Hindawi
Journal of Mathematics
Volume 2024, Article ID 1947996, 12 pages
https://doi.org/10.1155/2024/1947996

Research Article

A Second-Order Finite-Difference Method for Derivative-Free Optimization

Qian Chen,[1] Peng Wang ,[1,2] and Detong Zhu[3]

[1]*Mathematics and Statistics College, Hainan Normal University, Haikou 570203, Hainan, China*
[2]*Key Laboratory of the Ministry of Education, Hainan Normal University, Haikou 570203, Hainan, China*
[3]*Mathematics and Science College, Shanghai Normal University, Shanghai 200234, China*

Correspondence should be addressed to Peng Wang; pengwang621@163.com

Received 30 July 2023; Revised 28 February 2024; Accepted 1 March 2024; Published 15 March 2024

Academic Editor: Nan-Jing Huang

In this paper, a second-order finite-difference method is proposed for finding the second-order stationary point of derivative-free nonconvex unconstrained optimization problems. The forward-difference or the central-difference technique is used to approximate the gradient and Hessian matrix of objective function, respectively. The traditional trust-region framework is used, and we minimize the approximation trust region subproblem to obtain the search direction. The global convergence of the algorithm is given without the fully quadratic assumption. Numerical results show the effectiveness of the algorithm using the forward-difference and central-difference approximations.

1. Introduction

1.1. Problem Description Motivation. In this paper, we propose a trust region method for solving the derivative-free nonconvex optimization problems of the following:

$$(P)\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f(x)$: $\mathbb{R}^n \longrightarrow \mathbb{R}$ is a general nonlinear nonconvex twice Lipschitz continuously differentiable function, but none of its first-order or second-order derivative is explicitly available. The problem (P) arises frequently in computational science, engineering, and industry.

The problem (P) was widely applied in machine learning, but the first- and second-order derivatives were not available in computational science and engineering. Nesterov and Polyak [1] introduced the cubic regularization algorithm generating the iteration point converging to the second-order stationary point of smooth optimization problems. The adaptive regularization cubic method was introduced by Cartis, Gould, and Toint [2, 3]. They built a local cubic model and solved the model for obtaining an approximation solution in each iteration. The complexity rates are also given in this paper. Furthermore, Curtis, Robinson, and Samadi [4] proved that the worst-case complexity of the adaptive trust-region algorithm was $O(\varepsilon^{-3/2})$ and $O(\max\{\varepsilon_g^{-3/2}, \varepsilon_H^{-3}\})$ when it achieved the ε-first-order stationary point and the $(\varepsilon_g, \varepsilon_H)$-second-order stationary point, respectively. They also introduced a new updating rule for obtaining the trust-region radius. Curtis and Robinson [5] proposed a new algorithm which obtained the search direction and the step size for the dynamic choice technique. The algorithm chose a better direction from first-order and second-order descent directions such that the objective function has more significant reduction. Lee, Simchowitz, Jordan, and Recht [6] also proposed the vanilla gradient descent method for finding the strict saddle points with the probability one. Du, Jin, Lee, Jordan, Singh, and Poczos [7] proved this method converged to the second-order stationary point in an exponential number of steps.

In recent years, some authors study the finite-difference methods for solving the nonsmooth optimization, smooth stochastic convex optimization, and noisy derivative-free optimization. Auslender and Teboulle [8] identified a simple line search mechanism using the interior gradient and

proximal methods such that the method has the global convergence results. They also showed that the rate of the convergence of the algorithm is $O(k^{-2})$. Tseng and Yun [9] introduced the block coordinate gradient descent method for solving the smooth and separable convex functions combinatorial problem. The method generated a search direction by block coordinate descent and ensured sufficient descent and reiterated by the inexact line search. The local Lipschitzian error bound assumption was used for proving global and linear convergence of the algorithm. Auslender and Teboulle [10] solved the derivative-free convex optimization problems by the subgradient projection method. The non-Euclidean projection-like maps were proposed for obtaining the interior trajectories which relied on a single projection. Villa, Salzo, Baldassarre, and Verri [11] proposed an accelerated forward-backward splitting method for solving the composite optimization problem. Under a sufficiently fast decay condition, they proved that the method achieved the $1/k^2$ convergence rate. Furthermore, they accounted the cost of the algorithm to achieve an approximate optimal solution and gave global complexity analysis of the method. Keskar and Wachter [12] proposed the limited-memory quasi-Newton algorithm which contained the L-BFGS quasi-Newton approximation and the weak Wolfe line search technique. For avoiding the shortsightedness which the gradient was not obtained for a nonsmooth function, the ε-minimum norm subgradient was introduced to obtain the search directions in iterative corrective loop. Lewis and Overton [13] introduced a quasi-Newton algorithm to obtain the optimal solution of nonsmooth optimization problems. The analysis of the case of the Euclidean norm was given, which used the inexact line search in one variable and assumed that the line search was exact in two variables. The Clarke stationary point of the problem was found by the BFGS method with the inexact line search. At the same time, they also showed that the algorithm has R-linear convergence.

Berahas et al. [14] minimized noisy functions by presenting the finite-difference quasi-Newton method. The noise estimation techniques were given by More and Wild [15]. They presented a new EC noise algorithm for solving noise-level calculation functions. The convergence analysis of the algorithm was based on stochastic noise without the assumption of specific distribution for the noise. They also used the search direction to update the finite parameter when the line search technique was invalid for producing an acceptable point such that the method converged to the optimal solution faster. Hamming [16] used BFGS update for obtaining differencing intervals. Brekelmans et al. [17] analyzed different gradient estimate methods for noisy functions. Through estimating the different gradient of error criterion, they converted the total error into the sum of deterministic error and stochastic error. They derived optimal step sizes for deterministic errors and stochastic errors, respectively, such that the total error was minimized. Beraha et al. [18] analyzed finite differences, linear interpolation, Gaussian smoothing, and smoothing on a unit sphere which only used the function values for approximating the gradient of the objective function.

Nesterov and Spokoiny [19] proposed the method in which the search directions were normally distributed random Gaussian vectors. They proved new complexity bounds of the methods based on computation of the function value. They also proved the conclusion that the methods need at most n times iterations, where n was the dimension of problem. An accelerated scheme with the expected rate and a zero-order scheme with the expected rate for stochastic optimization were given. Gorbunov et al. [20] proposed an accelerated derivative-free algorithm for unconstrained smooth convex functions with noisy. The noise consisted of the combination of stochastic nature and unknown nature cases with absolute value bounded constraints. They also proposed a nonaccelerated derivative-free method similar to the stochastic-gradient-based method and proved an l_1-norm proximal setup has better complexity bound than the Euclidean proximal setup. Bellavia et al. [21] presented evaluation complexity bounds in the framework of a deterministic trust-region method. They also showed that the presence of intrinsic noise might dominate the bound and provided estimates of the optimality level achievable, should noise cause early termination. Finally, they shed some light on the impact of inexact computer arithmetic on evaluation complexity.

