Pacific Journal of Optimization
Vol. 21, No. 1, pp. 31–50, 2025

SOME GLOBAL OPTIMIZATION STRATEGIES OF FEASIBLE TRUST-REGION METHOD FOR EXTREME Z-EIGENVALUES OF SYMMETRIC TENSORS

Xiao Li, Chun-Lin Hao*, Yitian Xu, Jun-Yu Gao and Zu-Ping Li

Abstract: It is well-known that tensor eigenvalues have been widely used in signal processing, diffusion tensor imaging, independent component analysis and other fields. Effective methods for solving Z-eigenvalues of large-scale tensors tend to fall into a local optimal region. The feasible trust-region method is an effective method for computing the extreme Z-eigenvalue of large-scale tensors, but it also tends to fall into the local optimal. To overcome the problem, we propose three global optimization strategies based on the feasible trust-region method to improve the success rate of computing the extreme Z-eigenvalue. The first one is a multi-initial points algorithm. The second one is a simulated annealing algorithm. The third one is an infeasible trust-region algorithm which constructs an infeasible trust-region subproblem, expanding the search scope and expecting to improve the success rate. We prove the global convergence of the infeasible trust-region algorithm. The numerical results show that the success rate of calculating the extreme Z-eigenvalue is greatly increased with the help of the global optimization strategy.

Key words: *tensor Z-eigenvalue, trust-region subproblem with constraints, feasible trust-region method, global optimization strategy*

Mathematics Subject Classification: *15A18, 15A69, 90C55*

1 Introduction

Tensor is a multidimensional array. Let m and n be positive integers. An mth order n-dimensional real tensor $\mathcal{A}$ can be expressed as

$$\mathcal{A} = (a_{i_1,i_2,\ldots,i_m}),\ a_{i_1,i_2,\ldots,i_m} \in \mathbb{R},\ 1 \le i_1, i_2, \ldots, i_m \le n.$$

For example, a first-order tensor is a vector, which has one subscript for each element (a_i); a tensor of second-order is a matrix, which has two subscripts for each element (a_{ij}); tensors with order three or greater are called higher-order tensors, which have three or more subscripts for each element. The tensor $\mathcal{A}$ is symmetric if each entry $a_{i_1,i_2,\ldots,i_m}$ remains invariant under every permutation of $(i_1, i_2, \ldots, i_m)$. Let $\mathbf{T}^m(\mathbb{R}^n)$ represent the space encompassing all mth order n-dimensional real tensors, while $\mathbf{S}^m(\mathbb{R}^n)$ is the space of all symmetric tensors within $\mathbf{T}^m(\mathbb{R}^n)$. For $\mathcal{A} \in \mathbf{S}^m(\mathbb{R}^n), x \in \mathbb{R}^n$, we denote

$$\mathcal{A}x^m := \sum_{1 \le i_1, i_2, \ldots, i_m \le n} a_{i_1,i_2,\ldots,i_m} x_{i_1} x_{i_2} \ldots x_{i_m},$$

*Corresponding Author

which is an m degree homogeneous polynomial. Let $\mathcal{A}x^k$ be the symmetric tensor in $\mathbf{S}^{m-k}(\mathbb{R}^n)$, defined as

$$(\mathcal{A}x^k)_{i_1,\ldots,i_{m-k}} := \sum_{1\le j_1,\ldots,j_k\le n} \mathcal{A}_{i_1,\ldots,i_{m-k},j_1,\ldots,j_k} x_{j_1}\ldots x_{j_k},\ k \le m.$$

Eigenvalues of tensors were suggested by Qi [20] and Lim [14] in 2005. Tensor eigenvalues, similar to matrix eigenvalues, have wide significant application in practical problems. For example, signal processing [21], imaging analysis [3, 22], automatic control [16] and so on. Among them, Z-eigenvalues, especially those extreme ones, have many applications. The largest Z-eigenvalue gives rise to the best rank-1 approximation [25], and the smallest Z-eigenvalue can be applied to determine the positive definiteness of an even order symmetric tensor [13], which holds a significant role in medical image noise reduction and the stability of automatic control systems.

For tensor $\mathcal{A}$, if there exists $\lambda \in \mathbb{R}$ and $x \in \mathbb{R}^n$ such that

$$\begin{aligned} \mathcal{A}x^{m-1} &= \lambda x, \\ x^T x &= 1, \end{aligned} \tag{1.1}$$

then λ is considered a Z-eigenvalue of $\mathcal{A}$ and x is the corresponding Z-eigenvector. Then, any vector x satisfying (1.1) is a KKT point of the polynomial optimization problem

$$\begin{aligned} \max_{x\in\mathbb{R}^n}\ & \mathcal{A}x^m \\ \text{s.t.}\ & x^T x = 1 \end{aligned} \tag{1.2}$$

with $(\mathcal{A}x^m, x)$ being a Z-eigenpair.

The methods to compute the largest (smallest) Z-eigenvalues of symmetric tensors can be classified into the following types: the first type is the higher order power method (HOPM) [9–11, 26]. For instance, Kofidis [9] proposed a symmetric higher-order power method by extending the power method to higher-order supersymmetric tensor. Kolda and Mayo [10] added an adaptive method for choosing shifts in the HOPM to obtain extreme Z-eigenvalues. Zhou [26] drew inspiration from the inverse power method used for matrices and presented the shifted inverse power method. The second type is the semidefinite programming method [4, 8, 12, 17, 18]. Specifically, Lasserre [12] proposed the semidefinite relaxation method to get the largest(smallest) eigenvalue. Hu [8] introduced a tensor conic linear programming for calculating extreme Z-eigenvalues. Cui [4] applied Jacobian SDP relaxations to accurately compute all real eigenvalues. Nie [17] presented semidefinite relaxations that utilize sum-of-squares representations for tensors. The third type is the sequential subspace projection method (SSPM) [5, 23]. Hao [5] formed a subspace and solved the original optimization problem in the subspace for extreme Z-eigenvalues. Yu [23] devised an approach of adaptive gradient for generalized tensor eigenpairs. The fourth type is the feasible trust-region method (FTR) [1, 6]. In detail, Hao [6] presented a novel method for computing the extreme Z-eigenvalues using trust-region subproblems. Cao [1] used the self-adaptive technique in trust region method for computing extreme B-eigenvalues. In addition, there are alternative solutions, including the accelerated Levenberg-Marquardt method [2], the neural dynamical network [7], etc. The semidefinite programming method [4, 17] is often used to solve small-scale problems. The sequential subspace projection method (SSPM) [5] and the feasible trust-region method (FTR) [6] can be employed to handle large-scale problems, but it is worth noting that achieving a high success rate in the computation of extreme

Z-eigenvalues is often a challenging task. The success rate refers to the probability of obtaining the extreme Z-eigenvalues of tensors from any initial point. We have observed that many researches can get the extreme Z-eigenvalues. However, their success rate in computing the extreme Z-eigenvalues is far from satisfactory. The solution quality of the algorithm is greatly important, besides the number of iterations and the operation time. Inspired by this, this paper studies from the perspective of the success rate of computing the extreme Z-eigenvalues of symmetric tensors.

Although there are many methods for large-scale problems, FTR [6] has global convergence and local quadratic convergence. When calculating the extreme Z-eigenvalues of symmetric tensors, the number of iterations and the operation time of FTR are less, especially in high-dimension. FTR has improved the phenomenon of falling into the local optimal region, but it is far from satisfactory. Therefore, this paper studies the global optimization strategy based on the feasible trust-region method (FTR) [6] from the success rate of computing the extreme Z-eigenvalue of large-scale tensors.

