分类号： TP18　　学校代码：10363
密　级： 公开　　学　　号：2210330182

安徽工程大学
Anhui Polytechnic University

硕士学位论文

题目 基于新型粒子群算法的微电网调度研究

论 文 作 者　刘瑞
校 内 导 师　魏利胜（教授）
校 外 导 师　张平改
专业学位类别　电子信息
研究方向（领域）　控制工程与自动化技术

论文提交日期：2024 年 5 月 31 日

分类号：TP18　　单位代码：10363
密　级：公开　　学　　号：2210330182

题　目　基于新型粒子群算法的微电网调度研究

题　目　Research of Microgrid Dispatching Based on Novel Particle Swarm Optimization Algorithm

学　生　姓　名：刘瑞
校　内　导　师：魏利胜教授
校　外　导　师：张平改
专业学位类别　：电子信息
研究方向（领域）：控制工程与自动化技术

论文答辩日期：2024 年 5 月 12 日

安徽工程大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律后果由本人承担。

学位论文作者签名：

日期：　　2024　年　5　月　28 日

安徽工程大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权安徽工程大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密 □，在 ____ 年解密后适用本版权书。

本学位论文属于

不保密 √。

学位论文作者签名：　　指导教师签名：

日期：　2024 年 5 月 28 日　　日期：　2024 年 5 月 28 日

基于新型粒子群算法的微电网调度研究

摘　　要

在能源短缺、环境污染等问题日趋严峻的背景下，微电网作为一种新型电力系统，在改善能源利用效率、减少环境污染及提高电力系统运行可靠性方面具有显著功效。然而，微电网本身也存在出力波动大、控制复杂等不足，故对微电网优化调度进行研究具有重大意义。本文以微电网系统的最优调度为研究对象，构建了含电动汽车柔性负荷的新型微电网模型，在兼顾运行费用和污染气体处理成本的基础上，采用两种新型粒子群算法(Particle Swarm Optimization, PSO)实现其能源优化调度，具体研究内容如下：

首先，构建了一种融合电动汽车灵活负载的微电网优化调度模型。在明确实际运行约束的基础上，通过分析系统各发电单元的运行机理及数学模型，推导出运行费用最小、污染物治理成本最低的单目标函数及综合效益最优的多目标函数，为微电网最优运行提供理论基础。

其次，探讨了一种基于信息交互机制的自适应粒子群算法(An Adaptive Particle Swarm Optimization with Information Interaction Mechanism, APSOIIM)。通过采用混沌序列策略在算法初始化阶段生成均匀分布的粒子；接着引入交互信息机制，随着搜索过程的展开，提高种群的多样性，有效地与相邻粒子的最优信息实现交互；在此基础上，详细推导了 APSOIIM 算法的收敛性，并将其应用于微电网的两个单目标优化调度，结果表明所提算法的有效性。

最后，研究了一种混合粒子群优化与萤火虫算法(An Hybrid Particle Swarm Optimization and Fire-Fly, HPSOFF)。在随机参数初始化机制的基础上，通过引入非线性惯性权重和衰减因子，提高其搜索

能力并输出全局最优；接着以全局最优解为初始种群，自适应地构建临时库，以确保萤火虫算法(Fire-Fly Algorithm, FFA)精准的局部开发；在此基础上，分析了 HPSOFF 算法收敛性并对微电网系统多目标问题进行调度。实验结果表明，HPSOFF 算法可以有效找到折衷解，完成多目标问题的求解。

综上，本文研究的两种新型 PSO 算法能有效优化微电网系统调度，降低运行成本，具有重要的理论研究和实际应用价值。

关键词：粒子群优化算法；交互信息机制；微电网优化调度；混合优化；策略研究

Research of Microgrid Dispatching Based on Novel Particle Swarm Optimization Algorithm

ABSTRACT

Under the background of energy shortage and environmental pollution, microgrid, as a new type of power system, has remarkable effects in improving energy utilization efficiency, reducing environmental pollution and improving the reliability of power system operation. However, the microgrid itself has some shortcomings such as large output fluctuation and complex control, so it is of great practical significance to study the optimal dispatching of microgrid. This paper takes the optimal scheduling of microgrid system as the research object, constructs a new microgrid model with flexible load of electric vehicle, and adopts two new Particle Swarm Optimization Algorithm (PSO) to realize its energy optimal scheduling on the basis of taking into account the operating cost and the pollution gas treatment cost. The specific research contents are as follows:

Firstly, a microgrid optimal dispatching model with flexible load of electric vehicles is constructed. Based on the actual operation constraints, the single objective function with the lowest operation cost and pollutant treatment cost and the multi-objective function with the best comprehensive benefit are derived by analyzing the operation mechanism and mathematical model of each power generation unit in the system, which provides a theoretical basis for the optimal operation of microgrid.

Secondly, An Adaptive Particle Swarm Optimization with Information Interaction Mechanism (APSOIIM) based on information interaction mechanism is discussed. Uniformly distributed particles are

generated in the initialization stage of the algorithm by using chaotic sequence strategy; Then, the interactive information mechanism is introduced to improve the diversity of population and effectively interact with the optimal information of adjacent particles with the development of the search process; On this basis, the convergence of APSOIIM algorithm is deduced in detail, and it is applied to two single-objective optimal dispatching of microgrid. The results show the effectiveness of the proposed algorithm.

Finally, a hybrid particle swarm optimization and Fire-Fly Algorithm (HPSOFF) is studied. On the basis of random parameter initialization mechanism, nonlinear inertia weight and attenuation factor are introduced to improve its search ability and output global optimum; Then take the global optimal solution as the initial population, and build a temporary library adaptively to ensure the accurate local development of Fire-Fly Algorithm (FFA); On this basis, the convergence of HPSOFF algorithm is analyzed and the multi-objective scheduling of microgrid system is carried out. Experimental results show that HPSOFF algorithm can effectively find the tradeoff solution and complete the multi-objective problem.

To sum up, the two new PSO algorithms studied in this paper can effectively optimize the dispatching of microgrid system, reduce operating costs, and have important theoretical research and practical application value.

Liu Rui (Electronic Information)

Supervised by Wei Lisheng

Key words: Particle swarm optimization algorithm; Interactive information mechanism; Optimize scheduling; Hybrid optimization; Strategy research

目　录

插 图 清 单

插 表 清 单

第 1 章 绪论

1.1 选题背景及意义

改革开放以来，化石能源对我国工业和科技的发展起到至关重要的作用，在实现中国经济复苏过程中功不可没。然而，随着全球面临资源枯竭和环境污染等挑战，人们渐渐把重心放到了如何高效利用可再生能源上。截至目前，我国已在联合国大会上明确提出争取在 2030 年前碳排放抵达峰值，之后平稳下滑，在 2060 年前实现相对“零排放”的节能规划[1]。从宏观角度来看，我国主要是以煤炭等化石能源进行发电，目前二氧化碳的总排放量占全球总量的 29%，因此相对来说较难实现降低碳排放的任务。在此背景下，微电网的利用便显得格外重要。

微电网是一种由不同类型的分布式电源、储能设备及负荷等构成的独立发配电系统[2]，其结构如图 1-1 所示。微电网系统包括太阳能光伏、风能、生物质能等各类型的可再生能源，其能够根据实际情况自主调整能源的分配利用，以实现能源利用高效化并降低碳排放。微电网系统通常有两种运行模式：孤岛与并网。孤岛微电网系统是指完全独立于主电网运行的微电网系统，通常用于偏远地区、岛屿、战场等[1]；并网型微电网是与主电网相连的微电网，其可以实现与主电网的双向供电及能量交换，一般用于城市地区、工业园区、医院、学校等。

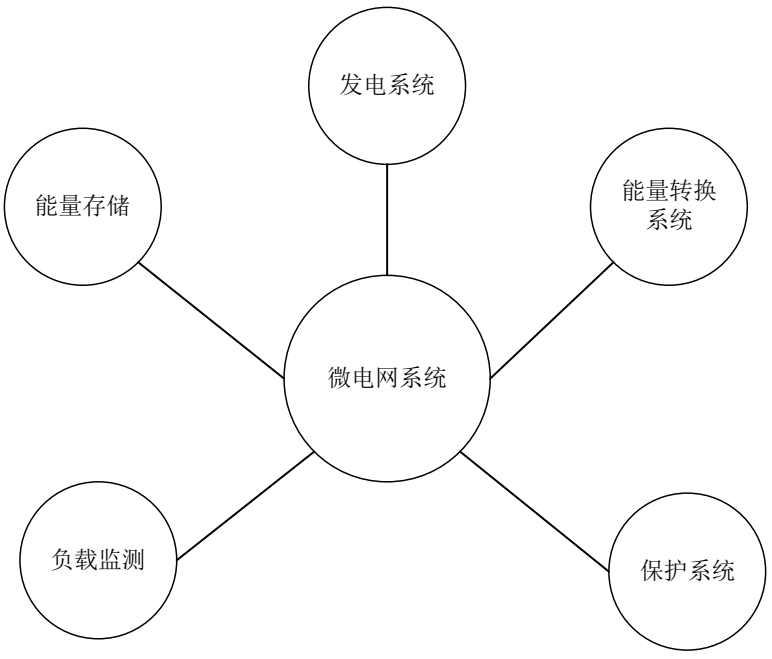


图 1-1 微电网系统组成示意图

随着社会的进步和技术的发展，传统的电力系统逐渐显露出一些固有的缺点。在此背景下，电动汽车(Electric Vehicle, EV)因其环境友好性，保有量越来越大并在电力需求方面引起了广泛关注。目前，许多国家加快了充电设施的建设，并出台了促进电动汽车发展的政策。根据中国《节能与新能源汽车产业发展规划》可知，2030 年电动汽车保有量将超过 6000 万辆[3]。在过去几年中，住宅区内的充电站数量急剧增加，然而，由于电动汽车的充电时间、位置、用户行为及负载曲线是高度动态的，不受控制和不协调的电动汽车大规模渗透到电力系统中将导致微电网系统高度波动并增加电力系统干扰的潜在来源。因此，研究如何实现微电网系统最优调度具有重要意义。

1.2 国内外研究现状

1.2.1 微电网调度模型研究现状

如何合理协调各分布式电源的出力一直是优化调度问题的核心，由于其模型的复杂性和影响因素的多样性，该问题受到了学术界广泛关注。目前，一些研究人员致力于探讨新型微电网调度模型。文献[4]为减少微电网系统运行的碳排放量，提出了一种基于鲁棒优化的微电网低碳经济调度模型；文献[5]为了降低微电网运行总成本，提高可再生能源的效率，探讨了一种基于日前调度和实时调度的两阶段调度模型，以优化微电网调度；文献[6]考虑风能-太阳能机组及燃料发电机组成的混合发电系统，构建了混合整数非线性规划微电网调度模型，实验结果表明电力成本降低了 25%；文献[7]基于分段线性化方法，提出了微电网经济调度期望值模型，以求出经济调度的全局最优解；文献[8]综合考虑两个微电网之间的功率交互以及微电网和主电网之间的相互作用，建立了多微电网的经济调度模型并利用粒子群优化算法求解该模型；文献[9]基于有限步长共识算法的分布式算法，以可控机组日常运行成本最小化为目标，提出了隔离式交直流混合微电网动态经济调度模型；文献[10]设置了 VESS 的 H-微电网动态经济调度(Dynamic Economic Dispatch, DED)模型；文献[11]为探究电动汽车充放电行为和需求侧响应资源对光伏并网微电网系统经济运行的影响，构建了一种含电动汽车、可转移负载等分布式发电的多目标微电网经济调度模型；文献[12]针对可再生能源、需求和价格在微电网实时运行中的不确定性，开发了一种预测控制策略，通过掌握系统当前状态和最新预测信息，以实现微电网日前经济优化调度；文献