1.2. Contribution of This Paper. In order to solve the derivative-free nonconvex unconstrained optimization problem (P) in which the first-order or second-order derivative cannot be explicitly available, the new trust-region method is introduced in this paper. The main contributions of this paper are as follows:

(i) The finite-difference trust-region method is proposed for finding second-order stationary points of derivative-free nonconvex optimization. We prove that the method globally converges to a second-order stationary point of the derivative-free nonconvex optimization problem (P).

(ii) The proposed algorithm uses the forward-difference or central-difference techniques approximating the gradients and Hessian matrix of objective function to build the trust-region subproblem. Hence, the trust-region subproblem of the algorithm is more accurate than the BFGS update. The algorithm can obtain better search directions.

(iii) The new finite-difference parameter update technique ensures that the algorithm generates a subsequence converging to the second-order stationary point of original nonconvex optimization problem (P) without the fully quadratic approximation.

This paper is organized into 6 sections: The finite-difference trust-region subproblems and the definition of second-order stationary points are introduced in Section 2. The finite difference trust-region algorithm is introduced in Section 3. We prove that there is a sequence generated by the algorithm converging to the second-order stationary point of problem (P) under the bounded approximation of the level set in Section 4. Numerical experiments illustrating the

practical performance of the algorithm are reported in Section 5. The final conclusion of this paper is given in Section 6.

1.3. Notation. In this paper, the derivative-free nonconvex unconstrained optimization problem are solving by the trust-region algorithm with forward-difference or central-difference technique. $x \in \mathbb{R}^n$ denotes the variables of problem (P), and x^* is the optimal solution of objective function $f(x)$. $\|\cdot\|$ denotes the Euclidean norm. $\nabla h(x_k)$ and $\nabla^2 h(x_k)$ denote the gradient and the Hessian matrix of any scalar function $h(x)$ on iteration point x_k, respectively.

2. The Trust-Region Subproblem and the Definition of Second-Order Stationary Point

In this paper, we propose the finite-difference trust-region method solving derivative-free nonconvex optimization problem (P). The trust-region subproblem needs the information of gradient and the Hessian matrix of objective function $f(x)$. But this information is not obtained. Hence, we will use the finite-difference method approximating this information. The finite-difference method relies on the computation of the finite-difference parameter t.

We applied the definition of the finite-difference method given in [15]. We define the i th component of the forward-difference approximate at x as follows:

$$[g(x)]_i = \frac{f\left(x + t_{\mathrm{FD}}^1 e_i\right) - f(x)}{t_{\mathrm{FD}}^1}, \tag{2}$$

where $e_i, i \in \{1, 2, \ldots, n\}$ is the n-dimensional unit vector in which i th element is 1.

The central-difference approximate is given as

$$[g(x)]_i = \frac{f\left(x + t_{\mathrm{CD}}^1 e_i\right) - f\left(x - t_{\mathrm{CD}}^1 e_i\right)}{2t_{\mathrm{CD}}^1}. \tag{3}$$

In the general finite-difference algorithm, the approximation of the Hessian matrix does not use the information of the original objective function of problem (P), so the calculation error is large. In order to reduce the error between the symmetric matrix of the trust-region subproblem and the Hessian matrix of the original problem, similar to estimating the gradient of $f(x)$, the second-order forward-difference technique by the objective function is expressed as follows:

$$[H(x)]_{ij} = \frac{\left(f\left(x + t_{\mathrm{FD}}^1 e_i + t_{\mathrm{FD}}^2 e_j\right) - f\left(x + t_{\mathrm{FD}}^2 e_j\right)\right) - \left(f\left(x + t_{\mathrm{FD}}^1 e_i\right) - f(x)\right)}{t_{\mathrm{FD}}^1 t_{\mathrm{FD}}^2}, \tag{4}$$

and the second-order central difference approximate is given by

$$[H(x)]_{ij} = \frac{\left(f\left(x + t_{\mathrm{CD}}^1 e_i + t_{\mathrm{CD}}^2 e_j\right) - f\left(x - t_{\mathrm{CD}}^1 e_i + t_{\mathrm{CD}}^2 e_j\right)\right) - \left(f\left(x + t_{\mathrm{CD}}^1 e_i - t_{\mathrm{CD}}^2 e_j\right) - f\left(x - t_{\mathrm{CD}}^1 e_i - t_{\mathrm{CD}}^2 e_j\right)\right)}{4t_{\mathrm{CD}}^1 t_{\mathrm{CD}}^2}, \tag{5}$$

where $[H(x)]_{ij}$ is the element in row i and column j of matrix $H(x)$ and $e_i, i \in \{1, 2, \ldots, n\}$ and $e_j, j \in \{1, 2, \ldots, n\}$ are n-dimensional unit vectors whose i th and j th elements are 1, respectively.

For obtaining the second-order stationary point of problem (P), we will give the approximation trust-region subproblem of problem (P) as follows:

$$\begin{aligned} &\min f(x_k) + g_k^T d + \frac{1}{2} d^T \left(H_k + 2\left[\lambda_{\min}(H_k)\right]_+ I\right) d \\ &\text{s.t. } \|d\| \le \Delta_k, \end{aligned} \tag{6}$$

where $g_k = g(x_k)$ and $H_k = H(x_k)$, $g(x)$ generate by (2) or (3) and $H(x)$ generate by (4) or (5). $\lambda_{\min}(H_k)$ is the minimum eigenvalue of symmetric matrix H_k and $[\lambda_{\min}(H_k)]_+ = \max\{-\lambda_{\min}(H_k), 0\}$.

Before introducing the finite-difference trust-region algorithm, we introduce the definition of the second-order stationary point of problem (P).

Definition 1. For derivative-free nonconvex optimization problem (P), if there exists a point $x^* \in \mathbb{R}^n$, for all vector $d \in \mathbb{R}^n$, such that

$$\begin{aligned} \nabla f(x^*)^T d &\ge 0, \\ d^T \nabla^2 f(x^*) d &\ge 0, \end{aligned} \tag{7}$$

then x^* is said to be second-order stationary point.

According to Definition 1, if let $\hat{\sigma} = \max\{\|\nabla f(x^*)\|, -\lambda_{\min}(\nabla^2 f(x^*))\} = 0$, then it is obviously that x^* is the second-order stationary point of problem (P) according to Definition 1.

3. Algorithm

In this section, a trust-region method is introduced for finding the approximation of the second-order stationary point of problem (P).

4. Convergence Analysis

In this section, we will provide the global convergence analysis of Algorithm 1; hence, we require the model to satisfy the following assumption.

Assumption 2. We define the level set as follows:

$$L(x_0) = \{x \in \mathbf{R}^n, f(x) \le f(x_0)\}. \tag{8}$$

Suppose that the level set is bounded.

As an important result, the following lemma shows the boundedness of oneself, the gradient, and the Hessian matrix of the objective function.

Lemma 3. *Under Assumption 2, for $\forall x \in L(x_0)$, there exist positive constants κ_f, κ_g and κ_{H} such that*

$$\begin{aligned} |f(x)| &\le \kappa_f, \\ \|\nabla f(x)\| &\le \kappa_g, \\ \|\nabla^2 f(x)\| &\le \kappa_H. \end{aligned} \tag{9}$$

Proof. Similar proof can be found in Lemma 3.2 of [22].