In this paper, we shall propose three global optimization strategies based on FTR [6] to calculate extreme Z-eigenvalues of symmetric tensors. They are respectively a multi-initial points algorithm, a simulated annealing algorithm and an infeasible trust-region algorithm. The multi-initial points algorithm and the simulated annealing algorithm are general heuristic global optimization strategies. The infeasible trust-region algorithm constructs an infeasible trust-region subproblem. Specifically, the constraint on the subproblem is derived by linearizing the constraint on eigenvalue problem, and each iteration point of the subproblem is not feasible. In this manner, the above three strategies can provide a better initial point for FTR, improving the success rate of computing the extreme Z-eigenvalues. The main contributions of this paper are summarized as follows:

• We propose three global optimization strategies based on FTR. The three optimization strategies are inspired by the lower success rate of computing the extreme Z-eigenvalues, and can extend the search range, thus providing a better initial point for FTR.

• The experimental results confirm that, the proposed optimization strategies can significantly improve the success rate of calculating the extreme Z-eigenvalues.

The rest of this paper is organized as follows. In section 2, we describe the three global optimization strategies. The convergence of the infeasible trust-region algorithm is demonstrated in Section 3. Numerical results are detailed in Section 4, which show that the success rate of the algorithm is greatly increased with the help of the global optimization strategies. The conclusions are drawn in the last section.

2 Global Optimization Strategy

In this section, we propose three global optimization strategies to provide a better initial point for FTR. Specifically, the three optimization algorithms are the multi-initial points algorithm, the simulated annealing algorithm, and the infeasible trust region algorithm. We first introduce the motivation of the strategies, then describe the detailed contents of our proposal, and finally give the algorithm procedure.

2.1 The Multi-initial Points Algorithm

Multi-initial points algorithm is a classical and easy method of global optimization. Many current works do not mention this algorithm, as it is relatively simple. Thus, we consider it as the preprocessing strategy based on FTR.

The problem (1.2) can be rewritten as

$$\begin{aligned} \max_{x\in\mathbb{R}^n} \quad & f(x) = \frac{1}{m}\mathcal{A}x^m \\ \text{s.t.} \quad & \frac{1}{2}(x^T x - 1) = 0. \end{aligned} \tag{2.1}$$

By this reformulation, the KKT point x^* and the corresponding Lagrange multiplier λ^* of the problem (2.1) is a Z-eigenpair of tensor $\mathcal{A}$.

Multi-initial points algorithm is to randomly generate multiple initial points $x_k(k = 1, 2, \ldots, l)$ in the unit sphere and select the initial point $x_j(1 \leq j \leq l)$ with the largest objective function. Specifically, the flowchart is listed in Algorithm 2.1.

Algorithm 2.1. (The multi-initial points algorithm)

Step 0. Generate multiple initial points $x_1, x_2, \ldots, x_l$ in the unit sphere.

Step 1. Compute $f(x_1), f(x_2), \ldots, f(x_l)$.

Step 2. Let $j = \underset{1\leq k\leq l}{\operatorname{argmax}}\{f(x_k)\}$, output x_j.

Meanwhile, choosing the initial point with the smallest objective function in the above algorithm can be used as a preprocessing algorithm for solving the smallest Z-eigenvalue of tensors.

2.2 The Simulated Annealing Algorithm

The method is a representative global optimization algorithm. Meanwhile, it has the advantages of easy implementation and strong robustness, then it is taken into account. Unlike the multi-initial points algorithm, the simulated annealing algorithm accepts the points with relatively poor target value with a certain probability, so it can escape from local maximum and get the maximum value in the global range. The basic idea of the algorithm is to accept the points with relatively poor target value in the neighborhood according to probability in the process of algorithm iteration, but the probability of accepting the points with relatively poor target value will gradually decrease. This method can avoid falling into the local maximum in the early stage and accelerate the convergence speed in the later stage.

Specifically, the flowchart is listed in Algorithm 2.2.

Algorithm 2.2. (The simulated annealing algorithm 1)

Step 0. Given an initial point x_0, set a large initial value parameter T, a lowest threshold $Tmin$, the rate of descent α, number of iterations ζ, $k := 0$.

Step 1. If the stopping criterion is satisfied, terminate.

Step 2. Generate a random point $x_{new} \in \mathcal{N}(x_k)$, Compute $f = \mathcal{A}(\frac{x_k}{\|x_k\|_2})^m$, $f_{new} = \mathcal{A}(\frac{x_{new}}{\|x_{new}\|_2})^m$, $\Omega := f_{new} - f$.

Step 3. If $\Omega \geq 0$, set $x_{k+1} = x_{new}$; if $\Omega < 0$ and $e^{-\frac{\Omega}{T}} > \kappa \in (0, 1)$, set $x_{k+1} = x_{new}$; otherwise, $x_{k+1} = x_k$ and go to Step 2, iterate ζ times.

Step 4. Let $T = T * \alpha, k = k + 1$, and go to Step 1.

Specifically, 'κ' represents a randomly generated number between (0,1) and we accept x_{new} with the probability $e^{-\frac{\Omega}{T}}$ as a new iteration point. $x_{new} \in \mathcal{N}(x_k)$ means x_{new} is generated randomly in the neighborhood of the iteration point x_k. In addition, we also can consider generating x_{new} in the whole real field. Therefore, Algorithm 2.2 can be rewritten as follows.

Algorithm 2.3. (The simulated annealing algorithm 2)

Step 0. Given an initial point x_0, set $T, Tmin, \zeta, \alpha, k := 0$.

Step 1. If the stopping criterion is satisfied, terminate.

Step 2. Randomly generate a point x_{new}, Compute $f = \mathcal{A}(\frac{x_k}{\|x_k\|_2})^m, f_{new} = \mathcal{A}(\frac{x_{new}}{\|x_{new}\|_2})^m$, $\Omega := f_{new} - f$.

Step 3. If $\Omega \geq 0$, set $x_{k+1} = x_{new}$; if $\Omega < 0$ and $e^{-\frac{\Omega}{T}} > \kappa \in (0,1)$, set $x_{k+1} = x_{new}$; otherwise, $x_{k+1} = x_k$ and go to Step 2, iterate ζ times.

Step 4. Let $T = T * \alpha, k = k + 1$, and go to Step 1.

The difference between Algorithm 2.2 and Algorithm 2.3 lies in the different ways of generating new iteration points. After the processing of Algorithm 2.3, the numerical effect of some test cases is better than that of Algorithm 2.2, as can be seen in section 4.

2.3 The Infeasible Trust-Region Algorithm

Since the previous methods were carried out under the constraints feasible. For instance, FTR takes advantage of the projection operation and introduces a trust-region subproblem. The specific optimization problem is

$$\begin{aligned} \max_{d\in\mathbb{R}^n} \quad & m_k(d) = f_k + g_k^T d + \frac{1}{2} d^T W_k d, \\ \text{s.t.} \quad & x_k^T d = 0, \\ & \|d\|_2 \leq \Delta_k, \end{aligned} \tag{2.2}$$

It is obvious that all iteration points x_{k+1} are feasible. Now we consider whether we can jump out of the local optimal region if the constraints are not feasible. Thus, we construct an infeasible subproblem to expand the search scope and expect to improve the success rate. We propose the infeasible trust-region algorithm to solve (1.2).

Since the problem (1.2) can be reformulated as problem (2.1), we consider the following trust-region subproblem with linear constraint at the current iteration point x_k,

$$\begin{aligned} \max_{d\in\mathbb{R}^n} \quad & m_k(d) = f_k + g_k^T d + \frac{1}{2} d^T W_k d, \\ \text{s.t.} \quad & x_k^T d = \frac{1}{2}(1 - x_k^T x_k), \\ & \|d\|_2 \leq \delta_k, \end{aligned} \tag{2.3}$$

where

$$\begin{aligned} f_k = f(x_k),\ g_k = g(x_k) = \nabla f(x_k) - \lambda_k x_k, \\ W_k = W(x_k) = \nabla^2 f(x_k) - \lambda_k I. \end{aligned}$$

are the function value, gradient and Hessian of Lagrange function

$$L(\lambda, x) = f(x) - \frac{\lambda}{2}(x^T x - 1),$$

at (x_k, λ_k), respectively. We prescribe

$$\lambda_k = \nabla f(x_k)^T x_k = \mathcal{A} x_k^m, \tag{2.4}$$

δ_k is the trust-region radius. The constraint on subproblem (2.3) is derived by linearizing the constraint of problem (2.1). Meanwhile, each iteration point of the subproblem (2.3) may not be feasible ($x_k^T x_k \neq 1$).

