Engineering with Computers (2024) 40:2595–2616
https://doi.org/10.1007/s00366-023-01928-2

ORIGINAL ARTICLE

Global optimization of optimal Delaunay triangulation with modified whale optimization algorithm

Yongjia Weng[1,2] · Juan Cao[1,2] · Zhonggui Chen[3]

Received: 7 June 2023 / Accepted: 20 November 2023 / Published online: 29 January 2024

Abstract
In this paper, we introduce an innovative approach to generate a high-quality mesh with a density function in a given domain. Our method involves solving a variational problem that optimizes the energy function of the optimal Delaunay triangulation (ODT). To achieve this, we have developed a modified whale optimization algorithm (MWOA) based population that is combined with the quasi-Newton method (L-BFGS) to optimize ODT energy on a global level. Our experiments have demonstrated the impressive efficiency of this optimization algorithm in searching for better minima and producing high-quality meshes. Remarkably, the algorithm's powerful global optimization capability makes it insensitive to initialization, which eliminates the need for any special initialization procedures. Furthermore, our proposed algorithm can easily handle complex domains and non-uniform density functions, making it a versatile tool for mesh generation. Overall, our method offers a promising solution for generating practicable meshes with a density function.

Keywords Mesh generation · Optimal Delaunay triangulation · Modified whale optimization algorithm · Global optimization

1 Introduction

Simplicial meshes are commonly used in the applications of computer graphics, computer-aided engineering [1–3], as well as computational medicine and biology [4, 5]. Triangular/Tetrahedral meshes are more unstructured and could be adapted to arbitrary geometries with high accuracy. However, unstructured meshing is not a trivial problem. There have been many surveys, papers and books talked about this topic [6, 7]. The well-known methods of unstructured mesh generation can be divided into four categories [8, 9]: advancing front method, quad/octree-based method, Delaunay method and variational approaches. Advancing front method [10, 11] and quad/octree-based method [12, 13] can generate meshes quickly, but meshes produced by these approaches are either of poor quality in the interior or on the boundary [9]. Delaunay method [14, 15] generally divides into two parts according to how they deal with the boundary constraints. Conforming Delaunay method splits the boundary to meet Delaunay conditions while constrained Delaunay method relaxes Delaunay conditions to hold the boundary constraints. However, conforming method may result in over refinement and constrained method may get poor-quality elements around the boundary. Nevertheless, mesh quality, including factors such as minimal/maximal dihedral angle, average radius ratio, etc. [16, 17] plays a significant role in influencing the accuracy and stability of numerical simulations, such as finite element analysis.

Variational approaches [18, 19] could obtain better quality meshes due to minimizing the quality-defined energy function. Most notably, the concept of ODT energy combines the vertex position and topology optimization in the function approximation [20]. However, the ODT energy is notoriously difficult to optimize as it has lots of

✉ Zhonggui Chen
chenzhonggui@xmu.edu.cn

Yongjia Weng
wengyongjia@stu.xmu.edu.cn

Juan Cao
juancao@xmu.edu.cn

[1] School of Mathematical Sciences, Xiamen University, Xiamen 361005, Fujian, China

[2] Fujian Provincial Key Laboratory of Mathematical Modeling and High-Performance Scientific Computation, Xiamen University, Xiamen 361005, Fujian, China

[3] School of Informatics, Xiamen University, Xiamen 361005, Fujian, China

local minimizers. To date, few studies have focused on the global optimization of ODT energy. Previous studies encountered the problem of balancing optimization time and mesh quality, especially when the initialization is poor [21].

In this paper, we develop a global optimization algorithm for minimizing the ODT energy to obtain much improved local minimal sites faster than previous works, especially when the initialization is far away from the optimal solution. Due to the C^0 continuity and nonconvexity of the ODT energy, it is difficult to get its global minimum by traditional minimization approaches like Newton method, and extremely easy to get stuck at a local minimum. Especially when the topology changes, ODT energy decreases greatly, which is similar to discontinuous function. The whale optimization algorithm (WOA) [22] is appropriate for both continuous and discrete search space and has been successfully used in many fields, like image segmentation [23], which has great potential for global optimization of ODT energy. However, it is non-trivial to apply the WOA to global optimization of ODT energy. The local optimization capability of WOA is inadequate for obtaining a high-quality mesh when optimizing ODT energy, in comparison to traditional gradient-based methods. Additionally, the problem's dimension increases significantly with the number of vertices, leading to a vast search space that makes finding the optimal solution challenging. Moreover, the effectiveness of global optimization in WOA depends heavily on the appropriate setting of key parameters. However, when optimizing ODT energy, the default parameter settings of WOA fail to achieve the desired results.

To overcome the above problems, we carefully designed a global optimization algorithm, which significantly improved the computational efficiency and quality. Our major contributions are:

- We propose a novel hybrid optimization algorithm that integrates L-BFGS with MWOA to achieve efficient and fast convergence to the global optimal solution. Our proposed MWOA reduces the search space by using vertex neighborhood information and initializing the design population. Additionally, L-BFGS complements the local optimization capability.
- Based on a thorough analysis of the proposed algorithm, a set of parameters is specifically tailored for ODT energy optimization. Furthermore, the utilization of OpenMP for parallel acceleration significantly boosts the efficiency of the optimization process, leading to improved results.
- Extensive experiments have demonstrated that our proposed hybrid optimization algorithm exhibits superior global optimization capability compared to existing global algorithms for ODT energy optimization. This suggests that our algorithm is well-suited for meshing with density function and does not require any special initialization.

The structure of the remainder of this paper is organized as follows. In Sect. 2, a brief review of ODT, CVT [24], and the global optimization of both is provided. Section 3 focuses on the preliminaries of optimal Delaunay triangulation and the whale optimization algorithm. The problem statement and an overview of our proposed optimization algorithm are presented in Sect. 4. Section 5 provides a detailed description of our optimization algorithm, including the initialization, local search, global search, and speedup strategies. The experimental results and conclusions are discussed in Sects. 6 and 7, respectively.

2 Related work

In this section, our focus is on the relevant literature related to variational approaches for mesh generation and their global optimization. For comprehensive surveys of unstructured meshing, readers are referred to [6] and [7]. Of particular significance in variational methods for mesh generation are the optimal Delaunay triangulation (ODT) and its dual centroidal Voronoi tessellations (CVT) [24]. Efficient and effective global optimization of ODT and CVT is essential. Therefore, we divide this review into three parts: ODT, CVT, and the global optimization of ODT and CVT, to provide a concise overview of the relevant references.

Optimal Delaunay Triangulation ODT, or Optimal Delaunay Triangulation, is a method proposed by Chen and Xu [20, 25] for generating high-quality meshes. It was then implemented in 3D and extended to graded meshes with sizing field and boundary handling [26]. Tournois et al. [27] introduced a composite approach that combines Delaunay refinement and ODT optimization to further improve mesh quality. They also proposed a boundary treatment called natural ODT, which perturbs slivers to improve mesh quality. Another approach to handle boundary vertices is the boundary-optimized Delaunay triangulation proposed by Gao et al. [28], where boundary vertices are allowed to slide along the boundary. In contrast to the one-vertex update used in previous methods [25–27], variants of Newton's method [21, 29] are used to obtain all the updated vertices of the mesh with fast convergence. Additionally, Chen et al. [30] combined edge flip with ODT to further improve the results. Connectivity regularization is introduced in the enhanced ODT method by Hai et al. [31] to escape from poor local minima. Chen et al. [21] observed that ODT generates fewer slivers compared to CVT, providing insights into the advantages of ODT in terms of mesh quality. To generate anisotropic meshes, Chen et al. extended uniform ODT by using interpolation

error to convex function f instead of $||\boldsymbol{x}||^2$ [32]. However, this approach was found to be too restrictive, and Chen et al. [29] presented a density function-based method for generating anisotropic meshes that is not limited to convex functions. Furthermore, Feng et al. [33] extended ODT to curved ODT, which allows for the generation of curved unstructured meshes.

Centroidal Voronoi Tessellations Since the introduction of CVT by Du et al. [24], this concept has found successful applications in various areas, including high-quality meshing [8, 34], point cloud resampling [35], and supervoxel generation [36]. Liu et al. [37] demonstrated the almost C^2 continuity property of the CVT energy function, while Lu et al. [38] proved its nonconvexity. Commonly used approaches to obtain CVT include Lloyd's algorithm [24, 39] and quasi-Newton methods [37]. To extend Euclidean CVT to manifold counterparts, several variations have been proposed, such as geodesic CVT [40, 41], intrinsic CVT [42], and restricted Voronoi diagrams (RVD) [43, 44]. Yan et al. [34] introduced a rapid computation method for clipped Voronoi diagram, enhancing the computational efficiency of CVT within 3D compact domains. Moreover, CVT has also been extended to high-dimensional spaces [45]. Additionally, anisotropic meshes can be generated using CVT with different distance measures [46–49]. Power diagrams have also shown potential for generating anisotropic meshes as an extension of CVT [50]. However, it should be noted that when generating isotropic meshes in 3D, CVT may produce more slivers compared to the ODT method. This is because CVT is based on Voronoi diagrams instead of being directly defined on tetrahedral meshes, which inherently limits its ability to effectively suppress slivers compared to the ODT method [21, 26].

Global optimization of ODT and CVT Lu et al. [38] proposed a global optimization method for computing Euclidean CVT using a Monte Carlo with minimization (MCM) framework, while Liu et al. [40] introduced the manifold differential evolution (MDE) method for obtaining geodesic CVT, which is insensitive to initialization and mesh tessellation. However, optimizing CVT can sometimes result in the generation of more slivers in the mesh. In contrast, ODT has been observed to have sliver-suppressing properties in 3D, and hence optimizing ODT is chosen as the mesh generation method in this paper. Although there is relatively less research on the global optimization of ODT due to the difficulty of optimizing ODT energy, Chen et al. [21] proposed a global ODT method that uses a combination of simulated annealing (SA) and a local solver (L-BFGS). However, this approach can often get stuck in local minima due to the sensitivity of the algorithm to initialization. In this paper, we propose a hybrid algorithm based on WOA for global optimization of ODT energy to obtain a better minimum. WOA is chosen because it can be used for both continuous and discrete space optimization, making it suitable for optimizing the ODT energy, which may have poor continuity. The customized hybrid algorithm aims to overcome the challenges of local minima and sensitivity to initialization in previous methods, and achieve improved results in generating high-quality meshes.

3 Preliminary

In this section, we provide a brief overview of the optimal Delaunay triangulation (ODT) and the whale optimization algorithm (WOA), which are the key concepts utilized in this paper.

3.1 Optimal Delaunay triangulation

In the following, we will introduce ODT via its definition, properties and the computation of energy and gradient.

3.1.1 Definition

Given a compact domain Ω, a finite point set $\boldsymbol{X} \subset \Omega$ and a density function $\rho(\boldsymbol{x})$ defined on Ω, the ODT energy is defined as [29]:

$$E^{\rho}_{\mathrm{ODT}}(\boldsymbol{X}, \mathcal{T}) = \int_{\Omega} \rho(\boldsymbol{x})|(f_{\mathrm{PWL}} - f)(\boldsymbol{x})|\mathrm{d}\boldsymbol{x}. \tag{1}$$

where $\mathcal{T}$ is the triangulation of $\boldsymbol{X}$, $f(\boldsymbol{x}) = \|\boldsymbol{x}\|^2, \boldsymbol{x} \in \Omega$ is the paraboloid function on domain Ω and $f_{\mathrm{PWL}}(\boldsymbol{x})$ is the piecewise linear function interpolating $f(\boldsymbol{x})$ at $\boldsymbol{X}$; see Fig. 1. The ODT energy function measures the deviation between the

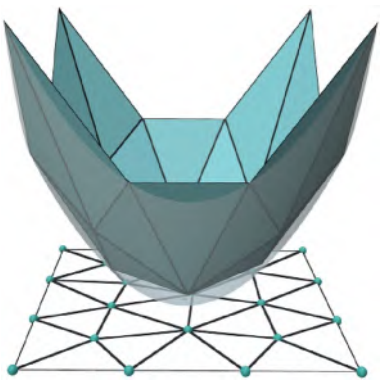


Fig. 1 Function approximation perspective of the ODT energy function. From bottom to top: a 2D triangular mesh $(\boldsymbol{X}, \mathcal{T})$ on the square-shaped domain Ω with vertices (or sites) $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=0}^{n}$ (marked in blue), the paraboloid function $f(\boldsymbol{x})$ and its piecewise linear interpolation f_{PWL}

paraboloid function $f(\boldsymbol{x}) = \|\boldsymbol{x}\|^2$ and its piecewise linear interpolation function $f_{\text{PWL}}(\boldsymbol{x})$ over the triangular mesh $\mathcal{T}$. An ODT is then defined as the triangulation that minimizes the ODT energy.

The ODT energy function is widely used for mesh optimization due to its ability to achieve equidistribution of weighted volumes and edge lengths among all simplices in the triangulation [29]. However, the ODT energy function poses a significant challenge for current optimization algorithms due to its combination of continuous and combinatorial optimization. The energy function involves optimizing the positions of vertices in a continuous space while simultaneously considering the combinatorial aspects of mesh connectivity. The integration of continuous and combinatorial optimization in the ODT energy function presents a substantial challenge for current optimization algorithms to achieve effective minimization and attain optimal mesh configurations.

3.1.2 Properties

Chen et al. conducted a systematic study on the properties of the ODT energy [21], which revealed that the ODT energy function possesses three key properties, namely Delaunay consistency, piecewise C^∞ continuity, and non-convexity.