The boundedness of g_k and H_k is important for the convergence of Algorithm 1; hence, we will prove the boundedness of the approximation gradient g_k of objective function and the approximation Hessian matrix H_k of objective function on iteration point x_k in the following lemma. □

Lemma 4. *Under Assumption 2, there exist $\hat{\kappa}_g > 0$ and $\hat{\kappa}_H > 0$ such that*

$$\begin{aligned} \|g_k\| &\le \hat{\kappa}_g, \\ \|H_k\| &\le \hat{\kappa}_H. \end{aligned} \tag{10}$$

Proof. Similar proof can be found in [23].

The following lemma shows the error is bounded between $\nabla f(x_k)$ and the finite-difference gradient g_k and the error is bounded between the finite-difference Hessian matrix of the objective function on the iteration point x_k generated by Algorithm 1. □

Lemma 5. *Under Assumption 2, for any iteration point x_k generated by Algorithm 1, we have that*

$$\|\nabla f(x_k) - g_k\| \le \frac{n}{2}\kappa_H t_k^1, \tag{11}$$

$$\|\nabla^2 f(x_k) - H_k\| \le \frac{n^2 L_H}{2}\left(3t_k^1 + t_k^2\right), \tag{12}$$

where n denotes the dimension of problem (P) and L_H is the Lipchitz constant of $\nabla^2 f(x)$.

Proof. Similar proof can be found in [23].

According to Lemma 5 and the method of selecting t in step 1 of Algorithm 1, we have

$$\begin{aligned} \|\nabla f(x_k) - g_k\| &\le \frac{n}{2}\kappa_H \Delta_k^2, \\ \|\nabla^2 f(x_k) - H_k\| &\le \frac{n^2 L_H}{2}(3\nabla_{\max} + 1)\Delta_k. \end{aligned} \tag{13}$$

In order to avoid the first-order and second-order descent assumptions in [24], we demonstrate that the search direction generated by our Algorithm 1 satisfies first-order and second-order descent. □

Lemma 6. *Under Assumption 2, there exists constant $\kappa_m \in (0,1/2)$ such that*

$$m_k(0) - m_k(d_k) \ge \kappa_m \max\left\{\|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{3\hat{\kappa}_H}\right\}, -\tau_k \Delta_k^2\right\}, \tag{14}$$

where $\tau_k = \lambda_{\min}(H_k)$ is the minimum eigenvalue of matrix $\mathrm{H_k}$.

Proof. If d_k is obtained by (5), then

$$\begin{aligned} m_k(0) - m_k(d_k) &\ge \kappa_m \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|H_k + 2[\lambda_{\min}(H_k)]_+ I\|}\right\} \\ &\ge \kappa_m \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|H_k\| + 2\|[\lambda_{\min}(H_k)]_+ I\|}\right\} \\ &= \kappa_m \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|H_k\| + 2|[\lambda_{\min}(H_k)]_+|}\right\}. \end{aligned} \tag{15}$$

The proof of (15) can be found in [25].

Because $|[\lambda_{\min}(H_k)]_+| = \max\{-\lambda_{\min}(H_k), 0\} \le \|H_k\| \le \hat{\kappa}_H$, by (16), we have that

$$m_k(0) - m_k(d_k) \ge \kappa_m \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{3\hat{\kappa}_H}\right\}. \tag{16}$$

If d_k is obtained by (5), then we have that there exists $\mu \ge 0$ such that

$$\begin{aligned} (H_k + 2(-\tau_k)I)d_k + \mu d_k &= -g_k, \\ \mu\left(\|d_k\| - \Delta_k\right) &= 0, \\ \|d_k\| - \Delta_k &= 0, \end{aligned} \tag{17}$$

hence, we can obtain that

$$\begin{aligned} -g_k^T d_k - \frac{1}{2}d_k^T(H_k - 2\tau_k I)d_k &= \frac{1}{2}d_k^T(H_k - 2\tau_k I)d_k + \mu\|d_k\|^2 \\ &\ge \frac{1}{2}d_k^T(H_k - 2\tau_k I)d_k \ge \frac{1}{2}(-\tau_k)\|d_k\|^2 \ge \kappa_m(-\tau_k)\|d_k\|^2 = \kappa_m(-\tau_k)\Delta_k^2. \end{aligned} \tag{18}$$

Input: Given initial iteration $x_0 \in \mathbb{R}^n$, the constants $\Delta_{\max} > \Delta_0 > 0$, $\varepsilon > 0$, $1 > \beta > 0$, $\gamma_2 > 1 > \gamma_1 > 0$ and $\eta_2 > \eta_1 > 0$. Set $k = 0$.
Main Step:

(1) Choose $t_k^1 \in (0, \Delta_k^2)$, calculate the approximate gradient g_k using (3) or (2). Choose $t_k^2 \in (0, \Delta_k)$, calculate the approximate Hessian matrix H_k using (5) or (4) and calculate $\sigma_m(x_k) = \max\{\|g_k\|, -\lambda_{\min}(H_k)\}$, where $\lambda_{\min}(H_k)$ is the minimum eigenvalue of symmetric matrix H_k.

(2) If $\|\sigma_m(x_k)\| \le \varepsilon$, then let $\Delta_k = \beta\Delta_k$. If $\Delta_k \le \varepsilon$, then stop, x_k is the second-order stationary point, else go to 1.

(3) If $\|\sigma_m(x_k)\| > \varepsilon$, then calculate the following subproblem,

$$\min m_k(d) = f(x_k) + g_k^T d + (1/2)d^T(H_k + 2[\lambda_{\min}(H_k)]_+ I)d$$
$$\text{s.t. } \|d\| \le \Delta_k$$

where $[\lambda_{\min}(H_k)]_+ = \max\{-\lambda_{\min}(H_k), 0\}$ and obtain the search direction d_k.

(4) Calculate

$$\rho_k = f(x_k) - f(x_k + d_k)/m_k(0) - m_k(d_k).$$

(5) If $\rho_k \ge \eta_1$ then $x_{k+1} = x_k + d_k$, else let $x_k = x_k$.

(6) Update the trust-region radius as follows:

$$\Delta_k = \begin{cases} \gamma_1\Delta_k & \text{if } \rho_k < \eta_1 \\ \Delta_k & \text{if } \eta_1 \le \rho_k < \eta_2 \\ \min\{\gamma_2\Delta_k, \Delta_{\max}\} & \text{if } \rho_k \ge \eta_2 \end{cases}$$

Let $k = k + 1$, go to 1.

Algorithm 1: Finite-difference algorithm.