We consider the constraints consistency of subproblem (2.3). If the constraints on this subproblem are consistent, the trial steps d_k is solved by interior-point method. Otherwise, if the constraints are inconsistent, a mechanism is introduced into the algorithm which computes the step in two stages. Gradually augmenting δ_k until the set of steps d_k that adhere to the linear constraints intersects the trust region for inconsistent cases is not advisable. Thus we consider to enhance the feasibility of constraints at each step and to satisfy them exactly in the limit. Therefore, we introduce a two-stage approach to solve the problem. In the first stage, we try to satisfy the linearized constraints within a good trust region, that is, solve the following subproblem

$$\begin{aligned} \min_{d_k^n \in \mathbb{R}^n} \quad & \frac{1}{2} \left| x_k^T d_k^n + \frac{x_k^T x_k - 1}{2} \right|^2, \\ \text{s.t.} \quad & \|d_k^n\|_2 \leq \xi \delta_k, \qquad \xi \in (0, 1). \end{aligned} \tag{2.5}$$

here $\xi = 0.8$. Then the total trial step d_k can be accepted by solving

$$\begin{aligned} \max_{d_k \in \mathbb{R}^n} \quad & m_k(d_k) = g_k^T d_k + \frac{1}{2} d_k^T W_k d_k, \\ \text{s.t.} \quad & x_k^T d_k = x_k^T d_k^n, \\ & \|d_k\|_2 \leq \delta_k. \end{aligned} \tag{2.6}$$

Therefore, for inconsistent case, we solve (2.5) and (2.6) to get the trial step d_k. Specifically, (2.5) and (2.6) are solved by conjugate gradient method.

Choosing the trust-region radius and judging whether to accept the trial step are two critical components in a trust-region algorithm. We define the ratio

$$\rho_k = \frac{f(x_k + d_k) - f(x_k)}{m_k(d_k) - m_k(0)} \tag{2.7}$$

where $f(x_k + d_k) - f(x_k)$ is the actual increase of $f(x)$, $m_k(d_k) - m_k(0)$ is the predicted increase. If the predicted increase $\Delta m := m_k(d_k) - m_k(0) > 0$, it indicates that a direction d can be found to rise the function $m_k(d)$. If the predicted increase $\Delta m \leq 0$, it shows that the function $m_k(d)$ cannot be increased along any direction d, then we accept the current trial step d_k and perform a unit projection

$$x_{k+1} = \frac{x_k + d_k}{\|x_k + d_k\|_2}.$$

The idea behind the algorithm is to allow the the algorithm exploring points that violate the constraints. Therefore, the decline of the objective function f is acceptable, but it

may further lead to a decrease in the objective function and a deviation from the sphere constraint. To address this issue, the constraints must remain feasible when accepting a decrease in the objective function. Hence, we employ the trial step and project $x_k + d_k$ onto the unit sphere. To be specific, the next iteration x_{k+1} is defined as

$$x_{k+1} = \begin{cases} \dfrac{x_k + d_k}{\|x_k + d_k\|_2}, & \Delta m \leq 0; \\ x_k, & \Delta m > 0, \rho_k \leq \sigma_0; \\ x_k + d_k, & \text{else.} \end{cases} \tag{2.8}$$

where σ_0 is a constant. If ρ_k is close to 1, there is a good agreement between the model m_k and the function f over this step, we expand the trust-region radius δ_k at the next iteration so that the objective function can rise sufficiently. If ρ_k is near zero or negative, we reduce δ_k in the next iteration. If ρ_k is belong to (σ_1, σ_2), we do not alter δ_k. Specifically, δ_{k+1} is updated as follows

$$\delta_{k+1} = \begin{cases} \dfrac{1}{2}\delta_k, & \rho_k \leq \sigma_1; \\ \min(2, 2\delta_k), & \rho_k \geq \sigma_2; \\ \delta_k, & \sigma_1 < \rho_k < \sigma_2. \end{cases} \tag{2.9}$$

where σ_1, σ_2 are constants with $0 < \sigma_0 < \sigma_1 < \sigma_2$, $\sigma_2 < 1$.

As described above, the flowchart are summarized in Algorithm 2.4.

Algorithm 2.4. (The infeasible trust-region algorithm for solving (2.3))

Step 0. Given an initial point x_0, set the parameters σ_0, σ_1, σ_2, δ_0. Let $k := 0$.

Step 1. Compute g_k, W_k, then solve (2.3) to obtain d_k.

Step 2. If $\|d_k\|$ satisfies the stop criterion, terminate and output (λ_k, x_k). Otherwise, go to Step 3.

Step 3. Compute ρ_k by (2.7).

Step 4. Update δ_k by (2.9).

Step 5. Update the next iteration x_k,
If $\Delta m = m_k(d_k) - m_k(0) \leq 0$, set

$$x_{k+1} = \frac{x_k + d_k}{\|x_k + d_k\|_2}, \lambda_{k+1} = \mathcal{A}x_{k+1}^m;$$

else if $\Delta m > 0$ and $\rho_k \leq \sigma_0$, set

$$x_{k+1} = x_k, \lambda_{k+1} = \lambda_k;$$

else if $\Delta m > 0$ and $\rho_k > \sigma_0$, set

$$x_{k+1} = x_k + d_k, \lambda_{k+1} = \mathcal{A}x_{k+1}^m;$$

end
Let $k := k + 1$, and go to Step 1.

If the maximization subproblem (2.3) in the above algorithm is changed to the minimization, it can be used as a preprocessing algorithm for solving the smallest Z-eigenvalue of tensors.

In some senses, the above three methods can be viewed as a plug-and-play module, which can be combined with existing tensor Z-eigenvalue solution methods. The general framework is to use the global optimization strategy to select better points and then employ methods to solve Z-eigenvalues.

3 Convergence Analysis

This section provides a proof of convergence for Algorithm 2.4 with respect to the problem (1.2).

Definition 3.1. x is called a strong stationary point of (1.2) if x is feasible and there exists a λ such that

$$\nabla f(x) - \lambda x = 0,$$

The concept on strong stationary point is originated from nonlinear programming [24].

Lemma 3.2. *For all x, y satisfying $1 \leq \|x\|_2 \leq 3$, $1 \leq \|y\|_2 \leq 3$, then*

$$\begin{aligned} \|W(x)\|_2 &\leq M, \\ \|g(x) - g(y)\|_2 &\leq L_0\|x - y\|_2, \\ \|\nabla^2 f(x) - \nabla^2 f(y)\|_2 &\leq L_1\|x - y\|_2, \end{aligned}$$

hold, where M, L_0, L_1 are positive constants [6].

Assumption 3.1. (1). The set of all trial points x in the algorithm is given by a non-empty, bounded and compact set D.

(2). There exists a subsequence $\{f_{k_j}\}$ of the sequence $\{f_k\}$, where the subsequence $\{f_{k_j}\}$ is monotonically increasing.

