RAIRO-Oper. Res.
https://doi.org/10.1051/ro/2019096

RAIRO Operations Research
www.rairo-ro.org

A METAHEURISTIC PENALTY APPROACH FOR THE STARTING POINT IN NONLINEAR PROGRAMMING

David R. Penas[1,*] and Marcos Raydan[2]

Abstract. Solving nonlinear programming problems usually involve difficulties to obtain a starting point that produces convergence to a local feasible solution, for which the objective function value is sufficiently good. A novel approach is proposed, combining metaheuristic techniques with modern deterministic optimization schemes, with the aim to solve a sequence of penalized related problems to generate convenient starting points. The metaheuristic ideas are used to choose the penalty parameters associated with the constraints, and for each set of penalty parameters a deterministic scheme is used to evaluate a properly chosen metaheuristic merit function. Based on this starting-point approach, we describe two different strategies for solving the nonlinear programming problem. We illustrate the properties of the combined schemes on three nonlinear programming benchmark-test problems, and also on the well-known and hard-to-solve disk-packing problem, that possesses a huge amount of local-nonglobal solutions, obtaining encouraging results both in terms of optimality and feasibility.

Mathematics Subject Classification. — Please, give AMS classification codes —.

Received March 20, 2019. Accepted September 29, 2019.

1. Introduction

One of the most important and delicate issues when solving nonlinear programming problems (NLP) is the choice of the starting point, often called the initial guess. In general, NLP have many local optima spread out over the feasible region that, due to the way in which nonlinear constraints can twist and curve, might be the union of multiple different and disconnected (also called discontiguous) not necessarily convex regions. Usually, the numerical optimization techniques iteratively improve the initial guess to converge to a near local solution. Consequently, the starting point determines not only in what parts of the disconnected feasible region the iterates will live, but also to which optimum the algorithm will converge.

An approach that has been extensively studied for several applications and incorporated in software packages is to explore in advance as much as possible the feasible region using many different starting points. These initial points can be generated either randomly, which promotes the use of stochastic heuristic schemes, or using the so-called grid search techniques, that consists in choosing a set of points evenly distributed throughout the space. Then a quick local search is performed starting at each one of them to get a set of points from which

Keywords. Nonlinear programming problems, Starting point strategy, Metaheuristics, Penalty methods, Disk packing problem.

[1] Modestya Research Group, Department of Statistics, Mathematical Analysis and Optimization, Institute of Mathematics (IMAT), University of Santiago de Compostela, Santiago, Spain.

[2] Centro de Matemática e Aplicações (CMA), FCT, UNL, 2829-516 Caparica, Portugal.

*Corresponding author: david.rodriguez.penas@usc.es

Article published by EDP Sciences

the best one, concerning optimality and feasibility, is chosen; see, *e.g.*, [2, 3, 9, 20, 23–25, 31, 32, 37, 40]. Due to the size of the space, for medium to large-scale problems, these type of preliminary searching schemes require a prohibitive amount of computational work, and as a consequence they become impractical.

There are two issues involved when choosing the starting point: feasibility and optimality, which can be considered simultaneously if we use a suitable penalty approach. Roughly speaking, a penalty method replaces a constrained optimization problem by a series of unconstrained problems whose solutions ideally converge to the solution of the original constrained problem. The unconstrained problems are formed by adding a term, called a penalty function, to the objective function that consists of several penalty parameters multiplied by measures of violation of the constraints. The measures of violation are nonzero when the associated constraints are violated, and are zero in the regions where constraints are not violated. See, *e.g.*, Luenberger [21] and Fiacco and McCormick [6] for details and for a general discussions on penalty methods.

Our proposal in this work, for choosing the starting point, is to perform a preliminary exploratory work based on a properly chosen penalty function for which the penalty parameters, instead of the space variables, are chosen stochastically using metaheuristic schemes, *i.e.*, using iterative generation processes that guide subordinate heuristics. A numerical unconstrained deterministic scheme can then be applied to this penalty function, using as a merit function for the metaheuristic algorithm a proper combination of the objective value of the original NLP objective function and a feasibility measure. That way, starting at the same neutral point and allowing the same CPU time or the same fixed small number of iterations to the same unconstrained optimization method, for each combination of penalty parameters, we identify the point that at the end achieves the best value for the merit function, and declare it the starting point to solve the NLP. For that it is fundamental to use only a few penalty parameters, to guarantee that the heuristic techniques work effectively in a low-dimensional space. Therefore, a key aspect of our approach is to divide the constraints into a few groups, and assign a penalty parameter to each one of the groups.

The rest of this work is organized as follows. In Section 2, we present the general NLP to be considered, we develop in detail the metaheuristic penalty approach for choosing the starting point, and we describe two different strategies for solving the NLP problem. In Section 3, we describe the three suitable metaheuristic schemes to be considered: Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Particle Swarm Optimization (PSO), and Simulated Annealing (SA). In Section 4, we describe the well-known disk packing problem and also the benchmark test problems to be used for illustrating the advantages of our approach, and we present and analyze the obtained experimental results, when using the three considered metaheuristic schemes, for several different dimensions and different weights of feasibility and optimality in the merit function. Finally, in Section 5 we present some final comments and conclusions.

2. Penalty approach and combined strategies

We are interested in solving nonlinear programming problems (NLP) of the form:

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & h(x) = 0 \\ & g(x) \leq 0, \end{array} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$, and $g : \mathbb{R}^n \to \mathbb{R}^s$ are sufficiently smooth. We are mainly concerned with large-scale NLP problems for which the user cannot supply a good starting point, but nevertheless is interested in obtaining a local feasible minimizer for which the function f reaches a sufficiently low value.

First, let us recall the main ideas of the classical and straightforward penalty approach for solving problem (2.1); see, *e.g.*, [6, 21]. Penalty methods transform constrained optimization problems into a sequence of unconstrained subproblems, whose solutions ideally converge to a solution of the original optimization problem. In that case, NLP problem (2.1) can be reduced to a sequence of unconstrained problems of the following form:

$$\min f(x) + \rho_k P(x), \quad x \in \mathbb{R}^n, \tag{2.2}$$

where $\rho_k > 0$ is the penalty parameter that increases at every k to penalize constraint violations, and the penalty function $P : \mathbb{R}^n \to \mathbb{R}$ is a continuous function that involves the constraints in such a way that $P(x) \geq 0$ for all $x \in \mathbb{R}^n$, and such that it enforces feasibility, *i.e.*, $P(x) = 0$ if and only if x is a feasible point for problem (2.1). Under some mild assumptions and some specific choices of P and the sequence $\{\rho_k\}$, it has been established that the sequence of solutions of problem (2.2) converges to a solution of (2.1) when k goes to infinity [6,21]. However, in practice, when the parameter $\rho_k > 0$ reaches very high values the convergence might be extremely slow and moreover the problem becomes ill-conditioned. That is why we are not interested in using this approach all the way to achieve convergence to a solution of problem (2.1), when k goes to infinity, but instead to use it as a framework to apply metaheuristic schemes on the choice of the penalty parameters, combined with very few iterations of an unconstrained numerical solver, to produce a good initial guess. For that, we will be interested in considering a vector $\rho \in \mathbb{R}^m$ of few penalty parameters, where $m \ll \min\{n, p+s\}$ and $\rho_i > 0$ for $1 \leq i \leq m$.

For our approach, there are many different ways of properly extending the classical penalty functions P which have been used for solving (2.1), including some ideas that combine evolutionary algorithms to explore the whole space of variables $\mathbb{R}^n$; see, *e.g.*, [4,15,26,39]. In particular, we will consider the following choice:

$$P(x;\rho) = \widehat{\rho}_1 \sum_{i=1}^{p_1} |h_i(x)|^\beta + \widehat{\rho}_2 \sum_{i=p_1+1}^{p_2} |h_i(x)|^\beta + \cdots + \widehat{\rho}_{m_1} \sum_{i=p_{m_1-1}+1}^{p} |h_i(x)|^\beta \tag{2.3}$$

$$+ \bar{\rho}_1 \sum_{i=1}^{s_1} \max(0, g_i(x))^\gamma + \bar{\rho}_2 \sum_{i=s_1+1}^{s_2} \max(0, g_i(x))^\gamma + \cdots + \bar{\rho}_{m_2} \sum_{i=s_{m_2-1}+1}^{s} \max(0, g_i(x))^\gamma,$$

where $m_1 + m_2 = m$, the integer constant powers β and γ can be either 1 or 2, and the vector ρ of penalty parameters is given by

$$\rho = [\widehat{\rho}_1, \widehat{\rho}_2, \ldots, \widehat{\rho}_{m_1-1}, \widehat{\rho}_{m_1}, \bar{\rho}_1, \bar{\rho}_2, \ldots, \bar{\rho}_{m_2-1}, \bar{\rho}_{m_2}]^T.$$

Notice in (2.3) that the term $|h_i(x)|^\beta$, as well as the term $\max(0, g_i(x))^\gamma$, is strictly positive when the related ith constraint is being violated at the vector x. Moreover, the larger the value of that term the larger the violation. Notice also that the constraints are grouped ideally based on common characteristics, usually identified by the user, and each equality or inequality constraint belongs to one and only one group, and each group is penalized by a different penalty entry of the vector ρ. To be precise, the equality constraints $h_i(x)$ for $1 \leq i \leq p_1$ are penalized by $\widehat{\rho}_1$, the ones for $p_1 + 1 \leq i \leq p_2$ are penalized by $\widehat{\rho}_2$, and so on until the last group which is penalized by $\widehat{\rho}_{m_1}$. Similarly, the inequality constraint groups are penalized by the penalty entries $\bar{\rho}_1$ up to $\bar{\rho}_{m_2}$.