Combining (16) and (18), we have that

$$m_k(0) - m_k(d_k) \ge \kappa_m \max\left\{\|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{3\hat{\kappa}_H}\right\}, -\tau_k\Delta_k^2\right\}. \tag{19}$$

In order to establish the global convergence of Algorithm 1, we first introduce the following notations,

$$\begin{aligned} \sigma_m(x) &= \max\{\|g(x)\|, \max\{-\lambda_{\min}(H(x)), 0\}\}, \\ \sigma(x) &= \max\left\{\|\nabla f(x)\|, \max\left\{-\lambda_{\min}\left(\nabla^2 f(x)\right), 0\right\}\right\}. \end{aligned} \tag{20}$$

The following lemma shows that the error is bounded between the smallest eigenvalues of Hessian matrix of objective function and a corresponding finite-difference model. □

Lemma 7. *Under Assumption 2, if H_k is generated by (4) or (5), then*

$$\left|\lambda_{\min}\left(\nabla^2 f(x_k)\right) - \lambda_{\min}(H_k)\right| \le \frac{n^2 L_H}{2}(3\Delta_{\max} + 1)\Delta_k. \tag{21}$$

Proof. Let s be a normalized eigenvector of smallest eigenvalue of matrix $(\nabla^2 f(x_k) - H_k)$, then

$$\begin{aligned} &\left|\lambda_{\min}\left(\nabla^2 f(x_k)\right) - \lambda_{\min}(H_k)\right| \\ &\le \left|s^T\left[\nabla^2 f(x_k) - H_k\right]s\right| \\ &\le \sum_{i=1}^{n}\left|\lambda_i\left(\nabla^2 f(x_k) - \lambda_i(H_k)\right)\right| \\ &\le \sum_{i=1}^{n}\left|\lambda_i\left(\nabla^2 f(x_k) - H_k\right)\right| \\ &\le n\left\|\nabla^2 f(x_k) - H_k\right\| \\ &\le \frac{n^3 L_H}{2}\left(3t_k^1 + t_k^2\right). \end{aligned} \tag{22}$$

And

$$\left|\lambda_{\min}(H_k) - \lambda_{\min}\left(\nabla^2 f(x_k)\right)\right| \le \frac{n^3 L_H}{2}\left(3t_k^1 + t_k^2\right), \tag{23}$$

then the result follows.

According to step 1 of Algorithm 1, we have that

$$\left|\lambda_{\min}(H_k) - \lambda_{\min}\left(\nabla^2(x_k)\right)\right| \le \frac{n^3 L_H}{2}(3\Delta_{\max} + 1)\Delta_k. \tag{24}$$

After the error is bounded between $\lambda_{\min}(\nabla^2 f(x_k))$ and $\lambda_{\min}(H_k)$, the error boundedness of $\sigma(x_k)$ and $\sigma_m(x_k)$ is given in the following lemma. □

Lemma 8. *Under Assumption 2, there exists $\kappa_\sigma > 0$ such that*

$$\left|\sigma(x_k) - \sigma_m(x_k)\right| \le \kappa_\sigma \Delta_k. \tag{25}$$

Proof. Similar proof can be found in Lemma 7.2 of [24].

We now show that an iteration must be successful for sufficiently small trust-region radius with respect to $\sigma_m(x_k)$. □

Lemma 9. *If $g(x)$ is generated by (2) or (3) and $H(x)$ is generated by (4) or (5) and*

$$\varepsilon_1 \left[\lambda_{\min}(H_k)\right]_+ \le \Delta_k \le \min\left[\frac{1}{3\hat{\kappa}_H}, \frac{\kappa_m(1-\eta_1)}{(n\kappa_H/2 + \kappa_\sigma + L_H)\Delta_{\max}}, \frac{\kappa_m(1-\eta_1)}{(n\kappa_H/2 + \kappa_\sigma + L_H)}\right]\sigma_m(x_k), \tag{26}$$

where $\varepsilon_1 \ll \min[1/3\hat{\kappa}_H, \kappa_m(1-\eta_1)/(n\kappa_H/2 + \kappa_\sigma + L_H)\,\Delta_{\max}, \kappa_m(1-\eta_1)/(n\kappa_H/2 + \kappa_\sigma + L_H)]$, then x_k is a successful iteration.

Proof. First, according to the fractions of Cauchy and eigenstep decrease,

$$m_k(0) - m_k(d_k) \ge \kappa_m \max\left\{\|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{3\hat{\kappa}_H}\right\}, -\tau_k \Delta_k^2\right\}. \tag{27}$$

There are two cases for $\sigma_m(x_k)$: either $\|g_k\| = \sigma_m(x_k)$ or $-\tau_k = -\lambda_{\min}(H_k) = \sigma_m(x_k)$.

If $\Delta_k \le \sigma_m(x_k)/\hat{\kappa}_H$, then

$$m_k(0) - m_k(d_k) \ge \kappa_\sigma \|g_k\| \Delta_k = \kappa_\sigma \sigma_m(x_k)\Delta_k. \tag{28}$$

Hence, from (11), (12), and (28), we have that

$$\begin{aligned}
\left|\rho_k - 1\right| &= \frac{\left|f(x_k) - f(x_{k+1})\right|}{\left|m_k(0) - m_k(d_k)\right|} \\
&\le \frac{\left|f(x_k) - f(x_{k+1}) - m_k(0) + m_k(d_k)\right|}{\kappa_m \sigma_m(x_k)\Delta_k} \\
&\le \frac{\|\nabla f(x_k) - g_k\|\|d_k\| + 1/2\|\nabla^2 f(x_k) - H_k\|\|d_k\|^2 + [\lambda_{\min}(H_k)]_+\|d_k\|^2 + L_H\|d_k\|^3}{\kappa_m \sigma_m(x_k)\Delta_k} \\
&\le \frac{n\kappa_H/2\Delta_k^3 + \kappa_\sigma\Delta_k^3 + [\lambda_{\min}(H_k)]_+\Delta_k^2 + L_H\Delta_k^3}{\kappa_m \sigma_m(x_k)\Delta_k} \\
&= \frac{(n\kappa_H/2 + \kappa_\sigma + L_H)\Delta_k^3 + [\lambda_{\min}(H_k)]_+\Delta_k^2}{\kappa_m \sigma_m(x_k)\Delta_k} \\
&\le \frac{(n\kappa_H/2 + \kappa_\sigma + L_H)\Delta_{\max} + [\lambda_{\min}(H_k)]_+}{\kappa_m \sigma_m(x_k)}\Delta_k \le 1 - \eta_1.
\end{aligned} \tag{29}$$

If $-\tau_k = \sigma_m(x_k)$, then

$$m_k(0) - m_k(d_k) \ge -\kappa_m \tau_k \Delta_k^2 = -\kappa_m \sigma_m(x_k)\Delta_k^2. \tag{30}$$

Similar to the proof of (29), we deduce from (11), (12), and (30) that

$$\begin{aligned}\left|\rho_k - 1\right| &= \frac{\left|f(x_k) - f(x_{k+1})\right|}{\left|m_k(0) - m_k(d_k)\right|} \\ &\le \frac{\left|f(x_k) - f(x_{k+1}) - m_k(0) + m_k(d_k)\right|}{\kappa_m \sigma_m(x_k)\Delta_k} \\ &\le \frac{(n\kappa_H/2 + \kappa_\sigma + L_H + 1/\varepsilon_1)\Delta_k^3}{\kappa_m \sigma_m(x_k)\Delta_k^2} \le 1 - \eta_1.\end{aligned} \tag{31}$$

Combining (29) and (31), we have that x_k is a successful iteration. □

Lemma 10. *Under Assumption 2, if for constants $\kappa_1 > 0$, we have $\sigma_m(x_k) \ge \kappa_1$ for all k. Then,*

$$\Delta_k \ge \kappa_2, \tag{32}$$

for all k, where $\kappa_2 > 0$ is a constant.

