# 基于深度 Q 网络的平面域 Delaunay 网格优化算法

张浩杰, 刘星, 李鸿晶*

(南京工业大学工程力学研究所　南京　211816)
(hjing@njtech.edu.cn)

**摘　要:** 网格优化是 Delaunay 网格生成后的必要步骤, 对于保证数值模拟的可靠性至关重要. 为了改善平面域 Delaunay 网格的质量, 提出一种基于深度 Q 网络(deep Q network, DQN)的网格优化算法. 首先, 对初始网格进行质量评估, 选出不满足要求的单元结点, 并将其按质量升序排列; 其次, 将结点移动描述为 Markov 决策过程, 建立并训练 DQN 模型; 再次, 利用模型训练后的经验参数加速网格质量优化; 最后, 以实际的隧道、气缸体、机械零件等为背景构建测试算例, 验证算法的适用性和可靠性, 并与既有典型算法进行对比试验. 研究结果表明, 本文算法能显著提高畸变单元的质量, 优化后的网格质量分布更为集中, 且优化过程不会产生无效单元.

**关键词:** Delaunay 网格; 网格优化; 深度 Q 网络; 深度强化学习
**中图法分类号:** TP391.41　　**DOI:** 10.3724/SP.J.1089.2022.19247

# Deep Q Network-Based Optimization Algorithm for Planar Delaunay Mesh

Zhang Haojie, Liu Xing, and Li Hongjing*

(*Engineering Mechanics Institute, Nanjing Tech University, Nanjing*　211816)

**Abstract:** It is necessary to conduct mesh optimization after generating Delaunay mesh, which is essential to ensure the reliability of numerical simulation. To improve the quality of Delaunay mesh on the plane domain, a mesh optimization algorithm based on deep Q network (DQN) is proposed. Firstly, the quality of the initial mesh is evaluated, and the element nodes that do not meet the quality requirements are selected and arranged in ascending order of their quality. Secondly, the node movement is described as Markov decision process, and the DQN model is established and trained. Thirdly, the empirical parameters of the model training are used to accelerate the optimal mesh quality. Finally, several test examples from practical tunnel, cylinder block, mechanical parts, etc., are employed to verify the applicability and reliability of the proposed algorithm. Compared with the existing typical algorithms, the test results show that the proposed algorithm can significantly improve the quality of poor elements, the quality distribution of optimized mesh will be more concentrated, and no invalid elements are produced during the optimization process.

**Key words:** Delaunay mesh; mesh optimization; deep Q network; deep reinforcement learning

有限元法(finite element method, FEM)是目前在求解科学和工程问题时被广泛应用的一种数值计算方法, 构建分解计算域的网格是有限元分析的基础性工作. 在有限元网格生成方法中, Delaunay 三角剖分算法是目前最流行的、通用的全自动网格生成方法之一[1]. 然而, 受内部结点插入、边

收稿日期: 2021-07-21; 修回日期: 2022-03-18. **基金项目**: 国家自然科学基金委员会-中国地震局地震科学联合基金(U2039208); 江苏省研究生科研创新计划(KYCX21_1144). **张浩杰**(1997—), 男, 硕士研究生, 主要研究方向为机器学习算法、有限元网格生成方法; **刘星**(2000—), 男, 在校学生, 主要研究方向为机器学习算法; **李鸿晶**(1966—), 男, 博士, 教授, 博士生导师, 论文通信作者, 主要研究方向为结构动力学、工程抗震.

界约束和单元尺寸过渡等因素的影响, Delaunay 三角剖分算法生成的初始网格一般会在局部区域内出现或多或少的畸变单元. 这些畸变单元的存在将直接影响后续有限元计算的精度和效率, 若畸变单元的质量太差, 则计算过程无法进行. 因此, 网格质量优化是网格生成后、数值模拟前的必要步骤[2].

网格质量优化的效果取决于优化策略的选择. 根据网格拓扑结构是否发生变化, 优化策略主要分为几何优化和拓扑优化[3]. 在实际的网格优化过程中, 这 2 种策略通常结合使用. 几何优化又被称为网格光顺, 与拓扑优化相比, 其不会改变结点个数和拓扑关系, 易于实现且应用广泛. 常用的网格光顺算法大致分为基于几何的光顺和基于优化的光顺 2 类.

在基于几何的光顺算法中, 最具有代表性的就是 Laplacian 光顺[4], 其特点是计算简便, 但对凹形区域进行优化时, 可能产生无效单元. Freitag[5]改进了 Laplacian 光顺方法, 提出只有当网格质量得到提升时方可移动结点. 虽然该方法能保证单元的有效性, 但结点位置并不是最优的. 而基于优化的光顺算法一般是将单元质量度量指标构建成目标函数, 采用如最速下降法[6-7]、共轭梯度法[8-9]、牛顿法[10-11]、高斯-牛顿法[12]和下山单纯形法[13]等最优化方法, 通过对目标函数最小化或最大化来获取移动方向. 其中质心 Voronoi 图法(centroidal Voronoi tessellation, CVT)[14]和最优 Delaunay 三角剖分法(optimal Delaunay triangulation, ODT)[15]是目前应用较广的 2 种基于优化的网格光顺算法. 该类方法虽然可实现较高质量的网格优化, 但计算复杂度较高.

