Simplified Tabu Search with Random-Based Searches for Bound Constrained Global Optimization

Ana Maria A. C. Rocha[1(✉)], M. Fernanda P. Costa[2], and Edite M. G. P. Fernandes[1]

[1] ALGORITMI Center, University of Minho, 4710-057 Braga, Portugal
{arocha,emgpf}@dps.uminho.pt
[2] Centre of Mathematics, University of Minho, 4710-057 Braga, Portugal
mfc@math.uminho.pt

Abstract. This paper proposes a simplified version of the tabu search algorithm that solely uses randomly generated direction vectors in the exploration and intensification search procedures, in order to define a set of trial points while searching in the neighborhood of a given point. In the diversification procedure, points that are inside any already visited region with a relative small visited frequency may be accepted, apart from those that are outside the visited regions.

The produced numerical results show the robustness of the proposed method. Its efficiency when compared to other known metaheuristics available in the literature is encouraging.

Keywords: Global optimization · Tabu search · Random searches

1 Introduction

This paper aims to present a simplified tabu search algorithm, in line of the Directed Tabu Search (DTS) [1], that uses random exploration and intensification local search procedures. This means that the exploration in the neighborhood of a current solution, as well as the final local intensification search are based only on randomly generated vectors. This proposal aims to contribute to the research area of nonlinear bound constrained global optimization (BCGO). The problem is formulated as follows:

$$\begin{aligned} &\min \ f(x) \\ &\text{subject to } \ x \in \Omega, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function and $\Omega = \{x \in \mathbb{R}^n : -\infty < l_i \leq x_i \leq u_i < \infty, i = 1, \ldots, n\}$ is a bounded feasible region. We do not assume that the objective function f is differentiable and convex and many local minima may exist in Ω. The optimal set X^* of the problem (1) is assumed to be nonempty

O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12251, pp. 606–619, 2020.
https://doi.org/10.1007/978-3-030-58808-3_44

and bounded. The global minimizer is represented by x^* and the global optimal value by f^*.

Solution methods for global optimization can be classified into two classes: the exact methods and the approximate methods [2,3]. Exact methods for global optimization are guaranteed to find an optimal solution within a finite run time that is problem-dependent. Thus, they solve a problem with a required accuracy in a finite number of steps. With approximate methods, it is not possible to guarantee an optimal solution. However, very good solutions can be obtained within a reasonable run time. Most of the approximate methods are stochastic. These methods differ from the deterministic ones in that they rely on random variables and random searches that use selected heuristics to search for good solutions. Metaheuristics are heuristics that can be applied to any general and complex optimization problem. Their search capabilities are not problem-dependent [4].

The Tabu Search (TS) is a trajectory-based metaheuristic that was primary developed in 1986 for combinatorial problems [5], and later on extended to solve continuous optimization problems, e.g., [6–8]. TS for continuous optimization guides the search out of local optima and continues the exploration of new regions for the global optimum. Important aspects of the TS are the definition of a neighborhood of the current solution and the management of a tabu list (a list of the already computed solutions). Along the iterative process, TS maintains a list of the most recent movements, which is used to avoid subsequent movements that may lead to solutions that have already been visited.

To overcome the slow convergence of TS, a local search strategy has been applied to the promising areas found by the TS algorithm, e.g., the simplex method of Nelder-Mead [9] or the Hooke-and-Jeeves [10]. Details can be seen in [1,11–13]. In general, this type of hybridization occurs in the final stage of the iterative process when the current solution is in the vicinity of the global optimum. The DTS method proposed in [1] bases its searches for the global optimum on three main procedures:

- the "neighborhood search procedure" - that aims to generate trial points - is based on local search strategies, like the Nelder-Mead [9] or the adaptive pattern search strategy [14], it generates a tabu list (TL) of already visited solutions and defines tabu regions (TRg) to prevent anti-cycling;
- the "diversification search procedure" goals the random generation of new points outside the already visited regions (VRg) and it also stores a list of the VRg, with their visited frequencies (VRf);
- the "intensification search procedure" aims to refine the best solution found so far and is applied at the final stage of the process.

Hybrid strategies that incorporate other popular metaheuristics into the TS have been proposed so far, but they are mostly applied in solving combinatorial problems. The diversity of TS applications - mostly in the combinatorial optimization area - are illustrated in [15]. The work also gives a detailed description of some of the TS structures and attributes, e.g., tabu list, neighborhood, move, threshold value, short-term memory, recency-based memory, frequency-

based memory. For continuous problems, the paper [16] presents a hybrid by the integration of the Scatter Search into the TS.

