Self-adaptive combination of global tabu search and local search for nonlinear equations

Gisela C.V. Ramadas and Edite M.G.P. Fernandes

Abstract

Solving systems of nonlinear equations is a very important task since the problems emerge mostly through the mathematical modelling of real problems that arise naturally in many branches of engineering and in the physical sciences. The problem can be naturally reformulated as a global optimization problem. In this paper, we show that a self-adaptive combination of a metaheuristic with a classical local search method is able to converge to some difficult problems that are not solved by Newton-type methods.

Keywords

Nonlinear equations; metaheuristic; tabu search; Hooke and Jeeves method

1. Introduction

In this paper, we consider solving the nonlinear system of equations

$$F(x) = 0, \quad F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^{\mathrm{T}} \tag{1}$$

where $F : \Omega \subset \mathbb{R}^n \to \mathbb{R}^n, \Omega \equiv [l, u] = \{x : l_i \leq x_i \leq u_i, i = 1, \ldots, n\}$ and the functions $f_i(x)$, $i = 1, 2, \ldots, n$ are continuously differentiable, using a combination of a metaheuristic with a classical local search method. Some problems in engineering, chemistry, physics, medicine and even economic areas, aim at determining the roots of a system of equations. In general, these problems are nonlinear and difficult to solve. The most famous techniques to solve nonlinear equations are based on Newton's method [6,10,14,18,38,45]. They require analytical or numeri- cal first derivative information. Newton's method is the most widely used algorithm for solving nonlinear systems of equations. It is computationally expensive, in particular if n is large, since the Jacobian matrix and the solution of a system of linear equations are required at each itera- tion. The quasi-Newton methods use less-expensive iterations than Newton, but their convergence properties are not very different. In general, quasi-Newton methods avoid either the necessity of computing derivatives, or the necessity of solving a full linear system per iteration or both tasks

[39]. In [19], a new technique for solving systems of nonlinear equations reshaping the system as a multiobjective optimization problem is proposed. The authors applied a technique of evolutionary computation to solve the problem obtained after the change. In [20], the authors proposed techniques for computing all the multiple solutions in nonlinear systems. Another technique to solve systems of nonlinear equations is presented in [25], where a heuristic continuous global optimization GRASP is applied. A genetic algorithm is proposed in [8]. Filled functions that guarantee convergence to global solutions are available in the literature to solve systems such as (1), see, for example, [51] and some references therein cited. The problem of solving a nonlinear system of equations can be naturally formulated as a global optimization problem. Problem (1) is equivalent, in the sense that it has the same solution, to finding the globally smallest value of the l_2-norm error function, usually known as merit function, related to solving the system of equations (1), defined by

$$\underset{x\in\Omega\subset\mathbb{R}^n}{\text{minimize}} M(x) \equiv \|F(x)\|_2. \tag{2}$$

Here, the global minimum, and not just a local minimum, of the objective function $M(x)$, in the set Q, is to be found. We denote the global minimum by x^*. Classical local search methods, like Newton-type methods, have some disadvantages, when compared with global search methods. In particular,

- the final solution is heavily dependent on the initial approximation of the iterative process;
- they can be trapped in local minima;
- they require differentiable properties of all the equations of the nonlinear system.

We use Example 1.1 to show this local trap behaviour.

Example 1.1 Consider the following system of nonlinear equations:

$$\begin{aligned} f_1(x_1, x_2) &\equiv x_1 - \sin(2x_1 + 3x_2) - \cos(3x_1 - 5x_2) = 0, \\ f_2(x_1, x_2) &\equiv x_2 - \sin(x_1 - 2x_2) + \cos(x_1 + 3x_2) = 0, \end{aligned} \tag{3}$$

and Figure 1 that shows the graphical representation of the l_2-norm error function $M(x)$. The multi-modal nature of M makes the process of detecting a global minimum a difficult one. We

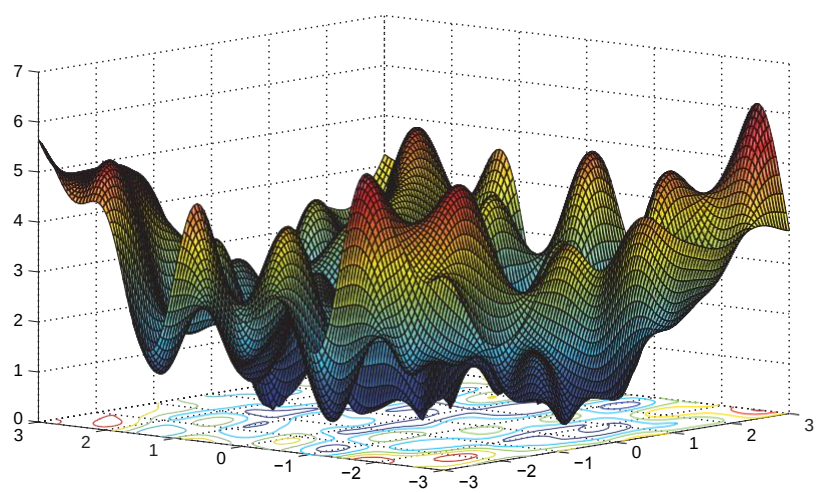

Figure 1. Graphical representation of $M(x)$, from Example 1.1.

Table 1. Solutions obtained by `fsolve` from MATLAB for different initial approximations, for Example 1.1.

Initial approximation	'exitflag'	(f_1, f_2) at solution	Number of iterations	Number of function evaluations
(0, 0)	1	(−4.1e−13, −2.4e−13)	5	18
(1, 1)	1	(3.6e−9, 2.3e−9)	6	21
(0, 1)	−2	(−3.0e−2, 9.4e−1)	27	60
(2, 2)	−2	(1.1e−1, 8.8e−1)	31	70
(−1, 1)	−2	(−5.5e−1, 6.5e−1)	55	130
(1, −1)	−2	(1.0e−2, −4.0e−2)	24	57
(−1, −1)	−2	(−5.2e−1, 1.0e−1)	31	74
(2, −2)	−2	(−1.6e−1, −5.7e−1)	32	71
(−2, −2)	−2	(−1.7e−1, −1.5e00)	34	77

solve Example 1.1 by `fsolve` from MATLAB™, using nine different initial approximations. In this MATLAB solver, the default trust-region dogleg algorithm with no analytical Jacobian is used [44]. The solver is able to converge to the solution only twice, although all the nine initial points are in the neighbourhood of the solution. Table 1 shows the results obtained from MATLAB. The first column in the table presents the tested initial approximations and the second column lists the value of the output parameter 'exitflag' of MATLAB. The value '1' means that the method converged to a root where the first-order optimality measure is less than a pre-specified tolerance, and '−2' means that it converged to a point which is not a root and the sum-of-squares of the function values is greater than or equal to a pre-specified tolerance.

Thus, to be able to converge to a global solution, a global search strategy is required. The most important global search techniques invoke exploration and exploitation search procedures aiming at:

(i) diversifying the search in all the search space;
(ii) intensifying the search in promising areas of the search space.

Local optimization techniques guarantee globality only under certain convexity assumptions. Nonconvex problems exhibit multiple global and local (nonglobal) solutions and are more efficiently solved by global optimization methods. Preventing premature convergence to a local solution, while trying to locate a global optimum, is an important property of global solution methods. Research concerning the problem of finding the global optimum of a continuous objective function over a compact convex set started in the early 1970s. A well-established classification of solution methods for global optimization defines two classes: the exact methods and the approximate ones. The reader is referred to papers with reviews on advances in global optimization [1,13,46]. Exact methods for global optimization are guaranteed to find an optimal solution within a problem dependent run time. Thus, they are able to solve a problem with a required accuracy in a finite number of steps. A theoretical analysis of convergence to a global optimum may be provided. On the other hand, approximate methods are not guaranteed to find an optimal solution although they are often able to find very good solutions in a reasonable time. Most approximate methods are stochastic. Exact methods are often known as deterministic. Unlike the stochastic methods, the outcome of a deterministic algorithm does not depend on pseudo-random variables. The design of a deterministic method relies on the mathematical attributes of the optimization problem; therefore, the performance depends heavily on the structure of the problem and the complexity grows very fast with problem's dimension [23,37]. Popular deterministic approaches have been proposed within the branch-and-bound framework to solve certain types of nonconvex problems, see, for example [3]. Other deterministic methods use concepts of DIviding RECTangle [29] and Interval Analysis. In the interval-based methods, the interval arithmetic is used to update variable bounds and compute bounds for the involved function values [23].