In our strategy, the entries of the penalty vector ρ will be automatically trained to produce a good starting point for NLP problems using stochastic metaheuristic schemes. For that we need a convenient merit function, given by

$$\widehat{f}(\rho) = \alpha f(x(\rho)) + (1 - \alpha) P(x(\rho); e), \tag{2.4}$$

where $0 < \alpha < 1$ is fixed, $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^{p+s}$, the optimality is clearly measured by the NLP objective function $f(x(\rho))$, the feasibility is measured by

$$P(x(\rho); e) = \sum_{i=1}^{p} |h_i(x(\rho))|^\beta + \sum_{i=1}^{s} \max(0, g_i(x(\rho)))^\gamma,$$

and $x(\rho) \in \mathbb{R}^n$ is obtained after applying a fixed small number of iterations of the same unconstrained numerical minimization solver to $f(x) + P(x;\rho)$, starting from the same initial guess.

2.1. Starting point strategy

Our starting point strategy is based on the following connection between the metaheuristic and the merit function. A metaheuristic algorithm is used iteratively to minimize the merit function given by (2.4), which is

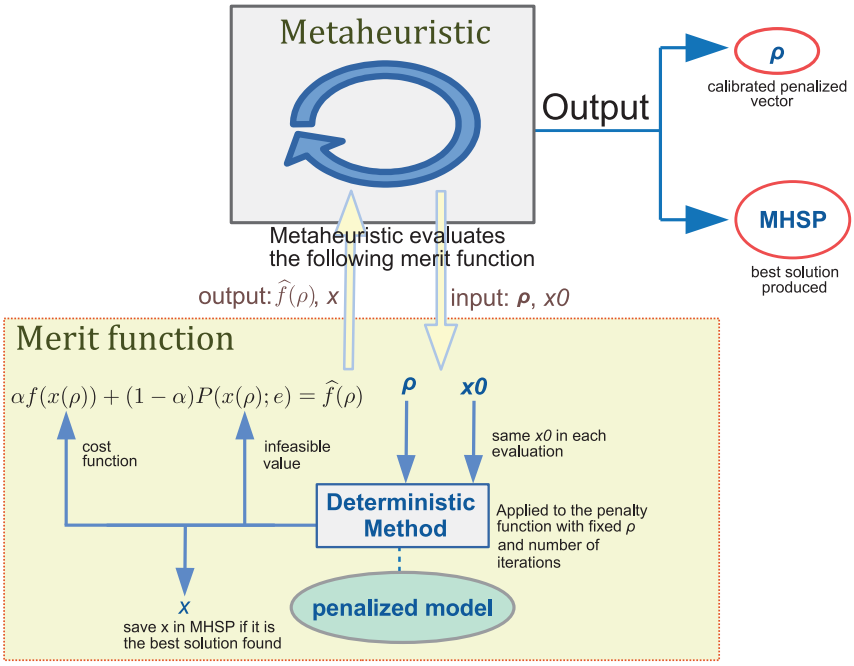

FIGURE 1. Connection between the metaheuristic and the merit function.

used as a black box, that is, the metaheuristic plays with the values of the penalty vector ρ without having any knowledge about what is happening inside the evaluation process of the merit function. Each time the metaheuristic needs to evaluate a specific penalty vector $\rho \in \mathbb{R}^m$, the merit function returns a fitness $(\widehat{f}(\rho))$ value. Throughout this optimization process, the heuristic algorithm will try to obtain a combination of values for the penalty parameters with the minimum associated value of $\widehat{f}(\rho)$ given by (2.4). Figure 1 shows a diagram of how the combined process works.

To evaluate the merit function at a given vector ρ, a call to a deterministic unconstrained numerical method is carried out internally, starting at the same neutral initial guess (called x_0 in the diagram), and for the same fixed small number of iterations. As a result it returns the infeasibility and the cost function used to obtain the value of the merit function given by (2.4). Finally, when the termination conditions of the metaheuristic are fulfilled, both the calibrated penalty vector ρ and the associated best solution $x(\rho)$ are returned. From now on, the best found solution $x(\rho)$ will be denoted as the Metaheuristic Starting Point (MHSP).

2.2. Combined strategies for solving the NLP problem

Once a metaheuristic algorithm is applied, as described in Section 2.1, generating the best found solution (called MHSP in Figs. 1 and 2), we can proceed using one of the two following strategies for solving the NLP problem (2.1): (see also Fig. 2)

- Strategy A: using MHSP as the starting point, a deterministic constrained numerical optimization method is applied to the original model given by (2.1) until convergence is achieved, and the generated solution is called *Solution1* (see Fig. 2).
- Strategy B: fixing the calibrated vector ρ generated by the metaheuristic (see Fig. 1), and using MHSP as the starting point, a deterministic unconstrained optimization numerical method is applied to the penalized model $P(x;\rho)$ (given by (2.3)) until convergence is achieved, and the generated solution is called *Solution2* (see Fig. 2). Then, using *Solution2* as the starting point, a deterministic constrained numerical optimization

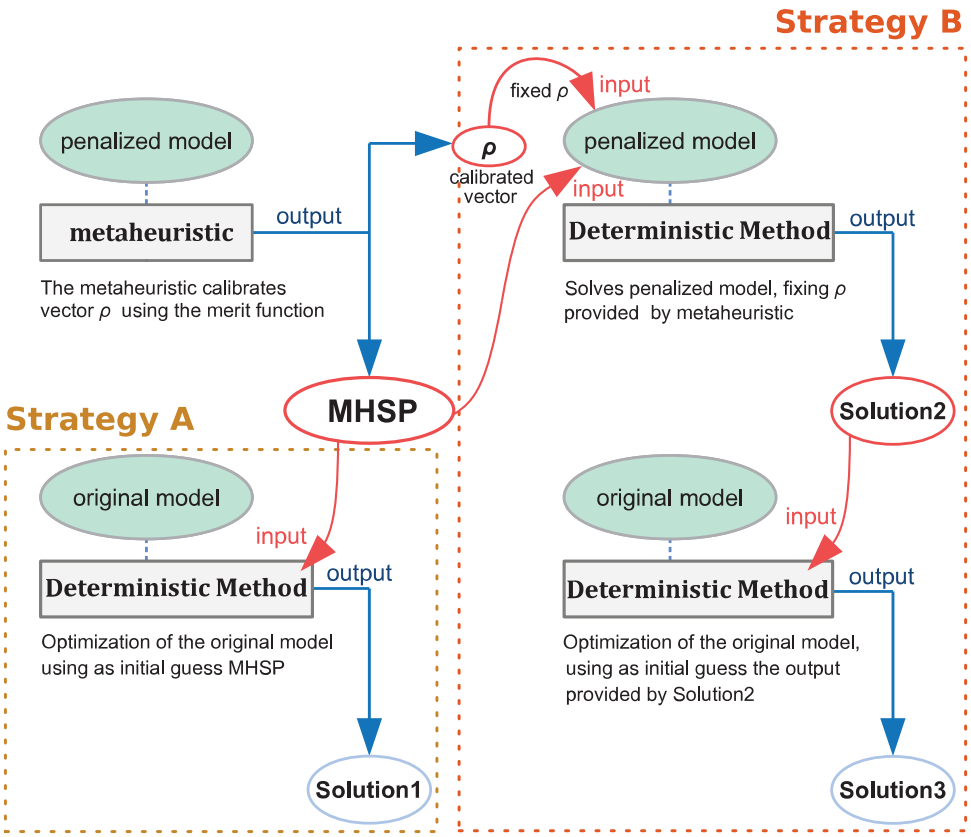

FIGURE 2. Two different strategies that use the starting point MHSP to solve the NLP problem.

method is applied to the original model given by (2.1) until convergence is achieved, and the generated solution is called *Solution3* (see Fig. 2).

Notice that, once MHSP has been obtained, the processes of computing *Solution1* and *Solution3* are independent. Both strategies (A and B) will be tested and compared in Section 4.

3. METAHEURISTIC ALGORITHMS

In the last two decades, metaheuristic algorithms have gained significant attention as efficient solvers for hard global optimization problems appearing in real engineering and science modeling applications. A metaheuristic is a higher-level procedure designed to find or generate a heuristic that may provide a sufficiently good solution to an optimization problem, especially when the set of solutions is too large to be fully sampled; see, *e.g.*, [22, 27, 28, 36]. The central common feature of all heuristic optimization methods is that they start off with a more or less arbitrary initial guess, iteratively produce new solutions by combining randomness and a generation rule, evaluate these new solutions using a suitable merit function, and eventually report the best solution found during the search process; see, *e.g.*, [2, 31]. The execution of the iterated search procedure is usually stopped when there has been no further improvement over a given number of iterations (or further improvements cannot be expected); when the found solution is good enough; or when the allowed CPU time (or some other external limit) has been reached.