1.1 The Contribution

The herein presented simplification of the TS algorithm, to solve bound constrained global optimization problems, is denoted by "Simplified Tabu Search (with random-based searches)" (S-TS), and focuses on the random generation of direction vectors to search in the neighborhood of a central point x. These random searches of trial points are implemented in the "exploration search procedure" and in the 'intensification search procedure":

- in the "exploration search procedure", n trial points around the central point x are computed, where each point is obtained by adding a randomly generated number to each component of x;
- In the "intensification search procedure", n normalized direction vectors are randomly generated, one different vector for each trial point computed around the central x.

Furthermore, during the "diversification procedure", our proposal allows the acceptance of a randomly generated point in $[l, u]$ that is inside an already VRg as long as its correspondent VRf is relatively small, apart from the acceptance of a point that is outside all the already VRg.

1.2 Notation

For quick reference, a summary of the notation used in this paper follows:

- $\mathcal{L}$: set of indices of points in the TL,
- $|\mathcal{L}|$: cardinal of $\mathcal{L}$,
- L_{edge}: basic edge length,
- R_{TRg}: radius of the TRg (fixed, for all TRg),
- $NTL_{\max}$: maximum number of points in the TL,
- N_{VRg}: number of VRg,
- VR^i: the center point of the VRg i, where $i = 1, \ldots, N_{VRg}$,
- VRf^i: the frequency of the VRg i (number of generated points that are inside the VRg i, including the center),
- R_{VRg}: radius of the VRg (fixed, for all VRg),
- $Perc$: percentage that allows that a point inside at least one of the already VRg is accepted as a new diversification point,
- $NnoI_{\max}$: allowed maximum number of consecutive iterations where an improved point is not found,
- δ: step size of the search along each random direction,
- ϵ: tolerance to analyze vicinity of f^* (the best known global minimum),
- $rand$: random number uniformly distributed in $[0, 1]$.

1.3 Organization

This paper is organized as follows. In Sect. 2, the most important parts of the new algorithm are presented, namely the new "exploration search procedures", "diversification procedure", "neighborhood search procedure" and the "intensification search procedure". The comparative experiments are shown in Sect. 3 and the paper is concluded in Sect. 4.

2 Simplified Tabu Search

This section describes the S-TS method. We remark that the bound constraints of the problem are always satisfied by projecting any component i of a computed trial point that lies outside the interval $[l_i, u_i]$ to the bound l_i or u_i or, alternatively, randomly projecting into the interval $[l_i, u_i]$ (as it is shown in (2)).

Initially, this variant of the TS algorithm randomly generates a point $x \in [l, u]$

$$x_j = l_j + rand\,(u_j - l_j)\,,\ j = 1, \ldots, n, \tag{2}$$

and the TL is initialized with x. Using x, the "exploration search procedure" explores randomly in the neighborhood of x to find a set of n trial points, denoted by $s^i, i = 1, \ldots, n$, using randomly generated numbers in $[-1, 1]$ and a positive step size $\delta \in \mathbb{R}^+$,

$$s^i_j = x_j + \delta\, rand^{[-1,1]}_j\,,\ j = 1, \ldots, n \tag{3}$$

for $i = 1, \ldots, n$, where $rand^{[-1,1]}_j = -1 + 2\, rand$ represents a random number in $[-1, 1]$ generated from a uniform distribution. (See Algorithm 1 below.) The best of the trial solutions (the one with least objective function value, $f_{\min}$) is saved at $x_{\min}$. If this point has not improved relative to x, the step size δ is reduced and a new set of points $s^i, i = 1, \ldots, n$ are randomly generated centered at x (in line of (3)). If there is an improvement relative to x, all the points $s^i, i = 1, \ldots, n$ are shifted by the same amount. The amount is $x_{\min} - 2x$ if the number of improved iterations so far exceeds n, otherwise is $x_{\min} - x$. This exploration algorithm terminates when the number of iterations exceeds $It_{\max}$, δ falls under $1E-06$, or the difference between the worst and the best of the trial points is considered small. See Algorithm 2 below.

Algorithm 1. Random exploration trial points algorithm

Require: n, x, l, u, δ;
1: **for** $i = 1, \ldots, n$ **do**
2: **for** $j = 1, \ldots, n$ **do**
3: Compute $s^i_j = x_j + \delta\, rand^{[-1,1]}_j$;
4: **end for**
5: Project s^i componentwise into $[l, u]$;
6: **end for**

