

基于强化学习的改进差分演化算法求解 Thomson 问题

张婉冰¹, 戴光明^{1,2}, 彭 雷^{1,2}, 王茂才^{1,2}, 宋志明^{1,2}, 陈晓宇^{1,2}, 袁卓铭¹

(1. 中国地质大学(武汉) 计算机学院; 2. 智能地学信息处理湖北省重点实验室, 湖北 武汉 430074)

摘 要: 针对物理学家 Thomson 提出的点电荷在球面上能量最低的最佳排列问题, 提出一种基于强化学习的改进差分演化算法 RLHDE_ILS。为提高算法的搜索能力, 设计一种基于强化学习自适应控制差分演化算法缩放因子的策略, 采用强化学习的智能体为每一代选择最合适的参数值; 同时为了增强算法的局部搜索能力, 提出一种结合随机梯度下降与序列二次规划的两阶段混合局部搜索算子, 在迭代后期进行精细搜索。对 8 种不同规模的 Thomson 问题进行仿真实验, 结果表明, 与其他 15 种具有代表性的相关算法相比, RLHDE_ILS 算法在处理 Thomson 问题方面具有更快的收敛速度和更高的求解精度。

关键词: 球面点分布; Thomson 问题; 差分演化算法; 强化学习; 局部搜索

DOI: 10.11907/rjtk.241166

开放科学(资源服务)标识码(OSID):

中图分类号: TP312

文献标识码: A

文章编号: 1672-7800(2025)005-0087-10



Reinforcement Learning-Based Improved Differential Evolutionary Algorithm for Solving Thomson Problems

ZHANG Wanbing¹, DAI Guangming^{1,2}, PENG Lei^{1,2}, WANG Maocai^{1,2}, SONG Zhiming^{1,2}, CHEN Xiaoyu^{1,2}, YUAN Zhuoming¹

(1. School of Computer Science, China University of Geosciences(Wuhan);

2. Hubei Key Laboratory of Intelligent Geo-Information Processing, Wuhan 430074, China)

Abstract: Aiming at the optimal arrangement problem of point charges with the lowest energy on a sphere proposed by physicist Thomson, an improved differential evolution algorithm based on reinforcement learning, RLHDE_ILS, is proposed. To improve the search capability of the algorithm, a strategy based on reinforcement learning adaptive control differential evolution algorithm scaling factor is designed, using reinforcement learning agents to select the most suitable parameter values for each generation; At the same time, in order to enhance the local search capability of the algorithm, a two-stage hybrid local search operator combining stochastic gradient descent and sequential quadratic programming is proposed for fine search in the later stages of iteration. Simulation experiments were conducted on 8 different scales of Thomson problems, and the results showed that compared with 15 other representative related algorithms, the RLHDE_ILS algorithm has faster convergence speed and higher solution accuracy in dealing with Thomson problems.

Key Words: spherical point distribution; Thomson problem; differential evolution algorithm; reinforcement learning; local search

0 引言

三维球体表面上点的配置有很多种方式, 其中一种求

解配置方式是将球面上的点视作点电荷, 优化求解点电荷之间的总势能, 又被称为 Thomson 问题。该问题在优化领域具有重要作用, 具有高度非线性、非凸以及搜索空间的规模呈指数级增长等特征, 是 21 世纪的数学难题之一, 也

收稿日期: 2024-04-15

扫描二维码阅读全文:



基金项目: 国家自然科学基金项目(42271391, 62006214); 装备预研教育部联合基金项目(8091B022148); 湖北省重点研发专项(2023BIB015); 湖北省优秀中青年科技创新团队计划项目(T2021031)

作者简介: 张婉冰(2000-), 女, CCF 会员, 中国地质大学(武汉)计算机学院硕士研究生, 研究方向为智能优化算法; 戴光明(1964-), 男, 博士, 中国地质大学(武汉)计算机学院教授、博士生导师, 研究方向为空间信息网络、智能优化算法。本文通讯作者: 戴光明。

是寻求最低势能分布的一大挑战^[1]。除物理意义外, Thomson 问题在其他领域也具有研究价值。在生物学中, 该问题可帮助求解构成球形病毒(例如腺病毒)外壳的蛋白质亚基排列。在结构化学中, 该问题可帮助寻找蛋白质 S 层(例如古细菌蛋白质)的排列规则。该问题还可应用于铱星 LEO 卫星星座布局、移动电话的小区接入点分布以及无线电的中继位置布局等问题的求解^[2]。

在优化领域, Thomson 问题是无约束非线性连续优化问题, 其难点在于即使是点数规模较小的情况下也需要寻找一种合适的方法来降低计算难度, 并且随着问题规模的增加, 局部极小值也随之呈指数形式增加, 使得求解变得更加复杂。这种复杂性的增加促使研究者们开发更为高效的优化算法, 包括粒子群算法、模拟退火算法、生物共生搜索算法以及遗传算法等, 以期在多局部极值盛行的高维搜索空间中精确定位到全局最优解^[2-5]。例如, Lakhbab 等^[6-7]提出两种方法求解 Thomson 问题: 第一种是基于综合粒子群算法(Comprehensive learning Particle swarm optimization, CLPSO)的混合算法 CLPSO_NSG, 首先由 CLPSO 进行全局搜索得到最优解, 然后使用非单调光谱梯度(Nonmonotone Spectral Gradient, NSG)方法进行精细搜索; 第二种是基于多种群进化算法的混合算法 EA_SG2, 种群被划分为 3 个部分, 对其进行不同的处理, 然后使用 NSG 方法进行精细搜索。Amal 等^[8]使用量子遗传算法求解 Thomson 问题, 将球体表面上点电荷的坐标映射到量子位, 将问题的解决方案编码为构成量子染色体的量子比特序列。杨承中等^[9]设计了一种新型群体分裂演化算法 PDEA(Population Decomposition Evolutionary Algorithm)来求解球面上的五点分布问题。差分进化算法(Differential Evolution Algorithm, DE)由 Price 等^[10]提出, 具有结构简单、容易实现、收敛快速和鲁棒性强等特点, 但参数调整对该算法性能的影响较大, 选择一种合适的参数调整策略对于问题求解至关重要。研究人员提出许多自适应机制使该算法能够在搜索过程中动态调整控制参数。例如, Tanabe 等^[11]提出基于成功历史参数的自适应 DE 算法 SHADE。为进一步提高 SHADE 算法的性能, Tanabe 等^[12]引入种群大小线性减少(Linear Population Size Reduction, LPSR)机制形成 L-SHADE 算法, 其中种群大小按照线性函数减少。Piotrowski 等^[13]在 L-SHADE 的变异策略中加入群体惯性项(Population-wide Inertia term, PWI)以提高上一代解的质量。Li 等^[14]对 L-SHADE 进行两部分改进, 首先引入一种新的变异策略 current-to-Amean/1, 其次引入一种带有变异方法的自适应选择策略。

强化学习通过智能体与环境交互的方式不断学习, 使预期回报最大化。由于具有平衡探索与利用之间关系的能力, 强化学习策略可应用于参数的自适应调整从而优

化算法性能。例如, Hu 等^[15]提出一种使用强化学习以增量形式自适应控制 DE 缩放因子 F 值的算法 RLDE; Huynh 等^[16]提出一种使用强化学习自适应选择 DE 缩放因子 F 和交叉率 CR 参数对的算法 qlDE; Zhao 等^[17]提出一种使用强化学习自适应选择变异策略的算法 RLBSO。