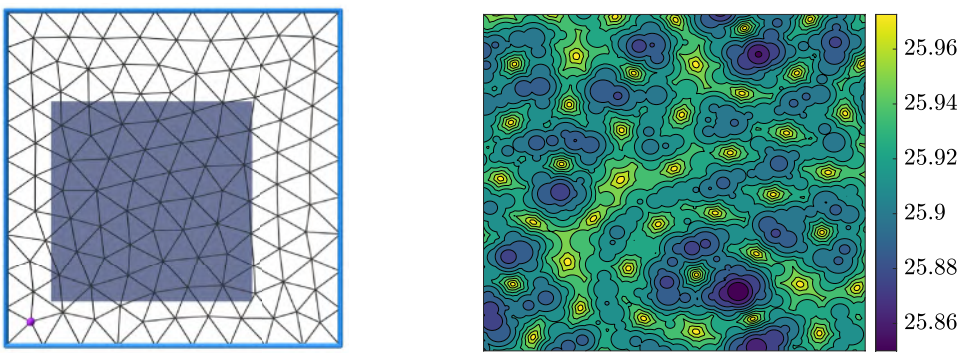
- *Delaunay consistent* It means $E^{\rho}_{\text{ODT}}(\boldsymbol{X}, \mathcal{T})$ achieve the minimum when $\mathcal{T}$ is Delaunay triangulation for a set of fixed vertices $\boldsymbol{X}$, i.e.,

$$\text{DT}(\boldsymbol{X}) = \arg\min_{\mathcal{T}} E^{\rho}_{\text{ODT}}(\boldsymbol{X}, \mathcal{T}), \tag{2}$$

  where $DT(\boldsymbol{X})$ is the Delaunay triangulation of the vertex set $\boldsymbol{X}$. Therefore, the optimization of the ODT energy function can be achieved by iteratively optimizing the vertex set and computing the Delaunay triangulation in an alternating fashion [26]. In other words, the optimization problem for the ODT energy function can be transformed from minimizing $E^{\rho}_{\text{ODT}}(\boldsymbol{X}, \mathcal{T})$ to minimizing $E^{\rho}_{\text{ODT}}(\boldsymbol{X}, \text{DT}(\boldsymbol{X}))$.
- *Piecewise C^∞ continuity* Suppose the density function $\rho(\boldsymbol{x})$ is smooth in Ω, meaning that it exhibits C^∞ continuity throughout the domain. The ODT energy function $E^{\rho}_{\text{ODT}}(\boldsymbol{X}, \text{DT}(\boldsymbol{X}))$ is C^0 continuous everywhere, but it is only piecewise C^∞ continuous. The C^∞ continuity of the function can be lost when the discrete connectivity is modified, as noted by Chen et al. [21]. This property make the ODT energy function prone to getting stuck in local minima when using common optimization algorithms such as Newton's method.
- *Non-convex* Extensive observations have confirmed the highly non-convex nature of the ODT energy function, characterized by the presence of numerous local minima. To provide numerical evidence supporting this claim, we conducted experiments in a 2D uniform scenario. Specifically, we distributed a total of $N = 100$ points in a regular pattern within the square domain $[-5, 5]^2$. Figure 2 illustrates the experimental setup, where only one point is permitted to move within the designated dusty blue region while the remaining points are held fixed. The visualization of the landscape of $E^{\rho}_{\text{ODT}}(\boldsymbol{X}, \text{DT}(\boldsymbol{X}))$ clearly exhibits the existence of numerous shallow local minima, which pose significant challenges in achieving global optimization.

3.1.3 Computational aspects

In optimization problems, the computation of energy and gradient is crucial. In the context of the ODT energy function, the density function is amenable to arbitrary functional forms. Within the confines of our best knowledge, numerical methods capable of exact integration for arbitrary functions remain elusive. When the density function assumes specific, readily integrable values, such as constants, the computation of the ODT energy and its gradient attains a state of exact calculability. For others, the precise computation of the ODT energy function and its gradient is deemed



(a) Based Mesh (b) Energy landscape

Fig. 2 Visualization of the energy landscape with one point moving within the highlighted region. **a** All points are fixed, except for the one marked by the purple dot, which can freely move within the designated dusty blue region. **b** The landscape of the ODT energy function $E^{\rho}_{\text{ODT}}(\boldsymbol{X}, \text{DT}(\boldsymbol{X}))$ as it varies with the position of the moving point

unfeasible. Instead, these quantities can be approximated using advanced numerical integration methods [51] with sufficient accuracy. Another approach is to estimate them by employing piecewise constant [29] or piecewise linear interpolation [21] of the density function, which is a reasonable strategy for smoothly varying density functions. In our framework, we utilize the efficient method proposed by Chen et al. [21] for computing the ODT energy and gradient.

3.2 Symbolic interpretation

For the sake of simplicity and clarity, we first explain some symbols used in this context. The symbol $\cdot$ denotes element-wise multiplication between two matrices, and $|\cdot|$ represents the absolute value of each element in the matrix:

$$|A \cdot B| = \begin{bmatrix} |a_{11}b_{11}| & \dots & |a_{1m}b_{1m}| \\ \vdots & \ddots & \vdots \\ |a_{n1}b_{n1}| & \dots & |a_{nm}b_{nm}| \end{bmatrix}_{n\times m}. \tag{3}$$

$\lfloor \boldsymbol{v} \rceil$ is a copying operator that takes a vector $\boldsymbol{v} \in \mathbb{R}^n$ and creates a matrix with two columns that are identical to $\boldsymbol{v} \in \mathbb{R}^n$:

$$\lfloor \boldsymbol{v} \rceil = \begin{bmatrix} v_1 & v_1 \\ \vdots & \vdots \\ v_n & v_n \end{bmatrix}_{n\times 2}. \tag{4}$$

$\phi(\boldsymbol{M})$ is an operator that normalize each row vector $\boldsymbol{m}_i = (m_i^1, m_i^2)$ of $\boldsymbol{M} \in \mathbb{R}^{n\times 2}$ as follows:

$$\phi(\boldsymbol{M}) = \begin{bmatrix} \frac{\boldsymbol{m}_1}{\|\boldsymbol{m}_1\|} \\ \vdots \\ \frac{\boldsymbol{m}_n}{\|\boldsymbol{m}_n\|} \end{bmatrix} = \begin{bmatrix} \frac{m_1^1}{\|\boldsymbol{m}_1\|} & \frac{m_1^2}{\|\boldsymbol{m}_1\|} \\ \vdots & \vdots \\ \frac{m_n^1}{\|\boldsymbol{m}_n\|} & \frac{m_n^2}{\|\boldsymbol{m}_n\|} \end{bmatrix}_{n\times 2}. \tag{5}$$

3.3 Whale optimization algorithm

The Whale Optimization Algorithm (WOA) is a stochastic swarm-based optimization algorithm that mimics the hunting behavior of humpback whales [22]. The algorithm starts by randomly initializing the whale population, and then updates the population in each iteration using three foraging operations: (1) encircling prey operation moves the whale population towards the current best solution; (2) bubble net operation creates a region around the best solution where the whales move chaotically; (3) searching for prey operation moves each whale towards a randomly selected position. The algorithm terminates when reaches a maximum number of iterations or finds a satisfactory solution.

When optimizing ODT, the whale population $\mathcal{P}$ consists of a series of triangular meshes, each with an equal number of vertices. These triangular meshes are referred to as agents in the whale population. The number of vertices in each agent is denoted by n, the coordinate dimension of the vertices is denoted by d, and the maximum global iteration number is denoted by K. Let $\mathbf{y}^k \in \mathbb{R}^{nd}$ denote the vector formed by all vertices of an agent in the k^{th} population. Moreover, let $\mathbf{y}_r^k$ and $\mathbf{y}_b^k$ denote an agent randomly selected from and the best of thek^{th} population, respectively. Define $\mathbf{a}^k = a\mathbf{r}^k$ as a vector related to the iteration number k and a random vector $\mathbf{r}^k \in [-1, 1]^{nd}$, where $a = 2(1 - \frac{k}{K})$. Random numbers p and l chosen from the intervals $[0, 1]$ and $[-1, 1]$, respectively, and random vectors $\mathbf{c} \in [0, 2]^{nd}$. Then, the ith element of vertex vector $\mathbf{y}^{k+1}$ of an agent in the $(k + 1)^{\text{th}}$ population is generated from the agents in k^{th} population as follows:

$$(\mathbf{y}^{k+1})_i = \begin{cases} (\mathbf{y}_b^k)_i - (\mathbf{a}^k)_i \cdot |(\mathbf{c})_i \cdot (\mathbf{y}_b^k)_i - (\mathbf{y}^k)_i| & p < 0.5, |(\mathbf{a}^k)_i| < 1 \\ (\mathbf{y}_b^k)_i + e^{sl}\cos(2\pi l)|(\mathbf{y}_b^k)_i - (\mathbf{y}^k)_i| & p \geq 0.5 \\ (\mathbf{y}_r^k)_i - (\mathbf{a}^k)_i \cdot |(\mathbf{c})_i \cdot (\mathbf{y}_r^k)_i - (\mathbf{y}^k)_i| & p < 0.5, |(\mathbf{a}^k)_i| \geq 1. \end{cases} \tag{6}$$

The three equations in Eq. (6) sequentially describe the three foraging operators mentioned earlier in the WOA, namely, encircling prey, bubble net, and searching for prey. The variables p and $\mathbf{a}^k$ are used to control the choice of foraging operations in the WOA. In the original WOA [22], the adjustable parameter s is set to 1. However, although the original WOA algorithm is flexible and able to avoid local optima, it may not be suitable for directly optimizing ODT meshes due to its weak local optimization and inability to handle large amounts of data. In Sect. 5.2, we will introduce modifications to the WOA algorithm to better adapt it to the specific requirements of ODT mesh optimization. By leveraging the unique properties of ODT, we aim to improve the performance of the algorithm.

4 Algorithm overview

The this section, we will revisit the problem of optimizing ODT energy and provide a brief overview of our algorithm's pipeline.

4.1 Problem statement

Given a domain Ω, a density function $\rho(\boldsymbol{x})$ defined over Ω, and a target number N of vertices, our objective is to generate a high-quality mesh $\mathcal{M} = (\boldsymbol{X}, \mathcal{T})$ with exactly N vertices inside Ω that minimizes the ODT energy defined in Eq. (1). As ODT energy is Delaunay consistent, we can optimize the

vertex positions and mesh topology iteratively. Therefore, we can formulate the problem as follows:

$$\min_{\boldsymbol{X}} E^{\rho}_{\mathrm{ODT}}(\boldsymbol{X}, \mathrm{DT}(\boldsymbol{X})). \tag{7}$$

The piecewise C^{∞} continuity and non-convex nature of the problem make it challenging to obtain the global optimal solution using conventional optimization algorithms such as the quasi-Newton method, which may get stuck in local optima. Designing a global optimization algorithm that can efficiently address this problem, especially when the initial mesh quality is poor, is of great importance. To this end, we propose a hybrid optimization algorithm to optimize the ODT energy function.

4.2 Overview of ODT optimization algorithm

Our algorithm takes as input the domain Ω, the desired number N of sites within the domain Ω, and the density function $\rho(\boldsymbol{x})$, where $\boldsymbol{x} \in \Omega$. The output is a high-quality mesh $\mathcal{M} = (\boldsymbol{X}, \mathcal{T})$ with a vertex distribution that follows the density function. An overview of our algorithm pipeline is presented in Fig. 3.

The main idea is to initialize the population using randomly generated mesh, then perform a global optimization

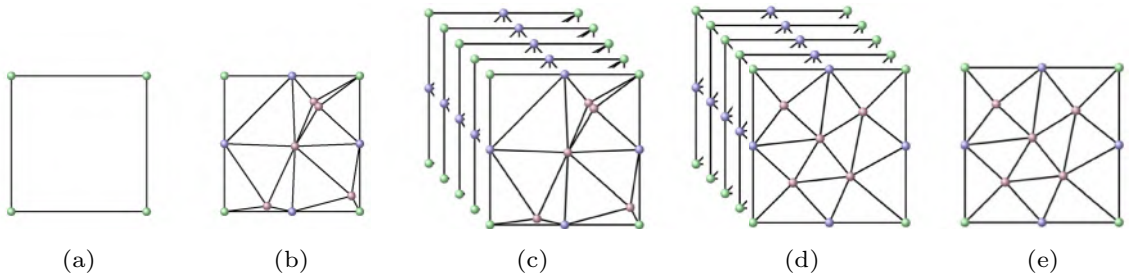


Fig. 3 Algorithm pipeline. **a** Input domain boundary corner sites; **b** initial mesh generated by placing boundary sites according to the given density function and random sites in the interior; **c** initial population generated by perturbing the initial random mesh; **d** updated population by performing the hybrid optimization and **e** the best agent extract from the population as the output high-quality mesh

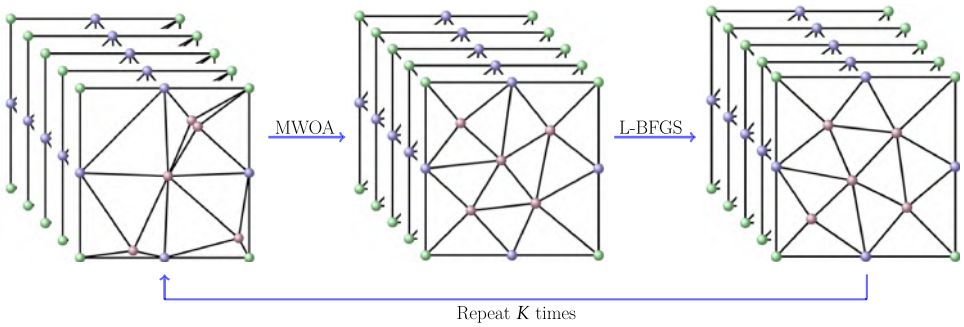


Fig. 4 Hybrid optimization. MWOA is used to update the population globally, while L-BFGS is used to update the population locally

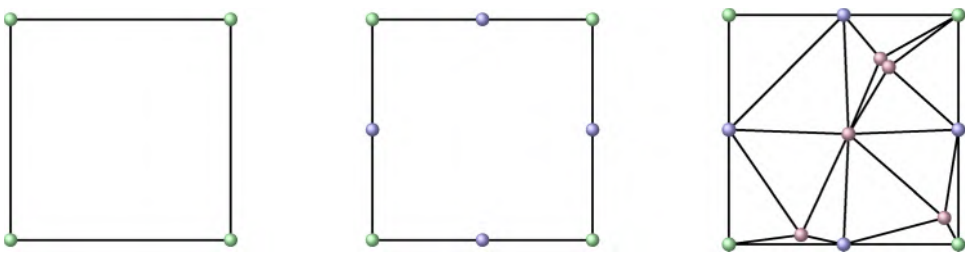


Fig. 5 Mesh initialization. **a** Input boundary corner sites; **b** sample the boundary sites according to the density function; and **c** generate interior sites and build the Delaunay triangulation

using our hybrid algorithm, and finally extract the mesh with the best energy from the population. To explore the global minimum efficiently, we modify the WOA algorithm, referred to as MWOA. During the hybrid optimization process, we employ a combination of MWOA and L-BFGS algorithms to achieve both global and local optimization. Specifically, we apply the MWOA strategy to optimize globally and the L-BFGS approach to obtain a local solution for each agent in the population for K iterations, as shown in Fig. 4. When the position of vertices is changed during optimization, the Delaunay triangulation needs to be recomputed due to the formulation in Eq. (7). If the updated position of a vertex falls outside the designated region, the vertex will not be updated. It is evident that the hybrid optimization approach increases the computational load due to the presence of a population. To mitigate this, we employ two speedup strategies, which are elaborated upon in Sect. 5.4. Algorithm 1 outlines the framework of our hybrid algorithm.