Lemma 3.3. *Suppose the first assumption of the standard assumptions holds. If there exists an infinite index set $\{k_i\}$ such that $\Delta m < 0$, then $\Phi(x_{k_i}) \to 0$ as $i \to +\infty$.*

Proof. We define $\Phi(x_{k_i}) = max\{0, \|x_{k_i}\|_2 - \delta_{k_i}\} + \left|x_{k_i}^T x_{k_i} - 1\right|$. Now assume that there exists a subsequence $\{k_j\} \subset \{k_i\}$ and a constant ε_1 such that

$$\Phi(x_{k_j}) \geq \varepsilon_1.$$

Since the first assumption of the standard assumptions holds, $\{\Phi(x_{k_j})\}$ is bounded. Thus, there exists a further subsequence $\{k_l\} \subset \{k_j\}$ such that

$$\lim_{l \to +\infty} \Phi(x_{k_l}) = \Phi(\bar{x}) \text{ and } \Phi(\bar{x}) \geq \varepsilon_1,$$

where $\bar{x}$ is an accumulation point of $\{x_{k_l}\}$. According to the hypothesis of the lemma and Algorithm 2.4, for each l, $\Phi(x_{k_l}) = 0$ holds. It is contrary to the hypothesis $\Phi(x_{k_l}) \geq \varepsilon_1$. Hence, $\Phi(x_{k_i}) \to 0$ as $i \to +\infty$. □

Lemma 3.4. *Suppose the first assumption of the standard assumptions holds. If the main sequence $\{x_k\}$ only contains a limited number of $\Delta m < 0$, then there exists an infinite index set $\{k_s\}$ such that $\Phi(x_{k_s}) \to 0$ as $s \to +\infty$.*

Proof. If the main sequence $\{x_k\}$ only contains a limited number of $\Delta m < 0$, then there exists constants $j_1, \varepsilon_2 > 0$ such that $\Delta m \geq \varepsilon_2$ for all $k > j_1$. If there exists $\rho_k > \sigma_0$ for any $k > j_1$, then $x_{k+1} = x_k + d_k$ and $\Delta f \geq \sigma_0 \Delta m \geq \sigma_0 \varepsilon_2$, that is,

$$f(x_{k+1}) - f(x_k) = \Delta f \geq \sigma_0 \varepsilon_2, \quad \forall k > j_1.$$

Obviously, $\{f(x_k)_{k>j_1}\}$ is monotonically increasing and bounded. As $k \to +\infty$, $f(x_{k+1}) - f(x_k) \to 0$. This contradicts the assumption stated above. Therefore, $\rho_k > \sigma_0$ can only occur a finite number of times for any $k > j_1$. Further, there exists an infinite index set $\{k_s\}$ such that

$$\rho_{k_s} \leq \sigma_0 < \sigma_1, \quad \forall k_s > j_1.$$

According to Algorithm 2.4, we have $\delta_{k_s+1} = \frac{1}{2}\delta_{k_s}$. As $k_s \to +\infty$, $\delta_{k_s} \to 0$. From the hypothesis of this lemma,

$$\Delta m \geq \varepsilon_2, \quad \forall k > j_1.$$

This implies that the subproblem (2.3) remains consistent. Then $d_{k_s} \to 0$, as $k_s \to +\infty$. Hence, $\Phi(x_{k_s}) \to 0$, $s \to +\infty$. □

Lemma 3.5. *The condition $\phi'(0) + \phi'' \leq 0$ is necessary and sufficient for $\phi(\alpha)$ to achieve a minimum at $\alpha = 1$, where $\phi(\alpha)$ is a quadratic function over the interval $\alpha \in [0, 1]$, $\phi'(0) < 0$. Thus, $\phi(0) - \phi(1) \geq -\frac{1}{2}\phi'(0)$.*

Lemma 3.6. *Let standard assumptions hold, and let $d \neq 0$ be a feasible point of the subproblem (2.3). Then it follows that $\Delta f \geq \min(0, \sigma_0 \varepsilon_3)$, $\left|\frac{1}{2}[(x_k + d)^T(x_k + d) - 1]\right| \leq \frac{1}{2}\delta^2$, where δ refers to the trust-region radius in the trust-region constraint.*

Proof. Given that $d \neq 0$ is a feasible point for the subproblem (2.3) and according to the Algorithm 2.4, then there exists constants $k > 0$, $\varepsilon_3 > 0$ such that

$$\Delta m \geq \varepsilon_3.$$

If $\rho_k > \sigma_0$, then $\Delta f \geq \sigma_0 \Delta m \geq \sigma_0 \varepsilon_3$. Otherwise, $\Delta f = 0$. Since $\|d\|_2^2 \leq \delta^2$, thus

$$\left|\frac{1}{2}[(x_k + d)^T(x_k + d) - 1]\right| = \left|\frac{1}{2}(x_k^T x_k - 1) + x_k^T d + \frac{1}{2} d^T d\right| \leq \frac{1}{2} d^T d = \frac{1}{2}\delta^2.$$

□

Lemma 3.7. *Under the standard assumptions, suppose x^* is a feasible point of (1.2) but not an optimal one, the MFCQ holds at x^*, then there exist a neighborhood $\mathcal{N}$ of x^* and a constant ε_4 such that, for $\forall x \in \mathcal{N} \cap D$, the subproblem (2.3) has a feasible solution d, which d satisfies $\Delta m \geq \frac{1}{2}\delta\varepsilon_4$, $\Delta f \geq \frac{1}{2}\delta\varepsilon_4\sigma_0$, where $\forall \delta < \tau$, $\tau = \min(\sqrt[3]{\frac{\varepsilon}{M}}, \bar{\delta})$ and $\bar{\delta}$ satisfies $\frac{M}{2}\bar{\delta} + C_1(\bar{\delta}) \leq \frac{(1-\sigma_0)\varepsilon_4}{2}$.*

Proof. Due to x^* is a feasible point of (1.2) and is not optimal, at which MFCQ holds, there exist d with $\|d\| = 1$, and a neighborhood $\mathcal{N}$ of x^* and $\varepsilon_4 > 0$ such that

$$\nabla f(x)^T d > \varepsilon_4,$$
$$x^T d = 0,$$

where $\nabla f(x)$ is the gradient for $\forall x \in \mathcal{N} \cap \{x | \frac{1}{2}(x^T x - 1) = 0\}$.
Part A: Let $d_\alpha = \alpha\delta d, \alpha \in [0, 1]$. Then we have $\|d_\alpha\|_2 \leq \delta$, which means d_α satisfies the trust

region constraint of the subproblem (2.3). Since $x \in \mathcal{N} \cap \{x|\frac{1}{2}(x^T x-1)=0\}$, $\frac{1}{2}(x^T x-1)=0$, $\frac{1}{2}(x^T x-1)+\alpha\delta d^T x=0$. So d_α satisfies the equality constraint of the subproblem (2.3), thus the subproblem (2.3) is consistent.
Part B: We consider the predicted increasement Δm. Define $\phi(\alpha)=-m(d_\alpha)$, then $\phi'(\alpha)=-\delta g^T d-\alpha\delta^2 d^T W d$, $\phi'(0)=-\delta d^T g=-\delta d^T\nabla f(x)-\lambda x^T d<-\delta\varepsilon_4$, $\phi''(\alpha)=-\delta^2 d^T W d\leq \delta^2\|d\|^2\|W\|\leq\delta^4 M$. Thus,

$$\phi'(0)+\phi''<-\delta\varepsilon_4+\delta^4 M\leq 0, \qquad if \quad \delta\leq\sqrt[3]{\frac{\varepsilon_4}{M}}.$$

Based on Lemma 3.5 and Lemma 3.6, it can be concluded that the minimum value of $\phi(\alpha)$ is achieved at $\alpha=1$ and $\phi(0)-\phi(1)\geq-\frac{1}{2}\phi'(0)$, that is $\Delta m:=m(\delta d)-m(0)$. Therefore, $\Delta m\geq\frac{1}{2}\delta\varepsilon_4$. Since

$$|\rho_k-1|=\left|\frac{f(x_k+d)-m_k(d)}{m_k(d)-m_k(0)}\right|,$$

thus,

$$\begin{aligned}|\rho-1| &= \left|\frac{\int_0^1 d_\alpha^T[\nabla f(x+td_\alpha)-\nabla f(x)]dt+\lambda x^T d_\alpha-\frac{1}{2}d_\alpha^T W d_\alpha}{m_k(d_\alpha)-m_k(0)}\right| \\ &= \left|\frac{\int_0^1 d_\alpha^T[\nabla f(x+td_\alpha)-\nabla f(x)]dt-\frac{1}{2}d_\alpha^T W d_\alpha}{m_k(d_\alpha)-m_k(0)}\right| \\ &= \frac{\frac{M}{2}\|d_\alpha\|^2+C_1(d_\alpha)\|d_\alpha\|}{m_k(d_\alpha)-m_k(0)},\end{aligned}$$

where we can achieve an arbitrarily small value for the scalar $C_1(d_\alpha)$ by limiting the size of d_α. If we set $\alpha=1$, the above proof still holds and we have

$$|\rho-1|\leq\frac{\frac{M}{2}\delta^2+C_1(\delta)\delta}{\frac{1}{2}\delta\varepsilon_4}.$$