目前基于 DE 算法求解 Thomson 问题的研究相对有限。为此, 本文提出一种基于强化学习并结合改进局部搜索策略的混合 DE 算法 RLHDE_ILS(An Improved Reinforcement Learning-Based Hybrid Differential Evolutionary Algorithm Combined with Improved Local Search)。主要研究内容及创新点为: ①针对 DE 算法中参数 F 取值影响算法搜索性能的特点, 提出一种基于强化学习自适应控制 F 参数池的方法, 通过构建一个包含多个离散值的参数池, 允许强化学习选择最适合的参数。该改进策略提高了算法的收敛速度, 并有效平衡了算法的开发与勘探能力; ②针对 DE 算法在演化后期局部搜索能力不足的问题, 提出将 DE 算法与局部搜索算法随机梯度下降法(Stochastic Gradient Decent, SGD)和序列二次规划(Sequential Quadratic Programming, SQP)相结合, 第一阶段采用 SGD 算子对当前最优个体进行快速梯度导向搜索, 第二阶段采用 SQP 算子进行更加细致和精确的局部搜索, 以寻找可能的改进方向, 得到的更优个体参与下一轮迭代并指导搜索。该混合策略提高了算法的收敛精度和收敛速度。

1 球面点分布问题

Thomson 问题旨在确定单位球面上的 N 个点电荷以何种排布可实现势能最低化, 是静电学领域的一个经典问题。该问题的求解方式为通过算法迭代优化以确保电荷间最大可能的相互距离, 从而达到能量最小状态。

1.1 Thomson 问题数学建模

假设有一个 $N \geq 2$ 的三维球面 S^2 , $S^2 = \{X: x^2 + y^2 + z^2 = 1\}$, 其中 x, y, z 为点的三维坐标; X_i 和 X_j 分别表示两个点 i 和 j , $\|X_i - X_j\|$ 表示球面上两点之间的距离。单位球体表面上分布 N 个粒子电荷所需的总能量为:

$$E(X_1, X_2, \dots, X_N) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \frac{1}{\|X_i - X_j\|} \quad (1)$$

为方便求解问题, 将直角坐标系转换为球面点坐标系下的 θ_i 和 φ_i 。则:

$$\begin{aligned} \|X_i - X_j\| &= d_j(\theta_i, \varphi_i, \theta_j, \varphi_j) \\ d_j(\theta_i, \varphi_i, \theta_j, \varphi_j) &= \sqrt{2(1 - \sin \varphi_i \sin \varphi_j \cos(\theta_i - \theta_j) - \cos \varphi_i \cos \varphi_j)} \end{aligned} \quad (2)$$

1.2 Thomson 问题优化模型

以 Thomson 问题作为最小化的优化函数, 则 Thomson 问题优化模型的定义为:

$$\begin{aligned} & \text{Minimize } E(u) \\ & u = (\theta_1, \varphi_1, \theta_2, \varphi_2, \dots, \theta_n, \varphi_n) \end{aligned} \quad (3)$$

式中: $E(u)$ 为 Thomson 问题中电荷的总能量, 可采用式(1)计算; 决策变量的上下界为 $\theta_i \in [0, 2\pi]$, $\varphi_i \in [0, \pi]$ 。

2 算法理论

2.1 DE 算法

DE 算法是一种简单有效的用于连续函数优化的智能算法, 主要包括变异、交叉以及选择算子 3 个步骤^[18]。初始种群由 NP 个个体组成, 每个个体是一个 D 维向量: $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$, $i = 1, \dots, NP$ 。大部分算法选择随机初始种群。

2.1.1 变异操作

变异算子生成变异向量。DE 算法具有多种变异策略, 本文选择的变异策略为“DE/best/1”。表示为:

$$V_i = X_{best} + F * (X_{r2} - X_{r3}) \quad (5)$$

式中: 索引 $r_1 \neq r_2 \neq r_3 \neq i$, 这些索引是不同的随机数; F 为 $[0, 1]$ 范围内的正数; X_{best} 为当前种群中适应度最好的个体。

2.1.2 交叉操作

试验向量 $U_i = (u_{i,1}, u_{i,2}, \dots, u_{i,D})$ 主要由二项式交叉算子生成。生成公式为:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } (rand_{ij}[0, 1] \leq Cr \text{ or } j = j_{rand}) \\ x_{i,j}, & \text{otherwise} \end{cases} \quad (6)$$

式中: $rand_{i,j} \in [0, 1]$ 表示范围为 0~1 之间均匀分布的随机数; $Cr \in [0, 1]$ 是交叉概率; j_{rand} 为小于 D 的整数。

2.1.3 选择操作

将新生成的试验向量与原始目标向量的适应度值进行比较, 选择最佳向量作为下一代种群。选择过程表示为:

$$X_i = \begin{cases} U_i, & \text{if } f(U_i) \leq f(X_i) \\ X_i, & \text{otherwise} \end{cases} \quad (7)$$

2.2 强化学习

强化学习模型由智能体、环境、动作、状态和奖励 5 个主要部分组成, 其运行过程可描述为状态、动作和奖励的循环。具体结构如图 1 所示。在每个时刻 t , 智能体从环境中感知到状态 s_t , 将执行相应的动作 a_t , 执行动作后智能体会得到奖励 r_{t+1} 并转移到下一个状态 s_{t+1} 。根据强化信号和当前环境状态, 智能体按照一定策略选择下一步行动, 并一直执行上述循环过程直至达到函数评估次数。

Q-learning 是一个典型的基于值函数的强化学习方法, 其中 Q 表示质量, 智能体通过每次选择动作的质量来获得未来奖励^[19]。假设 $S = \{s_1, s_2, \dots, s_m\}$ 为环境状态的集合, $A = \{a_1, a_2, \dots, a_m\}$ 为智能体可以执行的动作集。在每次迭代中赋予智能体一个状态并选择一个要执行的动作,

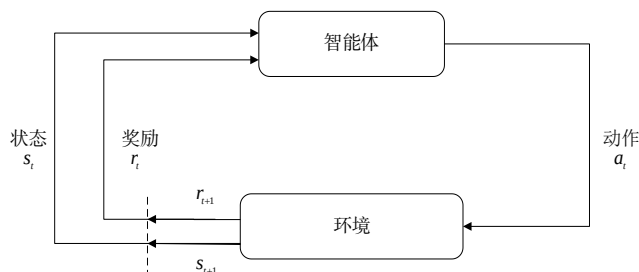


Fig. 1 Reinforcement learning model

图 1 强化学习模型

智能体可以根据累积的 Q 表来执行最佳动作。在动作执行之后, 环境给出奖励 r_{t+1} 和新状态 s_{t+1} , 智能体获得这些信息后形成对每个给定动作期望值 $Q(s_t, a_t)$ 的评估。Q-learning 使用期望方程更新价值函数, 其更新方程为贝尔曼方程。表示为:

$$\begin{aligned} Q_{t+1}(s_t, a_t) &= Q(s_t, a_t) \\ &+ \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \end{aligned} \quad (8)$$

式中: s_t 为智能体的状态; a_t 为智能体执行的动作; r 为智能体执行动作获得的即时奖励; γ 为区间内的折扣因子, 负责惩罚未来报酬。当 $\gamma = 0$ 时, Q-learning 只考虑当前奖励, 当 $\gamma = 1$ 时, Q-learning 寻找长期奖励; α 为区间 $[0, 1]$ 内的学习率; $Q_{t+1}(s_t, a_t)$ 为学习代理获得的总累积奖励。