Algorithm 2. Exploration search algorithm

Require: n, x, f, l, u, δ, ϵ, $It_{\max}$;
1: Set $Nf_{exp} = 0$, $It = 1$, $NImpr = 0$;
2: Based on x and δ, generate $s^i, i = 1, \ldots, n$ using Algorithm 1;
3: **repeat**
4: Compute $f(s^i)$, $i = 1, \ldots, n$, identify $x_{\min}$, $f_{\min}$ and $x_{\max}$; Update Nf_{exp};
5: **if** $f_{\min} < f$ **then**
6: Set $NImpr = NImpr + 1$;
7: **if** $NImpr > n$ **then**
8: Set $d_{\min} = x_{\min} - 2x$;
9: **else**
10: Set $d_{\min} = x_{\min} - x$;
11: **end if**
12: Set $x = x_{\min}$, $f = f_{\min}$;
13: Compute $s^i = s^i + d_{\min}$, project s^i componentwise into $[l, u]$, $i = 1, \ldots, n$;
14: **else**
15: Set $\delta = 0.5\delta$;
16: Based on x and δ, generate s^i, $i = 1, \ldots, n$ using Algorithm 1;
17: Set $x_{\min} = x$, $f_{\min} = f$;
18: **end if**
19: Set $It = It + 1$;
20: **until** $\|x_{\max} - x_{\min}\| \leq 10^2\epsilon$ or $It > It_{\max}$ or $\delta \leq 1E-06$

The formal description of the implemented S-TS algorithm for solving the BCGO problem (1), based on random searches to define a set of trial points in the neighborhood of a specified point is presented in Algorithm 3.

This S-TS algorithm aims to find a global optimal solution of a BCGO problem, $f(x_{better})$, within an error of $100\,\epsilon\%$ relative to f^*, i.e., the algorithm stops if the following condition holds:

$$|f(x_{better}) - f^*| \leq \epsilon \max\{1, |f^*|\}, \tag{4}$$

where ϵ is a small positive tolerance. However, if the above condition is not satisfied and the number of function evaluations required by the algorithm to reach the current approximation, Nfe, exceeds the target $Nfe_{\max}$, the algorithm also stops.

2.1 Diversification Procedure

The main loop of the S-TS algorithm (from line 9 to 23 in Algorithm 3) invokes a "diversification procedure" (see line 12 of the algorithm). Here, a randomly generated point y that is not inside any of the already VRg is accepted. Each VRg is defined by its center VR^i, the radius R_{VRg} and its frequency VRf^i, $i = 1, \ldots, N_{VRg}$, being N_{VRg} the number of visited regions defined so far. In this case, a new VRg is created centered at $VR^j = y$, VRf^j is set to 1 and N_{VRg} is updated. However, a point y that is inside any of the VRg may also be accepted. Let k be the index of the VRg that has its center VR^k closest to y,

Algorithm 3. Simplified Tabu Search algorithm

Require: n, l, u, L_{edge}, ϵ, f^*, $Nfe_{\max}$, ϵ_{TS}, Nfe_{TS}, $NnoI_{\max}$;
Set $NnoI = 0$, $\mathcal{L} = \emptyset$,
Randomly generate $x \in [l, u]$ according to (2), compute $f = f(x)$;
Set $x_{better} = x$, $f_{better} = f$, $Nfe = 1$, $\delta = 2L_{edge}$, $success = No$;
Based on x, use Algorithm 2 to provide $x_{\min}$ (after Nf_{exp} function evaluations);
Set $Nfe = Nfe + Nf_{exp}$;
Set $x_{better} = x_{\min}$, $f_{better} = f(x_{\min})$;
Initialize TL, $z^1 = x$ ($\mathcal{L} = \{1\}$), set $VR^1 = z^1$, $VRf^1 = 1$, $N_{VRg} = 1$;
Set $x_{old} = x = x_{\min}$, $f_{old} = f = f(x_{\min})$;
repeat
  Using x and Algorithm 5, provide $x_{\min}$ (after Nf_{nei} function evaluations);
  Set $Nfe = Nfe + Nf_{nei}$;
  Generate y using Algorithm 4;
  Set $x = y$, compute $f = f(x)$, $Nfe = Nfe + 1$; Identify z^W in TL;
  Update TL: set $z^W = x$, $f^W = f$;
  if $f(x_{\min}) < f_{old}$ **then**
    Set $NnoI = 0$, $x_{old} = x_{\min}$, $f_{old} = f(x_{\min})$;
  else
    Set $NnoI = NnoI + 1$;
  end if
  if $f(x_{\min}) < f_{better}$ **then**
    Set $x_{better} = x_{\min}$, $f_{better} = f(x_{\min})$;
  end if
until $NnoI \geq NnoI_{\max}$ or $Nfe \geq Nfe_{TS}$ or $|f_{better} - f^*| \leq \epsilon_{TS} \max\{1, |f^*|\}$
Set $\delta = 2L_{edge}$, $Nf_{\max}^{int} = Nfe_{\max} - Nfe$;
Using x_{better} and Algorithm 6, provide $x_{\min}$ (after Nf_{int} function evaluations);
Set $Nfe = Nfe + Nf_{int}$;
Set $x_{better} = x_{\min}$, $f_{better} = f(x_{\min})$;
if $|f_{better} - f^*| \leq \epsilon \max\{1, |f^*|\}$ **then**
  Set $success = Yes$;
end if

among all VRg that contain y. If its frequency VRf^k is small (relative to all the frequencies of the other VRg), i.e., if

$$\frac{VRf^k}{\sum_{i=1}^{N_{VRg}} VRf^i} < Perc,$$

the point is accepted, where $Perc$ is the percentage for the acceptance of a point y inside an already VRg. In this case the corresponding frequency is updated. The details concerning the acceptance issue and the updating of VRg are shown in Algorithm 4.

