Available online at www.sciencedirect.com

European Journal of Operational Research 170 (2006) 329–349

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Continuous Optimization

Tabu Search directed by direct search methods for nonlinear global optimization [☆]

Abdel-Rahman Hedar, Masao Fukushima [*]

Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

Received 2 July 2003; accepted 13 May 2004
Available online 15 September 2004

Abstract

In recent years, there has been a great deal of interest in metaheuristics in the optimization community. Tabu Search (TS) represents a popular class of metaheuristics. However, compared with other metaheuristics like genetic algorithm and simulated annealing, contributions of TS that deals with continuous problems are still very limited. In this paper, we introduce a continuous TS called Directed Tabu Search (DTS) method. In the DTS method, direct-search-based strategies are used to direct a tabu search. These strategies are based on the well-known Nelder–Mead method and a new pattern search procedure called adaptive pattern search. Moreover, we introduce a new tabu list conception with anti-cycling rules called Tabu Regions and Semi-Tabu Regions. In addition, Diversification and Intensification Search schemes are employed. Numerical results show that the proposed method is promising and produces high quality solutions.

Keywords: Global optimization; Metaheuristics; Tabu Search; Nelder–Mead method; Adaptive pattern search

1. Introduction

Tabu Search (TS) is one of the recent metaheuristics originally developed for combinatorial optimization problems [10,11]. TS has shown an appropriate performance when applied to these problems [10,26]. However, contributions of TS to solving continuous optimization problems are still very limited compared with other metaheuristics like Simulated Annealing and Genetic Algorithms. In this paper, we introduce a TS

[☆] This research was supported in part by a Grant-in-Aid for Scientific Research from the Ministry of Education, Science, Sports and Culture of Japan.
[*] Corresponding author. Tel.: +81 75 753 5519; fax: +81 75 753 4756.
E-mail address: fuku@amp.i.kyoto-u.ac.jp (M. Fukushima).

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.05.033

approach that deals with continuous global optimization problems. Specifically, we consider the unconstrained nonlinear minimization problem

$$\min_{x \in R^n} f(x),$$

where f is a real-valued function defined on R^n. Recently, there has been a growing interest in solving this problem using metaheuristics [25,26]. However, metaheuristics may suffer from costly computations due to their slow convergence. So combining metaheuristics with local search methods is a practical remedy to overcome the slow convergence of metaheuristics. In particular, a number of attempts have been made to develop such combined search methods especially for Simulated Annealing and Genetic Algorithms [12–14]. In this paper, we present continuous versions of TS called Directed Tabu Search (DTS) by hybridizing TS with direct search methods. The role of direct search methods is to stabilize the search especially in the vicinity of a local minimum. Specifically, instead of using completely blind random search in generating neighborhood trial moves, appropriate direct search strategies are responsible to generate these neighborhood moves. Moreover, new implementations of TS elements are employed in the proposed method.

Since the first presentation of Glover [8,9], a lot of studies have emerged in the area of TS. The majority of these studies are related to combinatorial optimization problems and relatively few attempts have been made to deal with continuous optimization problems [1–5,7,15]. One of the earliest TS methods was presented by Hu [15] for constrained optimization problems. Cvijovic and Klinowski [4,5] extended and generalized the TS to functions with variables that may be continuous or, if discrete, need not take integer values. Battiti and Tecchiolli [2] introduced an interesting continuous TS method called the continuous reactive Tabu Search. Their method tries to locate the most promising boxes, and then starting points for the local search are generated within those boxes. Al-Sultan and Al-Fawzan [1] gave a hybrid method that combines TS with the local search method of Hooke and Jeeves.

Recently, intensive studies on continuous TS have been conducted in [3,7]. In [3], the authors introduced a new algorithm called Enhanced Continuous Tabu Search (ECTS), which aims to follow Glover's basic approach as closely as possible. In order to cover a wide domain of possible solutions, the ECTS algorithm first performs a Diversification Search to locate the most promising areas. When the most promising areas are located, the algorithm proceeds to an Intensification Search within one promising area of the solution space. In [7], the authors presented a novel TS algorithm that explores a grid of points with a distance dynamically adjusted. When it collapses to a local minimum, it continues the local search from that point while accepting some non-improving points to allow the exploration of new regions in the search space.

The DTS method proposed in this paper differs from the previous continuous TS methods in many aspects. In the DTS method, three search procedures are employed; Exploration, Diversification and Intensification. In the Exploration Search, a new local search procedure is introduced to generate trial moves, based on the well-known Nelder–Mead method [24] and the recently proposed pattern search method [14]. Moreover, novel concepts of TS memory elements called Tabu Regions (TRs), Semi-TRs and a multi-ranked Tabu List (TL) are introduced to provide anti-cycling rules. Another memory element called Visited Regions List (VRL) is also introduced as a tool for the Diversification Search to diversify the search to unvisited areas of the solution space. Finally, assuming that one of the best points obtained by the exploration and Diversification Searches is close to a global minimum, the Intensification Search is applied again at the final stage to refine the elite solutions visited so far. Actually, the proposed Diversification and Intensification Searches try to follow some known strategies from the high level TS with a long term memory. Moreover, the DTS can be classified as a multi-start method. The multi-start methods aim to construct powerful search procedures by guidance of global exploration and local searches; as surveys for multi-start methods the reader is referred to [21,22,27]. Multi-start methods have been successfully applied to both nonlinear global optimization and combinatorial problems, see [22] and references therein. Finally, the numerical results reported below show the promise of the proposed method especially in producing high quality solutions.

The rest of the paper is organized as follows. The next section gives a detailed description of the proposed TS memory elements. In Section 3, we introduce neighborhood and local search strategies used to generate the trial moves. The main DTS method is presented elaborately in Section 4. Section 5 discusses the implementation of the proposed method and reports comprehensive experimental results. The conclusion makes up Section 6.

2. TS memory elements

The concept of memory plays a major role in TS, especially when it is used in a kind of learning process as in high level TS with a long term memory. Using an effective memory conception in Intensification and Diversification schemes makes TS behave as an intelligent search technique [10]. The optimization search methods can be classified in two categories; point-to-point methods and population-based methods. TS belongs to the first category. Keeping the diversity is one of the main problems that face the point-to-point methods compared with the population-based methods. However, the long term memory in TS makes it competitive with the population-based methods in keeping the diversity. In TS with long term memory, the search can be restarted from new diverse solutions whenever the Diversification is needed, or can be intensified to improve the elite solutions whenever the Intensification is needed. These TS concepts of Diversification and Intensification have turned out to be effective in many combinatorial optimization problems, see [10,20] for example. In this section, we introduce some new conceptions and implementations of the TS memory elements to continuous optimization problems. First, we let the multi-ranked Tabu List (TL) be a set of some visited solutions. The points in the TL are ranked and saved according to their recency and their objective function values. Therefore, some positions in the TL are kept for the best visited solutions, which helps an Intensification scheme to refine the search from these best solutions at the final stage. Around each solution saved in the TL, two types of regions are specified in the search space. The first one is a Tabu Region (TR) in which no new trial point is allowed to be generated. The other is a Semi-Tabu Region (Semi-TR) that comprises a surrounding region around TR. The main role of the Semi-TRs is to generate neighboring trial points in a special way so that returning back to a visited TR is avoided when the trial solution lies inside a Semi-TR. Another memory element introduced in this section is the Visited Region List (VRL). The centers of the visited regions and the frequency of visiting these regions are saved in the VRL in order to direct a Diversification scheme to explore the space outside these visited regions.

2.1. Multi-ranked Tabu List (TL)

Some of the previously visited solutions are stored in the TL. Let $\mathrm{TL} = \{t_i\}_{i=1}^{L}$. The elements in TL are ranked in ascending order according to their recency using the rank indices $I_i^r,\ i = 1, \ldots, L$, i.e., if the most recent element in TL is t_k, then $I_k^r = 1$, while if the most ancient element is $t_{k'}$, then $I_{k'}^r = L$. Also, the elements in TL are ranked in ascending order according to their objective function values using another set of rank indices $I_i^{fv},\ i = 1, \ldots, L$, i.e., if the best element in TL is t_j, then $I_j^{fv} = 1$, and if the worst element is $t_{j'}$, then $I_{j'}^{fv} = L$. In the ordering, ties are broken arbitrarily. We consider the TL to be a fuzzy set and associate its elements $\{t_i\}_{i=1}^{L}$ the membership values:

$$m_i = \max\{m_i^r, m_i^{fv}\}, \quad i = 1, \ldots, L, \tag{1}$$

where $m_i^r, m_i^{fv} \in [0, 1]$ are the recency and the function-value ranked values, respectively, for element t_i and they are computed as follows:

- *The recency ranked value m^r.* We use a linear ranking procedure that gives the most recent element the maximum ranked value $\eta_{\max}$ and the most ancient element the minimum ranked value $\eta_{\min}$, where $0 \leqslant \eta_{\min} < \eta_{\max} \leqslant 1$. Specifically, the recency ranked value for each element of TL is given by

$$m_i^r = \eta_{\min} + (\eta_{\max} - \eta_{\min})\left(\frac{L - I_i^r}{L - 1}\right), \quad i = 1, \ldots, L. \tag{2}$$

