To appear in *Engineering Optimization*
Vol. 00, No. 00, Month 20XX, 1–19

Sequential Linear Programming and Particle Swarm Optimization for the Optimization of Energy Districts

E. Riccietti [a*] and S. Bellavia[b] and S. Sello[c]

[a]*Dipartimento di Matematica e Informatica "Ulisse Dini", Università di Firenze, viale G.B. Morgagni 67a, 50134 Firenze, Italia.*; [b]*Dipartimento di Ingegneria Industriale, Università di Firenze, viale G.B. Morgagni 40, 50134 Firenze, Italia, stefania.bellavia@unifi.it*; [c]*Enel Ingegneria e Ricerca, Via Andrea Pisano 120, 56122 Pisa, Italia, stefano.sello@enel.com*

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

This article deals with the optimization of energy resources management of industrial districts, with the aim of minimizing the customer energy expenses. A modelling of the district is employed, whose optimization gives rise to a nonlinear constrained optimization problem. Here the focus is on its numerical solution. Two different methods are considered: a Sequential Linear Programming (SLP) and a Particle Swarm Optimization (PSO) method. Efficient implementations of both approaches are devised and the results of the tests performed on several energetic districts are reported, including a real case study.

Keywords: Nonlinear Optimization Problems; Sequential Linear Programming; Particle Swarm Optimization; Energy Districts; Penalty Functions

1. Introduction

In this article the numerical solution of optimization problems arising in energy districts is investigated.

The electric system has been experiencing a dramatic evolution in the last years, due to the growing use of unpredictable renewable energy sources and of the spread of distributed generation. On top of that, customers are increasing their attention towards a smarter approach to energy consumption, and paying more and more attention to convenient energy tariff schemes, choosing among different retailers. The problem of efficiently optimizing the resources management of a district is then considered, with the aim of providing the cheapest solution to the customers.

This problem may be addressed in different ways. A possibility is to formulate it as a mixed integer nonlinear programming (MINLP) problem, that can be addressed by adopting a proper linearization yielding a mixed integer linear programming (MILP) problem (Salgado and Pedrero 2008; Bischi et al. 2014). This in turn can be solved by one of the many commercial solvers available. In some cases the problem can even be directly formulated as a (MILP) (Schiefelbein et al. 2015). Another possibility is to formulate it as a multi-objective optimization problem (Ascione et al. 2016; Wang, Martinac, and Magny 2015), when the cost function is not the only objective that has to be minimized.

The research centre 'Enel Ingegneria e Ricerca' in Pisa, in collaboration with the Department of Civil and Industrial Engineering of the University of Pisa, addressed the

*Corresponding author. Email: elisa.riccietti@unifi.it

problem of optimizing an energy district adopting an alternative approach. The problem has been formulated as a nonlinear programming problem (NLP). A software package has been developed (Pannocchia and Mancuso 2014), which takes as an input information like energy market prices, specifications of the machines, current and forecast weather information, and builds a model of the energy district. It also computes the objective cost function (measuring the costs related to the exchange of energy with the network) and the functions modelling the physical constraints on the machines' variables. The aim of the optimization process is to find, one day ahead, the machines' parameters setting providing the optimal dispatching of local energy resources for the following day, in order to minimize the cost of energy at customer site (Ferrara, Riccardi, and Sello 2014).

Here the focus is on the solution of the arising nonlinear minimization problem. In solving such a problem a compromise between two needs should be found. On one hand one aims at finding the machine asset corresponding to the lowest possible value of the objective function. On the other hand the optimization process should be quick, as it needs to be performed several times in a day, when the needs of the district or the weather conditions affecting the renewable resources change.

The objective is a nonconvex multimodal function with hundreds of variables. It does not have a simple analytical expression and it is expensive to evaluate, as it is a sum over a large number of terms. Also, the function features some discontinuities. Thus, derivative free methods are particularly attractive. As an alternative, methods using derivatives can be used, if the objective function is approximated around discontinuities by a smooth surrogate function. While the function is still exactly evaluated, the derivatives of the surrogate function are employed. Just first order methods can be used anyway as, due to such an approximation, one cannot rely on second order derivatives. Also, the analytical form of just some components of the gradient is available and the remaining ones have to be approximated, for example via finite differences. Then, approximating also the second derivatives would be computationally too heavy anyway.

Having this in mind, it was decided to compare a Sequential Linear Programming (SLP) and a Particle Swarm Optimization (PSO). SLP is a classical optimization method. It is a first order iterative procedure that builds approximations to the solution of the original nonlinear problem by considering a sequence of linear subproblems approximating it (Fletcher and de la Maza 1989; Byrd et al. 2003). The generated sequence usually converges to a local minimum close to the starting guess. PSO (Hu and Eberhart 2002; Lu and Chen 2008) is a derivative free method, part of the family of more recently developed metaheuristic algorithms, which aim at finding a global minimum of an optimization problem. This class comprises also Genetic Algorithms (Tessema and Yen 2009), Cultural Algorithms (Jin and Reynolds 1999), Differential Evolution (Mezura-Montes, Miranda-Varela, and del Carmen Gómez-Ramón 2010). The literature amount on PSO methods is huge. Many variants of the original approach have been proposed to solve successfully engineering problems. Just to cite a few, in De-los Cobos-Silva et al. (2018); Zhang and Xie (2003); Liu, Cai, and Wang (2010) the basic scheme has been improved by integrating it with Differential Evolution, in Mazhoud et al. (2013) a new constraint-handling mechanism (based on a closeness evaluation of the solutions to the feasible region) is proposed and in Cagnina, Esquivel, and Coello (2011) a double population and a special shake mechanism to avoid premature convergence are used. Many strategies have also been proposed, that consist in using PSO as a global search method and coupling it with a fast local search algorithm, to speed up its convergence (Fu and Tong 2010; Fan and Zahara 2007; Vaz and Vicente 2009; Martelli and Amaldi 2014).

The aim of this work is to investigate and compare the performance of SLP and PSO methods on this kind of problems. Specifically, the study is intended to understand if it is worth the use of a method such PSO, that aims at returning a solution close to the global minimum but is expected to converge more slowly than SLP, which is a first order method.

Efficient and reliable implementations of the two procedures are also provided, specially designed for the problem at hand. The adopted SLP method is close to the one presented in (Robinson 1972), but is equipped with a trust-region approach to promote global convergence (Nocedal and Wright 2006). The PSO method is based on the constraint handling strategy introduced in Michalewicz and Attia (1994) and on a new strategy to prevent stagnation in local minima, proposed here.

The two solvers have been inserted in the software package developed by Enel centre and University of Pisa, and tested on many different realistic examples of energy districts, and on a real energy district provided by Enel. Preliminary analysis on this topic has been conducted by the authors in Riccietti, Bellavia, and Sello (2017), where the problem under study is introduced and preliminary numerical results are given. In this article the numerical solution of the arising optimization problem is deeply analysed.

The article is organized as follows: in Section 2 the district modelling is introduced, the machines that are part of the district are presented, focusing on the description of the optimization variables. In Section 3 the arising optimization problem is stated. In Sections 4 and 5 the SLP and PSO procedures implemented for the specific problem are respectively described. The convergence analysis for SLP is reported in Section 2 of the supplemental material. Finally, in Section 6 numerical results are presented. The two solvers are applied to examples of energetic districts and the results of the tests performed are shown. Further numerical results are reported in Section 1 of the supplemental material, where the two methods are applied to a well-known set of benchmark functions, usually employed to test performance of metaheuristic methods (Liang et al. 2005). The aim is to provide a comparison of the implemented methods with the state-of-the-art of metaheuristic methods.

2. The District Model

The optimization problem to be solved arises form the district modelling described below, that has been designed in Pannocchia and Mancuso (2014). The user has to specify which devices the district is compound of and the characteristic parameters of each of them, to simulate their real behaviour. The variables that need to be optimized are the physical parameters of the machines that describe their functioning, such as the electrical power produced by generators, that stored by batteries, the thermal power provided by boilers. The decision variables chosen are all continuous (Pannocchia and Mancuso 2014), as it is described in the following subsections, so that the arising problem is a NLP. The decision variables are dimensionless variables in $[0, 1]$ or $[-1, 1]$, obtained scaling the quantities under control with respect to their maximum value.

The aim of the optimization process is to individuate the optimal generation, load and energy storage profile for each device. A plan of the decision variables for each machine is built for the following day. The aim is to minimize the expenses related to the district management, taking into account real time informations such as the price of the electricity (to sell or to buy), wind speed, solar radiation and ambient temperature. The time unit is set to be $\tau = 15$ minutes, so that the day is divided into $N = \frac{24\times 60}{15} = 96$ quarters of an hour. The solver has to find the optimal decision variables for each device and for each time unit. Let i be the time index, $i = 1, \dots, N = 96$.

In the next subsections the four different types of machines that can be included as part of an energy district are described. They are: electrical generators, accumulators, electrical loads, thermal configurations (i.e. groups of heat/cold generation units, whose components are predefined) (Ferrara, Riccardi, and Sello 2014).

2.1 *Electrical generators and thermal configurations*

Thermal configurations are groups of heat/cold generation units, like cogeneration engines (or CHP, Combined Heat and Power), boilers, chillers. Examples of thermal configurations may be a CHP, a boiler and a heat pump connected in parallel or an absorption refrigerator connected to a CHP and an electric chiller. The user can choose among 12 different predefined thermal configurations. Electrical generators comprise photovoltaic, wind turbine and fuel burning generators (Pannocchia and Mancuso 2014).

Let N_{gen} denote the total number of generators and thermal configurations. For the k-th device $\alpha_k \in \mathrm{R}^N$ denotes the decision variables vector, $k = 1 \ldots, N_{gen}$. Its component $\alpha_k(i)$ is the load at i-th time unit:

$$\alpha_k(i) = \frac{P_k(i)}{PN_k}, \qquad 0 \le \alpha_k(i) \le 1, \qquad i = 1, \ldots, N, \tag{1}$$

namely the ratio between the actual power output and the maximum power output of the machine. The decision variables are then real, and used to decide whether the unit is on or off through the computation of the unit status. The unit status is a discontinuous function θ_k, defined as

$$\theta_k(x) = \begin{cases} 1 & \text{if } x \ge \alpha_{\min,k} \\ 0 & \text{if } x < \alpha_{\min,k} \end{cases}, \tag{2}$$

where $\alpha_{\min,k}$ is the minimum operating load for unit k. The cost functions for generators and thermal configurations depend on the unit status at each time instant i, that is evaluated as $\theta_k(\alpha_k(i))$. As a consequence, they feature some discontinuities.

For fuel burning generators and thermal configurations with CHP a constraint on the number of ignitions $NI(\alpha_k)$ is provided for. That is an integer number counting how many times unit k is turned on within a day. The number of ignitions must be bounded above, so that the following constraint arises:

$$NI(\alpha_k) = \frac{1}{2}\sum_{i=1}^{N} |\theta_k(\alpha_k(i)) - \theta_k(\alpha_k(i-1))| \le NI_{\max}. \tag{3}$$

Note that values of $NI(\alpha_k)$ are integers, but $NI(\alpha_k)$ is not a variable of the problem, subject to optimization, the decision variable is α_k that is real.

