Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/120102

Finding Multiple Roots of Systems of Nonlinear Equations by a Hybrid Harmony Search-Based Multistart Method

Gisela C. V. Ramadas[1,*], *Edite M. G. P. Fernandes*[2] *and Ana Maria A. C. Rocha*[2]

[1] Department of Mathematics, School of Engineering, Polytechnic of Porto, 4200-072 Porto, Portugal
[2] Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal

Received: 3 Jul. 2016, Revised: 27 Oct. 2017, Accepted: 7 Nov. 2017
Published online: 1 Jan. 2018

Abstract: A multistart (MS) clustering technique to compute multiple roots of a system of nonlinear equations through the global optimization of an appropriate merit function is presented. The search procedure that is invoked to converge to a root, starting from a randomly generated point inside the search space, is a new variant of the harmony search (HS) metaheuristic. The HS draws its inspiration from an artistic process, the improvisation process of musicians seeking a wonderful harmony. The new hybrid HS algorithm is based on an improvisation operator that mimics the best harmony and uses the idea of a differential variation, borrowed from the differential evolution algorithm. Computational experiments involving a benchmark set of small and large dimensional problems with multiple roots are presented. The results show that the proposed hybrid HS-based MS algorithm is effective in locating multiple roots and competitive when compared with other metaheuristics.

Keywords: nonlinear equations, multistart, harmony search, differential evolution

1 Introduction

Some problems in engineering, chemistry, physics, medicine and economics aim at determining the roots of a system of equations. In general, these problems are nonlinear and difficult to solve. The most popular technique to solve nonlinear equations is the Newton's method [1]. It is a computationally expensive method since the Jacobian matrix and the solution of a system of linear equations are required at each iteration. On the other hand, Quasi-Newton methods are less expensive since they avoid either the necessity of computing derivatives, or the need of solving a full linear system per iteration or both tasks [2,3,4]. However, they assume that the functions are smooth so that derivatives can be properly approximated by finite differences. Another disadvantage of the traditional Newton-type methods is that their convergence and practical performance are highly sensitive to the provided initial approximations. In most practical cases, it is not an easy task to guess a good initial approximation. Furthermore, they are only capable of finding one root at each run of the algorithm.

This study extends the work presented in [5,6]. Here, we aim at investigating the performance of a multistart (MS) method combined with the harmony search (HS) metaheuristic to compute multiple roots of a system of nonlinear equations of the form

$$f(x) = 0, \tag{1}$$

where $f(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, each $f_i : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, n$ is a continuous possibly nonlinear function in the search space and Ω is a closed convex set, herein defined as $[l,u] = \{x : -\infty < l_i \leq x_i \leq u_i < \infty, i = 1, \ldots, n\}$. We do not assume that the functions $f_i(x)$, $i = 1, \ldots, n$ are differentiable. Thus, we are interested in deriving a derivative-free technique that does not assume smoothness, convexity and differentiability. The problem of solving a nonlinear system of equations can be naturally formulated as a global optimization problem. Problem (1) is equivalent to

$$\min_{x \in \Omega \subset \mathbb{R}^n} \mathscr{M}(x) \equiv \sum_{i=1}^{n} f_i(x)^2, \tag{2}$$

* Corresponding author e-mail: gcv@isep.ipp.pt

in the sense that they have the same solutions. These required solutions are the global minima, and not just the local minima, of the function $\mathscr{M}(x)$, known as merit function, in the set Ω. Problem (2) is similar to the usual least squares problem for which many iterative methods have been proposed. They basically assume that the objective function is twice continuously differentiable. However, the objective $\mathscr{M}$ in (2) is only once differentiable if some, or just one, of the f_i, $(i = 1,\ldots,n)$ are not differentiable. Thus, methods for solving the least squares problem cannot be directly applied to solve (2).

Preventing premature convergence to a local while trying to locate global solutions of problem (2) is the goal of the present study. Here, we are concerned with the performance of a metaheuristic to solve global optimization problems without the use of derivative information. Metaheuristics are general heuristic methods which can be applied to a wide variety of optimization problems. In the last decade, algorithms based on metaheuristics have been proposed to solve systems of nonlinear equations mainly using the reformulation (2) [7, 8,9,10,11,12,13]. In general, they do not require any information concerning root location, since they are able to converge to the solutions starting from points that can be randomly generated in the search space. Furthermore, their performance do not depend on any type of derivative information.

Although finding a single root of a system of nonlinear equations is a trivial task, finding all roots is one of the most demanding problems. Approaches that combine metaheuristics with techniques that modify the objective function in problem (2) have been reported in the literature [8,10]. The technique in [8,12] relies on the assignment of a penalty term to each previously computed root so that a repulsion area around the root is created. In [10], an evolutionary optimization algorithm is used together with a type of polarization technique to create a repulsion area around previously computed roots. A multiobjective evolutionary approach is another technique available for locating multiple roots [14,15]. MS methods are very popular and simple stochastic techniques that emerge when multiple solutions to problems are required [16,17,18,19]. When an MS strategy is implemented, a search procedure is applied to a set of randomly generated points of the search space aiming to converge to the multiple solutions of the problem in a single run. However, the same solutions may be computed more than once. Convergence to previously computed solutions can be avoided by implementing a clustering technique which aims to define prohibited regions based on the closeness to the previously computed solutions [13,17]. Each time a point is randomly selected from these prohibited regions, it will be discarded since the implementation of the search procedure will eventually produce one of the previously computed solutions. Most MS approaches implement a local search algorithm rather than a global one, when trying to locate a solution. The basic idea is to search for a highly accurate solution in a specified and small region, starting from a randomly selected promising point. Good local search procedures, like the traditional BFGS Quasi-Newton method, have been used in the past. The well-known 'MinFinder' and the 'Ideal Multistart' are two examples of this kind of multistart paradigms [13,17, 19]. Unfortunately they use first derivatives of the involved functions to define the most promising direction to search for the required solution. However, in the present study we aim to find multiple solutions as efficient as possible without relying on analytic or numerical derivatives. Thus, a search procedure has to be implemented that is capable of finding high quality solutions targeting reduced computational requirements. The basic HS algorithm that emerged in 2001 relies on a set of points and is inspired by natural phenomena [20]. It draws its inspiration not from a biological or physical process like most metaheuristic optimization techniques, but from an artistic one – the improvisation process of musicians seeking a wonderful harmony. HS has efficient strategies for exploring the entire search space, as well as techniques to exploit locally a promising region to yield a high quality solution in a reasonable time. The dynamic updating of two important parameters in the HS algorithm has improved the efficiency and robustness of the metaheuristic [21].

In this study, we borrow the ideas present in the global best variant of HS [22], propose self-adaptive updating rules for two relevant parameters in HS, and hybridize the HS algorithm with a mutation strategy quite common in the differential evolution (DE) method introduced in [23]. The herein proposed HS algorithm will be denoted by hybrid HS algorithm. Previous studies related with the HS and DE algorithms confirm that these metaheuristics are effective in computing one single root of system (1) [5,6]. However, the present work extends some of the previous ideas to the problem that consists of locating multiple roots of nonlinear systems like (1). This issue is herein addressed by using an MS strategy that relies on an attraction radius to prevent convergence to previously computed solutions, so reducing the overall required computational effort.

This paper is concerned with computing multiple roots of a system of nonlinear equations using an MS paradigm and the HS algorithm hybridized with a DE mutation as an effective search procedure. Thus, Section 2 reports on the hybrid HS algorithm and Section 3 addresses the MS strategy. Then, some numerical experiments are shown in Section 4 and we conclude the paper in Section 5.