随着数值最优化方法的蓬勃发展, 许多学者开始探索启发式算法在网格生成以及优化中的应用, 试图寻找一种合适且高效的启发式算法. Fabritius 等[16]将遗传算法应用于有限体积网格的生成, 实现了多目标的控制优化. Yilmaz 等[17]采用粒子群算法优化目标函数, 提出一种六面体网格的局部结点重定位方法. Yang 等[18]将结点移动过程转换为随机过程, 采用 Markov 链蒙特卡洛方法搜索最佳移动方向. Kim 等[19]利用深度神经网络(deep neural networks, DNN)预测结点的最优位置, 虽然该方法能够提升网格质量, 但需要上万个带有标签的样本, 训练成本较高.

强化学习(reinforcement learning, RL)[20-21]是近年飞速发展的一种启发式算法, 也是最具自我学习性的机器学习方法, 在处理复杂的连续决策问题方面, 具有得天独厚的优势. 而深度 Q 网络(deep Q network, DQN)[22]是经典的深度强化学习算法, 其将 RL 和 DNN 相结合, 极大地提高了自我学习效率. 因此, 本文提出一种以 DQN 模型为决策指南的网格优化算法. 该算法将 Delaunay 网格中的结点移动转化为序列决策问题, 以单元质量最大化为目标建立 DQN 模型, 依次指导质量较差的内部结点移动来实现网格优化的效果.

## 1 算法原理

### 1.1 RL

RL 的基本思想是智能体以“试错”的方式不断与给定环境交互, 从而获得相应奖励, 根据奖励动态地调整行为策略. 整个交互流程可由 Markov 决策过程(Markov decision process, MDP)描述.

如图 1 所示, 智能体在 $t$ 时刻感知周围环境状态 $\boldsymbol{s}_t$, 根据决策规则执行动作 $\boldsymbol{a}_t$, 进而到达新的环境状态 $\boldsymbol{s}_{t+1}$, 获得即时奖励 $r_t$, 智能体不断探索, 直至终止状态时结束交互. RL 的目标是学习最优动作策略 $\boldsymbol{\pi}^*$, 使得该时刻的总未来奖励最大.

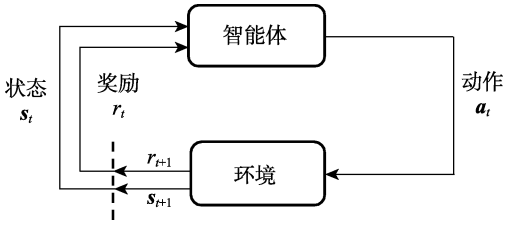



图 1 MDP 中智能体与环境的交互过程

### 1.2 Q-learning 算法

Q-learning 算法[23]是基于值函数的 RL 算法, 其动作的选择取决于动作值函数 $Q(\boldsymbol{s},\boldsymbol{a})$ 的大小. $Q(\boldsymbol{s},\boldsymbol{a})$ 值的计算一般采用时间差分法的方式进行求解, 即通过最大化下一个状态的 $Q(\boldsymbol{s}',\boldsymbol{a}')$ 值来更新当前 $Q(\boldsymbol{s},\boldsymbol{a})$ 值, 即

$$Q(\boldsymbol{s},\boldsymbol{a})=Q(\boldsymbol{s},\boldsymbol{a})+\alpha\left[r+\gamma\max_{\boldsymbol{a}'}Q(\boldsymbol{s}',\boldsymbol{a}')-Q(\boldsymbol{s},\boldsymbol{a})\right] \quad (1)$$

其中, $\alpha$ 为学习率; $\gamma\in[0,1]$ 为奖励衰减因子, 表示后续状态奖励对当前奖励的影响. Q-learning 算法的 $Q(\boldsymbol{s},\boldsymbol{a})$ 值通常用表格形式表示, 因此, 状态空间和动作空间需均为离散空间且不能太大.

### 1.3 DQN 算法

为了提高 Q-learning 算法在高维连续状态空间问题的适用性, DeepMind 公司提出了 DQN 算法. DQN 算法利用 DNN 近似逼近 $Q(\boldsymbol{s},\boldsymbol{a})$ 值, 即

$$Q(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\theta})\approx Q^{*}(\boldsymbol{s},\boldsymbol{a}) \tag{2}$$

DQN 算法引入 2 个结构完全相同的神经网络: 一个用于更新神经网络参数 $\boldsymbol{\theta}$ 的现实 Q 网络; 另一个用于计算 $Q$ 值的目标 Q 网络, 以此减少神经网络参数 $\boldsymbol{\theta}$ 更新和 $Q$ 值计算之间的依赖关系. 此外, DQN 算法采用经验回放机制, 将每次与环境交互后的样本信息保存至记忆库, 然后在训练过程中随机采样, 采用批量梯度下降的方式更新网络参数, 减轻单一交互序列的波动性, 提高样本的利用率. 通过神经网络参数的更新, 可根据式(2)重新计算 $Q$ 值, 从而最优策略表示为

$$\boldsymbol{\pi}^{*}=\arg\max Q(\boldsymbol{s},\boldsymbol{a};\boldsymbol{\theta}) \tag{3}$$

## 2　本文算法

本文算法主要包括筛选、缩放、优化、还原和更新 5 个部分. 图 2 所示为本文算法的流程图, 具体步骤如下.

