008b).

Only CPSO was re-implemented in Matlab on the basis of the related publication, because such a code is available for all the other methods. The Matlab implementation of PSwarm can be downloaded from the webpage http://www.norg.uminho.pt/aivaz/pswarm/. The Complex algorithm is available at the Matlab file exchange webpage http://www.mathworks.com/matlabcentral/fileexchange/25428-Complex-method-of-optimization. The GSS algorithm is available among the pattern search options of the "Matlab Optimization Toolbox" (see http://www.mathworks.it/products/optimization/ for further details). LTMADS, OrthoMADS as well as VNS-MADS are available among the polling options of NOMAD, the C++ code by Le Digabel (2011). NOMAD can be downloaded from http://www.gerad.ca/nomad/Project/Home.html and it can be executed in Matlab through the OPTI toolbox, a free Matlab toolbox developed by Currie and Wilson (2012) and available at http://www.i2c2.aut.ac.nz/Wiki/OPTI/index.php/Main/HomePage. The Matlab implementation of CMAES is available at https://www.lri.fr/~hansen/cmaes_inmatlab.html#matlab. The CRS algorithm of Kaelo and Ali (2006) is included in the free NLopt Nonlinear-Optimization Package by Johnson (2013), and it can be executed in Matlab through the OPTI toolbox. Both SSm and eSS are contained in the SSmGO Matlab toolbox made available by Egea et al. (2007, 2010) at http://www.iim.csic.es/~gingproc/ssmGO.html. Recall that GSS, LTMADS and OrthoMADS are local direct-search methods which tend to quickly converge to the local minima close to the starting point. Here these methods have been extended to global optimization with a multi-start strategy.

For the benchmark algorithms we have adopted (when available) the set-up parameter values suggested by the authors for non-smooth problems. In particular, for PSwarm, CPSO, Complex and GSS we have adopted the same parameter values considered respectively by Vaz and Vicente (2009), Hu and Eberhart (2002), Andersson (2001), and Lewis et al. (2007). For LTMADS and OrthoMADS, we have taken the default values of NOMAD regarding the mesh updating formula and the complete polling strategy with $2n$ polling directions and random seeds (see Abramson et al., 2009). For CMAES and CRS, we have considered the default options. For SSm, we have reproduced the same set-up suggested by the authors for non-smooth functions: we have deactivated the improvement method and used the solver NOMADm (a Matlab implementation of the MADS algorithm developed by Abramson, 2007) for a final refinement phase, as suggested by Egea et al. (2007). For eSS, we have adopted the same set-up as in Egea et al. (2010). Regarding VNS-MADS, we have used the same options of "Algorithm E" tested by Audet, Bechard, and Chaouki (2008a) and Audet, Bechard, and Le Digabel (2008b).

As far as the parameter values of PGS-COM are concerned, for the TCPSO step we have adopted the values of the CPSO parameters given in Kennedy and Eberhart (1995) (except for the number of particles N_p, and the size of the particle neighborhood N_{np}), for the GSS step those given in Lewis et al. (2007), and for the Complex step those given in Andersson (2001). To select the other parameter values of PGS-COM except N_p, N_{np} and the number of reflections in the Complex step N_{COM}, we have carried out a set of preliminary computational experiments on a few constrained non-smooth problems with up to 20 variables. This led to the following combination of parameter values:

- number of consecutive unsuccessful TCPSO iterations before performing the GSS step $N_{TCPSO} = 1$
- number of consecutive unsuccessful GSS iterations before performing the Complex step $N_{GSS} = 3$
- stopping tolerances for the TCPSO and Complex searches equal to 10^{-10}
- initial step size parameter for the GSS step $\alpha^0 = 0.1$
- maximum step size parameter for the GSS step $\alpha_{max} = 0.25$
- minimum step size parameter for the GSS step $\alpha_{min} = 10^{-10}$
- linear scaling of all the optimization variables in the range [0, 1].

On the basis of more extensive computational tests, we have selected the following values for the remaining parameters:

- number of particles $N_p = 30$,
- size of the particle neighborhood $N_{np} = 5$,
- number of reflections executed in the Complex step $N_{COM} = 2$.

Since N_{COM} affects the quality of the solutions returned by PGS-COM, we found that $N_{COM} = 2$ achieves a good trade-off between the solution quality and the computational load of the Complex step (see Section 7.4 for details).

In our experiments, we kept the PGS-COM parameter values constant for all the test problems, and thus did not over-tune them with respect to each specific problem. Although a systematic parameter tuning may clearly lead to improved performance on specific instances, in Sections 7 and 8 we shall see that PGS-COM with these parameter values turns out to be very competitive compared to the benchmark methods.

For a fair comparison, the same starting solutions (either all or a subset of them, depending on the algorithm requirements), generated with the initialization algorithm described in Section 4.4, are used for PGS-COM and the eleven benchmark methods.

As stopping criteria we have considered the maximum number of function evaluations for PSwarm, CPSO, Complex, CMAES, CRC, SSm, eSS, VNS-MADS and PGS-COM, and also the step size parameter for GSS, LTMADS and OrthoMADS. To obtain accurate solution values, we have selected a small minimum step size parameter of 10^{-13}. Since in most runs GSS, LTMADS and OrthoMADS converge to a solution within a small fraction of the allowed function evaluations, the algorithms are re-restarted from different solutions randomly chosen among those generated by the initialization algorithm until the maximum number of function evaluations is reached. At each run, LTMADS and OrthoMADS use a different seed for the pseudo-random generator.

To account for the stochastic nature of the algorithms, each algorithm is executed 20 times on each test instance with different randomly generated solutions (generated with the algorithm described in Section 4.4) for a maximum number of 500, 1000, 2500, 5000, 7500, 10,000, 20,000, 40,000, and 80,000 function evaluations. If not indicated differently, all the computational tests have been carried out on a 32 bit personal computer equipped with a four-core 2.67 GHz Intel i7 processor with 4 GB of random access memory.

7. Computational results on test problems

Computational results for the 11 test problems of Groups A and B are reported in Tables 1–3 and Figs. 11 and 12. Tables 1 and 2 show the average, best and worst values of the solutions returned by the

Table 1
Best, average and worst objective function values of the solutions returned over 20 runs by the twelve tested algorithms within 10,000 objective function evaluations for the test problems of Group A. The last row indicates the fraction of problems which are successfully solved (see the definition of "data profile" in Section 7.3) within a tolerance value of 10^{-4}. "Inf" indicates that, for such test problem with unrelaxable linear or nonlinear constraints, CMAES fails to find a feasible solution (since it is not possible to pass as input the starting set of solutions, it tries and often fails to find solutions which satisfies all the unrelaxable constraints).

Test problem	Run	PSwarm	CPSO	Complex	GSS	LTMADS	OrthoMADS	CMAES	CRS	SSm	eSS	VNS-MADS	PGS-COM
A1	Worst	2.95720527	2.48222367	2.72014378	3.60051196	3.62818493	4.936101377	Inf	2.26521135	2.24748228	3.08123632	4.37966967	2.25642766
	Average	2.52634251	2.29453058	2.27664938	2.96868310	2.89089786	3.325546763	Inf	2.22521779	2.23298647	2.31528831	3.413837346	2.22549821
	Best	2.24342648	2.21821432	2.21815087	2.28976196	2.29388890	2.481443312	Inf	2.21815153	2.22150556	2.21890072	2.46378317	2.21821086
A2	Worst	10,381.2272	10,045.9362	12,830.0049	10,067.7997	10,063.602	5,363,315.938	Inf	10,381.3277	8805.33297	9124.22297	10,081.50622	8798.33612
	Average	9232.77493	8887.95315	9510.98433	9050.51091	8892.7700	425,478.9375	Inf	9250.24116	8800.97081	8813.54844	9070.629212	8798.62317
	Best	8796.91743	8796.90224	8798.16655	8797.11964	8796.8910	8812.111956	8796.86224	8796.86269	8797.28451	8796.87093	8797.238622	8796.86224
A3	Worst	−27.794682	250.000000	−27.944588	−30.841531	−31.749952	4.02762E+12	−32.347738	−32.335695	21.7986912	−25.024188	−31.6957728	−32.180684
	Average	−29.803636	89.3081117	−32.037423	−31.322198	−32.176488	2.01381E+11	−32.348631	−32.34775	10.7115515	−31.677356	−31.9847679	-32.32454
	Best	−32.283869	20.0000000	−32.348679	−31.957686	−32.331490	−31.92784414	−32.348679	−32.348674	−11.289916	−32.348022	−32.2733162	−32.345385
A4	Worst	−30,182.981	−30,665.539	−29,888.653	−29,439.890	−30,528.289	−25,752.30356	Inf	−30,663.523	−30,657.292	−30,661.088	−30,565.4162	−30,665.492
	Average	−30,565.794	−30,665.539	−30,595.378	−29,919.139	−30,644.105	−30,065.48897	Inf	−30,665.346	−30,663.308	−30,664.334	−30,656.5387	−30,665.538
	Best	−30,665.536	−30,665.539	−30,665.539	−30,271.938	−30,665.538	−30,665.53833	−30,665.539	−30,665.537	−30,665.461	−30,664.895	−30,665.5379	−30,665.539
A5	Worst	−1.8026342	−1.8093775	−1.1064446	−1.8593193	−1.8295595	−0.335482281	Inf	−0.1041214	−0.9771045	−1.8299627	−1.81135162	−1.8115606
	Average	−1.8366765	−1.8350057	−1.8004777	−1.8595864	−1.8491607	−1.738795718	Inf	−0.4333679	−1.6639668	−1.8510586	−1.83187919	−1.8503087
	Best	−1.8596187	−1.8596187	−1.8591456	−1.8596177	−1.8589513	−1.859572229	Inf	−0.711068	−1.7985176	−1.859361	−1.85723582	−1.8596187
A6	Worst	721.225818	681.323750	687.617841	722.659598	682.407589	5348.321081	Inf	680.645055	693.039994	681.324513	5348.327922	681.213774
	Average	684.802169	680.822245	681.931981	692.121169	681.304321	914.6709037	Inf	680.633948	686.635282	680.776655	1381.324093	680.849254
	Best	680.818565	680.640810	680.630251	684.078438	680.880284	680.8864863	680.630057	680.630348	681.839510	680.656912	680.778647	680.65168
A7	Worst	13,754.0689	20,100.0000	20,100.0000	15,565.1992	13,360.8386	12,831.59656	Inf	30,000.0000	20,100.0000	11,100.0000	20,100.0000	11,100.0000
	Average	9552.68395	10,073.5860	10,266.1734	11,604.3024	9733.70482	9442.224543	Inf	14,514.2915	11,159.5494	9777.08489	11,764.61853	8510.6779
	Best	7350.25309	7159.86820	7231.70207	8673.88334	7194.16177	7255.700767	Inf	7359.88938	8733.27896	7242.524	7142.889048	7097.16961
A8	Worst	44.2198361	52.7450673	60.8397049	55.4615502	50.8913823	1195.603831	78.1172795	26.1461192	66.076833	46.0219607	51.4656793	44.4223759
	Average	29.1433059	40.7929959	31.3881542	31.8770708	35.9360825	143.2336319	38.0563892	24.8548094	42.9715949	32.8771244	38.8758475	31.8050756
	Best	25.0238491	27.1385335	24.4769089	24.8032045	25.0291487	25.04409768	24.3074721	24.5891823	32.6484231	24.4587343	25.11747662	24.4339876
A9	Worst	733.756257	1361.51983	69.9547057	32.2928267	33.5977126	36.21483233	24.3843024	26.2192155	130.865805	27.6015694	29.69203454	27.9836566
	Average	68.5435258	353.735124	29.3858811	26.4384586	27.5168676	29.08217874	24.3257078	24.8762411	70.0387934	25.6649044	26.96463473	25.9982048
	Best	24.8026349	32.2028969	24.3407911	24.8995841	24.8774028	24.86933023	24.3068157	24.4867034	42.8970745	24.4915786	25.25362982	24.5303489
A10	Worst	1,658,576.88	7,890,093.03	3,552,427.57	3,462,088.75	3,361,926.48	1,674,541.294	2,676,689.29	37,469.8098	10,800	157.30916	106,992,179.6	42,716.8544
	Average	660,378.744	834,489.085	1,211,371.64	878,067.101	654,146.325	432,236.6812	470,814.458	15,345.2894	7663.03193	79.5629561	17,497,690.03	13,559.402
	Best	182,987.463	4708.34471	66,203.7697	120,840.952	11,493.9251	53,354.07871	24,051.5775	5948.0523	1895.5389	42.8180788	594,454.0612	3947.67288
A11	Worst	−0.3448604	−0.30043505	−0.2266421	−0.5409624	−0.5085986	−0.595908871	−0.0682942	−0.1847184	−0.479115	−0.4721329	−0.59515589	−0.5869796
	Average	−0.5401232	−0.46890380	−0.2697497	−0.6106503	−0.6372634	−0.679385266	−0.2768513	−0.2535384	−0.5552904	−0.5734288	−0.70395134	−0.7155546
	Best	−0.6524537	−0.69151720	−0.3274912	−0.6885093	−0.7648467	−0.760508224	−0.5311755	−0.3704158	−0.6509533	−0.6667437	−0.78290762	−0.7799606
Success fraction	Best	0.64	0.55	0.73	0.45	0.55	0.64	0.64	0.64	0.36	0.55	0.55	0.82
	Average	0.18	0.36	0.27	0.36	0.45	0.09	0.27	0.45	0.18	0.45	0.27	0.55