- *The function-value ranked value m^{fv}.* To avoid reserving excessively many positions in the TL for the best elements and to give the recency some priority, this procedure ranks only $\overline{L}$ best elements so that the best element is given the ranked value $\mu_{\max}$, and the worst $L - \overline{L} + 1$ elements are given the ranked value $\mu_{\min}$, where $1 \leqslant \overline{L} \leqslant L$ and $0 \leqslant \mu_{\min} < \mu_{\max} \leqslant 1$. Specifically, the function-value ranked value for each element of TL is given by

$$m_i^{fv} = \begin{cases} \mu_{\min} + (\mu_{\max} - \mu_{\min})\left(\frac{\overline{L} - I_i^{fv}}{\overline{L} - 1}\right), & \text{if } I_i^{fv} = 1, \ldots, \overline{L}, \\ \mu_{\min}, & \text{if } I_i^{fv} = \overline{L} + 1, \ldots, L. \end{cases} \tag{3}$$

The Tabu Regions (TRs) are defined to be spheres with radius r_{TR} and their centers being the points of TL, where $r_{\text{TR}} > 0$. For each TR, we define Semi-TR to be the surrounding region around this TR with outer radius r_{STR} from its center, where $r_{\text{STR}} > r_{\text{TR}}$. If a trial solution lies in Semi-TRs, then a specific procedure is applied to create special neighborhood trial points to avoid returning back to a vicinity of a previously visited solution. We suggest the following procedure for this purpose.

Procedure 2.1

1. Let a trial point x lie in v Semi-TRs with centers $t_1, \ldots, t_v$. Compute the centroid $\bar{t}$ of the Semi-TRs' centers and the maximum distance $d_{\max}$ between x and these centers, i.e.,

$$\bar{t} = \frac{1}{v}\sum_{i=1}^{v} t_i, \tag{4}$$

$$d_{\max} = \max_{i=1,\ldots,v}\{\|x - t_i\|\}. \tag{5}$$

2. Construct neighborhood search directions that are parallel to the coordinate axes but point towards the direction $x - \bar{t}$, i.e., the neighborhood search directions are determined as $\text{sign}((x)_i - (\bar{t})_i)e_i$, $i = 1, \ldots, n$, where $e_i \in R^n$ is the ith unit vector in R^n. Neighborhood trial points are generated along these search directions with a suitable step size $\beta > 0$. In the case of $v > 1$, the step size β should be chosen greater than $d_{\max} + r_{\text{TR}}$ in order to avoid generating trial points inside a TR.

Fig. 1 illustrates how Procedure 2.1 works when a solution x lies in Semi-TRs in two dimensions. In this example, the solution x lies in two Semi-TRs with center t_1 and t_2. According to Procedure 2.1, the neighborhood search directions d_1 and d_2 are constructed to follow the vector $(x - \bar{t})$, where $\bar{t}$ is the centroid of the Semi-TRs' centers. It is noteworthy that the step size used to generate a trial point along search directions d_1 and d_2 is chosen to be greater than $r_{\text{TR}} + \max\{\|x - t_1\|, \|x - t_2\|\}$, to make sure that the close TRs with centers t_1 and t_2 will not be hit.

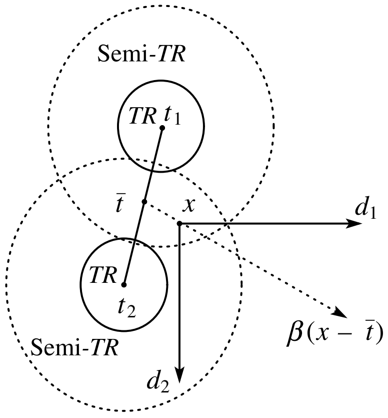

Fig. 1. Neighborhood search from a point in semi-TRs.

2.2. Visited region list (VRL)

Some historical information about the previously visited regions is stored in the VRL. More specifically, the center ζ_i of a visited region, which is a sphere with radius ρ_i, and the frequency φ_i of visiting this region comprise the information stored in the VRL, i.e., $\text{VRL} = \{(\zeta_i, \rho_i, \varphi_i)\}_{i=1}^{M}$, where M is the number of all listed visited regions. The information in VRL will be used to direct the search towards new regions whenever the current TS procedure fails to get improvement or whenever a Diversification scheme is needed. As a Diversification scheme, we try to generate new trial points outside the visited regions. However, generating trial points near to more frequently visited regions is discouraged. To this end, a function $\Phi(\varphi)$ is introduced to distinguish between more and less frequently visited regions. Specifically, we define the function Φ as

$$\Phi(\varphi) = \gamma(1 - \mathrm{e}^{-\gamma(\varphi-1)}), \tag{6}$$

where $\gamma \in (0,1]$ is a given constant. Note that the function Φ is strictly increasing and bounded above by the value γ. We will describe the role of γ in the Diversification scheme after we state Procedure 2.2 below.

In the following, we suggest a procedure that uses the VRL information to generate a new solution. The procedure allows accepting a trial point outside the visited regions, especially the more frequently visited ones.

Procedure 2.2

1. Generate a trial point x randomly in the search domain of f.
2. Compute the quantities $d_i = \|x - \zeta_i\|/(1 + \Phi(\varphi_i))$, $i = 1,\ldots,M$, where $\Phi(\varphi)$ is defined by (6). If $\min_{1\leqslant i\leqslant M} d_i/\rho_i \geqslant 1$, then accept x. Otherwise, return to Step 1.

A point x is accepted by Procedure 2.2 if it satisfies $\|x - \zeta_i\|/\rho_i \geqslant 1 + \Phi(\varphi_i)$ for all $i = 1,\ldots,M$. This means that no point can be accepted inside a previously visited region. Moreover, a point close to more frequently visited regions is hardly accepted. Therefore, the higher the value of γ is, the lower the possibility of accepting a point close to more frequently visited regions is. To avoid infinitely cycling in Procedure 2.2, we may also terminate it after a predetermined number of iterations and return with x corresponding to the maximum of the values of $\min_{1\leqslant i\leqslant M} d_i/\rho_i$ over all iterations.

3. Neighborhood–local search strategies

To explore the region around a solution and to generate the next move, we use neighborhood and local search strategies in which direct search methods are employed. Specifically, two search strategies are introduced to handle that job; Nelder–Mead Search (NMS) strategy and Adaptive Pattern Search (APS) strategy, which are based on the well-known Nelder–Mead method [24] and the recently proposed pattern search method [14], respectively. These neighborhood–local search strategies are invoked to generate trial points in the Exploration Search of the DTS method. More specifically, two types of trial points are generated by the neighborhood–local search strategy; neighborhood trial points and local trial points, which are needed in the Neighborhood Search and Local Search Steps, respectively, in Algorithm 4.1 stated in the next section. First, p trial points $\{y_i\}_{i=1}^{p}$ are generated in a neighborhood of the current solution x. This procedure is called a neighborhood search, and the trial points $\{y_i\}_{i=1}^{p}$ are called neighborhood trial points. Then, we try to improve the neighborhood trial points $\{y_i\}_{i=1}^{p}$ by executing another search procedure, which is called a local search, to generate q trial points $\{y_{p+i}\}_{i=1}^{q}$, which are called local trial points. The details of the neighborhood–local search strategies, NMS and APS, are given below.

3.1. Nelder–Mead search (NMS) strategy

In the NMS strategy, we generate $p(=n)$ neighborhood trial points $\{y_i\}_{i=1}^{n}$, and $q(=1$ or $0)$ local trial point. The neighborhood trial points are generated along search directions parallel to the coordinates axes starting from the current solution x with suitable step sizes. If the current solution x lies in a Semi-TR or in Semi-TRs, we apply Procedure 2.1 to construct the search directions and the step sizes. Otherwise, we construct search directions parallel to the coordinate axes in a random way, i.e., each of them is parallel to a positive or a negative coordinate direction. To generate a local trial point, we construct a simplex S that consists of the current solution x and the current n neighborhood trial points $\{y_i\}_{i=1}^{n}$, i.e., $S = \{x, y_1, \ldots, y_n\}$. Some iterations of the NM method are applied starting from S. If an improvement point is obtained from these NM iterations, then we set the local trial point y_{n+1} equal to this improvement point, i.e., $q = 1$. Otherwise, there is no trial point, i.e., $q = 0$.

For more explanation of the NMS strategy, we show an example in two dimensions in Fig. 2. Given the current solution x, two neighborhood trial points y_1 and y_2 are generated in a neighborhood of x as in Fig. 2(a). To find a local trial point, we construct a simplex whose vertices are $S = \{x, y_1, y_2\}$, as in Fig. 2(b). Assuming that the worst point in S is y_2, we apply the Nelder–Mead method operations described in Fig. 2(c) to find a better movement. If one exists, we refer to this better movement as a local trial point.