Another interesting idea which has been under discussion in the global optimization area, since the 1980s, uses auxiliary functions, namely Tunneling and Filled functions. The objective function is transformed into an auxiliary function using previously located local minimizers. To escape from local minimizers, gradient-based methods can then be used to descend from the local to a global one [54].

Stochastic methods rely on probabilistic elements, either in the problem data or in the algorithm itself, or in both. Compared with deterministic methods, the implementation of stochastic algorithms is often easier. In general, stochastic methods require no structural information about the problem at hand. For a challenging class of global optimization problems, the so-called black-box optimization problems, no structure is known and used when targeting the global solution. The convergence proofs for this type of methods involve the use of probability theory. Stochastic algorithms are at most able to provide a probabilistic convergence guarantee, for instance, convergence in mean square [28,48], convergence in probability, or convergence with probability one [27]. In practice, there is no guarantee that the obtained solution is actually the global one, or by how far the algorithm has failed to locate the true global solution.

A stochastic method is based on random searches that use selected heuristics to promote the search over the feasible set and guide the choice of the most promising candidate solutions. The general consensus about the word heuristic is that this is a procedure based on commonsense rules aimed at increasing the probability of solving a specific problem and providing an approximate solution with no guarantee that it is close to the optimal solution. Although heuristics are closely associated with random search techniques, there are also deterministic heuristic methods. A selected sequence of well-known stochastic methods, in chronological order, follows: evolution strategy (1965), genetic algorithm (1975), scatter search (1977), simulated annealing (1983), tabu search (TS, 1986), ant colony optimization (1992), particle swarm optimization (1995), differential evolution (1997), harmony search (2001), electromagnetism-like mechanism (2003), artificial bee colony (2005) and artificial fish swarm optimization (2005). They are mainly inspired by nature or by evolutionary and swarm intelligence theories. A survey on stochastic methods is presented in the textbook [53].

Recent hybrid algorithms are aimed at combining a heuristic algorithm with a local search operator, either a deterministic or a random search, to enhance its exploitation ability [5,21,22,47, 49]. A Multistart method is a hybrid, where local searches are performed starting from randomly generated points. The idea of clustering paired with multistart that appeared in the 1970s with great success aims at avoiding to perform a local search from another point that is closely associated with a found optimum solution [50]. In recent years, another type of hybrid strategy adds a second heuristic to the basic heuristic's design, for example, the Ant Colony System and the TS [30], the Harmony Search cooperating with Differential Evolution [36] or the integration of Scatter Search and TS [12].

A well-known class of global search techniques, the metaheuristics, use random procedures that invoke artificial intelligence tools and simulate nature behaviours. The word 'metaheuristics' is used to describe all approximate methods that combine basic heuristic methods into higher level frameworks to explore the search space in an efficient and effective manner. In the last three decades, these heuristics have proven to be computationally successful in solving combinatorial problems as well as continuous global optimization problems. Thus, the practical advantages of metaheuristics are their effectiveness and general applicability. Due to their random features, metaheuristics have, in general, slow convergence since they may fail to detect promising search directions in the neighbourhood of a global minimum. There are two classes of metaheuristics. A population-based heuristic that defines and maintains throughout the iterative process a set of solutions. The most known population-based heuristic is the Genetic Algorithm [17]. A point-to-point heuristic defines just one solution at the end of each iteration which will be used to start the next iteration. Simulated Annealing [21] and TS [15] are two examples of point-to-point methods.

The TS is a metaheuristic developed primary for solving combinatorial problems [15,16]. The TS introduced by Cvijovic´ and Klinowski [9] for continuous optimization guides the local search out of local optima and has the ability to explore new regions. It is an iterative procedure that maintains a list of the movements most recently made, avoiding in subsequent iterations the execution of movements that lead to solutions already known to have been visited. Usually, the slow convergence of TS is overcome by incorporating a classical local search strategy into the main algorithm. In general, this type of hybridization occurs in the final stage of the iterative process when the solution is in the vicinity of the global solution. An example of such a method is presented in [22]. The therein proposed method, called directed TS (DTS), uses strategies, like the Nelder–Mead (NM) method [43] and the adaptive pattern search (APS) [21], to direct a TS.

This paper is aimed at assessing the performance of the metaheuristic TS when solving a system of nonlinear equations (1), using the function $M(x)$ as a measure of the progress of the algorithm towards the solution. According to the formulation (2), this means that the fitness of each trial solution x is assessed by evaluating the merit function M at x. And, a solution x is better than y if $M(x) < M(y)$. In this paper, and due to the reported success when solving global optimization problems of the form (2), the DTS variant of the TS is extended to be able to solve nonlinear systems of equations. In particular, we aim at analysing the behaviour of TS-type methods when solving some difficult problems that are not solved by Newton-type methods.

Furthermore, we propose a new algorithm that combines DTS and a local search method in order to accelerate convergence to the solution. The issue related with the condition that defines the choice between the exploration and exploitation phases of the algorithm is also addressed with new self-adaptive weight factors. The numerical results show the goodness of the proposed self-adaptive strategies.

The paper is organized as follows. Section 2 briefly describes the DTS method, Section 3 overviews the classical local search method, known as Hooke and Jeeves (HJ), and Section 4 presents the proposed combined DTS and HJ searches algorithm, and the new self-adaptive strategies to choose between the exploration and exploitation phases. The numerical results and their discussion are included in Section 5.

2. The metaheuristic TS

2.1 Basic TS

TS is an iterative process which operates in the following way. The algorithm starts with a randomly generated initial solution, x, and by applying pre-defined moves in its neighbourhood, it generates a set Y of solutions. The objective function to be minimized is evaluated at all solutions in Y, and the best of all, y^{best}, becomes the current solution, $x \leftarrow y^{best}$ (even if it is worse than x). Accepting uphill moves, the algorithm avoids to get trapped in a local minimum. The previous procedure is repeated until a given stopping condition is reached. Furthermore, the algorithm also stops when the solution does not improve for a specified number of iterations. To avoid cycling, since a point already visited may be generated again, a set of points already visited are stored in a list, called tabu list (TL). The solutions in Y that belong to the TL are eliminated. This TS structure is known as short-term memory TS. The use of this type of flexible memory turns out to be advantageous, since the method is able to keep diversity, like population-based methods, in contrast to the rigid structures, like those used in branch-and-bound methods, or the lack of memory present in the simulated annealing method [7]. To improve the performance, long-term memory TS structures have been proposed to record important attributes such as the elite and the frequently visited solutions. The DTS [22] implemented in this paper for solving nonlinear systems of equations contains long-term memory structures.

2.2 Directed TS

The DTS method of Hedar and Fukushima [22] uses direct search methods in order to stabilize the search especially in the vicinity of a local minimum. Two variants of the DTS are therein proposed: one is based on the NM method, as a local search, inside the exploration step of the algorithm, and the other uses the APS strategy in the exploration step. Furthermore, the Kelley's modification of the NM method [31] is still used in the therein called intensification search, in the final stage of the process. We note that the DTS method can be classified as a multi-start method. The multi-start methods are powerful search procedures to guide both global exploration and local search. The DTS method is based on three main procedures: exploration, diversification and intensification search. The structure of the DTS is shown in Algorithm 2.1. (See details in [22].)