Table 2
Best, average and worst objective function values of the solutions returned over 20 runs by the twelve tested algorithms within 10,000 objective function evaluations for the test problems of Group B (constrained non-smooth with numerical noise). The last row indicates the fraction of problems which are successfully solved (see the definition of "data profile" in Section 7.3) within a tolerance value of 10^{-4}. "Inf" indicates that, for such test problem with unrelaxable linear or nonlinear constraints, CMAES fails to find a feasible solution (since it is not possible to pass as input the starting set of solutions, it tries and often fails to find solutions which satisfies all the unrelaxable constraints).

Test Problem	Run	PSwarm	CPSO	Complex	GSS	LTMADS	OrthoMADS	CMAES	CRS	SSm	eSS	VNS-MADS	PGS-COM
B1	Worst	3.49596003	2.56380914	2.4591985	3.68722145	3.46903559	3.92508698	Inf	2.43294921	2.23821365	3.5692142	5.564210819	2.24241734
	Average	2.54545638	2.32274245	2.26710645	2.8356073	2.8213771	2.94004024	Inf	2.2509392	2.22538558	2.39831694	3.73895037	2.26044652
	Best	2.22808019	2.21597477	2.21595801	2.42351456	2.32002288	2.38697723	Inf	2.21595865	2.21676883	2.21713652	2.22788094	2.21600326
B2	Worst	10,372.0787	10,375.2364	292,213.24	10,060.6511	9194.59787	311,275.031	356,694.501	10,375.2381	8804.58182	9444.02339	10,062.95437	9761.58577
	Average	9588.81964	8881.73612	23,923.5059	9206.28975	8862.30115	24,936.2786	27,076.5159	9294.83548	8792.2369	8823.02419	9141.478505	8855.01403
	Best	8789.00895	8788.32007	8788.28298	8792.22599	8788.08254	8790.92422	8816.5921	8788.07563	8788.5579	8788.19694	8788.55797	8788.07887
B3	Worst	−27.913912	249.926	−19.216785	−30.82811	−32.044834	7.951E+11	−32.130369	−32.378712	44.2704413	−15.203186	−31.29612397	−32.121255
	Average	−30.34059	109.314409	−30.601309	−31.38876	−32.205208	3.9755E+10	−32.327939	−32.379067	11.9514799	−31.016059	−31.83442784	−32.32504
	Best	−32.232103	20.0172218	−32.379232	−31.905081	−32.345848	−31.857798	−32.379232	−32.3792	−8.1769345	−32.378043	−32.1786093	−32.368292
B4	Worst	−30,213.335	−30,213.823	−30,187.799	−29,836.407	−29,653.271	−28,411.396	Inf	−30,688.087	−30,667.992	−30,674.514	−30,537.71924	−30,689.552
	Average	−30,593.408	−30,671.129	−30,544.983	−30,125.265	−30,332.127	−30,075.076	Inf	−30,694.903	−30,688.102	−30,691.725	−30,652.03823	−30,693.831
	Best	−30,695.497	−30,695.198	−30,687.801	−30,587.371	−30,680.561	−30,654.016	−30,693.847	−30,695.757	−30,695.014	−30,695.072	−30,695.28668	−30,695.762
B5	Worst	−1.6300519	−1.8121525	−1.5176661	−1.8236525	−1.8170316	−0.3728159	Inf	−0.0442307	−1.6046689	−1.8101468	−0.131669304	−1.8256873
	Average	−1.8156969	−1.8443058	−1.8102935	−1.8533611	−1.8489209	−1.628339	Inf	−0.3683332	−1.7172972	−1.8403532	−0.426489656	−1.8469406
	Best	−1.8609089	−1.8609228	−1.8609048	−1.8612368	−1.8603689	−1.8605111	Inf	−0.6316374	−1.8173647	−1.8613394	−0.631637369	−1.8609111
B6	Worst	699.199516	681.054149	5375.9533	732.685983	5348.4346	8067.96729	Inf	680.072503	695.066774	681.726082	5348.231336	681.039647
	Average	682.677817	680.369508	920.49546	694.064264	916.331491	1353.00724	Inf	680.031963	684.492548	680.659529	1148.200142	680.567667
	Best	680.157745	680.015424	680.391724	682.681415	680.82672	680.329918	680.093822	679.996332	681.190446	680.069784	680.3658983	680.073568
B7	Worst	11,097.9416	11,097.9	12,219.2066	20,507.1573	14,102.062	26,235.2634	Inf	29,088.9003	20,097.9003	11,097.9035	20,097.90018	11,097.9000
	Average	8951.69832	8858.05559	9226.01828	12,200.0578	9776.8959	11,838.9609	Inf	16,277.6936	10,346.3004	9605.24547	10,177.43734	8199.88868
	Best	7245.53492	7179.02238	7174.93512	7930.18314	7261.87033	7226.14851	Inf	7251.32641	7460.31243	7074.80897	7672.552252	7143.08576
B8	Worst	42.7916519	70.8544789	64.7553443	50.4458945	73.9842223	602.60886	79.8688307	27.9210513	60.7787404	44.9243735	59.47976456	45.1471644
	Average	28.5179219	34.822315	41.8877077	33.3158327	34.9213341	66.1721945	36.3033177	25.1379766	38.2385788	31.1445233	42.07236705	29.9327352
	Best	25.0676765	25.0516296	24.9934651	25.2971558	25.1881188	26.7375707	24.3053162	24.5946836	27.4621304	24.521271	26.19902696	24.4717428
B9	Worst	181.457634	1279.7567	47.2337708	43.8829091	36.1794353	11,721.8121	25.0169589	28.94924	50.924271	36.9437125	73.39158548	28.3838805
	Average	34.1301132	407.259801	30.0281571	28.1604101	28.8928531	782.872549	24.503173	24.9207396	35.7342293	26.4548771	48.28603788	26.2742962
	Best	24.6045782	29.7329381	24.6556992	24.9712936	26.2845899	25.0075369	24.3036452	24.5153717	27.4424342	24.4913246	33.69582707	24.7002037
B10	Worst	2,183,395.08	2,227,188.21	6,859,659.26	883,362.314	1,332,147.74	2,643,982.55	17,974,402.3	26,658.7093	5736.20248	143.559892	197,813,660.8	26,512.6947
	Average	697,679.965	529,864.194	2,685,065.36	365,758.148	516,681.552	596,714.46	3,039,614.98	13,488.5893	2321.87663	71.8426343	34,761,114.14	13,005.9807
	Best	121,510.24	2417.97179	524,600.12	87,941.1336	19,551.9829	43,906.6712	19,074.4713	4160.9621	718.00709	44.1766154	12,332.55506	4952.56738
B11	Worst	−0.4316943	−0.2747674	−0.1992372	−0.5061468	−0.411744	−0.5883468	Inf	−0.1980177	−0.4946438	−0.4459606	−0.648045136	−0.6120025
	Average	−0.5657821	−0.4591929	−0.2583222	−0.593291	−0.6258436	−0.6765585	Inf	−0.2436665	−0.5771995	−0.6089808	−0.727241901	−0.708624
	Best	−0.6569626	−0.6564671	−0.3119751	−0.7035565	−0.7795049	−0.7536285	−0.4247109	−0.3795814	−0.642517	−0.7633589	−0.788639347	−0.7790329
Success fraction	Best	0.55	0.45	0.55	0.36	0.45	0.45	0.55	0.64	0.36	0.55	0.45	0.73
	Average	0.18	0.27	0.18	0.27	0.27	0.09	0.27	0.36	0.18	0.45	0.18	0.45

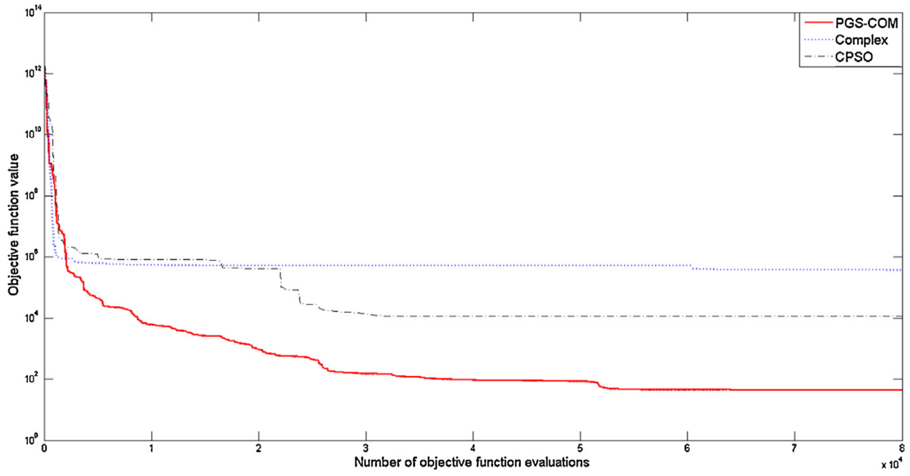

Fig. 11. Convergence curves (see definition in Section 7.1) of PGS-COM, CPSO and Complex for test problem A10.