表 1 为 Q 表的形式, 每个状态—动作对表示通过在状态中执行动作将获得的未来奖励。智能体通过与环境的交互获得有价值的反馈, 选择能够获得最大累积奖励的策略。

Table 1 Q table of corresponding state and action

表 1 状态与动作对应的 Q 表

状态	动作			
	a_1	a_2	$\dots$	a_n
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$\dots$	$Q(s_1, a_n)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$\dots$	$Q(s_2, a_n)$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
s_m	$Q(s_m, a_1)$	$Q(s_m, a_2)$	$\dots$	$Q(s_m, a_n)$

3 RLHDE_ILS 算法

RLHDE_ILS 算法的核心为自适应控制策略和混合策略, 其中自适应策略为 Q-learning 自适应控制缩放因子 F 参数池, 混合策略为混合局部搜索算子以及将其与 DE 混合。

3.1 基于 Q-learning 的参数池设置

较大的 F 值具有更广泛的搜索空间, 可提高算法的多样性, 有利于种群搜索全局最优解, 但会降低收敛效率; 较小的 F 值能增强算法的局部搜索能力, 提高算法的收敛速

度,但同时增加了陷入早熟收敛的风险。Q-learning的作用为在每次迭代开始时从参数池中选择最佳 F 值,范围为 $[0.1, 0.2, 0.3, \dots, 1]$,该参数池以0.1作为间隔在 $[0, 1]$ 的区间内均匀分布。这种参数设置策略旨在实现 F 值的灵活调整,最大化算法的搜索性能。智能体选择出最佳 F 值后,根据SoftMax策略计算智能体在状态 s_i 下选择动作 a_j 的概率。公式为:

$$\pi(s_i, a_j) = \frac{e^{Q_t(s_i, a_j)/T}}{\sum_{k=1}^n e^{Q_t(s_i, a_k)/T}} \quad (9)$$

式中: $Q_t(s_i, a_j)$ 为Q表中时间 t 处的对应值; T 为控制参数。

根据动作调整值评估目标函数的适应度值,根据适应度值的变化情况决定奖励。奖励可表示:

$$\text{reward}(i) = \begin{cases} 1, & \text{if } \text{fit_new}(i) < \text{fit_old}(i) \\ -1, & \text{otherwise} \end{cases} \quad (10)$$

式中: $\text{fit_old}(i)$ 和 $\text{fit_new}(i)$ 分别为第 i 个个体的原始适应度值与新适应度值。

Q-learning随着任务的进行同时学习,可以在DE算法进化过程中自动调整参数,平衡算法的探索与利用。设 fes 表示当前评价次数, Max_FES 表示最大迭代次数,在整个算法运行周期内,Q-learning算法的伪代码为:

算法1 Q-learning算法

输入:控制参数 $s_i, a_i, \gamma, \alpha, r$ 。
输出: s 。
1. 初始化Q表;
2. 随机选择一个初始状态;
3. while $fes \leq Max_FES$ do
4. 从Q表中为当前状态 s_i 选择最好的动作 a_i ;
5. 执行动作并获得即时奖励 r ;
6. 获得下一状态 s_{i+1} 的最大Q值;
7. 使用公式(8)更新Q表;
8. 更新当前状态 $s_i = s_{i+1}$;
9. end while

3.2 改进混合局部搜索策略

DE算法在演化后期存在收敛速度较慢、局部开发能力相对不足的问题,因此本文在算法后期设计一种局部搜索算子。改进的局部算子使用两阶段混合局部搜索策略,第一阶段使用SGD算法,第二阶段采用SQP算法。SGD有利于指导全局最佳个体位置的搜索方向,在算法陷入局部最优时可以将全局最佳个体提升至更有希望的区域,从而提高算法精度^[20]。如果函数 $F(x)$ 是连续的并且在点 X 处的邻域可微分,那么可以通过SGD来优化 $F(x)$ 。在强化学习训练一定的迭代次数后,将执行SGD算法来更新全局最优解,这样选择出来的最佳个体拥有更好的指导搜索的能力。更新公式为:

$$x_{new} = x_{old} - \delta \times \text{gradient} \quad (11)$$

式中: x_{new} 为更新后的最好个体, x_{old} 为更新前的最好

个体; δ 为学习率,其取值影响搜索效率和性能,本文实验中 δ 取0.01; gradient 表示目标函数在变量 x_{new} 处的梯度 $\nabla F(x_{new})$ 。

SQP算法在每一次迭代中使用准牛顿更新方法对Hessian矩阵进行近似处理,然后将其用于生成二次规划(Quadratic Programming, QP)子问题,这些子问题的解将用于形成搜索方向^[21]。SGD与SQP结合成为一种混合局部搜索算子,使用的变异策略为DE/best/1,该策略利用全局最优个体来引导搜索过程。因此,在算法运行的最后阶段(即进行过 $85\% \times Max_FES$ 次评估后),首先使用SGD算法对全局最优个体进行梯度下降,一旦找到比当前最优个体更好的解则退出循环,并替换之前的全局最优解,此时需给SGD算法设置一个最大迭代次数 max_iter 。SQP算法在SGD算法搜索结果的基础上继续进行局部搜索,起到精细搜索和补充搜索的作用。如果SQP搜索到的结果优于SGD,则SQP得到的最优解将替换之前的全局最优解,反之则以SGD找到的结果替换之前的全局最优解,新的全局最优解能更好地指引搜索方向。SQP评估次数比率为 P_{ls} ,若SQP算法找到更好的解,则 P_{ls} 设置为0.01,否则设置为0.001, SQP算法的评估次数 $C_{FE}_{ls} = Max_FES \times P_{ls}$ 。改进混合局部搜索算子伪代码为:

算法2 改进混合局部搜索策略

输入:种群中的最好个体 x_{best} 及其适应度值 fit_{best} ,控制参数 δ 。
输出: $x_{best}, \text{fit}_{best}$ 。
1. $x_{new} = x_{best}$;
2. 计算 x_{new} 的梯度 gradient ;
3. 使用公式(11)得到新的 x_{new} ;
4. 使用公式(1)计算 x_{new} 的适应度值 fit_{new} ;
5. 如果 $\text{fit}_{new} < \text{fit}_{best}$ 那么 $x_{best} = x_{new}, \text{fit}_{best} = \text{fit}_{new}$,并终止计算;
6. 对最好的个体 x_{best} 使用SQP评估 C_{FE}_{ls} 次;
7. 如果 $f(x_{SQP}) < f(x_{best})$ 那么 $P_{ls} = 0.01, x_{best} = x_{SQP}, \text{fit}_{best} = f(x_{best})$,否则 $P_{ls} = 0.001$;
8. 更新评估次数。

3.3 LPSR

LPSR采用缩减种群大小的方法,用于在DE运行期间动态调整种群大小,可有效提高算法性能^[22]。LPSR根据线性函数不断减少种群数量,其中第一代种群大小为 NP^{init} ,运行结束时的种群为 NP^{min} 。在每一代运行完成后,下一代种群规模 NP_{G+1} 的计算公式为:

$$NP_{G+1} = \text{round}[(\frac{NP^{min} - NP^{init}}{MAX_FES}) \cdot FES + NP^{init}] \quad (12)$$