2 Hybrid HS algorithm

First, the main ideas behind the basic HS algorithm [20] and two other popular versions of the algorithm available in the literature [21,22] are presented. Second, an introduction to the DE algorithm [23] is included. Finally, the hybrid version of the HS algorithm, combining the HS

paradigm with the DE mutation operator, aiming to increase the exploration feature of the HS algorithm, is described.

2.1 The HS algorithm

The HS algorithm was developed to solve global optimization problems in an analogy with the music improvisation process where music players improvise the pitches of their instruments to obtain better harmony [20, 24]. An overview of the existing variants of the HS is presented by Alia and Mandava in [25]. At each iteration k, the basic HS algorithm provides a set of solution vectors from which the best and the worst solutions, in terms of their fitness - objective function values - are selected. Table 1 lists the most relevant nomenclature used in the basic HS algorithm.

Table 1: Nomenclature for the HS algorithm

HM	harmony memory
HMS	size of the HM
NI	number of allowed improvisations/iterations
x^{best}	the best solution in HM
x^{worst}	the worst solution in HM
HMCR	harmony memory considering rate
PAR	the pitch adjusting rate
BW	distance bandwidth

The HM is a memory with HMS solution vectors that are maintained in memory throughout the iterative process. The HMCR and PAR parameters are used to find globally and locally improved solutions, respectively. After generating the HM randomly in the search space Ω, x^j, $j = 1,\ldots,$HMS, the vectors are evaluated and the best harmony, x^{best}, and the worst, x^{worst}, in terms of objective/merit function value are selected. Thereafter, a new harmony is improvised meaning that a new vector y is generated using three improvisation operators:

O_1.HM operator;
O_2.random selection operator;
O_3.pitch adjustment operator.

The HMCR parameter varies between 0 and 1 and gives the probability of choosing the component of the new harmony/vector from the HM (operator O_1). Otherwise, the component is randomly generated in Ω (operator O_2):

$$y_i = \begin{cases} x_i^j, j \text{ random } \in \{1,\ldots,\text{HMS}\}, & \text{if } rand() < \text{HMCR} \\ l_i + rand()(u_i - l_i), & \text{otherwise} \end{cases} \quad (3)$$

for $i = 1,\ldots,n$, where $rand()$ represents a random number in the range $(0,1)$. The operator O_3 is subsequently applied with probability PAR to refine only the components i produced by O_1, as follows:

$$y_i = \begin{cases} y_i \pm rand()\text{BW}, & \text{if } rand() < \text{PAR} \\ y_i, & \text{otherwise} \end{cases} \quad (4)$$

where BW is an arbitrary distance bandwidth. Finally, the HM is updated. The new harmony is compared with the worst harmony in the HM, in terms of $\mathscr{M}$ values. The new harmony is included in the HM, replacing the worst one if it is better than the worst harmony. Algorithm 1 presents the main steps of the basic HS algorithm. As

Data: HMS, NI, HMCR, PAR, BW
Set $k = 1$
Initialize HM: randomly generate x^j, $j = 1,\ldots,$HMS
Evaluate HM, select x^{best} and x^{worst}
while $k \leq NI$ **do**
  Improvise a new harmony y and evaluate
  Update HM and select x^{best} and x^{worst}
  Set $k = k+1$
  if x^{best} *is sufficiently accurate* **then**
    STOP
  end
end

Algorithm 1: HS algorithm

shown in (4), the classical HS algorithm uses fixed value for both PAR and BW. However, small values of PAR with large values of BW can considerably increase the number of iterations required to converge to an optimal solution of (2). Experience has shown that BW must take large values at the beginning of the iterative process to enforce the algorithm to increase the diversity of solution vectors. However, small BW values in the final iterations increase the fine-tuning of solution vectors. Furthermore, large values of PAR combined with small values of BW usually cause the improvement of best solutions in the final stage of the process. To eliminate the drawbacks due to fixed values of PAR and BW, an improved HS (I-HS) algorithm is proposed in [21]. The I-HS uses parameter values dynamically dependent on the iteration number k, as shown:

$$\text{PAR}(k) = \text{PAR}_{\min} + k\frac{(\text{PAR}_{\max} - \text{PAR}_{\min})}{NI} \quad (5)$$

where $\text{PAR}_{\min}$ and $\text{PAR}_{\max}$ are the minimum and maximum pitch adjusting rate respectively, and

$$\text{BW}(k) = \text{BW}_{\max} e^{ck}, \text{ for } c = \frac{\ln(\frac{\text{BW}_{\min}}{\text{BW}_{\max}})}{NI} \quad (6)$$

where $\text{BW}_{\min}$ and $\text{BW}_{\max}$ are the minimum and maximum bandwidth respectively.

In [22], a new variant of HS, called the global-best harmony search (gb-HS), is proposed. The

pitch-adjustment step of the HS is modified in a way that the new harmony can mimic the best harmony in the HM, adding a social dimension to the HS and replacing the BW related parameters altogether. Thus, the new pitch adjustment operator, O_3, is applied with probability $\mathrm{PAR}(k)$, computed from (5), to refine only the components produced by O_1, in the following way:

$$y_i = \begin{cases} x_t^{best}, t \text{ random } \in \{1,\ldots,n\}, & \text{if } rand() < \mathrm{PAR}(k) \\ y_i, & \text{otherwise} \end{cases} \tag{7}$$

where *best* is the index of the best harmony in the HM.

2.2 The DE algorithm

Algorithm 2 contains the pseudocode of the basic DE algorithm [23]. This is an evolutionary population-based technique that relies on three operators – *mutation*, *crossover* and *selection* – to define the m points for the next iteration. The most commonly used *mutation* is

Data: m, F, CR, $K_{\max}$
Set $k = 1$
Randomly generate target points $x^i \in \Omega$, $i = 1\ldots,m$
Evaluate the points and select x^{best}
while $k \leq K_{\max}$ **do**
  Perform *mutation* to generate the mutant points
  Perform *crossover* to generate the trial points
  Evaluate the trial points
  Perform *selection* to define target points, select x^{best}
  Set $k = k+1$
  if x^{best} *is sufficiently accurate* **then**
    STOP
  end
end

Algorithm 2: DE algorithm

referred to as DE/rand/1 and defines the mutant point, v^j, as follows:

$$v^j = x^{r_1} + F(x^{r_2} - x^{r_3}) \tag{8}$$

with uniformly chosen random indices r_1, r_2, r_3 from the set $\{1,2,\ldots,m\}$, mutually different and F is a real parameter in $[0,2]$ which controls the amplification of the differential variation, $x^{r_2} - x^{r_3}$. The indices r_1, r_2 and r_3 are also chosen to be different from the index j. x^{r_1} is called the base point. There are other frequently used mutation strategies, for instance, the DE/best/1, which uses the best point of the population as the base point:

$$v^j = x^{best} + F(x^{r_1} - x^{r_2}) \tag{9}$$

where x^{best} is the best point in the current population. The *crossover* operator aims to increase the diversity on the components of the mutant point. Thus, the crossover point, called trial point, y^j, is formed as:

$$y_i^j = \begin{cases} v_i^j, & \text{if } rand() \leq CR \text{ or } i = s_j \\ x_i^j, & \text{otherwise} \end{cases} \tag{10}$$

for $i = 1,\ldots,n$, where $rand()$ denotes a random number in $(0,1)$ and aims to perform the mixing of the component i of the points, $CR \in [0,1]$ is the parameter for crossover, and the index s_j, randomly selected from $\{1,\ldots,n\}$, ensures that y^j gets at least one component from v^j.