3.2. Adaptive pattern search (APS) strategy

The main idea behind the APS strategy is based on the approximate descent direction (ADD) method proposed in [14]. The ADD method is a derivative-free procedure with high ability of producing a descent direction. In the ADD method, several points around a given point $x \in R^n$ are used to generate an approximate descent direction of function f at x. We implement the same procedure as in [14] to produce a new adaptive direction from standard pattern directions. Specifically, we construct n pattern directions parallel to the coordinate axes emanating from the point x and generate n trial points $\{y_i\}_{i=1}^{n}$ along these directions with a suitable step size. The adaptive direction v, along which we may expect to decrease the function value, is computed using these trial points as follows:

$$v = \sum_{i=1}^{n} \omega_i u_i, \qquad (7)$$

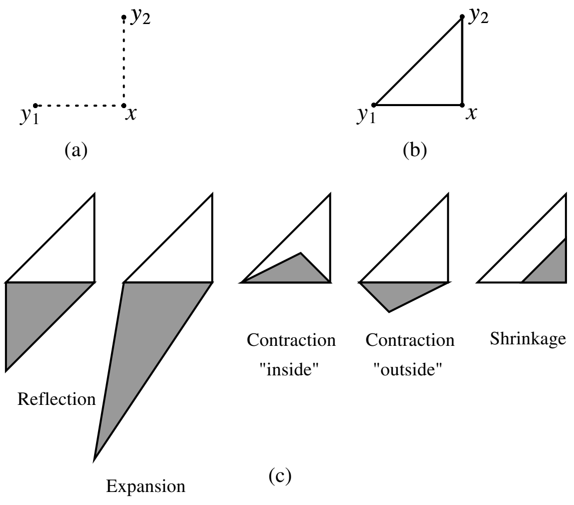

Fig. 2. NMS strategy in two dimensions.

where

$$\omega_i = \frac{\Delta f_i}{\sum_{j=1}^{n} |\Delta f_j|}, \quad i = 1,2,\ldots,n,$$

$$u_i = -\frac{(y_i - x)}{\|y_i - x\|}, \quad i = 1,2,\ldots,n,$$

$$\Delta f_i = f(y_i) - f(x), \quad i = 1,2,\ldots,n.$$

In the APS strategy, we generate $p(=n)$ neighborhood trial points $\{y_i\}_{i=1}^n$ using the standard pattern directions, and $q(=2)$ local trial points using an adaptive pattern direction. More specifically, we construct n pattern directions parallel to the coordinate axes emanating from the current solution x and generate n neighborhood trial points $\{y_i\}_{i=1}^n$ along these directions with some step size. The adaptive pattern direction v at x is computed using (7). Two local trial points y_{n+1} and y_{n+2} are generated along the vector v with two different step sizes.

An example in two dimensions is illustrated in Fig. 3 to describe the APS strategy. Two neighborhood trial points y_1 and y_2 are generated in a neighborhood of the current solution x as in Fig. 3(a). An approximate descent direction v is computed at x using the points y_1 and y_2, as in (7). If we assume that x is better than y_1 and y_2, then, by means of (7), the vector v is composed toward the vectors $x - y_1$ and $x - y_2$ with weights inversely proportional to $|f(x) - f(y_1)|$ and $|f(x) - f(y_2)|$, see Fig. 3(b). Finally, two local trial points y_3 and y_4 are generated along the vector v with two different step sizes in order to explore the area along the promising direction v as in Fig. 3(c).

4. Directed tabu search (DTS)

In this section, we describe some details about how a TS method is modified with the memory elements and neighborhood–local search strategies introduced in Sections 2 and 3 to compose the DTS method.

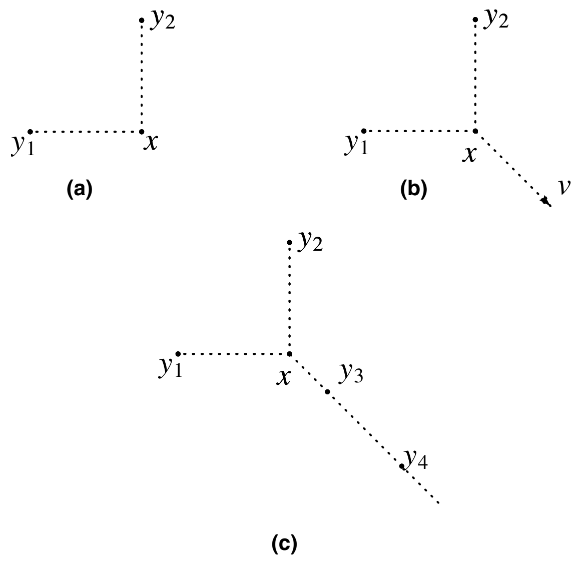

Fig. 3. APS strategy in two dimensions.

In the DTS method, three basic search procedure are used; exploration, Diversification and Intensification Search procedures. In the Exploration Search, we use the neighborhood–local search strategies, which are described in Section 3, to explore the solution space. Moreover, the multi-ranked TL, TR and Semi-TR restriction rules are applied to avoid revisiting recently visited solutions or being entrapped in local minima. Then, the Diversification Search is needed in order to diversify the search to other areas of the solution space that may have been overlooked in the Exploration Search. We use the VRL and Procedure 2.2 to manage the Diversification Search. Finally, in order to explore the close regions around the best points visited so far, the Intensification Search is applied to refine these best points. These search procedures are applied in such a way that they give the DTS method a better chance to explore the search space efficiently. Actually, the exploration and Diversification Search procedures are assembled to compose the DTS main loop and are repeated until the termination conditions are satisfied. Moreover, the Exploration Search procedure is included as an inner loop within the Diversification loop. We will use the superscript $j = 0, 1, \ldots,$ to represent the main loop iteration counter, the subscript $k = 0, 1, \ldots,$ to represent the inner loop iteration counter, and $x_k^{(j)}$ to denote a general iterate. In other words, the exploration and Diversification Search procedures compose a multi-start procedure with a long term memory. At the final stage, the Intensification Search procedure based on elite TS is needed to complete the DTS method. The main structure of the DTS method is shown in Fig. 4. More detailed description of the search procedures is given below.

4.1. Exploration–Diversification loop

The main loop of the DTS method, which consists of exploration and Diversification Searches, starts with an initial solution $x_0^{(0)}$. In each main loop iteration j, the Exploration Search procedure is repeatedly applied to obtain improvement by means of neighborhood–local search strategies, and then the Diversification Search procedure is applied to locate a new initial trial point $x_0^{(j+1)}$, from which the Exploration Search is restarted again. This main loop is repeated at most ℓ_{main} times, where ℓ_{main} is a predetermined positive integer.

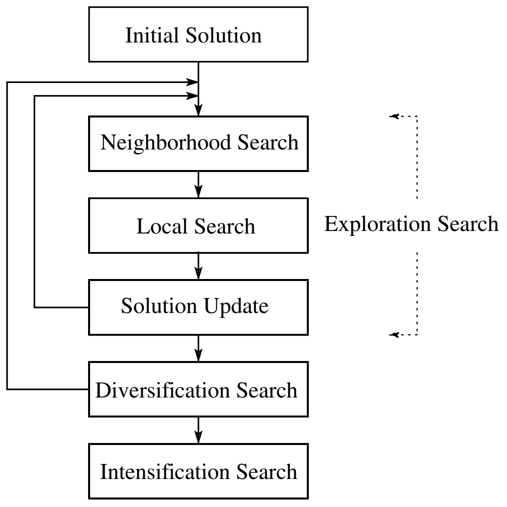

Fig. 4. Main structure of the DTS method.

Exploration Search. The Exploration Search starts with an initial solution $x_0^{(j)}$ at each main loop iteration j. In each iteration of the Exploration Search, a neighborhood–local search (NMS or APS) strategy is used to generate n neighborhood trial points $\{y_i\}_{i=1}^n$ in a neighborhood of the current iterate $x_k^{(j)}$. If a better movement is found among these trial points, we update the current iterate and proceed to the next inner loop iteration. Otherwise, i.e., $x_k^{(j)}$ is still better than all neighborhood trial points, the neighborhood–local search strategy continues to generate q local trial points $\{y_{n+i}\}_{i=1}^q$, where $q = 0$ or 1 in the NMS strategy and $q = 2$ in the APS strategy. Then, the current iterate $x_k^{(j)}$ is updated to be the best of neighborhood and local trial points, i.e., $x_{k+1}^{(j)} := \arg\min_{i=1,\dots,n+q}\{f(y_i)\}$. TL is also updated by letting $x_k^{(j)}$ replace the element with the smallest membership value. If a new region is reached, then VRL should be updated by adding the information on this region. This Exploration Search loop is repeated at most ℓ_{inner} times, where ℓ_{inner} is a predetermined positive integer.

Diversification Search. The Diversification Search is carried out when the Exploration Search either spends the inner iterations ℓ_{inner} times or fails to obtain an improvement in some consecutive iterations. With the current VRL, Procedure 2.2 is applied to generate a trial point $x_0^{(j+1)}$ in some new region. Then the Exploration Search is repeated again starting from $x_0^{(j+1)}$.