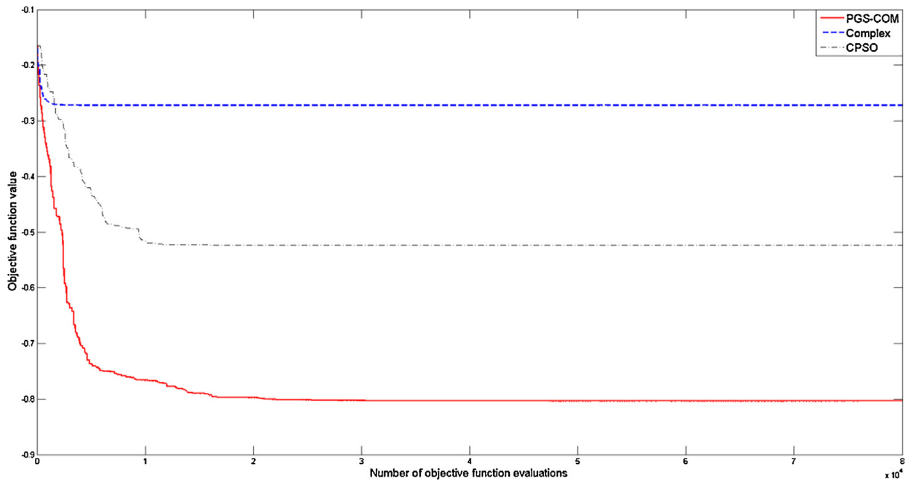

Fig. 12. Convergence curves (see definition in Section 7.1) of PGS-COM, CPSO and Complex for test problem A11.

different algorithms within 10,000 function evaluations. Table 3 indicates the total number of function evaluations carried out by the Complex search step of PGS-COM. Figs. 11 and 12 show the convergence curves (i.e., the current solution value as a function of the number of black-box evaluations) of PGS-COM, CPSO and Complex for test problems A11 and B11. Such plots correspond to the best convergence curve (the one leading to the lowest objective function value) over five runs started from the same set of solutions.

Table 3
Average total number of function evaluations carried out by the Complex search step of PGS-COM over 20 runs within 10,000 evaluations of the objective function.

Test problem	Group A	Group B	Max fraction, %
1	67.5	54.9	0.67
2	305.75	139.8	3.06
3	94.8	93.1	0.95
4	134.75	71.85	1.35
5	437.1	215.8	4.37
6	29.7	46.05	0.46
7	89.95	100.4	1.00
8	38.05	45.7	0.46
9	29.4	33.55	0.34
10	21.45	21.2	0.22
11	12.75	18.6	0.19

7.1. PGS-COM performance

According to the best solution values reported in Table 1, PGS-COM returns the lowest or close to the lowest best solution values for all the test problems of Group A except for the non-differentiable test problems A10 and A11. To successfully solve these problems, PGS-COM requires more than 20,000 function evaluations, as shown by the convergence curves in Figs. 11 and 12. The convergence curves plot for each algorithm the current best objective function value as a function of the number of objective function evaluations for the best run out of five consecutive ones starting from the same set of initial solutions (kept constant for each run and for each algorithm). The purpose of the convergence curves in Figs. 11 and 12 is just to provide a qualitative idea of the PGS-COM behavior with respect to those of Complex and CPSO.

On the basis of the criterion introduced in Section 7.3 and the best solution values, with 10,000 function evaluations PGS-COM can successfully solve within a tolerance value of 10^{-4} all the test problems except for A10 and A11. This fraction is larger than those achieved by the other algorithms. As far as the average and worst solution quality are concerned, PGS-COM performs rather well, finding the lowest or close to the lowest average and worst solution values for nine out of eleven problems (all except A8 and A10).

According to the results reported in Table 2, numerical noise does not significantly affect PGS-COM performance. Among the tested methods, PGS-COM still successfully solves the largest fraction of test problems (73% and 45% in terms of the best and, respectively, average solution values) and returns either the lowest or close to the lowest best, average and worst solution values. Instead, the Complex effectiveness is considerably affected by numerical noise, as indicated by the large decrease in the number of problems successfully solved (from 73% without noise to 55% with noise). This result confirms the observation mentioned in Section 4: PGS-COM is more tolerant than Complex toward numerical noise because the GSS step uses poll directions which span the feasible space around the current solution, and a step size parameter is bounded below by a strictly positive minimum value. These two features of the GSS step reduce the probability of premature convergence to suboptimal solutions, which is a limitation of the plain Complex (see Section 2.1).

Another important motivation for the GSS step is to limit the number of Complex iterations, as mentioned in Section 4. According to the results reported in Table 3, the tested version of PGS-COM achieves such goal, since the fraction of function evaluations carried out by the Complex steps is at most 4.5% of the tot.

Tables 1 and 2 also indicate that PGS-COM performs definitively better than plain CPSO and plain Complex. While CPSO and Complex provide good (within a tolerance value of 10^{-4}) best solution values for respectively 55% and 73% of the test problems, and good average solution values for respectively 36% and 27%, PGS-COM returns good best solutions for 82% of the problems, and good average solutions for 55% (see Table 1). In addition, as illustrated in Figs. 11 and 12, in the first iterations the PGS-COM convergence curve is very steep, much steeper than that of CPSO and rather close to that of Complex. This is likely due to the positive effect of the GSS and Complex steps following the unsuccessful particle swarm search steps. Then, after a few thousands function evaluations, while Complex tends to be trapped into a local minima and stops descending, PGS-COM keeps exploring the feasible region and then progressively finding better solutions by means of the particles which are still scattered over the feasible region. For this reason, in the PGS-COM algorithm it is important to slow down the particle swarm convergence rate by using the *l-best* velocity update formula and a small neighborhood size (see Section 4.6).

7.2. Comparison between existing methods

For non-smooth black-box problems without numerical noise, Complex appears to perform better than other existing methods in terms of best solution quality. However, the algorithm does not show a good reliability (its average solution values are quite poor) and robustness to numerical noise (the solution quality drops when switching to the noisy problems of Group B). CRS returns good quality best solution values for most of the test problems but rather poor quality average solution values. A key advantage of CRS compared to Complex is the robustness to numerical noise. CMAES provides high quality best solution values for five test problems of Group A and three test problems of Group B. However, since it was originally designed to tackle bounded problems and then extended to handle general nonlinear constraints with a penalty approach, it cannot deal with unrelaxable and hidden (linear as well as nonlinear) constraints. The major difficulty is in the generation of the starting set of solutions: the code does not allow to pass the set of starting solutions as an input and tries to generate feasible solutions by means of its default routine for bounded problems. For problems with unrelaxable constraints like problems A1, A2, A4, A5, A6 and A7, the initialization routine fails to find a feasible solution and the algorithm gets stuck. The same difficulty arises for the problems of Group B. PSwarm provides close-to-optimal best solution values for slightly more than half of the test problems, but the solution quality is rather poor. This may be due to the fact that, on the one hand, PSwarm drops particles which become too close to each other, and on the other hand, only a few polling directions are considered in the GPS polling step. Therefore, when particles get close to each other, most of the particles of the swarm are killed and the search proceeds with GPS poll steps which may fail in situation like those depicted in Fig. 5. PSwarm effectiveness appears to be essentially unchanged for noisy problems, thus confirming the stability of particle swarm and pattern search methods toward numerical noise. OrthoMADS returns close-to-optimal best solution values for most of the test problems but the quality of the solutions is pretty bad and the average performance is rather poor. Moreover, it is further penalized by numerical noise. Compared to OrthoMADS, LTMADS appears to be fairly reliable (in terms of average solution values and fraction of problems successfully solved) and more stable toward numerical noise.

As far as reliability, average performance and algorithm stability toward numerical noise are concerned, eSS turns out to be the best existing method. However, due to the lack of a local search solver or solution improvement algorithm (see Section 6), the solution quality is worse than that of the other methods.

Within the limit of 10,000 function evaluations, CPSO, VNS-MADS, GSS, and SSm turn out to be less competitive than the above-mentioned methods. We will see in the next subsection that SSm becomes very competitive when a larger number of function evaluations is available.

It is worth noting that, despite the asymptotic convergence guarantees of the MADS variants and the properties of their poll directions, our computational experiments show that LTMADS, OrthoMADS as well as VNS-MADS perform better than GSS (as claimed by Audet et al., 2006) but worse than Complex, CRS and CMAES (and of course PGS-COM) in terms of best solution values on most of the test problems. In principle this may be due to: (1) an inappropriate choice of the algorithm parameter values (e.g., excessively small initial mesh size, too small mesh contraction coefficient, ineffective scaling strategy of the optimization variables) which may lead to premature convergence of the mesh size parameter (see Section 2.5) to the minimum threshold value, (2) the limited number of runs (20) considered in the experiments. However, as far as point (1) is concerned, it is worth emphasizing that the average, worst and best solution values we obtained for the styrene problem described in Section 8.1 are better than those found by Audet, Bechard, and Chaouki (2008a) and Audet, Bechard, and Le Digabel (2008b). Concerning point (2), although such an increase in the number of runs is likely to lead to an improvement in the best solution values, in this work we are interested in real engineering applications in which the optimization is repeated just a few times, rather than assessing the performance of MADS algorithms with a very large number of runs (e.g., more than 100).

7.3. Data profiles

For a systematic comparison of derivative-free methods, Moré and Wild (2009) proposed the use of so-called “data profiles”, which measure the algorithm ability to improve the starting solution $\boldsymbol{x}_0$ as a function of the number of function evaluations. This is of particular interest when tackling optimization problems with computationally expensive objective function evaluations. Data profiles plot the fraction of test problems of a given set that can be solved by a given method within a given tolerance $\tau > 0$ with respect to the starting point value $f(\boldsymbol{x}_0)$, in at most N_f function evaluations. More precisely, a problem is considered successfully solved if the algorithm returns a solution value f_{solver} such that

$$f_{\text{solver}} - f_{\text{best}} \leq \tau \cdot (f(\boldsymbol{x}_0) - f_{\text{best}}), \quad (28)$$

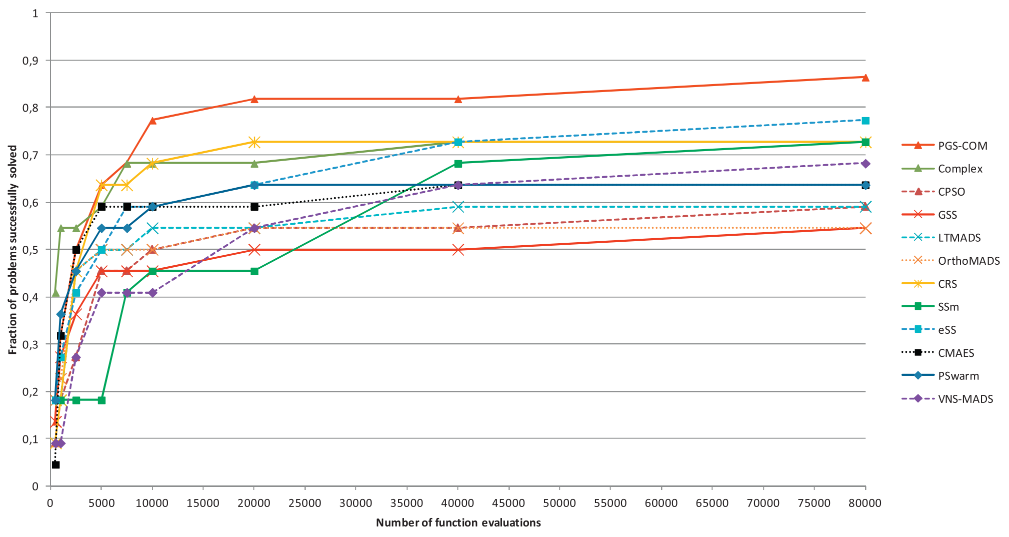

Fig. 13. Data profiles plotting the fraction of problems (of Groups A and B) that can be solved within a given tolerance $\tau = 10^{-4}$ as a function of the maximum number of the objective function evaluations. These plots refer to the best solution values found by each algorithm.

where f_{best} denotes the value of the best known solution of the test problem (or, alternatively, the value of the best solution found by the tested solvers) and τ is an accuracy parameter.