[13]为分析可再生能源发电和负荷不确定性对海岛微电网经济运行优化的影响，提出一种基于区间优化的不确定性下多目标经济最优调度模型；文献[14]开发了独立的模块化微电网数学模型以缩短每个变量的可行区域，并确定独立微电网系统的最佳运行策略，从而进行经济调度；文献[15]为最大限度地降低整体规划和运营成本，基于一种两阶段迭代启发式算法，提出了分布式发电机和储能系统的选址模型。

基于上述文献，尽管微电网系统经济调度模型从不同方面得到了加强，并取得了一定的积极成果，但含电动汽车的微电网优化调度尚处于开发阶段，如何进一步提高其模型的泛化性、精准性等值得探索。

1.2.2 微电网优化策略研究现状

另一方面，部分学者则是通过引入新型优化策略求解微电网调度模型。文献[16]提出了一种具有全局约束的 Nelder-Mead 算法，采用方差变量概率策略优化经济负荷分配问题，结果表明，与传统的遗传算法相比，该算法的收敛速度和精度均有明显提高，可以更有效地解决静态和动态经济调度问题；文献[17]探讨了一种新的灰狼优化算法，通过引入独立的局部搜索策略和非次解邻域，解决具有斜率限制和禁止作业面积约束的经济负荷分配问题；文献[18]提出了一种加速分布梯度算法，该方法在局部更新中引入了一个包含自定义加速度增益的动量项，通过给出步长和加速度增益，提高了算法的收敛速度，并在解决经济调度问题上呈现良好的效果；文献[19]提出一种基于差分演化的改进区间优化方法，用于微电网动态经济调度；文献[20]基于控制任务设计全局目标函数，引入一种基于分布式经济模型预测控制算法的微电网最优调度算法，实现最优经济效益微电网调度；文献[21]在基于时间的价格机制下，提出了一种基于多智能体的微电网经济调度协调调度策略，通过需求侧负载、电池和发电侧微源之间的协调控制，可以及时反馈微电网中智能体之间的信息，从而减少微电网运行故障；文献[22]为了提高微电网的经济性和电力营销的智能化服务，探讨了一种基于分时电价机制的微电网动态经济调度模型，采用非支配排序遗传算法(Non-Dominant Sorting Genetic Algorithm, NSGA)-II 算法的变体，同时引入外部惩罚函数来处理约束条件，便于求解微电网多目标优化模型；文献[23]采用改进的布谷鸟搜索(Cuckoo Search, CS)算法来优化多微电网的电力调度；文献[24]考虑光伏发电和风力发电的随机性，提出改进的人工蜂群算法优化，旨在满足负载和平衡需求的同时最大

限度地降低发电成本和气体排放，实现微电网最优经济调度；文献[25]提出了一种改进的细菌觅食优化算法，以实现微电网的经济优化调度，实验表明，新型细菌觅食优化算法能够有效降低微电网运行成本，提高经济效益，减少环境污染；文献[26]针对三种不同场景研究了低压微电网系统，在结合传统灰狼优化器、正弦余弦算法和乌鸦搜索算法特性的基础上提出了混合群体智能优化算法；文献[27]采用改进的布谷鸟搜索算法求解含源-网-荷-储设备的微电网电力调度模型，实验结果表明，该方法可以有效确定最佳功率调度；文献[28]提出修正谐波搜索算法，以解决考虑太阳能和风能成本函数的微电网联合经济排放调度(Combined Economic Emission Dispatch, CEED)问题，结果表明，修正谐波搜索算法寻优能力优于其他对比算法；文献[29]重点介绍了混沌磷虾群(Chaotic Krill Herd, CKH)算法在微电网中经济调度(Economic Dispatch, ED)和联合经济排放调度问题中的应用，结果清楚地表明，与标准 KH 和其他知名的元启发式搜索算法相比，CKH 算法在搜索最优解具有优越性；文献[30]提出了一种利用精确扩散策略实现稀疏通信网络下精确收敛的全分布式方法，优化包括分布式发电机和需求侧柔性负载在内的所有可调度组件的有功功率，实现资源的经济利用。

上述研究结果表明，目前学者们已提出各种优化算法进行微电网经济调度且取得了显著成果，但对于复杂多目标函数，寻优结果并不总是令人满意。因此，如何有效提高优化算法的寻优能力、收敛精度及迭代速度值得探讨。

1.3 主要研究内容

随着“双碳”目标的提出，微电网系统因其能够缓解分布式电源灵活、多样性的并网问题，实现对负荷多种能源形式的可靠供给而被广泛使用。本文主要围绕微电网优化调度模型的构建及其求解算法展开研究，首先构建了三种优化调度模型，其次探讨出两种新型 PSO 算法，并将其推广到微电网优化调度模型的求解上，全文框架及各章节安排如下：

第 1 章：阐述了论文的选题背景及意义，分析了当前微电网系统调度模型及优化策略研究现状；

第 2 章：构建了含电动汽车的微电网系统并进行负荷预测。阐述了微电网系统各发电单元机理与出力模型，包括风电机组、光伏发电、微型燃气轮机、柴油发电机和蓄电池，之后在考虑实际约束条件的基础上，对微电网优化调度模型进

行分析并建立系统运行费用最小、污染物治理成本最低的单目标函数及综合经济效益最优的多目标函数；

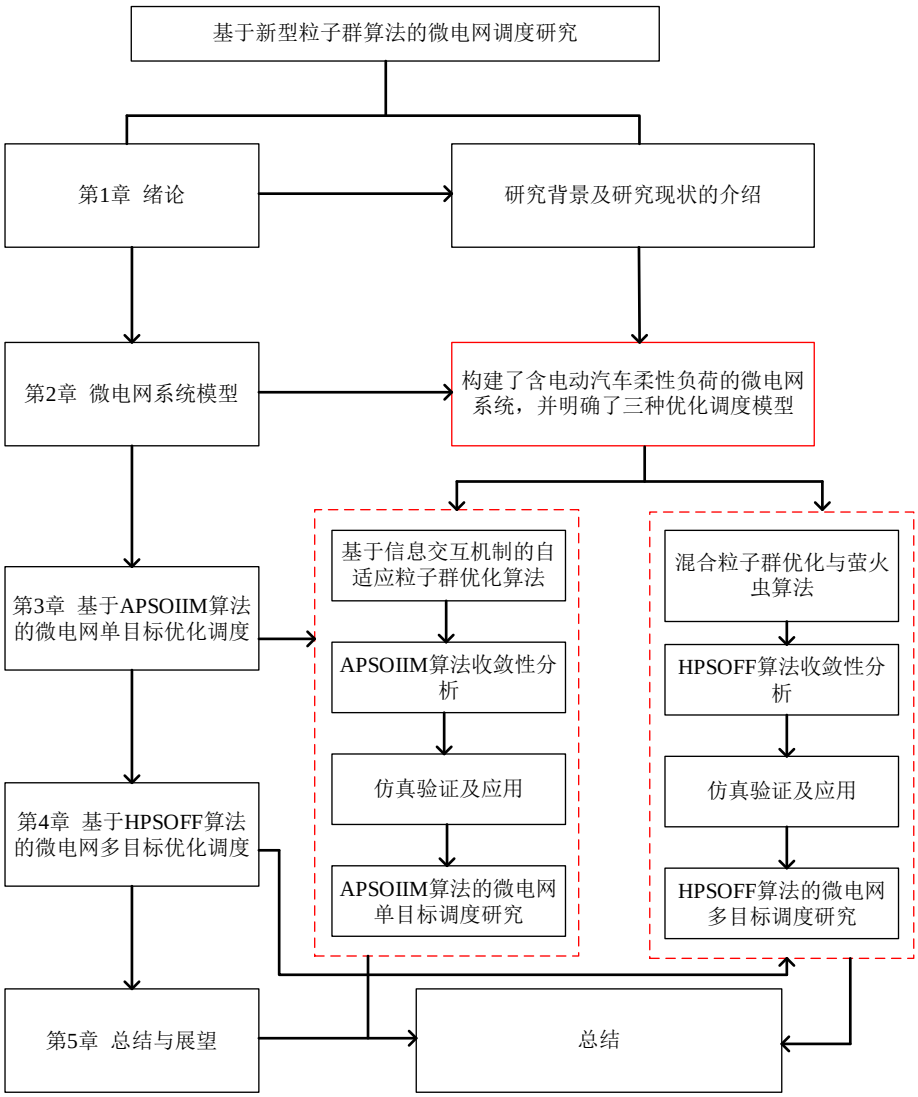


图 1-2 研究内容框架图

第 3 章：探讨了基于信息交互机制的自适应粒子群算法(An Adaptive Particle Swarm Optimization with Information Interaction Mechanism, APSOIIM)，并对其全局收敛性进行了推导。针对基本粒子群算法易陷入局部最优的缺点，引入交互信息机制以提高算法的性能，最后将其应用于 CEC2014、CEC2017 测试函数及实际问题进行寻优，结果表明 APSOIIM 算法可准确识别全局最优解，实现微电网单目标优化调度；

第 4 章：研究了混合粒子群优化与萤火虫算法(An Hybrid Particle Swarm Optimization and Fire-Fly, HPSOFF)。阐述了微电网系统单目标优化调度的束缚

性，探讨了动态种群萤火虫算法、改进 PSO 及混合算法优化步骤，最后结合实验研究结果验证 HPSOFF 算法在优化精度和收敛速度具有显著优势，能够有效解决综合经济效益最优的多目标优化问题；

第 5 章：对本文主要研究内容进行概括总结，并指出进一步研究思路及未来展望。

第 2 章 微电网系统模型

2.1 传统微电网系统数学模型

传统微电网系统由分布式电源、能量变换组件、储能装备等组成[2]，各个组件在系统中都有其独特的作用和功能，能够为电力系统的稳定运行和可持续发展提供重要支持。具体介绍如下：

1）风力发电

风是一种取之不尽的清洁来源，在沿海海岛、草原牧区和山地高原等缺水、交通不便的地区，风机发电(Wind Turbine, WT)具有广阔的前景。中国风能资源丰富，推动风电技术发展不仅能够缓解我国面临的雾霾问题，还有助于调整能源结构和转变经济发展模式，并网型风机发电系统如图 2-1 所示。

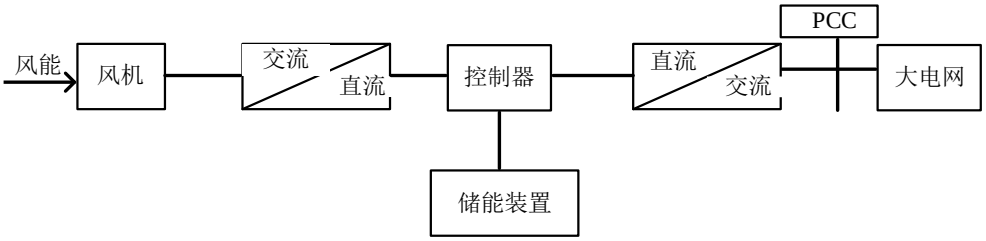


图 2-1 并网型风机发电系统图

图 2-2 表明了风速和风电机组的出力关系，可以发现在风速未达到临界速度时，风电机组不进行功率输出，当到额定速度时，风电机组稳定输出，当超过切出风速时，风电机组关断。