Algorithm 1 Framework of our hybrid optimization algorithm.

Input: The max iteration K; the density function $\rho(\boldsymbol{x})$; the initial mesh $\mathcal{R}$; the population of initial mesh $\mathcal{P}^0 = \cup\{\mathcal{M}_i^0\}_{i=1}^{n_a}$ with n_a agents.

Output: The optimized mesh $\mathcal{M}^*$.

1: Compute the uniformity U of initial mesh $\mathcal{R}$.
2: **for** k=1:K **do**
3: // Line $4-17$ is the MWOA optimization.
4: Initialize parameter $p, s, l, \tilde{a}, \mathbf{c}, \omega$ of each agent.
5: Find best agent $\mathcal{M}_b^k$ with minimal E_{ODT}^{ρ}.
6: **for** $\mathcal{M}_i^k$ in initial population $\mathcal{P}^{k-1}$ **do**
7: **if** $p < 0.5$ **then**
8: **if** $|\tilde{a}| \leq 1$ **then**
9: Do encircling prey (Eq. (8)) and store the new agent.
10: **else**
11: Do searching for prey (Eq. (9)) and store the new agent.
12: **end if**
13: **else**
14: Do bubble-net attacking (Eq. (10)) and store the new agent.
15: **end if**
16: **end for**
17: Construct the k-th population $\mathcal{P}^k$.
18: // Line $20-23$ is the L-BFGS optimization.
19: **if** $(U < 1)$ || $((U > 1)\&(k < m)\&(k \mod z == 0))$ **then**
20: **for** each agent in population $\mathcal{P}^k$ **do**
21: Compute E_{ODT}^{ρ} and ∇E_{ODT}^{ρ} of $\mathcal{M}_i^k$.
22: Do 20 iterations L-BFGS process with E_{ODT}^{ρ} and ∇E_{ODT}^{ρ}.
23: **end for**
24: **end if**
25: **if** all agents have equal E_{ODT}^{ρ} **then**
26: Break the iteration.
27: **end if**
28: **end for**
29: Compute the best agent $\mathcal{M}^*$ with minimal E_{ODT}^{ρ} of $\mathcal{P}^K$.
30: Further optimize $\mathcal{M}^*$ with 30 iterations of L-BFGS optimization.
31: **return** $\mathcal{M}^*$.

5 Algorithm

5.1 Initialization

To optimize the ODT energy using a population-based optimization algorithm, we require an initial population generated by an initial mesh in our framework. Therefore, the initialization process is divided into two parts: mesh initialization and population initialization.

Mesh initialization We begin the initialization by placing sites on the boundary of the domain, as shown in Fig. 5. The placement of these boundary sites is determined by the input density function $\rho(\boldsymbol{x})$. Specifically, the method we use to place the sites depends on whether the density function is uniform or not as follows:

- *Uniform density* The ODT energy tends to distribute points uniformly and results in equilateral triangles in the mesh. This usually leads to a point-to-triangle ratio of approximately 1:2 in an ODT. The length of the triangles (denoted by α) can be estimated by considering the total area of the domain Ω, as well as the number of faces. Therefore, we sample the boundary evenly, ensuring that the distance between two adjacent sampled points is approximately equal to the length of the triangle edges.
- *Non-uniform density* As opposed to the density function, the sizing field refers to a function that determines the ideal edge length within the domain [26]. The sizing field $h(\boldsymbol{x})$ over the domain Ω for sampling the boundary sites can be chosen as $h(\boldsymbol{x}) = \alpha * \rho(\boldsymbol{x})^{-\frac{1}{d+2}}$, where $d = 2, 3$ is the dimension of the problem [33]. Starting from any point $\boldsymbol{x}$ on the boundary, the next two sampling points on the boundary are chosen to be the points with a distance to $\boldsymbol{x}$ of $h(\boldsymbol{x})$. The sampling is propagated from any point on the boundary to cover the entire boundary. To achieve a more accurate distribution of boundary point locations according to the density function $\rho(\boldsymbol{x})$, we apply the averaged Voronoi vertex (AVV) method [21] to the boundary sites using 30 iterations.

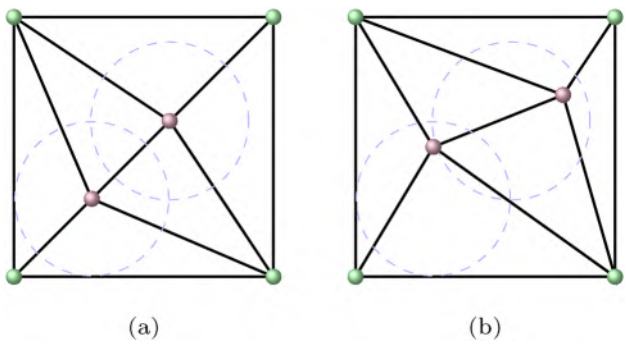


Fig. 6 Perturbation process for population initialization. **a** The initial mesh with two interior points (red dots), where the disturbance range of each inner point is represented by a circle (purple) with a radius equal to half the average length of its adjacent edges. **b** An agent in the population is generated by perturbing the inner vertices within the corresponding disturbance range (purple circle)

To create the initial mesh $\mathcal{R}$, we randomly distribute N vertices within the boundary and use all the sites to construct the Delaunay triangulation. However, since the vertices are randomly placed, the resulting mesh quality of $\mathcal{R}$ at this stage is unpredictable.

Population initialization To generate the initial mesh population, we perturb the initial mesh $\mathcal{R}$ given n_a agents. Each vertex in $\mathcal{R}$ is randomly perturbed in a unit direction with a maximum magnitude of 0.5 times the average length of its adjacent edges, as illustrated in Fig. 6. Our perturbation approach is similar to that described in [21] and [52]. This results in a population of meshes with corresponding vertices which have the same index. Here, 'corresponding' denotes that the particular vertex in the mesh agent is obtained through perturbation from a specific vertex in the initial mesh. Note that agents in the population with identical input vertices will produce the same triangulation and ODT energy. For simplicity, we consider only the agents with ordered sites in the population, where the input vertex set is a sequentially ordered collection based on the indices of the vertices. The initial mesh's vertex index is assigned according to the sequence of inserted vertices. In subsequent global optimization algorithms, the algebraic operations among agents degrade to the corresponding algebraic operations among the associated vertices. Our perturbation approach reduces the search space of vertices from the whole domain to the neighborhood area of corresponding vertices.

5.2 Global search

ODT energy function is non-convex and non-smooth, which means it can easily get trapped in local minima when only local search is performed. Therefore, a global optimization approach is needed to jump out of the local minimum and explore the search space efficiently. We believe that WOA is a suitable choice for our problem due to its simple operators, fewer parameters, and strong ability to jump out of local optima. Hence, we modify the WOA algorithm, referred to as MWOA, to better suit our particular optimization problem, and the resulting framework is presented in Algorithm 1.

5.2.1 Update formula

To adapt the ODT optimization problem, a weight ω and the average length of incident edges of vertices in mesh are

introduced in the MWOA. To enhance further discussions, we reintroduce a matrix representation for the coordinates of all vertices belonging to an agent, as opposed to using a vector.

In the kth iteration, considering the population of meshes denoted as $\mathcal{P}^k$, each agent is represented by a matrix $\mathbf{X}^k \in \mathbb{R}^{n\times d}$ that stores the vertex positions. The matrix $\mathbf{X}^k$ consists of n rows, where each row corresponds to the position of a vertex, and d represents the dimensionality of the vertex positions. Furthermore, $\boldsymbol{l}^k \in \mathbb{R}^n$ is a vector associated with an agent in the kth iteration. Each element of $\boldsymbol{l}^k$ represents the average length of the incident edges connected to the corresponding vertices of the agent. Additionally, in the kth iteration, $\mathbf{X}_r^k$ and $\mathbf{X}_b^k$, or $\boldsymbol{l}_r^k$ and $\boldsymbol{l}_b^k$, refer to the position matrix or average length vector, respectively, of a randomly selected agent and the best agent from the population $\mathcal{P}^k$.

The matrix $\mathbf{X}^{k+1}$ for each agent in the $(k+1)$th population is generated from the population $\mathcal{P}^k$ using one of the following three operators: encircling prey, searching for prey, and bubble-net attacking. The operator selection depends on the values of p and $\tilde{a}$. Here, $\tilde{a}$ is calculated as $\tilde{a} = ar_1$, where r_1 is a random number in the range $[-1, 1]$. The random variables p and l are sampled from the intervals $[0, 1]$ and $[-1, 1]$ respectively. Moreover, random vectors $\mathbf{c}$ are generated from the range $[0, 2]^n$. The parameters s, a and ω are real numbers that are chosen based on specific considerations, which will be discussed in subsequent section. Algorithm 1 provides detailed instructions on how these operators are applied in the evolutionary process.

- *Encircling prey*

$$\mathbf{X}^k + 1 = \mathbf{X}_b^k - \omega\tilde{a} \cdot \phi(\lfloor \mathbf{c} \rceil \cdot \mathbf{X}_b^k - \mathbf{X}^k) \cdot \lfloor \boldsymbol{l}_b^k \rceil. \tag{8}$$

- *Searching for prey*

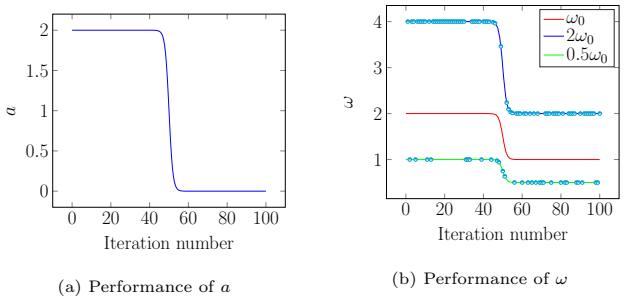
$$\mathbf{X}^k + 1 = \mathbf{X}_r^k - \omega\tilde{a} \cdot \phi(\lfloor \mathbf{c} \rceil \cdot \mathbf{X}_r^k - \mathbf{X}^k) \cdot \lfloor \boldsymbol{l}_r^k \rceil. \tag{9}$$

- *Bubble-net attacking*

$$\mathbf{X}^k + 1 = \mathbf{X}_b^k + \omega e^{sl} \cos(2\pi l) \cdot \phi(\mathbf{X}_b^k - \mathbf{X}^k) \cdot \lfloor \boldsymbol{l}_b^k \rceil. \tag{10}$$

5.2.2 Parameter settings

Our proposed MWOA (Modified Whale Optimization Algorithm) method incorporates three parameters: s, a, and ω. The parameter s and a are inherited from the original WOA, while ω is a newly introduced parameter. The value of s determines the range of steps used during the bubble-net attacking process. In our extensive testing with different initial meshes and varying vertex numbers, we discovered that adjusting the value of s within the range of [3.2, 6.2] had minimal impact on the resulting mesh energy. Consequently, for this specific case, we chose to set s to 3.6. However, we found that the parameters a and ω exerted a more substantial influence on the performance of the algorithm. Therefore, our focus has been on fine-tuning a and ω to achieve optimal results in our particular problem.



(a) Performance of a (b) Performance of ω

Fig. 7 Performance of the parameters in MWOA with $K = 100$. **a** The performance of parameter a with respect to the iteration. **b** The cyan circle which means the ω value of an agent. We can figure out the ω distribution of the agent as the number of iterations increases. The probability of choosing $2\omega_0$ is 80% before the 50th iteration, after the probability is decreasing to 50%

Parameter a controls the changes between the encircling prey and searching for prey operators. On the other hand, parameter ω determines the proximity relationship between $(k+1)^{\text{th}}$ population and either the best agent or a randomly selected agent from the k^{th} population. These parameters determine the extent to which the algorithm balances between random exploration of the search space and thorough investigation of promising areas. Given the unknown distribution of vertices in the initial mesh, it is natural to prioritize random exploration of the search space to identify potentially promising solution regions. Subsequently, further investigation can be conducted in these identified areas. Motivated by the effectiveness of the sigmoid function in binary classification tasks, a modified sigmoid function called the specular sigmoid function can be employed. The specular sigmoid function decreases within the interval $[\xi, \eta]$ and is denoted as $f(x) = \eta - (\eta - \xi)\frac{1}{1+e^{-x}}$. This function is utilized as a guide in the parameter selection process, enhancing the exploration and exploitation capabilities of MWOA.