4.2. Intensification Search

According to the principle of the multi-ranked TL, it reserves the best points visited so far. In order to improve these points, we complete the DTS method by applying another local search method starting from some of these points, which we call Intensification Search. We use Kelley's modification [16,17] of the Nelder–Mead (NM) method as a local search method for this task.

4.3. Main algorithm

We have two versions of the DTS method; DTS$_{\text{NMS}}$ and DTS$_{\text{APS}}$ that use the NMS strategy and the APS strategy, respectively, as neighborhood–local search strategies. First, a specific and formal description of DTS$_{\text{NMS}}$ is given in the following Algorithms 4.1.

Algorithm 4.1. $\mathbf{DTS}_{\mathrm{NMS}}(f, x_0^{(0)})$

```
1. Initialization. Choose positive integers $\ell_{\mathrm{main}}$, $\ell'_{\mathrm{main}}$, $\ell_{\mathrm{inner}}$ and $\ell'_{\mathrm{inner}}$. Choose an initial solution $x_0^{(0)}$, and set TL and VRL to be empty.
2. Exploration–Diversification Search (Main Loop). Let $j := 0$ and repeat this main loop until $\ell'_{\mathrm{main}}$ consecutive main iterations fail to obtain improvement or the main loop iteration counter $j$ exceeds $\ell_{\mathrm{main}}$.
    2.1. Exploration Search (NMS) (Inner Loop). Let $k := 0$ and repeat this inner loop until $\ell'_{\mathrm{inner}}$ consecutive inner iterations fail to obtain improvement or the inner loop iteration counter $k$ exceeds $\ell_{\mathrm{inner}}$.
        2.1.1. Search Directions. If the current iterate $x_k^{(j)}$ lies in Semi-TRs, use Procedure 2.1 to construct search directions $\{d_i\}_{i=1}^n$ and to choose step sizes $\{\Delta_i\}_{i=1}^n$. Otherwise, construct search directions $d_i := (-1)^{\tau_i} e_i, i = 1, \ldots, n$, where $e_i \in R^n$ is the $i$th unit vector in $R^n$ and $\tau_i$ is a random integer number, and choose suitable step sizes $\{\Delta_i\}_{i=1}^n$.
        2.1.2. Neighborhood Search. Generate $n$ neighborhood trial points $y_i := x_k^{(j)} + \Delta_i d_i$, $i = 1, \ldots, n$. Whenever a better movement is found during this process, stop generating points, set $x_{k+1}^{(j)}$ equal to this better movement, and go to Step 2.1.4.
        2.1.3. Local Search. Apply $n$ iterations of the NM method starting from the simplex $S := \{x_k^{(j)}, y_1, \ldots, y_n\}$. If an improvement point is obtained from these NM iterations, set local trial point $y_{n+1}$ equal to this improvement point, and set $q := 1$. Otherwise, set $q := 0$. Set $x_{k+1}^{(j)} := \arg\min_{i=1,\ldots,n+q}\{f(y_i)\}$.
        2.1.4. Parameter Update. Let $x_k^{(j)}$ replace the element with the smallest membership value in TL and re-rank the TL elements using (1). Update the VRT and set $k := k + 1$.
    2.2 Diversification Search. Generate a trial point $x_0^{(j+1)}$ using Procedure 2.2. Update the TL and VRT, and set $j := j + 1$.
3. Intensification Search. Apply the Kelley's modification [16] of the NM method starting from some elite solutions in the TL.
```

The $\mathrm{DTS}_{\mathrm{APS}}$ algorithm is the same as Algorithm 4.1 except Step 2.1, which should be changed as follows:

```
    2.1. Exploration Search (APS) (Inner Loop). Let $k := 0$, initialize a vector $v$ to be a random vector in $R^n$, and repeat this inner loop until $\ell'_{\mathrm{inner}}$ consecutive inner iterations fail to obtain improvement or the inner loop iteration counter $k$ exceeds $\ell_{\mathrm{inner}}$.
        2.1.1. Search Directions. If the current iterate $x_k^{(j)}$ lies in Semi-TRs, use Procedure 2.1 to construct search directions $\{d_i\}_{i=1}^n$ and to choose step sizes $\{\Delta_i\}_{i=1}^n$. Otherwise, construct search directions $d_i := \mathrm{sign}(v_i)e_i$, $i = 1, \ldots, n$, where $e_i \in R^n$ is the $i$th unit vector in $R^n$ and $v_i$ is the $i$th component of $v$, and choose suitable step sizes $\{\Delta_i\}_{i=1}^n$.
        2.1.2. Neighborhood Search. Generate $n$ neighborhood trial points $y_i := x_k^{(j)} + \Delta_i d_i$, $i = 1, \ldots, n$. Whenever a better movement is found during this process, stop generating points, set $x_{k+1}^{(j)}$ equal to this better movement, and go to Step 2.1.4.
        2.1.3. Local Search. Compute the direction $v$ at $x_k^{(j)}$ using $\{y_i\}_{i=1}^n$ as in (7). Choose two suitable step sizes $\alpha_1$ and $\alpha_2$ to generate local trial points $y_{n+i} = x_k^{(j)} + \alpha_i v/\|v\|$, $i = 1, 2$. Set $x_{k+1}^{(j)} := \arg\min_{i=1,\ldots,n+2}\{f(y_i)\}$.
        2.1.4. Parameter Update. Let $x_k^{(j)}$ replace the element with the smallest membership value in TL and re-rank the TL elements using (1). Update the VRT and set $k := k + 1$.
```

5. Implementation and experiments

In this section, we give more details about the implementation as well as the experimental results of the DTS algorithms.

5.1. Setting parameters

In this subsection, we discuss the suggested values of the parameters needed in the implementation of the DTS algorithms and the sensitivity of these parameters. First, the initial trial solution $x_0^{(0)}$ is chosen randomly from the predetermined range $[\mathfrak{L}, \mathfrak{U}]$ of the initial points for each test function, where $[\mathfrak{L}, \mathfrak{U}] := \{x \in R^n : l_i \leqslant x_i \leqslant u_i,\ i = 1, \ldots, n\}$. The other parameters can be classified into the following groups.

- *TR and Semi-TR parameters*: the radius r_{TR} of each TR, and the outer radius r_{STR} of the Semi-TR.
- *TL parameters*: the number L of elements in TL, the maximum and minimum recency ranked values $\eta_{\max}$ and $\eta_{\min}$, respectively, the number $\overline{L}$ of the function-value ranked elements, and the maximum and minimum function-value ranked values $\mu_{\max}$ and $\mu_{\min}$, respectively.
- *VRL parameters*: the radii ρ_j, $j = 1, \ldots, M$, of the visited regions.
- *Step sizes*: the step sizes Δ_i, $i = 1, \ldots, n$, used in generating neighborhood trial points in $\mathrm{DTS_{NMS}}$ and $\mathrm{DTS_{APS}}$, and α_1 and α_2 used to generate local trial points in $\mathrm{DTS_{APS}}$.
- *Diversification trials*: the parameter γ used in (6) and the maximum number $It_{\max}$ of iterations allowed in Procedure 2.2.
- *Intensification trials*: the number N_{best} of best points that are used in the Intensification Search.
- *Termination conditions*: the loop termination numbers ℓ_{main}, ℓ'_{main}, ℓ_{inner} and ℓ'_{inner}.

Proper values of these parameters have been studied through extensive numerical experiments by using the functions Branin (RC), Goldstein and Price (GP), Rosenbrock (R_2) and Zakhrov (Z_2) and (Z_5). In the tuning parameters experiments, we have tried to find a standard setting of the DTS parameters which is problem-independent as much as possible. Moreover, some relations between the parameters have been discussed to guide the user to set the DTS parameters whenever new implementations of the DTS algorithm are invoked. Below, we highlight the conclusion we got from the tuning parameters experiments.

First, the values of the parameter γ that we have studied are $0.10, 0.15, 0.2, \ldots, 0.4$. Recall that the main role of this parameter is to avoid generating a new diverse trial solution near to the more frequently visited regions. Since large γ may lead to a big area surrounding the more frequently visited regions left without exploration, we did not test a value of γ higher than 0.4. The performance of the DTS algorithms is almost the same for all runs using the above-mentioned values of γ. Moreover, at the end of running the DTS algorithms on many test functions, the centers of the Visited Regions Listed in the VRL are distributed almost uniformly for all tested values of γ. However, the value $\gamma = 0.25$ produces slightly more scattered distributions than the other values. Therefore, we set γ equal to 0.25. The parameter $It_{\max}$ is set equal to $100n$.

