Tabu Search directed by direct search methods for nonlinear global optimization*

Abdel-Rahman Hedar and Masao Fukushima

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

June 21, 2003

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 deal 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 direction.

1 Introduction

Tabu Search (TS) is one of the recent meta-heuristics initially developed for combinatorial optimization problems [10,11]. TS fights courageously when applied to these problems [10,20]. 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 approach that deals with continuous global optimization problems. Specifically, we consider the unconstrained nonlinear minimization problem

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

*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.

where f is a real-valued function defined on R^n. Recently, there has been a growing interest in solving this problem using metaheuristics [19,20]. 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 [12,13,14]. 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 instead of using completely blind random search. 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 a relatively few attampts have been made to deal with continuous optimization problems [1,2,3,4,5,7,15]. One of the earliest TS methods was presented by Hu [15] for constrainted optimization problems. Cvijovic and Klinowski [4,5] extented 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 [18] 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. The numerical results 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 rule in TS, especially when it is used in a kind of learning process. Using an effective memory conception in intensification and diversification schemes makes TS behave as an intelligent search technique [10]. In this section, we introduce some new conceptions of the memory elements. 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 current iterate 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 $TL = \{t_i\}_{i=1}^{L}$. 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 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:

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

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 $0 \leq \eta_{\min} < \eta_{\max} \leq 1$. Specifically, the recency ranked value for each element of TL is given by

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

- *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 $1 \leq \overline{L} \leq L$ and $0 \leq \mu_{\min} < \mu_{\max} \leq 1$. 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 $r_{TR} > 0$. For each TR, we define Semi-TR to be the surrounding region around this TR with outer radius r_{STR} from its center, where $r_{STR} > r_{TR}$. 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 lies in ν Semi-TRs with centers $t_1, \ldots, t_\nu$. Compute the centroid $\overline{t}$ of the Semi-TRs' centers and the maximum distance $d_{\max}$ between x and these centers, i.e.,

$$\overline{t} = \frac{1}{\nu} \sum_{i=1}^{\nu} t_i, \tag{4}$$

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

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

Figure 1 illustrates how Procedure 2.1 works when a solution x lies in Semi-TRs in two dimensions. In this example, 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 $\left(x - \overline{t}\right)$, where $\overline{t}$ 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 $r_{TR} + \max\{\|x - t_1\|, \|x - t_2\|\}$, to make sure that the close TRs with centers t_1 and t_2 will not be hit.

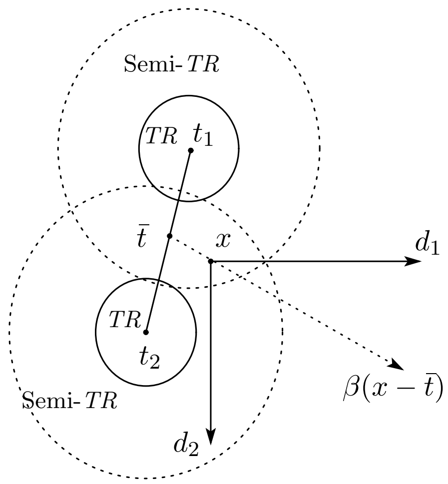

Figure 1: Neighborhood search from a point in Semi-*TRs*.

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., $VRL = \{(\zeta_i, \rho_i, \varphi_i)\}_{i=1}^{M}$, 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

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

where $\gamma \in (0, 1]$ is a given constant.

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 ones more frequently visited.

Procedure 2.2.

1. Generate a trial point x randomly in the search domain of f.

2. Compute the quantities $d_i = (1 + \Phi(\varphi_i))\,\|x - \zeta_i\|$, $i = 1, \ldots, M$, where $\Phi(\varphi)$ is defined by (6). If $\min_{1\le i\le M} d_i/\rho_i \ge 1$, then accept x. Otherwise, return to Step 1.

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 [18] 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 iterate x with suitable step sizes. If the current iterate 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 iterate 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., $q = 0$.

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:

$$v = \sum_{i=1}^{n} \omega_i u_i, \tag{7}$$

where

$$\begin{aligned}
\omega_i &= \frac{\Delta f_i}{\sum_{j=1}^{n} |\Delta f_j|}, \quad i = 1, 2, \ldots, n, \\
u_i &= -\frac{(y_i - x)}{\|y_i - x\|}, \quad i = 1, 2, \ldots, n, \\
\Delta f_i &= f(y_i) - f(x), \quad i = 1, 2, \ldots, n.
\end{aligned}$$

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 iterate 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.

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.

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 involves an inner 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. At the final stage, the Intensification Search procedure is needed to complete the DTS method. The 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.