2.2 *Electrical accumulators*

Let $\beta_b \in \mathrm{R}^N$ be the decision variables vector for the b-th accumulator, $b = 1, \ldots, N_{acc}$ with N_{acc} the total number of accumulators in the district. Its components $\beta_b(i)$ are the decision variables at the i-th time unit, $i = 1, \ldots, N$:

$$\beta_b(i) = -\frac{P_b(i)}{PB_b}, \qquad -1 \le \beta_b(i) \le 1, \qquad i = 1, \ldots, N, \tag{4}$$

where $P_b(i)$ is the power either supplied or absorbed by accumulator b at time i and PB_b is the rated power output of the b-th battery (Ferrara, Riccardi, and Sello 2014). Negative values of $\beta_b(i)$ means that the accumulator is providing power, positive values of $\beta_b(i)$ means that the accumulator is storing power. For each accumulator and for each time unit there are also two physical restrictions on the state of charge SOC_b:

$$SOC_{b\min} \leq SOC_b(i) \leq SOC_{b\max}, \qquad i = 1, \ldots, N. \tag{5}$$

SOC_b is a piecewise linear function of β_b, as it depends on the efficiency of the accumulator η_b, which takes two different values depending on the sign of β_b (charge and discharge mode):

$$SOC_b(i) = c_1 + c_2 \eta_b(i) \beta_b(i), \tag{6}$$

where c_1 is a constant depending on the state of charge of the previous time instant and on the daily auto-discharge of the accumulator, and c_2 is a constant depending on the nominal power and the capacity of the accumulator. SOC_b is then non-differentiable when β_b changes sign.

2.3 *Electrical loads*

Three different types of electrical loads are considered: *L1 loads*, that are mandatory electrical consumptions, *L2 loads*, that are electrical cycles that need to be completed one or more times at no specific time in the day, *L3 loads*, that are normally on, and can be shut down for a limited amount of time without compromising the related process operation. L1 loads are given parameters for each time unit, so these loads do not have decision variables associated and are not included in the count of total number of loads that is $N_{loads} = N_{L2} + N_{L3}$ where N_{L2} and N_{L3} are total numbers of L2 and L3 loads respectively. For the loads the decision variables are associated to the indexes of starting times of cycles (for L2 loads) and to indexes of switch-off and switch-on times (for L3 loads). These should be integer variables. In order to avoid working with a MINLP, in Enel package scalar continuous variables in $(0, 1]$ are considered as decision variables and they are then related to the integer quantities one actually wants to control (Pannocchia and Mancuso 2014). Let γ_m denote the vector of decision variables for m-th (L2 or L3) load. For L2 loads the decision variables are given by $\gamma_m \in \mathrm{R}^{N_m}$, where N_m is the number of cycles that need to be completed by m-th L2 load. γ_m is used to compute the starting time i_l of l-th cycle as: $i_l = \lceil \gamma_m(l) N \rceil$, $l = 1, \ldots, N_m$, so that $i_l \in \{1, 2, \ldots, N\}$. For m-th L3 load the vector of decision variables is $\gamma_m \in \mathrm{R}^{2NI_m}$, with NI_m maximum number of times that the electric load is activated. The odd components of γ_m are related to switch-off times s_l and the even ones to switch-on times a_l, for $l = 1, 2, \ldots, NI_m$:

$$s_l = \lceil \gamma_m(2l-1) N \rceil, \qquad\qquad a_l = \lceil \gamma_m(2l) N \rceil.$$

On the loads decision variables there are the following bound constraints:

$$\frac{1}{N} \leq \gamma_m(l) \leq 1,\ l = 1, \ldots, N_m, \qquad\qquad \frac{1}{N} \leq \gamma_m(j) \leq 1,\ j = 1, \ldots, 2NI_m, \tag{7}$$

and also some physical nonlinear constraints. For L2 loads one constraint ensures that the time between the starting points of two successive cycles is enough to complete the first of them. A second constraint ensures that the amount of time between the beginning of the last cycle and the end of the day is enough to complete the cycle. For L3 loads they guarantee three conditions. First, that the switch-off of the load precedes the switch-on. Second, that the load is not off for more than a given amount of time. Third, that if a load is off and is switched on, a suitable amount of time passes until it is switched off again (Ferrara, Riccardi, and Sello 2014).

It is worth underlining that an approximation is introduced in the problem when real variables are considered rather than integer ones. However, taking into account that the time unit is rather small, and that Enel is planning to significantly further reduce it in future work, it was preferred to allow such approximation, rather than that working with a MINLP.

3. Arising Optimization Problem

The objective cost function f represents the overall daily district expense for exchanging electricity with the network (Pannocchia and Mancuso 2014). If $\bar{f}_i$ denotes the cost at each time instant i, the objective function results to be:

$$f(x) = \sum_{i=1}^{N} \bar{f}_i(x).$$

Each $\bar{f}_i$ can be expressed as $\bar{f}_i = c_i f_i$, where c_i is the prize of electricity at instant i and f_i is the net electrical power exchanged between the district and the network at instant i, which is positive if the district is selling electricity to network or negative if the district is buying it. The exchange of power with the network can be expressed as the sum of all the partial exchanges of district devices for the i-th time unit:

$$f_i(x) = \sum_{k=1}^{N_{gen}} f_{i,k}(x) - \sum_{m=1}^{N_{loads}} f_{i,m}(x) + \sum_{b=1}^{N_{acc}} f_{i,b}(x),$$

where $f_{i,k}(x)$ is the power generated by the k-th generator, $f_{i,m}(x)$ is the power absorbed by the m-th electrical load, $f_{i,b}(x)$ is the power released by the b-th accumulator, which is negative when the accumulator is charged (Pannocchia and Mancuso 2014).

If n is the problem dimension, with

$$n = N[N_{gen} + N_{acc}] + \sum_{m=1}^{N_{L2}} N_m + \sum_{m=1}^{N_{L3}} NI_m,$$

then $f : \mathrm{R}^n \to \mathrm{R}$ is a nonlinear, nonconvex multimodal function, that features some discontinuities. As discussed in the previous section, a number of devices have physical constraints on their decision variables (see equations (3), (5)). The constraints are nonlinear and relaxable (minor violations are allowed). They can be stacked together obtaining a vector of constraints $g : \mathrm{R}^n \to \mathrm{R}^p$, with p total number of constraints:

$$g(x) = \left[g_1(x), \ldots, g_{N_{gen}}(x), \ldots, g_{N_{gen}+N_{acc}}(x), , \ldots, g_{N_{gen}+N_{acc}+N_{loads}}(x)\right]^T,$$

where $g_j(x)$ for $j = 1, \ldots, N_{gen} + N_{acc} + N_{loads}$ is a vector containing the constraints on the j-th device. The resulting optimization problem is the following nonlinear constrained problem:

$$\min_x \; f(x), \tag{8a}$$

$$x_{\min} \leq x \leq x_{\max}, \tag{8b}$$

$$g(x) \leq 0, \tag{8c}$$

where $x \in \mathrm{R}^n$ is the stacked vector of all devices decision variables, $x_{\min} \in \mathrm{R}^n$ and $x_{\max} \in \mathrm{R}^n$ denote the bound constraints from (1), (4), (7), i.e.

$$x = \left[\alpha_1, \ldots, \alpha_{N_{gen}}, \beta_1, \ldots, \beta_{N_{acc}}, \gamma_1, \ldots, \gamma_{N_{loads}}\right]^T,$$
$$x_{\min} = \left[0, \ldots, 0, -1 \cdots -1, \tfrac{1}{N}, \ldots, \tfrac{1}{N}\right]^T,\ x_{\max} = \left[1, \ldots, 1, \ldots 1,\right]^T.$$

4. Sequential Linear Programming

Sequential Linear Programming (SLP) is an iterative method to find local minima of nonlinear constrained optimization problems, by solving a sequence of linear programming problems, (Robinson 1972; Byrd et al. 2003). The adopted SLP method is close to the one presented in (Robinson 1972), but is equipped with a trust-region approach to promote the global convergence. Moreover a bound-constraints handling strategy is adopted, that is similar to the one used in (Herty et al. 2007), where an SLP approach without trust-region is presented for equality constrained problems. The proposed approach is different also from those presented in (Fletcher and de la Maza 1989; Byrd et al. 2003) as second order information is not used.

At each iteration k, f and g are approximated in a neighbourhood of the current solution approximation x_k with first order Taylor series:

$$m_k(d) = f(x_k) + \nabla f(x_k)^T d \tag{9}$$

$$g_i(x_k) + \nabla g_i(x_k)^T d \leq 0 \qquad i = 1, \ldots, p, \tag{10}$$

where $d = x - x_k$. It is worth reminding that functions depending on the status function θ_k and the constraints functions for accumulators feature some discontinuities. Then, while they are evaluated exactly, for the computation of gradients they are approximated in a neighbourhood of those points by a surrogate function obtained replacing the discontinuous status and efficiency functions with continuous and differentiable sigmoid functions (Pannocchia and Mancuso 2014).

To obtain a globally convergent method, a trust region strategy is employed (Conn, Gould, and Toint 2000). Then, a new constraint is added, that consists in a bound on the step-length of this form: $\|d\|_\infty \leq \Delta_k$, where Δ_k is called the trust-region radius. The new constraint is added to the bound constraints, so that at each iteration the following problem is solved:

$$\min_d \ m_k(d), \tag{11a}$$

$$g_i(x_k) + \nabla g_i(x_k)^T d \leq 0; \ \ i = 1, \ldots, p, \tag{11b}$$

$$\max((x_{\min} - x_k)_j, -\Delta_k) \leq d_j \leq \min((x_{\max} - x_k)_j, \Delta_k), \ \ j = 1, \ldots, n. \tag{11c}$$

At each iteration the computed solution d_k of (11) is used as a step to define the new solution approximation: $x_{k+1} = x_k + d_k$. Anyway, it is not possible to work directly with problem (11), as the linearized constraints could be inconsistent, i.e. it could be impossible to find a d for which (11b) holds, or the solution found d_k could be such that the new approximation $x_k + d_k$ does not satisfy the nonlinear constraints:

$$x_k + d_k \notin \Omega = \{x : \ x_{\min} \leq x \leq x_{\max}, \ \ g(x) \leq 0\}.$$

To deal with this, the constraints are usually added in the objective function as a penalty parameter, i.e. a new term is added to function f that is positive if the constraints are not satisfied, and is zero otherwise. The resulting new objective function is called *a penalty function*. This term can be chosen in many different ways, so that different penalty functions are obtained. Following (Fletcher and de la Maza 1989; Byrd et al. 2003) the l_1 penalty function was chosen and the following penalized objective function is obtained:

$$\Phi(x;\nu) = f(x) + \nu \sum_{i=1}^{p} \max(0, -g_i(x)) \tag{12}$$

where $\nu > 0$ is the penalty parameter. If ν is sufficiently large, i.e.

$$\nu \geq \nu^* = \max\{\lambda_i^*, i = 1, \dots, p\}, \tag{13}$$

where λ_i^*, $i = 1, \dots, p$ are the Lagrange multiplier of the inequality constraints $g(x) \leq 0$, it is possible to show that l_1 is an exact penalty function, (Nocedal and Wright 2006):

Definition 4.1 Exact Penalty Function *A penalty function $\Phi(x, \nu)$ is said to be exact if there exists a positive scalar ν^* such that for any $\nu \geq \nu^*$, any local solution of the nonlinear programming problem* (8) *is a local minimizer of $\Phi(x, \nu)$.*

Since it is necessary to take into account the bound constraints, the following problem is actually to be dealt with:

$$\min_x \Phi(x;\nu) = f(x) + \nu \sum_{i=1}^{p} \max(0, -g_i(x)) \tag{14a}$$

$$x_{\min} \leq x \leq x_{\max}. \tag{14b}$$

If (13) holds, local minimizers of (14) are equivalent to local solutions of (8), to a large extent (see Fletcher (1987); Han and Mangasarian (1979); Janesch and Santos (1997) for details). The general scheme of the algorithm is the same as before, with the difference that instead of linearising both f and the constraints, function Φ is linearised and at each iteration a problem with just bound constraints is solved, that surely has a solution. As choosing the right value of ν a priori is difficult, a sequence of linearized penalized problems is actually solved, adjusting the penalty parameter during the course of the computation. Then, at iteration k given the current iterate x_k and the current penalty parameter ν_k, the following linear programming problem has to be solved:

$$\min_d l_k(d) := f(x_k) + \nabla f(x_k)^T d + \nu_k \sum_{i=1}^{p} \max(0, -g_i(x_k) - \nabla g_i(x_k)^T d); \tag{15a}$$

$$\max((x_{\min} - x_k)_j, -\Delta_k) \leq d_j \leq \min((x_{\max} - x_k)_j, \Delta_k),\ j = 1, \dots, n. \tag{15b}$$

After the solution d_k is found, it is necessary to decide whether to accept the step or not. The step acceptance is based on the agreement between the model function l_k and the objective function Φ, which is measured by the ratio between the *actual reduction* and the *predicted reduction* (Conn, Gould, and Toint 2000):

$$\rho_k = \frac{\Phi(x_k;\nu_k) - \Phi(x_k + d_k;\nu_k)}{l_k(0) - l_k(d_k)} = \frac{\Delta\Phi_k}{\Delta l_k}. \tag{16}$$

If $\rho_k > \rho_{bad}$, where ρ_{bad} is a tolerance to be fixed, typically $\rho_{bad} \in \left(0, \frac{1}{4}\right)$, the step is accepted. In this case the trust-region radius is left unchanged or it is possibly enlarged. Otherwise the step is rejected, the trust-region is shrink and (15) is solved again.