For our specific goal of combining a metaheuristic algorithm with a penalty approach for finding a good starting point to solve NLP problem, we consider three state-of-the-art metaheuristic schemes: Evolution Strategies and Covariance Matrix Adaptation, Particle Swarm Optimization, and Simulated Annealing.

3.1. Covariance Matrix Adaptation Evolution Strategies (CMA-ES)

CMA-ES is a stochastic method that belongs to the general family of evolutionary algorithms for which the search strategy is not based on the use of function derivatives, but instead it is only based on the use of objective function evaluations, that in our case is given by $\widehat{f}$ in (2.4). In general, an initial population is generated, and then at every cycle new solution candidates or offsprings are generated from the solutions already stored in the population matrix. The new candidates are obtained by using random biologically-inspired combinations of their parents (crossover), and also by using random mutations; see, *e.g.*, [12–14]. At every cycle, based on the objective function evaluation, the new candidates are either accepted to be part of the current population, or rejected. An attractive feature of CMA-ES is that a covariance matrix is used to control the generation of new candidates. Moreover, the CMA-ES approach includes a dynamic self-updating process of the required parameters which are used for the generation of offsprings. Algorithm 1 summarizes the main steps of our specific CMA-ES metaheuristic approach.

Algorithm 1 Covariance Matrix Adaptation pseudocode.

procedure CMA( )
  $\tilde{\rho} = [\rho_1 \cdots \rho_\lambda] \leftarrow$ initial population matrix (size $m \times \lambda$)
  μ, C and $\sigma \leftarrow$ initial parameters
  while stopping conditions $\neq$ true **do**
    for $i = 1$ *to* λ **do**
      $\rho_i = N(\mu, \sigma^2 \mathrm{C})$ ▷ multivariate normal distribution
      $f_i = \widehat{f}(\rho_i)$ ▷ evaluate the new random solution generated
    end for
    $sort\_\tilde{\rho} \leftarrow$ Sort ρ_i's using f_i's
    $\mu \leftarrow$ update distribution mean using $sort\_\tilde{\rho}$
    $bestsol \leftarrow$ select best solution from $sort\_\tilde{\rho}$
    C $\leftarrow$ update and adapt the covariance matrix
    $\sigma \leftarrow$ update and adapt the variance
  end while
  return *bestsol*
end procedure

Notice that, in Algorithm 1, $\tilde{\rho}$ is the $m \times \lambda$ population matrix, C is the covariance matrix, μ is a vector with the mean distribution, keeping a weighted sum of the components of $\tilde{\rho}$ (in descending order, *i.e.*, the first elements of the vector have more weight than the last elements), σ is the variance of the population matrix, f is the fitness vector corresponding to the current solutions, and *bestsol* is the best solution found by the searching process.

3.2. Particle Swarm Optimization (PSO)

PSO is a heuristic optimization method which is biologically-inspired to simulate the behavior of swarms, for example, a flock of birds or a school of fish. It starts with a random population of candidate solutions, also called particles, and iteratively moves these particles around in the search-space according to simple mathematical formulae over the particle's current position and velocity. Each particle's movement is influenced by its local best known position, but it is also guided towards the best known global positions in the search-space, which are both updated as better positions are found by other particles. This scheme is expected to move the swarm towards the best solutions. The parameters or weights related to the local best known position and the global

one can be dynamically and stochastically updated during the search process; see *e.g.*, [16, 17, 29]. PSO shares many similarities with evolutionary techniques: the iterative process is initialized with a population of random solutions, it searches for optima by updating generations, and function derivatives are not required. However, unlike evolutionary schemes, PSO has no evolution operators such as crossover and mutation. Algorithm 2 summarizes the main steps of our specific PSO metaheuristic approach.

Algorithm 2 Particle Swarm Optimization pseudocode.

procedure PSO( )
  $\tilde{\rho} = [\rho_1 \cdots \rho_\lambda] \leftarrow$ initial population matrix of particles (size $m \times \lambda$)
  φ_l, φ_g and $\omega \leftarrow$ initial parameters
  Set $v = 0 \leftarrow$ size $m \times \lambda$
  for each $\rho_i \in \tilde{\rho}$ **set** $local_i = \rho_i$ **and** $flocal_i = \widehat{f}(\rho_i)$
  Set $fglobal =$ best $flocal_i$ **and** $global =$ best $local_i$
  while stopping conditions $\neq$ true **do**
    for $i = 1$ *to* λ **do**
      for $k = 1$ *to* m **do**
        $v_{i,k} = \omega v_{i,k} + \varphi_l r_l\ (local_{i,k}$ - $\rho_{i,k}) + \varphi_g r_g\ (global_k$ - $\rho_{i,k})$
        $new\_\rho_k = \rho_{i,k} + v_{i,k}$
      end for
      $f_i = \widehat{f}(new\_\rho)$
      if f_i improves $flocal_i$ **then** $local_i = new\_\rho$ **and** $flocal_i = f_i$
      if f_i improves $fglobal$ **then** $global_i = new\_\rho$ **and** $fglobal = f_i$
      $\rho_i = new\_\rho$
    end for
  end while
  return *global*
end procedure

Notice that, in Algorithm 2, $\tilde{\rho}$ is a set of random generated solutions, ρ_i is a particle or solution that belongs to $\tilde{\rho}$, *global* is the best solution found at the end of the search, $local_i$ is the best solution which can be reached from ρ_i, v_i is the accumulative vector of velocity, and r_l and r_g are two random numbers in the interval $[0, 1]$. Finally, there are three tunable parameters: the inertia weight (ω), the cognitive constant (φ_l), and the social constant (φ_g).

3.3. Simulated Annealing (SA)

SA is an iterative stochastic optimization technique that mimics the crystallization process of steel or glass during cooling or annealing: When the material is hot, the particles have high kinetic energy and move randomly regardless of their and the other particles positions. The cooler the material gets, however, the more the particles are moved towards the direction that minimizes the energy (temperature). The SA algorithm does the same when searching for the optimal values of the variables involved: It repeatedly suggests random modifications to the current solution, but progressively keeps only those that improve (cool down) the current situation, until the particles find a position of thermal equilibrium (global optimum); see *e.g.*, [8, 18, 34]. An interesting feature is that SA applies a probabilistic rule to decide whether the new solution replaces the current one or not. This rule considers the change in the objective function or equivalently the temperature which reflects the progress during the iterations. Algorithm 3 summarizes the main steps of our specific SA metaheuristic approach.

Algorithm 3 Simmulated Annealing pseudocode.

procedure SA( )
  $\rho \leftarrow$ initial solution
  $T = T_{\max}$
  $\Delta T \leftarrow$ cooling factor ▷ *i.e.* $\Delta T = T * 0.1$
  while $T > T_{\min}$ **and** stopping conditions $\neq$ true **do**
    $\rho' \leftarrow$ random selection of a neighbor solution
    $\Delta E = \widehat{f}(\rho')$ - $\widehat{f}(\rho)$
    if $\Delta E < 0$ **then**
      $\rho = \rho'$
    else
      $prob = e^{-\frac{\Delta E}{T}}$ ▷ An example of acceptance function
      $random \leftarrow$ random number between 0 and 1
      if $prob > random$ **then**
        $\rho = \rho'$
      end if
    end if
    $T = T * \Delta T$
  end while
  return ρ
end procedure

Notice that, in Algorithm 3, ρ is the current solution, ρ' is the trial solution created for each iteration, T is the temperature, *prob* is the acceptance function that depends on ΔE and T, ΔT is the cooling factor to reduce T at each iteration, and *random* is randomly generated in the interval $(0, 1)$.

4. Experimental results

To illustrate the properties of the strategies developed in Section 2, when combined with the metaheuristics described in Section 3, we will consider the well-known disk packing problem and some nonlinear programming benchmark-test problems. For the deterministic numerical optimization routines we use the solver package Knitro 11.0.1 [1] to obtain the penalty vector $\rho \in \mathbb{R}^m$ and the starting point, as well as to solve the NLP problem (4.1), once the starting point has been obtained. Knitro is a software package that implements a nonlinear interior method; for additional details see [3]. For comparisons, we report the performance results of our schemes and also the results obtained by Knitro directly on the NLP problem (4.1) from the same neutral initial guess, which is chosen randomly at the very beginning and then fixed to be used by all the metaheuristics. The direct use of Knitro on the NLP problem (4.1) includes its state-of-the-art preliminary exploratory work on the space of variables. In all cases, for Knitro directly on the NLP problem and also for our proposals, we run the same option 5 times and report the obtained average results.