Proof. By Lemma 9 and $\sigma_m(x_k) \ge \kappa_1$ for all k, if Δ_k satisfies the following condition

$$\varepsilon_1\kappa_1 \le \Delta_k \le \overline{\kappa}_2 = \min\left[\frac{\kappa_1}{3\hat{\kappa}_H}, \frac{\kappa_m\kappa_1(1-\eta_1)}{(n\kappa_H/2 + \kappa_\sigma + L_H)\Delta_{\max}}, \frac{\kappa_m\kappa_1(1-\eta_1)}{(n\kappa_H/2 + \kappa_\sigma + L_H)}\right], \tag{33}$$

then k th iteration must be successful, and from step 6 of Algorithm 1, we have $\Delta_{k+1} \ge \Delta_k \ge \overline{\kappa}_2 = \kappa_2$; the conclusion holds.

The following lemma shows that Algorithm 1 can generate a second-order stationary point under finitely successful iterations. □

Lemma 11. *If the number of successful iterations is finite, then*

$$\lim_{k \longrightarrow +\infty} \sigma(x_k) = 0. \tag{34}$$

Proof. Because there is a finite number of successful iteration, then there is an infinite unsuccessful iteration that such that the trust region is reduced. Hence, Δ_k is decreased and converges to zero.

Let us consider that

$$\begin{aligned}\left|\sigma(x_k)\right| &= \left|\sigma(x_k) - \sigma_m(x_k) + \sigma_m(x_k)\right| \\ &\le \left|\sigma(x_k) - \sigma_m(x_k)\right| + \left|\sigma_m(x_k)\right| \le \kappa_\sigma\Delta_k + \left|\sigma_m(x_k)\right|.\end{aligned} \tag{35}$$

If $|\sigma_m(x_k)| > \varepsilon > 0$ for a subsequence, then for sufficiently small Δ_k, the index k is successful, which yields a contradiction. Hence, we have

$$\lim_{k \longrightarrow +\infty} \sigma(x_k) = 0. \tag{36}$$

□

Lemma 12. *Under Assumption 2, if the sequence Δ_k is generated by Algorithm 1, then*

$$\lim_{k \longrightarrow +\infty} \Delta_k = 0. \tag{37}$$

Proof. Similar proof can be found in Lemma 7.7 of [24].

Next, we give the important conclusion as a corollary. □

Lemma 13. *Under Assumption 2, we have that*

$$\liminf_{k \longrightarrow +\infty} \sigma_m(x_k) = 0. \tag{38}$$

Proof. First, we assume that there exists $\kappa_1 > 0$ such that

$$\sigma_m(x_k) \ge \kappa_1, \tag{39}$$

for all k. By Lemma 10, we have that $\Delta_k \ge \kappa_2 > 0$ for all k. This is a contradiction with Lemma 11.

We prove that there exists a subsequence generated by Algorithm 1 convergent to a second-order stationary point of problem (P). □

Lemma 14. *Under Assumption 2, if*

$$\lim_{i \longrightarrow +\infty} \sigma_m\left(x_{k_i}\right) = 0, \tag{40}$$

for any subsequence $\{k_i\}$ holds, then

$$\lim_{i \longrightarrow +\infty} \sigma\left(x_{k_i}\right) = 0. \tag{41}$$

Proof. By (40), we have that $\sigma_m(x_{k_i}) \le \varepsilon$ for large enough i, and then, by Lemma 12, we have $\Delta_{k_i} \longrightarrow 0$ as $k_i \longrightarrow \infty$. By (19), we have that

$$\sigma\left(x_{k_i}\right) = \left(\sigma\left(x_{k_i}\right) - \sigma_m\left(x_{k_i}\right)\right) + \sigma_m\left(x_{k_i}\right) \le \kappa_\sigma\Delta_{k_i} + \sigma_m\left(x_{k_i}\right). \tag{42}$$

Combining (40) and (42), we have that (41) holds.

According to Lemmas 13 and 14, we can obtain the following global convergence of Algorithm 1. □

Theorem 15. *Under Assumption 2, we obtain that*

$$\liminf_{k \longrightarrow +\infty} \sigma(x_k) = 0. \tag{43}$$

Theorem 15 shows that Algorithm 1 generates a subsequence converging to second-order critical of problem (P). Next, we prove that the total sequence generated by Algorithm 1 converges to second-order critical of problem (P).

Theorem 16. *Under Assumption 2, we can obtain that*

$$\lim_{k \longrightarrow +\infty} \sigma(x_k) = 0. \tag{44}$$

Proof. Similar proof can be found in Theorem 7.11 of [24]. □

5. Numerical Results

In this section, in order to test the efficiency of our algorithm, we choose 49 unconstrained optimization test problems from [22]. The dimension and name of test problems are reported in Table 1. Before solving the problems in Table 1, we will introduce the parameters selected in the actual calculation of the algorithm proposed in this paper:

$$\begin{aligned} \Delta_0 &= 1, \\ \delta &= 0.8, \\ \eta_1 &= 0.3, \\ \eta_2 &= 0.7, \\ \Delta_{\max} &= 5, \\ \gamma_1 &= 0.8, \\ \gamma_2 &= 1.5. \end{aligned} \tag{45}$$

In order to solve the test problems in Table 1 by using the algorithm in this paper, we use MATLAB (2014a) to write the computer program, and the computer is HP (CPU is i7-8700, main frequency is 3.2 Hz, and memory is 16 G). The termination accuracy of the algorithm is $\varepsilon = 10^{-5}$.

In order to draw a comparison diagram of the results of the algorithms, we use the performance comparison formula of the algorithm proposed by Dolan and More [27] to calculate the computational efficiency between different algorithms. Here are the specific formulas:

$$r_{p,s} = \frac{\tau_{p,s}}{\min\{\tau_{p,u}: u \in S\}}, \tag{46}$$

where S denotes the set of algorithms. Let P denotes problem sets, $n_s = |S|$ and $n_p = |P|$. For $\forall t \geq 1$, let

$$\rho_s(t) = \frac{1}{n_p} \text{size}\{p \in P: r_{p,s} \leq t\}, \tag{47}$$

where $\rho_s(t)$ represents the efficiency of each solver.

In order to further test the effectiveness of our algorithm, we will select some derivative-free optimization problems for testing and calculate these problems using existed derivative-free algorithms. The calculation results are recorded in Figure 1. We will first introduce the tested problems. We list all the test problems in Table 1, which shows the name of the problem, dimension n, and the source of the problems. From Table 1, we can see that all test problems are the nonsmooth problems, and their derivatives are not obtained directly. Hence, it is more appropriate to use our algorithm to solve the problems in Table 1.