- *a setting* In the original WOA, the parameter a decreases linearly from 2 to 0 as the iterations progress. However, this linear decrease can lead to the algorithm getting trapped in local optima, especially in our problem where the ability to explore declines rapidly. To overcome this limitation, we propose a modification to the WOA algorithm by setting

$$a = 2 - 2\frac{1}{1 + e^{-(k-\frac{K}{2})}}, \tag{11}$$

where k is the current iteration and K is the maximal number of iterations. In Fig. 7a, the parameter a follows a pattern similar to the specular sigmoid function, gradually decreasing from 2 to 0 with a sharp drop around the midpoint ($K/2$). The initial high values of a indicate a preference for the exploration-based "searching for prey" operator, enhancing the algorithm's ability to explore the search space. This improved exploration contributes to better performance in finding the global optimal solution. However, this setting also increases the risk of premature convergence, where the algorithm converges to a suboptimal solution prematurely. To address this issue, we introduce another parameter, ω, to the algorithm.

- *ω setting* To control population diversity, we introduce the parameter ω and utilize the specular sigmoid function in its selection. To overcome premature convergence, we divide the population into two parts: one with a larger ω for exploring the search space, and the other with a smaller ω for investigating promising areas. Specifically, we set:

$$\omega = \begin{cases} 2\omega_0 & t \geq t_0 \\ 0.5\omega_0 & t < t_0 \end{cases}, \tag{12}$$

where

$$\omega_0 = 2 - \frac{1}{1 + e^{-(k-\frac{K}{2})}}, \quad t_0 = 0.2 + 0.3\frac{k}{K}, \tag{13}$$

and t is a random number in [0, 1]; see Fig. 7b.

5.3 Local search

To address the challenge of quickly obtaining a local minimum solution in the stochastic nature of MWOA, we incorporate the L-BFGS method, known for its energy decline and fast convergence as demonstrated in [21]. This integration takes advantage of the piecewise C^∞ continuity property of the ODT energy function when the triangulation's topology remains fixed. During our global optimization process, the L-BFGS method is applied to all agents in the population. We set a maximum of 20 iterations per agent, striking a balance between exploration and exploitation. To further refine the solution, we perform an additional 30 iterations of the L-BFGS method before selecting the mesh with the best energy. This additional step allows for further minimization and improvement of the selected mesh.

5.4 Speedup strategies

The algorithm can become time-consuming, especially during the local search phase, due to the requirement of recalculating the Delaunay triangulation whenever the positions of the mesh vertices are modified. To mitigate this issue and improve efficiency, we employ different optimization schemes tailored to meshes with varying initial qualities. Additionally, we parallelize the hybrid algorithm to further enhance its performance.

To mitigate computation time, we utilize the uniformity of simplex quality as an indicator of the mesh's quality and exclude local search during the early iterations. To measure the uniformity of a mesh $T = \{\tau\}$, we employ a metric proposed by Persson and Strang [53], which is defined as follows:

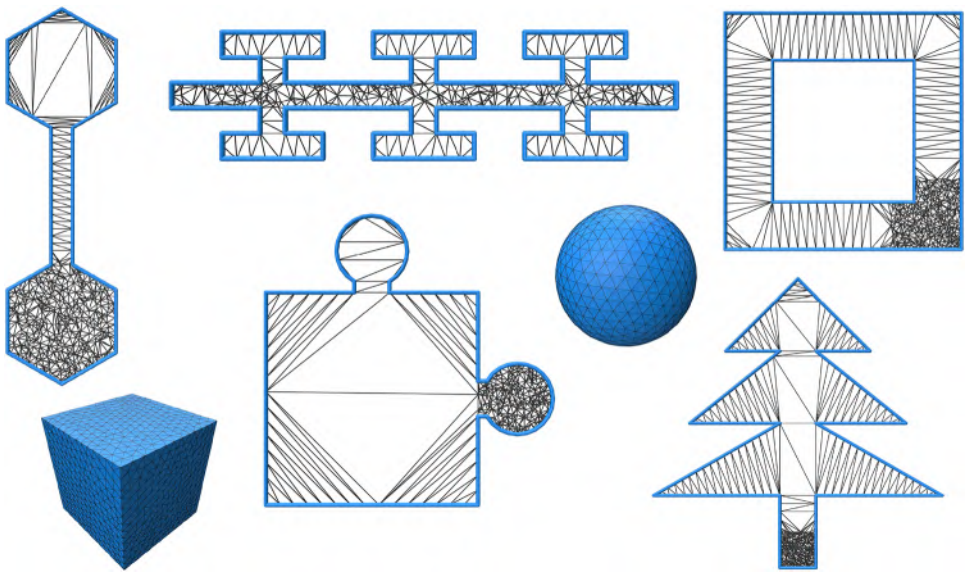


Fig. 8 The examples tested in this paper

$$U(T) = \sqrt{\frac{1}{M}\sum\nolimits_{\tau \in T}(\frac{Q_\tau}{Q^*} - 1)^2}, \tag{14}$$

where $Q_\tau = \int_\tau \rho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ represents the weighted area of a simplex τ with a density function $\rho(\boldsymbol{x})$ defined over the simplex τ, M is the number of simplex in the mesh and $Q^* = \frac{1}{M}\sum_{\tau \in T} Q_\tau$ is the average value of Q_τ over all the simplices in the mesh. If the value of the uniformity metric $U(T)$ for the mesh T exceeds a predefined threshold of 1, indicating poor mesh quality, we limit the optimization to MWOA's global search during the earlier iterations. To implement this approach, we introduce two parameters: the divided number z and the specific iteration m. If the current iteration k is not a multiple of z and is less than m, we exclusively employ MWOA without the use of L-BFGS. Through our experiments, we have found that setting $z = 3$ and $m = 30$ strikes a good balance between local and global search.

In addition to utilizing different optimization schemes and excluding local search in certain iterations based on mesh quality, we have parallelized the operations for different agents in each iteration to enhance efficiency. By leveraging the inherent parallelism of our methods, we assign separate threads to each agent, maximizing CPU utilization and reducing runtime. However, it is important to strike a balance between speed and overhead. With consideration of the CPU capacity of our device, we have set the number of threads to 10. This choice minimizes time spent on thread switching and termination while still achieving significant improvements in performance. In the upcoming section, we will present experimental results that demonstrate the time reduction achieved through this parallelization approach.

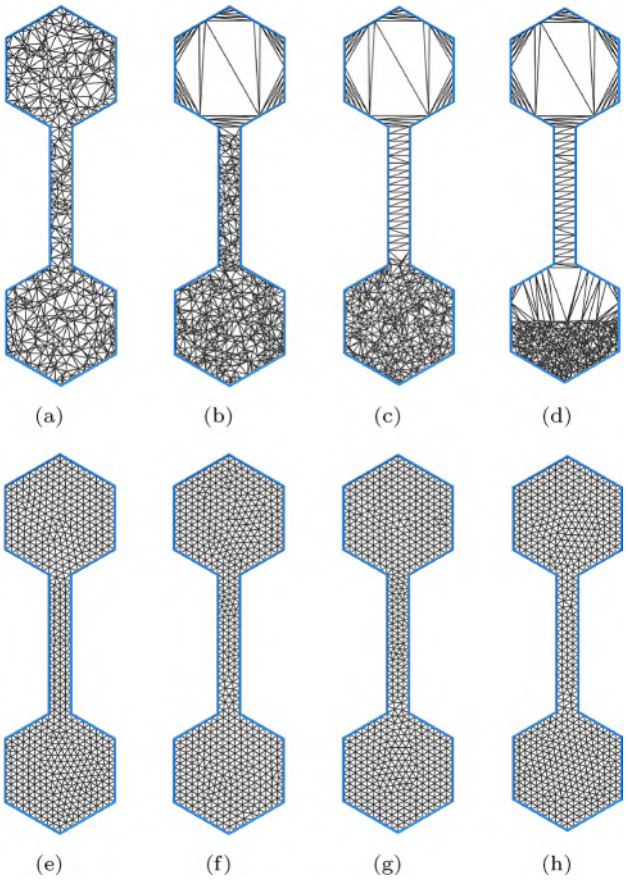


Fig. 9 Mesh results from different initial mesh. **a**–**d** illustrate different initial meshes, while **e**–**h** showcase the results obtained by the Global-MWOA algorithm corresponding to the initial meshes (**a**–**d**)

6 Experimental results

In this section, we present the results of our experiments on the proposed meshing method, referred as Global-MWOA, and compare them with those obtained using classic methods such as Tetgen [54] and Gmsh [55], as well as state-of-the-art methods such as Global-SA [21] and fTetWild [56]. We evaluate our approach on various 2D and 3D examples in Fig. 8 and dataset from paper [57], and provide a detailed analysis of the results using evaluation metrics such as aspect ratio, radius ratio, angle, the number of slivers, ODT energy, etc. In the following, uniformity is defined in Eq. (14), the angle represents the minimum angle of each triangle and dihedral angle denotes the dihedral angle within each tetrahedron. A tetrahedron is considered as a sliver if it has a dihedral angle less than 5° or 10°. The aspect ratio and radius ratio of each simplex are based on [58, 59]. Weighted area of a simplex τ can be computed as $Q_\tau = \int_\tau \rho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ with a density function $\rho(\boldsymbol{x})$ defined over the simplex τ. The color bar of each figure represents the average edge length of each element, with darker colors indicating larger lengths.

Our meshing method is implemented in C++, utilizing CGAL [60] for computing Delaunay triangulation in 2D and 3D domains. For local search, we use the HLBFGS [61] library to calculate the updated mesh population. During global optimization, we perform fast matrix operations with Eigen [62] and visualize the result using the Polyscope libraries [63]. We take advantage of OpenMP for parallel computing and measure the time on a PC with an 11th Gen Intel(R) Core(TM) i5-11400F 2.60GHz CPU and 16GB of RAM. In our experiments, we set the number of agents to $n_a = 100$ in 2D and $n_a = 50$ in 3D, the maximum iterations to $K = 100$, the number of partitions to $z = 3$, the specific iteration to $m = 30$, and the number of threads to $n_t = 10$, except for the acceleration ratio test.

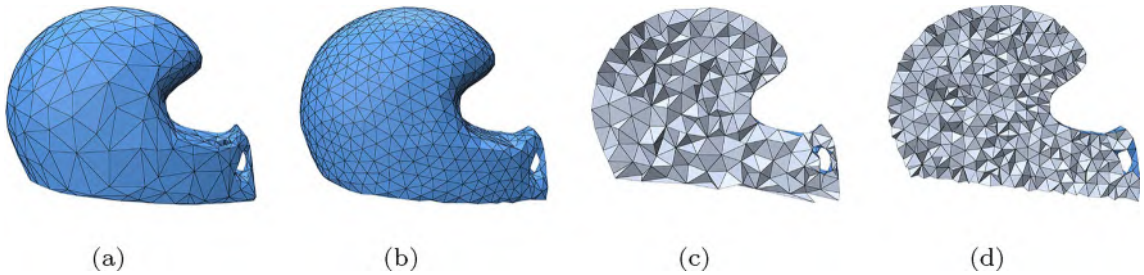


Fig. 10 Mesh results from different boundary sampling. **a**, **b** depict surface meshes with two distinct boundary sampling, while **c**, **d** showcase sectional views of the volumetric mesh results obtained using these two meshes as inputs for the Global-MWOA method

Table 1 Comparison of mesh quality with different boundary mesh

Boundary sampling	#Vert	Uniformity	Radius ratio		Dihedral angle				Slivers	
			Min	Avg	Min	Avg min	Max	Avg max	<5 °	<10 °
(a)	1142	0.410321	1.98892e−09	0.797824	0.0022646	46.2683	179.995	99.7562	118	172
(b)	2799	**0.198753**	**0.0913938**	**0.894991**	**4.24598**	**53.2274**	**172.429**	**91.9135**	**1**	**8**

(a), (b) refer to the Fig. 10a, b

Results in bold indicate superior performance

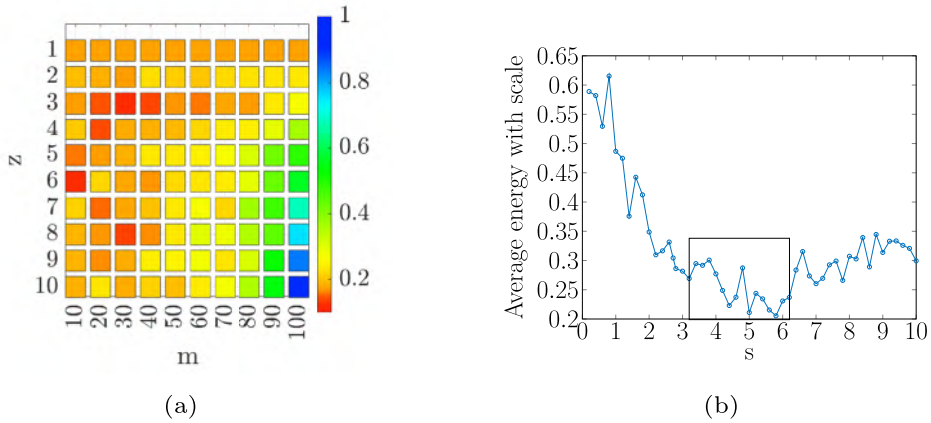


Fig. 11 The impact of parameter variations on the global optimization of the ODT energy function. **a** Parameters z and m. **b** Parameter s. The color representation in (**a**) and y-axis coordinates in (**b**) represent the average normalized energy values across multiple models

6.1 Evaluation of the algorithm

Different initial meshes Due to its exceptional global optimization capabilities, Global-MWOA consistently produces favorable results for various initializations, with minimal sensitivity to the choice of initialization, as illustrated in Fig. 9. Even for significantly diverse initializations, our Global-MWOA method always yields commendable outcomes.