In addition, we also compare our results with a custom multistart strategy, which is based on starting the solver Knitro several times with a different randomly generated initial point each time. Once a specific number of Knitro executions have been carried out (five runs as in our proposed method), the best obtained solution is selected and reported.

Concerning the package Knitro, we use the Interior/CG option and we set max_number_iter = 10 when it is used by the metaheuristic algorithms to obtain MHSP, and max_number_iter = 10 000 when it is used to obtain *Solution1*, *Solution2*, *Solution3*, and the solution obtained when Knitro is applied directly to the original problem. All the other required parameters in Knitro are chosen by default.

Concerning the metaheuristic techniques, we use Python implementations obtained from different repositories. For CMA-ES we use the one available at [11], setting the initial population size $\lambda = 20$, and for the stopping conditions we set $tolx = 10^{-11}$, $tolfun = 10^{-11}$, and max_number_iter = 1000. For PSO we use the implementation available at [19], with 20 random particles as the initial population, setting the inertia

weight $\omega = 0.7$, the cognitive constant $\varphi_l = 2$, and the social constant $\varphi_g = 2$, and for the stopping conditions we set max_number_iter = 1000. For SA we use the one available at [35], setting $T_{\max} = 1$ and the cooling factor $\Delta T = 0.95$, and for the stopping conditions we set $T_{\min} = 0.1$, and we use the default values max_number_rejections = 2500, and max_number_runs = 500.

4.1. Disk packing problem

The disk packing problem, which is simple to describe but hard to solve, consists in placing q circles of equal radius $r > 0$ into a rectangular box $[0, d_1] \times [0, d_2] \subset \mathbb{R}^2$ in such a way that the common radius r is maximized and the intersection between any pair of distinct circles is at most one point, *i.e.*, the circles do not overlap. In spite of the simplicity of its description, this problem has a large number of local maximizers that are not global solutions, for which the objective function value is quite similar to the global one.

Consider $p^i = (p_1^i, p_2^i)$, $i = 1, \ldots, q$, the centers of the desired circles, which together with the radius r form the set of variables of the optimization problem. We will focus our attention in the unit square case, *i.e.*, $d_1 = d_2 = 1$. In that case, the disk packing problem can be formulated as follows

$$\begin{cases} \text{Minimize} & -r & & \\ \text{subject to} & ||p^i - p^j||_2^2 \geq (2r)^2 & i, j = 1, \ldots, q \quad (i < j) \\ & r \leq p_1^i \leq 1 - r, & i = 1, \ldots, q, \\ & r \leq p_2^i \leq 1 - r, & i = 1, \ldots, q. \end{cases} \tag{4.1}$$

The global solutions of the NLP problem (4.1), for several dimensions, are reported in the following URL [38]:

http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html,

and some of the already known results are surprising, for example when $q = \bar{k}^2$ and $\bar{k} \in \{2, \ldots, 6\}$, the regular grid $\bar{k} \times \bar{k}$ has been geometrically proved to be the global minimum of the problem. Nevertheless, for $\bar{k} = 7$ it was computationally shown that the maximum radius is strictly larger than the radius $1/14$ associated to the regular grid. For our experimental results we tested different dimensions from $q = 9$ to $q = 49$. Notice that the number of constraints in (4.1) is equal to $(q^2 + 7q)/2$.

In order to use the penalty function P given by (2.3) and the merit function given by (2.4), we need to divide the constraints in problem (4.1) into m groups, where $m \ll \min\{n, p + s\}$, and then apply our optimization technique to produce the vector $\rho \in \mathbb{R}^m$ of penalty parameters, where $\rho_i > 0$ for $1 \leq i \leq m$. For our experiments, we set $m = 5$ for all values of q, all values of $0 < \alpha < 1$, and all possible metaheuristic schemes. The constraints in (4.1), which are all inequality constraints, are grouped as follows: $\bar{\rho}_1$ is associated with the non-overlapping constraints $-||p^i - p^j||_2^2 + (2r)^2 \leq 0$ for $i, j = 1, \ldots, q \quad (i < j)$, and for $i = 1, \ldots, q$, $\bar{\rho}_2$ with $r - p_1^i \leq 0$, $\bar{\rho}_3$ with $r - p_2^i \leq 0$, $\bar{\rho}_4$ with $p_1^i + r - 1 \leq 0$, and $\bar{\rho}_5$ with $p_2^i + r - 1 \leq 0$. Moreover, we set $\gamma = 1$ and hence in our case, the penalty function in (2.3) is given by:

$$\begin{aligned} P(x; \rho) = \bar{\rho}_1 \sum_{i<j}^{q} \max(0, -||p^i - p^j||_2^2 + 4r^2) + \bar{\rho}_2 \sum_{i=1}^{q} \max(0, r - p_1^i) \\ + \bar{\rho}_3 \sum_{i=1}^{q} \max(0, r - p_2^i) + \bar{\rho}_4 \sum_{i=1}^{q} \max(0, p_1^i + r - 1) + \bar{\rho}_5 \sum_{i=1}^{q} \max(0, p_2^i + r - 1). \end{aligned} \tag{4.2}$$

The results obtained by the two strategies described in Section 2, for different number of disks, are shown in the forthcoming tables. At the beginning of each table, the results of the best solution reported in the literature [38] are shown, as well as the ones obtained by applying Knitro directly to the original problem and the multistart method combined with Knitro. This means, that we compare our proposed method to generate initial points, with the one used automatically by Knitro as a default option, and with the best solution obtained by the multistart strategy.

TABLE 1. Performance of the proposed schemes for solving the disk packing problem with 9 disks.

Applying Knitro directly to the original problem			Multistart method combined with Knitro			Best found solution reported in [38]		
r	t(s)	unfeaC	r	t(s)	unfeaC	r	t(s)	unfeaC
.166657	0.05	0	.166666	0.21	0	.166666	–	0

			MHSP metaheuristic results			*Solution2* Knitro results using the penalized model		
exp	Metaheur	α	r	t(s)	unfeaC	r	t(s)	unfeaC
1	CMA-ES	.75	.080912	217	0.0226	.045888	0.05	0.0370
2	CMA-ES	.50	.064900	258	0.0080	.063714	0.06	0.0282
3	CMA-ES	.25	.056938	244	0.0005	.065293	0.09	0.0110
4	PSO	.75	.059831	157	0.0000	.039234	0.07	0.0319
5	PSO	.50	.078863	233	0	.050426	0.05	0.0132
6	PSO	.25	.059837	157	0.0000	.052224	0.10	0.0524
7	SA	.75	.078362	47	1.5026	.044198	0.08	0.0404
8	SA	.50	.071450	57	0.5982	.054957	0.07	0.0226
9	SA	.25	.060797	67	0.4635	.045296	0.06	0.0121

			Solution1 Knitro results using the original problem, with x_0 from MHSP			*Solution3* Knitro results using the original problem, with x_0 from *Solution2*		
exp	Metaheur	α	r	t(s)	unfeaC	r	t(s)	unfeaC
1	CMA-ES	.75	.166664	0.16	1.7E-9	.166664	0.02	3.4E-9
2	CMA-ES	.50	.166666	0.42	3.4E-9	.166662	0.02	0
3	CMA-ES	.25	**.166666**	0.10	1.7E-9	.166664	0.02	3.1E-9
4	PSO	.75	.166664	0.14	3.3E-9	.166664	0.02	3.4E-9
5	PSO	.50	.166666	0.19	3.3E-9	.166666	0.02	8.6E-9
6	PSO	.25	.166665	0.10	4.3E-9	.166666	0.02	3.4E-9
7	SA	.75	.166666	0.34	0	.166666	0.02	3.4E-9
8	SA	.50	.166666	0.23	5.1E-9	.166662	0.02	3.4E-9
9	SA	.25	.166666	0.36	6.9E-9	**.166666**	0.02	1.7E-9

The column exp represents the identification number of the experiment (understanding as experiment the same configuration for different runs), metaheur indicates the considered metaheuristic, α is the parameter used in (2.4), r is the average of the best radius values obtained by the different runs of the experiments, t(s) is the average runtime in seconds, and unfeaC is the summation in absolute value of the obtained infeasibility in the constrains. For each table there are four sub-tables that are related to the two different strategies as explained in Figure 2. These different sub-tables are related to each other by the experiment identification number.

Due to the stochastic nature of the considered metaheuristic algorithms, for each considered number of disks it is convenient to show the distribution of the final high quality solutions (*Solution1* and *Solution3*) *via* boxplots. At each boxplot, each box represents one experiment (combination of α with a specific metaheuristic). Moreover, the blue line marks the best solution found in the literature [38], and the green line marks the solution returned by Knitro when it is applied directly to the original problem.

Q3 In the case of solving the disk packing problem with 9 disks, the obtained results are reported in Table 1 and Figure 3. It can be observed that for all possible combinations of metaheuristic scheme and value of α, the obtained results in *Solution1* and *Solution3* improve the green line, and also that they are very close to the blue line of the best found solution reported in [38], which imply a clear success of our combined strategies.