In Table 1, many objective functions are given in the form of max, and in the actual calculation process, we equivalently rewrite the objective function into the form of a single objective function. If the objective function is $f(x) = \max\{f_1(x), f_2(x)\}$, then we can obtain the equivalent single objective function as

$$f(x) = \frac{f_1(x) + f_2(x) - |f_1(x) - f_2(x)|}{2}. \tag{48}$$

If there are more than two functions in the objective function of the original problem compared in size, we will repeatedly apply (48) to obtain a single objective function. For example, if the objective function is $f(x) = \max\{f_1(x), f_2(x), f_3(x)\}$, we can obtain the single objective function as follows:

$$\begin{aligned} f(x) &= \frac{f_1(x) + f_2(x) - |f_1(x) - f_2(x)|/2 + f_3(x) - |f_1(x) + f_2(x) - |f_1(x) - f_2(x)|/2 - f_3(x)|}{2} \\ &= \frac{f_1(x) + f_2(x) - |f_1(x) - f_2(x)| + 2f_3(x) - |f_1(x) + f_2(x) - |f_1(x) - f_2(x)| - 2f_3(x)|}{4}. \end{aligned} \tag{49}$$

The case where more functions are included is similar to the calculation in (49), which can result in a single objective function form. It is obvious that (48) and (49) are also nonsmooth, and their derivatives cannot be obtained directly.

Our algorithm mainly provides a dynamic method for adjusting the difference coefficient, so that in the actual calculation process, our algorithm does not use the traditional fixed difference coefficient calculation method but dynamically adjusts the difference coefficient based on the actual descent of the objective function. This approach can avoid the problem of fixed coefficients causing slow descent of the objective function, especially to overcome the problem of poor robustness caused by fixed difference coefficients in the algorithm.

DEO-TRNS and NOMAD are both famous derivative-free algorithms used to solve nonsmooth black-box optimization problems. They are based on the trust-region framework and construct derivative-free algorithms. They obtain the search direction by solving the approximate trust region subproblem and update the trust region radius by the actual degree of the descent of the objective function to obtain the optimal solution of the problem. By using our algorithm, DEO-TRNS, and NOMAD to calculate the

Table 1: Test problem.

Name	n	The form of the problem
Crescent [26]	2	$f(x) = \max\left\{x_1^2 + (x_2-1)^2 + x_2 - 1, -x_1^2 - (x_2-1)^2 + x_2 + 1\right\}$
CB2 [26]	2	$f(x) = \max\left\{x_1^2 + x_2^4, (2-x_1)^2 + (2-x_2)^2, 2e^{-x_1+x_2}\right\}$
CB3 [26]	2	$f(x) = \max\left\{x_1^4 + x_2^2, (2-x_1)^2 + (2-x_2)^2, 2e^{-x_1+x_2}\right\}$
DEM [26]	2	$f(x) = \max\{5x_1 + x_2, -5x_1 + x_2, x_1^2 + x_2^2 + 4x_2\}$
QL [26]	2	$f(x) = \max\{x_1^2 + x_2^2, x_1^2 + x_2^2 + 10(-4x_1 - x_2 - 4), x_1^2 + x_2^2 + 10(-x_1 - 2x_2 + 6)\}$
LQ [26]	2	$f(x) = \max\{-x_1 - x_2, -x_1 - x_2 + (x_1^2 + x_2^2 - 1)\}$
Mifflin 1 [26]	2	$f(x) = -x_1 + 20\max\{0, x_1^2 + x_2^2 - 1\}$
Mifflin 2 [26]	2	$f(x) = -x_1 + 2(x_1^2 + x_2^2 - 1) + 1.75\lvert x_1^2 + x_2^2 - 1\rvert$
Rosen-Suzuki [26]	4	$f(x) = \max\{f_1(x), f_1(x) + 10f_2(x), f_1(x) + 10f_3(x), f_1(x) + 10f_4(x)\}$ $f_1(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$ $f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8$ $f_2(x) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10$ $f_4(x) = x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5$
Shor [26]	5	$f(x) = \max_{1\le i\le 10}\left\{b_i\sum_{j=1}^{5}(x_j - 5)^2\right\}$, where a_{ij} and b_i are given in [26]
Colville [26]	5	$f(x) = \sum_{j=1}^{5} d_j x_j^3 + \sum_{i=1}^{5}\sum_{j=1}^{5} c_{ij}x_i x_j + \sum_{j=1}^{5} e_j x_j$ $+50\max \quad \left\{0, \max_{1\le i\le 10}(b_i - \sum_{j=1}^{5} a_{ij}x_j)\right\}$ where a_{ij}, c_{ij}, e_j and b_i are given in [26]
HS78 [26]	5	$f(x) = x_1x_2x_3x_4x_5 + 10\sum_{i=1}^{3}\lvert f_i(x)\rvert$ $f_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10$ $f_2(x) = x_2x_3 - 5x_4x_5$ $f_3(x) = x_1^3 + x_2^3 + 1$
El-Attar [26]	6	$f(x) = \sum_{i=1}^{51}\lvert x_1 e^{-x_2 t_i}\cos(x_3 t_i + x_4) + x_5 e^{-x_6 t_i} - y_i\rvert$ $y_i = 0.5e^{-t_i} - e^{-2t_i} + 0.5e^{-3t_i} + 1.5e^{-1.5t_i}\sin 7t_i + e^{-2.5t_i}\sin 5t_i$ $t_i = 0.1(i-1), 1 \le i \le 51$
Maxquad [26]	10	$f(x) = \max_{1\le i\le 5}(x^T A^i x - x^T b^i),\ A_{kj}^i = A_{jk}^i = e^{j/k}\cos(jk)\sin(i), j < k$ $A_{jj}^i = j/10\lvert\sin(j)\rvert + \sum_{k\ne j}\lvert A_{jk}^i\rvert,\ b_j^i = e^{j/i}\sin(ij)$
Gill [26]	10	$f(x) = \max\{f_1(x), f_2(x), f_3(x)\}$ $f_1(x) = \sum_{i=1}^{10}(x_i - 1)^2 + 10^{-3}\sum_{i=1}^{10}(x_i^2 - 1/4)^2$ $f_2(x) = \sum_{i=2}^{30}[\sum_{j=2}^{10} x_j(j-1)(i - 1/29)^{j-2} - (\sum_{j=1}^{10} x_j(i = 1/29)^2 - 1)^2 + x_1^2$ $+(x_2 - x_1^2 - 1)^2]$ $f_3(x) = \sum_{i=2}^{10}[100(x_i - x_{i-1}^2)^2 + (1 - x_i)^2]$
Maxl [26]	{10, 20, 30}	$f(x) = \max_{1\le i\le n}\{\lvert x_i\rvert\}$
TR48 [26]	{10, 20, 30}	$f(x) = \sum_{j=1}^{n} d_j \max_{1\le i\le n}(x_i - a_{ij}) - \sum_{j=1}^{n} s_i x_i a_{ij}$, s_i and d_j are given in [26]
L1HILB [26]	{10, 20, 30}	$f(x) = \sum_{i=1}^{n}\lvert\sum_{j=1}^{n} x_j/i + j - 1\rvert$
Shell Dual [26]	15	$f(x) = 2\lvert\sum_{i=1}^{5} d_i x_{i+10}^3\rvert + \sum_{i=1}^{5}\sum_{j=1}^{5} c_{ij}x_{i+10}x_{j+10} - \sum_{i=1}^{10} b_i x_i + 100\ (\sum_{i=1}^{5}\max$ $(0, P_i(x)) - Q(x))$ $P_i(x) = \sum_{j=1}^{10} a_{ji}x_j - 2\sum_{j=1}^{5} c_{ij}x_{j+10} - 3d_i x_{i+10}^2 - e_i, 1 \le i \le 5$ $Q(x) = \sum_{i=1}^{15}\min(0, x_i)$, where a_{ij}, c_{ij}, e_i, d_i and b_i are given in [26]
Chained LQ [28]	{10, 20, 30}	$f(x) = \sum_{i=1}^{n-1}\max\{-x_i - x_{i+1}, -x_i - x_{i+1} + (x_i^2 + x_{i+1}^2 - 1)\}$
Chained CB3 I [28]	{10, 20, 30}	$f(x) = \sum_{i=1}^{n-1}\max\left\{x_i^4 + x_{i+1}^2, (2 - x_i)^2 + (2 - x_{i+1})^2, 2e^{-x_i+x_{i+1}}\right\}$
Chained CB3 II [28]	{10, 20, 30}	$f(x) = \max\left\{\sum_{i=1}^{n-1}(x_i^4 + x_{i+1}^2), \sum_{i=1}^{n-1}((2 - x_i)^2 + (2 - x_{i+1})^2)\right\}, \sum_{i=1}^{n-1}(2e^{-x_i+x_{i+1}})$
Number of active faces [28]	{10, 20, 30}	$f(x) = \max_{1\le i\le n}\{\ln(\lvert - \sum_{i=1}^{n} x_i\rvert + 1), \ln(\lvert x_i\rvert + 1)\}$