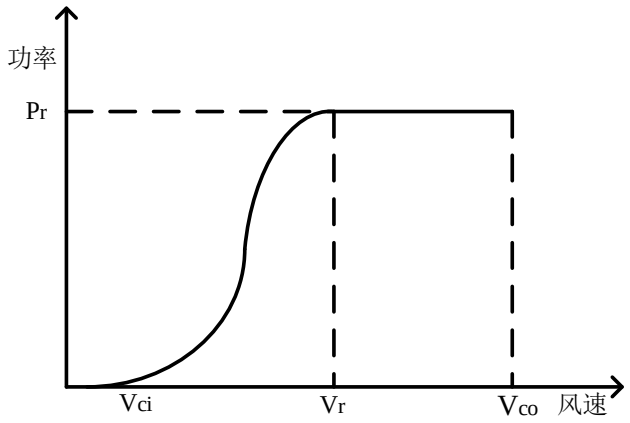


图 2-2 风机出力功率与风速的关系图

具体数学关系如式(2.1)所示：

$$P_{\mathrm{WT}}(t)=\begin{cases}0, & V(t)\le V_{\mathrm{ci}}(t)\\ \mathrm{a}V^{3}(t)+bV^{2}(t)+cV(t)+d, & V_{\mathrm{ci}}(t)<V(t)\le V_{\mathrm{r}}\\ P_{\mathrm{r}}, & V_{\mathrm{r}}<V(t)\le V_{\mathrm{co}}(t)\\ 0, & V_{\mathrm{co}}(t)<V(t)\end{cases} \tag{2.1}$$

式中：$P_{\mathrm{WT}}(t)$为风电机组的输出功率，其额定功率通常表示为P_{r}；$V(t)$为特定时刻的实际风速；$V_{\mathrm{ci}}(t)$及$V_{\mathrm{co}}(t)$分别代表临界风速及切出风速；额定风速通常表示为V_{r}；a、b、c、d为风速参数。

2）光伏发电

光伏发电(Photovoltaic Cell, PV)装置由太阳能电池板、控制器和逆变器等组件组成，该装置能将光能转换为电能，实现清洁能源的生产和利用，对环境保护和能源结构调整具有重要意义。光伏发电系统如图 2-3 所示。

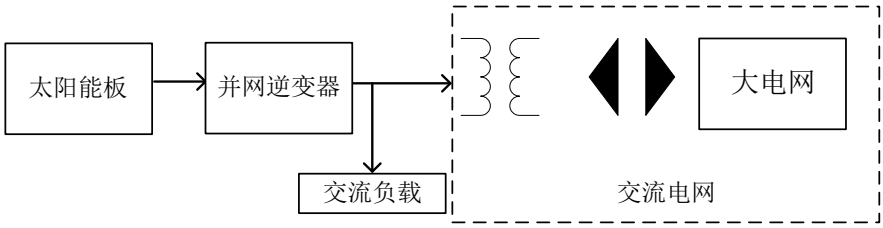


图 2-3 光伏发电系统图

其输出功率可表示为：

$$P_{\mathrm{PV}}(t)=R_{pv}q_{pv}\frac{I_T}{I_{STC}}[1+\alpha_p(T_c-T_{stc})] \tag{2.2}$$

式中：$P_{\mathrm{PV}}(t)$为光伏发电的出力功率，标准条件下光伏发电出力功率通常表示为R_{pv}；q_{pv}代表光伏的降额系数，一般设定为 0.8；I_T、I_{STC}分别表示实际及标准条件下的太阳辐射强度；α_p为 PV 电池板的温度系数；T_c、T_{stc}分别代表特定时刻及标准条件下的的 PV 电池温度。

3）微型燃气轮机

微型燃气轮机(Micro Turbine, MT)是一种利用燃气、燃油、空气作为工作介质的转子式热机，在常规工作条件下，利用燃气涡轮带动发电机工作，其能量流向如下图 2-4 所示。

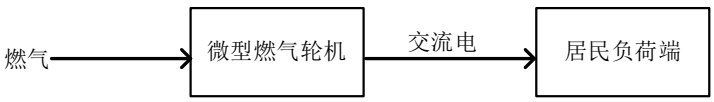


图 2-4 微型燃气轮机能量流向图

在考虑微型燃气轮机模型时，需要将其成本分为燃料成本、运行成本、环境成本三部分，具体如下:

$$\eta_{MT}(t)=0.0753[\frac{P_{MT}(t)}{65}]^3-0.3095[\frac{P_{MT}(t)}{65}]^2+0.4174\frac{P_{MT}(t)}{65}+0.1068 \tag{2.3}$$

$$C_{\mathrm{MT,F}}(t)=\frac{C}{LHV}\frac{P_{MT}(t)}{\eta_{MT}(t)} \tag{2.4}$$

$$C_{\mathrm{MT,OM}}(t)=K_{MT,OM}P_{MT}(t) \tag{2.5}$$

$$C_{\mathrm{MT,EN}}(t)=\sum_{k=1}^{n}(C_k\gamma_{mt,k})P_{MT}(t) \tag{2.6}$$

式中：$C_{\mathrm{MT,F}}(t)$为燃料成本、$C_{\mathrm{MT,OM}}(t)$为运行成本、$C_{\mathrm{MT,EN}}(t)$为环境成本；$P_{MT}(t)$表示在t时刻微型燃气轮机的输出功率；$K_{MT,OM}$是运行维护成本系数；C_k表示第k种污染物治理费用参数；$\gamma_{mt,k}$表示微燃机工作时污染物k的排放量。

4）柴油发电机

柴油机发电(Diesel Generator, DE)系统由柴油和发电机组成。柴油发电机作为一种常用的燃料型发电机，在各个领域中扮演着重要角色，其高效性和稳定性使得柴油发电机成为许多场合的首选，对保障电力供应的稳定性和可靠性具有重要意义，具体如下。

$$C_{\mathrm{DE,F}}(t)=\alpha P_{DE}^2(t)+\beta P_{DE}(t)+\gamma \tag{2.7}$$

$$C_{\mathrm{DE,OM}}(t)=K_{DE,OM}P_{DE}(t) \tag{2.8}$$

$$C_{\mathrm{DE,EN}}(t)=\sum_{k=1}^{n}(C_k\gamma_{de,k})P_{DE}(t) \tag{2.9}$$

式中：$C_{\mathrm{DE,F}}(t)$为燃料成本、$C_{\mathrm{DE,OM}}(t)$为运行成本、$C_{\mathrm{DE,EN}}(t)$为环境成本；$P_{DE}(t)$表示在t时刻处柴油发电机的功率；$K_{DE,OM}$是柴油发电机的运维成本系数；第k种污染物治理费用的参数表示为C_k；$\gamma_{de,k}$代表柴油发电机工作时污染物k的排放量；$\alpha=0.00011$，$\beta=0.1801$，$\gamma=6$。

5）蓄电池

蓄电池存储系统(Battery Energy Storage System, BESS)是一种常见的储能装置，其在能源领域具有显著作用。在实际生活中，由于蓄电池存储系统只能处于充电或放电状态，不能同时进行两个过程，因此其工作状态可表示为：

$$SOC(t)=\begin{cases}SOC(t-1)+\dfrac{1}{\eta^-}P_{BESS}(t), & P_{BESS}(t)\le 0\\ SOC(t-1)+\eta^+P_{BESS}(t), & P_{BESS}(t)>0\end{cases} \tag{2.10}$$

式中：$SOC(t)$代表蓄电池t时刻储能容量，蓄电池$t-1$时刻储能容量则表示为

$SOC(t-1)$；$P_{\mathrm{BESS}}(t)$为某时刻蓄电池充放电功率；η^{+}、η^{-}分别代表充放电效率。

2.2 计及电动汽车的微电网系统模型

2.2.1 微电网接入电动汽车的基本结构及简介

随着电动汽车的持续发展和普及，为了鼓励消费者购买电动汽车，国家制定一系列的优惠政策，越来越多的消费者将逐渐选择电动汽车作为自己的交通工具从而响应低碳生活的呼吁。电动汽车作为灵活负载，其参与微电网优化调度能够有效缓解对配电网的冲击，因此，探求含电动汽车的微电网经济优化调度具有重要意义。本节将以居民微电网模型为研究对象，构建含电动汽车灵活负载的微电网调度模型。由于居民微电网系统中用户对电动汽车的行为存在普遍规律，因此所建立的调度模型具有较好的拟合性及实用性，具体结构如图 2-5 所示：

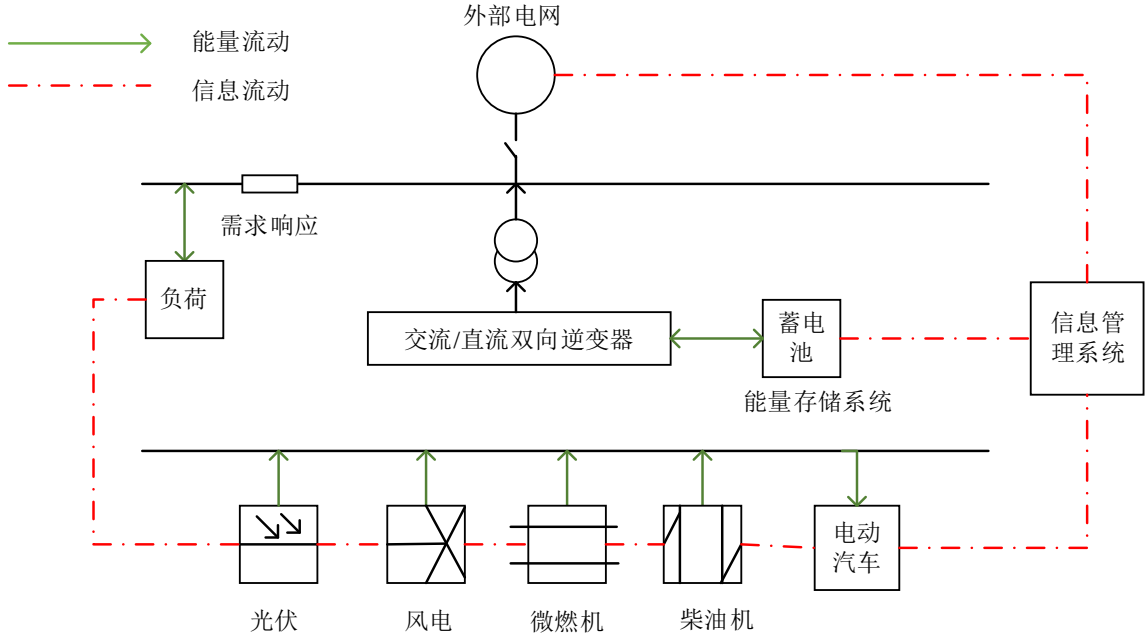


图 2-5 含电动汽车的微电网系统图

2.2.2 电动汽车充电负荷模型

1）电动汽车出行预测

对于朝九晚五的上班族来说，其出行及返回时间具有规律性。美国交通运输部 2009 年的调研结果显示，家用电动汽车出行时刻和返程时间均服从正态分布，日行驶里程满足对数正态分布[31]。

电动汽车出行时刻密度函数为：

$$f(t)=\begin{cases}\dfrac{1}{\delta_t\sqrt{2\pi}}\exp[\dfrac{(t+24-u_t)^2}{2\delta_t{}^2}], & 0<x\le(u_t-12)\\ \dfrac{1}{\delta_t\sqrt{2\pi}}\exp[-\dfrac{(t-u_t)^2}{2\delta_t{}^2}], & (u_t-12)<x\le 24\end{cases} \tag{2.11}$$

式中：t 为时间；$u_t=17.6$；$\delta_t=3.4$。

电动车日行驶里程密度函数为：

$$f(s)=\frac{1}{s\delta_s\sqrt{2\pi}}\exp[-\frac{(\ln s-u_s)^2}{2\delta_s{}^2}] \tag{2.12}$$

式中：s 为日行驶路程；$u_s=3.20$；$\delta_s=0.88$。

2）电动汽车负荷预测

为了预测电动汽车负荷需求，采用蒙特卡洛法[32]对居民微电网系统中 50 台电动汽车的日常用电行为进行仿真，并建立其起止电量的离散式概率模型。通过对概率模型随机采样，最终得到电动汽车无序充电模型，具体流程如图 2-6 所示。