In Figs. 13–15 we report the "data profiles" relative to the complete set of problems (Group A and B) for different τ values and with respect to both average and best solution values found by the twelve algorithms under consideration. According to Fig. 13, when the best run performance is concerned, PGS-COM performs better than all the other methods for $N_f \geq 7500$. For $N_f < 7500$ PGS-COM performs better than the other methods except for Complex. More precisely, PGS-COM outperforms the other available methods satisfying the criterion of Eq. (28) for:

- about 77% of the test problems for $N_f = 10{,}000$ and $\tau = 10^{-4}$, while CRS and Complex do so only on about 68% of the test problems;
- about 82% of the test problems for $N_f = 40{,}000$ and $\tau = 10^{-4}$, while CRS, Complex and eSS do so on about 73% of the test problems;
- about 86% of the test problems for $N_f = 80{,}000$ and $\tau = 10^{-4}$, while eSS and SSm progressively reach respectively about 78% and 73%, and CRS is stuck at 73%.

It is worth pointing out that for $N_f \geq 10{,}000$ PGS-COM returns also very accurate solutions for the largest fraction of test problems, as shown in Fig. 14. Indeed, PGS-COM satisfies the criterion of Eq. (28) with $\tau = 10^{-7}$ for more than 50% of the test problems, appreciably better than the other methods.

PGS-COM outperforms the other methods in terms of best solutions but it also performs quite well in terms of average solution values. Fig. 15 indicates that for $N_f < 5000$ PGS-COM provides good quality (with $\tau = 10^{-4}$) average solution values for about the same fraction of problems as eSS which performs best among the existing methods. Then, for $N_f \geq 7500$ PGS-COM performs better than eSS, showing good average solution values for about 60% of the test problems. For more than 60,000 objective function evaluations, SSm provides good average solution values for a larger fraction of problems than PGS-COM. However, the quality of the best solutions returned by SSm is worse than that of PGS-COM (as indicated by the data profile of Fig. 14).

7.4. *Tuning PGS-COM to improve the solution quality*

As mentioned in Section 6, the quality of the solutions found by PGS-COM may depend on the number of iterations (reflections) executed by the Complex step. For several test problems, increasing N_{COM} leads to an appreciable improvement in the best solution values. This is likely due to the fact that the larger number of reflection iterations allows the Complex step to determine the descent direction with higher accuracy, and provide to the GSS step (of the next iteration) more promising poll directions. On the other hand, increasing N_{COM} can substantially slow down the PGS-COM algorithm because a larger fraction of the function evaluations is carried out sequentially (within the Complex search step). For instance, on test problems A1 and A7 increasing N_{COM} from 2 to 3 would be advantageous since the best solution value achieved with 10,000 function evaluations is improved from 2.21821 to 2.21818, and respectively, from 7097.17 to 7070.45 (becoming quite close to the values of the global optima), while the fraction of function evaluations devoted to the Complex step increases by only 0.3 percentage points (i.e., respectively from 0.6% to 0.9%, and from 0.9% to 1.2% of the total function evaluations). Instead selecting N_{COM} equal to 6 leads to a relatively small gain in the solution quality (respectively 2.21816 and 7060.24 for problems A1 and A7), and a considerable increase in the function evaluations of the Complex step (respectively 1.8% and 3.1% of the total 10,000 function evaluations). Moreover, if 100,000 function evaluations are considered, the Complex step's computational load becomes as high as 20% of the total. Thus, if PGS-COM can be executed in parallel and the objective function evaluation is particularly time consuming, it is not worth exceeding $N_{\text{COM}} = 3$.

7.5. *Larger scale test problems*

In this subsection, we test PGS-COM on problems with 30 variables in order to check if its search effectiveness holds up on larger scale problems. We consider three of the largest scale test problems with inequality constraints from Mallipeddi and Suganthan (2010) for the "CEC 2010 Competition on Constrained Real-Parameter Optimization", namely problems C01, C07 and C08, and we have included numerical noise with Eqs. (25)–(27). All constraints are considered as unrelaxable (if violated, the objective function cannot be evaluated). Given the large number of variables and the high multimodality of the functions, we have increased the number of PGS-COM particles from 30 to 90 in order to improve the exploration capability of the particle swarm step. We have compared PGS-COM best, average and worst solution values obtained

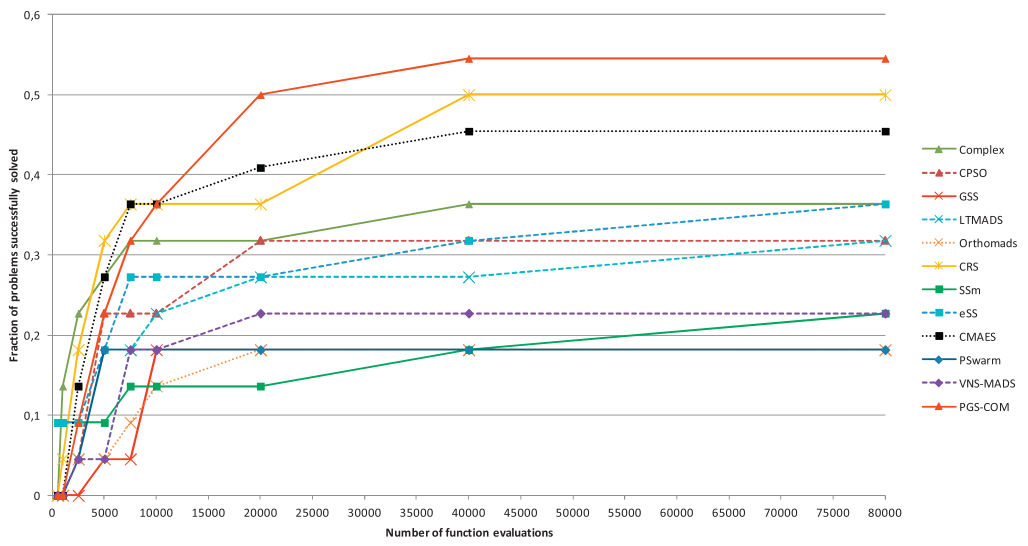

Fig. 14. Data profiles plotting the fraction of problems (of Groups A and B) that can be solved within a given tolerance $\tau = 10^{-7}$ as a function of the maximum number of objective function evaluations. These plots refer to the best solution values found by each algorithm.

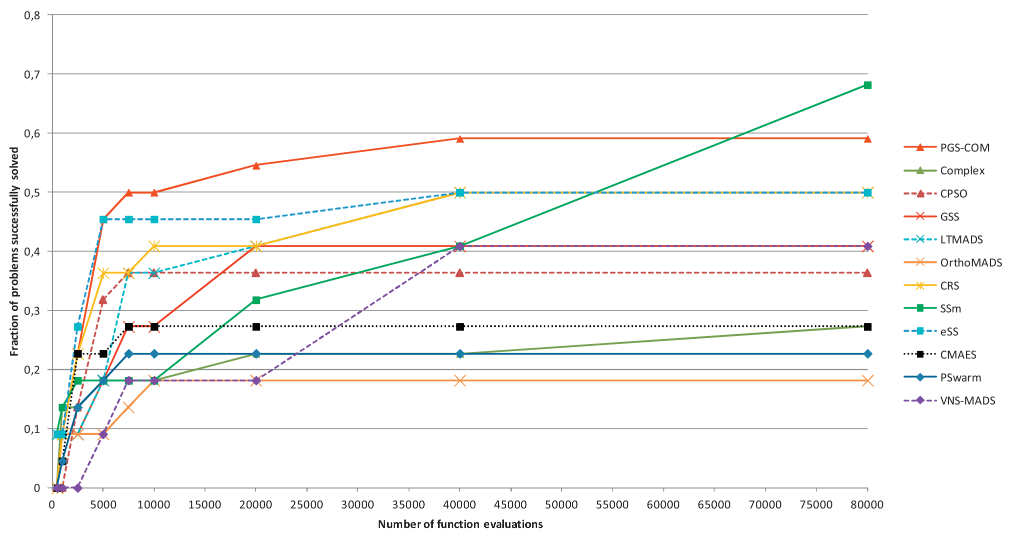

Fig. 15. Data profiles plotting the fraction of problems (of Groups A and B) that can be solved within a given tolerance $\tau = 10^{-4}$ as a function of the maximum number of objective function evaluations. These plots refer to the average solution values found by each algorithm.

after 300,000 objective function evaluations with those of CPSO, Complex, CRS, SSm, eSS and VNS-MADS (see Table 4).

PGS-COM finds either the lowest or close to the lowest best, average and worst solution values for all the larger scale test problems. This result is remarkable, given the large number of variables and the challenging features of the problems (multimodal objective function, unrelaxable constraints which do not allow the objective function evaluation, numerical noise). It is worth recalling that both SSm and eSS, which are particularly appropriate for large scale problems, are designed to handle constraints with the penalty function which is suitable for relaxable constraints but not for unrelaxable constraints.

Table 4
Best, average and worst objective function values of the solutions returned over 20 runs by the tested algorithms within 300,000 objective function evaluations for the large scale test problems C01, C07, C08 of the CEC 2010 competition.

Test problem	Run	CPSO	Complex	CRS	SSm	eSS	VNS-MADS	PGS-COM
C01	Worst	−0.50319	−0.2282999	−0.29663595	−0.73367	−0.55961	−0.7045879	−0.674060959
	Average	−0.56794	−0.3089571	−0.42680617	−0.78861	−0.69908	−0.7723417	−0.755645543
	Best	−0.73242	−0.386156	−0.51721142	−0.81935	−0.78756	−0.8084187	−0.814214794
C07	Worst	1520.786	8,163,664.55	223.6525458	220.7985	116.5842	10,110.3133	74.12204721
	Average	164.1593	463,118.493	35.59383562	130.6964	36.09778	1930.1025	18.07767872
	Best	0.05103	168.39598	9.26935E−05	74.4601	3.105138	2.66135558	0.024487812
C08	Worst	970,063	1,027,994,062	7286.155578	193.1937	7979.948	7958.2639	211.5097785
	Average	50,538.24	278,256,210	720.2104797	88.90821	1367.951	1573.21093	67.41424547
	Best	0.001183	43,576,573.7	0.00154131	35.35466	6.941823	13.4062806	0.030443494

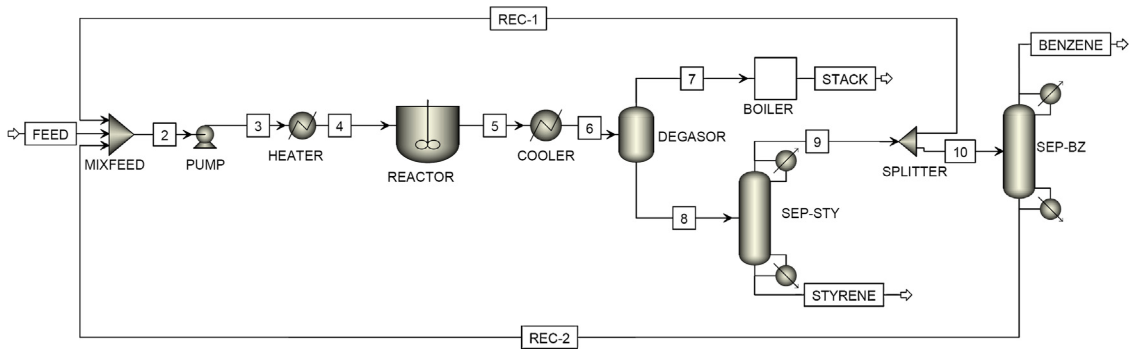

Fig. 16. Flowsheet of the styrene production process simulated and optimized in Audet et al. (2008a,b).