2.3 The hybrid HS algorithm

To develop a new hybrid HS algorithm, the ideas from the gb-HS algorithm to improvise a new harmony and the *mutation* operator present in DE to increase the explorative power of the classical HS are used. The new proposal replaces the improvisation operator O_1 by another one that mimics the best harmony and uses the *mutation* operator of the DE algorithm [23]. Basically, the idea is to generate a trial point by adding the weighted difference between two points to a third one [5]. Thus, the parameter HMCR sets the probability of choosing the component of the new harmony from the best harmony in HM adding a differential variation, i.e., for each $i = 1,\ldots,n$:

$$y_i = \begin{cases} x_i^{best} + F(x_i^{j_1} - x_i^{j_2}), & \text{if } rand() < \mathrm{HMCR} \\ & \text{and } j_1 \neq j_2 \\ x_t^j, t \text{ random } \in \{1,\ldots,n\} & \\ \quad j \text{ random } \in \{1,\ldots,\mathrm{HMS}\}, & \text{if } rand() < \mathrm{HMCR} \\ & \text{and } j_1 = j_2 \\ l_i + rand()(u_i - l_i), & \text{otherwise} \end{cases} \tag{11}$$

is used instead of (3), where $F \in [0,2]$ and the indices j_1, j_2 are randomly chosen values from the set $\{1,\ldots,\mathrm{HMS}\}$. However, when these indices are not different, a randomly chosen component of a point randomly selected from the HM is selected, to diversify the search.

We note that when defining y, some components can be generated outside the domain Ω. Thus, each component should be checked and a projection to the bounds is carried out:

$$y_i^j = \max\left\{l_i, \min\left\{y_i^j, u_i\right\}\right\}, \text{ for } i = 1,\ldots,n. \tag{12}$$

In the HS context, to further explore the search space for a promising region where a global solution lies, the pitch adjustment operator (O_3) is maintained although the parameters PAR and BW are updated according to new self-adaptive rules. The proposed updating rule for the probability PAR ensures that the larger values appear in the final iterations and are combined with small values of the bandwidth BW (as shown below in (13)):

$$\mathrm{PAR} = \frac{1}{1 + \mathscr{M}(x^{worst})}$$

where x^{worst} is the worst harmony in the HM in terms of the merit function (at the current iteration). For the BW, an adaptive value for each component $i = 1,\ldots,n$ is assigned as follows

$$\mathrm{BW}^i = \mathrm{BW}^i_{\max}\left(1 - \frac{1}{1+\mathscr{M}(x^{worst})}\right), \qquad (13)$$

where $\mathrm{BW}^i_{\max} = u_i - l_i$ depends on the lower and upper bounds of each component i.

3 Multistart algorithm

MS methods are mainly used to locate multiple solutions of bound constrained optimization problems [16,17,18,19,26], although they can be used to locate multiple solutions of other kind of problems [8,10,13]. MS repeatedly invokes a search procedure starting from a set of randomly generated points of the search space aiming to converge to the multiple solutions of the problem. However, the same solutions may be computed over and over again. To avoid convergence to previously computed solutions, a clustering technique that defines prohibited regions based on the closeness to the previously located solutions may be integrated into the multistart algorithm. This way, points that are generated from these prohibited regions are discarded since the search procedure would converge most certainly to a previously located solution.

Hence, to compute multiple roots of a set of nonlinear equations like the one in (1), computing global minimizers of the problem (2), the herein implemented MS methodology uses a clustering technique to avoid convergence to an already located solution. The exploration feature of the MS strategy is carried out by generating points randomly spread all over the search space Ω. The exploitation of promising regions is carried out by invoking a search procedure starting from each of the randomly generated points. In contrast to the line search BFGS method presented in [19], our proposal for the search procedure relies on a global search heuristic that is capable of computing global minimizers of a merit function with high quality and reduced computational effort.

The presented clustering technique also uses the concept of regions of attraction of previously identified solutions. First, the region of attraction of a minimizer, χ_i, associated with a search procedure, herein denoted by **HS**, is defined as:

$$A_i \equiv \{x \in [l,u] : \mathbf{HS}(x,\mathscr{R},[l,u]) = \chi_i\}, \qquad (14)$$

where χ_i is the global, eventually a local (non-global), minimizer produced by the hybrid HS algorithm which is applied in the search space $[l,u]$, starting with $x \in \mathscr{R}$ and randomly generating the remaining HMS-1 points of the HM inside the region $\mathscr{R}$. The ultimate goal of the MS algorithm is to invoke the procedure **HS** as many times, say N times, as the number of global solutions of (2). If a generated point $x \in [l,u]$ belongs to a region of attraction A_j then the solution χ_j would be obtained when the hybrid HS algorithm starts with a HM which includes the point x. The idea is to apply the hybrid HS algorithm only when x does not belong to any of the regions of attraction of already computed solutions, or equivalently to the union of those regions of attraction, since they do not overlap. The probability p that a randomly generated point will not belong to the union of the regions of attraction of r already computed solutions, can be estimated by

$$p = Prob[x \notin \cup_{i=1}^{r} A_i] = \prod_{i=1}^{r} Prob[x \notin A_i] \approx Prob[x \notin A_o]$$

where A_o is the region of attraction of the nearest to x solution χ_o (see details in [19]). The region of attraction A_o of a solution χ_o could be estimated by an exhaustive enumeration of all starting points that converge to χ_o. Alternatively, an *a priori* estimation involves the probability $Prob[x \notin B(\chi_o, R_o)]$ where $B(\chi, R)$ denotes the hyper-sphere (henceforth denoted by sphere) centered at χ with a radius R.

Let the maximum attractive radius of the minimizer χ_i be defined by:

$$R_i = \max_j \left\{ \left\| x_i^{(j)} - \chi_i \right\| \right\}, \qquad (15)$$

where $x_i^{(j)}$ is one of the generated points that made the hybrid HS algorithm to converge to χ_i. Given x, let $d_i = \|x - \chi_i\|$ be the distance of x to χ_i. We remark that:

1. if $d_i \geq R_i$ then x is likely to be outside the region of attraction of χ_i;
2. on the other hand, if $d_i < R_i$ and the direction from x to χ_i is descent then x is likely to be inside the region of attraction of χ_i; however, if the direction from x to χ_i is ascent then x is likely to be outside the region of attraction of χ_i.

In case 1., the hybrid HS search procedure could be invoked, with x as one of the points in the HM, since a new solution could be obtained with high probability. However, when the direction from x to χ_i is descent, there is a high probability that the search procedure will converge to the previous χ_i, thus the hybrid HS procedure might not be invoked. Thus, the probability that $x \notin A_i$ is estimated by:

$$Prob(x \notin A_i) \approx \begin{cases} 1, & \text{if } d_i \geq R_i \text{ or the direction from } x \text{ to } \chi_i \\ & \text{is ascent} \\ \phi, & \text{otherwise} \end{cases}$$

where $\phi \in [0,1)$. It is reasonable to expect that the probability ϕ is larger at the beginning of the iterative process and smaller towards the final iterations since at

the beginning there is a higher chance that the hybrid HS procedure will converge to a new solution. Thus, a dynamic updating scheme for ϕ is proposed, depending on the iteration number $k > 0$,

$$\phi_k = \gamma_p \left(1 - \frac{k}{\mathbf{k}_{\max}}\right) \tag{16}$$

where $\gamma_p \in (0,1)$ is a constant and $\mathbf{k}_{\max}$ is the maximum number of iterations defined for the process. The smaller the parameter γ_p, the lower the likelihood of invoking the search procedure **HS** when the direction from x to χ_i is not ascent.