The main loop terminates when the number of consecutive iterations with non-improved points exceeds $NnoI_{\max}$, the number of function evaluations exceeds a target value, Nfe_{TS}, or the best function value in the TL has an error of 100 ϵ_{TS}% relative to f^*.

Algorithm 4. Diversification search algorithm

Require: $VR^i, VRf^i, i = 1, \ldots, N_{VRg}$, R_{VRg}, $Perc$;
1: Set $accept = 0$;
2: **while** $accept = 0$ **do**
3: Randomly generate $y \in [l, u]$ according to (2);
4: Compute $d_{min} = \min_{i=1,\ldots,N_{VRg}}\{\|y - VR^i\|/R_{VRg}\}$; Set $k = \arg\min_i\{\|y - VR^i\|/R_{VRg}\}$;
5: **if** $d_{min} > 1$ **then**
6: Set $accept = 1$, $N_{VRg} = N_{VRg} + 1$, $VR^j = y$ with $VRf^j = 1$ where $j = N_{VRg}$;
7: **else**
8: **if** $(VRf^k / \sum_i VRf^i) < Perc$ **then**
9: Set $accept = 1$, $VRf^k = VRf^k + 1$;
10: **end if**
11: **end if**
12: **end while**

2.2 Neighborhood Search Procedure

Based on a starting point x, each iteration of the main loop of the S-TS algorithm also tries to compute a point better than the x, by invoking the "neighborhood search procedure" (see line 10 of the Algorithm 3). Firstly, the randomly generated set of trial points should be preferentially far away from the other points already in the TL. This is accomplished by selecting a specific value for the step size δ. The points in the TL, $z^i, i \in \mathcal{L}$, are the centers of the TRg, and the radius, R_{TRg}, is a fraction of L_{edge}. This means that if x is inside any of the TRg, δ is chosen to be a value greater than the maximum distance from x to the centers of those TRg that contain x, i.e., $\|x - z^i\| \leq R_{TRg}$ (for those i) holds. Otherwise, δ is greater than L_{edge}. The trial points around x are generated by the "exploration search procedure" that provides $x_{\min}$.

The point x is then added to the TL if the number of points in the TL is smaller than the threshold value $NTL_{\max}$. Otherwise, our proposal here is to replace the worst point in the TL, z^W, by x. Parallel to that, the frequencies of the VRg that contain x are updated or, a new VRg is defined centered at x with frequency set to 1, if none of the VRg contains x.

The point $x_{\min}$ is compared to x. If an improvement has been obtained (in terms of f), the counter for consecutive iterations with no improvement is set to 0, otherwise, the counter is updated. This iterative procedure terminates when the number of iterations exceeds a target, $It^N_{\max}$, or the number of consecutive iterations with no improvement computed points reaches a threshold value $NnoI_{\max}$. For the output of the procedure, $x_{\min}$ is identified as the best point (with least objective function value) among the points in TL. See the details in Algorithm 5.

Algorithm 5. Neighborhood search procedure

Require: n, x, f, L_{edge}, $z^i, i \in \mathcal{L}$ (points in the TL), $VR^i, VRf^i, i = 1, \ldots, N_{VRg}$, $NnoI_{\max}$, $It^N_{\max}$, $NTL_{\max}$, R_{TRg}, R_{VRg};