Different boundary sampling To generate high-quality volumetric meshes, Global-MWOA requires input surface meshes that are of good quality post-remeshing and consistently maintains the quality of the input surface mesh during optimization. Consequently, the quality of the input mesh, which is the result of boundary sampling, does have an influence on the quality of the generated volumetric mesh. Figure 10 illustrates surface meshes of varying quality obtained through two different boundary sampling approaches for the same model, along with corresponding volumetric mesh results generated using Global-MWOA. Table 1 presents quality metrics for the volumetric meshes corresponding to the images (c–d) in Fig. 10. The results from Fig. 10 and Table 1 indicate that Global-MWOA is sensitive to the quality of the input surface boundary sampling. The bolded results in the following table indicate superior performance.

Different algorithm parameters Parameters z and m are designed to control and optimize the initial stages of the L-BFGS process by assessing the uniformity of the initial

Table 2 Comparison of Global-SA and our algorithm performance in various domains

Domain (#Vert)	Method	ODT energy	Uniformity	Angle		Aspect ratio		Radius ratio		Time/s
				Avg	Min	Avg	Min	Avg	Min	
Dumbbell (500)	Global-SA	250.088	0.606432	53.4332	32.3589	0.923314	0.577173	0.972925	0.639748	49.958
	Ours	**181.443**	**0.0888114**	**55.8262**	**36.2725**	**0.953819**	**0.750708**	**0.988243**	**0.849305**	49.855
Square hole (500)	Global-SA	0.000210545	0.149884	53.7983	38.2578	0.929554	0.668689	0.98101	0.769393	58.96
	Ours	**0.000208115**	**0.112504**	**53.9536**	**38.7073**	**0.933018**	**0.759673**	**0.981499**	**0.866175**	58.848
Tree (500)	Global-SA	0.252239	0.197124	53.0149	24.5672	0.919317	0.522277	0.974237	0.561312	46.287
	Ours	**0.245853**	**0.12711**	**53.3616**	**24.5672**	**0.926572**	**0.522277**	**0.977807**	**0.561312**	46.22
Curve (200)	Global-SA	2.76596	0.275916	**48.5436**	30.0085	**0.86049**	0.579431	0.938976	0.65327	32.813
	Ours	**2.67666**	**0.198486**	48.0821	**33.6562**	0.859834	**0.654102**	**0.940317**	**0.744273**	32.766
Jigsaw (uniform) (250)	Global-SA	0.116355	0.163398	52.1271	25.8004	0.912936	0.548961	0.967865	0.624712	23.009
	Ours	**0.114592**	**0.124207**	**53.7039**	**30.3653**	**0.916763**	**0.595765**	**0.969254**	**0.678837**	22.974
Jigsaw (density) (250)	Global-SA	0.226459	0.344341	**51.6637**	32.6543	**0.901345**	0.662847	**0.965121**	0.761189	34.378
	Ours	**0.214488**	**0.325862**	51.4356	**34.4121**	0.897101	**0.680288**	0.963733	**0.785373**	34.337

The vertex numbers in the boundary are provided below each domain name. Results in bold indicate superior performance

Table 3 Quantitative measurement of consistency for Global-SA and our Global-MWOA method

Domain	Target probability (%)	Global-SA (%)	Global-MWOA (%)
1	1.3	**1.9**	2.5
2	43.2	38.8	**43.6**
3	16.8	23.2	**16.6**
4	18.5	13.5	**14.1**

Bold indicates that the probability distribution of vertices in this region is closer to the target probability

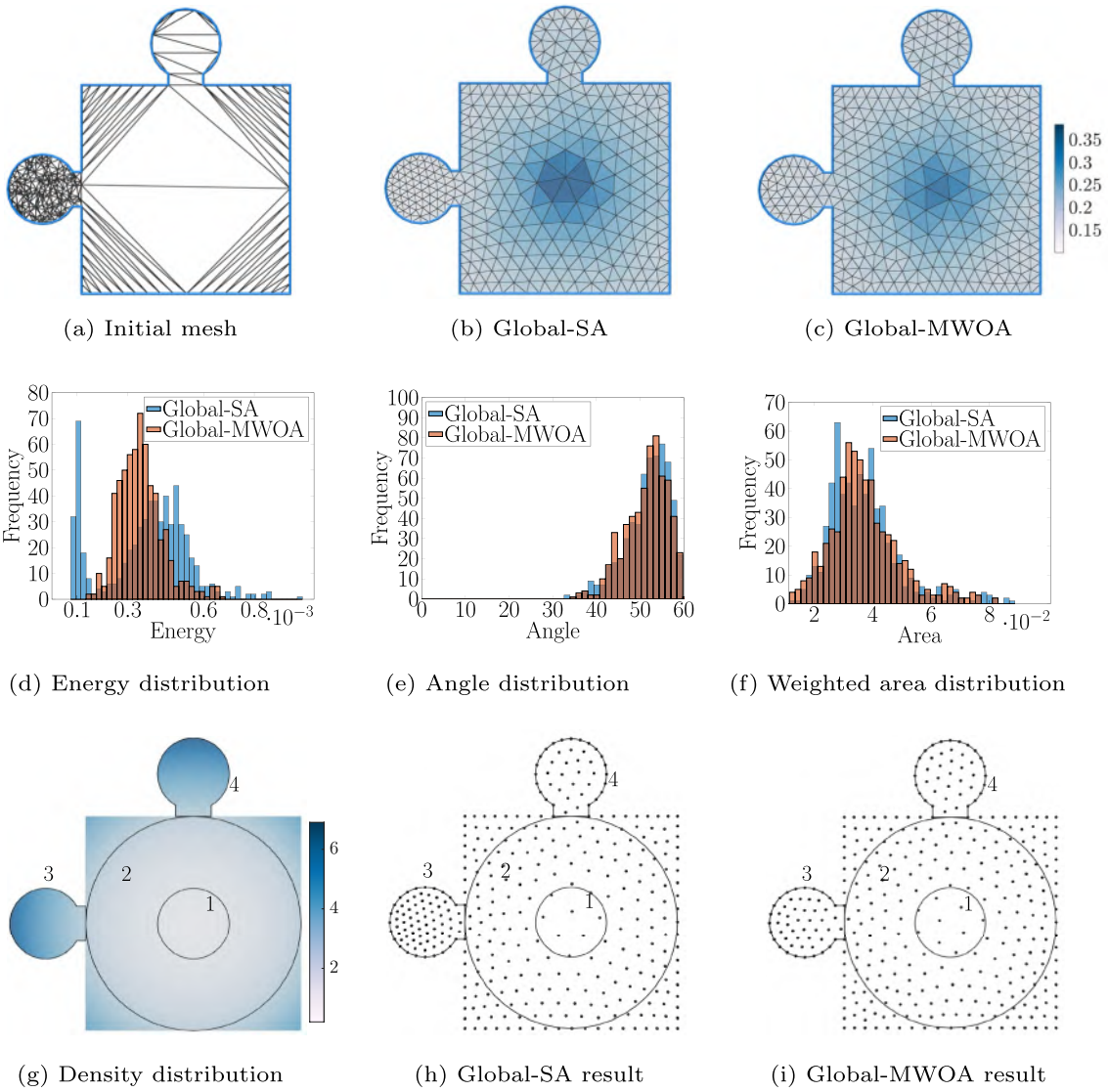
mesh, thereby reducing time consumption. To examine the impact of different values of parameters z and m on the algorithm, we randomly selected 11 models (including five 2D models and six 3D models) and devised extreme initializations. From Fig. 11a, it is observed that larger values of z and m lead to higher energy function values. To balance mesh quality and time consumption in our experiments, we set the parameters as $z = 3$ and $m = 30$.

Parameter s is utilized to control the step size range during the bubble-net attacking process in the Global-MWOA. Similarly, we randomly selected 26 models (comprising thirteen 2D and thirteen 3D models) with varying vertex counts ranging from 200 to 5000. The models exhibit diverse initializations in terms of uniformity, and the objective functions include both with and without constant density function formulations. We conducted experiments to assess the impact of parameter s on our algorithm. From Fig. 11b, we observe a general trend of decreasing average energy followed by an increase with the growth of s. We find that the choice of s in the interval [3.2, 6.2] has a limited impact on the experimental results. In our experiments, we set s to a specific value, namely, 3.6.

6.2 Comparisons on 2D domain

In the 2D domain, we conducted experiments to generate both uniform meshes and meshes with density functions. Our algorithm was tested on various 2D domains to evaluate its performance. To assess the effectiveness of our global optimization algorithm, we used extreme initializations, as depicted in Fig. 8, to create challenging scenarios for global optimizations. In our study, we exclusively compare our algorithm with the established method for generating meshes using ODT energy for global optimization. The competitor algorithm, known as Global-SA and proposed by Chen et al. [21], is used as a reference for our comparisons. We utilized the completion time of our algorithm as the stopping condition for the Global-SA method. In addition, we compared the ODT energy and mesh quality of meshes generated by both approaches over similar running times. Better mesh quality is characterized by lower energy, angles closer to 60°, aspect ratios and radius ratios closer to 1, and lower uniformity values (Eq. 14).

Table 2 demonstrates the advantage of our algorithm over Global-SA. Our algorithm consistently generates meshes with lower ODT energy and higher uniformity compared to Global-SA, starting from the same initial mesh. This suggests that our algorithm better conforms to the desired vertex distribution defined by the density function. Consequently, our optimization algorithm outperforms Global-SA by producing smaller local minima, showcasing its superior global optimization capabilities. Note that, in the curve and jigsaw (density) examples, our algorithm achieves slightly lower values for average angle, average aspect ratio, and



(a) Initial mesh (b) Global-SA (c) Global-MWOA

(d) Energy distribution (e) Angle distribution (f) Weighted area distribution

(g) Density distribution (h) Global-SA result (i) Global-MWOA result

Fig. 12 Comparison between Global-SA and our algorithm with density function $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 + 0.2$ in the boundary jigsaw

average radius ratio compared to Global-SA. However, this small discrepancy does not imply that Global-SA's results are superior. Figure 12 visually demonstrates the comparison between the resulting meshes generated by both methods for the jigsaw (density) example. Considering the density function $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 + 0.2$, it is expected that the left side circle of the jigsaw should have the same density as the circle above it. Thus, having a similar number of vertices in the two small circles aligns better with the density function. Simultaneously, we introduce a quantitative measure of consistency: the probability of vertices being in a specific region [64]. The density function implies the target probability of vertices being in certain regions. If the probability of vertices being in a particular region approaches the target probability, it indicates that the vertex distribution density of the mesh is closer to the density function. As depicted in Fig. 12, where (g) shows color representation of density function values with corresponding region numbers indicated, and (h-i) present the vertex distribution of mesh results obtained by Global-SA and Global-MWOA algorithms. Table 3 illustrates the target probabilities for different regions and the actual probabilities of vertex distribution obtained by the two methods. It is evident that our Global-MWOA results' vertex distribution probabilities are closer to the target probabilities, indicating a better alignment with the density function.

Figure 13 displays the energy curves over time for the six examples in Table 2. Our algorithm converges faster than Global-SA, and the energy remains lower over the same period of time.

We compare the uniformity results of two selected examples, the tree and dumbbell, as shown in Figs. 14 and 15,

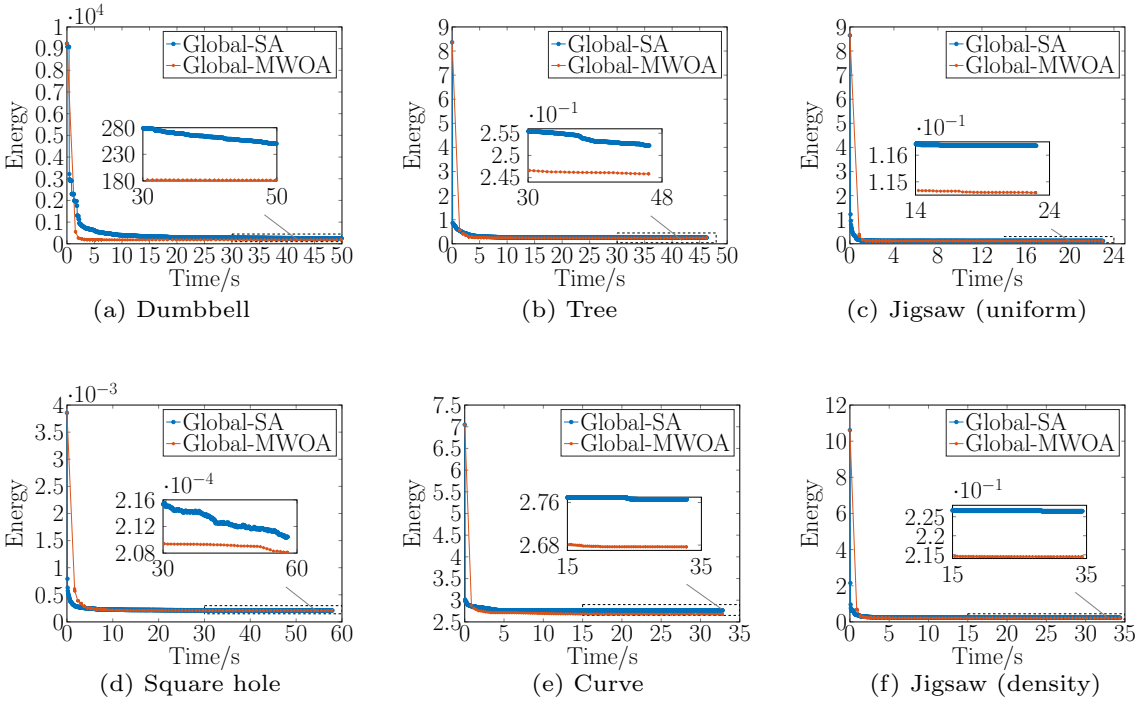


Fig. 13 Energy plot of Global-SA and our algorithm with respect to the time. **a**–**f** Respectively shown for six different examples. Here, 'jigsaw(uniform)' and 'jigsaw(density)' refer to cases where the target mesh is uniform and has density $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 + 0.2$ in the jigsaw region, respectively