输入. 初始网格 $M$.

输出. 优化后的网格 $M'$.

Step1. 计算每个内部结点 $v$ 的相邻单元质量 $q_e$ 的最小值, 作为该结点对应的质量 $q_v$.

Step2. 若结点质量 $q_v$ 小于阈值 $q_\varepsilon$, 则将该结点 $v$ 添加至点集 $V$, 并按质量升序排列.

Step3. 对点集 $V$ 中的每个结点 $v$, 执行:

Step3.1. 将结点 $v$ 与邻近结点的坐标缩放至 $[0,1]\times[0,1]$ 尺度空间, 以保证后续模型设计的状态空间尺度相同;

Step3.2. 根据邻近顶点个数, 使用对应的 DQN 模型确定移动位置;

Step3.3. 计算移动后的结点质量, 检查质量是否收敛, 若质量收敛或达到最大迭代次数, 则停止迭代, 还原结点 $v$ 的坐标并将其移出点集; 否则, 转 Step3.2.

Step4. 若 $V$ 为空集, 则更新网格内各结点坐标; 否则, 转 Step3.

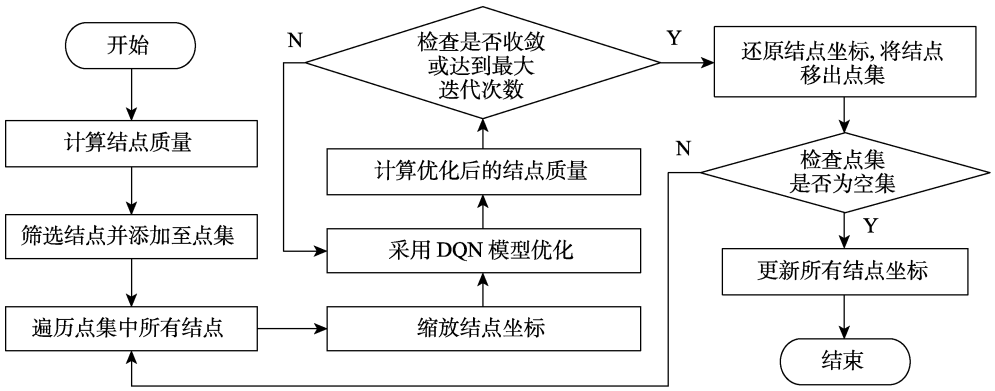



图 2　本文算法流程图

## 3　DQN 模型设计与训练

### 3.1　DQN 模型设计

DQN 模型设计的关键是将 MDP 的交互对象设定与结点移动紧密联系在一起, 本文设定如图 3 所示.

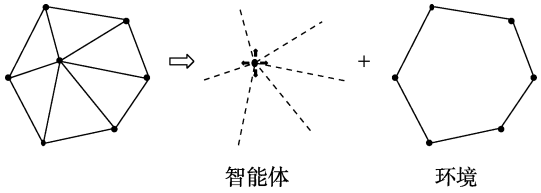



图 3　内部结点移动的交互对象表示

本文将待移动的内部结点 $v$ 抽象为智能体, 将邻近顶点围成的多边形抽象为环境, 从而将整个结点移动过程转化为智能体与环境的交互过程, 依据 MDP 框架分别定义模型的状态空间、动作空间以及奖励函数.

(1) 状态空间 $\boldsymbol{S}$ 是神经网络的输入, 描述了内部结点 $v$ 与相邻结点的坐标信息, 具体定义为

$$\boldsymbol{S}=\{\boldsymbol{s}_0,\boldsymbol{s}_1,\boldsymbol{s}_2,\cdots,\boldsymbol{s}_n\} \tag{4}$$

其中, $\boldsymbol{s}_i=(x,y)$, $i=0,1,2,\cdots,n$; $x$ 和 $y$ 分别表示横纵坐标; $\boldsymbol{s}_0$ 为内部结点 $v$ 的坐标向量; $n$ 为邻近顶点个数.

(2) 动作空间 $\boldsymbol{A}$ 是神经网络的输出, 描述了内部结点 $v$ 在每一时间步可采取的动作, 其基本动作主要有 4 个方向, 分别为向上、向下、向左和向右移动. 参考宫翔飞等[6]的研究成果, 将结点移动的步长均取为 $0.05\,l_{\min}$; 其中, $l_{\min}$ 表示邻近顶点围成的多边形的最短边长.

(3) 奖励函数 $R$ 的设置对于结点移动策略优

化和结点质量最大化至关重要. 因此, 奖励函数主要由结点移动前后的质量差决定. 此外, 当结点质量达到阈值时, 应给予极大奖励; 而当结点移动到邻近顶点围成的多边形以外, 即超出边界位置时, 应给予负奖励, 如表 1 所示.