Most of the DTS parameters listed above are distance parameters. These distance parameters are r_{TR}, r_{STR}, ρ, Δ_i, $i = 1, \ldots, n$, α_1 and α_2. Note that the radii ρ_j, $j = 1, \ldots, M$, of all visited regions are set equal to ρ. For more accurate setting of the values of these distance parameters, we consider the following:

(1) Since Semi-TRs are surrounding TRs, we let $r_{\mathrm{STR}} > r_{\mathrm{TR}}$. For easily escaping from TRs and Semi-TRs, it is desirable to let $\Delta_i > r_{\mathrm{STR}}$. Moreover, to avoid producing too many small visited regions, we let $\rho > \Delta_i$. This means that the desirable order of the distance parameters is $r_{\mathrm{TR}} < r_{\mathrm{STR}} < \Delta_i < \rho$. Since the step sizes α_1 and α_2, are used to search the area along an approximate descent direction, it is appropriate to let one of them be smaller than the usual step size Δ_i and the other be greater than Δ_i.

(2) To keep the distance parameters in the above order, we let their values relate to only one parameter δ which is the diameter of the range $[\mathfrak{L}, \mathfrak{U}]$ defined as $\delta := \max_{1 \leqslant i \leqslant n}(u_i - l_i)$.

The performance of the DTS algorithms were tested using different values for these parameters through many test functions. The suggested values of these parameters are given in Table 1, and r_{STR} is set equal to $2r_{\text{TR}}$. The performance of the DTS algorithms were almost insensitive with regard to all tested values of the distance parameters. In Table 1, we also suggest the value for each parameter which produces the best performance. For more efficient search, the step sizes may be randomly chosen close to some fixed mean values, rather than being set at fixed values. Specifically, we set the step sizes as follows:

$$\begin{aligned}
\Delta_i &= (0.1 + 0.025\omega_i)\delta, \quad i = 1, \ldots, n,\\
\alpha_1 &= (0.1 - 0.05\theta_1)\delta,\\
\alpha_2 &= (0.1 + 0.05\theta_2)\delta,
\end{aligned}$$

where ω_i, $i = 1,\ldots,n$, are random numbers from the interval $(-1,1)$, and θ_1 and θ_2 are random numbers from the interval $(0,1)$.

For the TL parameters, the values $5n$, $6n$, $7n$ and $8n$ and the values $2n$, $3n$ and $4n$ have been tested as possible choices for L and $\overline{L}$, respectively. The performance of the DTS algorithms using these values of L and $\overline{L}$ is almost the same. So, to avoid storing unnecessary information, we set L and $\overline{L}$ equal to the least possible values, i.e., $L = 5n$ and $\overline{L} = 2n$. The other TL parameters are set as $\eta_{\max} = \mu_{\max} = 1$ and $\eta_{\min} = \mu_{\min} = 1/L$.

The parameter N_{best} is set equal to 1 because the numerical results show that the best point found in the exploration–Diversification Search is close to global minima for most of the test function.

The last group of parameters are related to the termination conditions. Actually, choosing sufficiently large values for the loop termination numbers ℓ_{main}, ℓ'_{main}, ℓ_{inner} and ℓ'_{inner} is highly needed to avoid premature termination of the method. The numerical results have shown that the lowest values of these parameters that can give an acceptable performance of the DTS algorithms are $\ell_{\text{main}} = \ell_{\text{inner}} = 5n$, and $\ell'_{\text{main}} = \ell'_{\text{inner}} = 2n$. However, higher values for these numbers can increase the ability of finding global minima for some difficult test problems.

5.2. Numerical results

In this subsection, we discuss the performance of the DTS algorithms through two main experiments. The first experiment is to compare the results obtained by the DTS_{NMS} and DTS_{APS} and then compare the best version of them with other continuous versions of TS. In the second experiment, the performance of the best DTS algorithms is also compared with other metaheuristics.

5.2.1. Numerical results of DTS and other TS methods

To examine the performance of the DTS algorithms DTS_{NMS} and DTS_{APS}, we tested them on some well known functions [6,13] listed as Set A in Table 2. The characteristics of these test functions are diverse enough to cover many kinds of difficulties that arise in global optimization problems. To complete the evaluation of the DTS algorithms, they should be compared with other continuous versions of TS. However, it is not easy to show complete and fair comparisons due to the lack of some information especially on the

Table 1
Distance parameters

Parameter	Tested values	Suggested value
r_{TR}	0.01δ, 0.02δ, 0.03δ, 0.04δ	0.01δ
Δ_i	0.08δ, 0.1δ, 0.12δ	0.1δ
ρ	0.15δ, 0.2δ, 0.25δ	0.15δ

Table 2
Test functions (Set A)

No.	f	Function name	n	No.	f	Function name	n
1	RC	Branin RCOS	2	9	$S_{4,5}$	Shekel	4
2	ES	Easom	2	10	$S_{4,7}$	Shekel	4
3	GP	Goldstein&Price	2	11	$S_{10,7}$	Shekel	4
4	SH	Shubert	2	12	Z_5	Zakharov	5
5	Z_2	Zakharov	2	13	R_5	Rosenbrock	5
6	R_2	Rosenbrock	2	14	$H_{6,4}$	Hartmann	6
7	DJ	De Joung	3	15	Z_{10}	Zakharov	10
8	$H_{3,4}$	Hartmann	3	16	R_{10}	Rosenbrock	10

quality of solutions obtained by those continuous TS methods. Therefore, we try to compare our algorithms with other continuous TS methods in terms of the ability of obtaining global minima, the cost of function evaluations and the quality of computed solutions. Three continuous TS methods chosen to compare with the DTS algorithms are continuous reactive TS (CRTS) [2], enhanced continuous tabu search (ECTS) [3], and TS-based algorithm called DOPE [7]. The ECTS and DOPE methods are the most recent continuous TS methods and the quality of computed solutions are stated clearly in their original papers.

For each function in Set A, we applied the DTS codes 100 times with different starting points. For all these test functions, we used the same values of the DTS parameters as those presented in Section 5.1. Moreover, we used the same condition as that used by ECTS [3] to judge the success of a trial, which is given by

$$|f^* - \tilde{f}| < \epsilon_1 |f^*| + \epsilon_2, \tag{8}$$

where $\tilde{f}$ refers to the best function value obtained by the algorithm, f^* refers to the exact global minimum, and ϵ_1 and ϵ_2 are set equal to 10^{-4} and 10^{-6}, respectively. Note that the conditions for successful trials are not stated for CRTS and DOPE in the original papers [2,7].

The results of the two versions of DTS method, $\mathrm{DTS_{NMS}}$ and $\mathrm{DTS_{APS}}$, are reported in Table 3. These results represent the average number of function evaluations (Av. f-evals.) with minimum and maximum numbers in parentheses, the average errors (Av. Error) and the success rates (Suc.) for each function. The average number of function evaluations and the average error only relate to successful trials. The results shown in Table 3 reveal that the performance of $\mathrm{DTS_{APS}}$ is consistently better than $\mathrm{DTS_{NMS}}$ in terms of function evaluations and the ability of obtaining global minima. Moreover, it seems that $\mathrm{DTS_{NMS}}$ suffers from the curse of dimensionality as is seen from the Av. f-evals. for higher dimensional problems.

Table 4 compares $\mathrm{DTS_{APS}}$ with the above-mentioned continuous TS methods in terms of the average number of function evaluations. The results of CRTS, ECTS and DOPE methods are taken from their original papers [2,3,7]. The percentages in parentheses represent the success rates of reaching global minima. The quality of the computed solutions by those methods except the CRTS method is shown in Table 5, where the errors are measured in terms of function values at the computed and exact solutions. The quality of the produced solutions by the CRTS method is not stated clearly in [2], but it is only said that the statistical error on the CRTS is about 3%. Before judging the comparison of these method, some remarks are made in regard to the reported success rates of ECTS and the termination condition of DOPE.

• The ECTS method uses condition (9) to test the success of a trial [3]. However, the results marked by ($^{\circledast}$) in Tables 4 and 5 seem to contain some inconsistencies. In fact, from condition (9), the average errors for functions R_2, R_5 and Z_5 must be less than 10^{-6} because $f^* = 0$ for all these functions. However, in Table 5, they are reported to be greater than 10^{-6}. For instance, the average error for function R_5 in Table 5 is

Table 3
Results of DTS algorithms

f	DTS_{NMS}			DTS_{APS}		
	f-evals. Av. (min/max)	Av. Er.	Suc. (%)	f-evals. Av. (min/max)	Av. Er.	Suc. (%)
RC	274 (252/296)	4e−7	100	212 (181/243)	4e−7	100
ES	271 (202/285)	5e−9	30	223 (156/244)	4e−9	82
GP	293 (276/324)	5e−9	88	230 (207/282)	5e−9	100
SH	298 (282/319)	9e−6	44	274 (260/307)	9e−6	92
Z_2	273 (247/291)	6e−9	100	201 (183/225)	5e−9	100
R_2	358 (272/489)	6e−9	100	254 (207/321)	5e−9	100
DJ	650 (600/694)	5e−9	100	446 (393/516)	4e−9	100
$H_{3,4}$	670 (613/789)	2e−6	97	438 (389/493)	2e−6	100
$S_{4,5}$	1426 (1342/1473)	7e−7	39	819 (669/989)	3e−7	75
$S_{4,7}$	1425 (1372/1487)	4e−5	29	812 (675/973)	4e−5	65
$S_{4,10}$	1438 (1340/1493)	1e−5	22	828 (706/963)	1e−5	52
Z_5	2458 (2301/2597)	6e−9	100	1003 (903/1093)	7e−9	100
R_5	2895 (2523/3473)	7e−9	75	1684 (1326/2093)	6e−9	85
$H_{6,4}$	3978 (3618/4308)	2e−6	68	1787 (1489/2036)	2e−6	83
Z_{10}	16392 (14235/17821)	2e−8	100	4032 (3689/4809)	2e−8	100
R_{10}	19139 (16844/22416)	2e−8	78	9037 (6701/12879)	2e−8	85