式中: G 为当前迭代代数, FES 为当前适应度评价次数; MAX_FES 为最大适应度评价次数。当 $NP_{G+1} < NP_G$ 时, $(NP_G - NP_{G+1})$ 个排名最差的个体将从种群中删除。

3.4 RLHDE_ILS算法伪代码

经典DE算法的框架保持不变,在每次DE迭代开始时选择合适的缩放因子,然后执行交叉变异操作,迭代一定次数后对当前最优解进行局部搜索,以更好地收敛到全局

最优解。QDE_HLS算法伪代码为:

算法3 RLHDE_ILS算法

输入:种群数量 NP , 最大评估次数 Max_FES 。

输出:最优解,最优值。

1. 随机初始化种群,初始化动作 $action$, 状态 $state$ 等参数;
2. 从参数池里面选择 F 的值;
3. 交叉、变异及选择操作生成子代个体;
4. 根据等式(9)计算每个个体的动作选择概率;
5. 通过选择动作来选择 F 的值;
6. 更新每个智能体的动作;
7. 评估每个子代个体的适应度值;
8. 更新评估次数;
9. 如果 $f(x_{i,new}) < f(x_i)$ 那么 $x_i = x_{i,new}$, $f(x_i) = f(x_{i,new})$,
 $reward = 1$, $state = 1$, 否则 $reward = -1$, $state = 2$;
10. 如果 $FES \geq 0.85 \times Max_FES$, 则调用算法2, 更新迭代次数 FES ;
11. 如果不满足终止条件, 则转到步骤2。

4 实验方法与结果分析

为验证所提算法 RLHDE_ILS 的有效性, 设计2组比较实验, 一组为与10种DE算法的比较, 另一组为与求解 Thomson 问题的最新算法的比较。

4.1 实验设置

实验分别在规模为 $N=5, 15, 30, 50, 65, 100, 150, 200$ 的 Thomson 问题上进行, 对应的维度为 $D = 2 \times N$, 种群大小均设置为 $NP = 18 \times D$ 。随着规模变大, 求解难度增加, 在高维空间中的优化求解更加困难。分别将最大评估次数设置为 5 000、8 000、30 000、80 000、90 000、150 000、250 000 以及 450 000。RLHDE_ILS 算法中交叉概率 Cr 的取值范围一般为 0~1, 本文取 $Cr = 0.9$, 原因是较高的 Cr 值能够保证算法的多样性。最小种群 $NP^{min} = 4$ 。Q-learning 算法在上述评估次数中对 α 和 γ 的取值并不敏感, 因此遵循经验设置 $\alpha = 0.1$ 、 $\gamma = 0.9$ 。

为验证 RLHDE_ILS 算法的性能, 将其与 10 种 DE 系列最先进的算法进行比较, 包括 LSHADE^[12]、LSHADE-PW^[13]、ALSHADE^[14]、RLDE^[15]、qlDE^[16]、LSHADE-cnEp-Sin^[23]、IMODE^[24]、PaDE^[25]、IDE-EDA^[26] 以及 DISH^[27]。这些算法不仅具有很好的优化性能, 而且还是具有多种控制策略的最新 DE 变体算法。同时将其与求解 Thomson 问题的其他类最新算法进行比较, 包括 SOS^[2]、PSO^[3]、EA_SG2^[6]、ga_fminunc^[6]、CLPSO_SG^[7]。

为保证公平性, 所有算法均设置了相同的评估次数。为减少算法的随机性, 每个算法独立运行 30 次。所有比较实验均在配备 2.90 GHz Intel (R) Xeon (R) Platinum 8375C CPU 和 512 GB RAM 的 64 位 Windows Server 2019 中的 Matlab 2021a 软件上进行。

4.2 算法参数分析

算法参数取值会给实验结果造成一些影响, 而 SGD 中的参数 δ 对于算法的寻优能力具有较强影响。为确定最优 δ 值, 采用不同取值的 δ 进行比较实验, 通过 Friedman 分析对实验结果进行客观评价。表 2 为 Friedman 分析验证结果, 可以看出, 当 $\delta = 0.010$ 时, 算法能获得最好的 Friedman 排名。

Table 2 Friedman analysis results for different values of δ
表2 不同取值 δ 的 Friedman 分析结果

参数取值	Friedman Rank
$\delta = 0.005$	2.750 0
$\delta = 0.010$	2.333 3
$\delta = 0.030$	3.083 3
$\delta = 0.050$	3.250 0
$\delta = 0.100$	3.583 3

4.3 算法有效性验证

在不同规模 Thomson 问题上进行算法的比较分析, 使用两种非参数统计假设检验对实验结果进行统计分析, 分别为 MATLAB 中的 Wilcoxon 秩和检验分析方法和 Friedman 检验分析方法, 检验结果能体现出 RLHDE_ILS 算法与其他算法在求解各问题的性能方面是否存在显著差异。Wilcoxon 秩和检验方法的置信水平设置为 0.05, 通过 MATLAB 中的相关函数进行计算, 评估两组算法在每个问题上独立运行 30 轮的实验结果。Friedman 检验根据所有算法在所有测试问题上的表现进行打分, Friedman 测试分数越低表明算法性能越好。Friedman 检验通过 KEEL 软件进行^[28]。

表 3 为 RLHDE_ILS 算法与求解 Thomson 问题的其他算法在不同规模 Thomson 问题下的实验结果, 每个问题规模下记录各算法取得的最优值 (best)、均值 (mean) 以及方差 (std)。表 3 同时还记录了 Wilcoxon 秩和检验结果, 其中符号 “+” “-” 和 “=” 分别表示 RLHDE_ILS 与该算法比较更好、更差和相当。可以看出, 在每个问题上, RLHDE_ILS 算法得到的结果均优于 EA_SG2、CLPSO_SG 以及 SOS 算法。RLHDE_ILS 在 7 个问题上的表现优于 PSO, 在 1 个问题上与其相当。RLHDE_ILS 在 6 个问题上的表现优于 ga_fminunc, 在 2 个问题上与其相当。在求解规模 $N=5$ 的 Thomson 问题时, RLHDE_ILS 算法与 PSO、ga_fminunc 算法均能收敛到 6.47 的结果, 且方差和均值一致, 表明 RLHDE_ILS 在该规模上与 PSO 和 ga_fminunc 性能相当, 而其余算法的均值明显差于 RLHDE_ILS。当问题规模 $N=15$ 时, RLHDE_ILS 与 ga_fminunc 均能收敛到 80.67, 收敛精度与稳定性均优于其他算法。从实验数据上看, 若算法在规模为 30 的情况下超越其他算法, 那么在规模更大的问题下也会超越其他算法。

Table 3 Experimental results of QDE_HLS algorithm and other algorithms for solving Thomson problems under different scales of Thomson problems**表 3** QDE_HLS 算法与求解 Thomson 问题的其他算法在不同规模 Thomson 问题下的实验结果