1: Set $NnoI = 0$, $Nf_{nei} = 0$, $It = 0$;
2: **repeat**
3: Set $It = It + 1$, $\mathcal{L}_< = \emptyset$, $flag_{VRg} = 0$;
4: **for all** $i \in \mathcal{L}$ such that $\|x - z^i\| \leq R_{TRg}$ **do**
5: Set $\mathcal{L}_< = \mathcal{L}_< \cup \{i\}$;
6: **end for**
7: **if** $\mathcal{L}_< \neq \emptyset$ **then**
8: Compute $d_{\max} = \max\{\|x - z^i\|\}$ for all $i \in \mathcal{L}_<$;
9: $\delta = (1 + rand)d_{\max}$;
10: **else**
11: $\delta = (1 + rand)L_{edge}$;
12: **end if**
13: Using x and Algorithm 2, provide $x_{\min}$ (after Nf_{exp} function evaluations);
14: Set $Nf_{nei} = Nf_{nei} + Nf_{exp}$;
15: **if** $|\mathcal{L}| < NTL_{\max}$ **then**
16: Update TL: set $|\mathcal{L}| = |\mathcal{L}| + 1$, $z^i = x$, $f^i = f$ where $i = |\mathcal{L}|$;
17: **else**
18: Update TL: set $z^W = x$, $f^W = f$;
19: **end if**
20: **for all** i such that $\|x - VR^i\| < R_{VRg}$ **do**
21: Set $VRf^i = VRf^i + 1$; Set $flag_{VRg} = 1$;
22: **end for**
23: **if** $flag_{VRg} = 0$ **then**
24: Set $N_{VRg} = N_{VRg} + 1$, $VR^j = x$ and $VRf^j = 1$ where $j = N_{VRg}$;
25: **end if**
26: **if** $f(x_{\min}) < f$ **then**
27: Set $NnoI = 0$;
28: **else**
29: Set $NnoI = NnoI + 1$;
30: **end if**
31: Set $x = x_{\min}$, $f = f(x_{\min})$;
32: **until** $NnoI \geq NnoI_{\max}$ or $It \geq It^N_{\max}$
33: Identify $x_{\min}$, $f_{\min}$ in TL;

2.3 Intensification Search Procedure

Finally, the "intensification search procedure" is used to intensify the search around the best point found from the main loop of S-TS algorithm (see line 25 in Algorithm 3). It is simple to implement and it does not require any derivative information. Details of this procedure are shown in the Algorithm 6.

The search begins with a central point x (which on entry is the best point found so far, x_{better}) and a set of n trial approximations

$$s^i = x + \delta\, v^i, \tag{5}$$

where $v^i \in \mathbb{R}^n$ is a normalized vector with random components in $[-1, 1]$, for each $i = 1, \ldots, n$.

This procedure follows a strategy similar to the "exploration search procedure" although with important differences. For each point s^i a random direction vector v^i is generated and the reduction of the step size (when the best of the trial points has not improved over the central point x) is more moderate and δ is not allowed to fall below $1E-06$. We remark that when the best of the trial points improves relative to x, the algorithm resets δ to a fraction of the value on entry only if there was no improvement in the previous iteration. Furthermore, the termination of the algorithm is activated only when the best of the n trial approximations, denoted by $x_{\min}$, satisfies the stopping conditions as shown in (4), or when the number of function evaluations required by the algorithm exceeds the target $Nf_{\max}^{int} = Nfe_{\max} - Nfe$ (the remaining function evaluations until the maximum $Nfe_{\max}$ is attained, where Nfe is the function evaluations required until "intensification search procedure" is invoked).

Algorithm 6. Intensification search algorithm

Require: n, x, f, l, u, δ, $Nf_{\max}^{int}$, ϵ, f^*;
Set $Nf_{int} = 0$, $\delta_0 = \delta$, $flag_{noMove} = 0$;
Based on x and δ, generate s^i, $i = 1, \ldots, n$ using Algorithm 7;
repeat
  Compute $f(s^i)$, $i = 1, \ldots, n$; Update Nf_{int}; Identify $x_{\min}$, $f_{\min}$;
  if $f_{\min} < f$ **then**
    Set $d_{\min} = x_{\min} - x$, $x = x_{\min}$, $f = f_{\min}$;
    Compute $s^i = s^i + d_{\min}$, project s^i componentwise into $[l, u]$, $i = 1, \ldots, n$;
    if $flag_{noMove} = 1$ **then**
      Set $\delta = 0.95\delta_0$, $\delta_0 = \delta$, $flag_{noMove} = 0$;
    end if
  else
    Set $\delta = \max\{0.75\delta, 1E-06\}$, $flag_{noMove} = 1$;
    Based on x and δ, generate s^i, $i = 1, \ldots, n$ using Algorithm 7;
    Set $x_{\min} = x$, $f_{\min} = f$;
  end if
until $|f_{\min} - f^*| \leq \epsilon \max\{1, |f^*|\}$ or $Nf_{int} \geq Nf_{\max}^{int}$

Algorithm 7. Random intensification trial points algorithm

Require: n, x, l, u, δ;
for $i = 1, \ldots, n$ **do**
  Generate $v^i \in \mathbb{R}^n$ with random components in $[-1, 1]$ and $\|v^i\| = 1$;
  Compute $s^i = x + \delta\, v^i$; Project s^i componentwise into $[l, u]$;
end for

3 Numerical Results

To analyze the performance of our Algorithm 3, we use two sets of benchmark problems. The first set contains 9 problems and is known as Jones set: Branin (BR) with $n = 2$, Camel Six-Hump (C6) with $n = 2$, Goldstein & Price (GP) with $n = 2$, Hartman 3 (H3) with $n = 3$, Hartman 6 (H6) with $n = 6$, Shekel 5 (S5) with $n = 4$, Shekel 7 (S7) with $n = 4$, Shekel 10 (S10) with $n = 4$, Schubert (SHU) with $n = 2$ (see the full description in [1]).