8. Computational results for process engineering applications

In this section we assess the PGS-COM performance and compare it with that of the benchmark algorithms on two relevant process engineering problems. The first application, introduced in Audet, Bechard, and Chaouki (2008a) and Audet, Bechard, and Le Digabel (2008b), consists in the techno-economic optimization of a styrene production process by means of a sequential flowsheet simulation code. The function to be optimized turns out to be discontinuous, noisy and not defined on the points where the simulation solver fails to reach convergence (i.e., the problem has hidden constraints). In addition, the test problem is characterized by a set of unrelaxable techno-economic constraints. This application is representative of the typical process engineering problem arising when using sequential flowsheet simulation solvers. The second problem consists in the optimization of the design of a heat recovery steam cycle (HRSC) for an integrated gasification combined cycle (IGCC) with CO_2 capture. According to the design methodology of Martelli, Amaldi, et al. (2011), the optimization problem can be decomposed into a bilevel program with a constrained nonsmooth nonlinear program at the upper level, which needs to be tackled with a derivative-free method, and a linear program at the lower level. This test case is also representative of optimization problems which are decomposed into bilevel programs, such as the decomposition strategy described by Gassner and Marechal (2009) for the optimal synthesis and design of total-plants (process and utility systems).

It is worth noting that in both applications the objective and constraint functions are noisy because they are computed by iterative solvers.

8.1. Styrene process optimization

The process flowsheet of the styrene production process from ethylbenzene described by Audet, Bechard, and Chaouki (2008a) and Audet, Bechard, and Le Digabel (2008b) is reported in Fig. 16. It is essentially based on the arrangement described by Shiou-Shan Chen (2006). The feed stream is mixed with recycled ethylbenzene, pressurized, and then heated up to the reactor inlet temperature. The catalytic reactor promotes the endothermic conversion of ethylbenzene into styrene and hydrogen ($C_6H_5C_2H_5 \rightarrow C_6H_5C_2H_3 + H_2$). The products are cooled and sent to a separator, in which light gases (hydrogen, methane, ethylene), and organic liquid exit in separate streams. The hydrogen rich stream is burned into a boiler which provides heat for the chemical reaction. The organic liquid is distilled to separate benzene, toluene (produced by unwanted side reactions which may occur into the reactor) and unconverted ethylbenzene from the styrene product. The vapor stream is distilled in a second column so as to separate benzene from the other compounds (mainly unconverted ethylbenzene) which are recycled back to the process inlet.

The reactor kinetics, the process energy and mass balances as well as the sizing of the main equipment units are computed by a sequential/modular simulation solver which is coded in C++ and can be downloaded from the webpage of the MADS research project

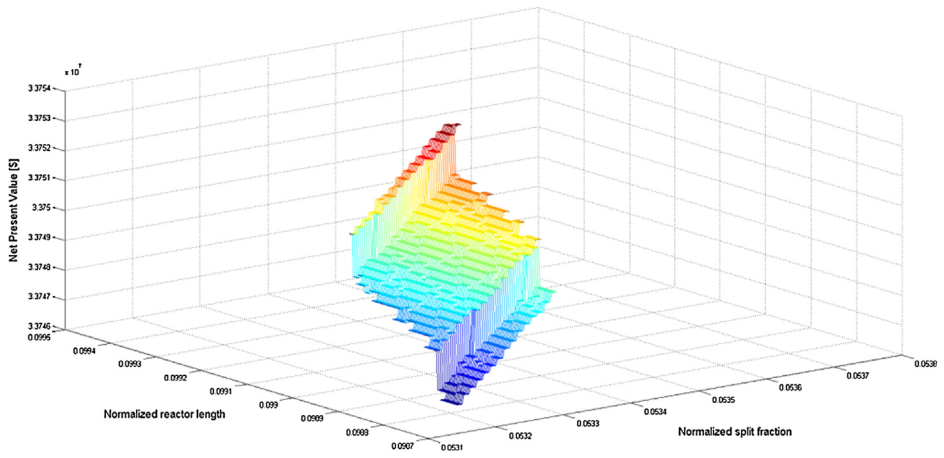

Fig. 17. Plot of the objective function (NPV) of the styrene process optimization problem with respect to the reactor length and the split fraction of the SPLITTER (see Fig. 16) around the best solution found by PGS-COM and CPSO (reported in Table 6). Note that the two variables have been normalized within the range 0–0.1, and the objective function value is only defined on a small part of the variable space.

(http://www.gerad.ca/nomad/Project/Home.html). The code also computes the costs of the main equipment units and the Net Present Value (NPV) of the overall plant. Such code is used as a black-box function within the following optimization problem:

- Decision variables: reactor temperature (between 600 and 1100 K), reactor length (between 2 and 20 m), light key fractions of the two distillation columns (between 10^{-4} and 0.1), reactor pressure (between 2 and 20 atm), split fraction of block SPLIT (between 0.01 and 0.5), air excess of the boiler (between 0.1 and 5), and cooling temperature of block COOLER (between 300 and 500 K).
- Objective: maximize the plant Net Present Value (NPV)
- Unrelaxable process constraints:
  - structural feasibility of column SEP-STY (see Fig. 16)
  - structural feasibility of column SEP-BZ
  - flammability of stream 7
  - upper limit on CO and NO_X concentrations in stream "STACK"
  - minimal purity of produced styrene
  - minimal purity of produced benzene
  - minimal overall ethylbenzene conversion rate into styrene
- Unrelaxable economic constraints:
  - maximum payout time
  - minimum discounted cashflow rate of return
  - maximum total investment
  - maximum annual equivalent cost

Since the iterative solver of the flowsheet may not converge for some values of the optimization variables, we also need to take into account hidden constraints. If one of the above mentioned constraints is not satisfied, the black-box sets the objective function value equal to $-\infty$ (extreme barrier approach). A plot of the objective function with respect to the reactor length and the split fraction of the SPLITTER (see Fig. 16) is reported in Fig. 17. It clearly shows that the function is discontinuous and not defined in a large portion of the region satisfying the variable bounds. The value of the best solution known is equal to 33,539,100 $ and it has been found with the hybrid VNS-MADS algorithm of Audet, Bechard, and Chaouki (2008a) and Audet, Bechard, and Le Digabel (2008b). Notice that the simulation code returns different values on different computing environments and the numerical results presented in this work are reproducible under the 32 bit Microsoft® "Windows Seven" environment. Instead, the best solution value of 33,618,600 $ reported in Audet et al. (2010) with LTMADS is obtained under the Mac OS 10.5.5 environment.

This instance has been solved with all the algorithms listed in Section 6 except CMAES, as it failed to generate starting solutions satisfying all the unrelaxable constraints. Since the evaluation of the objective function is quite time consuming (it requires about 0.9 s), we have compared the average, best and worst values of the solutions returned by the eleven algorithms within 2500, 5000, 7500 and 10,000 function evaluations. Each algorithm has been run 20 times on each test problem up to the four specified limits of function evaluations. The total computing time amounts to about 60 days. Unfortunately, it was not possible to execute PGS-COM and other algorithms in the parallel computing mode because we used a system call to run the black-box function of the process simulation model within Matlab.

The computational results are summarized in Table 5. First of all, it is worth noting that PGS-COM as well as CPSO find best solution values which are better than that reported in Audet, Bechard, and Chaouki (2008a) and Audet, Bechard, and Le Digabel (2008b) with NPV = 33,539,100 $. PGS-COM and CPSO provide a solution with NPV = 33,753,600 $ (see Table 6). Also PSwarm, Complex, OrthoMADS, CRS, eSS and VNS-MADS find best solutions better than that of Audet, Bechard, and Chaouki (2008a) and Audet,

Table 5
Computational results for the "styrene production process" optimization problem.

# f. eval.	Run	PSwarm	CPSO	Complex	GSS	LTMADS	OrthoMADS	CRS	SSm	eSS	VNS-MADS	PGS-COM
2500	Worst	31,352,500	32,770,400	32,230,400	25,832,800	29,322,700	24,723,000	27,599,000	29,008,800	31,849,400	32,608,200	32,865,400
	Average	32,798,325	33,098,540	32,415,985	29,842,935	32,415,220	30,102,050	31,154,770	31,679,545	32,854,625	32,873,070	33,069,015
	Best	33,406,700	33,144,300	33,520,200	32,597,400	33,001,900	32,910,800	33,146,000	33,124,600	33,324,400	33,034,600	33,625,300
5000	Worst	32,028,800	32,940,200	30,767,500	26,587,100	32,783,800	18,340,200	29,333,000	29,335,600	32,948,100	29,335,200	33,001,200
	Average	32,557,525	33,119,385	32,284,040	29,637,515	32,907,680	29,890,325	32,344,480	32,393,820	33,073,500	32,258,965	33,118,740
	Best	33,150,500	33,210,500	33,624,600	32,557,600	33,133,400	32,941,300	33,151,500	33,371,800	33,143,500	33,136,900	33,753,600
7500	Worst	32,430,400	32,948,900	30,498,400	27,193,200	32,597,900	28,070,700	29,345,500	32,177,500	30,930,100	29,330,600	33,120,300
	Average	32,853,915	33,099,695	32,452,535	30,591,920	32,941,475	32,394,515	32,559,375	33,041,015	32,988,330	32,728,705	33,217,035
	Best	33,326,800	33,643,600	33,519,500	32,500,600	33,367,800	33,463,500	33,632,800	33,509,400	33,624,700	33,566,900	33,645,200
10,000	Worst	30,744,300	32,865,499	30,767,500	28,130,800	32,847,600	29,278,800	28,823,400	32,947,600	32,862,500	32,846,300	33,149,000
	Average	32,981,310	33,182,525	32,500,095	30,903,065	33,096,170	32,396,780	31,843,270	33,122,040	33,167,545	33,091,945	33,296,715
	Best	33,638,900	33,753,600	33,494,300	32,886,490	33,519,100	33,623,000	33,153,200	33,330,900	33,621,900	33,580,600	33,753,600

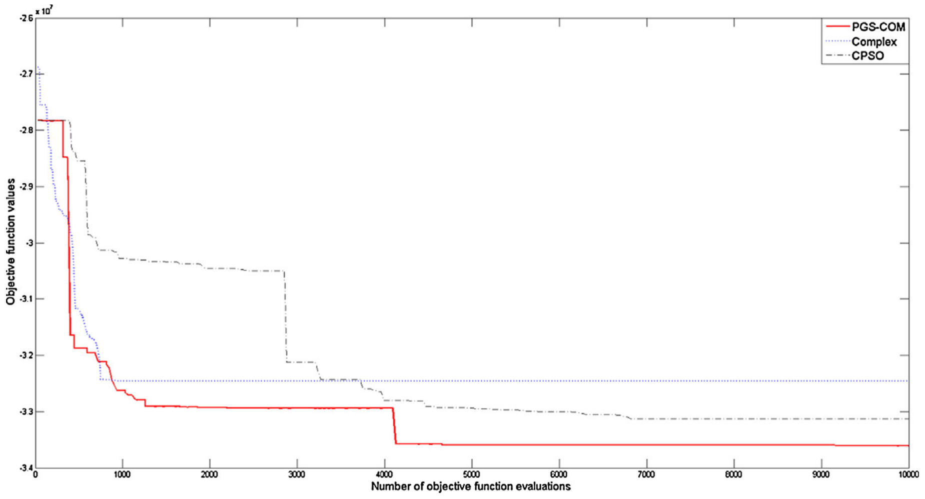

Fig. 18. Convergence curves (see definition in Section 7.1) of PGS-COM, CPSO and Complex for the styrene process design problem. Note that the plot reports the values of −NPV on the *y* axis (having formulated it as a minimization problem).