In Algorithm 1 it is sketched the k-th iteration of SLP method described above.

Algorithm 1 k-th iteration of SLP algorithm

(1) Given x_k, Δ_k, $\Delta_{\max}$, ν_k, $0 < \rho_{bad} < \rho_{good} < 1$.
(2) Evaluate $\nabla f(x_k)$ and $\nabla g_i(x_k)$ for $i = 1, \dots, p$.
(3) Solve the linear programming problem (15) obtaining a candidate step d_k.
(4) Let $\Phi(x;\nu) = f(x) + \nu \sum_{i=1}^{p} \max(0, -g_i(x))$, compute the step evaluation parameter ρ_k in (16).
  (a) If $\rho_k \le \rho_{bad}$, reduce the trust-region radius: $\Delta_{k+1} = \frac{1}{2}\Delta_k$, and go to step (5).
  (b) ElseIf $\rho_k \ge \rho_{good}$, and in addition $\|d_k\|_\infty \ge 0.8\Delta_k$, increase the trust-region radius
  $\Delta_{k+1} = \min(2\Delta_k, \Delta_{\max})$. Go to step (5).
  (c) Else set $\Delta_{k+1} = \Delta_k$ and go to step (5).
(5) If $\rho_k > \rho_{bad}$ accept the step, set: $x_{k+1} = x_k + d_k$ and choose $\nu_{k+1} > 0$.
Otherwise reject the step: $x_{k+1} = x_k$. Set $\nu_{k+1} = \nu_k$.

It is possible to prove the global convergence of the sequence $\{x_k\}$ generated by Algorithm 1 to a stationary point of problem (14), as stated in the following theorem. See Section 2 of the supplemental material for the proof.

Theorem 4.2 Global convergence of Algorithm 1
Let f and g be C^1 *functions and let $\{x_k\}$ be the sequence generated by Algorithm 1. Then, either there exists an iteration index $\bar{k}$ such that $x_{\bar{k}}$ is a stationary point for problem* (14), *or there exists a subsequence S of indexes such that $\{x_k\}_{k\in S}$ has an accumulation point x^* which is a stationary point for problem* (14).

Theorem 4.2 states the existence of an accumulation point x^* of the sequence $\{x_k\}$ generated by Algorithm 1 that is a stationary point of problem (14), regardless of the starting point. In Section 4.1 it will be described the chosen penalty parameter update strategy, which ensures that the penalty parameter ν is large enough to let x^* be a solution of the original problem.

4.1 *Implementation issues*

In this section three important features of the algorithm are discussed: the solution of subproblems (15), the stopping criterion and the updating strategy for the penalty parameter. Function l_k in (15a) is non-differentiable, but problem (15) can be written as the following equivalent smooth linear programming problem, introducing the vector of slack variables t, (Byrd et al. 2003):

$$\min_{d,t} \nabla f(x_k)^T d + \nu_k \sum_{i\in\mathcal{I}} t_i \tag{17a}$$

$$g_i(x_k) + \nabla g_i(x_k)^T d \le t_i, \quad i = 1, \dots, p \tag{17b}$$

$$\max((x_{\min} - x_k)_j, -\Delta_k) \le d \le \min((x_{\max} - x_k)_j, \Delta_k)_j, \quad j = 1, \dots, n, \tag{17c}$$

$$t \ge 0. \tag{17d}$$

Then, the solution d_k of the above problem is sought, along with the Lagrange multipliers vectors λ_k, π_k and $\bar{\lambda}_k$ of constraints (17b), (17c) and (17d) respectively. These multipliers are employed to implement a reliable stopping criterion for Algorithm 1. Algorithm 1 is stopped whenever a pair (x_k, λ_k) satisfies the following conditions:

$$\max\{\|\nabla f(x_k) + \nabla g(x_k)^T \lambda_k\|_\infty, \|g(x_k)^T \lambda_k\|_\infty\} < \epsilon(1 + \|\lambda_k\|_2), \tag{18}$$

$$\max\{\max_{i \in \mathcal{I}}(0, g_i(x_k)), \max(0, x_{\min} - x_k), \max(0, x_k - x_{\max})\} < \epsilon(1 + \|x_k\|_2), \tag{19}$$

with ϵ a tolerance to be fixed. This stopping criterion provides a measure of the closeness of the computed solution to a point satisfying first order optimality conditions for problem (8). See Theorem 2.5 in Section 2 of the supplemental material for a theoretical support for the employment of such criterion.

As far as the penalty parameter is concerned, it is desirable to have penalty function (12) to be exact, according to equation (13) and Definition 4.1. Then, at Step 5 of Algorithm 1, in case of successful iteration, the following updating strategy is adopted:

$$\text{If } \nu_k < \max\{\|\lambda_k\|_\infty, \|\bar{\lambda}_k\|_\infty\} \text{ then } \nu_{k+1} = \max\{\|\lambda_k\|_\infty, \|\bar{\lambda}_k\|_\infty\} \tag{20}$$

is set. Both in (18) and (20) current Lagrange multiplier estimates of the LP subproblems (17), provided by the function used to solve the LPs, are used as an approximation to those of the original problem.

5. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic evolutionary method designed to converge to a global minimum of a function f, (Kennedy 2011). It is inspired to the behaviour of bird swarms. Following the natural metaphor, PSO evolves a population of individuals, referred to as particles, within the search space, that behave according to simple rules and interact to produce a collective behaviour to pursuit a common aim, in this case the localization of a global minimum. The swarm is composed of s particles, each of them represents an approximation of the global minimum of the optimization problem and it is represented by a vector $x \in \mathrm{R}^n$. To each particle it is associated a velocity vector v too. The method is an iterative procedure where at each iteration k vectors x_k and v_k are updated as follows:

$$v_{k+1}^i = w v_k^i + c_1 r_1 (p_{best,k}^i - x_k^i) + c_2 r_2 (p_{best,k}^g - x_k^i) \tag{21}$$

$$x_{k+1}^i = x_k^i + v_{k+1}^i \tag{22}$$

where x_k^i and v_k^i are the position and velocity vector of the i-th particle, $i = 1, \ldots, s$, at the k-th iteration, $p_{best,k}^i$ and $p_{best,k}^g$ are respectively the best position reached by the i-th particle so far and the best position reached by the whole swarm (the best position is the one that corresponds to the lowest value of the objective function), c_1, c_2 and w are positive weights, r_1 and r_2 are random variables with uniform distribution in $[0, 1]$, $r_1, r_2 \sim U(0, 1)$. At each iteration vectors $p_{best,k}^i$ and $p_{best,k}^g$ are updated too:

$$p_{best,k+1}^i = \begin{cases} x_k^i & \text{if } f(x_k^i) < f(p_{best,k}^i) \\ p_{best,k}^i & \text{otherwise} \end{cases} \qquad p_{best,k+1}^g = \begin{cases} p & \text{if } f(p) < f(p_{best,k}^g) \\ p_{best,k}^g & \text{otherwise} \end{cases} \tag{23}$$

where $p = \arg\min_{i=1\ldots s} f(p_{best,k+1}^i)$.

The solution approximation provided by the procedure is $p_{best,k*}^g$ where k^* is the last iteration index. Bound constraints are handled bringing back on the nearest boundary a particle x that has left the search space. It is also necessary to change the particle velocity, otherwise at the next iteration it is likely to have a new violation: $(v_k^i)_j = -r(v_k^i)_j$, where $(v_k^i)_j$ is the j-th component of vector v_k^i and r is a random variable uniformly distributed in $(0,1)$. Originally PSO methods were developed to deal with problems with just bound constraints, and later they were employed to solve also constrained problems (Aziz et al. 2011; Parsopoulos, Vrahatis et al. 2002). See also (Mezura-Montes and Coello 2011; Jordehi 2015) for a more recent review on constraints handling strategies for PSO methods. Penalty function approaches are widely employed to make the method suitable to the solution of that kind of problems. Following (Michalewicz and Attia 1994; Michalewicz and Schoenauer 1996), the following quadratic penalty function is employed:

$$\Psi(x;\tau) = f(x) + \frac{1}{2\tau}\sum_{i=1}^{p} g_i(x)^2, \tag{24}$$

with τ penalty parameter that is decreased at each iteration so as to increase the weight of the penalty term in the objective function, to penalize with increasing severity constraints violations. Then, given a penalty parameter τ_k, function $\Psi(x;\tau_k)$ is used in (23) in place of the objective function $f(x)$. The k-th iteration of PSO procedure is sketched in Algorithm 2.

Algorithm 2 k-th iteration of PSO algorithm

(1) Given $c_1, c_2, w_{\min}, w_{\max}$, $k_{\max}$, τ_k, $x_{\max}$, $x_{\min}$, x_k^i, v_k^i, $p_{best,k}^i$, $p_{best,k}^g$ for $i = 1,\dots,s$, perform the following steps.
(2) Compute w_k in (26), $r_1, r_2 \sim U(0,1)$ and evolve the swarm according to (21), (22).
(3) Handle the bound constraints: for $i = 1,\dots,s$ and $j = 1,\dots,n$
 - If $(x_{k+1}^i)_j < (x_{\min})_j$ compute $r \sim U(0,1)$ and set

$$(x_{k+1}^i)_j = (x_{\min})_j,$$
$$(v_{k+1}^i)_j = -r(v_{k+1}^i)_j,$$

 - Elseif $(x_{k+1}^i)_j > (x_{\max})_j$ compute $r \sim U(0,1)$ and set

$$(x_{k+1}^i)_j = (x_{\max})_j,$$
$$(v_{k+1}^i)_j = -r(v_{k+1}^i)_j.$$

(4) Evaluate the objective function (24) of each particle of the swarm and update vectors

$$p_{best,k+1}^i = \begin{cases} x_k^i & \text{if } \Psi(x_k^i;\tau_k) < \Psi(p_{best,k}^i;\tau_k) \\ p_{best,k}^i & \text{otherwise} \end{cases}$$

$$p_{best,k+1}^g = \begin{cases} p & \text{if } \Psi(p;\tau_k) < \Psi(p_{best,k}^g;\tau_k) \\ p_{best,k}^g & \text{otherwise} \end{cases}$$

where $p = \arg\min_{i=1\dots s} \Psi(p_{best,k+1}^i;\tau_k)$ and choose $\tau_{k+1} < \tau_k$.