Algorithm 2.1 DTS algorithm

Require: randomly generated x^0
1: **while** the stopping criteria are not reached **do**
2: Exploration search procedure
3: Neighbourhood search
4: Local search
5: Solution update
6: Diversification search procedure
7: **end while**
8: Intensification search procedure

The main loop (outer cycle) of the DTS method, consisting of the exploration and diversification search procedures, begins with an initial randomly generated solution, but other initial approximation may be provided.

2.2.1 Exploration search

The exploration search aims to explore the search space Q. It uses direct search methods as neighbourhood search and local search strategies to generate trial points. These may be based on either the simplex method of NM or on the APS strategy. Here, we use the APS variant, as described in [21].

The cycling procedure is prevented not only with the standard TL but also with the inclusion of two novel concepts of tabu regions. The DTS method implements four TS memory elements: The multi-ranked TL, the tabu region, the semi-tabu region and the visited region list (VRL). They are long-term memory structures and are very important since they allow the method in the diversification search and intensification search procedures to behave as an intelligent search technique.

2.2.2 Diversification search

A diversification procedure aims to generate a new initial trial point outside the visited regions. The information stored in the VRL is used to direct the search towards new regions. This VRL serves as a diversification tool, with the aim of diversifying the search for areas that have not been visited in the search space.

2.2.3 Intensification search

When one of the best obtained trial solutions is sufficiently close to a global minimum, or its value has not been changed for a specified number of iterations, then the intensification search procedure is applied, at the final stage of the algorithm, to refine the best solution visited so far. In this case, DTS uses the Kelley's modification of the NM method [31,32]. A solution still closer to the global minimum is then obtained.

3. HJ local search method

The derivative-free method, known as the HJ method, is a deterministic local pattern search method that performs, at each iteration j, a series of exploratory moves along the coordinate axes around a current approximation, x^j, in order to find a new approximation $x^{j+1} = x^j + \lambda^j s^j$, with a lower merit function value while maintaining the approximation inside the set Q. Here, the index j is used for the iteration counter of this inner iterative process. The scalar λ^j represents the step length and the vector s^j determines the direction of the step. The step length is reduced whenever the previous iteration is unsuccessful, i.e. when no improvement in M is obtained, and it is maintained otherwise. The HJ method also performs a pattern move whenever a successful iteration is encountered. The vector $x^{j+1} - x^j$ defines a promising direction and a pattern move is then implemented, which means that the exploratory move is carried out around the trial point $x^{j+1} + (x^{j+1} - x^j)$, rather than around the current point x^{j+1}. Then, if the coordinate search is successful, the returned point is accepted as the new point; otherwise, the pattern move is rejected and the method reduces to a coordinate search around x^{j+1}. The reader is referred to [26] for the details concerning this classical local search method.

4. Combining global TS and local search

The herein proposed hybridization defines an algorithm that is able to combine two types of cycles: a global and a local one. The general idea is borrowed from the algorithm presented in [14], where a classical gradient-based quasi-Newton nonmonotone strategy is used to solve nonlinear systems of equations. The global search cycle is carried out by a modified version of DTS method, where the HJ local search method is used, instead of the NM method, as the intensification search procedure of the DTS algorithm (see Step 8 of Algorithm 2.1). On the other hand, the local search cycle uses the HJ pattern search method alone [26] (Section 3).

The most important issue here is to decide when to carry out a global exploration of the search space, or a local exploitation in the neighbourhood of a good approximation to the solution.

4.1 Merit function sufficient reduction

Our first proposal concerned with the condition that decides which cycle should be carried out, at each iteration k, depends on a sufficient reduction obtained in the merit function, when compared with the merit function value of the previous iteration. If a pre-specified sufficient reduction is verified, a local exploitation cycle is to be required, since a fast downhill progress has been detected. Thus, a promising region seems to be found by the algorithm. Otherwise, the region does not look promising yet and an exploration cycle is more appropriate where a diversifying search is to be carried out looking for a promising region with a global minimum. Figure 2 shows the general iterative structure of the combined global TS and local search (GTSLS) method. Details of the procedures are discussed below.

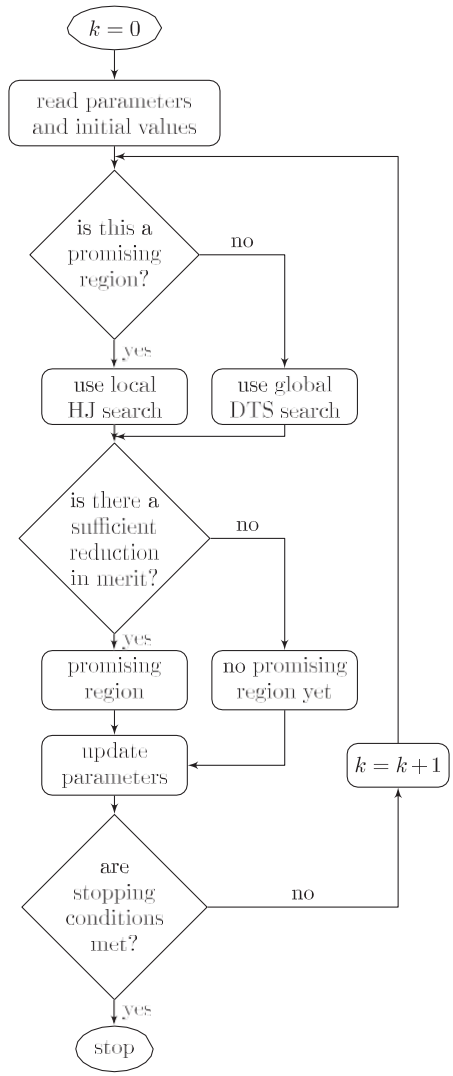

Figure 2. General flowchart of the GTSLS method.

In both global and local cycles of the algorithm, only an η_k-approximation x^{k+1} to the optimal solution is required, i.e. each search terminates when, at iteration k, the value of the merit function at the best solution found so far is less than η_k. The sequence of η_k values should decrease and approach zero, as k increases. The argument is that, when the iterative process begins far from the solution of problem (1), there is no point in spending too much resources computing an approximation with high accuracy. However, as the process approaches the solution, highly accurate approximations are important to speed the overall process. Algorithm 4.1 summarizes the main steps of the proposed GTSLS algorithm.

Condition $M(x^{k+1}) \leq \gamma_1 M(x^k)$ in the algorithm defines the sufficient reduction that we aim to observe in the merit function. The closer γ_1 is to 1, the smaller is the required reduction. In order to guarantee that the progress has been along a downhill step and a promising region has been located, we define $\gamma_1 = 0.5$. The other parameters in the algorithm are set as follows: $\eta^* = 10^{-6}$ (accuracy of the solution) and $k_{\max} = 10n$. A successful run is registered when the algorithm stops

Algorithm 4.1 GTSLS algorithm

Require: $x^0, 0 < \gamma_1, \gamma_2 < 1, 0 < \eta^* \ll 1, k_{\max} > 0$; set $k = 0$, $\eta_0 = 1$, flag=0;
1: **while** $M(x^k) > \eta^*$ and $k \leq k_{\max}$ **do**
2: **if** flag=1 **then**
3: use local HJ search to compute an η_k-approximation x^{k+1}
4: **else**
5: use global DTS search to compute an η_k-approximation x^{k+1}
6: **end if**
7: **if** $M(x^{k+1}) \leq \gamma_1 M(x^k)$ **then**
8: set flag=1
9: **else**
10: set flag=0
11: **end if**
12: set $k = k + 1$, set $\eta_k = \max\{\eta^*, \gamma_2 \eta_{k-1}\}$
13: **end while**

due to the condition $M(x^k) \leq \eta^*$. Two values of γ_2 were tested, 0.1 and 0.5, see Section 5. Each iterative process, either in the local or in the global cycle of the algorithm, is called inner cycle, in contrast to the process indexed by k, called outer cycle.