规模	指标	EA_SG2	CLPSO_SG	SOS	PSO	ga_fminunc	RLHDE_ILS
N=5	Best	6.48	6.49	6.51	6.47	6.47	6.47
	Mean	6.51	6.51	6.58	6.47	6.47	6.47
	Std	0.02	0.01	0.04	0.00	0.00	0.00
	Time/s	0.17	0.25	0.53	0.02	0.07	0.2
N=15	Best	81.36	81.62	84.30	81.03	80.67	80.67
	Mean	82.00	81.87	85.68	81.41	80.67	80.67
	Std	0.59	0.17	0.69	0.22	0.00	0.00
	Time/s	0.73	1.04	2.74	0.08	0.36	0.22
N=30	Best	360.83	361.28	387.31	360.78	359.60	359.60
	Mean	363.00	362.45	394.77	361.48	359.66	359.62
	Std	1.36	0.96	3.39	0.45	0.06	0.02
	Time/s	3.87	5.45	10.84	0.84	1.10	3.07
N=50	Best	1 057.13	1 057.41	1 131.57	1 057.84	1 055.55	1 055.18
	Mean	1 058.52	1 059.19	1 170.88	1 059.29	1 056.13	1 055.27
	Std	0.79	1.04	13.50	1.22	0.43	0.07
	Time/s	14.27	19.48	47.63	5.54	5.17	20.11
N=65	Best	1 826.20	1 826.57	1 936.99	1 830.16	1 824.62	1 823.68
	Mean	1 827.59	1 827.35	1 966.80	1 835.44	1 825.36	1 823.75
	Std	0.98	0.55	11.40	2.68	0.83	0.06
	Time/s	26.29	35.07	76.59	10.31	8.53	42.65
N=100	Best	4 452.83	4 453.44	4 916.60	4 475.47	4 453.63	4 448.53
	Mean	4 456.45	4 456.45	5 032.22	4 489.99	4 462.58	4 448.88
	Std	2.16	1.97	39.30	9.25	13.40	0.20
	Time/s	71.23	158.70	196.08	37.93	21.82	115.57
N=150	Best	10 246.64	10 247.08	11 260.59	10 327.10	10 254.36	10 236.89
	Mean	10 250.69	10 250.93	11 406.54	10 353.23	10 309.25	10 237.73
	Std	2.39	1.76	62.22	17.83	67.50	0.86
	Time/s	480.49	365.47	658.87	140.88	75.68	421.06
N=200	Best	18 456.55	18 456.68	20 233.22	18 576.59	18 465.26	18 439.72
	Mean	18 461.29	18 461.77	20 510.51	18 645.31	18 555.70	18 441.03
	Std	3.82	3.08	130.00	41.40	132.08	1.28
	Time/s	2 129.49	2 370.14	2 009.26	443.82	226.15	1 014.75
RLHDE_ILSvs.	+/-/=	8/0/0	8/0/0	8/0/0	7/0/1	6/0/2	/

表 4 为 RLHDE_ILS 算法与其他 DE 系列算法求解不同规模 Thomson 问题的实验结果。可以看出,RLHDE_ILS 在每个问题上的表现均优于除 RLDE 之外的其他算法,而 RLHDE_ILS 也只在处理低维问题(规模为 5)时与 RLDE 相当,在其他规模问题上的表现均优于 RLDE。

随着问题规模的增加,搜索空间变得更为复杂,问题优化难度加大,但是 RLHDE_ILS 算法仍然能很好地解决较大规模的 Thomson 问题,收敛精度、平均表现以及稳定性均明显优于对照算法。由于该算法存在两阶段局部算子,运行时间比其他 DE 变体算法更长,但比 EA_SG2 和 CLPSO_SG 算法短,因为这两个算法内部也存在局部算子。总体来说,RLHDE_ILS 算法在求解 Thomson 问题上的表现最好,在问题规模较大的情况下优势更加明显,其次是 RLDE 算法和 ga_fminunc 算法。同样使用强化学习控制参数的 RLDE 和 qlDE 算法的收敛精度和性能均不如 RLHDE_ILS,这凸显了本研究设计的特定强化学习控制策略和参数池在求解 Thomson 问题上的有效性。

如表 5、表 6 所示,RLHDE_ILS 在两类算法中的 Friedman 检验排名均最高,分别取得 1.06 和 1.19 的分数,这与 Wilcoxon 检验分析得出的结论一致。

4.4 算法收敛性验证

为展现各算法求解 Thomson 问题的收敛速度,在演化过程中设置 14 个记录点,每轮实验均记录在 14 个不同阶段下的最佳目标函数值,最终取 30 轮中 14 个收敛点的均值作为收敛曲线的连接点。图 2 为求解 Thomson 问题算法的收敛曲线,其中 y 轴表示目标函数值,x 轴表示函数评估次数,曲线表示目标函数随着函数评估次数的变化情况。可以看出,RLHDE_ILS 算法与 ga_fminunc 算法在除规模 N=100、200 之外的各问题上的最终收敛结果一致,但 RLHDE_ILS 的收敛速度优于 ga_fminunc。RLHDE_ILS 算法在问题规模 N=30、65、100、150、200 时的最终收敛结果与 EA_SG2 一致,但收敛速度优于 EA_SG2。EA_SG2 与 ga_fminunc 在最后一个收敛点处出现快速收敛是由于其是在演化算法趋于停滞后再进行局部搜索,而 RL-

Table 4 Experimental results of QDE_HLS algorithm and other DE series algorithms for solving Thomson problems of different scales