In the implemented PSO method a new strategy to help the swarm escape from local minima is proposed. When the swarm appears to be stuck, i.e. when after a certain number of iterations the value of the objective function is not decreased, the velocity updating scheme (21) is modified adding a new term in the equation, that is proportional to the distance of the global best position from the particle best position:

$$v_{k+1}^i = wv_k^i + c_1r_1(p_{best,k}^i - x_k^i) + c_2r_2(p_{best,k}^g - x_k^i) + c_3r_3(p_{best,k}^i - p_{best,k}^g), \tag{25}$$

where c_3 is a weighting parameter to be set, and $r_3 \sim U(0,1)$. When coefficient $c_3 \neq 0$ the weight of the term $(p^i_{best,k} - x^i_k)$ is increased while that of $(p^g_{best,k} - x^i_k)$ is decreased. In this way, the particles are less attracted by the global best position, fostering the swarm to escape from a stalemate. When a new improvement in the objective function is obtained, coefficient c_3 is set back to zero and the standard update is employed. Numerical experiments have shown that in some tests the addition of this term indeed helps the swarm to find a better solution approximation. In the following this PSO variant will be addressed as PSOc_3.

The main advantage of PSO methods is that they do not require neither regularity assumptions on the objective function nor to compute the first derivatives and are so suitable when few information on the objective function are available. Clearly the fact that few information on f are used, leads to a slow method that requires many iterations to converge. However, the method could be efficiently implemented on a parallel architecture.

These methods are heuristic and standard convergence results, like those proved for exact optimization methods, are not usually provided. However, a different kind on analysis of the algorithm can be performed. Using results from the dynamic system theory, it is possible to provide an understanding about how the swarm searches the problem space through the analysis of a single particle trajectory or of the the swarm seen as a stochastic system, (Trelea 2003; Clerc and Kennedy 2002). The analysis provides useful guidelines for the choice of the free parameters, to control the system's convergence tendencies.

5.1 *Implementation issues*

The choices regarding the method implementation are made taking into account several factors. The problems under consideration have hundreds of variables and the objective function is a sum over a large number of terms. Also, the algorithm must return results quickly to be useful in practise.

Regarding the number of particles, it is impossible to use wide swarms because each particle of the swarm requires two function evaluations, the objective and the constraints functions, at each iteration. On the other hand using few particles means low exploration capability. After several numerical tests, it was found that a swarm of 20 particles represents a good compromise between solution quality and execution time. For parameter w in (21), instead of using a fixed value, a linearly decreasing scheme is adopted, (Shi and Eberhart 1998):

$$w_k = w_{\max} - (w_{\max} - w_{\min})\frac{k}{k_{\max}}, \tag{26}$$

where $k_{\max}$ is the maximum number of allowed iterations. With this choice convergence velocity is slowed down ensuring a more accurate exploration of the search space. The process is stopped when a maximum number of iterations $k_{\max}$ is performed or when there are no improvements over a fixed number of iterations κ. An improvement is measured in terms of a decrease in the objective function, and it is judged not to be sufficient when the following condition is satisfied for κ consecutive iterations and for ϵ a tolerance to be fixed:

$$\frac{|f(x_{k-2}) - f(x_k)|}{|f(x_{k-2})|} < \epsilon \tag{27}$$

6. Numerical Tests

Both solvers were implemented in MATLAB and numerical tests were performed to study their practical performance. First the proposed methods have been benchmarked against some of the state-of-the-art metaheuristic methods, on a set of functions commonly used to test performance of such approaches. It emerged that the proposed methods can compare with other metaheuristic approaches recently proposed in the literature. The results of these tests are reported in Section 1 of the supplemental material.

Then, 13 different problems arising from the described industrial application and corresponding to models of synthetic energy districts were taken into account, plus one arising from the modelling of an existing district in Pisa, provided by Enel centre. To solve these problems, the two solvers were inserted in the software tool previously developed by Enel centre and University of Pisa.

The numerical experimentation is performed on an Intel(R)Xeon(R) CPU E5430, 2.66 GHz, 8.00 GB RAM, using MATLAB R2016a. The machine precision is $\epsilon_m \sim 2 \cdot 10^{-16}$.

Here the values of the free coefficients used in the procedures are specified. For PSOc_3 in (21) $c_1 = 1.3$ and $c_2 = 2.8$, in (25) $c_3 = 1$, in (26) $w_{\max} = 0.6$, $w_{\min} = 0.1$, the tolerance for the stopping criterion (27) is $\epsilon = 10^{-3}$, $\kappa = 20$ and $k_{\max} = 1700$. The swarm size is 20. The choice of this rather small value is motivated by the fact that in energy districts test cases the objective function is expensive to evaluate and the use of wider swarms would lead to prohibitive computational costs. In the PSO constraints handling strategy (24) it was set $\tau_{k+1} = (1 - 0.01)\tau_k$ and $\tau_0 = 0.1$. For SLP in (12) it was set $\nu_0 = 1$. In Algorithm 1 $\rho_{bad} = 0.10$ and $\rho_{good} = 0.75$, and the linear subproblems were solved using the Matlab function `linprog`(Interior Point Algorithm) with default choice parameters. Notice that the structure of the subproblems is not taken into consideration and a special purpose code could lead to a more efficient solution of the subproblems, but this is out of the scope of this article.

Results shown in the following tables are the average of those obtained over 100 runs, varying the starting point for SLP solver. In the tables for each one of the test cases the following statistics are reported: $\boldsymbol{\bar{f}}$ and $\boldsymbol{\bar{k}}$ arithmetic mean of the function values and the number of iterations, $\boldsymbol{\sigma_f}$ the standard deviation on function values, $\mathbf{min}\,\boldsymbol{f}$ and $\mathbf{max}\,\boldsymbol{f}$ minimum and maximum function values obtained, $\mathbf{time}(\cdot)$ total time in seconds or minutes.

6.1 *Examples of synthetic energy districts*

In this section the results gained by the two methods when applied to 13 synthetic test examples of energy districts are presented. The arising optimization problems have different dimensions, the number of variables is comprised between 294 and 494, the number of bound constraints between 588 and 796 and the number of process constraints is 10 for the first test and 213 for the others. The results of the tests performed are reported in Table 1.

At first glance, from these results it is possible to deduce the following remarks.

The convergence rate of SLP method is much higher then that of PSO method. This is a consequence of the fact that SLP solver is a gradient-based method. Indeed, as PSO method does not employ first order information, it requires an high number of iteration to allow the swarm to carefully explore the search space. In term of computational time an iteration of PSO method is cheaper than one of SLP. Each SLP iteration indeed, requires the solution of a Linear Programming problem and the computation of some derivatives by finite differences, which is computationally expensive. Despite this, the total time required by SLP is really lower than that required by PSO method, as SLP

Table 1.: Tests on 13 synthetic examples of energy districts and on Pisa district (last row), comparison of PSOc_3 and SLP solvers.

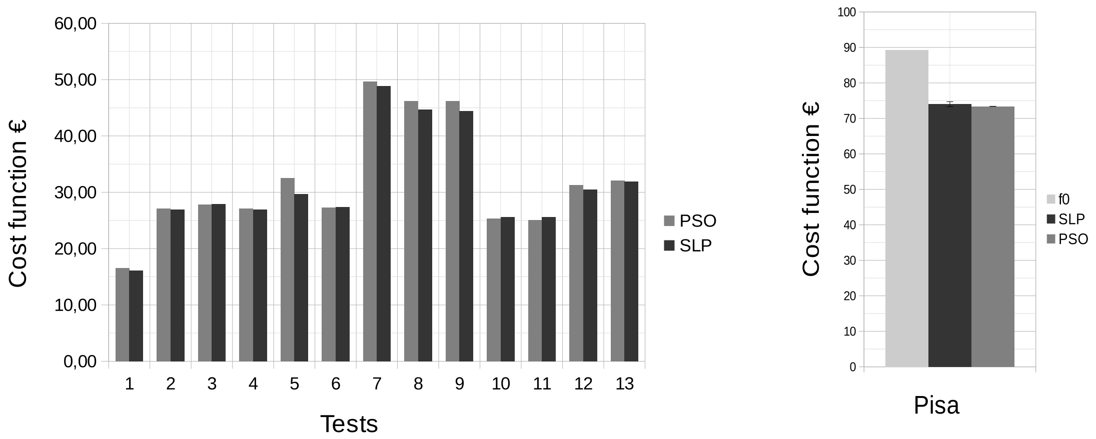
Problem	Solver	$\bar{f}$	σ_f	max f	min f	$\bar{k}$	time(m)
Test 1	PSOc_3	16.6	0.6	18.3	15.4	1207	1.8
	SLP	16.1	1.9	25.8	15.4	25.4	0.06
Test 2	PSOc_3	27.1	0.3	27.7	26.4	1585	13
	SLP	26.9	0.7	29.3	25.8	78.9	1.0
Test 3	PSOc_3	27.8	0.3	28.5	27.1	1554	14
	SLP	27.9	0.7	30.7	26.8	84	1.3
Test 4	PSOc_3	27.1	0.3	27.8	26.5	1583	11
	SLP	26.9	0.7	29.2	26.1	82.5	0.5
Test 5	PSOc_3	32.5	1.5	36.6	29.2	1493	14
	SLP	29.7	4.4	48.7	26.7	78.8	1.0
Test 6	PSOc_3	27.3	0.4	28.6	26.7	1588	11
	SLP	27.4	1.1	31.2	26.2	83.8	0.5
Test 7	PSOc_3	49.7	0.7	52.2	48.2	1509	10
	SLP	48.9	0.9	53.1	46.9	76.3	0.4
Test 8	PSOc_3	46.2	0.8	49.2	44.3	1531	10
	SLP	44.7	1.8	51.9	42.9	78.9	0.4
Test 9	PSOc_3	46.2	0.8	48.6	44.5	1543	15
	SLP	44.4	1.6	51.3	42.9	79.6	0.4
Test 10	PSOc_3	25.3	0.3	26.1	24.7	1484	14
	SLP	25.6	0.5	28.2	24.9	73.0	1.0
Test 11	PSOc_3	25.1	0.3	26.3	24.5	1406	14
	SLP	25.6	0.6	28.4	24.8	70.3	1.0
Test 12	PSOc_3	31.3	0.9	34.7	29.3	1379	12
	SLP	30.5	3.0	45.4	29.0	64.4	0.6
Test 13	PSOc_3	32.1	0.8	35.1	30.4	1385	11
	SLP	31.9	4.0	53.5	30.0	64.8	0.7
Pisa	PSOc_3	73.4	0.03	73.6	73.4	377	2.7
	SLP	74.0	0.7	80.6	73.4	23	0.08

performs far less iterations. As expected SLP is more suitable for real time optimization.

Regarding function values it can be observed that on many tests PSO presents higher means compared to SLP, but generally the values on different runs are close to the mean value, as the standard deviation is really low. On the other hand SLP provides lower means but high standard deviation, which means both higher worst values and lower best values. Then, the probability of finding a worst result on a single run using SLP is high, while for PSO is more likely to have a result close to the mean value. This is highlighted in Figure 1 (a), where for each test case the mean values together with the standard deviation for the two solvers are compared. The left bars (light grey) refer to PSOc_3 method and the right ones (dark grey) to SLP method, the standard deviation is highlighted by black vertical lines. Note that overall PSO provides good results, even if a small number of particles is employed. Its performance could be improved increasing the swarm size, to better explore the search space. The difference between PSO and SLP is indeed especially evident in Test 5, which is the one in which the search space has highest dimension. However, from the numerical experience it emerged that increasing the number of particles would be prohibitive from a computational point of view.

Then, a deeper statistical analysis of the results was performed. First it was checked if the results obtained in the tests satisfy the conditions for the use of a parametric test, cf.