Table 2 and Figure 4 show the results when solving the disk packing problem in the case of 25 disks. It can be observed, as before, that the obtained solutions (*Solution1* and *Solution3*), are close to the best known solution (blue line), and each of them improves the solution provided by Knitro when it is applied directly to the original

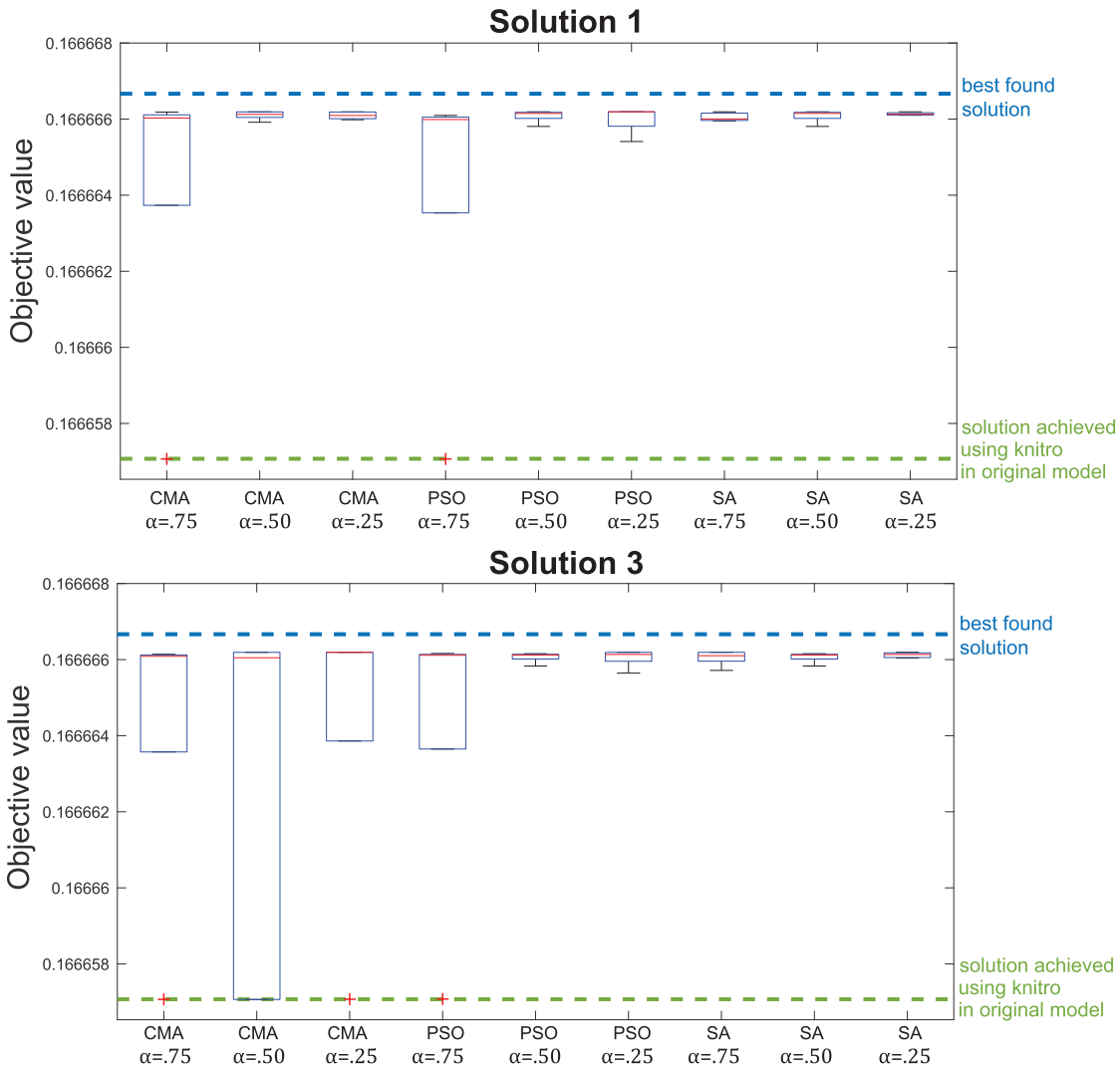

FIGURE 3. Distribution of final results for *Solution1* (top) and *Solution3* (bottom), when solving the disk packing problem with 9 disks.

problem (green line). In this case, the obtained solution by Knitro (green line) represents a local non-global solution for which the objective function value is approximately 4% less than the value at the global solution. It is worth recalling that for the disk packing problem a very small change in the radius could produce an ineffective distribution of the disks. It can also be observed from Table 2 that the value of infeasibility in some cases is slightly higher than the ones observed for 9 disks, but they still represent an insignificant numerical value when compared with the obtained optimality value. Thus, in spite of these small infeasibility values, our proposals out perform the results obtained by the multistart method. In fact, we note in Table 2 that for every possible value of α and for each possible metaheuristic, the result obtained by either *Solution1* or *Solution3* is better than the best solution obtained by the multistart method, and much better than the one obtained by Knitro on the original problem. Finally, it can be noted that the PSO metaheuristic stands out for obtaining *Solution1*, and the SA metaheuristic is the best option for obtaining *Solution3*.

TABLE 2. Performance of the proposed schemes for solving the disk packing problem with 25 disks.

Applying Knitro directly to the original problem			Multistart method combined with Knitro			Best found solution reported in [38]		
r	t(s)	unfeaC	r	t(s)	unfeaC	r	t(s)	unfeaC
.096323	0.29	0	.097877	1.07	0	.100000	–	0

			MHSP metaheuristic results			*Solution2* Knitro results using the penalized model		
exp	Metaheur	α	r	t(s)	unfeaC	r	t(s)	unfeaC
1	CMA-ES	.75	.030224	378	0.0038	.008638	1.17	0.0209
2	CMA-ES	.50	.025523	375	0.0039	.009631	0.61	0.0261
3	CMA-ES	.25	.026200	322	4E-09	.010731	0.79	0.0071
4	PSO	.75	.038030	319	0.0124	.016591	0.65	1.1E-2
5	PSO	.50	.030735	279	0.0023	.016377	0.81	0.0165
6	PSO	.25	.030665	339	9E-06	.016896	1.10	0.0007
7	SA	.75	.032978	80	0.4476	.019510	0.87	2.0E-3
8	SA	.50	.024775	103	0.4110	.015648	1.26	0.0157
9	SA	.25	.025844	116	0.8009	.018503	1.31	5.9E-6

			Solution1 Knitro results using the original problem, with x_0 from MHSP			*Solution3* Knitro results using the original problem, with x_0 from *Solution2*		
exp	Metaheur	α	r	t(s)	unfeaC	r	t(s)	unfeaC
1	CMA-ES	.75	.098614	0.26	8.6E-7	.098712	0.15	9.8E-7
2	CMA-ES	.50	.098710	1.63	1.0E-6	.097894	0.14	2.9E-7
3	CMA-ES	.25	.099171	1.11	2.0E-6	.098976	0.14	1.7E-6
4	PSO	.75	**.099986**	1.16	2.0E-6	.098701	0.12	2.5E-6
5	PSO	.50	.098600	0.84	6.0E-7	.097924	0.13	9.3E-9
6	PSO	.25	.097993	0.29	1.1E-6	.097898	0.16	7.9E-9
7	SA	.75	.098310	0.66	2.1E-7	.098501	0.14	8.6E-7
8	SA	.50	.098235	1.00	6.6E-7	.098865	0.12	1.5E-6
9	SA	.25	.098256	0.47	6.8E-7	**.099089**	0.14	1.0E-6

Table 3 and Figure 5 show the obtained results for 49 disks. In this case, the number of constraints significantly increase as compared to the previous two cases, which increases the complexity of the optimization problems to be solved, and as a consequence we observe an increase in the required CPU time. Nevertheless, it can also be observed that in many cases the results obtained in *Solution1* and *Solution3* improve the solution obtained by Knitro when applied directly to the original NLP problem. The infeasibility values are also not significant in this case. It can be seen from the boxplots in Figure 5 that for *Solution1* the average values usually fall above the green line, and some of them are very close to the blue line, which is a very good indication. We also note that in the average the results are better in *Solution1* than in *Solution3*. However, the best particular solutions (upper end of the boxes) are observed in *Solution3*.