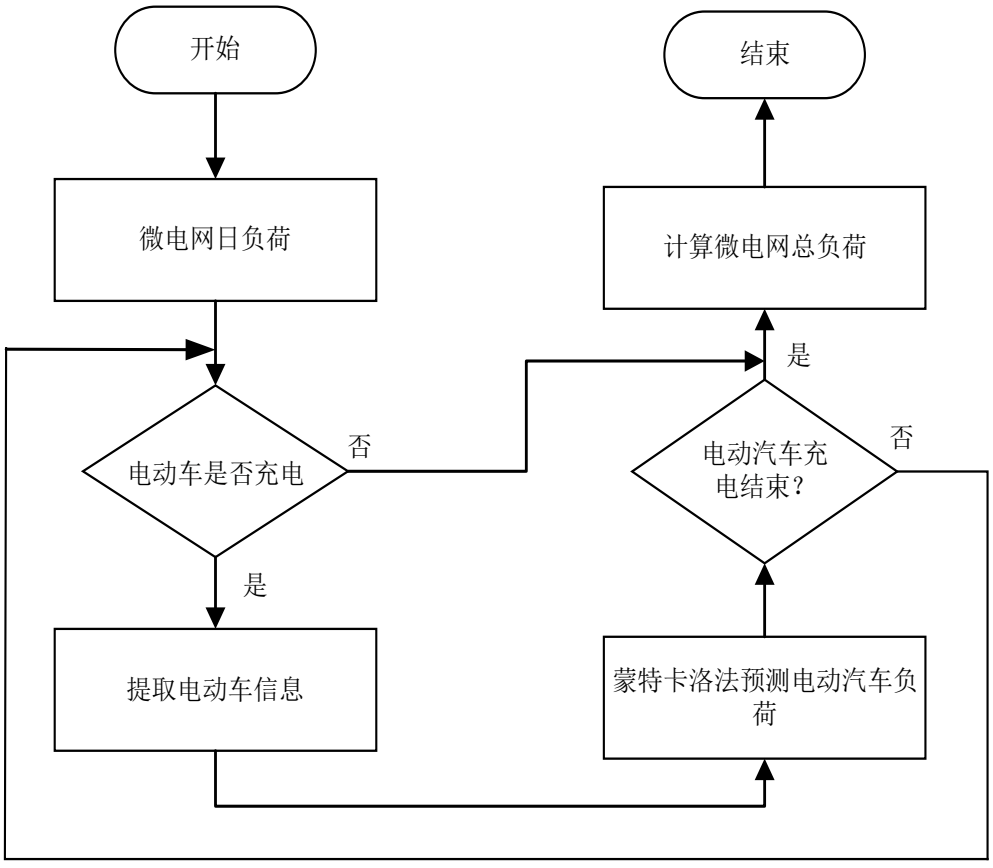


图 2-6 电动汽车无序充电流程图

其日负荷预测曲线如图 2-7 所示：

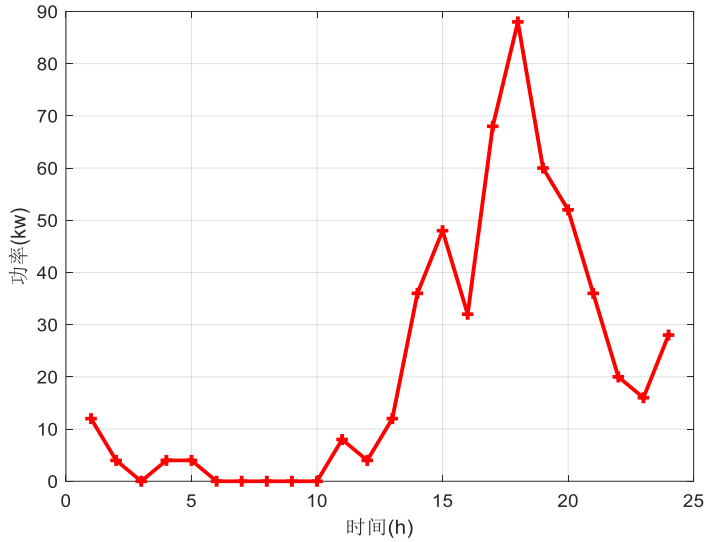


图 2-7 电动汽车负荷预测图

2.2.3 微电网运行约束条件

为了确保微电网系统安全稳定运行，需要考虑各个元件的参数阈值和整个系统的总体需求约束。只有当这些必要约束得到满足，微电网才能顺利运行，为用户提供可靠的供电服务。具体约束如下：

1）功率平衡约束

$$P_{\mathrm{WT}}(t)+P_{\mathrm{PV}}(t)+P_{\mathrm{MT}}(t)+P_{DE}(t)+P_{\mathrm{grid}}(t)+P_{BESS}(t)=P_{\mathrm{load}}(t)+P_{EV}(t) \tag{2.13}$$

式中：$P_{\mathrm{WT}}(t)$、$P_{\mathrm{PV}}(t)$、$P_{\mathrm{MT}}(t)$、$P_{DE}(t)$及$P_{\mathrm{grid}}(t)$分别为风机、光伏阵列、微型燃气轮机、柴油发电机及大电网在某时刻的出力；$P_{BESS}(t)$、$P_{\mathrm{load}}(t)$及$P_{EV}(t)$分别为蓄电池的储存功率、负荷端功率及电动汽车负荷功率。

2）各设备输出功率上下限约束

$$P_{\mathrm{Gi}}^{\min} \le P_{\mathrm{Gi}}(t) \le P_{\mathrm{Gi}}^{\max} \tag{2.14}$$

式中：$P_{\mathrm{Gi}}^{\min}$、$P_{\mathrm{Gi}}^{\max}$表示风机、光伏机组、微型燃气轮机及柴油发电机输出功率的上下限。

3）蓄电池约束

$$P_{BESS}^{\min}(t) \le P_{BESS}(t) \le P_{BESS}^{\max}(t) \tag{2.15}$$

$$SOC_{\min}(t) \le SOC(t) \le SOC_{\max}(t) \tag{2.16}$$

式中：$P_{BESS}^{\max}(t)$、$P_{BESS}^{\min}(t)$为蓄电池出力的上下限；$SOC_{\max}(t)$、$SOC_{\min}(t)$表示t时刻储能容量的上下限。

4）电能交互功率约束

$$P_{\mathrm{grid}}^{\min}(t) \le P_{\mathrm{grid}}(t) \le P_{\mathrm{grid}}^{\max}(t) \tag{2.17}$$

式中：$P_{\mathrm{grid}}^{\min}(t)$、$P_{\mathrm{grid}}^{\max}(t)$分别为电能交互功率最小值和最大值。

5）电动汽车充电约束

$$P_{EV}^{\min} \le P_{EV}(t) \le P_{EV}^{\max} \tag{2.18}$$

式中：$P_{EV}^{\min}$ 、$P_{EV}^{\max}$ 分别为电动汽车功率下限和上限。

6）电动汽车充电完成自动退出约束

电动汽车的车载电池若长期不充电，会极大地缩短其使用寿命，所以需要对其进行限制，当电动汽车 SOC 值到达上限时，自动退出充电模式，保证电池使用寿命。

$$SOC_{\min}^{EV} \le SOC_{t}^{EV} \le SOC_{\max}^{EV} \tag{2.19}$$

式中：$SOC_{\min}^{EV}$、$SOC_{\max}^{EV}$ 分别为电动汽车荷电状态最小值和最大值。

2.3 建立微电网优化调度模型

2.3.1 建立系统运行费用最小的微电网调度模型

从经济角度出发，建立微电网运行费用最小的数学模型。如式(2.20)所示，可将目标函数分为以下三个部分：

$$F_{yun} = \min \sum_{t=1}^{T} F_1(t) + F_2(t) + F_3(t) \tag{2.20}$$

式中：F_{yun} 为微电网系统的总运行费用；F_1、F_2 及 F_3 分别为微电网与主电网交互总成本、微燃机总运行成本及柴油发电机总运行成本；T 为调度的时段数；t 为时刻。

1）微电网与大电网交互总成本：

$$\begin{aligned} F_1(t) &= C_{sell}(t) + C_{buy}(t) \\ C_{buy}(t) &= c_{buy}(t) P_{buy}(t) \\ C_{sell}(t) &= c_{sell}(t) P_{sell}(t) \end{aligned} \tag{2.21}$$

式中：$P_{buy}(t)$、$P_{sell}(t)$ 分别表示在 t 时刻微电网与大电网的购电、售电功率；$c_{buy}(t)$、$c_{sell}(t)$ 表示在 t 时刻微电网与大电网的购售单价。

2）微型燃气轮机总运行成本：

$$F_2(t) = C_{\mathrm{MT,F}}(t) + C_{\mathrm{MT,OM}}(t) \tag{2.22}$$

式中：$C_{\mathrm{MT,F}}(t)$ 为燃料成本、$C_{\mathrm{MT,OM}}(t)$ 为运行成本。

3）柴油发电机总运行成本：

$$F_3(t) = C_{\mathrm{DE,F}}(t) + C_{\mathrm{DE,OM}}(t) \tag{2.23}$$

式中：$C_{\mathrm{DE,F}}(t)$ 为燃料成本、$C_{\mathrm{DE,OM}}(t)$ 为运行成本。

2.3.2 建立系统污染物治理成本最低的微电网调度模型

为了积极推动新能源的发展，构建绿色环保微电网[33]，本节考虑各设备的污染成本并建立系统污染物治理成本最低的目标函数。由于光伏阵列和风力机组作为清洁能源，其发电过程中不会产生污染性气体，故可忽略其污染气体治理费用。而微型燃气轮机、柴油发电机及大电网因其运行机理，必将产生污染物治理成本，具体为：

$$\begin{aligned} F_{huan} &= \min \sum_{t=1}^{T} C_{\mathrm{DE,EN}}(t) + C_{\mathrm{MT,EN}}(t) + C_{\mathrm{GRID,EN}}(t) \\ C_{\mathrm{GRID,EN}}(t) &= \sum_{k=1}^{n} (C_k \gamma_{grid,k}) P_{buy}(t) \end{aligned} \tag{2.24}$$

式中：T 表示调度时段数，t 为时刻；$C_{\mathrm{DE,EN}}(t)$ 为柴油发电机环境成本；$C_{\mathrm{MT,EN}}(t)$ 为微型燃气轮机污染气体治理费用；$C_{\mathrm{GRID,EN}}(t)$ 为大电网的环境成本，其运行产生的 k 类污染物排放量表示为 $\gamma_{grid,k}$；C_k 代表处理 k 类污染物的成本系数。

2.3.3 建立系统综合效益最优的微电网调度模型

如上所述，已分别建立运行费用最小、污染物治理成本最低的单目标优化调度模型，而在实际生活中，对某微电网系统模型进行评估时有多种因素需要综合考虑，故采用加权方式综合考虑运行成本以及环保成本，建立综合效益评估指标，具体为：

$$\min F_{zong} = \lambda_1 F_{yun} + \lambda_2 F_{huan} \tag{2.25}$$

$$\begin{cases} \lambda_1 + \lambda_2 = 1 \\ 0 \le \lambda_1 \le 1, 0 \le \lambda_2 \le 1 \end{cases} \tag{2.26}$$

式中：F_{zong} 为微电网系统综合效益；λ_1 和 λ_2 表示运行费用及污染物治理成本所占权重，本文令 $\lambda_1 = 0.5$，$\lambda_2 = 0.5$。

2.4 本章小结

本章以含电动汽车的并网型微电网系统为研究对象，分析各分布式电源的运行机理并构建相应数学模型，然后提出运行费用最小、污染物治理成本最低的单目标优化调度及系统综合效益最优的多目标联合优化调度，最后明确微电网系统运行须满足的约束条件，为后续的能源优化调度奠定基础。