4.2 Self-adaptive weight factors

Another approach is attempted to improve the efficiency of the GTSLS algorithm using a self-adaptive weight factor to check when the global exploration of the search space, or the local exploitation, is needed. At each iteration k, the weight factor is evaluated based on an 'index', I_k, defined by

$$I_k = \frac{M(x^0)}{M(x^k)} - 1 \tag{4}$$

that clearly shows the proximity of x^k to x^*. This 'index' changes according to the rate of the observed improvement on the merit function value. In the first iteration ($k = 0$), I_0 is zero but as x^k converges to x^*, I_k approaches infinity, since the required global optimum is 0. Since the self-adaptive weight factor w_k aims at checking which type of search is required, at iteration k, we use the idea of a transfer function [41]

$$w_k = \frac{1}{1 + \exp(-1/\alpha I_k)}, \tag{5}$$

where $0 < \alpha \leq 1$ is a constant, to define adequate weight values that vary as follows: $0.5 \leq w_k < 1$. A large weight factor, which emerges with a small I_k, means that $M(x^k)$ is very far from the merit function optimal value, and a global exploration search is required. On the other hand, a small w_k emerging from a large value of I_k means that a local exploitation search is needed since x^k is already near the optimal solution. The parameter α gives the speed of reduction in w. Figure 3 shows how w_k varies with α. Five values of α were used. The bigger the α is, the faster is the convergence to the lower bound of w. Preliminary experiments have shown that a moderate speed of reduction in w favours the efficiency of the algorithm. Another issue is related with a reference value w_R, of the weight factor w, below which the exploration of the space is no longer necessary and the exploitation around the neighbourhood of the best solution found so far is crucial for a

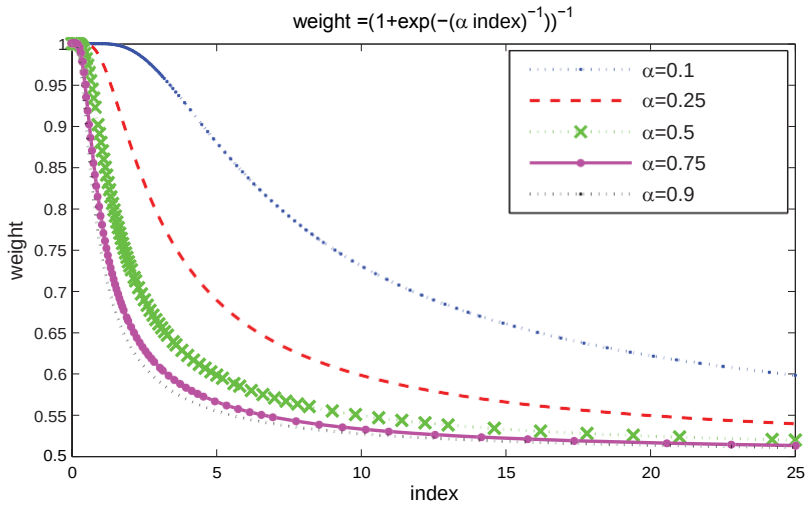

Figure 3. Weight factor w from (5) *vs.* 'index' for five different values of α.

highly accurate solution. From the numerical experiments done so far, the value 0.75 is a good choice. We remark that in this self-adaptive weight factor context, the condition on M in Step 7 of the Algorithm 4.1 is replaced by the following condition: $w_{k+1} \leq w_R$.

To overcome the need to define the parameter α in the formula for the weight factor (5), another idea borrowed from the work presented in [2] defines a weight factor ranging from 0.5 to 1 as follows:

$$w_k = 0.5\left(1 + \tanh\left(\frac{M(x^k)}{M(x^0)}\right)\right). \qquad (6)$$

The speed of reduction in the weight (6), as a function of the 'index' (4), behaves similar to that of weight (5), when $0.5 \leq \alpha < 1$. Figure 4 depicts these behaviours. We now solve Example 1.1 to analyse the effect of parameter α in the performance of the algorithm. We compare the results of the proposed GTSLS algorithm with those of DTS and HJ used separately. The notation

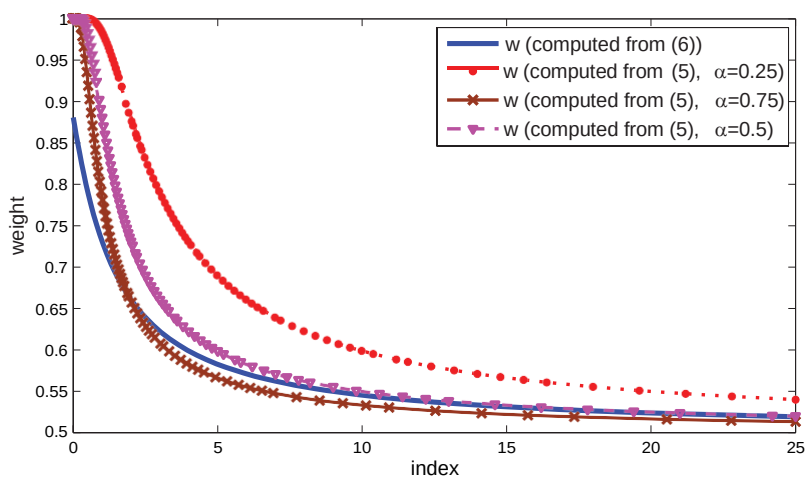

Figure 4. Speed reduction comparison: weights w from (5) and (6).

Table 2. Results from GTSLS, DTS and HJ, for Example 1.1.

Algorithm		k	k_{DTS+HJ}	k_{HJ}	$n.f.ev.$	M
GTSLS**1**		9	160	46	1186	5.0e−7
GTSLS**2**	($\sigma = 0.25, w_R = 0.75$)	7	91	1	414	3.3e−7
	($\sigma = 0.50, w_R = 0.75$)	8	129	6	665	2.8e−7
	($\sigma = 0.75, w_R = 0.75$)	11	188	6	985	8.8e−7
GTSLS**3**	($w_R = 0.75$)	8	114	1	500	2.8e−7
	($w_R = 0.60$)	8	122	6	645	8.2e−7
DTS		20	40	20	280	1.5e−3
HJ		20			99	2.1e−2

concerned with different versions of GTSLS is the following: GTSLS**1** for the combined algorithm as presented in Algorithm 4.1, GTSLS**2** for the adaptive version that uses (5) and GTSLS**3** for the version using (6) (Table 2). Due to the stochastic nature of TS-based algorithms, the problem was run 30 times and the best of the 30 obtained solutions is registered. In the table, 'k' represents the number of outer (or main) iterations, 'k_{DTS+HJ}' is the total number of inner iterations, 'k_{HJ}' is the total number of iterations required by HJ alone, '$n.f.ev.$' represents the average number of function evaluations, over the 30 runs, required to reach the presented solution, and 'M' is the value of the merit function at the obtained registered solution. The solutions reached by DTS and HJ do not have the required accuracy. We may conclude that the self-adaptive versions of the combined TS and local search algorithm – GTSLS**2** and GTSLS**3** – outperform GTSLS**1**.

To further analyse the effect of both formulae (5) and (6) on the performance of the algorithm GTSLS, a second example is used.

Example 4.1 Consider the following system of nonlinear equations

$$\begin{aligned} f_1(x_1, x_2) &\equiv 4x_1^3 + 4x_1x_2 + 2x_2^2 - 42x_1 - 14 = 0, \\ f_2(x_1, x_2) &\equiv 4x_2^3 + 4x_1x_2 + 2x_1^2 - 26x_2 - 22 = 0, \end{aligned} \tag{7}$$

where $-5 \le x_1, x_2 \le 5$ [51]. Figure 5 shows the graphical representation and the contours of $M(x)$ relative to this example. The merit function M has nine minima. The solutions are reported on the contour plot.