4.2. Constrained optimization benchmarks

For our next experiments we consider NLP models from a set of global optimization problems known as the *Cute subcollection of Nonlinear Optimization Models [33]*. We have selected three of them, taking into account the necessary features of the objective function and the constraints to guarantee the presence of local-nonglobal solutions:

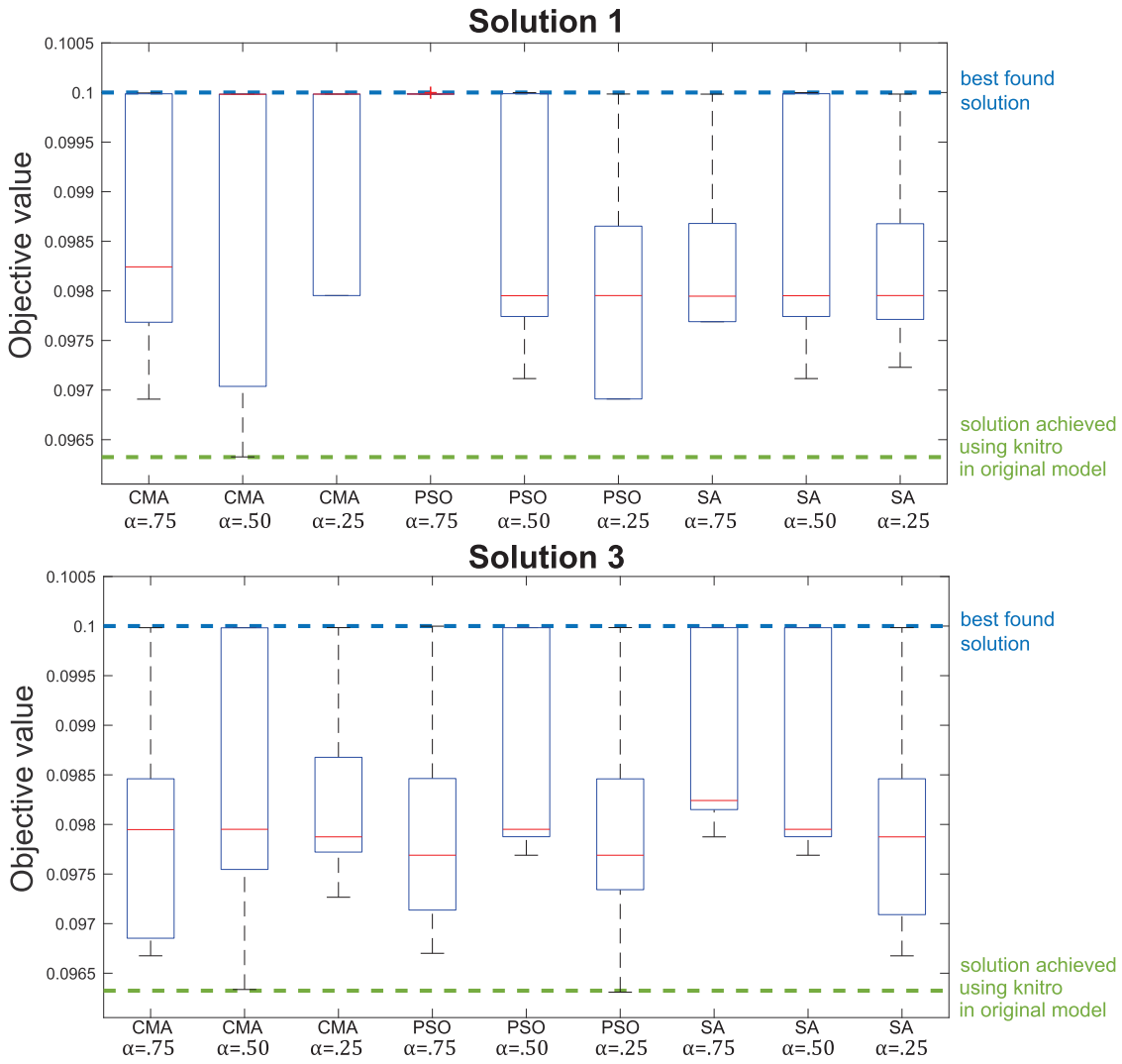

FIGURE 4. Distribution of final results for *Solution1* (top) and *Solution3* (bottom), when solving the disk packing problem with 25 disks.

(1) *eg3 model.* This NLP optimization problem contains 101 variables and 400 constraints (200 linear and 200 nonlinear). It is described in [5], and for $m = 5$ it can be formulated as follows

$$\begin{cases} \text{Minimize} & \frac{1}{2}((x_1 - x_{100})x_2 + x_{101})^2, & \\ \text{subject to} & C_1: \quad x_{101} + x_1 x_{i+1} + \frac{1+2}{i} x_i x_{100} \leq 0, & i = 1, \ldots, 99; \\ & C_2: \quad -0.5 + \sin(x_i)^2 \leq 0, & i = 1, \ldots, 100; \\ & C_3: \quad (x_1 + x_{100})^2 - 1 = 0, & \\ & C_4: \quad -1 - x_i \leq 0, & i = 1, \ldots, 100; \\ & C_5: \quad x_i - i \leq 0, & i = 1, \ldots, 100. \end{cases}$$

TABLE 3. Performance of the proposed schemes for solving the disk packing problem with 49 disks.

Applying Knitro directly to the original problem			Multistart method combined with Knitro			Best found solution reported in [38]		
r	t(s)	unfeaC	r	t(s)	unfeaC	r	t(s)	unfeaC
.071133	1.94	0	.071240	7.80	9.5E-11	.071692	–	0

			MHSP metaheuristic results			*Solution2* Knitro results using the penalized model		
exp	Metaheur	α	r	t(s)	unfeaC	r	t(s)	unfeaC
1	CMA-ES	.75	.018424	1612	0.0200	.004259	4.0	9.4E-4
2	CMA-ES	.50	.011636	1126	0.0155	.005267	5.7	0.0360
3	CMA-ES	.25	.010005	1331	0.0114	.003685	38	1.2E-2
4	PSO	.75	.018648	1077	0.0105	.005024	6.7	2.2E-2
5	PSO	.50	.011520	579	5.1E-4	.005266	5.4	1.0E-1
6	PSO	.25	.009121	772	2.2E-8	.009110	5.6	0.0416
7	SA	.75	.014964	524	1.4074	.004274	3.4	0.0421
8	SA	.50	.008912	838	4.7E-1	.004932	3.2	2.3E-3
9	SA	.25	.006936	1061	2.3740	.006847	3.3	5.4E-2

			Solution1 Knitro results using the original problem, with x_0 from MHSP			*Solution3* Knitro results using the original problem, with x_0 from *Solution2*		
exp	Metaheur	α	r	t(s)	unfeaC	r	t(s)	unfeaC
1	CMA-ES	.75	.071016	3.5	0	.070994	1.3	0
2	CMA-ES	.50	.071014	5.1	0	.071253	1.3	0
3	CMA-ES	.25	**.071260**	21.8	0	**.071320**	1.5	7.6E-8
4	PSO	.75	.071220	9.2	0	.071052	1.1	0
5	PSO	.50	.071211	25.2	0	.071000	1.1	5.6E-9
6	PSO	.25	.071023	14.3	0	.071179	1.2	3.0E-9
7	SA	.75	.071216	3.8	0	.070906	1.4	0
8	SA	.50	.071148	16	0	.071132	1.5	0
9	SA	.25	.071203	8.4	1.3E-9	.071071	1.3	2.0E-9

(2) *expfitc model*. This NLP problem consists of 5 variables and 502 linear inequality constraints. It was presented in [30], and for $m = 4$ it can be formulated as:

$$
\begin{cases}
\text{Minimize} & \sum_{i=1}^{251} \left(\dfrac{x_1 + x_2 t_i + x_3 t_i^2}{e^{t_i}(1 + x_4(t_i - 5) + x_5(t_i - 5)^2))} - 1 \right)^2, & \text{where } t_i = \dfrac{i-1}{50}, \\
\text{subject to} & C_1: \quad -(x_1 + x_2 t_i + x_3 t_i^2 - (t_i - 5)e^{t_i} x_4 & \\
 & \qquad -(t_i - 5)^2 e^{t_i} x_5 - e^{t_i}) \le 0 & i = 1, \ldots, 125; \\
 & C_2: \quad -(x_1 + x_2 t_i + x_3 t_i^2 - (t_i - 5)e^{t_i} x_4 & \\
 & \qquad -(t_i - 5)^2 e^{t_i} x_5 - e^{t_i}) \le 0 & i = 126, \ldots, 251; \\
 & C_3: \quad -((t_i - 5)x_4 + (t_i - 5)^2 x_5 + 0.99999) \le 0 & i = 1, \ldots, 125; \\
 & C_4: \quad -((t_i - 5)x_4 + (t_i - 5)^2 x_5 + 0.99999) \le 0 & i = 126, \ldots, 251.
\end{cases}
$$