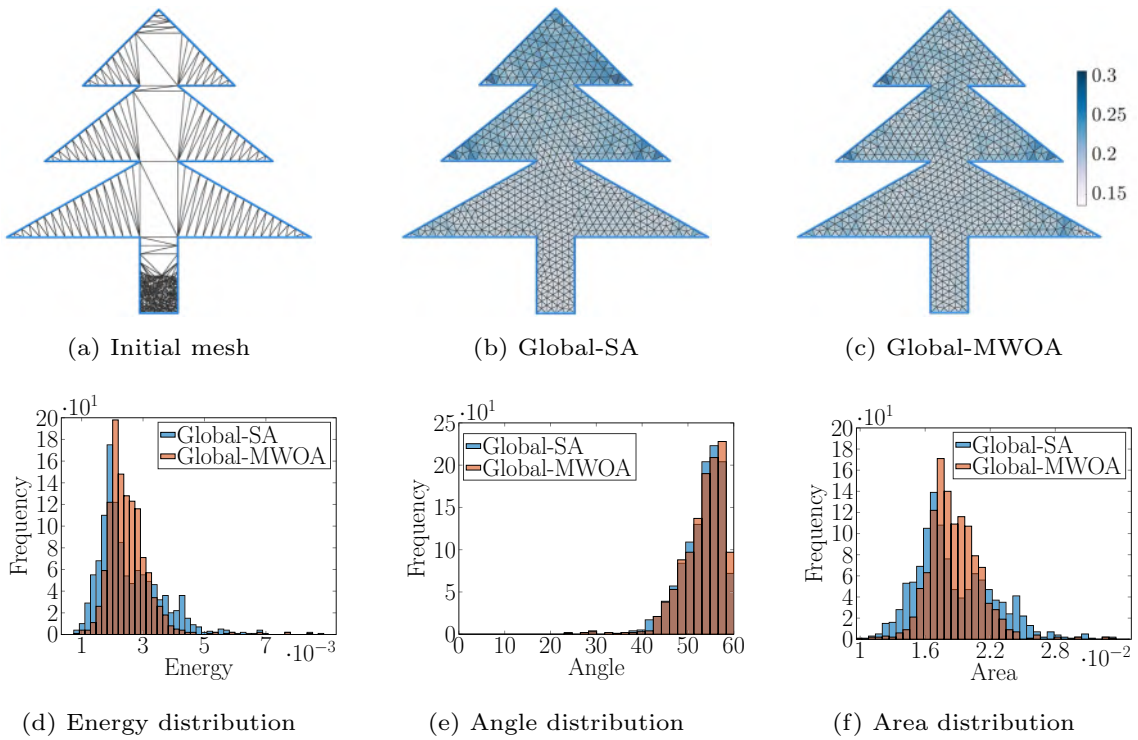


Fig. 14 Comparison between Global-SA and our algorithm to generate uniform results in the boundary tree

respectively. Both Global-SA and our approach are applied to the same initial mesh with the same running time. Compared to Global-SA, our approach yields results closer to the global optimum within a given time, where the given time corresponds to the execution duration of our algorithm, as illustrated in Figs. 14 and 15. In contrast, Global-SA fails to

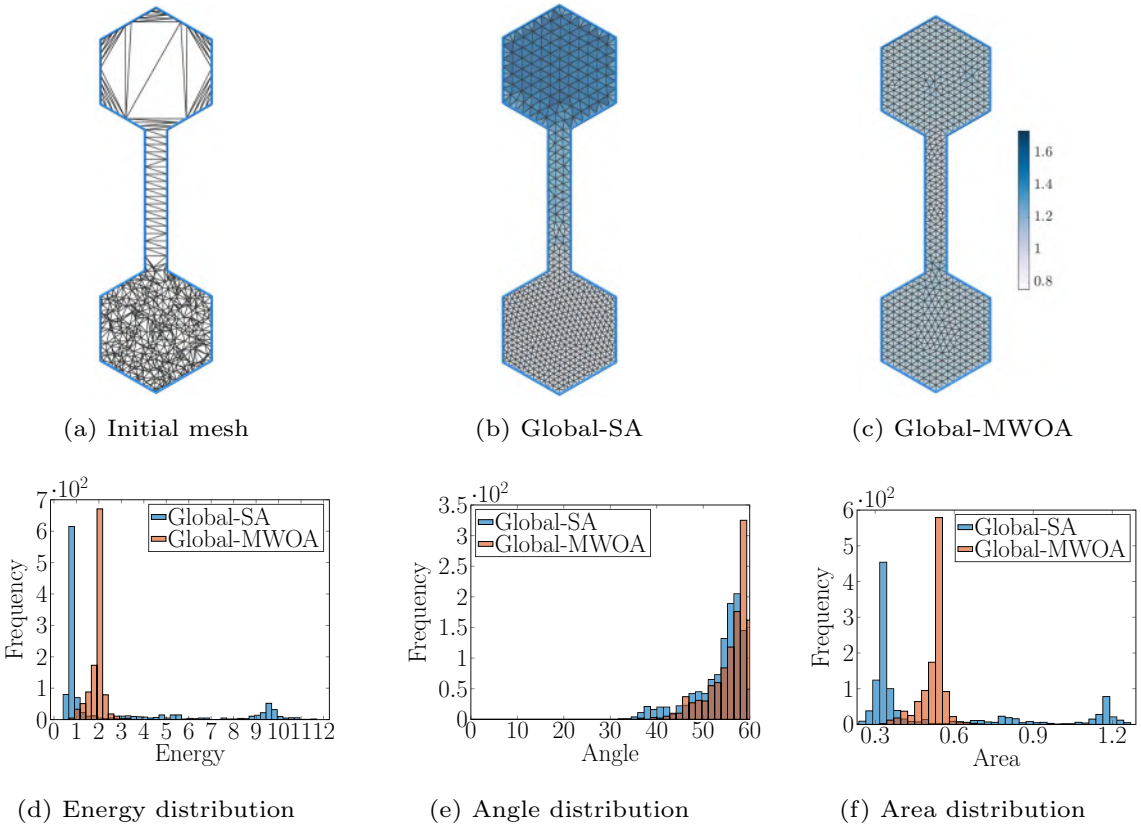


Fig. 15 Comparison between Global-SA and our algorithm to generate uniform results in the boundary dumbbell

Table 4 Comparison of the performance of Global-SA and our algorithm in 3D domains

Domain	#Vert	Method	ODT energy	Uniformity	Radius ratio		Dihedral angle		Slivers		Time/s
					Min	Avg	Min	Max	<5 °	<10 °	
Eraser ball	3710	Global-SA	0.0389095	0.802258	1.12612e−07	0.765257	0.512939	179.977	161	434	941.501
		Ours	**0.0387632**	**0.27613**	**0.0919047**	**0.897301**	**4.8119**	**173.251**	**1**	**4**	940.387
Hilbert cube	3303	Global-SA	5.37238e+09	0.667163	0.0327842	0.894748	1.92566	177.681	2	13	863.339
		Ours	**21195.8**	**0.189062**	**0.090314**	**0.895049**	**5.20237**	**173.29**	**0**	**6**	860.617

#Vert represents the number of vertices in the mesh. Results in bold indicate superior performance

Table 5 Comparison of mesh quality with meshes generated by different tools

Method	#Vert	Radius ratio		Dihedral angle				Slivers		Time/s
		Min	Avg	Min	Avg min	Max	Avg max	<5 °	<10 °	
Gmsh	6353	0.30029	0.801186	13.2039	47.3334	156.676	101.58	**0**	**0**	0.45888381
TetGen	6354	0.146943	0.793966	8.35435	47.3694	164.829	102.338	**0**	13	**0.062**
fTetWild	6442	**0.357043**	0.878641	**21.5749**	54.15	**143.288**	94.6218	**0**	**0**	3.25545
Ours	6354	0.166533	**0.91293**	7.35798	**54.3388**	166.159	**89.8309**	**0**	1	4490.96

Results in bold indicate superior performance

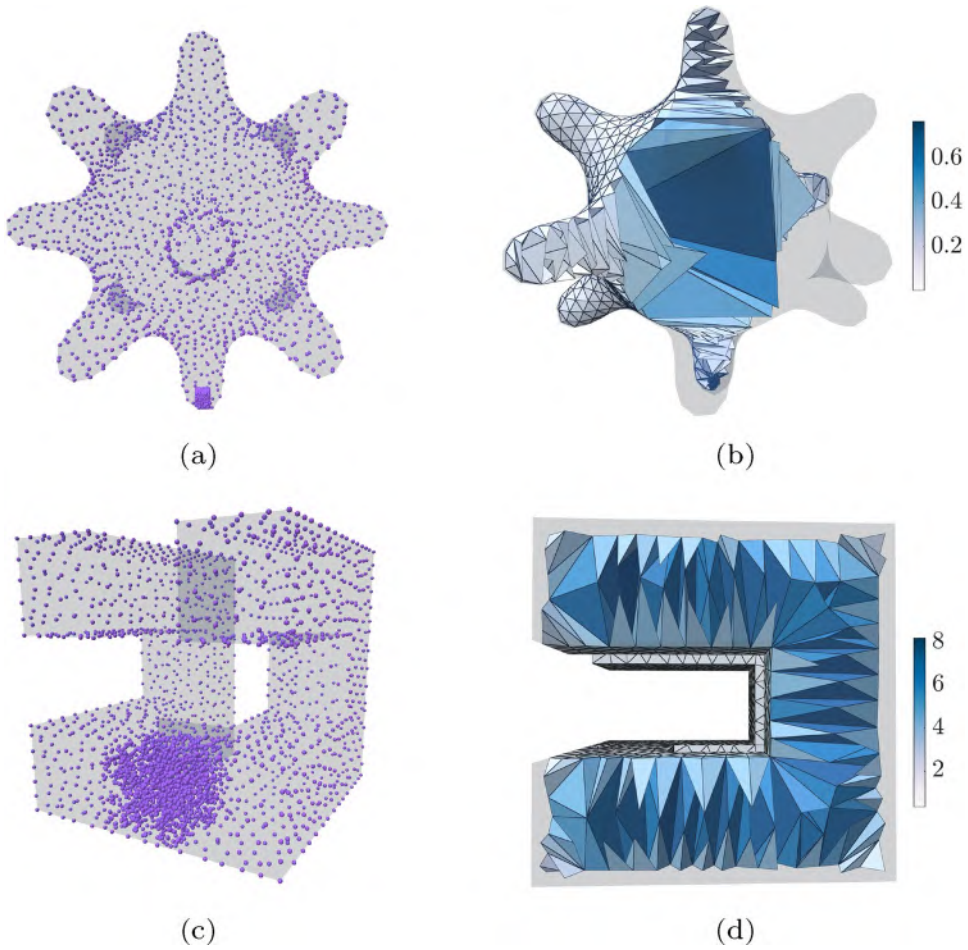


Fig. 16 Initialization of 3D examples. **a**–**d** Respectively represent the initial points and initial mesh of the eraser ball and the Hilbert cube

achieve similar results to Figs. 14c and 15c within 4000 s. Additionally, our algorithm performs better in density scenarios, as illustrated in Fig. 12.

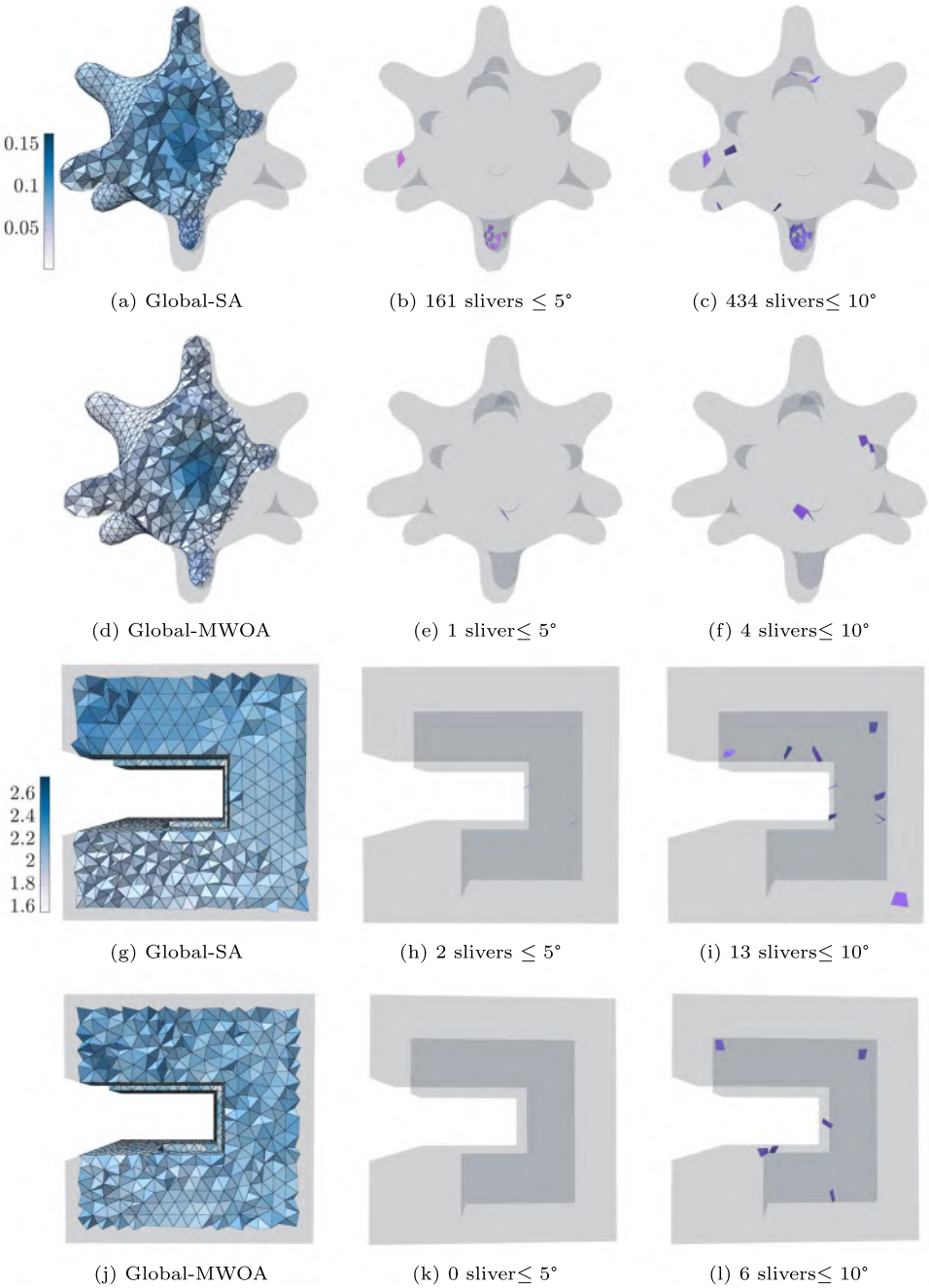
In terms of local search ability, MWOA is inferior to the simulated annealing method used in Global-SA. By evaluating the uniformity of the initial mesh, we can switch to a global–local optimization approach using Global-SA, resulting in faster convergence to the minimum. Within this framework, our proposed method not only generates high-quality meshes quickly but also mitigates the impact of initialization on the optimization of ODT energy generation meshes.