Bechard, and Le Digabel (2008b). Compared to CPSO and the other algorithms, PGS-COM is capable of reaching close-to-optimal best solutions within 5000 function evaluations, as shown also in Fig. 18.

As far as average and worst solutions are concerned, Table 5 indicates that PGS-COM finds also better average and worst solutions than CPSO and the other methods.

Note that that CRS, which performed very well on the synthetic test problems of Groups A and B, appears to be less effective on this real-world problem in terms of worst, average and best solution values. Population-based methods, namely, CPSO, eSS, SSm and PSwarm seem to be more suitable for this test problem. The two MADS variants outperform GSS, as claimed by the developers, but worse than PGS-COM and even CPSO. Finally, as previously mentioned, it is worth emphasizing that the MADS results (LTMADS and VNS-MADS) in Table 5 are better than those in Audet, Bechard, and Chaouki (2008a), Audet, Bechard, and Le Digabel (2008b) and Audet et al. (2010).

8.2. *Optimal design of a heat recovery steam cycle*

The problem consists in the optimization of the design of a triple pressure level heat recovery steam cycle (HRSC) for an integrated gasification combined cycle (IGCC) with carbon capture and storage. The gasification plant is described in Subsection 10.1 of Martelli, Kreutz, Carbo, Consonni, and Jansen (2011), while the HRSC design methodology is described in Martelli, Amaldi, et al. (2011). The HRSC is optimized by maximizing the net plant electric efficiency while guaranteeing the energy and mass balance equations and satisfying a set of techno-economic constraints derived from the industrial practice which limits the capital cost of the equipment units. As in Martelli, Amaldi, et al. (2011), the HRSC design problem is decomposed into a bilevel program with onsmooth noisy constrained nonlinear program at the upper level, and a linear program at the lower level. In Martelli, Amaldi, et al. (2011), Martelli et al. (2012, 2013), the upper level problem is tackled with the Constrained Particle Swarm while the lower level problem is solved with the Simplex linear programming method.

In this section, we consider the challenging upper level nonlinear program as a black-box test problem. Compared to Martelli, Amaldi, et al. (2011), we do not only optimize the three evaporation pressures, but also the deaerator pressure, the superheat and reheat temperatures, and the three fuel mass flow rates for supplementary firing within the duct burners of the heat recovery steam generator (placed downstream of the three evaporators). Therefore, the resulting optimization problem contains nine bounded variables (deaerator, low, medium and high evaporation pressure, superheat and reheat temperature, fuel mass flow rates to the HRSG duct burners), a black-box objective function $f(\boldsymbol{x})$, which corresponds to the solver of the lower level linear problem, and four linear constraints. As pointed out by Martelli, Amaldi, et al. (2011), the objective function is noisy and characterized by non-differentiabilities as well as step type discontinuities, as shown in Fig. 19. Moreover, it is worth pointing out that the objective function is not defined wherever:

- the lower level linear program turns out to be infeasible
- the solution of the lower level linear program fails for some numerical reason (e.g., the specified number of iterations is not sufficient to reach convergence)
- the oxygen molar flow rate in the flue gases is not sufficient to guarantee the complete oxidization of the fuel injected by the duct burners

Table 6
Comparison between the best solutions found by our PGS-COM algorithms and the hybrid VNS-MADS algorithm of Audet et al. (2008a,b).

Description	Measure unit	VNS-MADS in Audet et al. (2008a,b)	PGS-COM
Outlet temperature in block HEATER	K	1100	1100
Length of reactor	m	16.9836	19.833572
Light key fraction in block SEP-STY		0.0968282	0.099737962
Light key fraction in block SEP-BZ		0.0001	0.000116484
Outlet pressure of block PUMP	atm	2.0000	2.006786
Split fraction in block SPLITTER		0.224742	0.27143411
Air excess fraction in block BOILER		1.96261	1.7294362
Cooling temperature of block COOLER	K	403.025	404.3994
Net Present Value (objective function value)	$	33,539,100	33,753,600

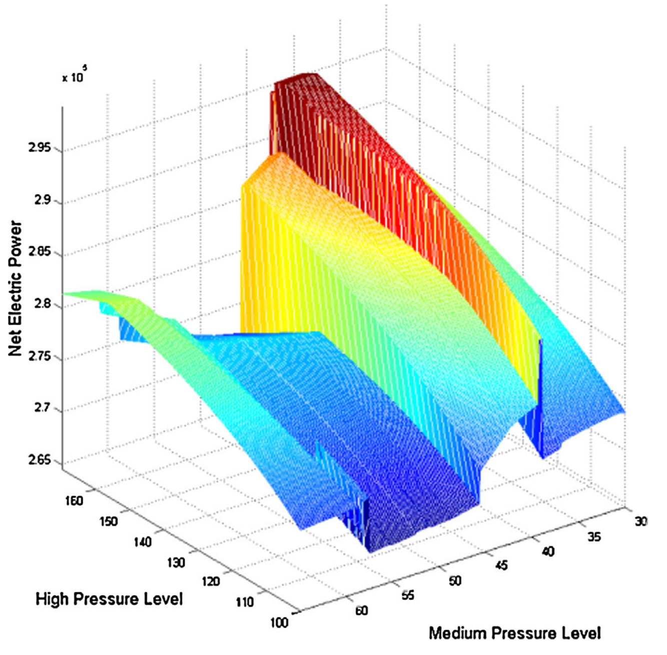

Fig. 19. Plot of the net electric power generated by the HRSC as a function of the evaporation pressures of the high and medium pressure level.

- the gas temperatures exceed the upper limit supported by the tube materials.

We have considered a maximum of 2500, 5000, 7500 and 10,000 function evaluations, and 20 runs for each test, as the for the styrene process problem. The total computing time required to carry out these tests amounts to about 20 days. For this problem, we executed PGS-COM, CPSO and GSS in parallel computing mode so as to take advantage of our multiple-core processor.

As far as average performance is concerned, PGS-COM outperforms the other methods returning solutions with an average value between 298,000 and 299,000 kW and worst value equal to about 296,000 kW (see Table 7). Note that also CPSO and PSwarm perform well in terms of average solutions. Concerning the quality of the best solutions, PGS-COM, CPSO and Complex provide slightly better solutions than PSwarm (301064 kW), CRS (300790 kW), eSS (300647 kW) and VNS-MADS (300747 kW). GSS appears to suffer from the premature convergence issue of the mesh size parameter described in Section 2.5, while the MADS algorithms show rather poor average performance and solution quality. Also in this application, like for the styrene process design problem, population-based algorithms seem to be more effective than the other ones.

Since in this test we optimize on 9 instead of 3 decision variables, it is not surprising that best solutions found by the algorithms have higher values than that (296360 kW) reported in Martelli, Amaldi, et al. (2011).

This application allows us to assess also the computational performance and the degree of parallelization achieved by PGS-COM. We have measured the average computational time required by PGS-COM, Complex and GSS to execute the runs with 10,000 function evaluations with parallel computing on our laptop, equipped with a 2.67 GHz Intel i7 four-core processor, and on a workstation, equipped with a 2.8 GHz Intel Xeon twelve-core processor, whose architecture is well suited for parallel computing. Complex does not take advantage of parallel computing, while GSS was coded for parallel computing by MathWorks (2013a). We defined four workers (see MathWorks, 2013b) on the laptop and eight on the workstation (the maximum number allowed by our Matlab version). The results are reported in Table 8.

Table 7
Comparative computational results for the optimization of a triple pressure level HRSC for an IGCC with CCS.

# f. eval.	Run	PSwarm	CPSO	Complex	GSS	LTMADS	OrthoMADS	CRS	SSm	eSS	VNS-MADS	PGS-COM
2500	Worst	295,176.7	295,357.6	284,902.7	283,054.3	283,829.4	280,395.4	289,711.2	286,198.2	292,105.8	288,607.5	290,163.0
	Average	296,429.4	296,660.7	292,433.0	290,657.5	292,865.7	292,895.9	293,055.2	292,689.8	294,151.9	291,982.7	297,710.9
	Best	301,365.6	300,792.0	299,797.2	296,324.3	297,837.2	297,848.0	298,551.2	296,253.0	298,778.9	296,394.0	301,402.9
5000	Worst	288,167.5	295,508.1	284,395.9	289,184.2	289,646.1	272,400.8	284,173.6	291,668.8	291,503.5	293,037.3	296,281.6
	Average	293,584.3	297,006.7	293,686.3	291,921.4	293,278.4	292,733.9	293,549.6	294,668.2	294,649.5	294,741.3	297,825.2
	Best	301,068.6	301,062.9	299,796.2	299,678.7	299,910.2	300,848.0	298,372.9	298,575.5	299,112.6	298,769.0	301,244.2
7500	Worst	295,156.3	295,511.0	282,108.1	282,886.2	288,772.5	288,833.3	285,944.6	291,692.9	291,839.5	293,669.4	296,226.3
	Average	296,487.8	297,373.7	294,902.1	295,907.7	294,827.3	294,756.5	294,563.7	295,664.6	296,843.3	294,793.3	299,272.9
	Best	299,893.1	301,601.9	301,639.1	299,679.0	299,848.0	298,808.7	299,356.0	298,142.9	300,647.3	300,376.6	301,648.4
10,000	Worst	296,043.0	294,826.5	288,390.8	295,506.8	280,316.9	288,835.1	289,712.1	291,687.0	291,837.4	290,669.4	296,211.7
	Average	296,497.9	297,181.8	294,393.8	296,063.5	294,536.7	295,413.6	295,021.2	296,820.8	296,241.7	295,581.0	298,773.7
	Best	301,064.9	301,628.5	301,648.7	299,679.0	299,550.6	299,947.0	300,790.1	299,551.0	299,678.7	300,747.1	301,647.5

Table 8
Comparison between the computational time of PGS-COM, GSS and Complex with and without parallel computing on two computers with different features.