Since derivative information is not used in the algorithm, the direction from x to χ_i is considered to be ascent if $\mathscr{M}(x+\beta(\chi_i - x)) - \mathscr{M}(x) > 0$ for a sufficiently small positive β. The pseudocode of the herein implemented MS method is presented in Algorithm 3 where Σ denotes the set where the computed minimizers are saved. To identify a solution that has not been previously located, $\chi \notin \Sigma$, one of the following conditions must hold: $\|\chi - \chi_j\| > tol$ or $\left|\mathscr{M}(\chi) - \mathscr{M}(\chi_j)\right| > tol$ for all $\chi_j \in \Sigma$, where tol is a small tolerance.

There are two important issues that should be noted whenever the HS search procedure is invoked in the Algorithm 3. First, the region $\mathscr{R}$ from which the remaining HMS-1 points of the HM are randomly selected is a sphere centered at x with a specified radius R, $B(x,R)$. This way, the likelihood that the hybrid HS will converge to the nearest to x minimizer (not yet located) is higher than that of any other far away previously located minimizer. Also, in the HS context, the (hyper-)box from which the new harmony y is generated (see (11) and (12)) depends on the closeness of x to χ_o. If the likelihood of x being inside the region of attraction of χ_o is high, the entire box is used; otherwise, a more restrictive box is defined $[\max\{l, x-R_o\}, \min\{x+R_o, u\}]$. Algorithm 3 is thus termed 'sphere-based MS' algorithm.

Although this type of algorithm is simple, it would not be effective if a stopping condition that prevents an exhaustive search of the search space is used. Besides allowing a reasonable number of iterations to be performed, $\mathbf{k}_{\max}$, the algorithm has an alternative stopping rule. The goal is to make the algorithm to stop when all roots have been located with certainty. Furthermore, it should not need to invoke the search procedure a large number of times to decide that all roots have been found. A simple condition uses an estimate of the fraction of uncovered space,

$$\mathbf{U}_s = \frac{r(r+1)}{t(t-1)}$$

where r is the number of different computed roots and t represents the number of times the search procedure has been invoked [19]. The MS algorithm then stops if $\mathbf{U}_s \leq \varepsilon$, for a small $\varepsilon > 0$.

Data: $\mathbf{k}_{\max}$, $\gamma_p \in (0,1)$, $\varepsilon < 1$, $\beta \ll 1$
Set $\Sigma = \emptyset$, $r = 1$
Randomly generate $x \in [l,u]$
Set $\mathscr{R} = B(x, \frac{\min_i\{u_i - l_i\}}{2})$
Compute $\chi_1 = \mathbf{HS}(x, \mathscr{R}, [l,u])$, set $t = 1$
Compute $R_1 = \|x - \chi_1\|$, set $\Sigma = \Sigma \cup \chi_1$, set $k = 1$
while $k \leq \mathbf{k}_{\max}$ **do**
  Randomly generate $x \in [l,u]$
  Set $o = \arg\min_{j=1,\ldots,r} d_j \equiv \|x - \chi_j\|$
  if $d_o < R_o$ **then**
    if *the direction from x to* χ_o *is ascent* **then**
      Set $p = 1$ and
      $(\hat{l}, \hat{u}) = (\max\{l, x-R_o\}, \min\{x+R_o, u\})$
    else
      Set $p = \phi_k$ using (16) and $(\hat{l}, \hat{u}) = (l,u)$
    end
  else
    Set $p = 1$ and
    $(\hat{l}, \hat{u}) = (\max\{l, x-R_o\}, \min\{x+R_o, u\})$
  end
  if $rand() < p$ **then**
    Set $\mathscr{R} = B(x, R_o)$
    Compute $\chi = \mathbf{HS}(x, \mathscr{R}, [\hat{l}, \hat{u}])$, set $t = t+1$
    if $\chi \notin \Sigma$ **then**
      Set $r = r+1$, $\chi_r = \chi$, $\Sigma = \Sigma \cup \chi_r$
      Compute $R_r = \|x - \chi_r\|$
    else
      Update $R_l = \max\{R_l, \|x - \chi_l\|\}$
    end
  else
    Update $R_o = \max\{R_o, \|x - \chi_o\|\}$
  end
  Set $k = k+1$
  if $\mathbf{U}_s \leq \varepsilon$ **then**
    STOP
  end
end

Algorithm 3: 'sphere-based MS' algorithm

4 Numerical results

In this section, we investigate the performance of the proposed 'sphere-based MS' method that uses the hybrid HS algorithm as the search procedure on a set of benchmark small and large dimensional problems. They are listed below as Problems 1 – 10. We set the parameters of the algorithms as follows:

– in the MS algorithm: $\varepsilon = 0.05$, $\gamma_p = 0.5$, $\beta = 0.001$, tol =0.5e-2 and $\mathbf{k}_{\max} = 30$ (unless otherwise stated);
– in the hybrid HS, HMS $= \{2n, 5n\}$, HMCR= 0.95, $F = 0.9$, NI is set to depend on the problem ($NI = 1000$ in Problems 1, 2 and 6, $NI = 50000$ in Problem 3, $NI = 2000$ in Problem 7, $NI = 10000$ in Problems 4 and 5, $NI = 10000n$ in Problems 8 and 9 and $NI = 2000n$ in Problem 10), and x^{best} is identified as a global minimizer if the merit function value falls

below 1e-10. We consider 1e-08 instead when solving Problems 8–10.

–in I-HS, additionally to the values described above, PAR_{min}=0.35, PAR_{max}=0.99, BW_{min}=0.000001 and BW_{max}=5.

The experiments were carried out on a PC Intel Core 2 Duo Processor E7500 with 2.9GHz and 4Gb of memory. The algorithms were coded in Matlab Version 8.0.0.783 (R2012b).

4.1 Comparative tests with small dimensional problems

We now consider a set of seven small dimensional problems. First, we use three examples to illustrate the behavior of the 'sphere-based MS' algorithm when trying to locate multiple roots.

Problem 1. [14]: A simple nonsmooth nonlinear system with two roots $(-0.5,0.5),(0.5,-0.5)$ in $[-3,3]^2$:

$$f(x)=\begin{cases} x_1^2-x_2^2=0 \\ 1-|x_1-x_2|=0 \end{cases}.$$

Fig. 1 illustrates the method on the merit function to locate the two global solutions. The generated points correspond to a specific run with $\mathbf{k}_{\max}=10$ and five calls to the hybrid HS algorithm. Solution $(-0.500004,0.500002)$ was obtained three times. The first time that the solution was located, the search procedure required 337 function evaluations (f.e.) and 0.019 seconds (sec.) to reach a merit value of 4.08e-11. Solution $(0.499993,-0.500000)$ was reached twice and the first time it was located a merit function of 8.73e-11 was obtained using 442 f.e. and 0.044 sec. The figure displays for each solution the initial point that converged for the first time to the solution (black '$\star$') the initial points that converged in second (or in third) place to the solution (blue '$\times$') and the points that were discarded and were not used to call the search procedure (magenta '$\circ$'). Overall, the 'sphere-based MS' algorithm required 2135 f.e. and 0.147 sec.