(García et al. 2009; Derrac et al. 2011). They were tested with the Kolmogorov-Smirnov test, via the Matlab function `kstest`. It emerged that the data are not normally distributed, meaning that a non-parametric analysis would be more meaningful. Therefore the data were analysed by the Wilcoxon signed ranks test, cf. (Derrac et al. 2011), based on the mean values found, through the Matlab function `signrank`. Wilcoxon test is a nonparametric test that is used to determine whether two independent samples were selected from populations having the same distribution, therefore it can be employed to detect significant differences between the behaviour of two algorithms. The null hypothesis is that the difference between two sample means is zero. From the test it emerges that the data produce an evidence that is sufficient to reject the null hypothesis, according to significance level $\alpha = 0.05$. If R^+ denotes the sum of ranks for the problems in which SLP outperforms PSOc_3, and R^- the sum of ranks for the opposite, one obtains $R^+ = 84, R^- = 21$. Then, the statistical tests confirm that SLP shows an improvement over PSO on these problems, if the mean values are considered.



(a) (b)

Figure 1.: (a) Comparison of objective function values obtained by PSOc_3 (left bars, light grey) and SLP (right bars, dark grey) on the 13 tests cases; (b) comparison of actual management ($f_0 = 89.3$, left bar) and optimized management provided by SLP (central bar) and PSOc_3 (right bar), on the test case of Pisa district.

In the next section results of the tests performed on a real energy district are shown.

6.1.1 Pisa District

This is a real district in Pisa provided by the research centre 'Enel Ingegneria e Ricerca', that comprises: a photovoltaic generator with rated power 14 kWe, a wind farm with rated power 3 kWe, 2 L1 loads related to lighting and heating consumptions. Moreover, a thermal group is also part of the district, that is composed of a CHP characterized by rated power 25 kWe and rated thermal power 75 kWt, a gas boiler with rated thermal power 35 kWt, a tank for the storage of hot water with capacity 9400 kJ/ °C. This was modelled choosing one among the predefined thermal configurations provided by the software tool. The decision variables for each of these machines are described in Section 2. As for synthetic energetic districts, the time horizon is one day, that is divided into $N = 96$ time steps. The arising optimization problem has 288 variables, 576 bound constraints and 1 physical constraint. The detailed results of the optimization process are reported in the last row of Table 1.

This test is much less constrained than those presented in the previous section, it has just one mild nonlinear constraint, and the dimensionality of the search space is also lower. In this case PSO algorithm performs better than SLP algorithm, providing a lower mean value of the objective function and also a really smaller standard deviation. Notice that in this case also for PSOc_3 algorithm the execution time is quite reasonable.

This test case is particularly interesting, because data referring to the cost of unoptimized management of local resources, i.e. the management that is actually running the district, are available. Specifically, the daily cost function of the district for energy consumption is $f_0 = 89.3$, as it is depicted in Figure 1 (b), left bar. By means of the proposed procedures, it is possible to predict values of the machines' decision variables which minimize the cost function. The district management gained setting the machine parameters to these optimal values will be referred to as optimal management. In Figure 1 (b), right bars, the optimal values of the objective function found by SLP and PSO respectively are reported. They represent the cost associated to the optimal management and can be compared to f_0 (the value of the unoptimized cost function) to evaluate savings arising from the employment of the optimized management. The comparison shows that the optimized approach let the district save 18% of the overall daily expense to buy energy from the network.

7. Conclusions

In this article a modelling of energy districts is considered that gives rise to a NLP. The main focus of the article is the numerical solution of such a problem. To this aim, a Particle Swarm Optimization solver and a Sequential Linear Programming solver were implemented and compared. The solvers were tested first on constrained tests taken from the literature, and it emerged that they can compare with other methods recently proposed in the literature. Then, the solvers were tested on many different examples of synthetic energy districts, including also a real case study. From this study it was highlighted that PSO provides values less distributed around the mean, but lower means are generally provided by SLP method. The results suggest that a hybrid PSO/SLP method may be a promising alternative solver, that will be object of future work. Noticeably, from the test performed on the real case study it arises that the optimized management of resources gained by the optimization package provides considerable savings in the energy bill.

Acknowledgements

We thank G. Pannocchia and G.M. Mancuso, Dipartimento di Ingegneria Civile e Industriale dell' Università di Pisa, for providing us the MATLAB code of the software tool for the test problems.
Work partially supported by INdAM-GNCS.

References

Ascione, F., N. Bianco, C. De Stasio, G. M. Mauro, and G.P. Vanoli. 2016. "Simulation-Based Model Predictive Control by the Multi-Objective Optimization of Building Energy Performance and Thermal Comfort." *Energy and Buildings* 111: 131–144.

Aziz, N.A.A., M.Y. Alias, A.W. Mohemmed, and K.A. Aziz. 2011. "Particle Swarm Optimization for Constrained and Multiobjective Problems: a Brief Review." In *International Conference on Management and Artificial Intelligence,* Vol. 6: 146–150. IACSIT Press, Bali, Indonesia.

Bischi, A., L. Taccari, E. Martelli, E. Amaldi, G. Manzolini, P. Silva, S. Campanari, and E. Macchi. 2014. "A Detailed MILP Optimization Model for Combined Cooling, Heat and Power System Operation Planning." *Energy* 74: 12–26.

Byrd, R.H., N.I.M. Gould, J. Nocedal, and R.A. Waltz. 2003. "An Algorithm for Nonlinear Optimization Using Linear Programming and Equality Constrained Subproblems." *Mathematical Programming* 100 (1): 27–48.

Cagnina, L.C., S.C. Esquivel, and C.A.C. Coello. 2011. "Solving Constrained Optimization Problems with a Hybrid Particle Swarm Optimization Algorithm." *Engineering Optimization* 43 (8): 843–866.

Clerc, M., and J. Kennedy. 2002. "The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space." *IEEE Transactions on Evolutionary Computation* 6 (1): 58–73.

Conn, A.R., N.I.M. Gould, and Ph.L. Toint. 2000. *Trust Region Methods*. Vol. 1. Siam, Philadelphia, Pennsylvania, USA.

De-los Cobos-Silva, S. G., R.A. Mora-Gutiérrez, M.A. Gutiérrez-Andrade, E.A. Rincón-García, A. Ponsich, and P. Lara-Velázquez. 2018. "Development of Seven Hybrid Methods Based on Collective Intelligence for Solving Nonlinear Constrained Optimization Problems." *Artificial Intelligence Review* 49 (2): 245–279.

Derrac, J., S. García, D. Molina, and F. Herrera. 2011. "A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms." *Swarm and Evolutionary Computation* 1 (1): 3–18.

Fan, S.K.S., and E. Zahara. 2007. "A Hybrid Simplex Search and Particle Swarm Optimization for Unconstrained Optimization." *European Journal of Operational Research* 181 (2): 527 – 548.

Ferrara, W., J. Riccardi, and S. Sello. 2014. "Enel Customer Optimization Service Expert System Development." In *Technical Report,* Enel Ingegneria e Ricerca, Pisa.

Fletcher, R. 1987. "Practical Methods of Optimization." *Wiley-Interscience Publication, Hoboken, New Jersey, USA* .

Fletcher, R., and E.S. de la Maza. 1989. "Nonlinear Programming and Nonsmooth Optimization by Successive Linear Programming." *Mathematical Programming* 43 (1-3): 235–256.

Fu, Q., and N. Tong. 2010. "A Complex-Method-Based PSO Algorithm for the Maximum Power Point Tracking in Photovoltaic System." In *2010 Second International Conference on Information Technology and Computer Science, Kiev, Ukraine,* 134–137. IEEE, Piscataway, New Jersey, USA.

García, S., D. Molina, M. Lozano, and F. Herrera. 2009. "A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms Behaviour: a Case Study on the CEC2005 Special Session on Real Parameter Optimization." *Journal of Heuristics* 15 (6): 617–644.

Han, S.P., and O. Mangasarian. 1979. "Exact Penalty Functions in Nonlinear Programming." *Mathematical programming* 17 (1): 251–269.

Herty, M., A. Klar, A.K. Singh, and P. Spellucci. 2007. "Smoothed Penalty Algorithms for Optimization of Nonlinear Models." *Computational Optimization and Applications* 37 (2): 157–176.

Hu, X., and R. Eberhart. 2002. "Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization." In *6th world Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA,* 203–206. IIIS, Orlando, Florida, USA.

Janesch, S.M.H., and L.T. Santos. 1997. "Exact Penalty Methods with Constrained Subproblems." *Investigacion Operativa* 7 (1-2): 55–65.

Jin, X., and R. G. Reynolds. 1999. "Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach." In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, Washington, DC, USA,* Vol. 3, IEEE, Piscataway, New Jersey, USA.

Jordehi, A.R. 2015. "A Review on Constraint Handling Strategies in Particle Swarm Optimisation." *Neural Computing and Applications* 26 (6): 1265–1275.

Kennedy, J. 2011. "Particle Swarm Optimization." In *Encyclopedia of Machine Learning,* 760–766. Springer, Berlin, Germany.

Liang, J.J., T. P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello, and K. Deb. 2005. "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization." *Technical Report, Nanyang Technological*

University, Singapore. Available at: http://www.lania.mx/ emezura .

Liu, H., Z. Cai, and Y. Wang. 2010. "Hybridizing Particle Swarm Optimization with Differential Evolution for Constrained Numerical and Engineering Optimization." *Applied Soft Computing* 10 (2): 629–640.

Lu, H., and W. Chen. 2008. "Self-Adaptive Velocity Particle Swarm Optimization for Solving Constrained Optimization Problems." *Journal of Global Optimization* 41 (3): 427–445.

Martelli, E., and E. Amaldi. 2014. "PGS-COM: A Hybrid Method for Constrained Non-Smooth Black-Box Optimization Problems." *Computers & Chemical Engineering* 63: 108 – 139.

Mazhoud, I., K. Hadj-Hamou, J. Bigeon, and P. Joyeux. 2013. "Particle Swarm Optimization for Solving Engineering Problems: A New Constraint-Handling Mechanism." *Engineering Applications of Artificial Intelligence* 26 (4): 1263 – 1273.

Mezura-Montes, E., and C.A.C. Coello. 2011. "Constraint-Handling in Nature-Inspired Numerical Optimization: Past, Present and Future." *Swarm and Evolutionary Computation* 1 (4): 173–194.

Mezura-Montes, E., M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón. 2010. "Differential Evolution in Constrained Numerical Optimization: An Empirical Study." *Information Sciences* 180 (22): 4223–4262.

Michalewicz, Z., and N. Attia. 1994. "Evolutionary Optimization of Constrained Problems." In *Proceedings of the 3rd annual conference on evolutionary programming, San Diego, California, USA,* 98–108. World Scientific, Singapore.

Michalewicz, Z., and M. Schoenauer. 1996. "Evolutionary Algorithms for Constrained Parameter Optimization Problems." *Evolutionary Computation* 4 (1): 1–32.

Nocedal, J., and S. Wright. 2006. *Numerical Optimization.* Springer Science & Business Media, New York, USA.

Pannocchia, G., and G.M. Mancuso. 2014. "Steady State Optimizer of Energy Districts." In *Technical Report,* Dipartimento di Ingegneria Civile e Industriale, Pisa.

Parsopoulos, K.E., M.N. Vrahatis, et al. 2002. "Particle Swarm Optimization Method for Constrained Optimization Problems." *Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies* 76 (1): 214–220.

Riccietti, E., S. Bellavia, and S. Sello. 2017. "Numerical Methods for Optimization Problems Arising in Energetic Districts." In *Proceedings of ECMI 2016, Mathematics in Industry, Santiago de Compostela, Spain,* edited by Germany Springer, Berlin.

Robinson, S.M. 1972. "A Quadratically-Convergent Algorithm for General Nonlinear Programming Problems." *Mathematical Programming* 3 (1): 145–156.

Salgado, F., and P. Pedrero. 2008. "Short-Term Operation Planning on Cogeneration Systems: A Survey." *Electric Power Systems Research* 78 (5): 835–848.