	Laptop with 4 workers (Intel i7 2.67 GHz four-core processor)	Workstation with 8 workers (Intel Xeon 2.8 GHz 12-core processor)
Complex	3465 s	3282 s
GSS (no parallel computing)	3012 s	2905 s
GSS (with parallel computing)	1321 s	573 s
PGS-COM (no parallel computing)	2890 s	2795 s
PGS-COM (with parallel computing)	1372 s	475 s

On both computers PGS-COM takes about the same computational time as GSS, with and without parallel computing. When activating parallel computing, PGS-COM reduces its computational time by a factor of 2.10 on the four-core laptop, and by a factor of 5.9 on the eight-core workstation. As a comparison, GSS reduces its computational time by a factor of 2.3 on the laptop, and 5.1 on the workstation. These results highlights that on the HRSC design problem PGS-COM achieves a very good degree of parallelization, which is similar to that of the MathWorks' GSS.

9. Conclusions

In this work, we have proposed the hybrid direct-search algorithm PGS-COM, which successfully combines the positive features of Constrained Particle Swarm, Generating Set Search and Complex, to tackle black-box problems with non-differentiable and/or discontinuous objective function and unrelaxable as well as hidden constraints, arising for instance in flowsheet design optimization and total-site optimization problems. The performance of PGS-COM has been assessed on a selection of 25 test problems as well as two challenging process engineering applications, and compared to that of eleven well-known available direct-search methods.

The computational results obtained for 22 test problems with 4 up to 20 variables indicate that PGS-COM performs better than the other considered direct-search methods. For more than 7500 function evaluations, PGS-COM returns either optimal or very accurate best solution values for the largest fraction of the test problems, even for highly multimodal, non-differentiable, discontinuous objective functions subject to linear as well as nonlinear unrelaxable constraints. Below 7500 function evaluations, PGS-COM effectiveness is close to that of the best benchmark method (Complex) in terms of best solution quality. As far as the algorithm reliability is concerned, below 5000 function evaluations PGS-COM provides good quality average solution values for about the same fraction of problems as eSS, which performs best among the existing methods. Between 7500 and 60,000 function evaluations, PGS-COM performs even better than eSS, showing good average solution values for about 60% of the test problems. Only above 60,000 function evaluations PGS-COM is overtaken by SSm which provides good average solution values for a larger fraction of problems. Our computational results also indicate that the search effectiveness of PGS-COM holds up well on larger scale problems with 30 variables.

PGS-COM remarkable effectiveness and reliability become even clearer for the two challenging process engineering optimization problems. Indeed, in both applications it finds the highest quality best and average solution values. Since PGS-COM also yields good quality worst solution values over 20 runs, it appears to be particularly suitable for problems with computationally expensive function evaluations, like total-site optimization problems, for which the user typically runs the optimization only a few times. Another positive feature of the proposed algorithm is its good degree of parallelization. According to our tests with two different multi-core computers, the algorithm execution time and the degree of parallelization are essentially equal to those of GSS, the pattern search parallel code by MathWorks (2013a).

Our computational experiments also provide an interesting comparison between the eleven existing algorithms we used for benchmarking. Complex turns out to outperform the other methods in terms of best solution quality on most of the non-smooth and nonlinearly constrained test problems, but it shows a limited exploration capability and a limited robustness with respect to numerical noise in the objective function. CRS finds best solutions with close-to-optimal values for a large fraction of the problems, even with numerical noise, but quite poor average solution values when tackling real-world problems. Although LTMADS, OrthoMADS and VNS-MADS provide better quality solution values than GSS, they exhibit worse performance on most of the problems compared to Complex, CRS and CPSO. The hybrid Scatter Search algorithm eSS turns out to be particularly competitive in terms of average solution values and robustness toward numerical noise, but the solution quality is penalized by the lack of a local search method. Even though SSm yields slightly poorer solution values than eSS within a few thousands of function evaluations, it performs very well in terms of average (not the best) solution values for more than 60,000 function evaluations. Although CMAES is rather effective for non-smooth problems without unrelaxable constraints, its applicability to real-world engineering problems with unrelaxable/hidden constraints is limited because it may fail to generate the set of starting solutions. Finally CPSO, which does not perform that well on the 25 test problems, provides the best solutions (both average and best values) among the existing methods for the two engineering applications, showing robustness toward unrelaxable/hidden constraints as well as numerical noise and discontinuities. To conclude, the computational results summarized in this article indicate that PGS-COM is a very robust global optimization method for non-smooth black-box optimization problems arising, for instance, in simulation-based process design as well as utility (e.g., steam cycles and steam networks) and total-plant design/synthesis. Compared to Complex, which is currently included in well-known sequential process simulation codes (e.g., Aspen Plus® and Hysys®), PGS-COM is advantageous not only in terms of performance and robustness but also from the computational point of view, since the function evaluations can be parallelized.

Acknowledgements

The authors acknowledge Paolo Moschini and Marco Gabba, researchers at LEAP (Laboratorio Energia Ambiente Piacenza), for providing support on the information technology issues, as well as Jonathan Currie, researcher and lecturer at Auckland University of Technology, for prompt help with the OPTI toolbox. Last but not least, the authors kindly thank the anonymous reviewers for the useful suggestions that helped to improve the paper, as well as all the developers of the codes CMAES, NOMAD, NLopt, the SSmGO toolbox, the OPTI toolbox, the Matlab Global Optimization toolbox.

Appendix A. Supplementary data

Supplementary material related to this article can be found in the online version.

References

Abramson, M. A. (2007). *NOMADm version 4.5.* Wright-Patterson AFB, OH: Air Force Institute of Technologies.

Abramson, M. A., Audet, C., Dennis, J. E., & Le Digabel, S. (2009). OrthoMADS: A deterministic MADS instance with orthogonal directions. *SIAM Journal on Optimization*, *20*(2), 948–966.

Adam, T. A., & Seider, W. (2008). Practical optimization of complex chemical processes with tight constraints. *Computers and Chemical Engineering*, *32*, 2099–2112.

Adelman, A., & Stevens, W. F. (1972). Process optimization by the "Complex" method. *AIChE Journal*, *18*(1), 20.

Andersson, J. (2001). *Multiobjective optimization in engineering design: Application to fluid power systems* (Ph.D. thesis). Sweden: Department of Mechanical Engineering, Linköpings universitet., ISBN 91-7219-943-1

Astolfi, M., Romano, M. C., Bombarda, P., & Macchi, E. (2013). Binary ORC power plants for the exploitation of medium-low temperature geothermal sources – Part B: Techno-economic optimization. *Energy*, http://dx.doi.org/10.1016/j.energy.2013.11.057 (in press)

Audet, C., & Dennis, J. E. (2003). Analysis of generalized pattern searches. *SIAM Journal on Optimization*, *13*, 89–903.

Audet, C., & Dennis, J. E. (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, *17*(2), 188–217.

Audet, C., Bechard, V., & Chaouki, J. (2008). Spent potliner treatment process optimization using a MADS algorithm. *Optimization and Engineering*, *9*, 143–160.

Audet, C., Bechard, V., & Le Digabel, S. (2008). Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, *41*, 299–318.

Audet, C., Dennis, J. E., & Le Digabel, S. (2010). Globalization strategies for mesh adaptive direct search. *Computational Optimization and Applications*, *46*, 193–215.

Banga, J. R., & Seider, W. D. (1996). *Global optimization of chemical processes using stochastic algorithms. State of the art in global optimization: Computational methods and applications* (vol. 7).

Biegler, L. T. (2010). *Nonlinear programming: Concepts, algorithms and applications to chemical processes*. *MOS-SIAM Series on Optimization* , ISBN 978-0-898717-02-0.

Biegler, L. T., & Cuthrell, J. E. (1985). Improved infeasible path optimization for sequential modular simulators – II: The optimization algorithm. *Computers and Chemical Engineering*, *9*(3), 257–267.

Biegler, L. T., Grossmann, I. E., & Westerberg, A. W. (1997). *Systematic methods of chemical process design*. Upper Saddle River, New Jersey: Prentice Hall., ISBN 0-13-492422-3.

Booker, A. J., Dennis, J. R., Frank, P. D., Moore, D. W., & Serafini, D. B. (1998). Managing surrogate objectives to optimize a helicopter rotor design – Further experiments. In *AIAA paper 1998, 8th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*.

Box, M. J. (1965). A new method of constrained optimization and a comparison with other methods. *Computer Journal*, *8*(1), 42–52.

Brooks, S. H. (1958). A discussion of random methods for seeking minima. *Operations Research*, *6*, 244–261.

Buzzi-Ferraris, G., & Manenti, F. (2010). A combination of parallel computing and object-oriented programming to improve optimizer robustness and efficiency. In *20th European symposium on computer aided process engineering, ESCAPE* (p. 20).

Carter, R., Gablonsky, J., Patrick, A., Kelley, C., & Eslinger, O. (2001). Algorithms for noisy problems in gas transmission pipeline optimization. *Optimization and Engineering*, *2*, 139–157.

Chiesa, P., Consonni, S., Kreutz, T., & Williams, R. (2005). Co-production of hydrogen, electricity, and CO_2 from coal with commercially ready technology. Part A: Performance and emissions. *International Journal of Hydrogen Energy*, *30*, 747–767.

Chiesa, P., & Lozza, G. (2005). Using hydrogen as a Gas Turbine Fuel. *Journal of Engineering for Gas Turbines and Power*, *127*(1), 73–81.

Chootinan, P., & Chen, A. (2006). Constraint handling in genetic algorithms using a gradient-based repair method. *Computers and Operations Research*, *33*(8), 2263–2281.

Colmenares, T. R., & Seider, W. D. (1989). Synthesis of utility systems integrated with chemical processes. *Industrial and Engineering Chemistry Research*, *28*, 84–93.

Conn, A. R., Scheinberg, K., & Vicente, L. N. (2009). *Introduction to derivative free optimization*. *MPS-SIAM series on optimization*. ISBN 978-0-898716-68-9.

Consonni, S. (1992). *Performance prediction of gas/steam cycles for power generation* (Ph.D. thesis). Princeton, NJ: Princeton University.

Cramer, E. J., Dennis, J. R., Frank, P. D., Lewis, R. M., & Shubin, J. R. (1994). Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, *4*(4), 754–776.

Currie, J., & Wilson, D. I. (2012). OPTI: Lowering the barrier between open source optimizers and the industrial MATLAB user. In *Foundations of computer-aided process operations* Georgia, USA.

Custódio, A. L., Dennis, J. E., Jr., & Vicente, L. N. (2008). Using simplex gradients of nonsmooth functions in direct search methods. *IMA Journal of Numerical Analysis*, *28*, 770–784.

De Servi, C., Tizzanini, A., Pietra, C., & Campanari, S. (2013). Enhancement of the electric efficiency of commercial fuel cell units by means of an organic rankine cycle: A case study. *Journal of Engineering for Gas Turbines and Power*, http://dx.doi.org/10.1115/1.4023119

Dong, Y., Tang, J. F., Xu, B. D., & Wang, D. (2005). An application of swarm optimization to nonlinear programming. *Computers and Mathematics with Applications*, *49*(11–12), 1655–1668.

Drazi'c, M., Lavor, C., Maculan, N., & Mladenovi'c, N. (2004). A continuous VNS heuristic for finding the tridimensional structure of a molecule. In *Technical Report G-2004–22, Les Cahiers du GERAD, Montréal*.