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 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., $x_{k+1}^{(j)} := \arg\min_{i=1,\ldots,n+q} \{f(y_i)\}$. *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. We may expect that one of these best points is close to a global minimum. So we complete the DTS method by applying another local search method starting from some of these points, which we call Intensification Search. We use the 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_{NMS} and DTS_{APS} that use the NMS strategy and the APS strategy, respectively, as the neighborhood-local search strategy. First, a specific and formal description of DTS_{NMS} is given in the following Algorithms 4.1.

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

1. Initialization. Choose positive integers ℓ_{main}, ℓ'_{main}, ℓ_{inner} and ℓ'_{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 ℓ'_{main} consecutive main iterations fail to obtain improvement or the main loop iteration counter j exceeds ℓ_{main}.

2.1. Exploration Search(NMS) (*Inner Loop***).** Let $k := 0$ and repeat this inner loop until ℓ'_{inner} consecutive inner iterations fail to obtain improvement or the inner loop iteration counter k exceeds ℓ_{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 ith unit vector in R^n and τ_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 $S := \left\{x_k^{(j)}, y_1, \ldots, y_n\right\}$. 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 DTS$_{APS}$ 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 ℓ'_{inner} consecutive inner iterations fail to obtain improvement or the inner loop iteration counter k exceeds ℓ_{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 $d_i := sign(v_i)e_i$, $i = 1, \ldots, n$, where $e_i \in R^n$ is the ith unit vector in R^n and v_i is the ith componant 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 α_1 and α_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

The suggested values of the parameters needed in the implementation of the DTS algorithms are given in the following.

Initial trial. The initial point $x_0^{(0)}$ is chosen randomly from the predetermined range $[\mathfrak{L}, \mathfrak{U}]$ of the initial points for each test function, where

$$[\mathfrak{L}, \mathfrak{U}] := \{x \in R^n : l_i \leq x_i \leq u_i,\ i = 1, \ldots, n\}. \tag{8}$$

The diameter of the range $[\mathfrak{L}, \mathfrak{U}]$ is defined as $\delta := \max_{1 \leq i \leq n} (u_i - l_i)$.

***TR* and Semi-*TR* parameters.** The radius r_{TR} of each *TR* is set equal to $\delta/100$, and the outer radius r_{STR} of the Semi-*TR* equal to $2r_{TR}$.

***TL* parameters.** The number L of elements in *TL* is set equal to $5n$. The maximum and minimum recency ranked values $\eta_{\max}$ and $\eta_{\min}$ are set equal to 1 and $1/L$, respectively. The number $\overline{L}$ of the function-value ranked elements is set equal to $2n$. The maximum and minimum function-value ranked values $\mu_{\max}$ and $\mu_{\min}$ are set equal to 1 and $1/L$, respectively.

***VRL* parameters.** The radius of all visited regions is fixed and set equal to $\delta/5$, i.e. $\rho_i = \delta/5,\ i = 1, \ldots, M$.

NMS parameters. The step sizes are set at each iteration as $\Delta_i = \delta(1 + \chi_i)/10,\ i = 1, \ldots, n$, where χ_i is a random number from the interval $(0, 1)$. If one of these step sizes yeilds a neighborhood trial point that hits any *TR,* then this step size should be stretched to generate a spare trial point outside this hit *TR*.

APS parameters. The step sizes used to generate neighborhood trial points are chosen in the same way as in NMS. The step sizes α_1 and α_2 used to generate local trial points are set equal to $\sigma_1\delta/10$ and $\sigma_2\delta/10$, where σ_1 and σ_2 are random numbers from the intervals $(0, 1)$ and $(1, 2)$, respectively.

Diversification trials. The parameter γ used in (6) is set equal to 0.25.

Intensification trials. The number of best points that are used in the Intensification Search is set equal to 1.

Termination conditions. The loop repeation numbers are set as $\ell_{main} = 10n$, $\ell'_{main} = 2n$,

$\ell_{inner} = 3n$ and $\ell'_{inner} = n$. However, for Shubert function (SH) and Easom function (ES), which have many local minuma, ℓ_{main} is set equal to $20n$.

5.2 Numerical Results

To examine the performance of the DTS algorithms DTS$_{NMS}$ and DTS$_{APS}$, we tested them on some well known functions [6,13]. The characteristics of these test functions are diverse enough to cover many kinds of difficulties that face global optimization problems. To complete the evaluation of the DTS methods, 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 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, we applied the DTS codes 100 times with different starting points. We used the same condition as that used by ECTS [3] to judge the success of a trial, which is given by

$$\left| f^* - \widetilde{f} \right| < \epsilon_1 \left| f^* \right| + \epsilon_2, \tag{9}$$

where $\widetilde{f}$ refers to the best function value obtained by the method, 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, DTS$_{NMS}$ and DTS$_{APS}$, are reported in Table 1. These results represent the average number of function evaluations (Av. f-evals.), the average errors (Av. Error) and the success rates (Suc.) for each function. The average number of function evaluations and the average error are related to only successful trials. The results shown in Table 1 reveal that the performance of DTS$_{APS}$ is consistently better than DTS$_{NMS}$ in terms of function evaluations and the ability of obtaining global minima. Moreover, it seems that DTS$_{NMS}$ suffers from the curse of dimensionality as it is clear from the Av. f-evals. for higher dimensional problems.