Problem 2. [10] This system of two nonlinear equations has three roots $(0,-2),(1,-1),(0.7071,-1.5)$ in $[-3,3]^2$:

$$f(x)=\begin{cases} x_1^2-x_2-2=0 \\ x_1+\sin(\frac{\pi x_2}{2})=0 \end{cases}.$$

The 'sphere-based MS' algorithm located three roots in 0.476 sec. and required 8526 f.e.. During this single run of 10 iterations, the hybrid HS search procedure was called five times. Fig. 2 illustrates the multistart method on the merit function to locate the three roots. The root $(-0.000009,-1.999999)$ was located twice, and the first time required 512 f.e. and 0.036 sec. to reach the merit value 9.99e-11. The solution $(0.707120,-1.499983)$ was located just once after 799 f.e., 0.043 sec. with a merit

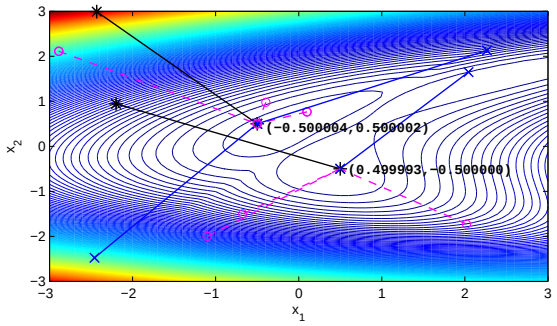


Fig. 1: Illustration of the 'sphere-based MS' on the merit function of Problem 1

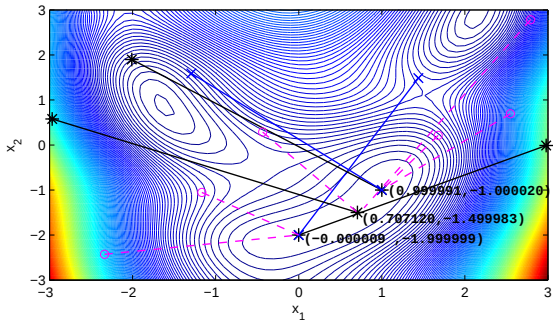


Fig. 2: Illustration of the 'sphere-based MS' on the merit function of Problem 2

value of 3.11e-11. Finally, the root $(0.999991,-1.000020)$ was located twice and the first time required 760 f.e., 0.050 sec. with a merit value of 8.68e-11.

Problem 3. [12] This is the Himmelblau problem and the number of roots in $[-5,5]^2$ is nine:

$$f(x)=\begin{cases} 4x_1^3+4x_1x_2+2x_2^2-42x_1-14=0 \\ 4x_2^3+2x_1^2+4x_1x_2-26x_2-22=0 \end{cases}.$$

Fig. 3 shows the nine located roots and displays the merit function value and the number of function evaluations (inside parentheses) for each found root in a single run of the algorithm. The 'sphere-based MS' algorithm run for 30 iterations and the **HS** search procedure was invoked 13 times. The first located root was (3.584428, -1.848126) with merit value of 9.99e-11 and after 0.858 sec. and was located again in the third iteration. The second root was (-0.270845, -0.923039) with merit 4.44e-11 and after 0.848 sec. and was again recovered in the 7th, 10th, 13th, 17th, 18th and 29th iterations. The third root (-2.805118, 3.131312) with merit 9.94e-11 was located after 0.302 sec., the fourth

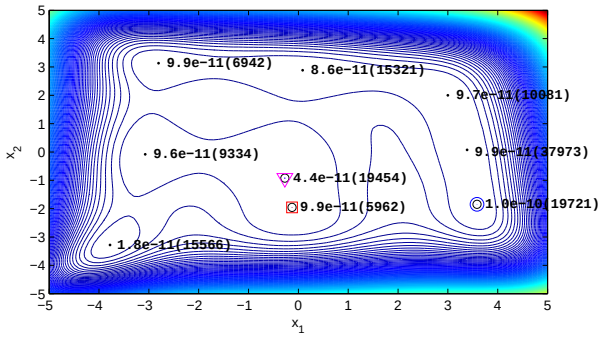


Fig. 3: Merit function values and number of function evaluations for the located roots of Problem 3

was (-3.779310, -3.283186) with merit 1.79e-11 and 0.676 sec., the fifth was (-0.127961, -1.953715) with merit 9.92e-11 and 0.306 sec. (and recovered again in the 14th and 20th iterations), the sixth was (3.000000, 2.000000) with a merit value of 9.73e-11 and 0.441 sec., the seventh (0.086677, 2.884255) with merit 8.57e-11 was located after 0.664 sec. and the last two (3.385154, 0.073851) with merit 9.85e-11 and (-3.073026, -0.081353) with merit 9.56e-11 were located after 1.645 and 0.407 sec., respectively. Overall, this run required 241724 f.e. and 10.541 sec. to locate all the roots of the problem.

To analyze further the convergence performance of the proposed algorithm, 30 independent runs were carried out and the average values of three performance criteria – number of located roots, 'n.r.', number of function evaluations, 'n.f.e.', and CPU time in seconds, 'time' – are reported. The results produced by the algorithm are also compared with other heuristics. To analyze the effect of different HS variants as well as some parameter values on the algorithm's convergence behavior, the next three tables contain the results of

Table 2 *i*) MS with $\mathscr{R} = [l,u], \gamma_p = 0.5$ invoking the hybrid HS as search procedure;
ii) MS with $\mathscr{R} = [l,u], \gamma_p = 0.5$ invoking I-HS as search procedure;

Table 3 *i*) Algorithm 3 with $(\hat{l},\hat{u}) = (l,u)$ in all cases, and $\gamma_p = 0.5$;
ii) Algorithm 3 with $(\hat{l},\hat{u}) = (l,u)$ in all cases, and $\gamma_p = 0.05$;

Table 4 *i*) Algorithm 3 as it is, with $\gamma_p = 0.5$;
ii) Other results from the literature.

Besides the previously described, other small dimensional problems are the following.

Problem 4. (Application in robotic) [10] This equation has six roots in $[-1,1]$

$$k_0 + k_2x^2 + k_4x^4 + k_6x^6 + (k_1x + k_3x^3 + k_5x^5)\sqrt{1-x^2} = 0$$

where $k_0 = 3.9852, k_1 = -8.8575, k_2 = -10.039, k_3 = 20.091, k_4 = 7.2338, k_5 = -11.177, k_6 = -1.17775$.

Problem 5. (Geometry problem) [9,10] The system has two roots in $[0,50]^3$

$$f(x) = \begin{cases} x_1x_2 + (x_1 - 2x_3)(x_2 - 2x_3) - 165 = 0 \\ \dfrac{x_1x_2^3}{12} - \dfrac{(x_1 - 2x_3)(x_2 - 2x_3)^3}{12} - 9369 = 0 \\ \dfrac{2(x_2 - x_3)^2(x_1 - x_3)^2x_3}{x_1 + x_2 - 2x_3} - 6835 = 0 \end{cases}.$$

Problem 6. (Floudas problem) [8,12,13] This system has two roots in $[0.25,1] \times [1.5,2\pi]$

$$f(x) = \begin{cases} 0.5\sin(x_1x_2) - 0.25\dfrac{x_2}{\pi} - 0.5x_1 = 0 \\ \left(1 - \dfrac{0.25}{\pi}\right)(\exp(2x_1) - e) + e\dfrac{x_2}{\pi} - 2ex_1 = 0 \end{cases}.$$

Problem 7. (Merlet problem) [8,12,13] The system has 13 roots in $[0,2\pi]^2$

$$f(x) = \begin{cases} -\sin(x_1)\cos(x_2) - 2\cos(x_1)\sin(x_2) = 0 \\ -\cos(x_1)\sin(x_2) - 2\sin(x_1)\cos(x_2) = 0 \end{cases}.$$

Table 2 aims to show the effect of two HS variants, the proposed hybrid HS and the I-HS, on the results produced by a more classical MS algorithm where $\mathscr{R} = [l,u]$ is the search space to generate the points of the HM and the constraint $[l,u]$ is the box to create the new harmony y. It is possible to conclude that both variants perform almost evenly although the hybrid HS is more efficient since it requires less function evaluations and CPU time mostly, and I-HS is more consistent in finding new roots.