Table 1: Continued.

Name	n	The form of the problem
Nonsmooth generalization of brown function 2 [28]	{10, 20, 30}	$f(x) = \sum_{i=1}^{n-1} (\lvert x_i \rvert^{x_{i+1}^2+1} + \lvert x_{i+1} \rvert^{x_i^2+1})$
Chained Mifflin2 [28]	{10, 20, 30}	$f(x) = \sum_{i=1}^{n-1} (-x_i + 2(x_i^2 + x_{i+1}^2 - 1) + 1.75\lvert x_i^2 + x_{i+1}^2 - 1\rvert)$
Chained crescent I [28]	{10, 20, 30}	$f(x) = \max\left\{\sum \sum_{i=1}^{n-1} x_i^2 + (x_{i+1} - 1)^2 + x_{i+1} - 1, \sum_{i=1}^{n-1} (-x_i^2 - (x_{i+1} - 1)^2 + x_{i+1} + 1)\right\}$
Chained crescent II [28]	{10, 20, 30}	$f(x) = \sum_{i=1}^{n-1} \max\left\{x_i^2 + (x_{i+1} - 1)^2 + x_{i+1} - 1, -x_i^2 - (x_{i+1} - 1)^2 + x_{i+1} + 1\right\}$

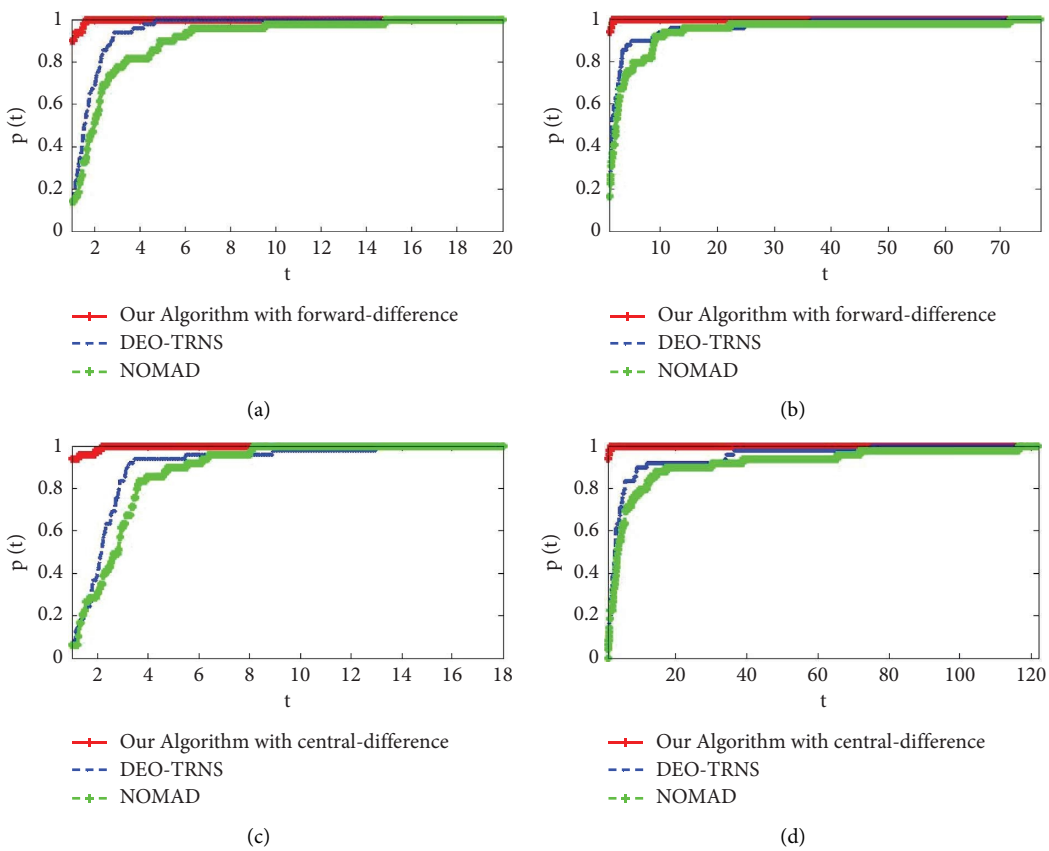

Figure 1: (a) The comparison results of the number of iterations of forward-difference with DEO-TRNS and NOMAD. (b) The comparison results of CPU time of forward-difference with DEO-TRNS and NOMAD. (c): The comparison results of the number of iterations of central-difference with DEO-TRNS and NOMAD. (d) The comparison results of CPU time of central-difference with DEO-TRNS and NOMAD.

problems in Table 1, we obtained the corresponding calculation results and recorded the results in Figure 1.