第 3 章 基于 APSOIIM 算法的微电网单目标优化调度

粒子群算法由于其结构简单，易于实现，现已成功应用于诸多优化问题。然而，标准粒子群算法往往陷入局部最优，后期表现出较弱的可搜索性。为了解决该问题，本章将提出一种具有信息交互机制的自适应粒子群优化算法，从初始化方式、寻优机制及粒子更新策略方面丰富种群多样性以提升寻优性能。此外，对所提算法进行严格收敛性证明。最后，将 APSOIIM 算法实际应用于 CEC2014、CEC2017 测试集、实际工程问题和微电网系统下的优化调度，其系统流程图如图 3-1 所示。

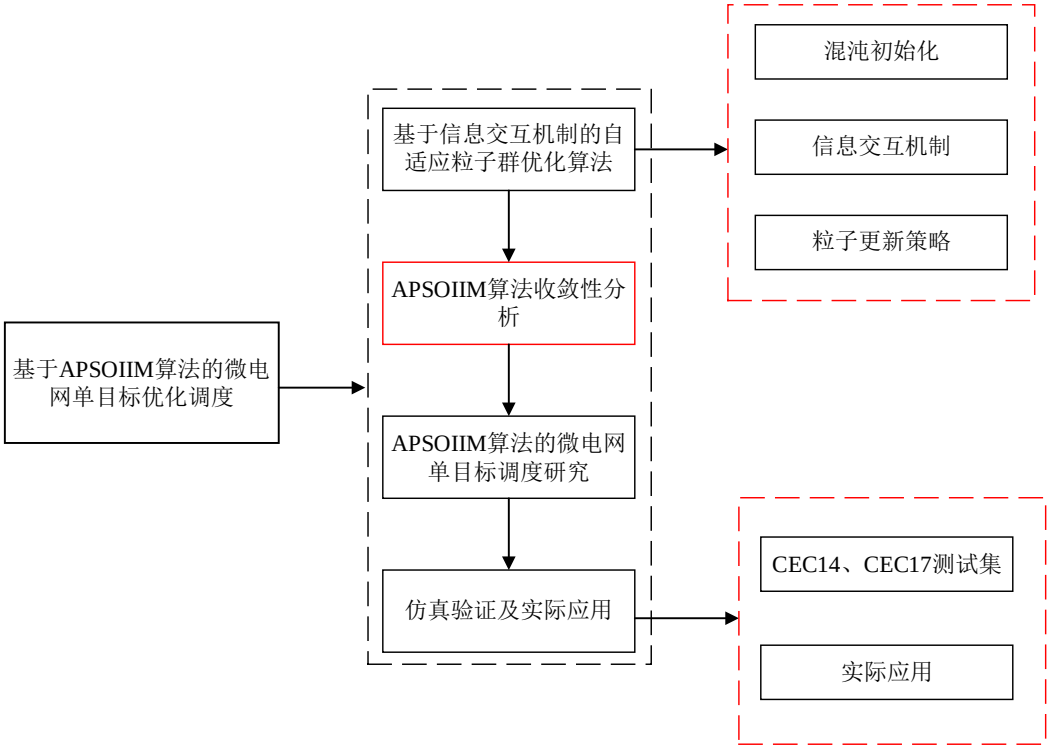


图 3-1 APSOIIM 系统流程图

3.1 基于信息交互机制的自适应粒子群优化算法

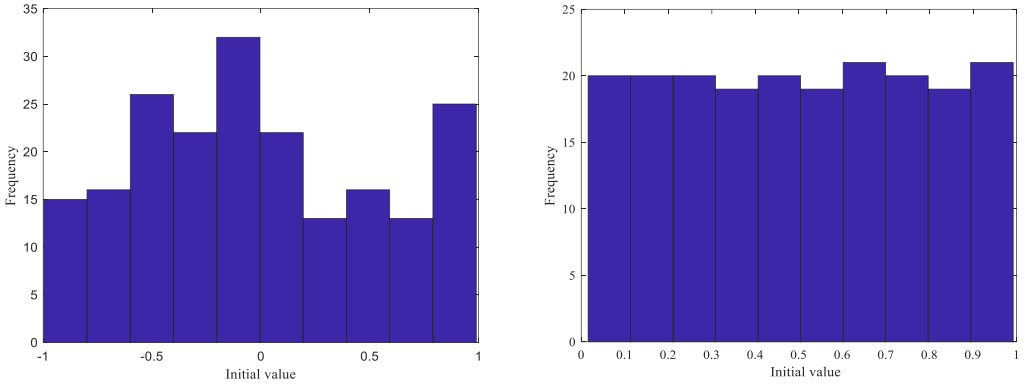
3.1.1 混沌初始化

为了尽可能实现种群粒子均匀分布，引入了混沌初始化策略。首先随机生成初始粒子，然后通过式(3.2)执行混沌操作，获得的粒子将均匀分布在解空间内，其效果如图 3-2 所示。

$$X_i = X_{\min} + (X_{\max} - X_{\min}) * r_1 \tag{3.1}$$

$$X_{i+1}=\mathrm{mod}\left(X_i+0.2-\frac{0.5}{2\pi}\times\sin(2\pi X_i),1\right) \tag{3.2}$$

式中：X_i 为第 i 个粒子的位置；X_{i+1} 是经混沌操作后的第 i 个粒子的位置；$X_{\min}$ 和 $X_{\max}$ 为位置限制；r_1 是介于 0 和 1 之间的随机数。



(a).随机初始化　　(b).混沌初始化

图 3-2 粒子分布直方图

图 3-2 表明，混沌初始化产生随机值的频率分布明显优于随机初始化，验证了混沌操作能够有效改善粒子分布，为后续优化奠定坚实的基础。

3.1.2 信息交互机制

针对标准粒子群算法普遍易陷入局部最优的问题，研究了信息交互机制以增强种群多样性。所提机制基于全局模型拓扑结构，从各个角度评估粒子更新，不仅考虑个体最优和全局最优，同时追踪邻近粒子个体最优，从多方面引导粒子趋向实际最优，具体如下图 3-3 所示。

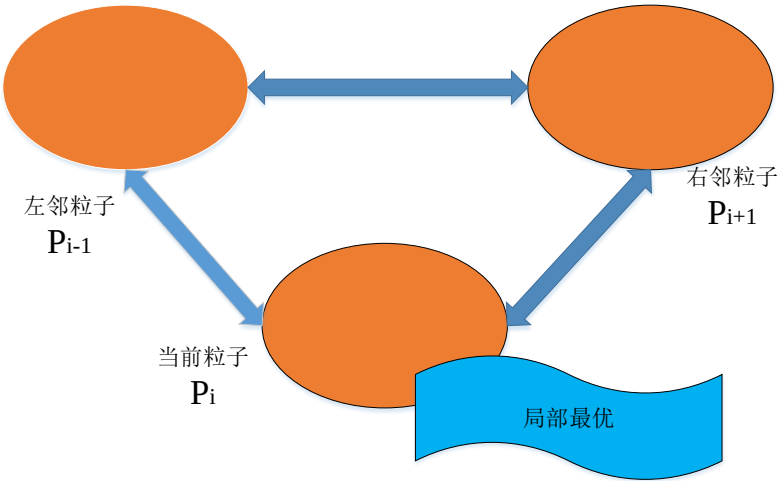


图 3-3 交互信息机制示意图

如图 3-3 所示，信息交互机制下的每次升级迭代都从单粒子转为多粒子（当前粒子、左邻粒子及右邻粒子）。在这种情况下，无论三个粒子中的任何一个落

入局部最优，与其相邻的另外两个粒子通过信息共享仍然能保持确切的寻优方向，降低陷入局部最优概率。

3.1.3 粒子更新策略

针对 PSO 迭代后期精度低且收敛性差的问题，将对其拓扑结构进行改进。常见 PSO 拓扑结构是全局模型，即在算法的迭代演化中仅考虑当前个体最优和全局最优，极易陷入局部最优。此外，还有局部模型，该模型由于只能在相邻粒子之间交换信息，在避免陷入局部最优方面确实有效，但其迭代时间将大幅度增加并伴有迭代速度变慢，故开发了全新迭代形式，优化了粒子速度更新公式，使其不仅与邻近粒子信息交互，同时考虑全局模型，具体如式(3.3)和式(3.4)所示。通过这种方法，粒子更新不仅受到 $pbest_{i,j}^{t}$ 和 $gbest_{i,j}^{t}$ 的影响，同时也会受相邻粒子局部最优的影响。这种综合考虑粒子演化的迭代方式，在保持迭代速度的同时丰富了种群的多样性。

此外，值得注意的是，将第 t 次迭代中粒子的适应度值与上一代的最优适应度值进行比较，可确定下个粒子的更新策略，如式(3.5)所示[34]。

$$v_{i,j}^{t+1}=\lambda\times[v_{i,j}^{t}+P] \tag{3.3}$$

$$x_{i,j}^{t+1}=x_{i,j}^{t}+v_{i,j}^{t+1} \tag{3.4}$$

$$P=\begin{cases} c_1r_1(pbest_{i-1,j}^{t}-x_{i,j}^{t})+c_2r_2(pbest_{i,j}^{t}-x_{i,j}^{t}) & \\ \quad +c_3r_3(pbest_{i+1,j}^{t}-x_{i,j}^{t}) & if\ fitness(particle_{i,j}^{t})>fitness(gbest_{i,j}^{t-1}) \\ c_4r_4(gbest_{i,j}^{t}-x_{i,j}^{t}) & if\ fitness(particle_{i,j}^{t})\le fitness(gbest_{i,j}^{t-1}) \end{cases} \tag{3.5}$$

式中：r_1、r_2、r_3 及 r_4 是介于 0 和 1 之间的随机数；$v_{i,j}^{t}$ 和 $x_{i,j}^{t}$ 表示速度分量和位置分量；$pbest_{i,j}^{t}$ 是第 t 次迭代下第 i 个粒子在第 j 维度中的个体最优；$gbest_{i,j}^{t}$ 是第 t 次迭代下第 i 个粒子在第 j 维度中的全局最优；λ 是压缩因子；c_1、c_2、c_3 及 c_4 是加速因子。

通过式(3.5)可以看出，若第 t 次迭代的适应度值优于上一代，则应将重点置于局部搜索上，并将参数 c_1、c_2、c_3 设置为 0，即 $c_1=c_2=c_3=0$。相反，若没有找到更好的适应度值，则应保持全局搜索，即 $c_4=0$。λ 和 c 利用以下公式进行更新。

$$\lambda=\frac{2}{\left|(2-c-\sqrt{c^2-4c})\right|} \tag{3.6}$$

$$c = \sum_{i=1}^{4} c_i \tag{3.7}$$

$$c_i = \begin{cases} c_{iM} - (c_{iM} - c_{iN}) \times \dfrac{Iter}{Iter_{\max}}, & i = 1,2,3 \\ c_{iN} + (c_{iM} - c_{iN}) \times \dfrac{Iter}{Iter_{\max}}, & i = 4 \end{cases} \tag{3.8}$$

式中：$Iter$ 指当前迭代次数；$Iter_{\max}$ 表示迭代最大值；c_{iN} 表示初始值，c_{iM} 是最终值，c_{iN}=0.25，c_{iM}=1.05。

APSOIIM 的伪代码在算法 1 中给出：

Algorithm 1: Pseudocode of the proposed APSOIIM

Require: P, λ, c // the control parameters of algorithm
Require: $Iter_{\max}$ // the maximal generation number
Require: N // the population size
Require: M // the dimension of solutions
// Initialization
$Iter$ =0
Initialize the population x randomly
Initialize the velocity v randomly
Chaotic operation of population x
Calculate the fitness value *fitness*
//main loop
while $Iter < Iter_{\max}$ **do**
  for i =1 to N **do**
    if $fitness(particle_{i,j}^{t}) \le fitness(gbest_{i,j}^{t-1})$
      Save current position $x_{i,j}^{t}$
      Update particle velocity by Eq (3.3) and Eq (3.5)
      Update particle position by Eq (3.4)
    else
      Update particle velocity by Eq (3.3) and Eq (3.5)
      Update particle position by Eq (3.4)
    end if
    Check position and velocity range limitations
    Calculate the fitness value *fitness*
    Update new $pbest$, $gbest$
  end for
end while