Table 2: MS with $\mathscr{R} = [l,u]$, $\gamma_p = 0.5$: comparison of HS variants

	MS + hybrid HS			MS + I-HS		
Prob.	n.r.	n.f.e.	time	n.r.	n.f.e	time
1	1.9	10506	0.426	2.0	12254	0.469
2	2.6	26026	1.074	2.6	28289	1.091
3	6.3	491116	20.209	7.0	683488	26.259
4	5.0	159007	5.521	4.5	135745	4.313
5	1.9	90128	4.418	1.9	134951	5.801
6	1.8	12664	0.542	1.2	15277	0.623
7	6.5	33817	1.520	7.8	29052	1.150

Table 3 reports the results produced by the 'sphere-based MS' algorithm, in which the region $\mathscr{R}$ is a sphere and the constraint $[l,u]$ is always used as the search space to define the new harmony y, i.e., $(\hat{l},\hat{u}) = (l,u)$ in all the three considered cases (see Algorithm 3). Results from the second, third and fourth columns correspond to $\gamma_p = 0.5$ and the remaining correspond to $\gamma_p = 0.05$. As previously stated, the lowest value of γ_p yields fewer calls to the **HS** search procedure and consequently less function evaluations and smaller

Table 3: Algorithm 3 with $(\hat{l},\hat{u}) = (l,u)$ in all three cases

	$\gamma_p = 0.5$			$\gamma_p = 0.05$		
Prob.	n.r.	n.f.e.	time	n.r.	n.f.e	time
1	2.0	7265	0.321	2.0	5000	0.218
2	3.0	14159	0.613	2.8	9824	0.450
3	5.4	388304	16.888	4.6	291348	12.922
4	4.4	96165	3.728	4.3	76227	3.039
5	1.9	58257	3.027	1.7	35240	1.879
6	2.0	12928	0.575	1.7	8448	0.386
7	4.4	881	0.038	4.1	575	0.024

execution time. Thus, efficiency can be improved by reducing the value of the parameter γ_p. However, we observe from the table that the average number of located roots has slightly decreased. Here the main goal is to improve consistency (in locating different roots) even if with additional computational requirements. When a comparison is made with the results of Table 2, it is observed that the efficiency has in general slightly improved, the consistency has improved with Problems 1, 2 and 6 but has worsened with Problems 3, 4 and 7.

Finally, Table 4 reports the average results produced by the 'sphere-based MS' algorithm, as it is designed in Algorithm 3, and aims to compare with those from the literature [8,9,10,12,13,14], where '–' indicates that the information is not available. We note that the results reported in [8] are obtained after 100 runs and those in [10] seem to correspond to one run. In [13], a sample size of 20 randomly generated points is used and the two sets of values shown in the table correspond to the best and the worst results obtained with two different stopping rules in the MS algorithm. It is noteworthy that the results produced by the Algorithm 3 in Table 4 are better than those shown in previous tables.

During these 30 runs, the two roots of Problem 1 were located by the 'sphere-based MS' algorithm in all runs. On average, each run of 30 iterations invoked the **HS** search procedure 11 times, required 7283 f.e. and 0.323 sec. to locate both roots, as reported in Table 4. However, the best run required 4003 f.e. and 0.183 sec. and the worst 9445 f.e. and 0.416 sec.

The 'sphere-based MS' algorithm located the three roots of Problem 2 in all runs. Fig. 2 displays the position of the roots in the search space $[-3,3]^2$. The **HS** search procedure was invoked on average 13.1 times, and the average number of f.e. and time were 15908 and 0.719 sec., respectively. The best run required 7159 f.e and 0.320 sec. and the worst 21989 f.e. and 1.165 sec.

The nine roots of Problem 3 were located in two out of the 30 runs. The roots displayed in Fig. 3 and reported above when describing Problem 3 were obtained in the best of these two runs. The algorithm found eight roots in 11 runs, seven roots in 13 runs and six roots in four runs. Considering the 30 runs, the hybrid HS algorithm was invoked on average 16.7 times and the average number of

Table 4: Comparative results

	Algorithm 3			other works			
Prob.	n.r.	n.f.e.	time		runs	n.r.	n.f.e	time
1	2.0	7283	0.323	[14]	–	2†	–	–
2	3.0	15908	0.719	[10]	1	3	–	–
3	7.4	460192	19.917	[12]	30	9	253877	5
4	5.3	147664	5.345	[10]	1	6	–	–
5	2.0	83187	4.367	[9]	–	2‡	–	–
				[10]	1	1	–	–
6	2.0	10968	0.505	[8]	100	2	–	0.461
				[13]	30	2	69627	0.53
					30	2	4273	0.03
				[12]	30	2	211652	0.607
7	7.9	18857	0.881	[8]	100	13	–	20
				[13]	30	13	8033	0.08
					30	13	3297	0.03
				[12]	30	13	401021	46

† the reported solutions are not the global minimizers of $\mathscr{M}$.
‡ only the best solutions are reported; one solution is outside Ω.

located roots, the average number of function evaluations and the average CPU time are reported in Table 4.

The six roots -0.999627, -0.880777, 0.483253, 0.876481, 0.958852 and 0.991557, with merit function values ranging from 4.86e-12 to 1.11e-10, of the Problem 4 have been identified in the best of the ten runs that located six different roots. The run invoked the **HS** procedure 10 times, required 75186 f.e. and 2.998 sec. The remaining 20 runs located five roots. On average, the **HS** search procedure was invoked 18.4 times, and the algorithm required 147664 f.e. and 5.345 sec.

When solving Problem 5 the two roots were located in all runs. The best run required 55119 f.e., 2.921 sec. and 11 calls to the **HS** procedure. The **HS** procedure needed 0.532 sec. to locate the root (43.154623, 10.128917, 12.944063) with merit 2.01e-09, and 0.188 sec. to locate (7.602990, 24.541978, 11.576714) with merit 6.97e-12. The worst of the 30 run invoked 15 times the **HS** procedure, required 106218 f.e. and 5.508 sec. Considering all the runs, the average number of calls to the **HS** procedure was 12.2 and the algorithm required an average of 83187 f.e. and 4.367 sec.

The two roots of Problem 6 were found in all runs. The best run invoked the **HS** procedure 6 times, lasted for 0.218 sec. and used 4722 f.e.; the worst invoked the **HS** procedure 15 times, lasted for 0.616 sec. and required 13824 f.e. During the best run, the root (0.299447, 2.836914) with merit value 7.27e-11 was located after 731 f.e. and 0.032 sec. and (0.499997, 3.141584) with merit 4.51e-11 was located after 582 f.e. and 0.026 sec. Overall, each run required on average 0.505 sec., 10968 f.e. to find the two roots (see Table 4) and invoked on average 12.9 times the **HS** procedure.