Schiefelbein, J., J. Tesfaegzi, R. Streblow, and D. Müller. 2015. "Design of an Optimization Algorithm for the Distribution of Thermal Energy Systems and Local Heating Networks within a City District." In *Proceedings of the 28th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS 2015), Pau, France,* LaTEP, Pau, France.

Shi, Y., and R. Eberhart. 1998. "A Modified Particle Swarm Optimizer." In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence,* 69–73. IEEE, Piscataway, New Jersey, USA.

Tessema, B., and G.G. Yen. 2009. "An Adaptive Penalty Formulation for Constrained Evolutionary Optimization." *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 39 (3): 565–578.

Trelea, I.C. 2003. "The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection." *Information Processing Letters* 85 (6): 317 – 325.

Vaz, A.I.F., and L.N. Vicente. 2009. "PSwarm: a Hybrid Solver for Linearly Constrained Global Derivative-Free Optimization." *Optimization Methods and Software* 24 (4-5): 669–685.

Wang, C., I. Martinac, and A. Magny. 2015. "Multi-Objective Robust Optimization of Energy Systems for a Sustainable District in Stockholm." In *Proceedings of the 14th Conference of the International Building Performance Simulation Association, Hyderabad, India,* BSP, Hyderabad, India.

Zhang, W.J., and X.F. Xie. 2003. "DEPSO: Hybrid Particle Swarm with Differential Evolution

Operator." In *IEEE International Conference on Systems, Man and Cybernetics, Washington, DC, USA,* Vol. 4: 3816–3821. IEEE, Piscataway, New Jersey, USA.

To appear in *Engineering Optimization*
Vol. 00, No. 00, Month 20XX, 1–10

Supplemental material for the paper: Sequential Linear Programming and Particle Swarm Optimization for the Optimization of Energy Districts

E. Riccietti [a∗] and S. Bellavia[b] and S. Sello[c]

[a]*Dipartimento di Matematica e Informatica "Ulisse Dini", Università di Firenze, viale G.B. Morgagni 67a, 50134 Firenze, Italia.*; [b]*Dipartimento di Ingegneria Industriale, Università di Firenze, viale G.B. Morgagni 40, 50134 Firenze, Italia, stefania.bellavia@unifi.it*; [c]*Enel Ingegneria e Ricerca, Via Andrea Pisano 120, 56122 Pisa, Italia, stefano.sello@enel.com*

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

1. Comparison with the state-of-the-art

In this section the performance of the proposed PSO and SLP methods is evaluated on the set of CNOP (constrained nonlinear optimization problems) called G-suite, proposed in (Liang et al. 2006) for the competition on constrained optimization of the Congress on Evolutionary Computation in 2006 (CEC'2006). This set was chosen since it provides a challenging set of functions and it makes possible to compare the proposed methods to the state-of-the-art of metaheuristic methods. Indeed, a large part of papers dealing with evolutionary methods refers to this set of functions, see for example (Das and Suganthan 2011; de-los Cobos-Silva et al. 2016; Liu et al. 2016; Mezura-Montes and Lopez-Ramirez 2007). The G-suite is composed of 24 CNOP. Among them, those subject just to inequality constraints were selected, as the proposed methods are not designed to handle equality constraints. Table 1 sums up the relevant features of each test function. In the heading, n is the number of variables, $\rho = |F|/|S|$ is the estimated ratio between the feasible region and the search space, LI is the number of linear inequality constraints and NI is the number of nonlinear inequality constraints.

The tests are performed on an Intel(R) Core(TM) i7-4510U 2.00GHz, 16 GB RAM; using Matlab2016a, the machine precision is $\epsilon_m \sim 2 \cdot 10^{-16}$.

Here the values of the free coefficients used in the procedures are specified. For PSOc_3 in

$$v^i_{k+1} = w_k v^i_k + c_1 r_1 (p^i_{best,k} - x^i_k) + c_2 r_2 (p^g_{best,k} - x^i_k) + c_3 r_3 (p^i_{best,k} - p^g_{best,k}),$$

it was set $c_1 = 1.3$, $c_2 = 2.8$, $c_3 = 1$, and

$$w_k = w_{\max} - (w_{\max} - w_{\min}) \frac{k}{k_{\max}},$$

∗Corresponding author. Email: elisa.riccietti@unifi.it

Table 1. Details of the test problems.

Prob.	n	**Type**	ρ	**LI**	**NI**
g01	13	quadratic	0.0111%	9	0
g02	20	nonlinear	99.9971%	0	2
g04	5	quadratic	52.1230%	0	6
g06	2	cubic	0.0066%	0	2
g07	10	quadratic	0.0003%	3	5
g08	2	nonlinear	0.8560%	0	2
g09	7	polynomial	0.5121%	0	4
g10	8	linear	0.0010%	3	3
g12	3	quadratic	4.7713%	0	1
g16	5	nonlinear	0.0204%	4	34
g18	9	quadratic	0.0000%	0	13
g19	15	nonlinear	33.4761%	0	5
g24	2	linear	76.6556%	0	2

with $w_{\max} = 0.6$, $w_{\min} = 0.1$. The tolerance for the stopping criterion

$$\frac{|f(x_{k-2}) - f(x_k)|}{|f(x_{k-2})|} < \epsilon \tag{1}$$

is $\epsilon = 10^{-3}$, and the process is stopped either when (1) is satisfied for $\kappa = 20$ consecutive iterations, or if $k_{\max} = 1700$ iterations are performed. The swarm size is 100. This is bigger than the swarm size used in Section 6 of the main paper. This choice is motivated by the fact that in energy districts test cases the objective function is expensive to evaluate and the use of wider swarms would lead to prohibitive computational costs. On the other hand in test problems taken from the literature this is not the case, so a higher number of particle is employed to allow a better exploration of the search space. Moreover for these test problems it was found harder to satisfy the constraints than in the tests of energy districts. Then for both PSO and SLP algorithm constraints violations are penalized with higher severity here than in Section 6 of the main paper. Then, in the PSO quadratic penalty function

$$\Psi(x; \tau_k) = f(x) + \frac{1}{2\tau_k} \sum_{i=1}^{p} g_i(x)^2,$$

it was set $\tau_{k+1} = (1 - 0.01)\tau_k$ and $\tau_0 = 10^{-6}$.

For SLP in

$$\Phi(x; \nu) = f(x) + \nu \sum_{i=1}^{p} \max(0, -g_i(x))$$

it was set $\nu_0 = 5$. In Algorithm 1 (reported in the main paper) $\rho_{bad} = 0.10$ and $\rho_{good} = 0.75$, and the linear subproblems were solved using the Matlab function `linprog`(Interior Point Algorithm) with default choice parameters.

For each test function 25 runs were performed, as required by the CEC'2006 Special Session. The results are reported in Table 2. For each one of the test cases the following statistics are reported: $\bar{f}$ and $\bar{k}$ arithmetic mean of the function values and the number of iterations, σ_f the standard deviation on function values, **min** f and **max** f minimum and maximum function values obtained, **time(s)** total time in seconds, **feas** the percentage of runs in which a feasible solution is found.

Noticeably, PSOc_3 always finds feasible solutions in all runs, while SLP method is not so robust from this point of view, in 4 out of 13 tests (g07, g08, g09, g10) a feasible solution is found in 75%-90% of the runs and especially in instance g06 a feasible solution is never found. The results reported refers just to the feasible runs. In 9 out of 13 instances PSOc_3 manages to find the best known solution (g01, g04, g06, g07, g08, g09, g12, g16, g24). In two instances (g18, g19) it finds the best values reported in (Liang et al. 2006), but recently even better values have been found in (de-los Cobos-Silva et al. 2016). In the remaining two instances values close to the minimum are found, but the global minimum is not reached. Regarding SLP method, in 6 instances it finds the global minimum (g01, g04, g06, g07, g09, g24). In instances g08 and g12 the best value found is close to it. In instances g02 and g10 its performance is worst than that of PSOc_3 and as PSOc_3 in instances g18, g19 SLP finds the best values reported in (Liang et al. 2006), but not the ones found in (de-los Cobos-Silva et al. 2016). Noticeably, in many runs the standard deviation is really low.

Table 2. Tests on 13 test functions taken from the G-suit of CEC'2006. Comparison of PSOc_3 and SLP solvers.

Test	**Solver**	$\bar{f}$	σ_f	**max** f	**min** f	$\bar{k}$	**time(s)**	**feas**
g01	PSOc_3	-14.54	0.94	-12.00	-15.00	130.0	3.31	100%
	SLP	-14.28	0.76	-12.66	-15.00	68.7	1.62	100%
g02	PSOc_3	-0.60	0.05	-0.49	-0.69	130.0	3.36	100%
	SLP	-0.19	0.06	-0.076	-0.27	17.6	2.10	100%
g04	PSOc_3	-30665	0.10	-30665	-30666	130.0	2.80	100%
	SLP	-30666	1.18e-08	-30666	-30666	300.0	6.67	100%
g06	PSOc_3	-6951.6	7.32	-6935	-6961.1	130.0	2.72	100%
	SLP	-6961.8	1.70e-05	-6961.8	-6961.8	200.0	3.90	100%
g07	PSOc_3	25.20	0.70	26.56	24.35	1322.9	33.50	100%
	SLP	24.55	0.9784	28.22	24.31	47.7	1.40	80%
g08	PSOc_3	-0.0958	3.18e-17	-0.0958	-0.0958	180.0	5.08	100%
	SLP	-0.0120	0.03	0.0460	-0.0860	28.2	0.68	90%
g09	PSOc_3	681.28	0.39	682.63	681.03	636.3	17.26	100%
	SLP	680.63	8.86e-09	680.63	680.63	62.1	1.80	90%
g10	PSOc_3	7760.8	775.93	10607	7241.6	1520.6	33.53	100%
	SLP	14237	40400	22300	8617	15.6	1.34	75%
g12	PSOc_3	-1	0	-1	-1	180.0	7.32	100%
	SLP	-0.799	0.11	-0.5540	-0.9864	20.6	0.62	100%
g16	PSOc_3	-1.90	1.5e-3	-1.8997	-1.9041	239.9	8.82	100%
	SLP	-	-	-	-	-	-	0%
g18	PSOc_3	-0.73	0.13	-0.49	-0.8577	180.0	4.52	100%
	SLP	-0.84	0.07	-0.6750	-0.8660	43.4	1.25	100%
g19	PSOc_3	40.71	5.89	57.64	34.44	1229.1	35.08	100%
	SLP	32.66	3.42e-09	32.66	32.66	52.3	1.46	100%
g24	PSOc_3	-5.5080	5.24e-09	-5.5080	-5.5080	180.0	4.81	100%
	SLP	-4.28	1.08	-2.23	-5.5080	4.3	0.07	100%

As suggested for example in (Derrac et al. 2011; García et al. 2009) a nonparametric analysis of the results was performed. First, the Wilcoxon test was used to compare the methods performance, based on the mean values found, through the Matlab function `signrank`. Wilcoxon test is a nonparametric test that is used to determine whether two independent samples were selected from populations having the same distribution, therefore it can be employed to detect significant differences between the behaviour of two algorithms. The null hypothesis is that the difference between two sample means is zero. It emerged that the null hypothesis of equivalence of the two algorithm cannot be rejected at significance level $\alpha = 0.05$. Denoting with R^+ the sum of ranks for the problems in which SLP outperforms PSOc_3, and R^- the sum of ranks for the opposite, $R^+ = 33$ and $R^- = 58$ are obtained. Anyway it is worth remembering that the test is performed without taking into account that SLP solver does not find a feasible solution in all runs.

It is then possible to conclude that both solvers find good results, both in terms of mean values and in terms of runtime. PSO method manages to find a solution approximation requiring a really low number of function evaluations, that is about 150000 for three test cases (g07, g10, g19) but is much lower for the others: less than 50000 for g09 and g16 and less than 20000 for all the others test cases. SLP requires considerably less function evaluations despite n extra function evaluations for iteration are needed to approximate the gradient. Indeed, the maximum number of f-evaluations required is 1800 for g04. As problems dimensions are really small and a quite large number of particles is used for PSO method, the execution time for SLP is lower than for PSO.