**表 1 DQN 模型的奖励函数设置表**

| 状态 | 奖励函数 |
|---|---|
| 正常情况 | $(q_{v,t}-q_{v,t-1})\times 100$ |
| 结点质量满足阈值 | 10 |
| 结点移动到超出边界位置 | −10 |

表 1 中的 $q_{v,t-1}$ 和 $q_{v,t}$ 分别表示在 $t$ 时刻结点 $v$ 移动前后的质量. 本文模型采用的单元质量度量指标 $q_e$ 为三角单元的面边比[24], 即

$$q_e = 4\sqrt{3}A_e \Big/ \sum_{i=1}^{3} l_i^2 \tag{5}$$

其中, $A_e$ 表示单元面积; $l$ 表示单元边长. 当面边比趋近于 1 时, 单元形状向理想的正三角形靠近, 质量逐渐提升.

### 3.2 模型训练

由于每个待移动结点 $v$ 所在的邻近顶点个数 $n$ 不固定, 因此, 需要依据邻近顶点个数建立不同的 DQN 模型. 考虑邻近顶点个数 $n$ 为 3~10, 本文分别建立 8 种 DQN 模型. 在模型训练过程中, 每种模型的 2 个 Q 网络结构相同. 第 1 层的输入层神经单元分别对应着状态空间 $\boldsymbol{S}$ 里所有结点的坐标信息, 故有 $2(n+1)$ 个神经单元; 第 2 层的全连接层有 20 个神经单元, 激活函数为 ReLU 函数; 第 3 层的输出层神经单元分别对应着结点 $v$ 在当前状态下采用动作空间 $\boldsymbol{A}$ 里 4 种动作所计算得到的 $Q$ 值, 故有 4 个神经单元.

训练参数设置如下: 学习率 $\alpha$ =0.01, 奖励衰减因子 $\gamma$ =0.9, 探索率 $\varepsilon$ =0.9, 记忆库 $\boldsymbol{D}$ 容量 $N$ =500, 批量梯度下降样本数 $m$ =32, 优化器采用 RMSProp, 最大回合数 $M$ =5 000, 单个回合的最大步长 $T$ =200, 目标 Q 网络参数的更新频率 $C$ =100. 神经网络参数 $\boldsymbol{\theta}$ 更新的具体过程如下.

输入. 状态空间 $\boldsymbol{S}$, 动作空间 $\boldsymbol{A}$, 学习率 $\alpha$, 衰减因子 $\gamma$, 探索率 $\varepsilon$, 现实 Q 网络 $Q$, 目标 Q 网络 $Q'$, 批量数 $m$, 最大回合数 $M$, 最大步长 $T$, 目标 Q 网络更新频率 $C$.

输出. 现实 Q 网络参数 $\boldsymbol{\theta}$.

Step1. 初始化记忆库 $\boldsymbol{D}$、现实 Q 网络参数 $\boldsymbol{\theta}$ 和目标 Q 网络参数 $\boldsymbol{\theta}'$.

Step2. 对于每轮回合, 执行:

Step2.1. 初始化状态 $\boldsymbol{S}_t$, 并将其作为状态序列的第 1 个状态;

Step2.2. 在现实 Q 网络 $Q$ 中将 $\boldsymbol{S}_t$ 作为输入, 用 $\varepsilon$-greedy 策略法选取动作 $\boldsymbol{A}_t$, 获得奖励 $R_t$, 进入 $\boldsymbol{S}_{t+1}$;

Step2.3. 将 $\{\boldsymbol{S}_t, \boldsymbol{A}_t, R_t, \boldsymbol{S}_{t+1}\}$ 四元组存入记忆库 $\boldsymbol{D}$, 并将状态 $\boldsymbol{S}_t$ 更新为 $\boldsymbol{S}_{t+1}$;

Step2.4. 从记忆库 $\boldsymbol{D}$ 中随机采样 $m$ 个样本 $\{\boldsymbol{S}_j, \boldsymbol{A}_j, R_j, \boldsymbol{S}_{j+1}\}$, 若 $\boldsymbol{S}_{j+1}$ 为终止状态, 则现实 $Q$ 值 $y_j = R_j$; 否则,

$$y_j = R_j + \gamma \max Q'(\boldsymbol{S}_{j+1}, \boldsymbol{A}_{j+1}, \boldsymbol{\theta}') \tag{6}$$

Step2.5. 对 $\left(y_j - Q(\boldsymbol{S}_j, \boldsymbol{A}_j, \boldsymbol{\theta})\right)^2$ 进行梯度下降更新现实 Q 网络参数 $\boldsymbol{\theta}$;

Step2.6. 每隔 $C$ 个步长, 更新一次目标 Q 网络参数 $\boldsymbol{\theta}'$.

Step2.7. 若达到最大步长 $T$, $\boldsymbol{S}_{j+1}$ 是终止状态, 则迭代结束, 转 Step3; 否则, 转 Step2.2.

Step3. 若达到最大回合数 $M$, 则停止训练; 否则, 转至 Step2.

待神经网络参数 $\boldsymbol{\theta}$ 更新后, 可通过式(2)计算 $Q$ 值, 然后根据式(3)确定结点的移动策略.