When solving Problem 7, the 13 roots (0, 0), (3.141582, 6.283185), (4.712388, 4.712389), (6.283185,

0), (1.570791, 1.570801), (6.283185, 6.283185), (9.885e-08, 3.141593), (0, 6.283185), (3.141593, 3.141595), (1.570776, 4.712411), (6.283185, 3.141584), (3.141589, 9.480e-07), and (4.712362, 1.570822) were obtained although not all in the same run. The best run that found 10 roots, invoked the hybrid HS algorithm 15 times, required 10148 f.e. and lasted 0.470 sec. The times required by the 10 calls to the search procedure that converged to the 10 different roots were 0.0009, 0.0002, 0.0003, 0.029, 0.094, 0.038, 0.0098, 0.0006, 0.094 and 0.094 sec. Nine roots were found in six runs, eight roots in 13 runs and seven roots in ten runs. The average number of roots found by the 'sphere-based HS' algorithm was 7.9 after 18857 f.e. and 0.881 sec. (see Table 4).

4.2 Problem with a large number of roots

We now aim to analyze the performance of the proposed 'sphere-based HS' algorithm when a large number of roots are present. The following problem is particularly interesting since the number of roots increases with the magnitude of the set Ω.

Problem 8. [13] This problem is known as Effati-Grosan_1 and has been tested for different values of a in the set $\Omega = [-a,a]^2$. Although the exact number of roots is unknown, it has been reported one root when $a = 2$, 13 roots when $a = 10$ and 127 when $a = 100$ [13]:

$$f(x) = \begin{cases} \cos(2x_1) - \cos(2x_2) - 0.4 = 0 \\ 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 1.2 = 0 \end{cases}.$$

Table 5 reports the average results produced by Algorithm 3 when solving the three instances of Problem 8. We compare our results with two sets of results reported in [13]. These correspond to the best and the worst results obtained with different stopping rules in the MS algorithm. We remark that the therein used multistart approach is based on a quasi-Newton BFGS variant as local search procedure. Thus, the convergence speed is expected to be higher than that of our algorithm, since approximations to first and second derivatives are used.

On average, each run of 30 iterations invoked the **HS** search procedure 9 times, required 10785 f.e. and 0.623 sec. to locate one root of the instance identified with $a = 2$, as reported in Table 5. When solving the instance with a set to 10, we allowed the Algorithm 3 to run for a maximum of 200 iterations. On average, the **HS** search procedure was invoked 59 times and an average of 12.5 roots were found. The best run of the set found the 13 roots after invoking the **HS** procedure 61 times, where each one required an average of 4406 f.e. and 0.258 sec., with a merit value of 3.528e-09. Finally, when the instance with $a = 100$ was solved, the algorithm was allowed to run for a maximum of 500 iterations, invoked on average the **HS** search procedure 364.6 times and was able to locate on average 117.2 roots. The best run found 120 roots after invoking the **HS** procedure 360 times, where each one required on average 36155 f.e. and 1.992 sec., with a merit function value of 3.524e-09.

Table 5: Results from Algorithm 3 and [13], when solving the 3 instances of Problem 8

	Algorithm 3			[13]		
a	n.r.	n.f.e.	time	n.r.	n.f.e	time
2	1	1.1e+04	0.623	1	4.0e+03	0.04
				1	3.7e+03	0.03
10	12.5	2.82+05	15.63	13	2.0e+04	0.15
				13	5.3e+03	0.04
100	117.2	1.2e+07	692.1	127	1.6e+05	1.35
				127	7.3e+04	0.59

4.3 Tests with large dimensional problems

To analyze the convergence behavior of the Algorithm 3 when solving large dimensional problems, we use two problems with varied dimensions. With each problem we test the algorithm with the following dimensions 10, 20, 30, 40.

Problem 9. [13] This is the Yamamutra problem that has three roots in $[-2,2]^n$, and it can be tested for different values of n:

$$f_i(x) = x_i - \frac{1}{2n}\left(\sum_{j=1}^{n} x_j^3 + i\right) = 0,\ i = 1,\ldots,n.$$

Table 6 reports the average results produced by Algorithm 3 when solving the four instances of Problem 9. We also report the results available in [13] for comparison. The 'emphasized' values inside parentheses show the average number of function evaluations and the average time in seconds required by each call of the **HS** search procedure. Besides the criteria 'n.r.', 'n.f.e.' and 'time', the table also shows the 'HMS' value used in these experiments. When solving the instance with $n = 10$, the **HS** procedure was invoked an average of 10.4 times and the average number of located roots was 2.6. On the other hand, when solving the instance with $n = 20$, the **HS** procedure was invoked an average of 10.9 times and the average number of located roots was 2.4. When solving the instance with $n = 30$, the best run found the 3 roots after 16 calls to the **HS** procedure. Overall the averaged values of the three criteria are 1.4 ('n.r.'), 2.1e+06 ('n.f.e.') and 614.04 ('time'), as shown in Table 6. Finally, when setting $n = 40$ in Problem 9, the algorithm located an average of 1.5 roots, after 9.4 calls to the **HS** search procedure, and required 3.0e+06 f.e. and 1077.1 sec. Further, the best run that found 3 roots required 5.18e+06 f.e., 1.82e+03 sec. and invoked 16 times the **HS** procedure.

Table 6: Results from Algorithm 3 and [13], when solving the 4 instances of Problem 9

	Algorithm 3				[13]		
n	HMS	n.r.	n.f.e.	time	n.r.	n.f.e	time
10	10	2.6	1.5e+05	41.68	3	1.9e+05	4.47
			(*1.4e+04*)	(*4.01*)	3	1.6e+04	0.36
20	10	2.4	3.6+05	69.56	3	1.2e+05	7.27
			(*3.3e+04*)	(*6.38*)	3	2.3e+04	1.46
30	30	1.4	2.1e+06	614.04	3	1.5e+05	17.89
			(*2.4e+05*)	(*68.99*)	3	2.8e+04	3.53
40	40	1.5	3.0e+06	1077.1	3	1.7e+05	37.55
			(*3.2e+05*)	(*114.58*)	3	3.3e+04	6.89

Problem 10. [27]: Consider the Broyden tridiagonal system, a frequently used problem for testing sensitivity to provided initial approximations, which has been tested for different values of n and has one root in $[-1,0]^n$:

$$f(x)=\begin{cases} f_1(x)=(3-2x_1)x_1-2x_2+1=0 \\ f_i(x)=(3-2x_i)x_i-x_{i-1}-2x_{i+1}+1=0, i=2,\ldots,n-1 \\ f_n(x)=(3-2x_n)x_n-x_{n-1}+1=0 \end{cases}.$$

Table 7 reports the average results produced by Algorithm 3 when solving the four instances of Problem 10 (when n is set to 10, 20, 30, 40). This table also includes the average number of calls to the **HS** procedure, 'n.calls'. With this example we aim to analyze the effect of the size of the harmony memory in the hybrid HS-based algorithm. We note here that the root was found in all runs whatever the instance and HMS. From the results we conclude that the HMS does not affect the performance of the 'sphere-based HS' algorithm.