表 4 RLHDE_ILS 算法与其他 DE 系列算法求解不同规模 Thomson 问题的实验结果

规模	指标	LSHADE	LSHADE_c		LSHADE_P	IMODE	PaDE	ALSHADE	IDE-EDA	DISH	RLDE	qlDE	RLHDE_ILS
			nEpSin	WI									
N=5	Best	6.48	6.50	6.48	6.48	6.48	6.48	6.48	6.48	6.48	6.47	6.49	6.47
	Mean	6.49	6.52	6.48	6.48	6.48	6.48	6.50	6.51	6.54	6.47	6.53	6.47
	Std	0.01	0.02	0.00	0.00	0.00	0.00	0.01	0.01	0.02	0.00	0.02	0.00
	Time/s	0.102 8	0.148 4	0.097 2	0.118 0	0.120 2	0.087 4	0.181 0	0.048 0	0.136 6	0.192 4	0.204 8	
N=15	Best	82.53	82.53	82.06	81.61	82.06	82.08	82.88	82.99	82.99	80.67	82.99	80.67
	Mean	83.75	84.20	82.79	81.86	82.72	83.10	84.00	83.89	81.29	84.03	80.67	
	Std	0.58	0.67	0.43	0.17	0.48	0.61	0.43	0.50	0.92	0.46	0.00	
	Time/s	0.184 4	0.210 2	0.161 4	0.200 6	0.221 0	0.184 0	0.205 8	0.163 4	0.335 2	0.868 0	0.217 2	
N=30	Best	375.78	379.68	366.41	364.19	367.67	367.65	372.75	371.76	359.83	381.26	359.60	
	Mean	380.77	384.44	370.24	365.47	372.37	376.21	380.56	379.60	361.16	386.46	359.62	
	Std	1.89	2.47	2.54	0.57	3.87	2.85	3.72	3.44	1.35	2.63	0.02	
	Time/s	1.137 6	1.259 8	1.154 4	1.226 0	1.143 4	1.094 6	1.172 0	1.060 8	2.329 6	1.938 6	3.067 0	
N=50	Best	1 103.72	1 124.43	1 072.46	1 067.02	1 073.75	1 083.46	1 099.49	1 105.41	1 055.61	1 128.40	1 055.18	
	Mean	1 115.69	1 138.10	1 079.63	1 069.84	1 078.17	1 096.27	1 116.63	1 115.03	1 058.77	1 142.27	1 055.27	
	Std	5.52	7.44	6.83	1.19	2.83	10.80	7.55	5.11	2.44	6.20	0.07	
	Time/s	6.161 4	6.097 0	5.319 6	5.671 2	5.366 0	5.141 6	5.231 8	5.051 2	10.384 4	7.852 8	20.114 8	
N=65	Best	1 894.20	1 908.14	1 851.47	1 851.69	1 853.85	1 859.37	1 895.30	1 897.43	1 824.82	1 921.32	1 823.68	
	Mean	1 905.58	1 933.75	1 856.19	1 854.94	1 859.18	1 876.06	1 914.03	1 910.54	1 830.19	1 939.29	1 823.75	
	Std	6.27	8.96	3.89	1.78	2.25	13.90	9.02	6.64	5.31	8.33	0.06	
	Time/s	10.463 6	10.505 2	8.218 8	9.741 2	8.040 4	7.990 0	8.253 4	8.085 0	18.140 6	12.107 6	42.649 0	
N=100	Best	4 729.06	4 748.11	4 573.27	4 518.29	4 564.04	4 595.15	4 730.22	4 734.85	4 480.83	4 780.53	4 448.53	
	Mean	4 776.47	4 845.17	4 583.78	4 532.57	4 591.57	4 677.69	4 769.53	4 772.21	4 519.55	4 881.56	4 448.88	
	Std	23.80	31.50	7.70	5.49	9.22	58.70	23.00	21.60	30.00	32.18	0.20	
	Time/s	29.276 2	29.482 4	21.393 8	28.476 0	21.542 4	21.165 8	21.706 6	21.380 4	54.353 4	29.124 2	115.569 2	
N=150	Best	10 912.22	11 091.68	10 557.63	10 419.16	10 570.08	10 648.58	10 906.78	10 911.22	10 320.44	11 169.07	10 236.89	
	Mean	11 020.36	11 199.63	10 591.40	10 448.49	10 612.79	10 769.28	11 003.21	11 013.36	10 440.15	11 276.69	10 237.73	
	Std	50.44	47.32	21.22	11.97	23.98	99.39	42.84	43.41	50.33	45.68	0.86	
	Time/s	97.162 2	97.983 4	69.753 6	93.799 2	71.044 0	69.624 2	73.455 6	72.171 6	201.370 6	91.694 0	421.062 2	
N=200	Best	19 675.15	20 098.31	18 943.04	18 747.82	18 954.49	19 100.48	19 671.06	19 626.53	18 538.09	20 205.03	18 439.96	
	Mean	19 769.71	20 192.80	18 995.93	18 776.54	19 043.86	19 301.92	19 847.83	19 861.33	18 875.17	20 385.76	18 441.77	
	Std	58.80	58.53	28.88	17.25	30.63	188.98	75.52	75.38	144.41	81.52	2.25	
	Time/s	270.534 8	271.175 6	204.676 6	265.711 0	208.967 6	202.863 2	213.250 4	207.545 8	543.131 0	263.107 2	1 014.750 8	
RLHDE_ILSvs.	+/-/=	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	8/0/0	7/0/1	8/0/0	

Table 5 Friedman results of DE variant algorithm

表 5 DE 变体算法的 Friedman 检验结果

算法	Ranking
LSHADE	7.75
LSHADE_cnEpSin	10.12
LSHADE-PWI	4.25
IMODE	3.00
PaDE	4.62
ALSHADE	6.12
IDE-EDA	8.00
DISH	7.12
RLDE	2.06
qlDE	10.88
RLHDE-ILS	1.06

HDE_ILS 算法能够协调全局搜索与局部搜索之间的关系,在各个环节上均能取得很好表现。总而言之,RLHDE_ILS 算法的改进混合局部搜索策略能与全局搜索相辅相成,在规模较大的 Thomson 问题求解上具有很好效果并且收敛快速。

Table 6 Friedman results of the algorithm for solving Thomson problem

表 6 求解 Thomson 问题算法的 Friedman 检验结果

算法	Ranking
EA_SG2	3.50
CLPSO_SG	3.50
SOS	6.00
PSO	4.12
ga-fminunc	2.69
RLHDE-ILS	1.19

图 3 为 DE 系列算法的收敛曲线。可以看出,在各评估次数下,RLHDE_ILS 算法的收敛性均明显优于其他算法,其他算法并不能收敛到比 RLHDE_ILS 更优的结果。

4.5 消融实验

为验证算法各改进策略的有效性,对不同改进策略组合后进行消融实验,得到的均值使用 Friedman 检验进行分析。结果如表 7 所示。其中,RLDE 表示 Q-learning 算法改进 DE 的算法,RLDE-L 是在 RLDE 的基础上加入 LPSR 的算法,RLDE-L-LS1 表示 RLDE-L 加入 SGD 算法进行局部