The second set contains sixteen problems: Booth (BO) with $n = 2$, Branin (BR) with $n = 2$, Camel Six-Hump (C6) with $n = 2$, Dekkers & Aarts (DA) with $n = 2$, Goldstein & Price (GP) with $n = 2$, Hosaki (HSK) with $n = 2$ Matyas (MT) with $n = 2$, McCormick (MC) with $n = 2$, Modified Himmelblau (MHB) with $n = 2$, Neumaier2 (NF2) with $n = 4$, Powell Quadratic (PWQ) with $n = 4$, Rastrigin with $n = 2$, $n = 5$, $n = 10$ (RG-2, RG-5, RG-10), Rosenbrock (RB) with $n = 2$ and Wood (WF) with $n = 4$, see the full description in [17].

The MATLAB® (MATLAB is a registered trademark of the MathWorks, Inc.) programming language is used to code the algorithm and the tested problems.

The values for the parameters are set as follows: $L_{edge} = 0.1 \min_{i=1,\ldots,n}(u_i - l_i)$, $R_{VRg} = 2L_{edge}$, $R_{TRg} = 0.2L_{edge}$, $NnoI_{\max} = 2n$. Unless otherwise stated, in the stopping condition (4), the tolerance is set to $\epsilon = 1E-04$, and $Nfe_{\max} = 50000$. We also set $\epsilon_{TS} = 1E+02\,\epsilon$, $Nfe_{TS} = 0.2\,Nfe_{\max}$ and $Perc = 0.25$. In the "neighborhood search procedure", we set the number of iterations with consecutive no improvement points $NnoI_{\max} = 2n$, and $It^N_{\max} = 3n$, $NTL_{\max} = 5n$. The parameter for the "exploration search procedure" is $It_{\max} = 2n$.

Because there are elements of randomness in the algorithm, each problem was solved several times (100 and 30 depending on the set of problems) by the algorithm using different starting seed for the pseudo-random number generator.

The subsequent tables report the average number of function evaluations required to achieve the stopping condition (4). To test the robustness of the algorithm, the rate of success, i.e., the percentage of runs in which the algorithm obtains a solution satisfying (4), is also shown (inside parentheses). The average number of the objective function evaluations is evaluated only in relation to the successful runs.

With the Tables 1 and 2, we aim to compare our results to those of the Enhanced Continuous TS (ECTS) in [7], the DTS method [1], the Continuous greedy randomized adaptive search procedure (C-GRASP) [18], the Hybrid Scatter TS (H-STS) [16], a simulated annealing hybridized with a heuristic pattern search (SAHPS) [14], a mutation-based artificial fish swarm algorithm (m-AFS) [19], and an improved TS with the Nelder-Mead local search (iTS-NM) available in [12]. The results obtained from our algorithm emanate from the two implemented scenarios when a component of a point lies outside $[l_j, u_j]$:

Case 1 - projecting randomly into $[l_j, u_j]$;
Case 2 - projecting to the bound l_j or u_j.

A comparison with the results of the opposition-based differential evolution (ODE) [20], where the error tolerance in (4) is reduced to $\epsilon = 1E-08$ and $Nfe_{\max} = 1E+06$ (only for the Case 1 scenario) is also shown. The results of the listed algorithms are taken from the original papers.

Table 1. Average number of function evaluations and rate of success over 100 runs

	Algorithm 3†	ECTS †,a	DTS †,b	C-GRASP †,c	H-STS †,d
	Case 1				
BR	242 (100)	245 (100)	212 (100)	10090 (100)	1248 (100)
C6	199 (100)	–	–	–	–
GP	338 (100)	231 (100)	230 (100)	53 (100)	809 (100)
H3	708 (100)	548 (100)	438 (100)	1719 (100)	298 (100)
H6	1015 (100)	1520 (100)	1787 (83)	29894 (100)	1263 (100)
S5	1445 (99)	–	819 (75)	9274 (100)	9524 (100)
S7	1586 (97)	–	812 (65)	11766 (100)	3818 (100)
S10	1742 (96)	–	828 (52)	17612 (100)	3917 (100)
SHU	456 (100)	370 (100)	274 (92)	18608 (100)	1245 (100)

† Results based on $\epsilon = 1E-04$ in (4) and $Nfe_{\max} = 50000$.
[a] Results reported in [7]; [b] results reported in [1];
[c] Results reported in [18]; [d] results reported in [16] (25 runs).
"–" Information not available.