We test both versions GTSLS**2** and GTSLS**3**, with three initial approximations, and compare with `fsolve` and the results in [51] (Table 3). The information reported in this table is similar to that of Table 2 with the additional information concerned with the solution x^*. In the table, '–' stands for unavailable information. GTSLS**3** always converges to the same solution whatever the initial approximation, while GTSLS**2** detects three solutions. The algorithm in [51] and the solver `fsolve` converge to three different solutions depending on the provided initial approximation.

4.3 Escaping from local minima

To further check whether the proposed algorithm is able to escape from local minima or from valleys that contain them, while searching for a global one, two small nonconvex optimization problems are selected from the literature (Examples 4.2 and 4.3 [23]).

Example 4.2 This function is the well-known six-hump camelback function

$$f(x) = 4x_1^2 - 2.1x_1^4 + \tfrac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \quad \text{with } Q = [-2, 2] \times [-2, 2],$$

which has six minimizers and the two global solutions are located at $(-0.089842, 0.71266)$ and $(0.089842, -0.71266)$ with a value of -1.0316.

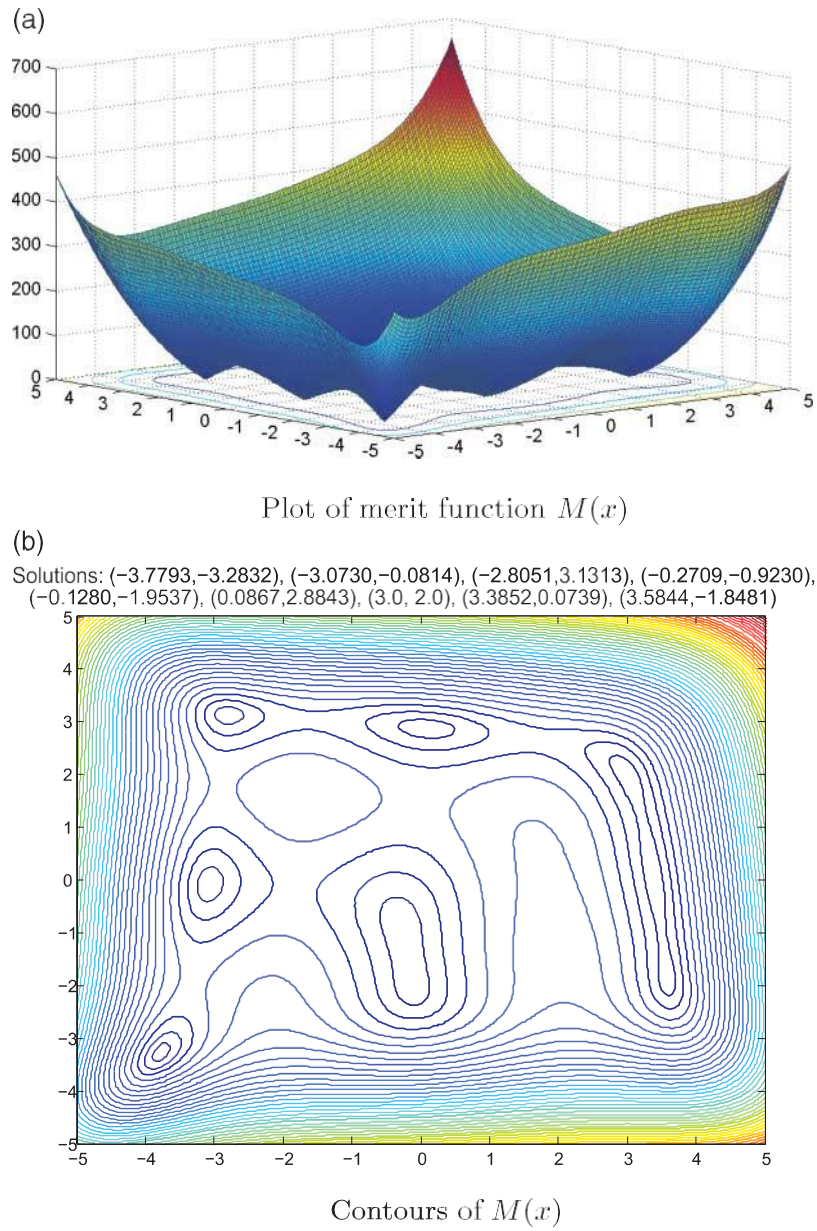

Figure 5. Plot of $M(x)$ and contours, from Example 4.1.

Example 4.3 The bi-spherical function is nonsmooth and has one global minimizer at (1, 0) with $f = 0$ and a local minimizer at $(-1, 0)$ with $f = 0.1$:

$$f(x) = \min\{(x_1 - 1)^2, (x_1 + 1)^2 + 0.1\} + x^2 \text{ with } Q = [-2, 2] \times [-1, 1].$$

By allowing uphill moves, the exploration nature of the DTS algorithm searches beyond a local minimum. Furthermore, by preventing the algorithm from visiting again the points previously evaluated, a better exploration of the problem space can be enforced. We illustrate the behaviour of the

Table 3. Results from self-adaptive GTSLS, `fsolve` and [51], for Example 4.1.

Algorithm	k	$k_{\text{DTS+HJ}}$	$n.f.ev.$	x^*	M
Initial approximation $(-5,-3)$					
GTSLS**2**	6	68	380	$(-0.1280, -1.9537)$	7.3e−7
GTSLS**3**	8	116	566	$(-0.2708, -0.9230)$	9.5e−7
in [51]	3		–	$(-3.7793, -3.2832)$	3.8e−7
`fsolve`	5		18	$(-3.7793, -3.2832)$	3.5e−10
Initial approximation $(1, 3)$					
GTSLS**2**	7	84	449	$(3.5844, -1.8481)$	6.04e−7
GTSLS**3**	6	74	396	$(-0.2708, -0.9230)$	7.3e−7
in [51]	3		–	$(0.0867, 2.8843)$	6.8e−7
`fsolve`	4		15	$(0.0867, 2.8843)$	7.1e−10
Initial approximation $(2, 3)$					
GTSLS**2**	7	91	486	$(-0.2708, -0.9230)$	2.5e−7
GTSLS**3**	8	115	565	$(-0.2708, -0.9230)$	6.6e−7
in [51]	3		–	$(3.3852, 0.0739)$	7.9e−7
`fsolve`	5		18	$(3.0, 2.0)$	5.8e−7[a]

[a]First-order optimality measure.

GTSLS**3** algorithm on the two multi-modal functions over convex closed feasible sets and compare with the solver `fmincon` from MATLAB. Figure 6 contains the contours of both functions and depicts the convergence behaviour starting from four different initial approximations.

Figure 6(a) shows the convergence behaviour of GTSLS**3** on Example 4.2 starting from the initial points, x^0: $(2, 2)$, $(-2, 0)$, $(-1.7036, 0.7961)$ and $(1.2302, 0.1623)$. These two last points are local minima. The algorithm is able to explore all the feasible region, escape from the local minima and converge to global minima. Table 4 contains the results obtained by the GTSLS**3** algorithm and `fmincon` from MATLAB for comparison. We may observe that the solver `fmincon` could not escape from the two local minima, converges just once to a global minimum and identifies a saddle point.

When starting from the initial points $(0, 0)$, $(-1, 1)$, $(-2,-1)$ and $(-0.5, 0)$, the GTSLS**3** algorithm is able to converge to the global minimum of Example 4.3 in all cases, while `fmincon` locates the local one in two cases. See, Table 4 for details. Figure 6(b) illustrates the iterated points obtained by GTSLS**3**.