Table 4
Average numbers of function evaluations for continuous TS methods

f	DTS_{APS}	ECTS	DOPE	CRTS	
				CRTS_{Ave}	CRTS_{Min}
RC	212	$245^{\circledast}$	31	38	41
ES	223 (82%)	$1284^{\circledast}$	290	–	–
GP	230	$231^{\circledast}$	248	248	171
SH	274 (92%)	370	466	–	–
Z_2	201	195	81	–	–
R_2	254	$480^{\circledast}$	692	–	–
DJ	446	338	131	–	–
$H_{3,4}$	438	$548^{\circledast}$	135	513	609
$S_{4,5}$	819 (75%)	825 (75%)	–	812	664
$S_{4,7}$	812 (65%)	910 (80%)	–	960	871
$S_{4,10}$	828 (52%)	898 (75%)	–	921	693
Z_5	1003	$2254^{\circledast}$	424	–	–
R_5	1684 (85%)	$2142^{\circledast}$	2512	–	–
$H_{6,4}$	1787 (83%)	$1520^{\circledast}$	421	750	1245
Z_{10}	4032	4630	8695	–	–
R_{10}	9037 (85%)	15720 (85%)	5133	–	–

0.08, which means that there are some trials that did not satisfy condition (9). Nevertheless, the rate of success is reported to be 100%. Moreover, the results of ECTS for functions RC, ES, GP, $H_{3,4}$ and $H_{6,4}$ in Tables 4 and 5 also contain similar inconsistencies.

• According to [7], DOPE is terminated when either a maximum number of function evaluations is reached or the global minimum (if it is a priori known) is found. Since the information on global minima is not available in practice, we did not use the latter termination condition in our numerical experiments.

Table 5
Average errors for continuous TS methods

f	DTS$_{APS}$	ECTS	DOPE	f	DTS$_{APS}$	ECTS	DOPE
RC	4e−7	5e−2$^{\circledast}$	5e−2	$S_{4,5}$	3e−7	1e−2	–
ES	4e−9	1e−2$^{\circledast}$	1e−2	$S_{4,7}$	4e−5	1e−2	–
GP	5e−9	2e−3$^{\circledast}$	2e−3	$S_{4,10}$	1e−5	1e−2	–
SH	9e−6	1e−3	1e−3	Z_5	7e−9	4e−6$^{\circledast}$	4e−6
Z_2	5e−9	2e−7	2e−7	R_5	6e−9	8e−2$^{\circledast}$	8e−2
R_2	5e−9	2e−2$^{\circledast}$	2e−2	$H_{6,4}$	2e−6	5e−2$^{\circledast}$	5e−2
DJ	5e−9	3e−8	3e−8	Z_{10}	2e−8	2e−7	2e−2
$H_{3,4}$	2e−6	9e−2$^{\circledast}$	9e−2	R_{10}	2e−8	2e−2	2e−7

This termination condition may explain the extremely small number of function evaluations of DOPE for some test functions.

From these remarks, the comparisons of the DTS methods with the ECTS and DOPE methods do not seem to yield a definitive fair answer. However, in terms of the quality of computed solutions, the DTS$_{APS}$ algorithm seems to outperform ECTS and DOPE as shown in Table 5. Moreover, the DTS$_{APS}$ algorithm seems to outperform ECTS in terms of the number of function evaluations for functions in Set A. However, the drawback we have noticed on the DTS algorithms is its deterioration in high dimensional problems ($n > 30$). Actually, this can be expected since the search in the DTS algorithms is mainly controlled by direct search methods and it has been shown, for instance, in [18] that the latter methods deteriorate with the increase of the dimension, i.e., suffer from the curse of dimensionality. To show the limit of deterioration of the DTS performance with the dimensionality, we report some results for high dimensional problems. The results have been obtained by running the Matlab code of DTS$_{APS}$, with the parameter setting given in Section 5.1, on Pentium 2.8-GHz machine. For Rosenbrock R_{50} function, the DTS$_{APS}$ algorithm converged to a point close to the global minimum with function value 4.46×10^{-7} using 510,505 function evaluations in 1085 CPU seconds. For Zakharov Z_{50} and Rosenbrock R_{100} functions, the DTS$_{APS}$ algorithm obtained points not so close to the global minimum at distances 1.404 and 4.1057 with function values 1.972 and 4.106 using 177,125 and 3,202,879 function evaluations in 1,043 and 15,270 CPU seconds, respectively. These results of Z_{50} and R_{100} are the best among 5 runs. For Zakharov Z_{100}, the DTS$_{APS}$ algorithm failed to obtain a point near the global minimum by 5 runs using the same setting of parameters. As far as the results in Table 4 common to all methods are concerned, CRTS may be considered the best among the continuous TS methods in terms of the ability of obtaining global minima and the number of function evaluations.

5.2.2. The performance of DTS$_{APS}$ against other metaheuristics

The performance of DTS$_{APS}$ is compared with other metaheuristics using the test functions listed as Set B in Table 6 [19]. We choose two other metaheuristics proposed for the continuous optimization problem; Genetic algorithm for numerical optimization of constrained problems (Genocop III) [23], and Scatter Search (SS) [19]. We define the optimality gap (GAP) [19] as the quantity on the left-hand side of (8). Table 7 shows the average GAP for all 40 test functions in Set B. In Table 7, the figures related to Genocop III and SS are taken from [19] and represent the average GAP for all test functions in Set B at intermediate stages during the search. Since the DTS$_{APS}$ algorithm consists of two complementary parts (exploration–Diversification Search and Intensification Search), its results in Table 7 are the average GAP for all test functions in Set B obtained by running the DTS$_{APS}$ code 7 times for each test function with the termination condition that the number of function evaluations exceeds 100, 500, 1000, 5000, 10,000, 20,000 and 50,000, respectively. Since Genocop III has a bad performance on the test function No. 23 (SC_6), the results excluding this function are also included.

Table 6
Test functions (Set B)

No.	f	Function name	n	No.	f	Function name	n
1	RC	Branin RCOS	2	21	$PS_{8,18,44,114}$	Power Sum	4
2	B_2	Bohachevsky	2	22	$H_{6,4}$	Hartmann	6
3	ES	Easom	2	23	SC_6	Schwefel	6
4	GP	Goldstein&Price	2	24	T_6	Trid	6
5	SH	Shubert	2	25	T_{10}	Trid	10
6	BL	Beale	2	26	RT_{10}	Rastrigin	10
7	BO	Booth	2	27	G_{10}	Griewank	10
8	MT	Matyas	2	28	SS_{10}	Sum Squares	10
9	HM	Hump	2	29	R_{10}	Rosenbrock	10
10	SC_2	Schwefel	2	30	Z_{10}	Zakharov	10
11	R_2	Rosenbrock	2	31	RT_{20}	Rastrigin	20
12	Z_2	Zakharov	2	32	G_{20}	Griewank	20
13	DJ	De Joung	3	33	SS_{20}	Sum Squares	20
14	$H_{3,4}$	Hartmann	3	34	R_{20}	Rosenbrock	20
15	CV	Colville	4	35	Z_{20}	Zakharov	20
16	$S_{4,5}$	Shekel	4	36	PW_{24}	Powell	24
17	$S_{4,7}$	Shekel	4	37	DP_{25}	Dixon&Price	25
18	$S_{4,10}$	Shekel	4	38	L_{30}	Levy	30
19	$P_{4,0.5}$	Perm	4	39	SR_{30}	Sphere	30
20	$P^0_{4,0.5}$	Perm	4	40	AK_{30}	Ackley	30

Table 7
Average optimality gap values

f-evals.	100	500	1000	5000	10,000	20,000	50,000
Genocop III[a]	5.37E+25	2.39E+17	1.13E+14	636.37	399.52	320.84	313.34
Genocop III[b]	1335.45	611.30	379.03	335.81	328.66	324.72	321.20
Scatter Search	134.45	26.34	14.66	4.96	3.60	3.52	3.46
DTS$_{\mathrm{APS}}$	5.04E+04	43.06	24.26	4.22	1.80	1.70	1.29

[a] Average values for all test functions.
[b] Average values for all test functions except function 23.