3.2 收敛性分析

如上所述，通过式(3.3)至式(3.8)可以推导出任意 k 时刻，第 i 个粒子的速度 $v_i^{(k+1)}$、位置 $x_i^{(k+1)}$ 更新公式。

$$v_i^{(k+1)} = \lambda v_i^{(k)} + \lambda c_{i1}^{(k)} r_{i1}^{(k)} (p_{i-1}^{(k)} - x_i^{(k)}) + \lambda c_{i2}^{(k)} r_{i2}^{(k)} (p_i^{(k)} - x_i^{(k)}) + \lambda c_{i3}^{(k)} r_{i3}^{(k)} (p_{i+1}^{(k)} - x_i^{(k)}) + \lambda c_{i4}^{(k)} r_{i4}^{(k)} (g_i^{(k)} - x_i^{(k)}) \tag{3.9}$$

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)} \tag{3.10}$$

式中：$i = 1,2,\cdots,m$；$x_i^{(k)}$、$v_i^{(k)}$分别代表粒子当前位置矢量及速度矢量；$p_{i-1}^{(k)}$为第$i-1$个粒子位置最优；$p_i^{(k)}$为粒子i个体位置最优；$p_i^{(k+1)}$为粒子$i+1$个体位置最优；$g_i^{(k)}$为群体位置最优；$c_{i1}^{(k)}, c_{i2}^{(k)}, c_{i3}^{(k)}, c_{i4}^{(k)}$是加速因子；$r_{i1}^{(k)}, r_{i2}^{(k)}, r_{i3}^{(k)}, r_{i4}^{(k)}$是[0,1]之间的随机数；$\lambda$是收敛因子。

又由于$v_i^{(k)} = x_i^{(k)} - x_i^{(k-1)}$，则结合式(3.9)可以推导出 APSOIIM 算法的迭代方程，具体如式(3.11)所示：

$$x_i^{(k+1)} = x_i^{(k)} + \lambda (x_i^{(k)} - x_i^{(k-1)}) + \lambda c_{i1}^{(k)} r_{i1}^{(k)} (p_{i-1}^{(k)} - x_i^{(k)}) + \lambda c_{i2}^{(k)} r_{i2}^{(k)} (p_i^{(k)} - x_i^{(k)}) + \lambda c_{i3}^{(k)} r_{i3}^{(k)} (p_{i+1}^{(k)} - x_i^{(k)}) + \lambda c_{i4}^{(k)} r_{i4}^{(k)} (g_i^{(k)} - x_i^{(k)}) \tag{3.11}$$

定义 1[35]：设$\{y^{(k)}, k \geq 0\}$为一列数值离散的随机变量，离散变量围成集合为$S = \{j\}$，S为状态空间，对任意$k \geq 1, i^{(l)} \in S \ (l \leq k+1)$有：

$$p(y^{(k+1)} = i^{(k+1)} / y^{(k)} = i^{(k)}, \cdots, y^{(0)} = i^{(0)}) = p(y^{(k+1)} = i^{(k+1)} / y^{(k)} = i^{(k)}) \tag{3.12}$$

则称其为马尔科夫链。

定理 3-1：APSOIIM 中粒子i在k时刻的状态$\varepsilon_i^{(k)}$组成序列$\{\varepsilon_i^{(k)}, k \geq 1\}$为马尔科夫链。

证明：令 APSOIIM 中粒子i在k时刻的状态$\varepsilon_i^{(k)}$为$(x_i^{(k)}, x_i^{(k-1)}, p_{i-1}^{(k)}, p_i^{(k)}, p_{i+1}^{(k)}, g_i^{(k)})$，即$\varepsilon_i^{(k)} = (x_i^{(k)}, x_i^{(k-1)}, p_{i-1}^{(k)}, p_i^{(k)}, p_{i+1}^{(k)}, g_i^{(k)})$，根据式(3.11)可知序列$\{\varepsilon_i^{(k)}, k \geq 1\}$是离散随机变量，且粒子下一时刻的状态$\varepsilon_i^{(k+1)} = (x_i^{(k+1)}, x_i^{(k)}, p_{i-1}^{(k+1)}, p_i^{(k+1)}, p_{i+1}^{(k+1)}, g_i^{(k+1)})$仅与前一时刻的状态$\varepsilon_i^{(k)}$有关，此时，对于任意的$k \geq 1, \varepsilon_i^{(l)} \in S \ (l \leq k+1)$有：

$$p(\varepsilon_i^{(k+1)} / \varepsilon_i^{(k)}, \cdots, \varepsilon_i^{(0)}) = p(\varepsilon_i^{(k+1)} / \varepsilon_i^{(k)}) \tag{3.13}$$

因此定理 3-1 得证。

定理 3-2：当 APSOIIM 算法中加速度因子$\lambda c_{i1}^{(k)} r_{i1}^{(k)}$、$\lambda c_{i2}^{(k)} r_{i2}^{(k)}$、$\lambda c_{i3}^{(k)} r_{i3}^{(k)}$、$\lambda c_{i4}^{(k)} r_{i4}^{(k)} \sim U(0,c)$，粒子$i$由$x_i^{(k)}$转移到$x_i^{(k+1)}$的一步转移概率为：

$$p(\varepsilon_i^{(k+1)} / \varepsilon_i^{(k)}) = \frac{\varepsilon}{\lambda \left\| x_i^{(k)} - x_i^{(k-1)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_{i-1}^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_i^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_{i+1}^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon}{c \left\| g_i^{(k)} - x_i^{(k)} \right\|} \tag{3.14}$$

证明：考虑单个粒子模型，可知$x_i^{(k+1)}$的取值由λ、$\lambda c_{i1}^{(k)} r_{i1}^{(k)}$、$\lambda c_{i2}^{(k)} r_{i2}^{(k)}$、$\lambda c_{i3}^{(k)} r_{i3}^{(k)}$、

$\lambda c_{i4}^{(k)} r_{i4}^{(k)}$ 确定。此时有：

$$\begin{aligned} p(x_i^{(k+1)} / \varepsilon_i^{(k)}) &= \frac{\int_{x_i^{(k+1)}-0.5\varepsilon}^{x_i^{(k+1)}+0.5\varepsilon} dy}{\int_{x_i^{(k)}}^{x_i^{(k)}+w(x_i^{(k)}-x_i^{(k-1)})} dy} \cdot \frac{\int_{\lambda c_{i1}^{(k)} r_{i1}^{(k)}-0.5\varepsilon}^{\lambda c_{i1}^{(k)} r_{i1}^{(k)}+0.5\varepsilon} dy}{\int_{x_i^{(k)}}^{x_i^{(k)}+c(p_{i-1}^{(k)}-x_i^{(k-1)})} dy} \cdot \frac{\int_{\lambda c_{i2}^{(k)} r_{i2}^{(k)}-0.5\varepsilon}^{\lambda c_{i2}^{(k)} r_{i2}^{(k)}+0.5\varepsilon} dy}{\int_{x_i^{(k)}}^{x_i^{(k)}+c(p_i^{(k)}-x_i^{(k-1)})} dy} \\ &\cdot \frac{\int_{\lambda c_{i3}^{(k)} r_{i3}^{(k)}-0.5\varepsilon}^{\lambda c_{i3}^{(k)} r_{i3}^{(k)}+0.5\varepsilon} dy}{\int_{x_i^{(k)}}^{x_i^{(k)}+c(p_{i+1}^{(k)}-x_i^{(k-1)})} dy} \cdot \frac{\int_{\lambda c_{i4}^{(k)} r_{i4}^{(k)}-0.5\varepsilon}^{\lambda c_{i4}^{(k)} r_{i4}^{(k)}+0.5\varepsilon} dy}{\int_{x_i^{(k)}}^{x_i^{(k)}+c(g_i^{(k)}-x_i^{(k-1)})} dy} \\ &= \frac{\varepsilon}{\lambda\left\|x_i^{(k)}-x_i^{(k-1)}\right\|} \cdot \frac{\varepsilon}{c\left\|p_{i-1}^{(k)}-x_i^{(k)}\right\|} \cdot \frac{\varepsilon}{c\left\|p_i^{(k)}-x_i^{(k)}\right\|} \\ &\cdot \frac{\varepsilon}{c\left\|p_{i+1}^{(k)}-x_i^{(k)}\right\|} \cdot \frac{\varepsilon}{c\left\|g_i^{(k)}-x_i^{(k)}\right\|} \end{aligned} \tag{3.15}$$

又因为 $p_{i-1}^{(k+1)}$， $p_i^{(k+1)}$， $p_{i+1}^{(k+1)}$， $g_i^{(k+1)}$ 由 $x_i^{(k+1)}$ 确定，因此粒子 i 从状态 $\varepsilon_i^{(k)}$ 转移到 $\varepsilon_i^{(k+1)}$ 的概率为：

$$p(\varepsilon_i^{(k+1)} / \varepsilon_i^{(k)}) = \frac{\varepsilon}{\lambda\left\|x_i^{(k)}-x_i^{(k-1)}\right\|} \cdot \frac{\varepsilon}{c\left\|p_{i-1}^{(k)}-x_i^{(k)}\right\|} \cdot \frac{\varepsilon}{c\left\|p_i^{(k)}-x_i^{(k)}\right\|} \cdot \frac{\varepsilon}{c\left\|p_{i+1}^{(k)}-x_i^{(k)}\right\|} \cdot \frac{\varepsilon}{c\left\|g_i^{(k)}-x_i^{(k)}\right\|}$$

定义 2[35]：设优化问题的全局最优为 g^*，那么粒子最优状态集为：$M=\left\{\varepsilon^*=(x^{(k)},x^{(k-1)},g^*,g^*,g^*,g^*),\ k\geq 1\right\}$，其中，$\varepsilon^*\in S$ 为粒子最优状态。

定理 3-3：粒子最优状态集 M 是闭集。

证明：根据式(3.5)可知，APSOIIM 算法采取不同的更新策略进行寻优从而保留精英粒子，此时可实现当粒子状态 $\varepsilon_i^{(k)}=(x_i^{(k)},x_i^{(k-1)},p_{i-1}^{(k)},p_i^{(k)},p_{i+1}^{(k)},g_i^{(k)})$ 为最优状态时，其下一时刻状态 $\varepsilon_i^{(k+1)}=(x_i^{(k+1)},x_i^{(k)},p_{i-1}^{(k+1)},p_i^{(k+1)},p_{i+1}^{(k+1)},g_i^{(k+1)})$ 同为最优状态，即若 $\varepsilon_i^{(k)}\in M, k\in[0,T)$，则 $\forall l\in[k+1,T],\ \varepsilon_i^l\in M$。具体表示为：

$$\begin{cases} P(\varepsilon_i^{(k+1)}\in M / \varepsilon_i^{(k)}\in M)=1 \\ P(\varepsilon_i^{(k+1)}\notin M / \varepsilon_i^{(k)}\in M)=0 \end{cases} \tag{3.16}$$

故定理 3-3 得证。

定理 3-4：当粒子 i 在 k 时刻的状态 $\varepsilon_i^{(k)}\notin M$ 时，其下一时刻的状态 $\varepsilon_i^{(k+1)}\notin M$ 的概率为：

$$\begin{aligned} P(\varepsilon_i^{(k+1)}\notin M / \varepsilon_i^{(k)}\notin M) &= 1-\frac{\varepsilon_k}{\lambda\left\|x_i^{(k)}-x_i^{(k-1)}\right\|} \cdot \frac{\varepsilon_k}{c\left\|p_{i-1}^{(k)}-x_i^{(k)}\right\|} \cdot \frac{\varepsilon_k}{c\left\|p_i^{(k)}-x_i^{(k)}\right\|} \\ &\cdot \frac{\varepsilon_k}{c\left\|p_{i+1}^{(k)}-x_i^{(k)}\right\|} \cdot \frac{\varepsilon_k}{c\left\|g_i^{(k)}-x_i^{(k)}\right\|} \end{aligned}$$