It can be observed that the right-hand-side is bounded for all sufficiently small values of δ. By choosing $\bar{\delta}$ to be small enough and $\delta\leq\bar{\delta}$, we can ensure

$$\frac{M}{2}\delta+C_1(\delta)\leq\frac{(1-\sigma_0)\varepsilon_4}{2}.$$

Therefore, we have

$$|\rho-1|\leq 1-\sigma_0,$$

that is, $\rho>\sigma_0$. According to Algorithm 2.4, we have $\frac{\Delta f}{\Delta m}>\sigma_0$. Hence,

$$\Delta f\geq\frac{1}{2}\delta\varepsilon_4\sigma_0, \qquad if \quad \delta\leq\bar{\delta}.$$

In brief, for $\delta\leq\tau$, δd is a feasible point of the subproblem (2.3) and $\Delta f\geq\frac{1}{2}\delta\varepsilon_4\sigma_0$, where $\tau=\min(\sqrt[3]{\frac{\varepsilon}{M}},\bar{\delta})$. □

Lemma 3.8. *Under the standard assumptions, then the iteration loop (Step 1→ Step 2→ Step 3→ Step 4→ Step 5→ Step 1) terminates in a finite number of iterations.*

Proof. If x_k is a KKT point of the problem (1.2), then $d = 0$ is the solution of the subproblem (2.3) and the algorithm is terminated. Otherwise, there are two cases to consider.

Case(i): If $d \neq 0$ solves the subproblem (2.3), but $\lambda_{x_k+d} \geq \eta$ according to different examples adjust η, then the algorithm is terminated.

Case(ii): If the iteration loop does not conclude within a finite number of steps, then $\delta_k \to 0$, thus $d \to 0$, and the algorithm is terminated. □

Theorem 3.9. *The algorithm yields one of the following outcomes:*

(1).A better point of the problem (1.2) *that can jump out of the local area is found.*

(2).A KKT point of the problem (1.2) *is founded.*

(3).Let x^* be an accumulation point of the sequence $\{x_k\}$. If the MFCQ holds at x^*, and the algorithm terminates within a finite number of iterations, then x^* is a strong stationary point of the problem (1.2).

Proof. We only need to consider the case (1) and case (3).

(1).Based on the process of Algorithm 2.4, if the third termination condition ($\lambda_k \geq \eta$) is satisfied, a superior point that jumps out of the local optimal area can be found.

(3).By Lemma 3.8, the iteration loop is always terminated within a finite number of iterations. As all iteration points x belong to a bounded set D, the sequence will have one or more accumulation points.

We consider the sequence $\{x_k\}$. Because D is bounded, then there exist an accumulation point x^* and a subsequence $\{x_{k_j}\}$ of $\{x_k\}$, where $\{x_{k_j}\} \to x^*$.

• If $\{x_{k_j}\}$ is feasible, then x^* is a feasible point.

• If $\{x_{k_j}\}$ is infeasible, then x^* is an infeasible point.

There exists an index k_0 such that all iterations satisfies $f_{k+1} \geq f_k$ for all $k \geq k_0$. By assumptions 3.1, the subsequence $\{f_{k_j}\}$ of the sequence $\{f_k\}$ is monotonically increasing for $k_j > k_0$. Due to $f(x)$ is bounded on D, $\Sigma_k \Delta f_k$ is convergent and $x_k \to x^*$. For sufficient large $k > k_0$, $x^* \in \mathcal{N} \cap D$. As assumed in Lemma 3.6, if x^* is not a KKT point of the problem (1.2) and the MFCQ holds at x^*. By Lemma 3.7, there exists a problem (2.3) which can generate a feasible step d and $\Delta f \geq \frac{1}{2}\delta\varepsilon_4\sigma_0$. It contradicts with the fact that $\Sigma_k \Delta f_k$ is convergent. It follows from Lemma 3.3 and Lemma 3.4 that x^* is a feasible point. By the definition 3.1, when x^* is feasible, x^* is a strong stationary point of the problem (1.2). □

4 Numerical Experiments

In this section, we verify the performance of the proposed global optimization strategies. Specifically, TFTR uses the infeasible trust-region algorithm for preprocessing and then enters FTR for calculation. RFTR is preprocessed by the multi-initial points algorithm and then enters FTR for calculation. MFTR is preprocessed by the simulated annealing algorithm and then enters FTR for calculation. We verify the effectiveness of the proposed optimization strategies by comparing them.

All the experiments are preformed on a Lenovo desktop with Intel(R)Core(TM)i5-3470 CPU at 3.20 GHz, OS 32-bit. Our codes are implemented in MATLAB R2014a. The parameters of the infeasible trust-region algorithm are set to be

$$\sigma_0 = 0.1,\ \sigma_1 = 0.25,\ \sigma_2 = 0.75,\ \epsilon = 1e^{-6},\ \delta_0 = 2, \tag{4.1}$$

where ϵ is the tolerance of the stopping criterion. η, $itermax$ are constants which refer to the threshold and maximum number of iterations, respectively. Their values depend on different

situations and examples. The stopping criterion of the infeasible trust-region algorithm is

$$k \geq itermax \tag{4.2}$$

or

$$\|d_k\|_2 \leq \epsilon \tag{4.3}$$

or

$$\lambda_k \geq \eta \tag{4.4}$$

or

$$\left|(x_k^T d_k - \frac{1}{2}(1 - x_k^T x_k)\right| \geq \epsilon,\ \|d_k\|_2 - \delta_k \geq \epsilon \tag{4.5}$$

If one of the four stop criteria is satisfied, the algorithm will be stopped. If the condition (4.2) is satisfied, the maximum number of iterations is exceeded. If the condition (4.3) is satisfied, the solution $d_k \to 0$ of the subproblem (2.3). If the condition (4.4) is satisfied, $\lambda_k \geq \eta$ is found in the process of pretreatment, where η is obtained based on the λ found by FTR. For example, $\eta = 0.8\lambda$. If the condition (4.5) is satisfied, the solution of the subproblem (2.3) is infeasible. The number of initial points in the multi-initial points algorithm is set to be

$$l = 10. \tag{4.6}$$

The parameters of the simulated annealing algorithm are set to be

$$T = 1000,\ Tmin = 1e^{-2},\ \alpha = 0.5,\ \gamma = 10,\ \zeta, \tag{4.7}$$

where ζ is a constant depending on different situations and examples and γ is the maximum number of iterations. The stopping criterion of the simulated annealing algorithm is

$$T \leq Tmin \quad or \quad k > \gamma. \tag{4.8}$$

After the above global optimization strategies, we use FTR to solve the subproblem (2.1). The parameters of FTR are

$$\sigma_0 = 0.1,\ \sigma_1 = 0.25,\ \sigma_2 = 0.75,\ \epsilon = 1e^{-6},\ \Delta_0 = 2. \tag{4.9}$$

The stopping criterion for FTR is

$$\|d_k\|_2 \leq \epsilon. \tag{4.10}$$

The initial points in the four models (TFTR, RFTR, MFTR, FTR) are randomly chosen. We tested the four models for the following examples. For some testing examples, the accurate extreme Z-eigenvalues can be analytically computed.

In the infeasible trust-region algorithm, when the trust-region constraint and the linear constraint on the subproblem (2.3) is consistent, we use the interior-point method combined with the Global function in MATLAB to solve the trial steps d_k. If the constraints are inconsistent, we use the conjugate gradient method from *Numerical Optimization* to obtain d_k [19].