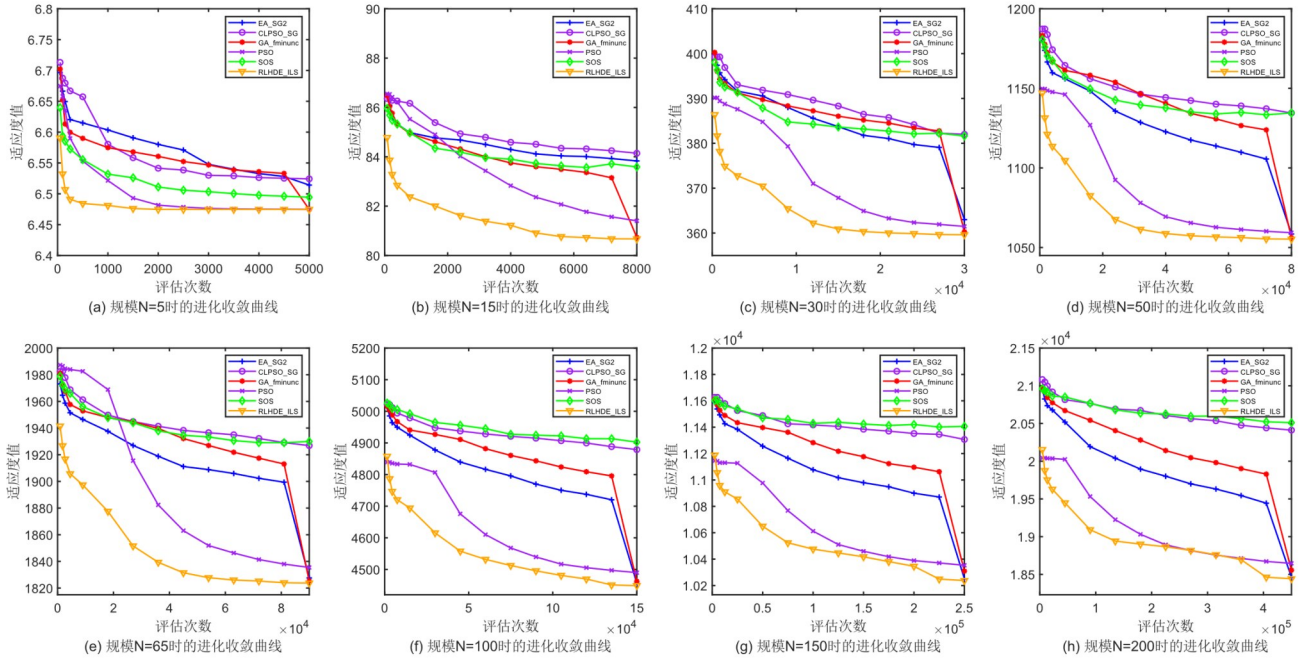


Fig. 2 Convergence curve of algorithm for solving Thomson problem

图2 求解 Thomson 问题算法的收敛曲线

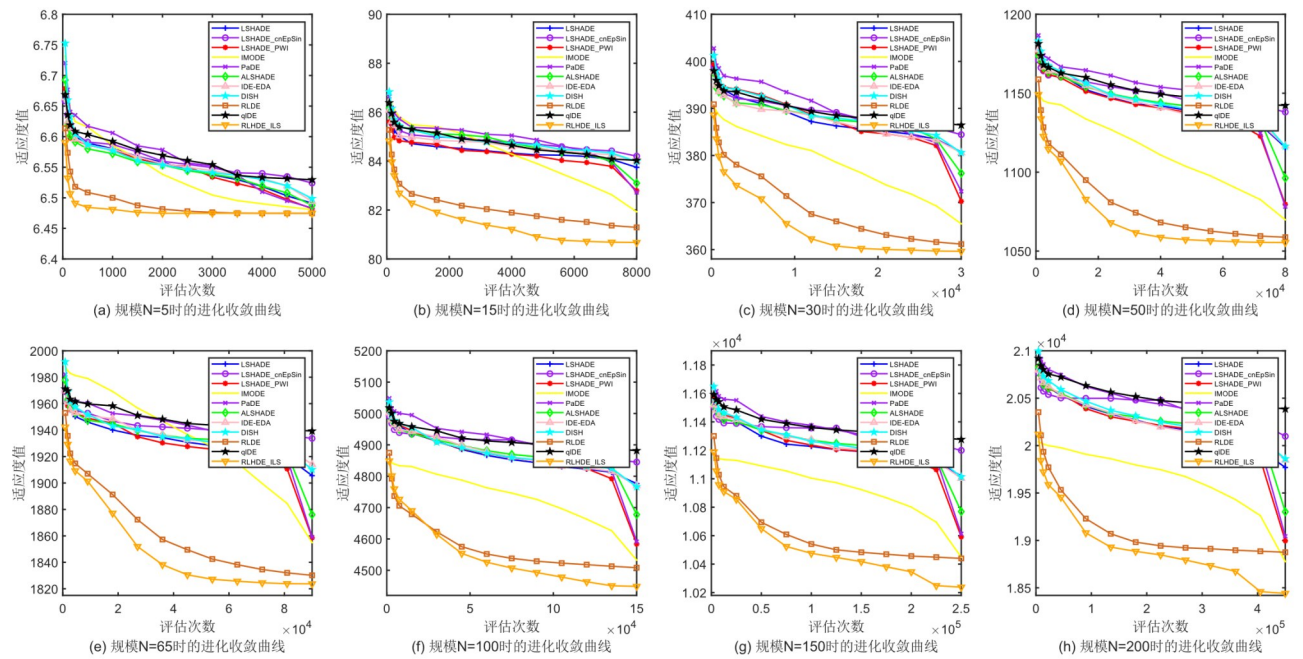


Fig. 3 Convergence curves of DE series algorithm

图3 DE 系列算法的收敛曲线

搜索后的算法, RLDE-L-LS2 表示 RLDE-L 加入 SQP 算法进行局部搜索后的算法, RLHDE-ILS 为最终改进算法。通过 Friedman 检验, RLHDE-ILS 得到 1.333 3 的测试分数, 低于其他算法, 说明改进算法能够综合各部分的优势, 比组合单个策略的算法具有更好的性能。

4.6 仿真实验

为深入展示算法性能, 对优化结果进行可视性分析, 考察方式为点电荷在球面上呈能量最低的一种分布情况。图 4 为规模分别为 5、15、30、50、65、100、150、200 时的仿真

Table 7 Results of ablation experiment

表 7 消融实验结果

算法	Ranking
RLDE	3.500 0
RLDE-L	3.333 3
RLDE-L-LS1	2.833 3
RLDE-L-LS2	4.000 0
RLHDE-ILS	1.333 3

结果。根据算法的运行规律, 初始点电荷随机分布在球面上, 经过算法演化逐渐逼近到目标函数的全局最小值, 此

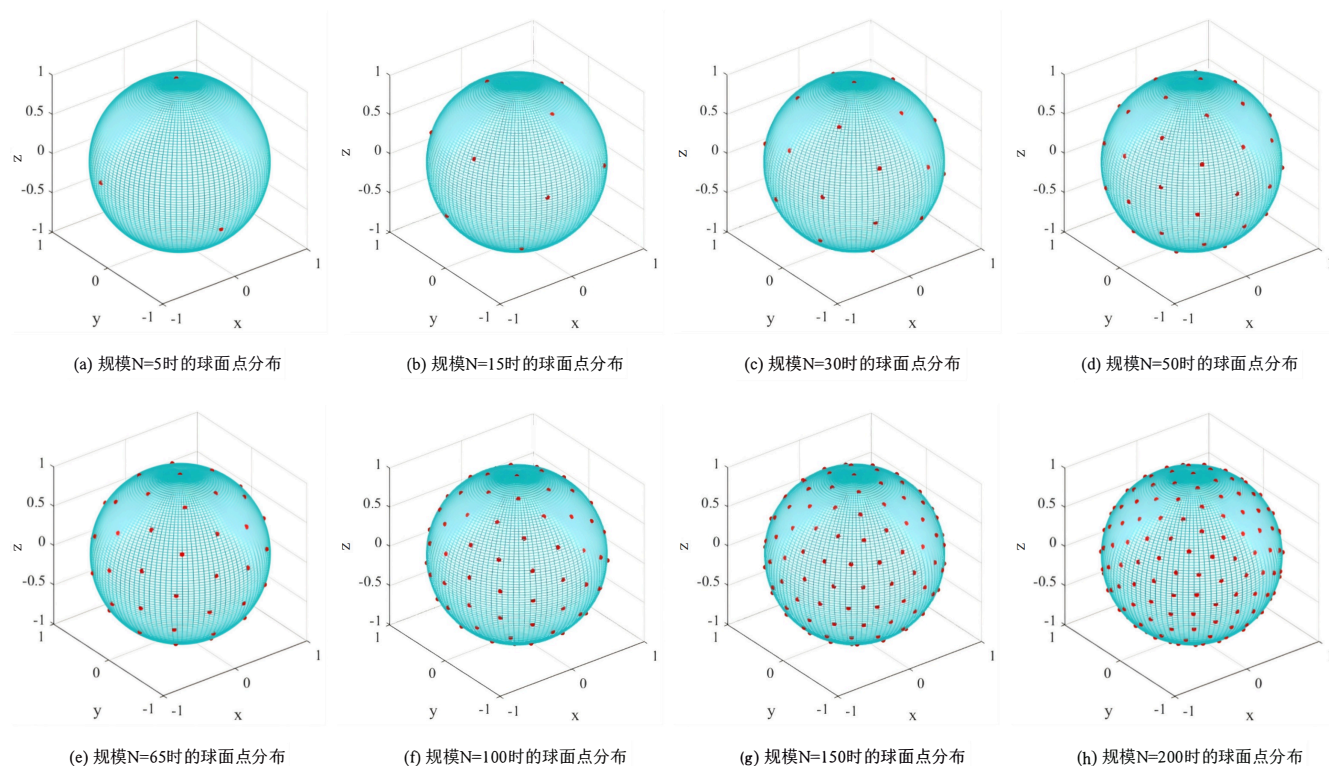


Fig. 4 Spherical point simulation experiment result

图4 球面点仿真实验结果

时球面上的总体势能达到最低,反映在图像上即为点电荷在球面上均匀且稳定排列。在规模为5的仿真图上,为了实现最终势能最低的状态,每个点电荷之间的最小距离被最大化,表现为5个点电荷分散占据球面上5块不同的区域。由于球面对称性以及算法本身的随机特性,同一能量最小化条件下的点电荷配置会出现不同的分布状态。通过对不同规模球面点分布仿真结果的观察,发现所提算法在优化过程中表现出色。在演化的最后阶段,RLHDE_ILS算法能够有效逼近目标函数的全局最小值,且使得球面点在球体上分布均匀,验证了其在求解全局优化问题中的优越性和稳定性。