式中：当 $g^*\in R,\ \varepsilon_k=\varepsilon$，当 $g^*\notin R,\ \varepsilon_k=0$。

证明：当粒子 i 在 k 时刻的状态 $\varepsilon_i^{(k)} \notin M$ 时，由于加速度因子具有随机性，其下一时刻的位置将存在于一个可行域 R 内，此时全局最优 g^* 的位置和可行域 R 之间的内在联系尚不明确，故无法直接判断粒子 i 在 $k+1$ 时刻的状态 $\varepsilon_i^{(k+1)}$ 是否进入最优状态集。根据定理 3-2，在 k 时刻 $\forall \varepsilon_i^{(k+1)} \in R\ (k \geq 1)$，有：

$$p(\varepsilon_i^{(k+1)} / \varepsilon_i^{(k)}) = \frac{\varepsilon}{\lambda \left\| x_i^{(k)} - x_i^{(k-1)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_{i-1}^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_i^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_{i+1}^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon}{c \left\| g_i^{(k)} - x_i^{(k)} \right\|} \tag{3.17}$$

则：

$$p(\varepsilon_i^{(k+1)} \in M / \varepsilon_i^{(k)} \notin M) = \begin{cases} 0 & g^* \notin R \\ \dfrac{\varepsilon}{\lambda \left\| x_i^{(k)} - x_i^{(k-1)} \right\|} \cdot \dfrac{\varepsilon}{c \left\| p_{i-1}^{(k)} - x_i^{(k)} \right\|} \cdot \dfrac{\varepsilon}{c \left\| p_i^{(k)} - x_i^{(k)} \right\|} \cdot \dfrac{\varepsilon}{c \left\| p_{i+1}^{(k)} - x_i^{(k)} \right\|} \cdot \dfrac{\varepsilon}{c \left\| g_i^{(k)} - x_i^{(k)} \right\|} & g^* \in R \end{cases} \tag{3.18}$$

又因为：

$$P(\varepsilon_i^{(k+1)} \notin M / \varepsilon_i^{(k)} \notin M) = 1 - p(\varepsilon_i^{(k+1)} \in M / \varepsilon_i^{(k)} \notin M) \tag{3.19}$$

则有：

$$P(\varepsilon_i^{(k+1)} \notin M / \varepsilon_i^{(k)} \notin M) = \begin{cases} 1 & g^* \notin R \\ 1 - \dfrac{\varepsilon_k}{\lambda \left\| x_i^{(k)} - x_i^{(k-1)} \right\|} \cdot \dfrac{\varepsilon_k}{c \left\| p_{i-1}^{(k)} - x_i^{(k)} \right\|} \cdot \dfrac{\varepsilon_k}{c \left\| p_i^{(k)} - x_i^{(k)} \right\|} \cdot \dfrac{\varepsilon_k}{c \left\| p_{i+1}^{(k)} - x_i^{(k)} \right\|} \cdot \dfrac{\varepsilon_k}{c \left\| g_i^{(k)} - x_i^{(k)} \right\|} & g^* \in R \end{cases} \tag{3.20}$$

最后可得出：

$$P(\varepsilon_i^{(k+1)} \notin M / \varepsilon_i^{(k)} \notin M) = 1 - \frac{\varepsilon_k}{\lambda \left\| x_i^{(k)} - x_i^{(k-1)} \right\|} \cdot \frac{\varepsilon_k}{c \left\| p_{i-1}^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon_k}{c \left\| p_i^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon_k}{c \left\| p_{i+1}^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon_k}{c \left\| g_i^{(k)} - x_i^{(k)} \right\|} \tag{3.21}$$

式中：当 $g^* \in R,\ \varepsilon_k = \varepsilon$，当 $g^* \notin R,\ \varepsilon_k = 0$。定理 3-4 得证。

上述皆为从单粒子角度对 APSOIIM 算法马氏链进行分析，类似地，可对群体马氏链进行探索。

定义 3[35]：假设 APSOIIM 算法群体状态为$\gamma^{(k)}=(\varepsilon_1^{(k)},\varepsilon_2^{(k)},\cdots,\varepsilon_m^{(k)})$，种群粒子数为$m$，当前迭代次数为$k$。

定理 3-5：APSOIIM 算法种群状态序列$\left\{\gamma^{(k)}=\varepsilon_1^{(k)},\varepsilon_2^{(k)},\cdots,\varepsilon_m^{(k)},k\geq 1\right\}$为马尔科夫链。

证明：令 APSOIIM 算法种群状态序列$\left\{\gamma^{(k)},k\geq 1\right\}$的状态空间为$\Lambda$，对于任意的$k\geq 1,\gamma^{(l)}\in\Lambda\ (l\leq k+1)$，有：

$$p(\gamma^{(k+1)}/\gamma^{(k)},\cdots,\gamma^{(1)})=p(\varepsilon_1^{(k+1)},\cdots,\varepsilon_m^{(k+1)}/\varepsilon_1^{(k)},\cdots,\varepsilon_m^{(k)},\cdots\varepsilon_1^{(1)},\cdots,\varepsilon_m^{(1)}) \tag{3.22}$$

根据定理 3-1 可知，$p(\varepsilon^{(k+1)}/\varepsilon^{(k)},\cdots,\varepsilon^{(1)})=p(\varepsilon^{(k+1)}/\varepsilon^{(k)})$，因此：

$$p(\gamma^{(k+1)}/\gamma^{(k)},\cdots,\gamma^{(1)})=p(\varepsilon_1^{(k+1)},\cdots,\varepsilon_m^{(k+1)}/\varepsilon_1^{(k)},\cdots,\varepsilon_m^{(k)})=p(\gamma^{(k+1)}/\gamma^{(k)}) \tag{3.23}$$

故定理 3-5 得证。

定义 4[35]：假设 APSOIIM 群体状态为$\gamma^{(k)}=(\varepsilon_1^{(k)},\varepsilon_2^{(k)},\cdots,\varepsilon_m^{(k)})$，定义群体最优状态为$\gamma_k^*$，对于$\gamma_k^*=(\varepsilon_1^{(k)},\varepsilon_2^{(k)},\cdots,\varepsilon_m^{(k)})$，$\exists i\in[1,m]$，使得$\varepsilon_i^{(k)}\in M$群体最优状态集$\Gamma=\left\{\gamma_k^*,k\geq 1/\gamma_k^*=(\varepsilon_1^{(k)},\varepsilon_2^{(k)},\cdots,\varepsilon_m^{(k)})\right\}$。

定理 3-6：APSOIIM 算法种群最优状态集Γ为闭集。

证明：由定理 3-3 可知M为闭集，即：

$$P(\varepsilon_i^{(k+1)}\in M/\varepsilon_i^{(k)}\in M)=1 \tag{3.24}$$

定义 4 已表明若在k时刻群体状态$\gamma_k^*=(\varepsilon_1^{(k)},\varepsilon_2^{(k)},\cdots,\varepsilon_m^{(k)})\in\Gamma$，即意味着对于群体状态$\gamma^{(k)}$，$\exists i\in[1,m]$，使得$\varepsilon_i^{(k)}\in M$。那么在$k+1$时刻必有$\varepsilon_i^{(k+1)}\in M$，因此在$k+1$时刻群体状态$\gamma^{(k+1)}=(\varepsilon_1^{(k+1)},\varepsilon_2^{(k+1)},\cdots,\varepsilon_m^{(k+1)})$，$\exists i\in[1,m]$，使得$\varepsilon_i^{(k+1)}\in M$，因此$\gamma^{(k+1)}\in\Gamma$，即：

$$\begin{cases}P(\gamma^{(k+1)}\in\Gamma/\gamma^{(k)}\in\Gamma)=1\\ P(\gamma^{(k+1)}\notin\Gamma/\gamma^{(k)}\in\Gamma)=0\end{cases} \tag{3.25}$$

则定理 3-6 得证。

定理 3-7：APSOIIM 算法在$k+1$时刻的群体状态$\gamma^{(k+1)}\in\Gamma$的概率为：

$$\begin{aligned}P(\gamma^{(k+1)}\in\Gamma)=1-p(\gamma^{(1)}\notin\Gamma)\cdot\prod_{l=1}^{k}\prod_{i=1}^{m}(1-\frac{\varepsilon_l}{\lambda\left\|x_i^{(l)}-x_i^{(l-1)}\right\|}\cdot\frac{\varepsilon_l}{c\left\|p_{i-1}^{(l)}-x_i^{(l)}\right\|}\cdot\\ \frac{\varepsilon_l}{c\left\|p_i^{(l)}-x_i^{(l)}\right\|}\cdot\frac{\varepsilon_l}{c\left\|p_{i+1}^{(l)}-x_i^{(l)}\right\|}\cdot\frac{\varepsilon_l}{c\left\|g_i^{(l)}-x_i^{(l)}\right\|})\end{aligned} \tag{3.26}$$

证明：根据全概率公式，APSOIIM 算法中群体在$k+1$时刻的状态不在最优状态集的$\gamma^{(k+1)}\notin\Gamma$的概率为：

$$\begin{aligned} p(\gamma^{(k+1)} \notin \Gamma) = p(\gamma^{(k+1)} \notin \Gamma / \gamma^{(k)} \notin \Gamma) \cdot p(\gamma^{(k)} \notin \Gamma) \\ + p(\gamma^{(k+1)} \notin \Gamma / \gamma^{(k)} \in \Gamma) \cdot p(\gamma^{(k)} \in \Gamma) \end{aligned} \tag{3.27}$$

因 $\gamma^{(k+1)} \notin \Gamma / \gamma^{(k)} \in \Gamma = 0$，则：

$$\begin{aligned} p(\gamma^{(k+1)} \notin \Gamma) &= p(\gamma^{(k+1)} \notin \Gamma / \gamma^{(k)} \notin \Gamma) \cdot p(\gamma^{(k)} \notin \Gamma) \\ &= p(\gamma^{(k+1)} \notin \Gamma / \gamma^{(k)} \notin \Gamma) \cdot p(\gamma^{(k)} \notin \Gamma / \gamma^{(k-1)} \notin \Gamma) \\ &\quad \cdot p(\gamma^{(k-1)} \notin \Gamma / \gamma^{(k-2)} \notin \Gamma) \cdots\cdots p(\gamma^{(2)} \notin \Gamma / \gamma^{(1)} \notin \Gamma) \\ &= p(\gamma^{(1)} \notin \Gamma) \cdot \prod_{l=1}^{k} \gamma^{(l+1)} \notin \Gamma / \gamma^{(l)} \notin \Gamma \end{aligned} \tag{3.28}$$

又由于 $\gamma^{(k+1)} \notin \Gamma$ 表示 $\forall i \in [1, m]$，不存在 $\varepsilon_i^{(k+1)} \in M$，则：

$$\begin{aligned} p(\gamma^{(k+1)} \notin \Gamma / \gamma^{(k)} \notin \Gamma) &= \prod_{i=1}^{m} p(\varepsilon_i^{(k+1)} \notin M / \varepsilon_i^{(k)} \notin M) \\ &= \prod_{i=1}^{m} (1 - \frac{\varepsilon_k}{\lambda \left\| x_i^{(k)} - x_i^{(k-1)} \right\|} \cdot \frac{\varepsilon_k}{c \left\| p_{i-1}^{(k)} - x_i^{(k)} \right\|} \\ &\quad \cdot \frac{\varepsilon_k}{c \left\| p_i^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon_k}{c \left\| p_{i+1}^{(k)} - x_i^{(k)} \right\|} \cdot \frac{\varepsilon_k}{c \left\| g_i^{(k)} - x_i^{(k)} \right\|}) \end{aligned} \tag{3.29}$$