According to (46) and (47), from the subgraphs in the upper left and right corners of Figure 1, it can be seen that our algorithm can effectively solve the problem in Table 1 by approximating the derivative of the problem using either central-difference or forward-difference techniques. The main manifestation is that at $t = 0$, the red curve representing our algorithm using forward-difference has achieved 0.9, but the blue curve representing DEO-TRNS and the green curve representing NOMAD only achieved 0.18 and 0.17, respectively. When using the central-difference technique, the red curve representing our algorithm has achieved 0.95, but the blue curve representing DEO-TRNS and the green curve representing NOMAD only achieved 0.22 and 0.19. The red curve representing our algorithm is much higher than the blue curve representing DEO-TRNS and the green curve representing NOMAD. Moreover, the speed at which the red curve tends to 1 is generally higher than that of the blue and green curves. This means that our algorithm can solve most problems with minimal iteration costs. From the subgraphs in the lower left and right corners of Figure 1, we can also see that both red curves have achieved 0.95 when $t = 0$, but the values corresponding to the blue and green curves are much lower than those corresponding to the red curve. The speed at which the red curve tends to 1 is generally higher than that of the blue and green curves. From the values in the abscissa, it can be seen that our algorithm

spends far fewer CPU time solving the problems in Table 1 than DEO-TRNS and NOMAD. This indicates that our algorithm is very effective in solving the actual problems in Table 1.

6. Concluding Remarks

This paper proposes a trust-region method with forward-difference or central-difference approximation techniques for obtaining the second-order stationary points of derivative-free nonconvex optimization problem (P). The finite-difference technique is used to approximate the gradient and Hessian matrix of the objective function. The search direction is obtained by solving the trust-region subproblem. We prove global convergence of the algorithm proposed by this paper without the fully quadratic approximation; i.e., the algorithm generates a sequence converging to the second-order stationary point of the problem (P). In the numerical calculation section, first, we compute 49 test problems using our algorithm, NOMAD, and DEO-TRNS, respectively. The results show that our algorithm spent fewer iterations and CPU time than other algorithms. We also compute 46 test problems using forward-difference and central-difference techniques. The results of comparing iterations and CPU time show that the central-difference approximation uses smaller iteration than the forward-difference approximation in solving most problems.

Data Availability

The data supporting this meta-analysis are from previously reported studies and datasets, which have been cited.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The author thanks the support of the National Natural Science Foundation (11371253), the Hainan Natural Science Foundation (120MS029) and and Hainan Provincial Natural Science Foundation (120MS028).

References

[1] Y. Nesterov and B. T. Polyak, "Cubic regularization of Newton method and its global performance," *Mathematical Programming*, vol. 108, no. 1, pp. 177–205, 2006.

[2] C. Cartis, N. I. Gould, and P. L. Toint, "Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results," *Mathematical Programming*, vol. 127, no. 2, pp. 245–295, 2011.

[3] C. Cartis, N. I. Gould, and P. L. Toint, "Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function-and derivative-evaluation complexity," *Mathematical Programming*, vol. 130, no. 2, pp. 295–319, 2011.

[4] F. E. Curtis, D. P. Robinson, and M. Samadi, "A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization," *Mathematical Programming*, vol. 162, no. 1-2, pp. 1–32, 2017.

[5] F. E. Curtis and D. P. Robinson, "Exploiting negative curvature in deterministic and stochastic optimization," 2017, https://arxiv.org/abs/1703.0041.

[6] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent converges to minimizers," 2016, https://arxiv.org/abs/1602.04915.

[7] S. S. Du, C. Jin, J. D. Lee, M. I. Jordan, A. Singh, and B. Poczos, "Gradient descent can take exponential time to escape saddle points," in *Advances in Neural Information Processing Systems*, p. 1067C1077, MIT Press, Cambridge, MA, USA, 2017.

[8] A. Auslender and M. Teboulle, "Projected subgradient methods with non-Euclidean distances for non-differentiable convex minimization and variational inequalities," *Mathematical Programming*, vol. 120, no. 1, pp. 27–48, 2009.

[9] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming*, vol. 117, no. 1-2, pp. 387–423, 2009.

[10] A. Auslender and M. Teboulle, "Interior gradient and proximal methods for convex and conic optimization," *SIAM Journal on Optimization*, vol. 16, no. 3, pp. 697–725, 2006.

[11] S. Villa, S. Salzo, L. Baldassarre, and A. Verri, "Accelerated and inexact forward-backward algorithms," *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1607–1633, 2013.

[12] N. S. Keskar and A. Wächter, "A limited-memory quasi-Newton algorithm for boundconstrained non-smooth optimization," *Optimization Methods and Software*, vol. 34, no. 1, pp. 150–171, 2019.

[13] A. S. Lewis and M. L. Overton, "Nonsmooth Optimization via BFGS," 2008.

[14] A. S. Berahas, R. H. Byrd, and J. Nocedal, "Derivative-free optimization of noisy functions via quasi-Newton methods," *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 965–993, 2019.

[15] J. J. Moré and S. M. Wild, "Estimating computational noise," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1292–1314, 2011.

[16] R. W. Hamming, *Introduction to Applied Numerical Analysis*, Courier Corporation, Chelmsford, MA, USA, 2012.

[17] R. C. M. Brekelmans, L. T. Driessen, H. J. M. Hamers, and D. D. Hertog, "Gradient estimation schemes for noisy functions," *Journal of Optimization Theory and Applications*, vol. 126, no. 3, pp. 529–551, 2005.

[18] A. S. Berahas, L. Cao, K. Choromanski, and K. Scheinberg, "A theoretical and empirical comparison of gradient approximations in derivative-free optimization," 2019, https://arxiv.org/abs/1905.01332.

[19] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[20] E. Gorbunov, P. Dvurechensky, and A. Gasnikov, "An accelerated method for derivative-Free smooth stochastic convex optimization," *Optimization and Control*, vol. 20, pp. 1–38, 2020.

[21] S. Bellavia, G. Gurioli, B. A. Morini, and P. L. Toint, "The impact of noise on evaluation complexity: the deterministic trust-region case," *Journal of Optimization Theory and Applications*, vol. 196, no. 2, pp. 700–729, 2023.

[22] K. Schittkowski, "More test examples for nonlinear programming codes," *Lecture Notes in Economics and Mathematical Systems*, vol. 282, pp. 1–261, 1981.

[23] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1987.

[24] A. R. Conn, K. Scheinberg, and L. N. Vicente, "Global convergence of general derivative-free trust-region algorithms to

first- and second-order critical points," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 387–415, 2009.

[25] M. J. D. Powell, "A new algorithm for unconstrained optimization," in *Nonlinear Programming*, J. B. Rosen, O. L. Mangasarian, and K. Ritter, Eds., pp. 31–36, Academic Press, New York, NY, USA, 1971.

[26] L. Lukšan and J. Vlček, "Test problems for nonsmooth unconstrained and linearly constrained optimization," Technical report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 2000.

[27] E. D. Dolan and J. J. More, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

[28] N. Karmitsa, *Test Problems for Large-Scale Nonsmooth Minimization, Reports of the Department of Mathematical Information Technology, Series B, Scientific Computing, No. B.4*, Department of Mathematical Information Technology, University of Jyväskylä, Jyväskylä, Finland, 2007.