Table 2 compares 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 denote the success rates of reaching global minima. The quality of computed solutions by those methods except the CRTS method is shown in Table 3, 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.

Table 1: Results of DTS algorithms.

	DTS$_{NMS}$			DTS$_{APS}$		
f	*Av. f-evals.*	*Av. Er.*	*Suc.*	*Av. f evals.*	*Av. Er.*	*Suc.*
RC	406	4e–7	100%	243	4e–7	100%
ES	743	4e–9	69%	412	5e–9	90%
GP	423	6e–9	94%	263	5e–9	100%
SH	767	9e–6	57%	437	9e–6	91%
Z_2	404	8e–9	100%	236	1e–8	100%
R_2	485	6e–9	100%	290	7e–9	100%
DJ	917	6e–9	100%	496	5e–9	100%
$H_{3,4}$	940	2e–6	99%	494	2e–6	100%
$S_{4,5}$	2025	3e–7	41%	858	3e–7	60%
$S_{4,7}$	2025	4e–5	31%	856	4e–5	59%
$S_{4,10}$	2031	1e–5	27%	854	1e–5	45%
Z_5	3426	8e–9	100%	1178	6e–9	100%
R_5	3806	7e–9	74%	1577	8e–9	89%
$H_{6,4}$	4915	2e–6	62%	1666	2e–6	83%
Z_{10}	19978	2e–8	100%	4587	2e–8	100%
R_{10}	21153	2e–8	87%	6856	2e–8	90%

- The ECTS method uses condition (9) to test the success of a trial [3]. However, the results marked by ($^{\circledast}$) in Tables 2 and 3 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 3, they are reported to be greater than 10^{-6}. For instance, the average error for function R_5 in Table 3 is 0.08, which means 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 2 and 3 also contain similar inconsistencies.

- According to [7], DOPE is terminated either when a maximum number of function evaluations is reached or when the global minimum (if it is a priori known) is found. Since using the information on global minima does not seem applicable in practice, we did not use the latter termination condition in our numerical experiments. This termination condition may explain the extremely small number of function evaluations of DOPE for some test functions.

Considering these remarks, the comparisons of the DTS method with the ECTS and DOPE methods do not seem to yield a definitive fair answer. As far as the results in Table 2 common to all methods are concerned, CRTS may be considered to be the best among the continuous TS methods in terms of the ability of obtaining global minima and the number of function evaluations. In term of the quality of computed solutions, the DTS method seems to outperform ECTS and DOPE as shown in Table 3.

Table 2: Average numbers of function evaluations for continuous TS methods.

f	DTS$_{APS}$	ECTS	DOPE	CRTS	
				CRTS$_{Ave}$	CRTS$_{Min}$
RC	243	245$^{\circledast}$	31	38	41
ES	412(90%)	1284$^{\circledast}$	290	–	–
GP	263	231$^{\circledast}$	248	248	171
SH	437(91%)	370	466	–	–
Z_2	236	195	81	–	–
R_2	290	480$^{\circledast}$	692	–	–
DJ	496	338	131	–	–
$H_{3,4}$	494	548$^{\circledast}$	135	513	609
$S_{4,5}$	858(60%)	825(75%)	–	812	664
$S_{4,7}$	856(59%)	910(80%)	–	960	871
$S_{4,10}$	854(45%)	898(75%)	–	921	693
Z_5	1178	2254$^{\circledast}$	424	–	–
R_5	1577(89%)	2142$^{\circledast}$	2512	–	–
$H_{6,4}$	1666(83%)	1520$^{\circledast}$	421	750	1245
Z_{10}	4587	4630	8695	–	–
R_{10}	6856(90%)	15720(85%)	5133	–	–

Table 3: 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	5e–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	6e–9	4e–6$^{\circledast}$	4e–6
Z_2	1e–8	2e–7	2e–7	R_5	8e–9	8e–2$^{\circledast}$	8e–2
R_2	7e–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

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.

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 and 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, to appear.

[14] A. Hedar, M. Fukushima, Heuristic Pattern Search and Its Hybridization with Simulated Annealing for Nonlinear Global Optimization, Technical Report 2003-001, Department of Applied Mathematics and Physics, Kyoto University, 2003.

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

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

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