6.3 Comparisons on 3D domain

We also show the performance of our algorithm in 3D. For the sake of simplicity, a remeshing surface mesh with the desired density distribution is adopted as input. It has been proven that minimizing the ODT energy function will attempt to equidistribute weighted volumes and edge lengths of all simplices in the triangulation when the density function is piecewise constant [29]. Based on this theorem, we can draw a hypothesis that optimizing ODT energy globally can reduce slivers when generating graded mesh.

Similar to the 2D example, we employed extreme initialization to evaluate the global optimization abilities of both methods. Figure 16 illustrates the vertex positions and cross-section meshes of our initialization. Our algorithm outperformed Global-SA in terms of global minimum and mesh quality within the same time frame, as indicated in Table 4. With the same initial mesh, Figs. 17 and 18 demonstrate that the resulting mesh generated by our algorithm is better than Global-SA, both in terms of ODT energy, number of slivers and distribution of radius ratio and dihedral angle.

A desirable distribution of radius ratios and dihedral angles aims to maintain high uniformity while ensuring that each simplex closely approaches the optimal values of 1 for the radius ratio and 60 degrees for the dihedral angle. The Global-SA method shows that while many elements approach the optimal values, some deviate significantly, even reaching extreme values. In contrast, our algorithm achieves a more concentrated distribution of radius ratios and dihedral angles within the simplices, resulting in improved overall uniformity. This indicates our algorithm is robust to the initial mesh and does not rely on matching the point



(a) Global-SA (b) 161 slivers $\leq 5°$ (c) 434 slivers$\leq 10°$

(d) Global-MWOA (e) 1 sliver$\leq 5°$ (f) 4 slivers$\leq 10°$

(g) Global-SA (h) 2 slivers $\leq 5°$ (i) 13 slivers$\leq 10°$

(j) Global-MWOA (k) 0 sliver$\leq 5°$ (l) 6 slivers$\leq 10°$

Fig. 17 Comparison of the meshes and its slivers between Global-SA and our algorithm with density function $\rho(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 + 0.2$ in eraser ball (top) with density sites near the boundary and with uniform sites around Hilbert cube (bottom)

initialization to the density function. It can generate meshes that conform to the density function regardless of the initial mesh configuration.

Furthermore, when generating tetrahedral meshes, our algorithm optimizes the ODT energy function, resulting in high-quality meshes with better average quality of elements. To demonstrate this, we compare our algorithm with existing mesh generation tools such as fTetWild [56], Gmsh [55], and TetGen [54] using the same input surface mesh and uniform density function. Table 5 presents the statistics of the quality of cubic meshes generated by different tools. The results highlight the superiority of our algorithm in terms of average radius ratio, average minimum/maximum dihedral angle. However, it should be noted that our algorithm requires more time due to the optimization process based on population.

Dataset We conducted experiments on the dataset presented in [57], which comprises a total of 108 models. As our algorithm aims to generate high-quality volumetric meshes, we excluded three open models from the dataset. We first preprocessed the remaining 105 models using CGAL, subjecting them to isotropic remeshing.

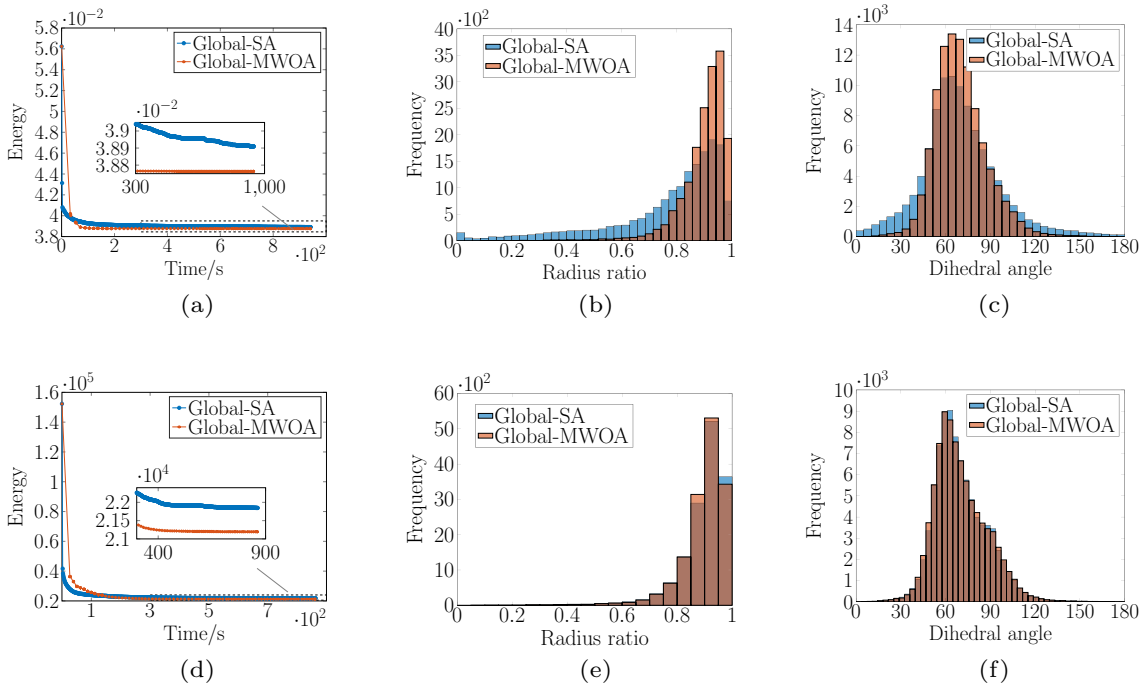


Fig. 18 Comparison of energy graph, the distribution of radius ratio and dihedral angle between Global-SA and our algorithm in eraser ball (top) and Hilbert cube (bottom)

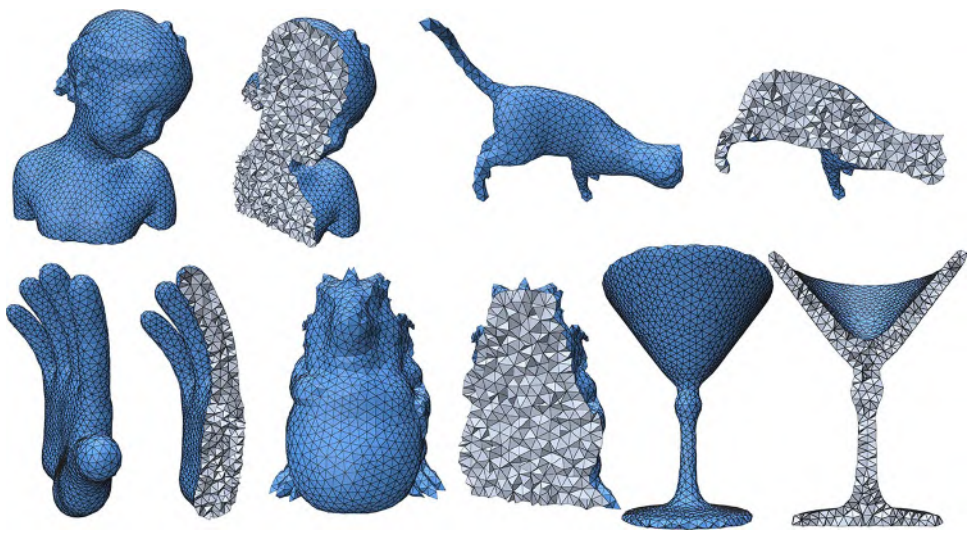


Fig. 19 Part models of the dataset

Subsequently, these remeshing models serve as input for our Global-MWOA method to produce high-quality volumetric meshes. Partial results of the high-quality meshes generated using the Global-MWOA method on this dataset are depicted in Fig. 19. We employed the fTetWild method to obtain results corresponding to the Global-MWOA method, with a comparable number of mesh vertices. Furthermore, we computed and analyzed the average radius ratio, as well as the distribution of average minimum/maximum dihedral angles for both fTetWild and Global-MWOA methods on the dataset. Figure 20 illustrates that, under this dataset, our method achieves superior average radius ratio and average minimum/maximum dihedral angle distributions compared to fTetWild.

Numerical examples An ODT implies the triangulation in which the vertex distribution aligns closely with the density function or exhibits overall better quality. For solving partial differential equations using finite element methods, the overall average mesh quality significantly influences the solution outcomes [21]. In this context, we consider testing a 3D

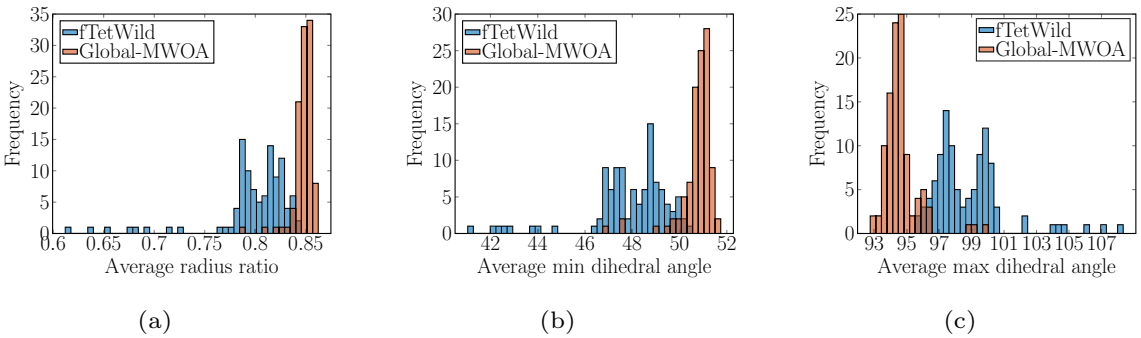
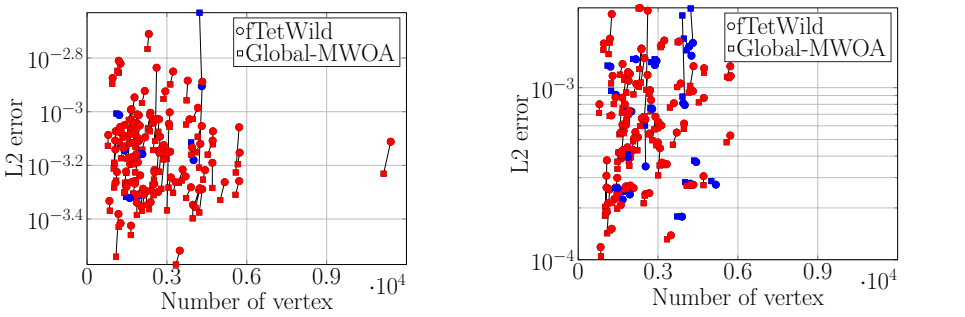


Fig. 20 The distribution of average radius ratio, average min/max dihedral angle of fTetWild and Global-MWOA under 105 models

Fig. 21 L_2 error vs. number of vertex for the dataset. Lines connect two points from the same model. The red instances in the pairs indicate models where the Global-MWOA method produces mesh solutions with lower errors compared to fTetWild, while the blue pairs represent the opposite scenario. **a**, **b** depict 98 and 83 red pairs, respectively



(a) $u_{exc}(\boldsymbol{x}) = x^2 + y^2 + z^2$ (b) $u_{exc}(\boldsymbol{x}) = ln(\sqrt{(x^2 + y^2 + z^2)})$

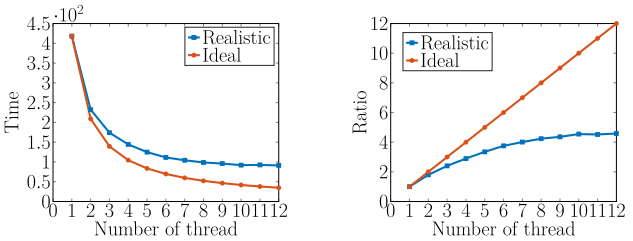


Fig. 22 Performance of acceleration effect. **a** The ideal time reduction and the realistic time reduction. **b** The ideal acceleration ratio and the realistic acceleration ratio

Poisson problem with an exact solution $u_{exc}(\boldsymbol{x}) = x^2 + y^2 + z^2$ and $u_{exc}(\boldsymbol{x}) = ln(\sqrt{(x^2 + y^2 + z^2)})$ in domain Ω, as stated in Eq. (15).

$$\begin{aligned} \Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}) \text{ in } \Omega \\ u(\boldsymbol{x}) &= g(\boldsymbol{x}) \text{ on } \partial\Omega \end{aligned} \tag{15}$$

We conducted tests on the dataset presented in [57], generating volumetric meshes using both fTeWild and Global-MWOA methods. The comparison involves assessing the normalized L_2 error between the approximate solution and the exact solution as $\|u_{exc} - u_h\|_{L_2} = \frac{\sqrt{\int_\Omega (u_{exc} - u_h)^2 \mathrm{d}\boldsymbol{x}}}{\sqrt{\int_\Omega u_{exc}^2 \mathrm{d}\boldsymbol{x}}}$. Utilizing FEATool Multiphysics [65] to solve the Poisson equation yieldedthe error results depicted in Fig. 21. It is evident that, inmany models, the Global-MWOA method yields volumetricmeshes with vertices similar to those of fTetWild and, inmost cases, exhibits smaller normalized L_2 errors when compared to fTetWild.