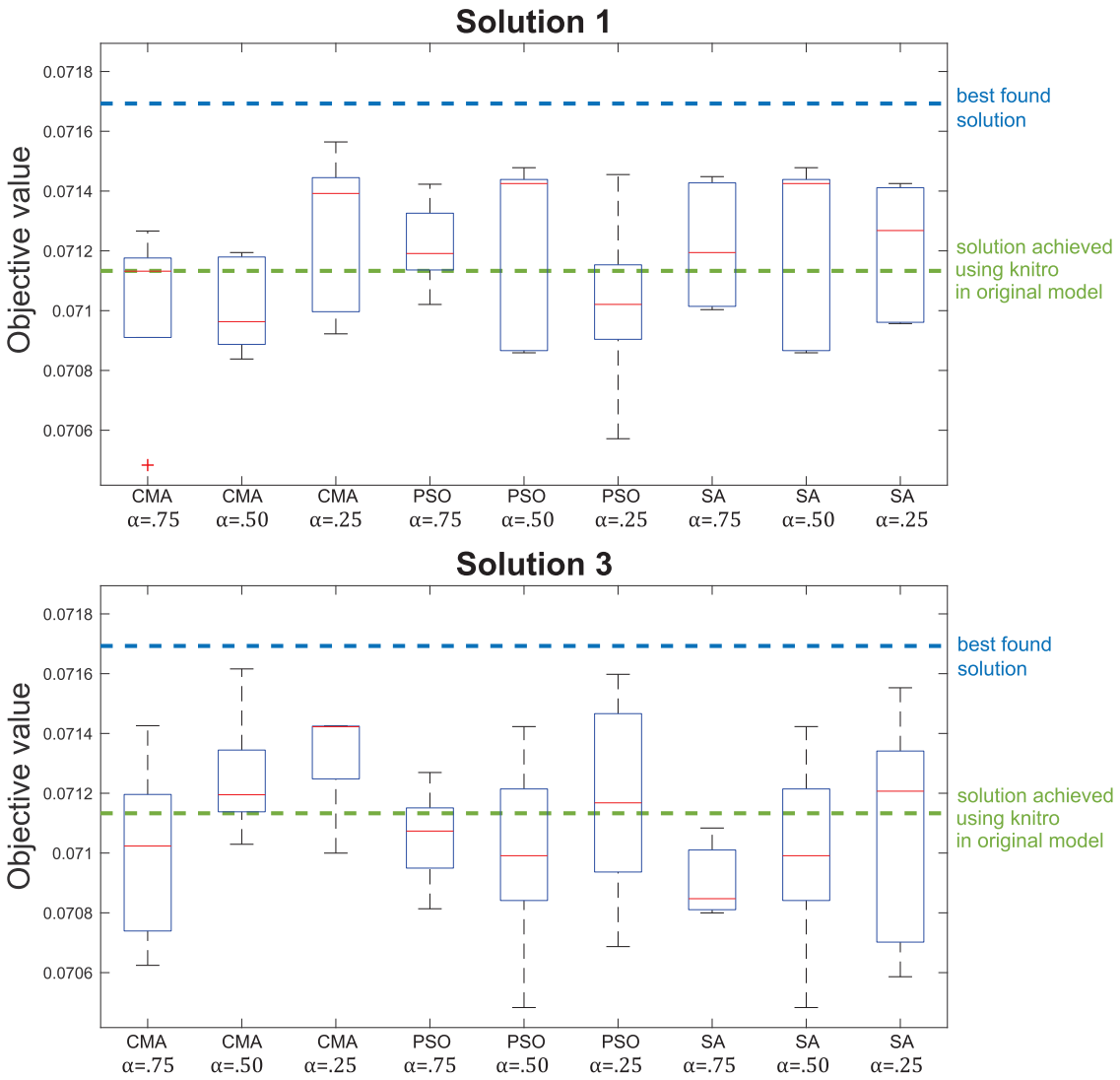

FIGURE 5. Distribution of final results for *Solution1* (top) and *Solution3* (bottom), when solving the disk packing problem with 49 disks.

(3) *haldmads model*. This NLP problem has 6 variables and 42 nonlinear inequality constraints (see [10] for additional details). For $m = 4$ it can be formulated as follows:

$$
\begin{cases}
\text{Minimize} & x_6, \quad \text{where } y_i = -1 + 0.1(i-1) & & i = 1, \ldots, 21, \\
\text{subject to} & C_1: \quad \dfrac{x_1 + y_i x_2}{1 + x_3 y_i + x_4 y_i^2 + x_5 y_i^3} - x_6 - e^{y_i} \leq 0 & & i = 1, \ldots, 10; \\
& C_2: \quad \dfrac{x_1 + y_i x_2}{1 + x_3 y_i + x_4 y_i^2 + x_5 y_i^3} - x_6 - e^{y_i} \leq 0 & & i = 11, \ldots, 21; \\
& C_3: \quad -\dfrac{x_1 + y_i x_2}{1 + x_3 y_i + x_4 y_i^2 + x_5 y_i^3} - x_6 + e^{y_i} \leq 0 & & i = 1, \ldots, 10; \\
& C_4: \quad -\dfrac{x_1 + y_i x_2}{1 + x_3 y_i + x_4 y_i^2 + x_5 y_i^3} - x_6 + e^{y_i} \leq 0 & & i = 11, \ldots, 21;
\end{cases}
$$

TABLE 4. Performance of the proposed schemes for the Cute NLP test models.

	Applying Knitro directly to the original problem			Multistart method combined with Knitro			Best found solution reported in [7]	
Problem	$f(x)$	t(s)	unfeaC	$f(x)$	t(s)	unfeaC	$f(x)$	unfeaC
eg3	0.31407	0.05	0	0.06718	0.39	5.3E-2	0	0
expfitc	57.415	0.07	0	0.02330	0.79	0	0.0233	0
haldmads	2.5885	0.03	0	0.03206	0.17	0	1.2E-4	0

			MHSP metaheuristic results			*Solution2* Knitro results using the penalized model		
Problem	Metaheur	α	$f(x)$	t(s)	unfeaC	$f(x)$	t(s)	unfeaC
eg3	CMA-ES	.5	0.24515	146	5.5E-5	0.24515	0.02	0
eg3	PSO	.5	0.11927	699	2.4E-1	0.11920	0.02	9.1E-2
eg3	SA	.5	0.21669	261	2.3E-	0.21667	0.02	2.2E-2
expfitc	CMA-ES	.5	50.384	65	8.2E-6	49.264	0.05	4.5E-2
expfitc	PSO	.5	33.729	730	8.5E0	32.358	0.33	9.09E0
expfitc	SA	.5	434.18	166	8.7E-2	337.99	0.6	1.3E-2
haldmads	CMA-ES	.5	0.06002	132	2.8E-3	0.05163	0.01	0
haldmads	PSO	.5	0.04508	230	1.9E-1	0.09414	0.01	0
haldmads	SA	.5	0.11303	257	2.48E0	0.02019	0.02	0

			Solution1 Knitro results using the original problem, with x_0 from MHSP			*Solution3* Knitro results using the original problem, with x_0 from *Solution2*		
Problem	Metaheur	α	$f(x)$	t(s)	unfeaC	$f(x)$	t(s)	unfeaC
eg3	CMA-ES	.5	0.06718	0.02	4.7E-2	0.06718	0.03	4.7E-2
eg3	PSO	.5	0.06718	0.02	5.1E-2	0.06718	0.03	5.1E-1
eg3	SA	.5	0.06718	0.02	4.8E-2	0.06718	0.03	4.8E-2
expfitc	CMA-ES	.5	0.02330	0.05	0	0.02331	0.17	0
expfitc	PSO	.5	0.02330	0.33	0	0.02331	0.25	0
expfitc	SA	.5	0.02513	0.6	0	0.02450	0.12	0
haldmads	CMA-ES	.5	0.35340	0.01	0	0.00675	0.01	6.7E-9
haldmads	PSO	.5	0.33917	0.01	5.E-10	0.01364	0.01	7.0E-9
haldmads	SA	.5	0.33670	0.02	0	0.02023	0.01	0

As in Section 4.1, to build a penalized model of these NLP problems we need to add the penalty term $P(x;\rho)$ to the objective function. For that, $P(x;\rho)$ is obtained multiplying each group of constraints C_i by an associated penalty parameter as indicated in (2.3).

Table 4 shows the results of all the considered methods, using an AMPL implementation of eg3, expfitc and haldmads, provided by [7]. For our combined strategies, only the results for $\alpha = 0.5$ have been reported since this choice is the one that produced the best results. In all the results, the solutions generated by our calibration method remain close to the best known solution, and out perform the results reported by Knitro when it is applied directly to the original problem. On the other hand, the comparison with the multistart strategy is also positive: the average solution values of our proposals are competitive or slightly better than the solutions reported by the multistart method, when solving problems eg3 and expfitc. Moreover, in the particular case of problem haldmads, the results obtained by *Solution3* are clearly much better than the ones obtained by the multistart method and also by Knitro on the original problem, specially when using the CMA-ES metaheuristic in which an improvement up to an order of magnitude is observed.

We close this section with some special comments concerning the execution time required by the metaheuristic calibration, which in our tables is significantly higher than the time required by the Knitro package. This difference has a simple computational explanation: Knitro is a very efficient software written in a high level language (C/C++). In contrast, the metaheuristics used in this work are implemented in Python, downloaded from free code repositories, without an implementation based on efficiency. Moreover, the AMPL API used by our strategies for the evaluation of the merit function, and also to manipulate the mathematical optimization problems, clearly produces an additional communication overhead (between AMPL, Knitro, and Python) that affects the performance in terms of required time. On the other hand, Knitro incorporates in its structure the optimization problems, and moreover it does not need to call any API to evaluate the merit function. Since our most important objective in this work is to evaluate the performance of new strategies for obtaining the starting point in terms of quality of the solution, we do not consider at this point the required calculation time to be an important issue. For a fair comparison concerning required time, our combined algorithms should also be implemented in a compiled language, and the problem should be charged in a native structure to avoid overload in the process of calling APIs. In that case, the execution time required by our combined schemes would be drastically reduced by a few orders of magnitude. Nevertheless, for the sake of completeness, we have reported the required execution time for all the experiments.