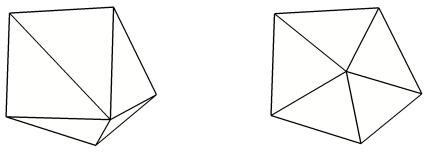
## 4 算例分析

本文基于神经网络框架 TensorFlow 1.14.0 利用 Python 3.6 语言搭建了 DQN 模型及训练算法. 各 DQN 模型在测试前均采用未优化的网格样本训练 5 000 次, 故在测试阶段直接加载已被训练过的模型进行测试.

计算机配置为 AMD Ryzen 7 4800H CPU, 16 GB 内存. 本文将通过 5 个算例分别阐述本文算法的有效性和适用性.

### 4.1 有效性分析

算例 1 选取了仅含 1 个内部结点的五边形网格. 图 4 所示为算例 1 优化前后的对比图, 从中可以明显看出网格经过本文算法优化后的效果.



a. 优化前 b. 优化后

图 4 算例 1 优化前后对比图

如图 5 所示, 算例 1 的内部结点在移动 30 步后, 基本稳定在结点质量约为 0.92 的局部最优位置, 说明本文算法是可行的和有效的.

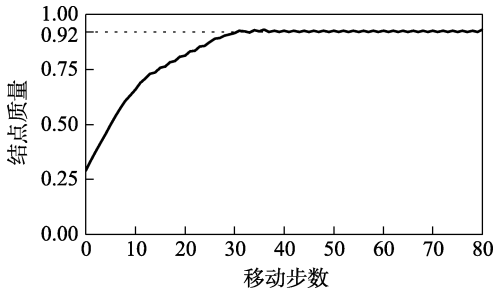



图 5　结点质量收敛图

图 6 展示了该内部结点的移动轨迹, 进一步证明本文算法能指导结点以最短步数移动到质量局部最优位置.

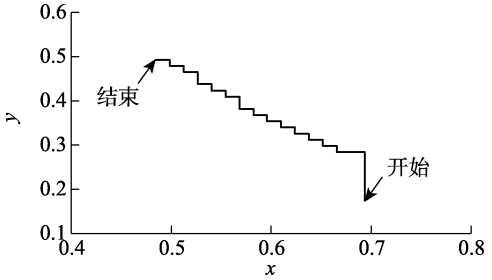



图 6　结点移动轨迹图

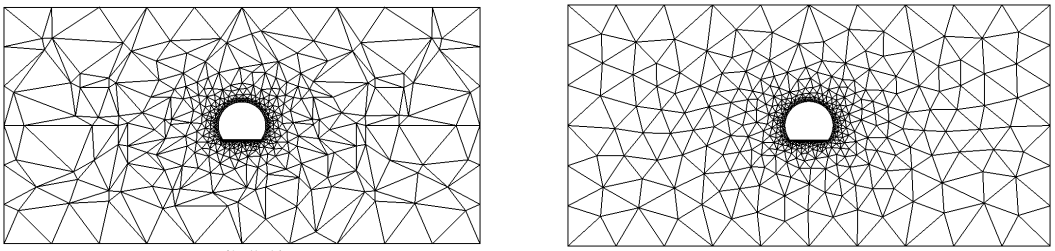
### 4.2　适用性分析

算例 2~算例 5 分别选取了内部结点分布显著不均的隧道网格、分布较为均匀的隧道网格、某气缸体截面网格以及某机械零件网格. 算例 2 包含 523 个结点和 926 个单元; 算例 3 包含 526 个结点和 963 个单元; 算例 4 包含 1 059 个结点和 1 880 个单元; 算例 5 包含 1 505 个结点和 2 662 个单元.

图 7~图 10 所示分别为算例 2~算例 5 优化前后的对比图. 初始网格均是通过有限元网格生成器 Gmsh[25]采用 Delaunay 三角剖分算法生成处理得到的.

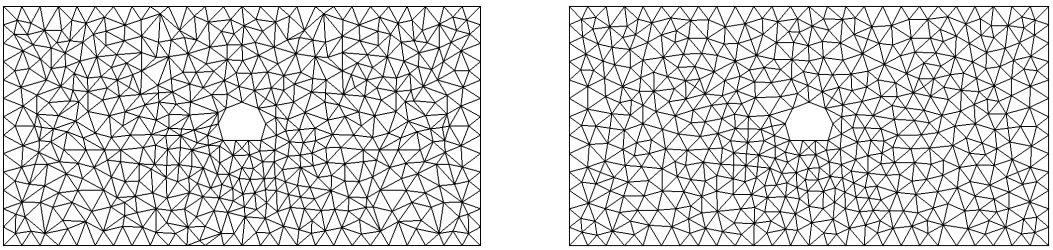
从图 7~图 10 可以看出, 本文算法在不同单元尺度、不同几何外形的网格中均能表现较好的优化效果. 此外, 本文还采用 Laplacian 光顺[4], CVT[14]以及 ODT 算法[15]对算例 2~算例 5 进行优化对比.

表 2 给出了算例 2~算例 5 经过 4 种不同算法优化后的测试结果. 其中, $\bar{q}$ 表示网格的平均单元质量; $q_{\min}$ 表示网格的最差单元质量; $t$ 表示优化算法的运行时间.