5. Numerical results and discussion

To analyse the performance of the proposed combined GTSLS algorithm, we selected, coded in MATLAB™, and solved by the version GTSLS**3**, 99 test problems from the literature. For comparative purposes, the problems were also solved by `fsolve` from MATLAB, the HJ method, and the extended version of the DTS method presented in [22]. The problems in our database are referred to as P1, P2, . . ., P99. They represent systems of nonlinear equations of different sizes and complexity. P99 and P49 are Examples 1.1 and 4.1, respectively. The results of the numerical experiments were obtained in a personal computer with an AMD Turion 2.20 GHz processor and 3GB of memory.

Since the computational effort required in each outer/main iteration, by the algorithms in comparison, is different, we choose to stop the algorithms when the number of function evaluations exceeds $100n^2$. Meanwhile, if a solution is found with a merit function value less or equal to $\eta^*(=10^{-6})$ (tolerance for M at the best solution found so far) the algorithms stop. This way, we aim at analysing and comparing the accuracy of the obtained solutions.

To resume the main achievements of our numerical experiments, we show in Table 5 the results of 22 problems (from the database of 99 problems) for which `fsolve` was not able to converge

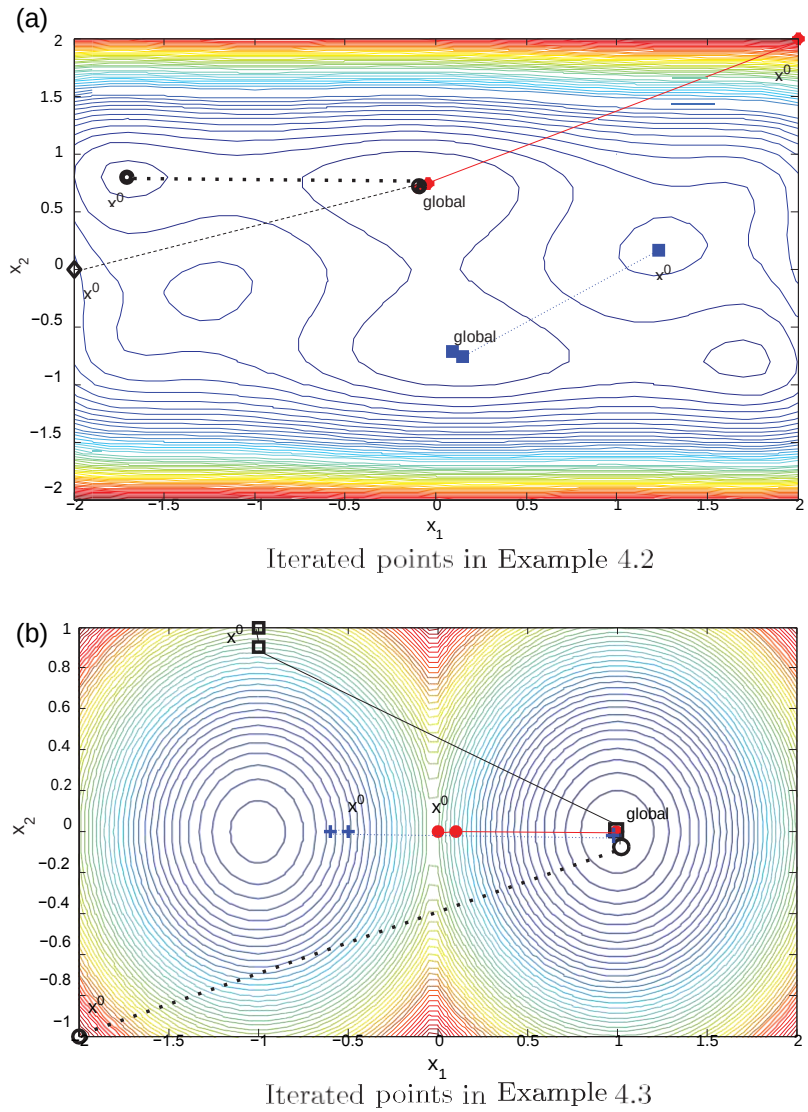

Figure 6. Iterated points from different initial approximations. (a) Iterated points in Example 4.2. (b) Iterated points in Example 4.3.

to a solution with a specified tolerance within $10n$ iterations or $100n$ function evaluations (values defined by default). The output parameter 'exitflag' is '–2' for P44 and P72 and is '0' for all the others. All problems were solved with a specific initial approximation: the null vector for P50 and P52, the vector of all ones for P56, P62, P63, P70, P77 and P84, and the initial vector provided in the literature for the remaining problems. Although the literature does not provide the bounds $[\ell, u]$ for most problems, we selected a specific range for each problem, as shown in Table 6. The lower and upper limits are the same for all the components of the vector x.

For the methods HJ, DTS, GTSLS**3** with $\gamma_2 = 0.1$ and GTSLS**3** with $\gamma_2 = 0.5$, Table 5 shows the value of M (at the best solution found after $100n^2$ function evaluations) and the required number of outer/main iterations. When the parameter γ_2 is set to 0.1 in GTSLS**3**, the algorithm

Table 4. Convergence results for Examples 4.2 and 4.3.

Example	x^0	GTSLS**3** algorithm Solution	f	k (*n.f.ev.*)	`fmincon` from MATLAB Solution	f
4.2	(2, 2)	$(-0.0898, 0.7128)$	−1.03163	3 (60)	$(-0.1\text{E}{-6}, 0.01\text{E}{-6})$[a]	8E−14
	$(-2, 0)$	$(-0.0899, 0.7127)$	−1.03163	4 (103)	$(-0.0898, 0.7127)$	−1.03163[b]
	$(-1.7036, 0.7961)$	$(-0.0899, 0.7128)$	−1.03163	4 (122)	$(-1.7036, 0.7961)$	−0.2155[c]
	(1.2302, 0.1623)	$(0.0897, -0.7127)$	−1.03163	4 (101)	(1.2302, 0.1623)	2.4963[c]
4.3	(0, 0)	(0.9993, 0)	4E−7	2 (25)	(1, 0)	7E−17[b]
	$(-1, 1)$	$(0.9996, -0.0003)$	2E−7	4 (108)	$(-1, 0)$	0.1[c]
	$(-2, -1)$	$(0.9997, -0.0001)$	9E−8	4 (93)	$(0.9999, -0.0001)$	7E−9[b]
	$(-0.5, 0)$	(0.9998, 0.0008)	6E−7	3 (77)	$(-1, 0)$	0.1[c]

[a] Saddle point.
[b] Global minimum.
[c] Local minimum.

Table 5. Results from `fsolve`, HJ, DTS and self-adaptive GTSLS**3**.