The proposed methods were compared with seven hybrid procedures introduced in (de-los Cobos-Silva et al. 2016). These procedures, namely MMC-DE, MMC-DE-SC, MMC-DE-interleaved, MMC-DE-SC-interleaved, MMC-DE-batch, MMC-DE-SC-batch and PSO-3P-SC, are based on metaheuristic approaches. The comparison was carried out through pairwise comparison by the Wilcoxon test, using the statistics provided in (de-los Cobos-Silva et al. 2016). It emerges that the null hypothesis of equivalence should be rejected for the comparisons of PSOc_3 with MMC-DE-SC-interleaved, PSO-3P-SC at significance level $\alpha = 0.01$, and with MMC-DE at significance level $\alpha = 0.03$. PSOc_3 indeed shows an improvement over these methods, getting respectively $R^+ = 91, 78, 67$, where in this case R^+ is the sum of ranks for the problems in which PSOc_3 outperforms the method it is compared with. On the other hand SLP method results to be equivalent to all the hybrids, except for MMC-DE-SC-interleaved for which SLP shows an improvement at significance level $\alpha = 0.01$ and $R^+ = 77$, with R^+ the sum of ranks for the problems in which SLP outperforms MMC-DE-SC-interleaved. Then a Friedman test was performed to compare PSOc_3, PSO-3P-SC, MMC-DE, MMC-DE-SC-interleaved and SLP, followed by a post-hoc analysis through the Matlab function `multcompare`. This function, using the statistics provided by the Friedman test, performs $N \times N$ multiple comparisons, where N is the number of algorithms to be tested, and estimates the difference of average ranks (MathWorks 2017; King and Mody 2010). The result of the test is a matrix of multiple comparison results, returned as an p-by-6 matrix of scalar values, where p is the number of pairs. Each row of the matrix contains the result of one paired comparison test. The matrix obtained in the test is shown in Table 3. **Method 1** and **Method 2** denote the methods being compared, **lb**, **diff** and **ub** denotes respectively the lower confidence interval, the estimate, and the upper confidence interval, **p-value** is the p-value for the hypothesis test that the corresponding mean difference is not equal to 0. Then, in each row the numbers indicate that the mean of Method 1 minus the mean of Method 2 is estimated to be **diff**, and a 95% confidence interval for the true difference of the means is [**lb**, **ub**]. If the confidence interval contains 0, the difference is significant at the 5% significance level, otherwise it is not. The post-hoc analysis confirms all the differences detected by the Wilcoxon test, except the one between PSOc_3 and MMC-DE. It emerged indeed that PSOc_3 and PSO-3P-SC (p=0.0178), PSOc_3 and MMC-DE-SC-interleaved (p=0.001) and MMC-DE-SC-interleaved and SLP (p=0.0221) have mean ranks significantly different, as it is shown in Figure 1. In the figure, estimates and comparison intervals are shown for each method. Each method mean is represented by the symbol "o", and the interval is represented by a line extending out from the symbol. Two methods means are significantly different if their intervals are disjoint,

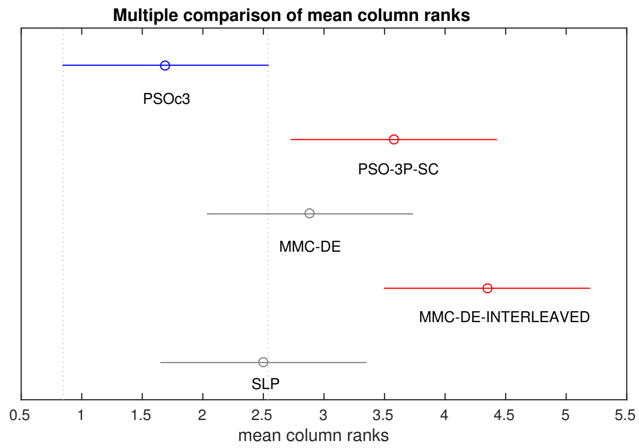


Figure 1. Post-hoc analysis through Matlab function `multcompare`.

while they are not significantly different if their intervals overlap, (MathWorks 2017; King and Mody 2010).

Table 3. Matrix of multiple comparisons from Matlab function `multcompare`.

Method 1	Method 2	lb	diff	ub	p-value
PSOc_3	PSO-3P-SC	-3.5301	-1.8846	-0.2391	**0.0154**
PSOc_3	MMC-DE	-2.8378	-1.1923	0.4532	0.2775
PSOc_3	MMC-DE-SC-inter	-4.2994	-2.6538	-1.0083	**0.0001**
PSOc_3	SLP	-2.4532	-0.8077	0.8378	0.6668
PSO-3P-SC	MMC-DE	-0.9532	0.6923	2.3378	0.7810
PSO-3P-SC	MMC-DE-SC-inter	-2.4147	-0.7692	0.8763	0.7066
PSO-3P-SC	5.0000	-0.5686	1.0769	2.7224	0.3822
MMC-DE	MMC-DE-SC-inter	-3.1071	-1.4615	0.1840	0.1092
MMC-DE	5.0000	-1.2609	0.3846	2.0301	0.9689
MMC-DE-SC-inter	5.0000	0.2006	1.8462	3.4917	**0.0188**

Finally, efficiency and reliability of the proposed version of PSO was compared against the basic version of PSO method introduced by Hu and Eberhart (2002) and to MEIGO-ESS proposed by Egea et al. (2014). MEIGO is a toolbox, whose Matlab code is available, in which an enhanced scatter search method (ESS), a population-based metaheuristic, is implemented. In the toolbox an option is available to apply a local search to the current population to increase the convergence to optimal solutions. In the experiments this option was not enabled, as it employs first-order information (approximated by finite differences) and therefore a comparison with PSO methods, that uses only function values, would not be fair.

To compare the codes they were applied to the problems in Table 1 and the performance profile approach in Moré and Wild (2009) was employed. The main idea behind the performance profile approach is briefly outlined here, for a more detailed description see Dolan and Moré (2002). In this approach, when m solvers are compared on a test set, the performance of each solver in the solution of a test is measured by the ratio of its computational effort and the best computational effort by any solver on this test. Here the number of performed function and constraints evaluations was used as a performance measure, they will be referred to as f-evaluations for sake of brevity. Specifically, for each

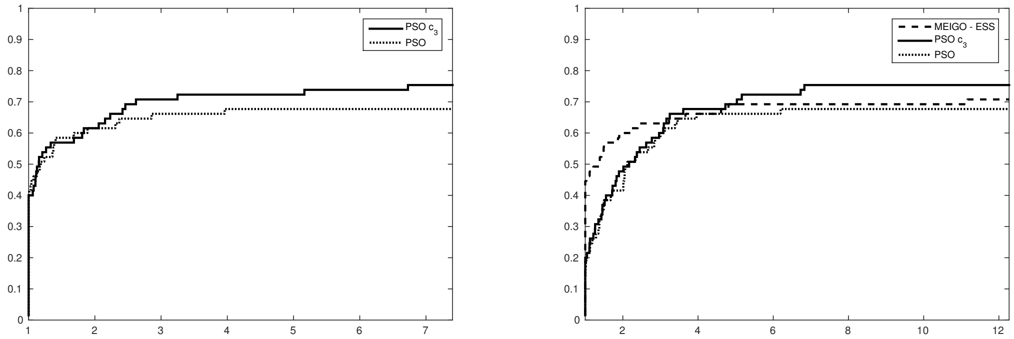


Figure 2. Performance profiles, based on number of function evaluations.

test t solved by the solver s, let $F_{s,t}$ denote the total number of f-evaluations required by the solver s to solve the test t. Moreover, let $\bar{F}_t$ denote the lowest number of function evaluations required by all the solvers to solve test t. Then, the ratio

$$f_{s,t} = \frac{F_{s,t}}{\bar{F}_t}$$

measures the performance of solver s on test t with respect to the best performance among all the solvers on such test. Then, the performance profile of solver s is defined as

$$\pi_s(\tau) = \frac{n^o\, tests\, s.t.\, f_{s,t} \le \tau}{n^o\, tests}, \qquad \tau \ge 1.$$

As a convergence test, the one proposed in Moré and Wild (2009) for derivative-free methods was used, that measures the decrease in function value:

$$f(x_0) - f(x) \ge (1 - \tilde{\tau})(f(x_0) - f_L), \tag{2}$$

where $\tilde{\tau} = 10^{-3}$ was chosen, x_0 is a starting point and for each problem f_L is the optimal known value of f, given in Liang et al. (2006).

Performance profiles are shown in Figure 2. The test set is constituted by ten runs of each problems in Table 1. A failure was declared when the maximum allowed number of f-evaluations was reached without (2) being satisfied. In the profiles on the left just PSOc_3 and the basic version of PSO in Hu and Eberhart (2002), labelled simply as PSO, are considered. The aim is to show the advantages of the new PSO algorithm proposed in this work compared to the basic version. On the right all the three algorithms are considered, to compare the proposed PSO method also to the state-of-the-art metaheuristic method MEIGO-ESS. Focusing on the comparison between PSO and PSOc_3, it can be noticed that the new strategy introduced here strengthen robustness of PSO. Indeed, PSOc_3 shows 75% of successful runs against 67% of PSO, as can be seen from the right side of the left plot. This shows that using the new velocity update the swarm is less likely trapped in local minima. Focusing on the comparison with MEIGO-ESS, the right plot shows that MEIGO-ESS is the most efficient solver on the 45% of runs and outperforms PSOc_3 in terms of efficiency, but PSOc_3 is more robust (MEIGO-ESS solves about 70% of the problems).

2. Convergence analysis for SLP method

In this section Algorithm 1, reported in the main paper, is considered. Theorem 4.2 of the main paper is proved, which states the global convergence of the sequence $\{x_k\}$ generated by Algorithm 1 to a stationary point of problem (14) (stated in the main paper). Here the problem is restated in a more general form, as the theorem is valid not only when l_1 penalty function is chosen, but also for all polyhedral penalty convex functions H (Fletcher and de la Maza 1989). Then, Φ is going to be expressed in the following general compact form: $\Phi(x;\nu) = f(x) + H(g(x;\nu))$, where $H(g(x;\nu))$ is the penalty term and $g : \mathbb{R}^n \to \mathbb{R}^p$ and, for sake of simplicity, in the following analysis $g(x;\nu)$ and $\Phi(x;\nu)$ will be denoted as $g(x)$ and $\Phi(x)$ respectively. Moreover for the step a generic norm $\|d_k\|$ can be considered, while, for seek of simplicity, bound constraints are not considered.

So, as a consequence of the above assumptions, theoretical results are referred to the following unconstrained problem:

$$\min_x \Phi(x) = f(x) + H(g(x)), \tag{3}$$

and subproblem (15) of the main paper becomes:

$$\min_d l_k(d) = f(x_k) + \nabla f(x_k)^T d + H(g(x_k) + \nabla g(x_k)^T d) \tag{4a}$$

$$\|d\| \leq \Delta_k, \tag{4b}$$

where $\nabla g(x) \in \mathbb{R}^{p\times n}$ is the Jacobian matrix of $g(x)$.

To prove the theorem it is necessary to take into account that objective functions of problems (3) and (4) are non differentiable, so to characterize their stationary points it is necessary to introduce KKT conditions for problems with non-smooth objective function, (Fletcher 1987).