The numerical results of four examples are given below.

Example 4.1. Calculate the largest Z-eigenvalue of the 4th order n-dimensional diagonal tensor A.

$$\mathcal{A}_{i_1=i_2=i_3=i_4} = 10n$$

Table 1: The comparison of the four algorithms results for example 4.1

		TFTR			FTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
3	30	97%	3.98	1.25e+1	45%	4.47	2.50e-2
4	40	71%	3.93	3.12e+1	40%	4.25	2.51e-2
5	50	65%	3.97	3.66e+1	35%	4.15	2.41e-2
6	60	58%	3.68	1.88e+1	30%	4.21	2.52e-2
7	70	59%	3.43	5.02e+1	27%	4.03	2.66e-2
8	80	48%	3.56	3.48e+1	23%	3.90	3.54e-1

		RFTR			MFTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
3	30	100%	2.50	2.61e-2	100%	4	3.30e-1
4	40	100%	2.45	2.43e-2	99%	4.02	3.31e-1
5	50	100%	3.01	2.71e-2	96%	3.95	3.34e-1
6	60	98%	4.48	3.07e-2	92%	4.21	6.43e-1
7	70	97%	3.27	3.12e-2	89%	3.95	6.49e-1
8	80	92%	5.33	3.62e-2	85%	4.10	3.35e-1

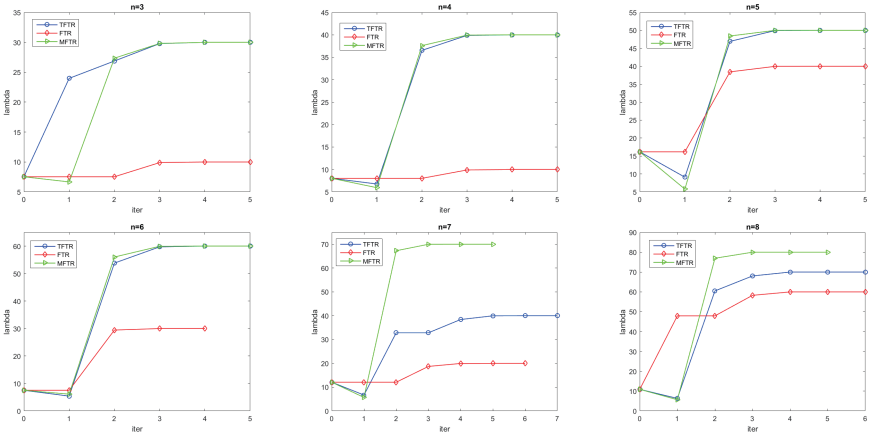

Figure 1: Z-eigenvalues obtained by each iteration under a random initial point when $m = 4$, $n = 3 : 1 : 8$

Figure 1 depicts the Z-eigenvalues ($n = 3, 4, 5, 6, 7, 8$) obtained at each iteration for the three algorithms (TFTR, MFTR, FTR) in a group of experiments, all implemented from the same initial point.

In the Table 1-4, 'n' is the dimension, 'λ_{max}' is the largest Z-eigenvalue of the example by current algorithms, 'ratio' is the probability of obtaining the largest Z-eigenvalue in 1000 experiments, 'iter' stands for the average iteration numbers of the feasible trust region part, 'time' means the average CPU time of preprocessing stage plus the feasible trust region stage in seconds. In the Figure 1-4, the horizontal axis represents the iterative steps, whereas the vertical axis represents the corresponding Z-eigenvalues.

For diagonal tensors, the maximum value of the elements along the superdiagonal is equal to the largest Z-eigenvalue of the tensor. We can observe from the Table 1 that TFTR, RFTR and MFTR have a significantly higher success rate in obtaining the largest Z-eigenvalue compared to FTR. Except for a few cases, the iteration for TFTR, RFTR, MFTR is less than that of FTR; the CPU time for RFTR is slightly slower than that of FTR; TFTR and MFTR require more time than that of FTR.

Since the multi-initial points algorithm cannot guarantee the same initial point, we compare the Z-eigenvalues obtained by each iteration of TFTR, MFTR and FTR at the same initial point. From Figure 1, it is evident that TFTR and MFTR are more likely to obtain the largest Z-eigenvalue. When $n = 7, 8$, only MFTR can obtain the largest Z-eigenvalue, which is also consistent with the data in Table 1, that is, the success rate of TFTR and MFTR is often higher than that of FTR, but with the increase of dimension, the success rate of MFTR is higher than that of TFTR. In this case, the constraints on the subproblem (2.3) are consistent in the preprocessing of the TFTR.

The following examples are different from the diagonal tensor. For the following tensors, the largest Z-eigenvalue of the tensor cannot be determined. The success rate below refers to the success rate of obtaining the largest Z-eigenvalue that can be calculated by the current algorithms.

Example 4.2. Calculate the largest Z-eigenvalue of the 3th order n-dimensional symmetric tensor A.

$$\mathcal{A}_{i_1,i_2,i_3} = \frac{(-1)^{i_1}}{i_1} + \frac{(-1)^{i_2}}{i_2} + \frac{(-1)^{i_3}}{i_3}$$

Table 2: The comparison of the four algorithms results for example 4.2

		TFTR			FTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
10	1.7800e+01	100%	3.60	5.03	55%	3.93	3.17e-2
20	3.4159e+01	100%	3.40	1.28e+1	54%	4.25	4.00e-2
30	5.0138e+01	93%	3.47	4.48e+1	54%	4.45	2.77e-1
40	6.5926e+01	76%	3.62	8.53e+1	52%	4.66	9.78e-1
50	8.1593e+01	74%	3.58	1.42e+1	54%	4.59	9.84e-1
60	9.7177e+01	81%	3.68	2.39e+1	51%	4.51	1.21
70	1.1270e+02	85%	3.64	3.87e+1	52%	4.50	1.00
80	1.2817e+02	84%	3.71	6.45e+1	50%	4.62	1.08

		RFTR			MFTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
10	1.7800e+01	100%	4.63	3.88e-2	100%	5	3.25e-1
20	3.4159e+01	97%	3.62	4.86e-2	100%	5	3.31e-1
30	5.0138e+01	98%	3.02	3.90e-2	100%	5	3.80e-1
40	6.5926e+01	90%	3.05	4.46e-2	100%	5	3.62e-1
50	8.1593e+01	79%	3.11	4.05e-2	100%	5	3.63e-1
60	9.7177e+01	82%	3.21	4.06e-2	100%	5	3.72e-1
70	1.1270e+02	80%	3.04	4.30e-2	100%	5	3.88e-1
80	1.2817e+02	79%	3.56	5.93e-2	100%	5	4.47e-1

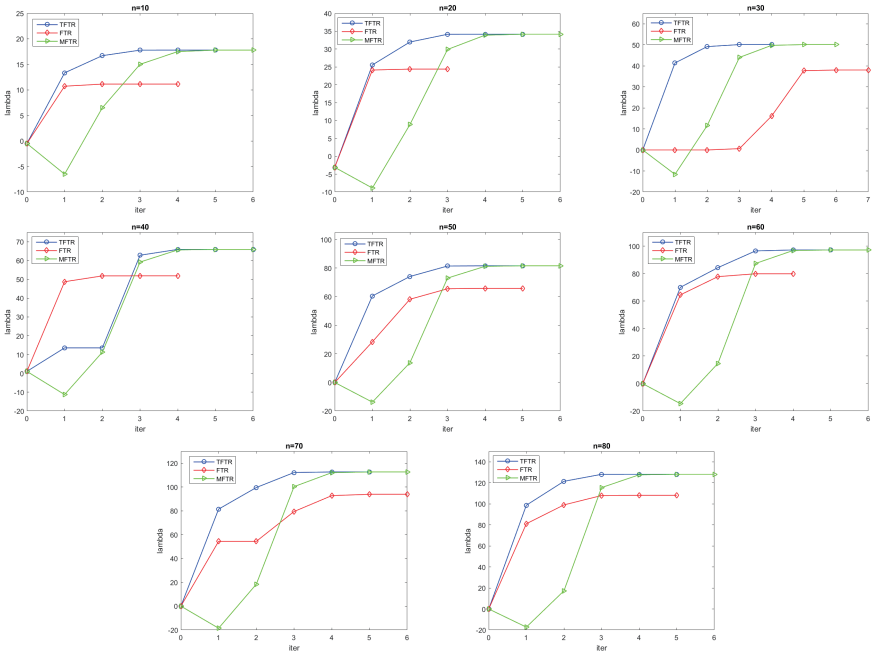

Figure 2: Z-eigenvalues obtained by each iteration under a random initial point $m = 3$, $n = 10 : 10 : 80$

Select a group of experiments, and use the same initial point for the three algorithms (TFTR, MFTR, FTR) to compare the Z-eigenvalues obtained in each iteration ($n = 10, 20, 30, 40, 50, 60, 70, 80$). Please refer to Figure 2.