Table 7: Results from Algorithm 3, when solving the 4 instances of Problem 10

		Algorithm 3			
n	HMS	n.calls	n.r.	n.f.e.	time
10	10	4.9	1	8.0e+04	7.22
20	10	4.5	1	1.6e+05	21.72
	20	4.8	1	1.7e+05	23.66
30	10	4.9	1	2.7e+05	50.02
	20	4.4	1	2.4e+05	44.85
	30	3.9	1	2.2e+05	41.11
40	10	3.8	1	2.9e+05	68.64
	20	4.6	1	3.5e+05	82.55
	30	4.4	1	3.4e+05	79.22
	40	3.4	1	2.6e+05	60.49

According to the results presented in the last four tables, we may conclude that the proposed 'sphere-based MS' algorithm is able to locate multiple roots of small as well as large dimensional systems of nonlinear equations. Since the hybrid HS-based search procedure does not require any derivative information, the algorithm is general-purpose and can be applied even to nonsmooth problems.

5 Conclusions

In this paper, a hybrid HS-based MS method to compute multiple roots of a system of nonlinear equations has been presented. The method, denoted by 'sphere-based MS' algorithm, relies on some new ideas aiming to increase the exploration power of an MS algorithm and enhance the exploitation of more restrictive regions of the search space. The search procedure that is invoked in the MS paradigm is a global search algorithm from the metaheuristic class. The name 'hybrid HS' algorithm comes from the hybridization of a classical HS algorithm with a *mutation* operator present in the DE algorithm. Furthermore, self-adaptive rules for the parameters PAR and BW of the pitch adjustment operator are proposed. The main differences between the proposed MS-type algorithm and the others in the literature lie in the regions that are provided to the search procedure to look for a new solution (instead of the usual search space Ω). In the HS context, those regions that are used to create the HM are spheres centered at the sampled point with appropriate radius. Further, the new harmony is generated inside a box that may coincide with the given box constraints of the problem or a more restrictive box, depending on the position of the sampled point relative to the previously located solutions. From the numerical experiments carried out with a set of ten benchmark small and large dimensional problems we may conclude that the proposed 'sphere-based MS' algorithm with the hybrid HS search procedure is effective in locating multiple roots and competitive when compared with other metaheuristics.

Acknowledgement

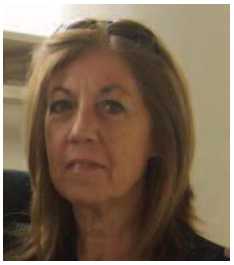
The authors are grateful to the anonymous referees for their helpful suggestions to improve the paper. This research has been supported by CIDEM (Centre for Research & Development in Mechanical Engineering, Portugal), by COMPETE POCI-01-0145-FEDER-007043 and FCT (Foundation for Science and Technology, Portugal) within the projects UID/EMS/0615/2016 and UID/CEC/00319/2013.

References

[1] J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall Inc., 1983.
[2] M.D. González-Lima and F.M. Oca, Numerical Algorithms **52**, 479–506 (2009).
[3] J.M. Martínez, Journal of Computational and Applied Mathematics **124**, 97–122 (2000).
[4] U. Nowak and L. Weimann, A family of Newton codes for systems of highly nonlinear equations, Technical Report. Tr-91-10, K.-Z.-Z. Inf. Berlin, 1991.

[5] G.C.V. Ramadas and E.M.G.P. Fernandes, 13th International Conference Computational and Mathematical Methods in Science and Engineering, J.V. Aguiar et al. (Eds.), ISBN: 978-84-616-2723-3 Vol. IV, 1176–1186, June 2013.
[6] G.C.V. Ramadas and E.M.G.P. Fernandes, 11th International Conference of Numerical Analysis and Applied Mathematics 2013 AIP Conf. Proc. Vol. 1558, 582–585 (2013).
[7] C.H. Chen, 2003 Joint Conference on AI, Fuzzy System and Gray System, Taipei, Taiwan, 4–6 (2003).
[8] M.L. Hirsch, P.M. Pardalos and M. Resende, Nonlinear Analysis: Real World Applications **10**, 2000–2006 (2009).
[9] M. Jaberipour, E. Khorram and B. Karimi, Computers and Mathematics with Applications **62**, 566–576 (2011).
[10] E. Pourjafari and H. Mojallali, Swarm and Evolutionary Computation 4, 33–43 (2012).
[11] G.C.V. Ramadas and E.M.G.P. Fernandes, International Journal of Computer Mathematics **89**, 1847–1864 (2012).
[12] R.M.A. Silva, M.G.C. Resende and P.M. Pardalos, Journal of Global Optimization **60**, 289–306 (2014).
[13] I.G. Tsoulos and A. Stavrakoudis, Nonlinear Analysis: Real World Applications **11**, 2465–2471 (2010).
[14] C. Grosan and A. Abraham, IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans **38**, 698–714 (2008).
[15] C. Grosan and A. Abraham, International Journal of Innovative Computing, Information and Control **4**, 2161–2170 (2008).
[16] M.M. Ali and M.N. Gabere, Journal of Computational and Applied Mathematics **233**, 2661–2674 (2010).
[17] I.G. Tsoulos and I.E. Lagaris, Computer Physics Communications **174**, 166–179 (2006).
[18] W. Tu and R.W. Mayne, International Journal for Numerical Methods in Engineering **53**, 2239–2252 (2002).
[19] C. Voglis and I.E. Lagaris, Applied Mathematics and Computation **213**, 1404–1415 (2009).
[20] Z.W. Geem, J.H. Kim and G. Loganathan, Simulation **76**, 60–68 (2001).
[21] M. Mahdavi, M. Fesanghary and E. Damangir, Applied Mathematics and Computation **188**, 1567–1579 (2007).
[22] M.G.H. Omran and M. Mahdavi, Applied Mathematics and Computation **198**, 643–656 (2008).
[23] R. Storn and K. Price, Journal of Global Optimization **11**, 341–359 (1997).
[24] K.S. Lee and Z.W. Geem, Computational Methods and Applied Mechanical Engineering **194**, 3902–3933 (2004).
[25] O.M. Alia and R. Mandava, Artificial Intelligence Review **36**, 49–68 (2011).
[26] R. Marti, Multi-start methods, In: Handbook of Metaheuristics, F. Glover, G. Kochenberger (Eds), Kluwer Academic Publishers, 355–368 (2003).
[27] J. More, B. Garbow and K. Hillstrom, ACM Transactions on Mathematical Software **7**, 17–41 (1981).

Gisela C.V. Ramadas is an Adjunt Professor at the Department of Mathematics in the School of Engineering, Polytechnic of Porto. She received the PhD degree in Production and Systems Engineering from the University of Minho in 2004. Her research interests are in the area of applied mathematics including the metaheuristics for nonlinear systems of equations. She has published more than 10 research papers in indexed international journals, book chapters and conference proceedings in the mathematical and engineering sciences areas.



Edite M.G.P. Fernandes is a retired Full Professor at the University of Minho. She received the PhD degree in Mathematics from the University of Oxford in 1980 and her Habilitation in Systems Engineering and Industrial Processes from the University of Minho in 2002. At present, her main research interests are in the areas of global optimization and constraint handling. She has published more than 40 papers in reputed international journals of applied mathematics and optimization and 56 papers as book chapters in the field of nonlinear optimization.

Ana Maria A.C. Rocha is an Assistant Professor at the University of Minho. She received the PhD degree in Production and Systems Engineering from the University of Minho in 2005. Her research interests include global optimization, stochastic methods, penalty techniques and dynamic systems optimization. She has published more than 20 papers in indexed international journals, 23 as book chapters and 20 in indexed conference proceedings.