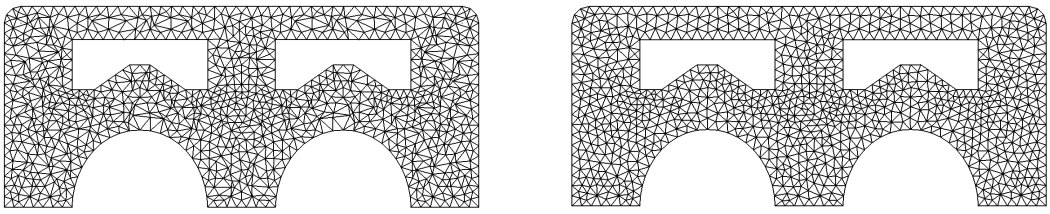
a. 优化前　　b. 优化后

图 7　算例 2 优化前后对比图



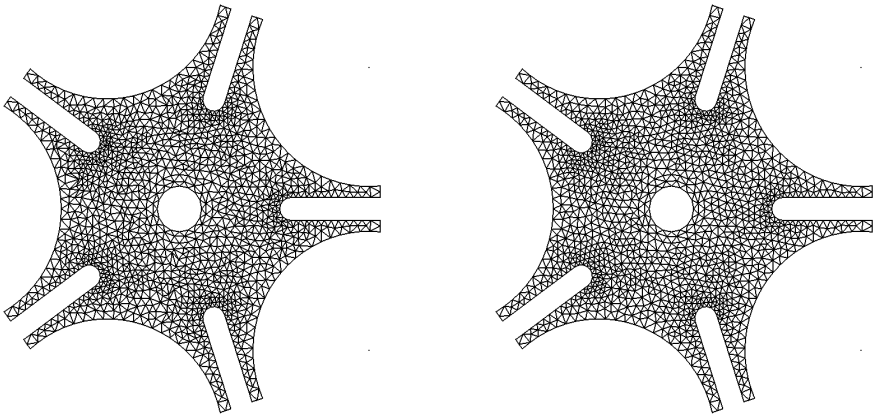
a. 优化前　　b. 优化后

图 8　算例 3 优化前后对比图



a. 优化前　　b. 优化后

图 9　算例 4 优化前后对比图



a. 优化前 b. 优化后

图 10 算例 5 优化前后对比图

**表 2 不同算法网格优化测试结果**

| 网格类型 | 算例 2 | | | 算例 3 | | | 算例 4 | | | 算例 5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\bar{q}$ | $q_{\min}$ | $t$ | $\bar{q}$ | $q_{\min}$ | $t$ | $\bar{q}$ | $q_{\min}$ | $t$ | $\bar{q}$ | $q_{\min}$ | $t$ |
| 初始网格 | 0.848 | 0.290 | | 0.882 | 0.377 | | 0.865 | 0.447 | | 0.907 | 0.380 | |
| Laplacian 光顺[4] | 0.902 | 0.552 | 1.96 | 0.942 | 0.732 | 1.65 | 0.945 | 0.754 | 3.01 | 0.945 | 0.744 | 5.83 |
| CVT 算法[14] | 0.915 | 0.645 | 2.26 | 0.942 | 0.779 | 2.36 | 0.946 | 0.735 | 3.44 | 0.947 | 0.745 | 6.64 |
| ODT 算法[15] | **0.935** | 0.749 | 2.73 | 0.943 | 0.751 | 2.90 | **0.950** | 0.763 | 4.62 | 0.950 | 0.747 | 7.39 |
| 本文算法 | 0.933 | **0.768** | 2.34 | **0.944** | **0.797** | 2.27 | 0.948 | **0.809** | 3.35 | **0.951** | **0.751** | 6.23 |

注. 粗体表示最优结果.

从表 2 可以看出, Laplacian 光顺的运行时间最短, 但由于部分区域的几何外形, 导致结点质量无法得到有效提升, 优化后的网格整体质量最低, 尤其是对于算例 2 单元密度显著不均的网格. 而 CVT 算法和 ODT 算法由于计算复杂度较高, 使得运行时间相对较长. 虽然 CVT 算法的运行时间短于 ODT 算法, 但 ODT 算法优化后的网格质量总体要优于 CVT 算法. 与其他 3 种算法相比, 本文算法优化后的网格中最差单元质量最高, 平均单元质量也较高, 几乎与优化效果较好的 ODT 算法相同, 且运行时间明显短于 ODT 算法.

值得注意的是, 有限元网格的最差单元质量直接影响刚度矩阵的求解计算. 最差单元质量越高, 刚度矩阵的条件数就越小, 有限元计算的复杂度和离散误差也就越小. 因此, 本文算法优化后的网格在总体质量较高的前提下, 最差单元质量最高. 这意味着本文算法优化后的网格更有利于获得较为精确的有限元计算结果.

此外, 由于算例 3~算例 5 经过 4 种算法优化后的网格平均质量相近, 本文进一步计算各网格的质量方差, 结果如图 11 所示.