Table 2 suggests that the success rate of TFTR, RFTR and MFTR for computing the largest Z-eigenvalue is also much higher than that of FTR. The number of iterations required for TFTR and RFTR is often lower than that of FTR. TFTR requires more time than that of FTR; the CPU time of both RFTR and MFTR is slightly slower than that of FTR when $n \leq 30$; the CPU time of RFTR and MFTR is quicker than that of FTR when $n \geq 30$. It is clear that the success rate of MFTR is reached 100%, whereas the success rates of TFTR and RFTR are not as high as that of MFTR. They are still significantly higher than that of FTR.

As illustrated in Figure 2, both TFTR and MFTR have a higher success rate than FTR when it comes to obtaining the largest Z-eigenvalue. TFTR performs much faster than MFTR when $n = 20, 30, 50, 60, 70, 80$. Similarly, the constraints on the subproblem remain consistent during the preprocessing of the TFTR.

Example 4.3. Calculate the largest Z-eigenvalue of the 4th order n-dimensional symmetric tensor A.

$$\mathcal{A}_{i_1,i_2,i_3,i_4} = \sin(i_1 + i_2 + i_3 + i_4)$$

Figure 3 depicts the Z-eigenvalues ($n = 10, 20, 30, 40, 50, 60$) obtained at each iteration for the three algorithms (TFTR, MFTR, FTR) in a group of experiments, all implemented from the same initial point.

For this example, it is found that the simulated annealing algorithm 2.2 performs bad at the cases of $n = 10, 30, 60$. The simulated annealing algorithm 2.3 has better numerical

Table 3: The comparison of the four algorithms results for example 4.3

		TFTR			FTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
10	2.7243e+01	56%	6.00	1.03e+2	51%	6.89	4.19e-2
20	1.0180e+02	51%	6.73	4.86e+1	50%	7.80	5.93e-2
30	2.3095e+02	56%	5.65	5.47e+1	49%	7.16	7.28e-2
40	4.1438e+02	54%	6.10	1.64e+2	52%	7.66	1.57e-1
50	6.3281e+02	53%	7.68	1.43e+2	50%	7.51	2.42e-1
60	9.0942e+02	55%	7.82	2.13e+2	51%	8.26	4.22e-1

		RFTR			MFTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
10	2.7243e+01	90%	5.36	4.32e-2	53%	6.00	1.14
20	1.0180e+02	86%	4.70	4.60e-2	79%	11.14	7.47e-1
30	2.3095e+02	90%	3.34	5.50e-2	51%	6.86	2.15
40	4.1438e+02	78%	3.73	1.01e-1	100%	7.14	1.26
50	6.3281e+02	55%	4.02	2.14e-1	32%	7.59	7.22
60	9.0942e+02	46%	4.00	3.74e-1	49%	7.33	8.31

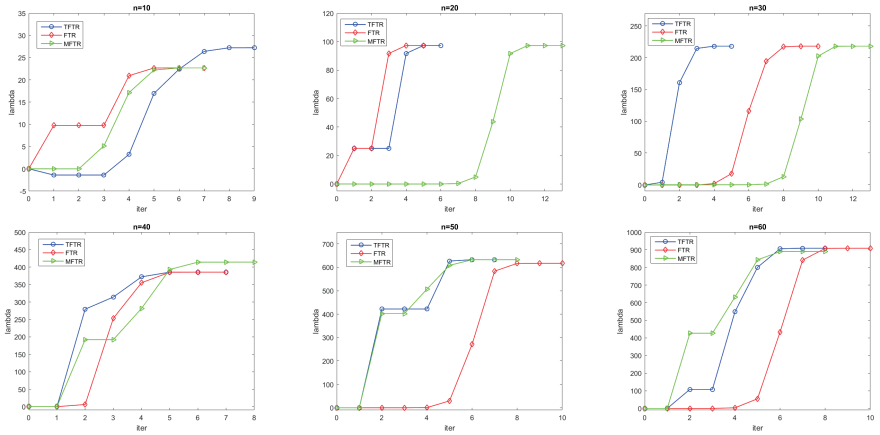

Figure 3: Z-eigenvalues obtained by each iteration under a random initial point $m = 4$, $n = 10 : 10 : 60$

results, thus we use the algorithm 2.3 for the cases of $n = 10, 30, 60$. As shown in Table 3, the success rate of TFTR is slightly higher than that of FTR. Except for $n = 50, 60$, the success rate of both RFTR and MFTR is higher than that of FTR. With the exception of some cases of MFTR ($n = 20, 50$), the iteration for all TFTR, MFTR and RFTR is lower than that of FTR; the CPU time of RFTR is slightly quicker than that of FTR, the time of TFTR and MFTR is slower than that of FTR.

It can be seen from Figure 3 that TFTR and MFTR are often easier to obtain the largest Z-eigenvalue than FTR. The number of iterations of TFTR is less than that of FTR when $n = 30, 50, 60$, the number of iterations of MFTR is more than that of FTR when

$n = 20, 30, 40$, MFTR takes less iterations than FTR when $n = 50, 60$. The above conforms to the data in Table 3. Unlike the previous examples, the subproblem (2.3) is occasionally inconsistent during the preprocessing of the TFTR, but this is not a common occurrence. We solve this problem using the conjugate gradient method [19].

Example 4.4. Calculate the largest Z-eigenvalue of the 5th order n-dimensional symmetric tensor A.

$$\mathcal{A}_{i_1,i_2,i_3,i_4,i_5} = (-1)^{i_1} ln(i_1) + (-1)^{i_2} ln(i_2) + (-1)^{i_3} ln(i_3) + (-1)^{i_4} ln(i_4) + (-1)^{i_5} ln(i_5)$$

Table 4: The comparison of the four algorithms results for example 4.4

		TFTR			FTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
5	1.1001e+02	99%	5.44	3.74e+1	55%	5.33	3.10e-2
10	8.8328e+02	79%	4.04	3.29e+1	54%	5.26	3.47e-2
20	6.2367e+03	58%	4.79	1.18e+1	51%	5.12	1.07e-1
30	1.9439e+04	51%	4.62	5.52	49%	5.10	1.65

		RFTR			MFTR		
n	λ_{max}	ratio	iter	time	ratio	iter	time
5	1.1001e+02	100%	3.84	3.10e-2	100%	5.26	3.57e-1
10	8.8328e+02	87%	5.00	3.93e-2	100%	5.07	7.84e-1
20	6.2367e+03	81%	4.02	1.40e-1	100%	4.07	3.08
30	1.9439e+04	71%	4.01	1.38	52%	3.89	1.10e+1

Select a group of experiments, and use the same initial point for the three algorithms (TFTR, MFTR, FTR) to compare the Z-eigenvalues obtained in each iteration ($n = 5, 10, 20, 30$). Please refer to Figure 4.