Definition 2.1 *Let f be a convex function defined in $\mathcal{D} \subseteq \mathbb{R}^n$. A vector v is a subgradient of f at $x \in \mathcal{D}$, if $f(y) \geq f(x) + v^T(y-x)$ for all $y \in \mathcal{D}$.*

Definition 2.2 *Let f be a convex function defined in $\mathcal{D} \subseteq \mathbb{R}^n$. The subdifferential $\partial f(x)$ of f at x is the set of all subgradients:*

$$\partial f(x) = \{v : v^T(y-x) \leq f(y) - f(x) \quad \textit{for all} \quad y \in \mathcal{D}\}.$$

First order necessary KKT conditions for x^* to solve (3) are that there exists vectors of multipliers $\lambda^* \in \partial H(g^*)$ such that, (see Theorem 14.2.1 in (Fletcher 1987)): $\nabla f(x^*) + \nabla g(x^*)\lambda^* = 0$. First order conditions for subproblem (4) are that there exist multipliers $\lambda_k \in \partial H(g(x_k) + \nabla g(x_k)^T d_k)$, $w_k \in \partial\|d_k\|$ and $\pi_k \geq 0$, (see Theorem 14.6.1 in (Fletcher 1987)), such that:

$$\nabla f(x_k) + \nabla g(x_k)^T \lambda_k + \pi_k w_k = 0, \tag{5}$$

$$\pi_k(\|d_k\| - \Delta_k) = 0. \tag{6}$$

The following Lemma is proved in (Fletcher 1987) and is useful for the convergence theorems.

Lemma 2.3 *[Lemma 14.2.1 of (Fletcher 1987)] Let $f : \mathcal{K} \to \mathbb{R}$ be a convex function, $\mathcal{K} \subset \mathbb{R}^n$ a convex set. Then $\partial f(x)$ is a closed convex set and it is bounded for all $x \in$*

$B \subset \overset{o}{\mathcal{K}}$ *where* B *is compact and* $\overset{o}{\mathcal{K}}$ *denotes the interior of* $\mathcal{K}$.

The following theorem is the reformulation of Theorem 4.2 of the main paper in a slightly more general form. It is proved following the lines of the proof of Theorem 2.1 in (Fletcher and de la Maza 1989). Note that in (Fletcher and de la Maza 1989) it is assumed to have at disposal an approximation of the Hessian matrix of the Lagrangian function, while the method considered in this article exploits just first order informations.

THEOREM 2.4 *[Global convergence of Algorithm 1]*
Let f *and* g *be* $\mathbb{C}^1$ *functions and let* $H(g)$ *be a convex function. Let* $\{x_k\}$ *be the sequence generated by Algorithm 1. Either there exists an iteration index* $\bar{k}$ *such that* $x_{\bar{k}}$ *is a KKT point for* $\Phi(x)$*, or* $\Phi(x_k) \to -\infty$ $k \to \infty$*, or if the sequence* $\{x_k\}$ *is bounded, then there exists a subsequence* S *of indexes such that* $\{x_k\}_{k\in S}$ *has an accumulation point* x^* *which satisfies the KKT conditions for* $\Phi(x)$*, that is it exists a vector of multipliers* λ^* *such that:*

$$\nabla f(x^*) + \nabla g(x^*)\lambda^* = 0. \tag{7}$$

Proof. To prove the theorem it is sufficient to consider the case in which $\{\Phi_k\}$ is bounded below and $\{x_k\}$ is bounded. Because $\{x_k\}$ is bounded, there exists a subsequence S of iterations such that $\{x_k\}_{k\in S} \to x^*$. Suppose that:
a) d_k does not satisfy $\rho_k > \rho_{bad}$ for any $k \in S$ and $\{\Delta_k\}_{k\in S} \to 0$ and hence $\{\|d_k\|\}_{k\in S} \to 0$.
Let define $\Delta\Phi_k = \Phi(x_k) - \Phi(x_k + d_k)$ and $\Delta l_k = l_k(0) - l_k(d_k) = \Phi(x_k) - l_k(d_k)$. A consequence of $\mathbb{C}^1$ continuity of f and g, convexity of $H(g)$ and boundedness of $\partial H(g)$, which follows from Lemma 2.3, and of the use of the first order Taylor expansion, is that: $\Delta\Phi_k = \Delta l_k + o(\|d_k\|)$ and hence $\Delta\Phi_k/\Delta l_k \to 1$ as $k \to \infty$, which contradicts the fact that $\rho_k = \frac{\Delta\Phi_k}{\Delta l_k} > \rho_{bad}$ fails for all $k \in S$. Therefore this case is inconsistent and it certainly exists a subsequence of indexes S such that:
b) d_k satisfies $\rho_k > \rho_{bad}$ and $\liminf_{k\in S} \Delta_k > 0$.
In fact, let S' be a sequence of indexes of unsuccessful iterations. If S' is finite, then clearly $\rho_k > \rho_{bad}$ for k sufficiently large. Otherwise suppose $k_0 \in S'$. Since $\{\Delta_k\}_{k\in S'} \nrightarrow 0$, otherwise case (a) is obtained again, for each $k_0 \in S'$ it exists i such that $k_0 + i$ is a successful iteration with $k_0 + i \notin S'$. Therefore $x_{k_0+i} = x_{k_0}$ and the subsequence $\{x_{k_0+i}\}_{k_0\in S'} \to x^*$. Let $S = \{k_0 + i, k_0 \in S'\}$, then $\{x_k\}_{k\in S}$ is the subsequence of case (b). In case (b) it can be assumed that $\liminf_{k\in S} \Delta_k > \bar{\Delta} > 0$, as if $\liminf_{k\in S} \Delta_k = 0$ thus imply $\liminf \Delta_k = 0$ and this yields case (a) that has be proved to be inconsistent. Because $\Phi_1 - \Phi^* \geq \sum_{k\in S} \Delta\Phi_k$, it follows that $\sum_{k\in S} \Delta\Phi_k$ converges. Then, $\rho_k \geq \rho_{bad}$, i.e. $\Delta\Phi_k \geq \Delta l_k \rho_{bad}$, yields the convergence of the series $\sum_{k\in S} \Delta l_k$, and hence $\{\Delta l_k\} \to 0$.
Define $l^*(d) = f(x^*) + \nabla f(x^*)d + H(g(x^*) + \nabla g(x^*)^T d)$. Let

$$\bar{d} = \arg\min l^*(d), \quad s.t. \ \|d\| \leq \bar{\Delta}$$

and denote $\bar{x} = x^* + \bar{d}$. Then

$$\|\bar{x} - x_k\| \leq \|\bar{x} - x^*\| + \|x^* - x_k\| = \|\bar{d}\| + \|x^* - x_k\| \leq \bar{\Delta} + \|x^* - x_k\| \leq \Delta_k$$

for all k sufficiently large, $k \in S$. Thus $\bar{x}$ is feasible for problem (4), so $l_k(\bar{x} - x_k) \geq l_k(d_k) = \Phi(x_k) - \Delta l_k$. In the limit, for $k \in S$, $\nabla f(x_k) \to \nabla f(x^*)$, $g(x_k) \to g(x^*), \nabla g(x_k) \to \nabla g(x^*), \bar{x} - x_k \to \bar{d}$, and $\Delta l_k \to 0$, so it follows that $l^*(\bar{d}) \geq \Phi(x^*) = l^*(0)$. Thus $d = 0$ also minimizes $l^*(d)$ subject to $\|d\| \leq \bar{\Delta}$, and since the latter constraint is not active it follows from (6) that $\pi^* = 0$ and from (5) that it exists λ^* such that $\nabla f(x^*) + \nabla g(x^*)\lambda^* = 0$, then x^* is a KKT point. □

Theorem 2.4 states that in case the objective function is not unbounded, which is ensured in the particular case considered in the main paper, due to the presence of the bound constraints, it exists an accumulation point x^* of the sequence $\{x_k\}$ generated by Algorithm 1 that satisfies KKT conditions (7) for $\Phi(x)$, regardless of the starting point.

It is also possible to prove the convergence of multipliers, i.e. if the subsequence S of Theorem 2.4 exists, than the subsequence of multipliers of subproblems (4) approximates multipliers of problem (3). In fact, the following theorem holds:

Theorem 2.5 Convergence of multipliers, Theorem 2.2 in (Fletcher and de la Maza 1989) *Let $f, g \in \mathbb{C}^1$, $H(g)$ a convex function, π_k and λ_k multipliers of subproblems* (4), *defined in* (5) *and* (6). *If the subsequence S in the statement of Theorem 2.4 exists, then $\{\pi_k\}_{k\in S} \to 0$. Moreover any accumulation point λ^* of the multiplier vectors λ_k, $k \in S$, satisfies $\lambda^* \in \Lambda^*$, where $\Lambda^* = \{\lambda : \lambda$ satisfies KKT conditions* (7) *at $x^*\}$, and such an accumulation point exists.*

The stopping criterion employed for Algorithm 1, that is described in Section 4.1 of the main paper, relies on this important theoretical result, that allows to use the multipliers of subproblems (4), provided by the function used to solve the LPs, as an approximation of those of problem (3).

References

Das, S., and P.N. Suganthan. 2011. "Differential Evolution: A Survey of the State-of-the-Art." *IEEE Transactions on Evolutionary Computation* 15 (1): 4–31.

de-los Cobos-Silva, Sergio Gerardo, Roman Anselmo Mora-Gutiérrez, Miguel Angel Gutiérrez-Andrade, Eric Alfredo Rincón-García, Antonin Ponsich, and Pedro Lara-Velázquez. 2016. "Development of Seven Hybrid Methods Based on Collective Intelligence for Solving Nonlinear Constrained Optimization Problems." *Artificial Intelligence Review* 1–35.

Derrac, J., S. García, D. Molina, and F. Herrera. 2011. "A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms." *Swarm and Evolutionary Computation* 1 (1): 3–18.

Dolan, E.D., and J.J. Moré. 2002. "Benchmarking optimization software with performance profiles." *Mathematical Programming* 91 (2): 201–213. https://doi.org/10.1007/s101070100263.

Egea, J.A., D. Henriques, T. Cokelaer, A.F. Villaverde, A. MacNamara, D.P. Danciu, J.R. Banga, and J. Saez-Rodriguez. 2014. "MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics." *BMC Bioinformatics* 15 (1): 136. https://doi.org/10.1186/1471-2105-15-136.

Fletcher, R. 1987. "Practical Methods of Optimization." *Wiley-Interscience* .

Fletcher, Roger, and E Sainz de la Maza. 1989. "Nonlinear programming and nonsmooth optimization by successive linear programming." *Mathematical Programming* 43 (1-3): 235–256.

García, S., D. Molina, M. Lozano, and F. Herrera. 2009. "A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms Behaviour: a Case Study on the CEC2005 Special Session on Real Parameter Optimization." *Journal of Heuristics* 15 (6): 617–644.

Hu, X., and R. Eberhart. 2002. "Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization." In *6th world multiconference on systemics, cybernetics and informatics,* 203–206.

King, Michael R, and Nipa A Mody. 2010. *Numerical and statistical methods for bioengineering: applications in MATLAB.* Cambridge University Press.

Liang, J.J., T. P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello, and K. Deb. 2006. "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization." *Journal of Applied Mechanics* 41 (8).

Liu, J., K. L. Teo, X. Wang, and C. Wu. 2016. “An Exact Penalty Function-Based Differential Search Algorithm for Constrained Global Optimization.” *Soft Computing* 20 (4): 1305–1313.

MathWorks. 2017. “Multiple Comparison Test.” https://it.mathworks.com/help/stats/multcompare.html.

Mezura-Montes, E., and B.C. Lopez-Ramirez. 2007. “Comparing Bio-Inspired Algorithms in Constrained Optimization Problems.” In *2007 IEEE Congress on Evolutionary Computation,* 662–669. Sept.

Moré, J.J., and S.M. Wild. 2009. “Benchmarking Derivative-Free Optimization Algorithms.” *SIAM Journal on Optimization* 20 (1): 172–191. https://doi.org/10.1137/080724083. https://doi.org/10.1137/080724083.