	`fsolve`				HJ		DTS		GTSLS**3**, $\gamma_2 = 0.1$			GTSLS**3**, $\gamma_2 = 0.5$	
	n	M	*n.f.ev.*	k	M	k	M	k	M	k	$M_{(k\le 10n)}$	M	k
P6	2	2.5e−3	57	20	1.0e−4	30	2.8e−4	28	1.0e00	3	1.6e−4	1.0e00	3
P9	2	7.0e00	49	20	7.0e00	73	9.1e−4	28	8.0e−5	6	8.9e−7	3.0e−4	13
P25	10	1.4e+1	1001	90	8.8e−1	455	4.3e−1	291	1.0e−1	7	5.9e−2	8.8e−2	21
P26	3	6.6e−1	106	30	6.6e−1	47	1.7e−4	70	7.0e−7	7	5.9e−7	3.1e−5	15
P27	3	1.0e−1	109	30	7.9e−2	77	7.9e−2	67	2.0e−2	6	7.9e−2	8.4e−2	16
P44	33	2.2e00	1170	47	2.2e00	47	9.4e−1	1055	1.3e00	32	1.1e00	9.4e−1	40
P45	33	4.0e−1	3306	104	2.6e−1	77	1.6e00	883	2.7e−1	8	2.7e−1	2.7e−1	24
P50	2	2.5e−1	47	20	2.5e−1	37	2.5e−1	26	2.5e−1	3	2.5e−1	2.5e−1	4
P52	5	2.2e−3	261	50	3.7e−2	237	3.3e−1	137	1.1e−1	4	8.5e−5	5.6e−1	4
P53	6	2.8e00	427	60	2.8e00	296	1.4e−2	134	2.2e−3	6	1.0e−4	1.9e−2	15
P55	6	1.4e+2	385	60	1.4e+2	40	1.4e+2	66	1.1e+7	2	1.4e+2	7.8e+4	10
P56	8	4.9e−1	609	80	2.6e−6	190	3.7e−5	219	2.8e−6	7	2.8e−6	3.6e−6	20
P62	6	4.7e−2	379	60	7.0e12	5	1.2e10	50	1.2e14	1	5.9e+2	9.6e11	1
P63	6	6.2e−3	427	60	3.0e−4	277	4.1e−6	157	2.5e−6	12	7.4e−6	2.4e−7	20
P64	8	3.3e00	657	80	4.1e−5	99	4.8e−5	235	1.3e−5	13	8.0e−5	1.1e−5	22
P70	10	2.6e+1	981	100	2.1e+1	61	1.0e+1	288	2.1e+1	3	1.0e+1	3.6e+1	20
P72	51	2.7e00	1989	50	2.7e00	47	1.8e00	1611	6.6e−1	50	9.4e−1	6.6e−1	57
P77	2	4.6e00	45	20	4.6e00	36	7.4e−1	25	7.4e−1	7	7.4e−1	7.4e−1	14
P80	3	6.3e−2	124	30	1.6e−5	140	9.0e−4	69	9.3e−6	7	9.1e−6	1.7e−4	17
P84	3	5.0e−4	124	30	3e−21	24	3.9e−13	19	7.6e−9	2	2.3e−8	3.4e−8	1
P88	10	7.5e−1	66	5	3.9e−2	479	5.6e−2	288	1.6e−2	5	7.5e−4	1.9e−2	16
P96	2	2.8e00	47	20	1.0e−6	41	2.9e−5	27	6.9e−7	7	5.7e−7	1.9e−5	16

requires less iterations and attains in general higher accuracy results than with $\gamma_2 = 0.5$. This is expected since outer iterations based on a larger tolerance reduction provide solutions with higher accuracy although computationally more expensive. Overall the accuracy of the results is not yet as we expected. We then run GTSLS**3** for a maximum of $10n$ outer iterations and obtained the solutions reported in the 12th column of the table identified as '$M_{(k\le 10n)}$'. The results show some improvements, but further research is still needed.

To avoid the search along the coordinates during the HJ local exploitation search of the GTSLS algorithm, another local search procedure will be used in the near future. Since a componentwise search requires large amounts of function evaluations, the idea proposed in the random walk with direction exploitation method, recently applied to the stochastic differential evolution algorithm [36], will be considered.

Table 6. Characteristics of the problems.

	n	$[l_i, u_i]$	Reference	Problem's name in cited paper
P6	2	$[-10, 10]$	[42]	P3-Powell badly scaled function
P9	2	$[-10, 15]$	[42]	P2-Freudenstein and Roth function
P25	10	[10,50]	[14]	D6-Shifted and augmented trigonometric function with an Euclidian sphere
P26	3	$[-10, 10]$	[14]	D7-Diagonal of three variables premultiplied by a quasi-orthogonal matrix
P44	33			
P27	3	$[-10, 10]$	[14]	D8-Diagonal of three variables premultiplied by an orthogonal matrix, combined with inverse trigonometric function
P45	33			
P50	2	$[-10, 10]$	[25]	pp. 2004
P52	5	$[-20, 20]$	[40]	Combustion of propane chemical equilibrium equations
P53	6	$[-3, 3]$	[42]	14-Wood function
P55	6	$[-10, 10]$	[45]	Semiconductor boundary condition
P56	8	$[-10, 30]$	[4]	2.3-The human heart dipole
P62	6	$]0,60]$	[24]	Problem 2
P63	6	$[-10, 10]$	[35]	Example 2
P64	8	$[-10, 15]$	[11]	Equation 3.1
P70	10	$[-100, 100]$	[52]	Example 4.1 – Nonlinear resistive circuit
P72	51	$[-5, 5]$	[14]	D7-Diagonal of three variables premultiplied by a quasi-orthogonal matrix
P77	2	[0,3.5]	[8]	Example 1
P80	3	[0,4]	[42]	5-Beale function
P84	3	[0,1]	[34]	Example 6.2
P88	10	$[-10, 10]$	[33]	Example 2 – The Beam problem
P96	2	$[-10, 10]$	[18]	pp. 498 (Problem N4)

In this paper, we show that nonlinear systems of equations can be effectively solved by implementing a global optimization method to the merit function, which represents the l_2-norm error function related to the solving of the system of equations. The application of an extended version of the metaheuristic DTS, proposed in [22], for solving complex and difficult nonlinear systems of equations has been analysed and tested. We also propose a novel TS-type algorithm that combines the extended DTS, for a global exploration search, with the local HJ search algorithm, for the exploitation search procedure. At each iteration, the choice between implementing a global search or a local search relies on self-adaptive weight factors that depend on the rate of improvement on the merit function value. The numerical results allow us to conclude that the herein proposed self-adaptive GTSLS algorithm is able to converge to the solution of some difficult problems that were not successfully solved by Newton-type methods and outperforms both HJ local search and DTS algorithms.

Acknowledgements

The authors thank an anonymous referee and the editor whose comments and suggestions helped to improve the paper.

This research has been supported by CIDEM (Centre for Research & Development in Mechanical Engineering, Portugal), FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) Project PEst-OE/EME/UI0615/2011, FEDER COMPETE (Programa Operacional Fatores de Competitividade/Operational Programme Thematic Factors of Competitiveness) and FCT Project FCOMP-01-0124-FEDER-022674.

References

[1] A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht (eds.), *Foundations of Computational Intelligence: Global Optimization*, Vol. 3, Studies in Computational Intelligence Vol. 203, Springer-Verlag, Berlin, Heidelberg, 2009.