From Table 4, we can see that iteration of TFTR is less than that of FTR besides $n = 5$, the time of all TFTR, MFTR and RFTR is generally slower than that of FTR. In this example, the subproblem constraints are still consistent during the preprocessing of TFTR. It can be seen from Figure 4 that TFTR and MFTR are easier than FTR to obtain the largest Z-eigenvalue. In particular, when $n = 20, 30$, MFTR is much easier to obtain the largest Z-eigenvalue, which is consistent with the data in Table 4.

Based on the test cases above, it can be found that TFTR, MFTR and RFTR are all capable of computing the largest Z-eigenvalue. Moreover, these methods generally exhibit a higher success rate for obtaining the largest Z-eigenvalue compared to FTR. Except example 4.2, the three algorithms generally take less iterations than that of FTR, and the CPU time of all RFTR, TFTR and MFTR is generally slower than that of FTR algorithm.

5 Conclusions

We have proposed three global optimizations preprocessing strategies based on the feasible trust-region method, which improves the success rate of the largest (smallest) Z-eigenvalue. It can offer a greater likelihood of evading saddle points and local maxima/minima during

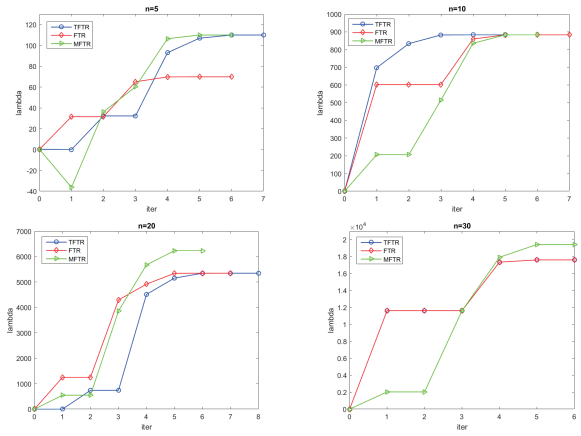

Figure 4: Z-eigenvalues obtained by each iteration under a random initial point $m = 5$, $n = 5, 10 : 10 : 30$

the optimization process. Under certain assumptions, we prove the convergence of the infeasible trust-region algorithm. In our numerical experiments, it is found that TFTR, RFTR, and MFTR generally exhibit a higher success rate than FTR for obtaining the largest Z-eigenvalue. This is a clear indication that our strategies are highly effective. Our results demonstrate a significant improvement in the success rate of FTR by employing the global optimization strategy.

References

[1] M. Cao, Q. Huang and Y. Yang, A self-adaptive trust region method for extreme B-eigenvalues of symmetric tensors, *Numer. algorithms*. 81 (2019) 407–420.

[2] M. Cao and Y. Yang, A nonmonotone accelerated Levenberg-Marquardt method for the B-eigenvalues of symmetric tensors, *Int. Trans. Oper. Res.* 29 (2021) 113–129.

[3] Y. Chen, Y. Dai, D. Han and W. Sun, Positive semidefinite generalized diffusion tensor imaging via quadratic semidefinite programming, *SIAM J. Imaging Sci.* 6 (2013) 1531–1552.

[4] C. Cui, Y. Dai and J. Nie, All real eigenvalues of symmetric tensors, *SIAM J. on Matrix Anal. Appl.* 35 (2014) 1582–1601.

[5] C. Hao, C. Cui and Y. Dai, A sequential subspace projection method for extreme Z-eigenvalues of supersymmetric tensors, *Numer. Linear Algebra Appl.* 22 (2015) 283–298.

[6] C. Hao, C. Cui and Y. Dai, A feasible trust-region method for calculating extreme Z-eigenvalues of symmetric tensors, *Pac. J. Optim.* 11 (2015) 291–307.

[7] M.L. Che, A. Cichocki and Y.M. Wei, Neural networks for computing best rank-one approximations of tensors and its applications, *Neurocomputing*. 267 (2017) 114–133.

[8] S. Hu, Z. Huang and L. Qi, Finding the extreme Z-eigenvalues of tensors via a sequential semidefinite programming method, *Numer. Linear Algebra Appl.* 20 (2013) 972–984.

[9] E. Kofidis and P. Regalia, On the best rank-1 approximation of higher-order supersymmetric tensors, *SIAM J. on Matrix Anal. Appl.* 23 (2002) 863–884.

[10] T. Kolda and J. Mayo, Shifted power method for computing tensor eigenpairs, *SIAM J. on Matrix Anal. Appl.* 32 (2011), 1095–1124.

[11] T. Kolda and J. Mayo, An adaptive shifted power method for computing generalized tensor eigenpairs, *SIAM J. on Matrix Anal. Appl.* 35 (2014), 1563–1581.

[12] J. Lasserre, Global optimization with polynomials and the problem of moments, *SIAM J. Optim.* 11 (2001) 796–817.

[13] C. Li and Y. Li, An eigenvalue localization set for tensors with applications to determine the positive (semi-)definiteness of tensors, *Linear Multilin. Algebra.* 64 (2016), 587–601.

[14] L. Lim, Singular values and eigenvalues of tensors: a variational approach, in: Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing(CAMSAP'05), Vol.1, IEEE Computer Society Press: Piscataway, New Jersey, 2005, pp.129–132.

[15] X. Liu, G. Perakis and J. Sun, A robust SQP method for mathematical programs with linear complementarity constraints, *Comput. Optim. Appl.* 34 (2006) 5–33.

[16] Q. Ni, L. Qi and F. Wang, An eigenvalue method for testing positive definiteness of a multivariate form, *IEEE Trans. Automat. Control.* 53 (2008), 1096–1107.

[17] J. Nie and L. Wang, Semidefinite relaxations for best rank-1 tensor approximations, *SIAM J. on Matrix Anal. Appl.* 35 (2014) 1155–1179.

[18] J. Nie and X. Zhang, Real eigenvalues of nonsymmetric tensors, *Comput. Optim. Appl.* 70 (2018) 1–32.

[19] J. Nocedal and S. Wright, *Numerical Optimization*, Springerverlang, USA, 1999.

[20] L. Qi, Eigenvalues of a real supersymmetric tensor, *J. Symbolic Comput.* 40 (2005) 1302–1324.

[21] L. Qi and K. Teo, Multivariate polynomial minimization and its application in signal processing, *J. Global Optim.* 26 (2003, 419–433.

[22] L. Qi, G. Yu and E. Wu, Higher order positive semidefinite diffusion tensor imaging, *SIAM J. Imaging Sci.* 3 (2010) 416–433.

[23] G. Yu, Z. Yu, Y. Xu and Y. Song, An adaptive gradient method for computing generalized tensor eigenpairs, *Comput. Optim. Appl.* 65 (2016) 781–797.

[24] X.W. Liu and Y.X. Yuan, A robust algorithm for optimization with general equality and inequality constraints, *SIAM J. Sci. Comput.* 22 (2000) 517–534.

[25] X. Zhang, C. Ling and L. Qi, The best rank-1 approximation of a symmetric tensor and related spherical optimization problems, *SIAM J. on Matrix Anal. Appl.* 33 (2012) 806–821.

[26] S. Zhou and N. Qin, Computing tensor Z-eigenvalues via shifted inverse power method, *J. Comput. Appl. Math.* 398 (2021): 113717.

Manuscript received 22 June 22 2022
revised 9 August 2023
accepted for publication 12 August 2023

Xiao Li
College of Information and Electrical Engineering
China Agricultural University, Beijing 100083, China
E-mail address: xiaoli_2019@163.com

Chun-Lin Hao
School of Mathematics, Faculty of Science
Beijing University of Technology, Beijing 100124,P.R. China
E-mail address: haochl@bjut.edu.cn

Yitian Xu
College of Science, China Agricultural University
Beijing 100083, China
E-mail address: xytshuxue@126.com

Jun-Yu Gao
The Mathematics and Physics teaching Department
Tianjin TianShi College, Tianjin 301700
E-mail address: 350216867@qq.com

Zu-Ping Li
The Mathematics, Chinese and Physical Education Department
WeiFang Vocational College, WeiFang 261041
E-mail address: lizuping2007@163.com