First, when the results of Case 1 are compared to those of Case 2, it is possible to conclude that Case 1 is slightly more robust and efficient in general. The first comparison is with the results of DTS. Our S-TS algorithm wins clearly as far as robustness is concerned. When we compare our results to those of C-GRASP and H-STS - algorithms with similar robustness level - we conclude that Algorithm 3 wins on efficiency. From Table 2, it is possible to conclude that Algorithm 3 wins on robustness - although requires more function evaluations - when compared to SAHPS and iTS-NM. The comparison with ODE is also favourable to our algorithm.

Table 3 contains the results obtained with the second set of problems and aims to compare our Algorithm 3 to the stochastic coordinate descent method (St-CS) available in [21]. A comparison with the results of the ODE [20], where the error tolerance is reduced to $\epsilon = 1E-08$ and $Nfe_{\max} = 1E+06$ (only for the Case 1 scenario) is also shown. We note that Case 2 is now slightly more efficient than Case 1. The comparison with St-CS is favourable to the Algorithm 3 as far as robustness and efficiency are concerned. From the comparison with ODE, we may conclude that Algorithm 3 was not able to converge to the solution, with the required accuracy, on 4 of the 16 tested problems, but produced very good results (in terms of robustness and efficiency) to the other problems.

Table 2. More comparisons of function evaluations and rate of success over 100 runs

	Algorithm 3 †	SAHPS †,a	m-AFS †,b	iTS-NM †,c	Algorithm 3 §	ODE §,d
	Case 2				Case 1	
BR	255 (100)	318 (100)	475 (–)	178 (100)	656 (100)	2804 (100)
C6	192 (100)	–	247 (–)	–	n.a. (0)	2366 (100)
GP	354 (100)	311 (100)	417 (–)	165 (100)	871 (100)	2370 (100)
H3	751 (100)	517 (95)	1891 (–)	212 (100)	1573 (100)	1796 (100)
H6	1143 (100)	997 (72)	2580 (–)	880 (66)	3276 (100)	n.a. (0)
S5	1940 (84)	1073 (48)	1183 (–)	777 (75)	3067 (100)	n.a. (0)
S7	1686 (94)	1059 (57)	1103 (–)	751 (89)	2594 (97)	n.a. (0)
S10	1778 (82)	1031 (48)	1586 (–)	751 (89)	2841 (93)	2316 (100)
SHU	504 (100)	450 (86)	523 (–)	402 (100)	908 (100)	–

† Results based on $\epsilon = 1E-04$ in (4) and $Nfe_{\max} = 50000$;
§ results based on $\epsilon = 1E-08$ in (4) and $Nfe_{\max} = 1E+06$.
[a] Results reported in [14]; [b] results reported in [19] (30 runs);
[c] results reported in [12]; [d] results reported in [20] (50 runs).
"–" Information not available; "n.a." not applicable.

Table 3. Average number of function evaluations and rate of success over 30 runs

	Algorithm 3 †		St-CS †,a	Algorithm 3 §	ODE §,b
	Case 1	Case 2		Case 1	
BO	201 (100)	244 (100)	1555 (100)	686 (100)	–
BR	244 (100)	232 (100)	239 (100)	650 (100)	2804 (100)
C6	175 (100)	224 (100)	512 (100)	n.a. (0)	2366 (100)
DA	518 (100)	485 (100)	1020 (100)	n.a. (0)	1116 (100)
GP	326 (100)	353 (100)	1564 (100)	790 (100)	2370 (100)
HSK	134 (100)	175 (100)	110 (100)	n.a. (0)	1654 (100)
MT	128 (100)	120 (100)	2159 (100)	436 (100)	1782 (100)
MC	134 (100)	128 (100)	172 (100)	n.a. (0)	1528 (100)
MHB	450 (100)	449 (93)	1450 (100)	937 (83)	–
NF2	10236 (33)	8962 (40)	n.a. (0)	13497 (3)	364300 (8)
PWQ	1230 (100)	1130 (100)	n.a. (0)	52084 (100)	3998 (100)
RG-2	544 (97)	515 (87)	2074 (100)	1023 (93)	–
RG-5	1532 (100)	1356 (100)	6981 (100)	4080 (100)	–
RG-10	4241 (100)	4107 (100)	20202 (100)	12068 (100)	170200 (24)
RB	905 (100)	939 (100)	n.a. (0)	2324 (100)	–
WF	13034 (100)	11856 (100)	n.a. (0)	23305 (100)	54136 (84)

† Results based on $\epsilon = 1E-04$ in (4) and $Nfe_{\max} = 50000$;
§ results based on $\epsilon = 1E-08$ in (4) and $Nfe_{\max} = 1E+06$.
[a] Results reported in [21]; [b] results reported in [20] (50 runs).
"–" Information not available; "n.a." not applicable.