[2] A. Alfi, *PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems*, Acta Automatica Sinica 37(5) (2011), pp. 541–549.
[3] I.P. Androulakis, C.D. Maranas, and C.A. Floudas, *αBB: A global optimization method for general constrained nonconvex problems*, J. Global Optim. 7 (1995), pp. 337–363.
[4] B.M. Averick, R.G. Carter, and J.J. Moré, *The Minpack-2 Test Problem Collection (Preliminary Version)*, Technical Memorandum N. 150, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1991.
[5] S. Babaie-Kafaki, R. Ghanbari, and N. Mahdavi-Amiri, *Two effective hybrid metaheuristic algorithms for minimization of multimodal function*, Int. J. Comp. Math. 88(11) (2011), pp. 2415–2428.
[6] E. Bodon, A. Del Popolo, L. Lukšan, and E. Spedicato, *Numerical performance of ABS codes for systems of nonlinear equations*, Tech. Rep. DMSIA 01/2001, Universitá Degli Studi Di Bergamo, Bergamo, Italy, 2001.
[7] L. Cavique, C. Rego, and I. Themido, *Estruturas de vizinhança e procura local no problema de clique máxima (in portuguese)*, Investigação Operacional 22 (2002), pp. 1–18.
[8] C.H. Chen, *Finding roots by genetic Algorithms*, 2003 Joint Conference on AI, Fuzzy System and Gray System, Taipei, Taiwan, 2003, pp. 4–6.
[9] D. Cvijovic´ and J. Klinowski, *Taboo search: An approach to the multiple minima problem*, Science 267 (1995), pp. 664–666.
[10] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall Inc., New Jersey, 1983.
[11] M. Drăgan, *On solving large sparse systems of nonlinear equations using threads*, Proceedings of NMCM2002, An Euro Conference on Numerical Methods and Computational Mechanics, Miskolc, Hungary, 2002, pp. 49–56.
[12] A. Duarte, R. Martí, F. Glover, and F. Gortazar, *Hybrid scatter tabu search for unconstrained global optimization*, Ann. Oper. Res. 183(1) (2011), pp. 95–123.
[13] C.A. Floudas and C.E. Gounaris, *A review of recent advances in global optimization*, J. Global Optim. 45 (2009), pp. 3–38.
[14] A. Friedlander, M.A. Gomes-Ruggiero, D.N. Kozakevich, J.M. Martínez, and S.A. Santos, *Solving nonlinear systems of equations by means of Quasi-Newton methods with a nonmonotone strategy*, Optim. Meth. Software 8 (1997), pp. 25–51.
[15] F. Glover, *Future paths for integer programming and links to artificial intelligence*, Comput. Oper. Res. 13(5) (1986), pp. 533–549.
[16] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.
[17] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, 1989.
[18] M.D. González-Lima and F.M. Oca, *A Newton-like method for nonlinear system of equations*, Numer. Algorithms 52(3) (2009), pp. 479–506.
[19] C. Grosan and A. Abraham, *A new aproach for solving nonlinear equations systems*, IEEE Trans. Syst. Man Cyber. – Part A: Syst. Humans 38(3) (2008), pp. 698–714.
[20] C. Grosan and A. Abraham, *Multiple solutions for a system of nonlinear equations*, Int. J. Innov. Comput. Inf. Control 4(9) (2008), pp. 2161–2170.
[21] A. Hedar and M. Fukushima, *Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization*, Optim. Meth. Software 19 (2004), pp. 291–308.
[22] A. Hedar and M. Fukushima, *Tabu search direct by direct search methods for nonlinear global optimization*, Eur. J. Oper. Res. 170 (2006), pp. 329–349.
[23] E.M.T. Hendrix and B. G.-Tóth, *Introduction to Nonlinear and Global Optimization*, Springer-Verlag, Berlin, Heidelberg, 2010.
[24] K.L. Hiebert, *An evaluation of mathematical software that solves systems of nonlinear equations*, ACM Transact. Math. Software 8(1) (1982), pp. 5–20.
[25] M.L. Hirsch, P.M. Pardalos, and M. Resende, *Solving systems of nonlinear equations with continuous grasp*, Nonlinear Anal. Real World Appl. 10 (2009), pp. 2000–2006.
[26] R. Hooke and T.A. Jeeves, *Direct search solution of numerical and statistical problems*, J. Assoc. Comp. 8 (1961), pp. 212–229.
[27] M. Ji and J. Klinowski, *Convergence of taboo search in continuous global optimization*, Proc. Roy. Soc. A 462 (2006), pp. 2077–2084.
[28] M. Jiang, Y.P. Luo, and S.Y. Yang, *Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm*, Info. Proc. Lett. 102 (2007), pp. 8–16.
[29] D.R. Jones, *Direct global optimization algorithm*, in *Encyclopedia of Optimization*, 2nd ed., C.A. Floudas and P.M. Pardalos, eds., Springer-Verlag, Berlin, Heidelberg, 2009, pp. 725–735.
[30] A. Karimi, H. Nobahari, and P. Siarry, *Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions*, Comput. Optim. Appl. 45 (2010), pp. 639–661.
[31] C.T. Kelley, *Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition*, SIAM J. Optim. 10 (1999), pp. 43–55.
[32] C.T. Kelley, *Iterative Methods For Optimization*. *Frontiers in Applied Mathematics*, Vol. 18, SIAM, Philadelphia, PA, 1999.
[33] C.T. Kelley, L. Qi, X. Tong, and H. Yin, *Finding a stable solution of a system of nonlinear equations arising from dynamic systems*, J. Ind. Manag. Optim. 7(2) (2011), pp. 497–521.
[34] L.N. Kolev, *Interval Methods for Circuit Analysis*, World Scientific, Singapore, 1993.

[35] L.V. Kolev, *An improved interval linearization for solving nonlinear problems*, Numer. Algorithms 13(1–4) (2004), pp. 213–224.
[36] T.W. Liao, *Two hybrid differential evolution algorithms for engineering design optimization*, Appl. Soft Comput. 10 (2010), pp. 1188–1199.
[37] G. Liuzzi, S. Lucidi, and V. Piccialli, *A partition-based global optimization algorithm*, J. Global Optim. 48 (2010), pp. 113–128.
[38] J.M. Martínez, *Algorithms for solving nonlinear systems of equations*, in *Continuous Optimization: The State of Art*, E. Spedicato, ed., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 81–108.
[39] J.M. Martínez, *Practical Quasi-Newton methods for solving nonlinear systems*, J. Comput. Appl. Math. 124 (2000), pp. 97–122.
[40] K. Meintjes and A.P. Morgan, *Chemical equilibrium systems as numerical test problems*, ACM Trans. Math. Software 16(2) (1990), pp. 143–151.
[41] H. Modares, A. Alfi, and M.-B.N. Sistani, *Parameter estimation of bilinear systems on an adaptive particle swarm optimization*, Eng. Appl. Artif. Intell. 23 (2010), pp. 1105–1111.
[42] J.J. Moré, B.S. Garbow, and K.E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software 7(1) (1981), pp. 17–41.
[43] J.A. Nelder and R. Mead, *A simplex method for function minimization*, Computing Journal 7 (1965), pp. 308–313.
[44] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer-Verlag, Berlin, Heidelberg, 1999.
[45] U. Nowak and L. Weimann, *A family of Newton codes for systems of highly nonlinear equations*, Tech. Rep. Tr-91-10, K.-Z.-Z. Inf. Berlin, 1991.
[46] P.M. Pardalos, H.E. Romeijn, and H. Tuy, *Recent developments and trends in global optimization*, J. Comp. Appl. Math. 124 (2000), pp. 209–228.
[47] A.M.A.C. Rocha and E.M.G.P. Fernandes, *Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems*, Int. J. Comp. Math. 86 (2009), pp. 1932–1946.
[48] A.M.A.C. Rocha, T.F.M.C. Martins, and E.M.G.P. Fernandes, *An augmented Lagrangian fish swarm based method for global optimization*, J. Comp. Appl. Math. 235 (2011), pp. 4611–4620.
[49] M.-J. Tahk, H.-W. Woo, and M.-S. Park, *A hybrid optimization method of evolutionary and gradient search*, Eng. Optim. 39 (2007), pp. 87–104.
[50] C. Voglis and I.E. Lagaris, *Towards "Ideal Multistart". A stochastic approach for locating the minima of a continuous function inside a bounded domain*, Appl. Math. Comput. 213 (2009), pp. 1404–1415.
[51] C. Wang, R. Luo, K. Wu, and B. Han, *A new filled function method for an unconstrained nonlinear equation*, J. Comput. Appl. Math. 235 (2011), pp. 1689–1699.
[52] K. Yamamura, H. Kawata, and A. Tokue, *Interval solution of nonlinear equations using linear programming*, BIT – Numer. Math. 38(1) (1998), pp. 186–199.
[53] A. Zhigljavsky and A. Zilinskas, *Stochastic Global Optimization*, Optimization and Its Applications, Springer-Verlag, Berlin, Heidelberg, 2007.
[54] W. Zhu and M.M. Ali, *Solving nonlinearly constrained global optimization problem via an auxiliary function method*, J. Comput. Appl. Math. 230 (2009), pp. 491–503.