6.4 Acceleration effect

We have utilized OpenMP to optimize our algorithm on the CPU. Due to the limitation of the device's CPU core, we have only tested the performance using 1–12 threads. For evaluating the acceleration performance, we have selected the dumbbell example with 1000 interior points. The results obtained are presented in Fig. 22. As we increase the number of threads, the runtime decreases and the speedup ratio increases. However, as the number of threads continues to increase, the speedup ratio levels off. This is because we have only parallelized the optimization part during each iteration and had to wait for all

threads to complete before proceeding to the next iteration. Therefore, the runtime and acceleration ratio are likely to level off. We can observe that the best acceleration ratio is achieved when using 10 threads. Consequently, we set the number of threads to 10 in our subsequent experiments.

7 Conclusions

This paper presents a hybrid algorithm, that combines MWOA with L-BFGS to address the optimization problem of the ODT energy function. The algorithm takes advantage of the global search capabilities of MWOA and the local optimization capabilities L-BFGS. Additionally, OpenMP is employed to parallelize computations and decrease the running time for the local optimization process with poor initialization. The computational results demonstrate that our algorithm is robust to the initial mesh and achieves superior mesh quality compared to existing approaches. Furthermore, the algorithm exhibits improved running time efficiency.

One main drawback of our current algorithm is its relatively long running time, particularly in the 3D domain. In our algorithm, as the number of mesh vertices increases, large-scale matrix operations become prominent. GPUs, equipped with numerous small cores, are well-suited for accelerating such operations, demonstrating the potential to expedite our method. However, extending the algorithm to the GPU may encounter challenges in handling frequent branching and data transfers between CPU and GPU during the transition, which could impede performance improvement. Future work includes addressing the challenge of improving running time by extending the algorithm to utilize GPU acceleration. Additionally, we plan to explore the application of our algorithm to other problems, such as mesh simplification. Similar to the ODT energy, there are various forms of energy composed of the Jacobian matrix, employed for assessing mesh quality. In the future, we aim to extend the algorithm to encompass a broader range of energies derived from the Jacobian matrix, enhancing its applicability in diverse scenarios.

Acknowledgements This work was supported by the National Key R&D Program of China (No. 2022YFB3303400), National Natural Science Foundation of China (Nos. 61972327, 62272402, and 62372389), Natural Science Foundation of Fujian Province (No. 2022J01001), and Fundamental Research Funds for the Central Universities (No. 20720220037).

Author contributions YW: Methodology, Investigation, Software, Visualization, Writing-Original draft preparation, Writing-Reviewing and Editing. JC, ZC: Methodology, Validation, Supervision, Formal analysis, Writing-Reviewing and Editing.

Data availability The data that support the findings of this study are available upon reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Carey GF (1997) Computational grids: generations, adaptation & solution strategies. CRC Press, Boca Raton
2. Teng SH, Wong CW (2000) Unstructured mesh generation: theory, practice, and perspectives. Int J Comput Geom Appl 10(03):227–266
3. Baker TJ (2005) Mesh generation: art or science? Prog Aerosp Sci 41(1):29–63
4. Zhang Y (2013) Challenges and advances in image-based geometric modeling and mesh generation. Springer, Berlin
5. Zhang YJ (2018) Geometric modeling and mesh generation from scanned images. CRC Press, Boca Raton
6. Shewchuk JR (2012) Unstructured mesh generation. Comb Sci Comput 12(257):2
7. Owen SJ (1998) A survey of unstructured mesh generation technology. IMR 239(267):15
8. Alliez P, De Verdière ÉC, Devillers O et al (2005) Centroidal Voronoi diagrams for isotropic surface remeshing. Graph Model 67(3):204–231
9. Hu Y, Schneider T, Gao X et al (2019) Triwild: robust triangulation with curve constraints. ACM Trans Graph (TOG) 38(4):1–15
10. George JA (1971) Computer implementation of the finite element method. Stanford University, Stanford
11. Zhou Q, Wang Q, Yu Z (2022) SAFT: shotgun advancing front technique for massively parallel mesh generation on graphics processing unit. Int J Numer Methods Eng 123(18):4391–4406
12. Yerry M, Shephard M (1983) A modified quadtree approach to finite element mesh generation. Comput Graph Appl 3(01):39–46
13. Jaillet F, Lobos C (2022) Fast quadtree/octree adaptive meshing and re-meshing with linear mixed elements. Eng Comput 38(4):3399–3416
14. Aurenhammer F, Klein R, Lee DT (2013) Voronoi diagrams and Delaunay triangulations. World Scientific Publishing Company, Singapore
15. Cheng SW, Dey TK, Shewchuk J et al (2013) Delaunay mesh generation. CRC Press, Boca Raton
16. Guo J, Yan DM, Chen L et al (2016) Tetrahedral meshing via maximal Poisson-disk sampling. Comput Aided Geom Des 43:186–199
17. Liang X, Zhang Y (2014) An octree-based dual contouring method for triangular and tetrahedral mesh generation with guaranteed angle range. Eng Comput 30:211–222
18. Canann SA, Stephenson MB, Blacker T (1993) Optismoothing: an optimization-driven approach to mesh smoothing. Finite Elem Anal Des 13(2–3):185–190
19. Canann SA, Muthukrishnan S, Phillips R (1996) Topological refinement procedures for triangular finite element meshes. Eng Comput 12(3):243–255
20. Chen L, Xu J (2004) Optimal Delaunay triangulations. J Comput Math 22:299–308
21. Chen Z, Wang W, Lévy B et al (2014) Revisiting optimal Delaunay triangulation for 3D graded mesh generation. SIAM J Sci Comput 36:A930–A954
22. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
23. Abdel-Basset M, Mohamed R, AbdelAziz NM et al (2022) HWOA: a hybrid whale optimization algorithm with a novel

local minima avoidance method for multi-level thresholding color image segmentation. Expert Syst Appl 190:116145
24. Du Q, Faber V, Gunzburger M (1999) Centroidal Voronoi tessellations: applications and algorithms. SIAM Rev 41(4):637–676
25. Chen L (2004) Mesh smoothing schemes based on optimal Delaunay triangulations. IMR 109–120
26. Alliez P, Cohen-Steiner D, Yvinec M et al (2005) Variational tetrahedral meshing. In: ACM SIGGRAPH 2005 papers, pp 617–625
27. Tournois J, Wormser C, Alliez P et al (2009) Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. ACM Trans Graph (TOG) 28(3):1–9
28. Gao Z, Yu Z, Holst M (2012) Quality tetrahedral mesh smoothing via boundary-optimized Delaunay triangulation. Comput Aided Geom Des 29(9):707–721
29. Chen L, Holst M (2011) Efficient mesh optimization schemes based on optimal Delaunay triangulations. Comput Methods Appl Mech Eng 200(9–12):967–984
30. Chen Z, Cao J, Yang C (2011) Topology improvement for constructing optimal Delaunay triangulation. J Comput-Aided Des Comput Graph 23:1967–1974
31. Yq Hai, Yf Guo, Dong M et al (2022) Enhanced optimal Delaunay triangulation methods with connectivity regularization. Appl Math-A J Chin Univ 37(3):453–469
32. Chen L, Sun P, Xu J (2007) Optimal anisotropic meshes for minimizing interpolation errors in L^p norm. Math Comput 76(257):179–204
33. Feng L, Alliez P, Busé L et al (2018) Curved optimal Delaunay triangulation. ACM Trans Graph (TOG) 37(4):1–16
34. Yan DM, Wang W, Lévy B et al (2013) Efficient computation of clipped Voronoi diagram for mesh generation. Comput Aided Des 45(4):843–852
35. Chen Z, Zhang T, Cao J et al (2018) Point cloud resampling using centroidal Voronoi tessellation methods. Comput Aided Des 102:12–21
36. Dong X, Chen Z, Liu YJ et al (2021) GPU-based supervoxel generation with a novel anisotropic metric. IEEE Trans Image Process 30:8847–8860
37. Liu Y, Wang W, Lévy B et al (2009) On centroidal Voronoi tessellation-energy smoothness and fast computation. ACM Trans Graph (TOG) 28(4):1–17
38. Lu L, Sun F, Pan H et al (2012) Global optimization of centroidal Voronoi tessellation with Monte Carlo approach. IEEE Trans Vis Comput Graph 18(11):1880–1890
39. Lloyd S (1982) Least squares quantization in PCM. IEEE Trans Inf Theory 28(2):129–137
40. Liu YJ, Xu CX, Yi R et al (2016) Manifold differential evolution (MDE) a global optimization method for geodesic centroidal Voronoi tessellations on meshes. ACM Trans Graph (TOG) 35(6):1–10
41. Ye Z, Yi R, Yu M et al (2019) Geodesic centroidal Voronoi tessellations: theories, algorithms and applications. arXiv preprint arXiv:1907.00523
42. Wang X, Ying X, Liu YJ et al (2015) Intrinsic computation of centroidal Voronoi tessellation (CVT) on meshes. Comput Aided Des 58:51–61
43. Yan DM, Lévy B, Liu Y et al (2009) Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Comput Graph Forum 28(5):1445–1454
44. Hou W, Zong C, Wang P et al (2022) SDF-RVD: restricted Voronoi diagram on signed distance field. Comput Aided Des 144:103166
45. Telsang B, Djouadi SM (2022) Computation of centroidal Voronoi tessellations in high dimensional spaces. IEEE Contr Syst Lett 6:3313–3318
46. Du Q, Wang D (2005) Anisotropic centroidal Voronoi tessellations and their applications. SIAM J Sci Comput 26(3):737–761
47. Lévy B, Liu Y (2010) L^p centroidal Voronoi tessellation and its applications. ACM Trans Graph (TOG) 29(4):1–11
48. Ekelschot D, Ceze M, Garai A et al (2018) Robust metric aligned quad-dominant meshing using L^p centroidal Voronoi tessellation. In: 2018 AIAA Aerospace Sciences Meeting, p 1501
49. Ekelschot D, Ceze M, Murman SM et al (2019) Parallel high-order anisotropic meshing using discrete metric tensors. In: AIAA Scitech 2019 forum, p 1993
50. Xiao Y, Chen Z, Cao J et al (2018) Optimal power diagrams via function approximation. Comput Aided Des 102:52–60
51. Mamatha TM, Venkatesh B (2019) Numerical integration over arbitrary tetrahedral element by transforming into standard 1-cube. IOP Conf Ser Mater Sci Eng 577(1):012172
52. Zhang WX, Wang Q, Guo JP et al (2022) Constrained remeshing using evolutionary vertex optimization. Comput Graph Forum 41(2):237–247
53. Persson PO, Strang G (2004) A simple mesh generator in matlab. SIAM Rev 46(2):329–345
54. Hang S (2015) Tetgen, a Delaunay-based quality tetrahedral mesh generator. ACM Trans Math Softw 41(2):11
55. Geuzaine C, Remacle JF (2009) Gmsh: a 3-D finite element mesh generator with built-in pre-and post-processing facilities. Int J Numer Methods Eng 79(11):1309–1331
56. Hu Y, Schneider T, Wang B et al (2020) Fast tetrahedral meshing in the wild. ACM Trans Graph (TOG) 39(4):117–1
57. Yang Y, Zhang WX, Liu Y et al (2020) Error-bounded compatible remeshing. ACM Trans Graph (TOG) 39(4):113–1
58. Frey PJ, Borouchaki H (1999) Surface mesh quality evaluation. Int J Numer Methods Eng 45(1):101–118
59. Tecchio C, Basso E, de Azevedo JLF et al (2014) Mesh improvement for multiblock grids in store separation problems. In: CONEM 2014
60. Jamin C, Alliez P, Yvinec M et al (2015) CGALmesh: a generic framework for Delaunay mesh generation. ACM Trans Math Softw 41(4):1–24
61. Liu Y (2010) HLBFGS. https://xueyuhanlang.github.io/software/HLBFGS/
62. Gaël Guennebaud BJ et al (2018) Eigen. https://eigen.tuxfamily.org/
63. Sharp N et al (2019) Polyscope. www.polyscope.run
64. Chen Z, Yuan Z, Choi YK et al (2012) Variational blue noise sampling. IEEE Trans Vis Comput Graph 18(10):1784–1796
65. Precise Simulation (2023) FEATool Multiphysics. GitHub https://github.com/precise-simulation/featool-multiphysics