4 Conclusions

A simplified version of the TS metaheuristic, denoted by S-TS, is presented. The simplifications are concerned mainly with the diversification, exploration and intensification procedures. When searching in the neighborhood of a given central point, both "exploration search procedure" and "intensification search procedure" rely solely on randomly direction vectors. In the "diversification procedure", our S-TS algorithm also allows the acceptance of a randomly generated point that falls inside any already visited region, as long as its visited frequency is small. The new algorithm has been tested and compared to other well-known metaheuristics in the literature.

The testified robustness and efficiency when solving BCGO problems are encouraging and induce us to extend this simplified TS algorithm to solving general nonlinear constrained global optimization problems. The idea is to handle constraint violation and objective function values separately although giving priority to trial points that are feasible with respect to the general constraints.

Acknowledgments. The authors wish to thank two anonymous referees for their comments and suggestions to improve the paper.

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020, UIDB/00013/2020 and UIDP/00013/2020 of CMAT-UM.

References

1. Hedar, A.-R., Fukushima, M.: Tabu Search directed by direct search methods for nonlinear global optimization. Eur. J. Oper. Res. **170**, 329–349 (2006)
2. Hendrix, E.M.T., Boglárka, G.T.: Introduction to Nonlinear and Global Optimization. In: Optimization and its Applications, vol. 37. Springer, New York (2010). https://doi.org/10.1007/978-0-387-88670-1
3. Stork, J., Eiben, A.E., Bartz-Beielstein, T.: A new taxonomy of continuous global optimization algorithms, 27 August 2018. arXiv:1808.08818v1
4. Sörensen, K.: Metaheuristics - the metaphor exposed. Int. Trans. Oper. Res. **22**, 3–18 (2015)
5. Glover, F.W.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
6. Siarry, P., Berthau, G.: Fitting of tabu search to optimize functions of continuous variables. Int. J. Num. Meth. Eng. **40**(13), 2449–2457 (1997)
7. Chelouah, R., Siarry, P.: Tabu search applied to global optimization. Eur. J. Oper. Res. **123**, 256–270 (2000)
8. Franzè, F., Speciale, N.: A tabu-search-based algorithm for continuous multiminima problems. Int. J. Numer. Meth. Eng. **50**, 665–680 (2001)
9. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
10. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. J. ACM **8**, 212–229 (1961)

11. Chelouah, R., Siarry, P.: A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multiminima functions. Eur. J. Oper. Res. **161**, 636–654 (2005)
12. Mashinchi, M.H., Orgun, M.A., Pedrycz, W.: Hybrid optimization with improved tabu search. Appl. Soft. Comput. **11**, 1993–2006 (2011)
13. Ramadas, G.C.V., Fernandes, E.M.G.P.: Self-adaptive combination of global tabu search and local search for nonlinear equations. Int. J. Comput. Math. **89**(13–14), 1847–1864 (2012)
14. Hedar, A.-R., Fukushima, M.: Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. Optim. Method. Softw. **19**(3–4), 291–308 (2004)
15. Glover, F., Laguna, M., Martí, R.: Principles and strategies of tabu search. In: Gonzalez, T.F. (ed.) Handbook of Approximation Algorithms and Metaheuristics: Methologies and Traditional Applications, 2nd edn., vol. 1, pp. 361–375. Chapman and Hall, London (2018)
16. Duarte, A., Martí, R., Glover, F., Gortázar, F.: Hybrid scatter tabu search for unconstrained global optimization. Ann. Oper. Res. **183**(1), 95–123 (2011)
17. Ali, M.M., Khompatraporn, C., Zabinsky, Z.B.: A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J. Glob. Optim. **31**(4), 635–672 (2005). https://doi.org/10.1007/s10898-004-9972-2
18. Hirsch, M.J., Meneses, C.N., Pardalos, P.M., Resende, M.G.C.: Global optimization by continuous GRASP. Optim. Lett. **1**(2), 201–212 (2007). https://doi.org/10.1007/s11590-006-0021-6
19. Rocha, A.M.A.C., Costa, M.F.P., Fernandes, E.M.G.P.: Mutation-based artificial fish swarm algorithm for bound constrained global optimization. In: AIP Conference Proceedings, vol. 1389, pp. 751–754 (2011)
20. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-based differential evolution. IEEE Trans. Evol. Comput. **12**(1), 64–79 (2008)
21. Rocha, A.M.A.C., Costa, M.F.P., Fernandes, E.M.G.P.: A population-based stochastic coordinate descent method. In: Le Thi, H.A., Le, H.M., Pham Dinh, T. (eds.) WCGO 2019. AISC, vol. 991, pp. 16–25. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-21803-4_2