5. Concluding remarks

We have developed a new approach for choosing the starting point in nonlinear programming that combines modern metaheuristic stochastic schemes with numerical unconstrained deterministic optimization techniques. For that we have presented a convenient penalty function that involves very few penalty parameters, and also a suitable merit function, that combines optimality and feasibility, to measure the quality of the penalty parameters. The metaheuristic schemes are in charge of producing different combinations of penalty parameters, whereas the numerical optimization techniques evaluates the merit function for each different combination. Using this combined approach, we have presented a starting point strategy, and also two different strategies (denoted as *Solution1* and *Solution3*) to solve the NLP problem. We have illustrated the properties of our proposal on three nonlinear programming benchmark-test problems and also on the disk-packing problem, that has many local-nonglobal solutions. We have used as the deterministic optimization technique the software package Knitro, and we have considered and adapted three different metaheuristic schemes: CMA-ES, PSO, and SA.

Our experiments show that in general the average results obtained by *Solution1* and *Solution3* are clearly better than the average ones obtained by applying directly the software Knitro on the NLP problem, and are competitive or slightly better than best ones (out of 5) obtained by the multistart strategy. Our results indicate that, in general, it is convenient to spend some extra time and effort using our two combined strategies to obtain a starting point with a good option of converging to either a feasible global solution or to a feasible local solution for which the objective function value is very close to the value at the global ones. We note that even a relatively small improvement in the objective function value, while keeping feasibility, could produce a more convenient solution vector $x \in \mathbb{R}^n$, in the sense that it could represent a significant practical gain when solving real industrial applications. In particular, our conclusion is that in the overall, the best combined strategies are the ones obtained when using the CMA-ES or the PSO metaheuristics.

It is worth mentioning, that in general, a software package like Knitro with local behavior, when applied directly on the NLP problem tends to converge fast to a local non-global solution from which it cannot escape. At this point, we notice that instead of using Knitro we could have used, for our experiments, any other deterministic numerical package as long as it includes trustable unconstrained as well as constrained optimization routines. Similarly, concerning the metaheuristic techniques, in addition to the ones considered in this work we can combine our approach with some other available options; see, *e.g.*, [20, 22, 36]. Moreover, we could also increase their strength and capacity, for example increasing the maximum number of iterations or runs, specially for solving larger and harder NLP problems.

Finally, concerning the so-called neutral initial guess, we have used for our experiments a fixed one that has been generated at random at the beginning. Nevertheless, a much better neutral initial guess could be supplied by the user, in such a way that the combined strategies take advantage of the user's knowledge of the specific problem to be solved. It is worth recalling that the user's guess not necessarily leads the convergence process to a good point, and hence our combined approach is still needed.

Acknowledgements. We would like to thank two anonymous referees for their constructive comments and suggestions that helped us to improve the final version of this paper. The second author was financially supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the project UID/MAT/00297/2019 (CMA).

References

[1] Artelys, Knitro nonlinear optimization solver. https://www.artelys.com/en/optimization-tools/knitro (2019).
[2] J.R. Banga, Optimization in computational systems biology. *BMC Syst. Biol.* **2** (2008) 1–47.
[3] R.H. Byrd, J. Nocedal and R.A. Waltz, Knitro: an integrated package for nonlinear optimization. In: Large-scale Nonlinear Optimization. Springer (2006) 35–59.
[4] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Comput. Methods Appl. Mech. Eng.* **191** (2002) 1245–1287.
[5] A.R. Conn, G. Gould and P.L. Toint, In: Vol. 17 of *LANCELOT: A Fortran Package for Large-scale Nonlinear Optimization (Release A)*. Springer Science & Business Media (2013).
[6] A.V. Fiacco and G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques. John Wiley and Sons, New York (1968).
[7] GAMS World site, Cute models section. http://www.gamsworld.org/performance/princetonlib/htm/group5stat.htm (2019).
[8] W.L. Goffe, G.D. Ferrier and J. Rogers, Global optimization of statistical functions with simulated annealing. *J. Econ.* **60** (1994) 65–99.
[9] J.D. Griffin and T. Kolda, Nonlinearly constrained optimization using heuristic penalty methods and asynchronous parallel generating set search. *Appl. Math. Res. Express* **2010** (2010) 36–62.
[10] J. Hald and K. Madsen, Combined lp and quasi-newton methods for minimax optimization. *Math. Program.* **20** (1981) 49–62.
[11] N. Hansen, A Python implementation of CMA-ES. https://github.com/CMA-ES/pycma (2017).
[12] N. Hansen and S. Kern, Evaluating the CMA evolution strategy on multimodal test functions. In: International Conference on Parallel Problem Solving from Nature. Springer (2004) 282–291.
[13] N. Hansen and A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation. IEEE (1996) 312–317.
[14] N. Hansen, S.D. Müller and P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **11** (2003) 1–18.
[15] A. Homaifar, C.X. Qi and S.H. Lai, Constrained optimization via genetic algorithms. *Simulation* **62** (1994) 242–253.
[16] M. Kaucic, A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization. *J. Global Optim.* **55** (2013) 165–188.
[17] J. Kennedy and R. Eberhart, Pso optimization. In: Vol. 4 of *Proc. IEEE Int. Conf. Neural Networks*. IEEE Service Center, Piscataway, NJ (1995) 1941–1948.
[18] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing. *Science* **220** (1983) 671–680.
[19] A. Lee, Particle swarm optimization (PSO) with constraint support. https://github.com/tisimst/pyswarm (2015).
[20] K.S. Lee and Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **194** (2005) 3902–3933.
[21] D.G. Luenberger, Linear and Nonlinear Programming. Addison-Wesley, Menlo Park, CA (1984).
[22] S. Luke, Essentials of Metaheuristics, 2nd edition. Lulu. Available for free at http://cs.gmu.edu/~sean/book/metaheuristics/ (2013).
[23] M.E.-B. Menai and M. Batouche, Efficient initial solution to extremal optimization algorithm for weighted maxsat problem. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer (2003) 592–603.
[24] Z. Michalewicz, Genetic algorithms for numerical optimization. *Int. Trans. Oper. Res.* **1** (1994) 223–240.
[25] Z. Michalewicz and C.Z. Janikow, Genetic algorithms for numerical optimization. *Stat. Comput.* **1** (1991) 75–91.
[26] K.E. Parsopoulos and M.N. Vrahatis, Particle swarm optimization method for constrained optimization problems. *Int. Technol.–Theory App.: New Trends Intell. Technol.* **76** (2002) 214–220.
[27] D.R. Penas, P. González, J.A. Egea, J.R. Banga and R. Doallo, Parallel metaheuristics in computational biology: an asynchronous cooperative enhanced scatter search method. *Proc. Comput. Sci.* **51** (2015) 630–639.

[28] D.R. Penas, D. Henriques, P. González, R. Doallo, J. Saez-Rodriguez and J.R. Banga, A parallel metaheuristic for large mixed-integer nonlinear dynamic optimization problems, with applications in computational biology. *PLoS One* **12** (2017) 1–32.
[29] R. Poli, J. Kennedy and T. Blackwell, Particle swarm optimization. *Swarm Intell.* **1** (2007) 33–57.
[30] M. Powell, A tolerant algorithm for linearly constrained optimization calculations. *Math. Program.* **45** (1989) 547–566.
[31] R.L. Rardin and R. Uzsoy, Experimental evaluation of heuristic optimization algorithms: a tutorial. *J. Heuristics* **7** (2001) 261–304.
[32] M. Rieck, M. Richter, M. Bittner and F. Holzapfel, Generation of initial guesses for optimal control problems with mixed integer dependent constraints. In: ICAS 29th International Conference (2014).
[33] J. Robert, Vanderbei website. University of Princeton. https://vanderbei.princeton.edu/ampl/nlmodels/ (2019).
[34] B. Suman and P. Kumar, A survey of simulated annealing as a tool for single and multiobjective optimization. *J. Oper. Res. Soc.* **57** (2006) 1143–1160.
[35] T. Takahashi, Metaheuristic Algorithms Python. https://github.com/tadatoshi/ (2015).
[36] E. Talbi, Metaheuristics: From Design to Implementation, 1st edition. Wiley (2009). ISBN:9780470278581.
[37] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly and R. Martí, Scatter search and local NLP solvers: a multistart framework for global optimization, *INFORMS J. Comput.* **19** (2007) 328–340.
[38] University of Magdeburg, The best known packings of equal circles in a square. http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html (2013).
[39] Ö. Yeniay, Penalty function methods for constrained optimization with genetic algorithms. *Math. Comput. App.* **10** (2005) 45–56.
[40] F. Zhang, A.C. Reynolds and D.S. Oliver, An initial guess for the levenberg–marquardt algorithm for conditioning a stochastic channel to pressure data. *Math. Geol.* **35** (2003) 67–88.