According to the results in Table 7, the performance of DTS$_{\mathrm{APS}}$ is generally better than Genocop III and SS when the number of function evaluations is greater than 1000. However, in the early stage of computations, SS performs better than DTS$_{\mathrm{APS}}$. This can be expected since DTS$_{\mathrm{APS}}$ is a point-to-point search method while SS is a population-based search method. So, DTS$_{\mathrm{APS}}$ may need more iterations, and therefore more function evaluations, to explore the search space well especially for high dimensional functions ($n \geqslant 6$).

Since the data related to Genocop III and SS in Fig. 5 are taken from [19], we also made the successful trial test [19] for the DTS$_{\mathrm{APS}}$ results in our experiments. We say that a method approximately finds an optimal solution if

$$\mathrm{GAP} \leqslant \begin{cases} 0.001, & \text{if } f^* \neq 0, \\ 0.001 f^*, & \text{otherwise.} \end{cases}$$

The graphs in Fig. 5 show the number of test functions from Set B that were approximately solved by each method. Fig. 5 shows that DTS$_{\mathrm{APS}}$ could approximately find global minima for 14 test functions within

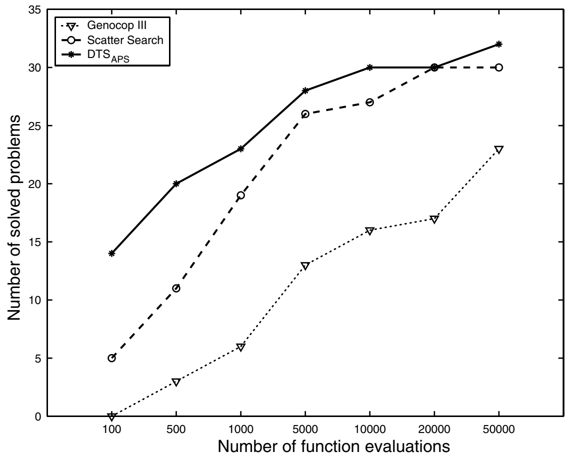

Fig. 5. Number of solved problems.

only 100 function evaluations. Actually, the dimensions of these 14 test functions are less than or equal to 6. Fig. 5 also show that DTS_{APS} generally outperforms the other two methods, Genocop III and SS.

6. Conclusion

In this paper, we have presented a continuous TS method called directed tabu search (DTS) method. First, neighborhood–local search strategies are introduced to provide more powerful search procedures to generate trial moves. A new pattern search procedure and the NM method are used to construct these neighborhood–local search strategies. Moreover, new memory elements called TR, Semi-TR and multi-ranked TL are applied to compose anti-cycling rules and to escape from local minima. Finally, a Diversification scheme based on the memory element VRL is introduced to explore broad areas of the solution space. The numerical results show the promise of the proposed method.

Acknowledgement

The authors thank Rafael Marti and Manuel Laguna for providing us the data of SS and Genocop III used in Fig. 5.

Appendix A. Test functions

(AK_n) **Ackley function**
Definition: $AK_n(\mathbf{x}) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{-\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$.
Search space: $-15 \leqslant x_i \leqslant 30$, $i = 1,2,\ldots,n$.
Global minimum: $\mathbf{x}^* = (0,\ldots,0)$; $AK_n(\mathbf{x}^*) = 0$.

(B_2) **Bohachevsky function**
Definition: $B_2(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$.
Search space: $-50 \leqslant x_i \leqslant 100$, $i = 1,2$.
Global minimum: $\mathbf{x}^* = (0,0)$; $B_2(\mathbf{x}^*) = 0$.

(BL) **Beale function**
Definition: $BL(\mathbf{x}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$.
Search space: $-4.5 \leqslant x_i \leqslant 4.5$, $i = 1,2$.
Global minimum: $\mathbf{x}^* = (3, 0.5)$; $BL(\mathbf{x}^*) = 0$.

(BO) **Booth function**
Definition: $BO(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$,
Search space: $-10 \leqslant x_i \leqslant 10$, $i = 1,2$
Global minimum: $\mathbf{x}^* = (1,3)$; $BO(\mathbf{x}^*) = 0$.

(CV) **Colville function**
Definition: $CV(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2)$ $+19.8(x_2 - 1)(x_4 - 1)$.
Search space: $-10 \leqslant x_i \leqslant 10$, $i = 1,\ldots,4$.
Global minimum: $\mathbf{x}^* = (1,1,1,1)$; $CV(\mathbf{x}^*) = 0$.

(DJ) **De Joung function**
Definition: $DJ(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$.
Search space: $-2.56 \leqslant x_i \leqslant 5.12$, $i = 1,2,3$.
Global minimum: $\mathbf{x}^* = (0,0,0)$; $DJ(\mathbf{x}^*) = 0$.

(DP_n) **Dixon and Price function**
Definition: $DP_n(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$.
Search space: $-10 \leqslant x_i \leqslant 10$, $i = 1,\ldots,n$.
Global minimum: $x_i^* = 2^{-\left(\frac{2^i-2}{2^i}\right)}$, $i = 1,\ldots,n$; $\mathrm{DP}_n(\mathbf{x}^*) = 0$.

(ES) **Easom function**
Definition: $ES(\mathbf{x}) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$.
Search space: $-100 \leqslant x_i \leqslant 100$, $i = 1,2$.
Global minimum: $\mathbf{x}^* = (\pi,\pi)$; $ES(\mathbf{x}^*) = -1$.

(G_n) **Griewank function**
Definition: $G_n(\mathbf{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$.
Search space: $-300 \leqslant x_i \leqslant 600$, $i = 1,\ldots,n$.
Global minimum: $\mathbf{x}^* = (0,\ldots,0)$, $G_n(\mathbf{x}^*) = 0$.

(GP) **Goldstein and Price function**
Definition: $GP(\mathbf{x}) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) * (30 + (2x_1 - 3x_2)^2$ $(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2))$.
Search space: $-2 \leqslant x_i \leqslant 2$, $i = 1,2$.
Global minimum: $\mathbf{x}^* = (0,-1)$; $GP(\mathbf{x}^*) = 3$.

($H_{3,4}$) **Hartmann function**
Definition: $H_{3,4}(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left[-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right]$, $\alpha = [1, 1.2, 3, 3.2]^{\mathrm{T}}$,

$$A = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad P = 10^{-4}\begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$$

Search space: $0 \leqslant x_i \leqslant 1$, $i = 1,2,3$.
Global minimum: $\mathbf{x}^* = (0.114614, 0.555649, 0.852547)$; $H_{3,4}(\mathbf{x}^*) = -3.86278$.

($H_{6,4}$) **Hartmann function**
Definition: $H_{6,4}(\mathbf{x}) = -\sum_{i=1}^{4}\alpha_i \exp\left[-\sum_{j=1}^{6}B_{ij}(x_j - Q_{ij})^2\right]$, $\alpha = [1, 1.2, 3, 3.2]^{\mathrm{T}}$,

$$B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \quad Q = 10^{-4}\begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}.$$

Search space: $0 \leqslant x_i \leqslant 1$, $i = 1,\ldots,6$.
Global minimum: $\mathbf{x}^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300)$; $H_{6,4}(\mathbf{x}^*) = -3.32237$.

(HM) **Hump function**
Definition: $HM(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$.
Search space: $-5 \leqslant x_i \leqslant 5$, $i = 1,2$.
Global minimum: $\mathbf{x}^* = (0.0898, -0.7126)$, $(-0.0898, 0.7126)$; $HM(\mathbf{x}^*) = 0$.

(L_n) **Levy function**
Definition: $L_n(\mathbf{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1}[(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1))] + (y_n - 1)^2(1 + 10\sin^2(2\pi y_n))$, $y_i = 1 + \frac{x_i - 1}{4}$, $i = 1,\ldots,n$.
Search space: $-10 \leqslant x_i \leqslant 10$, $i = 1,\ldots,n$.
Global minimum: $\mathbf{x}^* = (1,\ldots,1)$; $L_n(\mathbf{x}^*) = 0$.

(MT) **Matyas function**
Definition: $MT(\mathbf{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$.
Search space: $-5 \leqslant x_i \leqslant 10$, $i = 1,2$.
Global minimum: $\mathbf{x}^* = (0,0)$; $MT(\mathbf{x}^*) = 0$.

($P_{n,\beta}$) **Perm function**
Definition: $P_{n,\beta}(\mathbf{x}) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i^k + \beta)((x_i/i)^k - 1)\right]^2$.
Search space: $-n \leqslant x_i \leqslant n$, $i = 1,\ldots,n$.
Global minimum: $\mathbf{x}^* = (1,2,\ldots,n)$; $P_{n,\beta}(\mathbf{x}^*) = 0$.

($P^0_{n,\beta}$) **Perm function**
Definition: $P^0_{n,\beta}(\mathbf{x}) = \sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i + \beta)(x_i^k - (1/i)^k)\right]^2$.
Search space: $-n \leqslant x_i \leqslant n$, $i = 1,\ldots,n$.
Global minimum: $\mathbf{x}^* = (1, \frac{1}{2}, \ldots, \frac{1}{n})$; $P^0_{n,\beta}(\mathbf{x}^*) = 0$.