从图 11 可以观察到, 本文算法优化后的网格质量方差最小, 说明网格质量分布更为集中, 各单元质量高度趋于同一质量值. 这意味着本文算法的优化结果使结构的刚度矩阵中的元素不致相差太大, 进一步减少有限元误差.

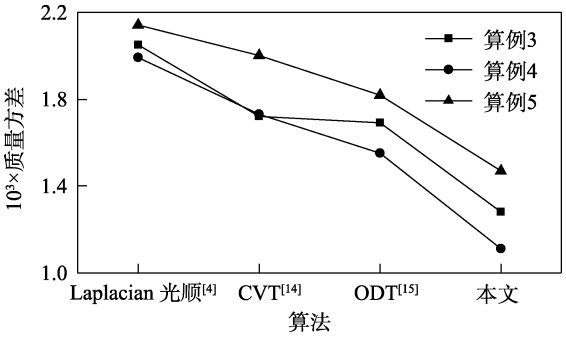



图 11 各算法对不同算例优化后的网格质量方差

图 12 给出了算例 2~算例 5 优化前后的网格质量分布, 同样展现了本文算法在畸变单元质量提升方面的优势.

综上所述, 本文算法的主要优点如下: (1) 实际测试可直接加载已被训练过的模型, 可减少大量的神经网络参数更新时间, 从而提高网格的优化效率; (2) 优先处理畸变单元, 在确保不产生无效单元的前提下, 对畸变单元的结点进行指导移动, 使得网格的最差单元质量有所提升, 平均单元质量也进一步得到改善.

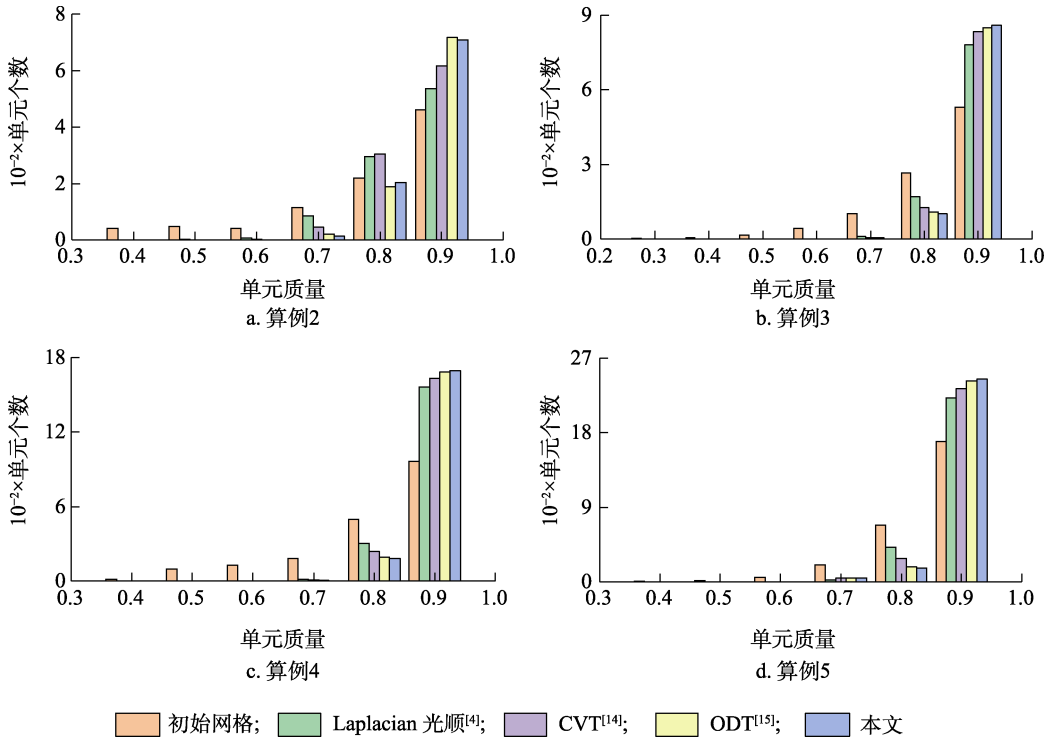



图 12　各算法对不同算例优化前后的网格质量分布

## 5　结　语

针对 Delaunay 网格生成过程中可能出现的畸变单元, 本文提出了一种利用 DQN 模型指导内部结点移动的网格优化算法. 该算法的核心要点在于 DQN 模型的设计与训练: 首先, 通过奖励函数的设计, 可保证各单元在优化过程中始终有效; 其次, 利用模型训练后的神经网络参数, 可在实际应用中加速结点移动优化.

研究结果表明, 本文算法能够有效地提升畸变单元质量, 进而改善整体网格质量, 为人工智能技术在网格优化中的应用提供一种新的思路. 本文目前仅考虑网格内部结点的移动优化, 未来会发展新型的边界顶点优化方法进行结合. 此外, 接下来会更换合适的模型要素及训练参数, 将该算法的框架扩展至二维四边形网格、三维网格的优化, 以及应用于兼顾单元形状、单元大小和插值误差等多目标的智能优化问题.