Egea, J. A., Marti, R., & Banga, J. R. (2010). An evolutionary method for complex-process optimization. *Computers and Operations Research*, *37*, 315–324.

Egea, J. A., Rodrigues-Fernandez, M., Banga, J. R., & Marti, R. (2007). Scatter search for chemical and bio-process optimization. *Journal of Global Optimization*, *37*, 481–503.

Fan, S. S., & Zahara, E. (2007). A hybrid simplex search and particle swarm optimization for unconstrained optimization. *European Journal of Operational Research*, *181*, 527–548.

Finkel, D. E. (2003). *DIRECT optimization algorithm user guide*. Downloaded from: www4.ncsu.edu/Direct/DirectUserGuide_pdf.pdf

Friedman, P., & Pinder, K. L. (1972). Optimization of a simulation model of a chemical plant. *Industrial & Engineering Chemistry Process Design and Development*, *11*(4), 512.

Fu, Q., & Tong, N. (2010). A complex-method-based PSO algorithm for the maximum power point tracking in photovoltaic system. In *Second IEEE international conference on information technology and computer science* (pp. 134–137).

Gablonsky, J. (2001). *Modifications of the direct algorithm* (PhD thesis). North Carolina State University.

Gaines, L. D., & Gaddy, J. L. (1976). Process optimization by flow sheet simulation. *Industrial and Engineering Chemistry Process Design and Development*, *15*(1), 206–211.

Gassner, M., & Marechal, F. (2009). Methodology for the optimal thermo-economic, multi-objective design of thermochemical fuel production from biomass. *Computers and Chemical Engineering*, *33*, 769–781.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Science*, *8*(1), 156–166.

Glover, F., Laguna, M., & Martì, R. (2004). *Scatter search and path relinking: Foundations and advanced designs. New optimization techniques in engineering*. Berlin Heidelberg: Springer-Verlag.

Goulcher, R., & Casares, L. (1978). The solution of steady-state chemical engineering optimisation problems using a random-search algorithm. *Computers and Chemical Engineering*, *2*(1), 33–36.

Gross, B., & Roosen, P. (1998). Total process optimization in chemical engineering with evolutionary algorithms. *Computers and Chemical Engineering*, *22*, S229–S236.

Guin, J. A. (1968). Modification of the complex method of constraint optimization. *Computer Journal*, *10*, 416–417.

Han, J., Kokkolaras, M., & Papalambros, P. Y. (2008). Optimal design of hybrid fuel cell vehicles. *Journal of Fuel Cell Science and Technology*, *5*, 1–8.

Hansen, N., Muller, S. D., & Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, *11*(1), 1–18.

Hansen, P., & Mladenovi'c, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operations Research*, *130*(3), 449–467.

Hedar, A. R., & Fukushima, M. (2004). Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, *35*(4), 521–549.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Hooke, R., & Jeeves, T. A. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association of Computing Machinery*, *8*, 212–229.

Hu, X., & Eberhart, R. (2002). Solving constrained nonlinear optimization problems with particle swarm optimization. In *Proceedings of the 6th world multiconference on systematics, cybernetics and informatics* Orlando, USA.

Hu, X., Eberhart, R., & Shi, Y. (2003). Engineering optimization with particle swarm. In *Proceedings of the IEEE swarm intelligence symposium* Indianapolis, USA, (pp. 53–57).

Jones, D., Perttunen, C., & Stuckman, B. (1993). Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application*, *79*, 157–181.

Johnson, S. G. (2013). *The NLopt nonlinear-optimization package*. http://ab-initio.mit.edu/nlo

Kaelo, P., & Ali, M. M. (2006). Some variants of the controlled random search algorithm for global optimization. *Journal of Optimization Theory and Application*, *130*(2), 253–264.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (pp. 1942–1948).

Khan, K. A., Yunt, M., & Barton, P. I. (2011). Global optimization of nonsmooth dynamic systems. In *Proceedings of the 2011 AIChE annual meeting*.

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

Kolda, T. G., Lewis, R. M., & Torczon, V. (2006a). Stationarity results for generating set search for linearly constrained optimization. *SIAM Journal on Optimization*, *17*, 943–968.

Kolda, T. G., Lewis, R. M., & Torczon, V. (2006b). A generating set direct search augmented Lagrangian algorithm for optimization with a combination of general and linear constraints. In *SANDIA Report SAND2006-5315*.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Journal of Evolutionary Computation*, *7*(1), 19–44.

Laguna, M., & Martì, R. (2003). *Scatter search: Methodology and implementations in C*. Boston: Kluwer Academic Publishers.

Le Digabel, S. (2011). Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, *37*(4), 44:1–44:15.

Lewis, R. M., & Torczon, V. (1999). Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, *9*, 1082–1099.

Lewis, R. M., & Torczon, V. (2000). Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, *10*(3), 917–941.

Lewis, R. M., & Torczon, V. (2002). A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, *12*(4), 1075–1089.

Lewis, R. M., Shepherd, A., & Torczon, V. (2007). Implementing generating set search methods for linearly constrained minimization. *SIAM Journal of Scientific Computation*, *29*(6), 2507–2530.

Luksan, L., & Vicek, J. (2000). *Test problems for non-smooth unconstrained and linearly constrained optimization*. Technical Report, Institute of Computer Science of the Czech Republic.

Luus, R., & Jaakola, T. H. (1973a). A direct approach to optimization of a complex system. *AIChE Journal*, *19*(3), 645–646.

Luus, R., & Jaakola, T. H. (1973b). Optimization by direct search and systematic reduction of the size of search region. *AIChE Journal*, *19*(4), 760–766.

Luus, R., & Jaakola, T. H. (1974). A note on the application of nonlinear programming to chemical-process optimization. *Operations Research*, *22*(2), 415–417.

Mallipeddi, R., & Suganthan, P. N. (2010). *Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization*. Technical Report. Available at: http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC10-Const/TR-April-2010.pdf

Martelli, E. (2010, May). In E. Martelli (Ed.), *Optimal design of heat recovery steam cycles and heat integration* , ISBN 978-3-8383-5884-0. Lambert Academic Publishing.

Martelli, E., Kreutz, T., Carbo, M., Consonni, S., & Jansen, D. (2011). Shell coal IGCCS with carbon capture: conventional gas quench vs. innovative configurations. *Applied Energy*, *88*(11), 3978–3989.

Martelli, E., Amaldi, E., & Consonni, S. (2011). Numerical optimization of heat recovery steam cycles: Mathematical model, two-stage algorithm and applications. *Computers and Chemical Engineering*, *35*(12), 2799–2823.

Martelli, E., Nord, L., & Bolland, O. (2012). Design criteria and optimization of heat recovery steam cycles for integrated reforming combined cycles with CO_2 capture. *Applied Energy*, *92*, 255–268.

Martelli, E., Kreutz, T., Gatti, M., Chiesa, P., & Consonni, S. (2013). Numerical optimization of steam cycles and steam generators designs for a coal to FT plant. *Chemical Engineering Research and Design*, *91*(2013), 1467–1482.

MathWorks. (2013a). *Global optimization toolbox*. http://www.mathworks.it/it/products/global-optimization/

MathWorks. (2013b). *Parallel computing toolbox*. http://www.mathworks.it/products/parallel-computing/

Meza, J. C., & Martinez, M. L. (1994). On the use of direct search methods for the molecular conformation problem. *Journal of Computational Chemistry*, *15*, 627–632.

Michalewicz, Z., & Nazhiyath, G. (1995). Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints. In *Proceedings of the second IEEE international conference on evolutionary computation* (pp. 647–651). IEEE Press.

Moles, C., Mendes, P., & Banga, J. (2003). Parameter estimation in biochemical pathways: A comparison of global optimization methods. *Genome Research*, *13*, 2467–2474.

Moré, J. J., & Wild, S. W. (2009). Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, *20*(1), 172–191.

Morin, A., Wahl, P. E., & Molnvik, M. (2011). Using evolutionary search to optimise the energy consumption for natural gas liquefaction. *Chemical Engineering Research and Design*, *89*, 2428–2441.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, *7*, 308–313.

Price, W. L. (1983). Global optimization by controlled random search. *Journal of Optimization Theory and Application*, *40*(3).

Rangaiah, G. P. (1982). Effect of dimension on direct search methods for constrained optimization. *Computers and Chemical Engineering*, *9*(4), 405–406.

Richtárik, P. (2009). *Improved algorithms for convex minimization in relative scale*. Technical Report.

Rios, L., & Sahinidis, N. (2013). Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, *56*(3), 1247–1293.

Rodrigues-Fernandez, Egea, J. A., & Banga, M. (2006). Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, *7*, 483–501.

Rodríguez, N., Mussati, S., & Scenna, N. (2011). Optimization of post-combustion CO_2 process using DEA–MDEA mixtures. *Chemical Engineering Research and Design*, *89*, 1763–1773.

Romeijn, H. E., & Smith, R. L. (1994). Simulated annealing for constrained global optimization. *Journal of Global Optimization*, *5*(2), 101–126.

Santarelli, M., & Pellegrino, D. (2005). Mathematical optimization of a RES-H2 plant using a black box algorithm. *Renewable Energy*, *30*, 493–510.

Schittkowski, K. (2002). *Numerical data fitting in dynamical systems – A practical introduction with applications and software*. Dordrecht, the Netherlands: Kluwer Academic Publishers.

Shiou-Shan Chen. (2006). Styrene. In *Kirk-Othmer encyclopedia of chemical technology*. New York: John Wiley and Sons.

Smith, R. L. (1984). Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, *32*, 1296–1308.

Spaans, R., & Luus, R. (1992). Importance of search-domain reduction in random optimization. *Journal of Optimization Theory and Applications*, *75*(3), 635–638.

Stein, M. (1987). Large sample properties of simulations using latin hypercube sampling. *Technometrics*, *29*(2), 143–151.

Takahama, T., & Sakai, S. (2006). Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In *Proceedings of the second IEEE international conference on evolutionary computation*.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, *7*, 1–25.

Vaz, A. I. F., & Vicente, L. N. (2007). A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, *39*, 197–219.

Vaz, A. I. F., & Vicente, L. N. (2009). PSwarm: A hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods and Software*, *24*, 669–685.

Vicente, L. N., & Custodio, A. L. (2012). Analysis of direct searches for discontinuous functions. *Mathematical Programming*, *133*, 299–325.

Wah, B. W., & Wang, T. (1999). Constrained simulated annealing with applications in nonlinear continuous constrained global optimization. *Proceedings of the IEEE international conference on tools with artificial intelligence*.

Yu, W. C. (1979). Positive basis and a class of direct search techniques. *Scientia Sinica, Special issue of Mathematics*, *1*, 53–67.

Zabinsky, Z. B., Smith, R. L., McDonald, J. F., Romeijn, H. E., & Kaufman, D. E. (1993). Improving hit and run for global optimization. *Journal of Global Optimization*, *3*, 171–192.

Zahara, E., & Hu, C. H. (2008). Solving constrained optimization problems with hybrid particle swarm optimization. *Engineering Optimization*, *40*(11), 1031–1048.

Zhang, Y., & Gao, L. (2003). On numerical solution of the maximum volume ellipsoid problem. *SIAM Journal on Optimization*, *14*, 53–76.