($PS_{b_1,\ldots,b_n}$) **Power sum function**
Definition: $PS_{b_1,\ldots,b_n}(\mathbf{x}) = \sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n}x_i^k\right) - b_k\right]^2$.
Search space: $0 \leqslant x_i \leqslant n$, $i = 1,\ldots,n$.
Global minimum for $PS_{8,18,44,114}(\mathbf{x})$: $\mathbf{x}^* = (1,2,2,3)$; $PS_{8,18,44,114}(\mathbf{x}^*) = 0$.

(PW_n) **Powell function**
Definition: $PW_n(\mathbf{x}) = \sum_{i=1}^{n/4}(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$.
Search space: $-4 \leqslant x_i \leqslant 5$, $i = 1,\ldots,n$.
Global minimum: $\mathbf{x}^* = (3,-1,0,1,3,\ldots,3,-1,0,1)$; $PW_n(\mathbf{x}^*) = 0$.

(R_n) **Rosenbrock function**
Definition: $R_n(\mathbf{x}) = \sum_{i=1}^{n-1}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$.
Search space: $-5 \leqslant x_i \leqslant 10$, $i = 1,2,\ldots,n$.
Global minimum: $\mathbf{x}^* = (1,\ldots,1)$,$R_n(\mathbf{x}^*) = 0$.

(RC) **Branin RCOS function**
Definition: $RC(\mathbf{x}) = (x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$.
Search space: $-5 \leqslant x_1 \leqslant 10, 0 \leqslant x_2 \leqslant 15$.
Global minima: $\mathbf{x}^* = (-\pi,12.275),(\pi,2.275),(9.42478,2.475)$; $RC(\mathbf{x}^*) = 0.397887$.

(RT_n) **Rastrigin function**
Definition: $RT_n(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$.
Search space: $-2.56 \leqslant x_i \leqslant 5.12$, $i = 1,\ldots,n$.
Global minimum: $\mathbf{x}^* = (0,\ldots,0)$, $RT_n(\mathbf{x}^*) = 0$.

($S_{4,m}$) **Shekel function**
Definition: $S_{4,m}(\mathbf{x}) = -\sum_{j=1}^{m}\left[\sum_{i=1}^{4}(x_i - C_{ij})^2 + \beta_j\right]^{-1}$, $\beta = \frac{1}{10}[1,2,2,4,4,6,3,7,5,5]^{\mathrm{T}}$,

$$C = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix}.$$

Search space: $0 \leqslant x_i \leqslant 10$, $i = 1,\ldots,4$.
Global minima: $\mathbf{x}^* = (4,4,4,4)$; $S_{4,5}(\mathbf{x}^*) = -10.1532$, $S_{4,7}(\mathbf{x}^*) = -10.4029$ and $S_{4,10}(\mathbf{x}^*) = -10.5364$.

(SC_n) **Schwefel function**
Definition: $SC_n(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n}\left(x_i \sin\sqrt{|x_i|}\right)$.
Search space: $-500 \leqslant x_i \leqslant 500$, $i = 1,2,\ldots,n$.
Global minimum: $\mathbf{x}^* = (1,\ldots,1)$, $SC_n(\mathbf{x}^*) = 0$.

(SH) **Shubert function**
Definition: $SH(\mathbf{x}) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$,
Search space: $-10 \leqslant x_i \leqslant 10$, $i = 1,2$
Global minima: 18 global minima and $SH(\mathbf{x}^*) = -186.7309$.

(SR_n) **Sphere function**
Definition: $SR_n(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$,
Search space: $-2.56 \leqslant x_i \leqslant 5.12$, $i = 1,\ldots,n$
Global minimum: $\mathbf{x}^* = (0,\ldots,0)$, $SR_n(\mathbf{x}^*) = 0$.

(SS_n) **Sum squares function**
Definition: $SS_n(\mathbf{x}) = \sum_{i=1}^{n} ix_i^2$,
Search space: $-5 \leqslant x_i \leqslant 10$, $i = 1,\ldots,n$
Global minimum: $\mathbf{x}^* = (0,\ldots,0)$, $SS_n(\mathbf{x}^*) = 0$.

(T_n) **Trid function**
Definition: $T_n(\mathbf{x}) = \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$,
Search space: $-n^2 \leqslant x_i \leqslant n^2$, $i = 1,\ldots,n$
Global minima: (a) $n = 6$, $x_i^* = i(7 - i)$, $i = 1,\ldots,n$, $T_n(\mathbf{x}^*) = -50$, (b) $n = 10$, $x_i^* = i(11 - i)$, $i = 1,\ldots,n$, $T_n(\mathbf{x}^*) = -210$,

(Z_n) **Zakharov function**
Definition: $Z_n(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$.
Search space: $-5 \leqslant x_i \leqslant 10$, $i = 1,2,\ldots,n$
Global minimum: $\mathbf{x}^* = (0,\ldots,0)$, $Z_n(\mathbf{x}^*) = 0$.

References

[1] K.S. Al-Sultan, M.A. Al-Fawzan, A tabu search Hooke and Jeeves algorithm for unconstrained optimization, European Journal of Operational Research 103 (1997) 198–208.

[2] R. Battiti, G. Tecchiolli, The continuous reactive tabu search: Blending combinatorial optimization and stochastic search for global optimization, Annals of Operations Research 63 (1996) 153–188.

[3] R. Chelouah, P. Siarry, Tabu Search applied to global optimization, European Journal of Operational Research 123 (2000) 256–270.

[4] D. Cvijovic, J. Klinowski, Taboo search: An approach to the multiple minima problem, Science 667 (1995) 664–666.

[5] D. Cvijovic, J. Klinowski, Taboo search: An approach to the multiple-minima problem for continuous functions, in: P.M. Pardalos, H.E. Romeijn (Eds.), Handbook of Global Optimization, Kluwer Academic Publishers, Boston, MA, 2002, pp. 387–406.

[6] C.A. Floudas, P.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer, C.A. Schweiger (Eds.), Handbook of Test Problems for Local and Global Optimization, Kluwer Academic Publishers, Boston, MA, 1999.

[7] F. Franze, N. Speciale, A tabu-search-based algorithm for continuous multiminima problems, International Journal for Numerical Engineering 50 (2001) 665–680.

[8] F. Glover, Tabu Search—Part I, ORSA Journal on Computing 1 (1989) 190–206.

[9] F. Glover, Tabu Search—Part II, ORSA Journal on Computing 2 (1990) 4–32.

[10] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, MA, USA, 1997.

[11] F. Glover, E. Taillard, D. Werra, A user's guide to Tabu Search, Annals of Operations Research 41 (1993) 3–28.

[12] A. Hedar, M. Fukushima, Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization, Optimization Methods and Software 17 (2002) 891–912.

[13] A. Hedar, M. Fukushima, Minimizing multimodal functions by simplex coding genetic algorithm, Optimization Methods and Software 18 (2003) 265–282.

[14] A. Hedar, M. Fukushima, Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, Optimization Methods and Software 19 (2004) 291–308.

[15] N. Hu, Tabu search method with random moves for globally optimal design, International Journal for Numerical Engineering 35 (1992) 1055–1070.

[16] C.T. Kelley, Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition, SIAM Journal on Optimization 10 (1999) 43–55.

[17] C.T. Kelley, Iterative Methods for Optimization, Frontiers Appl. Math. 18, SIAM, Philadelphia, PA, 1999.

[18] T.G. Kolda, R.M. Lewis, V. Torczon, Optimization by direct search: New perspectives on some classical and modern methods, SIAM Review 45 (2003) 385–482.

[19] M. Laguna, R. Martí, Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions, Technical Report, University of Colorado at Boulder, 2002.

[20] M. Laguna, R. Martí, V. Campos, Intensification and Diversification with elite tabu search solutions for the linear ordering problem, Computers and Operations Research 26 (1999) 1217–1230.

[21] R. Martí, Multi-start methods, in: F. Glover, G. Kochenberger (Eds.), Handbook of MetaHeuristics, Kluwer Academic Publishers, Boston, MA, 2002, pp. 355–368.

[22] R. Martí, J.M. Moreno, Métodos multi-arranque, Inteligencia Artificial 19 (2003) 49–60.

[23] Z. Michalewicz, G. Nazhiyath, Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints, in: Proceedings of the second IEEE ICEC, Perth, Australia, 1995.

[24] J.A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (1965) 308–313.

[25] P.M. Pardalos, M.G.C. Resende (Eds.), Handbook of Applied Optimization, Oxford University Press, Oxford, 2002.

[26] C.C. Ribeiro, P. Hansen (Eds.), Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, Boston, MA, 2002.

[27] F. Schoen, Two phase methods for global optimization, in: P.M. Pardalos, H.E. Romeijn (Eds.), Handbook of Global Optimization, Kluwer Academic Publishers, Boston, MA, 2002, pp. 151–178.