## 参考文献(References):

[1] Guan Zhenqun, Song Chao, Gu Yuanxian, *et al*. Recent advances of research on finite element mesh generation methods[J]. Journal of Computer-Aided Design & Computer Graphics, 2003, 15(1): 1-14(in Chinese)
(关振群, 宋超, 顾元宪, 等. 有限元网格生成方法研究的新进展[J]. 计算机辅助设计与图形学学报, 2003, 15(1): 1-14)

[2] Zheng Yao, Chen Jianjun. Unstructured mesh generation: theory, algorithms and applications[M]. Beijing: Science Press, 2016: 22-55(in Chinese)
(郑耀, 陈建军. 非结构网格生成: 理论、算法和应用[M]. 北京: 科学出版社, 2016: 22-55)

[3] Owen S J. A survey of unstructured mesh generation technology[OL]. [2021-07-21]. http://ima.udg.edu/~sellares/ComGeo/OwenSurv.pdf

[4] Field D A. Laplacian smoothing and Delaunay triangulations[J]. Communications in Applied Numerical Methods, 1988, 4(6): 709-712

[5] Freitag L A. On combining Laplacian and optimization-based mesh smoothing techniques[OL]. [2021-07-21]. https://digital.library.unt.edu/ark:/67531/metadc691827/m2/1/high_res_d/505716.pdf

[6] Gong Xiangfei, Zhang Shudao. One mesh smoothing algorithm combining Laplacian and local optimization-based mesh smoothing techniques[J]. Transactions of Beijing Institute of Technology, 2010, 30(5): 616-621(in Chinese)
(宫翔飞, 张树道. Laplacian 与局部优化方法相结合的网格光滑技术[J]. 北京理工大学学报, 2010, 30(5): 616-621)

[7] Lee E K, Shin M, Kim J. Improved simultaneous mesh untangling and quality improvement methods of 2D triangular meshes[J]. International Journal of Computational Methods, 2019, 16(8): Article No.1850119

[8] Kim J, Panitanarak T, Shontz S M. A multiobjective mesh optimization framework for mesh quality improvement and mesh untangling[J]. International Journal for Numerical Methods in Engineering, 2013, 94(1): 20-42

[9] Mittal K, Fischer P. Mesh smoothing for the spectral element method[J]. Journal of Scientific Computing, 2019, 78(2):

1152-1173

[10] Kim J. An efficient approach for solving mesh optimization problems using Newton's method[J]. Mathematical Problems in Engineering, 2014, 2014: Article No.273732

[11] Roda-Casanova V, Sanchez-Marin F. A simple procedure for generating locally refined 2D quadrilateral finite element meshes of gears[J]. Mechanism and Machine Theory, 2021, 157: Article No.104185

[12] Xu H T, Newman T S. An angle-based optimization approach for 2D finite element mesh smoothing[J]. Finite Elements in Analysis and Design, 2006, 42(13): 1150-1164

[13] Kim J, Shin M, Kang W. A derivative-free mesh optimization algorithm for mesh quality improvement and untangling[J]. Mathematical Problems in Engineering, 2015, 2015: Article No.264741

[14] Du Q, Gunzburger M. Grid generation and optimization based on centroidal Voronoi tessellations[J]. Applied Mathematics and Computation, 2002, 133(2/3): 591-607

[15] Alliez P, Cohen-Steiner D, Yvinec M, *et al*. Variational tetrahedral meshing[J]. ACM Transactions on Graphics, 2005, 24(3): 617-625

[16] Fabritius B, Tabor G. Improving the quality of finite volume meshes through genetic optimisation[J]. Engineering with Computers, 2016, 32(3): 425-440

[17] Yilmaz A E, Kuzuoglu M. A particle swarm optimization approach for hexahedral mesh smoothing[J]. International Journal for Numerical Methods in Fluids, 2009, 60(1): 55-78

[18] Yang F, Zhang D J, Ren H, *et al*. 2D mesh smoothing based on Markov chain method[J]. Engineering with Computers, 2020, 36(4): 1615-1626

[19] Kim J, Choi J, Kang W. A data-driven approach for simultaneous mesh untangling and smoothing using pointer networks[J]. IEEE Access, 2020, 8: 70329-70342

[20] Sutton R S, Barto A G. Reinforcement learning: an introduction[M]. Cambridge: MIT Press, 1998: 53-84

[21] Liu Quan, Zhai Jianwei, Zhang Zongzhang, *et al*. A survey on deep reinforcement learning[J]. Chinese Journal of Computers, 2018, 41(1): 1-27(in Chinese)
(刘全, 翟建伟, 章宗长, 等. 深度强化学习综述[J]. 计算机学报, 2018, 41(1): 1-27)

[22] Mnih V, Kavukcuoglu K, Silver D, *et al*. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533

[23] Watkins C J C H, Dayan P. Q-learning[J]. Machine Learning, 1992, 8(3/4): 279-292

[24] Lo S H. A new mesh generation scheme for arbitrary planar domains[J]. International Journal for Numerical Methods in Engineering, 1985, 21(8): 1403-1426

[25] Geuzaine C, Remacle J F. Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities[J]. International Journal for Numerical Methods in Engineering, 2009, 79(11): 1309-1331