5 结语

为有效求解 Thomson 问题,本文提出一种创新算法 RLHDE-ILS,通过 Q-learning 算法自适应控制 DE 算法缩放因子参数池,通过一种改进的两阶段混合局部搜索算法增强搜索能力。与其他 15 种最新算法的比较实验结果表明,改进算法在各种规模的 Thomson 问题求解上均展现出卓越性能,能够获得更好的能量近似值。仿真实验中,点在球面上的排列逐渐趋于均匀,亦证实了改进算法的正确性。

然而,本文研究仍存在一些局限性。例如,为增强算法的收敛性和收敛精度而设计的混合局部算子导致在大规模问题求解上时间消耗较大。未来工作将分为3个方

面:一是实现大规模 Thomson 问题更快速、更准确的求解;二是将 Thomson 问题的研究成果扩展应用于卫星星座布局等相关领域;三是探索强化学习与 DE 算法更深层次的融合,进一步提升算法性能。

参考文献:

- [1] SMALE S. Mathematical problems for the next century[J]. The Mathematical Intelligencer, 1998(2):20.
- [2] KANIMOZHI G, RAJATHY R, KUMAR H. Minimizing energy of point charges on a sphere using symbiotic organisms search algorithm[J]. International Journal on Electrical Engineering and Informatics, 2016, 8(1): 29.
- [3] BAUTU A, BAUTU E. Energy minimization of point charges on a sphere with particle swarms[J]. Romanian Journal of Physics, 2009, 54(1-2): 29-36.
- [4] XIANG Y, SUN D Y, FAN W, et al. Generalized simulated annealing algorithm and its application to the Thomson model[J]. Physics Letters A, 1997, 233(3):216-220.
- [5] MORRIS J R, DEAVEN D M, HO K M. Genetic-algorithm energy minimization for point charges on a sphere[J]. Physical Review B, 1996, 53(4): R1740.
- [6] LAKHBAB H, BERNOUSSI S E. A new hybrid approach for tackling Thomson problem[C]//Proceedings of 2013 International Conference on Industrial Engineering and Systems Management, 2013: 1-4.
- [7] LAKHBAB H, BERNOUSSI S E, HARIF A E. Energy minimization of point charges on a sphere with a hybrid approach[J]. American Journal of Clinical Nutrition, 2014, 95(4):916-924.
- [8] AMAL R S, IVAN J S. A quantum genetic algorithm for optimization prob-

- lems on the Bloch sphere[J]. Quantum Information Processing, 2022, 21(2): 43.
- [9] YANG C Z, KANG Z, ZHOU A M, et al. Numerical solution of the 5-point distribution problem on a sphere[J]. Journal of Wuhan University (Science Edition), 2005(3): 313-318.
- 杨承中, 康卓, 周爱民, 等. 球面上 5 点分布问题的数值解法[J]. 武汉大学学报(理学版), 2005(3): 313-318.
- [10] PRICE K, STORN M R. A simple and efficient adaptive scheme for global optimization over continuous space[R]. Technical Report TR-95-012, ICSI, 1995.
- [11] TANABE R, FUKUNAGA A. Evaluating the performance of SHADE on CEC 2013 benchmark problems[C]//2013 IEEE Congress on Evolutionary Computation, 2013: 1952-1959.
- [12] TANABE R, FUKUNAGA A S. Improving the search performance of SHADE using linear population size reduction[C]//2014 IEEE Congress on Evolutionary Computation, 2014: 1658-1665.
- [13] PIOTROWSKI A P. L-SHADE optimization algorithms with population-wide inertia[J]. Information Sciences, 2018, 468: 117-141.
- [14] LI Y, HAN T, ZHOU H, et al. A novel adaptive L-SHADE algorithm and its application in UAV swarm resource configuration problem[J]. Information Sciences, 2022, 606: 350-367.
- [15] HU Z, GONG W, LI S. Reinforcement learning-based differential evolution for parameters extraction of photovoltaic models[J]. Energy Reports, 2021, 7: 916-928.
- [16] HUYNH T N, DO D T T, LEE J. Q-Learning-based parameter control in differential evolution for structural optimization[J]. Applied Soft Computing, 2021, 107: 107464.
- [17] ZHAO F, HU X, WANG L, et al. A reinforcement learning brain storm optimization algorithm (BSO) with learning mechanism[J]. Knowledge-Based Systems, 2022, 235: 107645.
- [18] ZHANG Y Y. Research on hybrid differential evolutionary algorithm and its application in the design of Earth-Moon low-energy transfer orbit[D]. Wuhan: China University of Geosciences, 2022.
- 张燕云. 混合差分演化算法研究及在地月低能转移轨道设计中的应用[D]. 武汉: 中国地质大学, 2022.
- [19] GU Y F. A review of portfolio optimisation based on reinforcement learning[J]. Software Guide, 2021, 20(9): 74-77.
- 顾一凡. 基于强化学习的组合优化综述[J]. 软件导刊, 2021, 20(9): 74-77.
- [20] CHEN Y T. Cooperation coevolution differential evolution with gradient descent strategy for large scale[C]//International Conference on Swarm Intelligence, 2017: 429-439.
- [21] POWELL M J D. The convergence of variable metric methods for nonlinearly constrained optimization calculations[J]. Nonlinear Programming, 1978, 3(2): 27-63.
- [22] BREST J, SEPESY M M. Population size reduction for the differential evolution algorithm[J]. Applied Intelligence, 2008, 29: 228-247.
- [23] AWAD N H, ALI M Z, SUGANTHAN P N. Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems[C]//Donostia: IEEE Congress on Evolutionary Computation, 2017.
- [24] SALLAM K M, ELSAYED S M, CHAKRABORTTY R K, et al. Improved multi-operator differential evolution algorithm for solving unconstrained problems[C]//IEEE Congress on Evolutionary Computation, 2020: 1-8.
- [25] MENG Z, PAN J S, TSENG K K. PaDE: an enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization[J]. Knowledge-Based Systems, 2019, 168: 80-99.
- [26] LI Y, HAN T, TANG S, et al. An improved differential evolution by hybridizing with estimation-of-distribution algorithm[J]. Information Sciences, 2023, 619: 439-456.
- [27] VIKTORIN A, SENKERIK R, PLUHACEK M, et al. DISH algorithm solving the CEC 2019 100-digit challenge[C]//2019 IEEE Congress on Evolutionary Computation, 2019: 1-6.
- [28] ALCALÁ-FDEZ J, SANCHEZ L, GARCIA S, et al. KEEL: a software tool to assess evolutionary algorithms for data mining problems[J]. Soft Computing, 2009, 13: 307-318.

(责任编辑:尹晨茹)