所以：

$$p(\gamma^{(k+1)} \notin \Gamma) = \prod_{l=1}^{k} \prod_{i=1}^{m} (1 - \frac{\varepsilon_l}{\lambda \left\| x_i^{(l)} - x_i^{(l-1)} \right\|} \cdot \frac{\varepsilon_l}{c \left\| p_{i-1}^{(l)} - x_i^{(l)} \right\|} \cdot \frac{\varepsilon_l}{c \left\| p_i^{(l)} - x_i^{(l)} \right\|} \cdot \frac{\varepsilon_l}{c \left\| p_{i+1}^{(l)} - x_i^{(l)} \right\|} \cdot \frac{\varepsilon_l}{c \left\| g_i^{(l)} - x_i^{(l)} \right\|}) \cdot p(\gamma^{(1)} \notin \Gamma)$$

最后得出种群在 $k+1$ 时刻的状态 $\gamma^{(k+1)} \in \Gamma$ 的概率为：

$$\begin{aligned} P(\gamma^{(k+1)} \in \Gamma) &= 1 - P(\gamma^{(k+1)} \notin \Gamma) \\ &= 1 - p(\gamma^{(1)} \notin \Gamma) \cdot \prod_{l=1}^{k} \prod_{i=1}^{m} (1 - \frac{\varepsilon_l}{\lambda \left\| x_i^{(l)} - x_i^{(l-1)} \right\|} \cdot \frac{\varepsilon_l}{c \left\| p_{i-1}^{(l)} - x_i^{(l)} \right\|} \\ &\quad \cdot \frac{\varepsilon_l}{c \left\| p_i^{(l)} - x_i^{(l)} \right\|} \cdot \frac{\varepsilon_l}{c \left\| p_{i+1}^{(l)} - x_i^{(l)} \right\|} \cdot \frac{\varepsilon_l}{c \left\| g_i^{(l)} - x_i^{(l)} \right\|}) \end{aligned} \tag{3.30}$$

在 APSOIIM 算法中，由于某些时刻可以使得最优位置 $g* \in R$，且参数 λ, c 满足条件，此时有 $j \in \{1, 2, 3, \cdots n\}$，使得：

$$\begin{aligned} p(\varepsilon_i^{(kj+1)} \in M / \varepsilon_i^{(kj)} \notin M) = \frac{\varepsilon}{|\lambda| \left\| x_i^{(kj)} - x_i^{(kj-1)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_{i-1}^{(kj)} - x_i^{(kj)} \right\|} \cdot \frac{\varepsilon}{c \left\| p_i^{(kj)} - x_i^{(kj)} \right\|} \\ \cdot \frac{\varepsilon}{c \left\| p_{i+1}^{(kj)} - x_i^{(kj)} \right\|} \cdot \frac{\varepsilon}{c \left\| g_i^{(kj)} - x_i^{(kj)} \right\|} > \delta \end{aligned} \tag{3.31}$$

式中：δ 为一个趋于 0 的值。

对其进行变形，式(3.31)可被表示为：

$$1-\frac{\varepsilon}{|\lambda|\left\|x_i^{(kj)}-x_i^{(kj-1)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_{i-1}^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_i^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_{i+1}^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|g_i^{(kj)}-x_i^{(kj)}\right\|}<1-\delta \tag{3.32}$$

因此有：

$$\prod_{j=1}^{\infty}(1-\frac{\varepsilon}{|\lambda|\left\|x_i^{(kj)}-x_i^{(kj-1)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_{i-1}^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_i^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_{i+1}^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|g_i^{(kj)}-x_i^{(kj)}\right\|})=0 \tag{3.33}$$

当$k\to\infty$，则有：

$$\begin{aligned}p(\gamma^{(k+1)}\in\Gamma)&=1-p(\gamma^{(1)}\notin\Gamma)\cdot\prod_{l=1}^{k}\prod_{i=1}^{m}(1-\frac{\varepsilon_l}{\lambda\left\|x_i^{(l)}-x_i^{(l-1)}\right\|}\cdot\frac{\varepsilon_l}{c\left\|p_{i-1}^{(l)}-x_i^{(l)}\right\|}\cdot\frac{\varepsilon_l}{c\left\|p_i^{(l)}-x_i^{(l)}\right\|}\cdot\frac{\varepsilon_l}{c\left\|p_{i+1}^{(l)}-x_i^{(l)}\right\|}\cdot\frac{\varepsilon_l}{c\left\|g_i^{(l)}-x_i^{(l)}\right\|})\\&=1-p(\gamma^{(1)}\notin\tau)\cdot\prod_{j=1}^{n}\prod_{i=1}^{m}(1-\frac{\varepsilon}{|\lambda|\left\|x_i^{(kj)}-x_i^{(kj-1)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_{i-1}^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_i^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|p_{i+1}^{(kj)}-x_i^{(kj)}\right\|}\cdot\frac{\varepsilon}{c\left\|g_i^{(kj)}-x_i^{(kj)}\right\|})\\&=1\end{aligned} \tag{3.34}$$

由式(3.34)知当$n\to\infty$时，$\lim_{n\to\infty}p(\gamma^{(k+1)}\in\tau)=1$，说明所提算法全局收敛。

3.3 仿真验证及实际应用

3.3.1 仿真验证

为了验证交互信息机制的有效性，采用 CEC2014、CEC2017 测试集中常见单峰函数、多峰函数及离散函数作为测试函数，将所提算法与 PSO、SPSO、APSO、PSOCF、AIWPSO 及 SLFPSO 算法进行对比实验以验证各算法的寻优性能，具体函数介绍如表 3-1 所示。此外，为了确保对比实验的准确性，所涉及的测试函数分别固定在 10 维和 30 维。在所有比较算法中，种群大小设置为 40，迭代终止标准为 1000，评估总数为 1000×40=40000，其余详细参数见表 3-2。此举将确保算法不会增加额外计算量，并对所有算法保持公平，避免偏差。全部实验均在系统配置为 Windows 11 处理器英特尔酷睿 i7-12700 CPU、@2.10 GHz、16 GB RAM 和 MATLAB(R2021a)的软件上进行。

表 3-1 常见基准测试功能概述

Name	Test function	Domain	Acceptable precision1	Acceptable precision2	Type
Sphere(F_1)	$F_1=\sum_{i=1}^{N}x_i^2$	[-100,100]	0.01	1E-06	Unimodal
*Schwefel*2.22(F_2)	$F_2=\sum_{i=1}^{N}\lvert x_i\rvert+\prod_{i=1}^{N}\lvert x_i\rvert$	[-10,10]	0.01	1E-06	
Rosenbrock($F3$)	$F_3=\sum_{i=1}^{N}[100(x_{i+1}-x_i^{2})^2+(x_i-1)^2]$	[-10,10]	10	5	
Sumpower($F4$)	$F_4=\sum_{i=1}^{N}\lvert x_i\rvert^{i+1}$	[-10,10]	0.01	1E-06	
SumSquares($F5$)	$F_5=\sum_{i=1}^{N}ix_i^2$	[-10,10]	0.01	1E-06	
Zakharov($F6$)	$F_6=\sum_{i=1}^{N}x_i^2+(\sum_{i=1}^{N}0.5x_i)^2+(\sum_{i=1}^{N}0.5x_i)^4$	[-10,10]	0.01	1E-06	
Ackley($F7$)	$F_7=-20\exp(-0.2\sqrt{1/N\sum_{i=1}^{N}x_i^{2}})-\exp(1/N\sum_{i=1}^{N}\cos(2\pi x_i))+20+e$	[-32,32]	10	1	Multimodal
Rastrigin($F8$)	$F_8=\sum_{i=1}^{N}(x_i^2-10\cos(2\pi x_i)+10)$	[-5.12,5.12]	10	5	
Girewank($F9$)	$F_9=1/4000\sum_{i=1}^{N}x_i^{2}-\prod_{i=1}^{N}\cos(x_i/\sqrt{i})+1$	[-600,600]	0.01	1E-06	
Levy($F10$)	$F_{10}=\sin^2(\pi w_1)+\sum_{i=1}^{N-1}(w_i-1)^2[1+10\sin^2(\pi w_i+1)]+(w_N-1)^2[1+\sin^2(2\pi w_N)]$	[-10,10]	0.01	1E-06	
Quadric($F11$)	$F_{11}=\sum_{i=1}^{N}(\sum_{j=1}^{i}x_j)$	[-100,100]	0.01	1E-06	
Noncontinuous *–Rastrigin*($F12$)	$F_{12}=\sum_{i=1}^{N}(y_i^{2}-10\cos(2\pi y_i)+10)+\sum_{i=1}^{N}u(x_i,10,100,4)$	[-5.12,5.12]	10	5	Discrete

注：在 $F12$，$y_i=\begin{cases}x_i & \lvert x_i\rvert\le 0.5\\ round(2x_i)/2 & \lvert x_i\rvert\ge 0.5\end{cases}$，$u(x_i,a,k,m)=\begin{cases}k(x_j-a)^m & x_j>a\\ 0 & -a\le x_j\le a\\ k(-x_j-a)^m & x_j<-a\end{cases}$

表 3-2 算法参数设置

Algorithms	Parameter settings	References
PSO	$N=40,\ c_1=1.05,\ c_2=1.05$	文献[36]
SPSO	$N=40,\ c_1=1.05,\ c_2=1.05,\ w_{\max}=0.9,\ w_{\min}=0.4$	文献[37]
APSO	$N=40,\ c_1=1.05,\ c_2=1.05$	文献[38]
PSOCF	$N=40,\ c_1=2.05,\ c_2=2.05$	文献[39]
AIWPSO	$N=40,\ w_{\max}=0.9,\ w_{\min}=0.1$	文献[40]
SLFPSO	$N=40,\ c=1.49445,\ w_{\max}=0.9,\ w_{\min}=0.4,\ Tpre=10$	文献[41]
APSOIIM	$N=40,\ c_{iN}=0.25,\ c_{iM}=1.05$	本文方法

1）搜索精度测试

为对比各算法的寻优精度，分别进行 10 维和 30 维下的搜索精度测试。值得一提的是，表中数据均为经过 50 次运行后所得平均值，具体见表 3-3 和表 3-4。

表 3-3 10 维测试结果统计

Function	Iteration	PSO	SPSO	APSO	PSOCF	AIWPSO	SLFPSO	APSOIIM
F_1	1000	1.53E-02	1.23E-02	8.99E-05	5.76E-06	3.84E-07	1.58E-07	**6.83E-08**
F_2	1000	9.36E-02	9.22E-02	1.42E-01	1.11E-01	8.92E-03	**4.02E-04**	8.61E-03
F_3	1000	8.09E+00	7.98E+00	5.11E+00	6.64E+00	9.89E+00	2.06E+00	**1.39E+00**
F_4	1000	1.71E-02	3.36E-03	1.47E-03	3.47E-04	3.68E-04	**3.01E-05**	4.31E-04
F_5	1000	1.03E-02	6.31E-03	1.03E-04	3.49E-04	5.36E-05	7.00E-04	**4.26E-05**
F_6	1000	1.31E-02	2.57E-02	1.58E-03	6.10E-02	**7.21E-05**	7.05E+00	2.36E-02
F_7	1000	2.18E+00	2.24E+00	1.41E+00	1.63E+00	2.27E+01	3.30E+00	**1.63E-01**
F_8	1000	1.44E+01	1.52E+01	1.43E+01	1.26E+01	2.98E+01	5.05E+01	**4.64E+00**
F_9	1000	1.50E-03	3.40E-03	6.89E-05	1.08E-06	2.20E-03	3.39E-05	**4.53E-07**
F_{10}	1000	1.57E-01	1.27E-01	1.09E-01	9.38E-02	2.34E-01	5.63E-02	**3.61E-03**
F_{11}	1000	6.01E-04	1.83E-03	1.70E-07	1.59E-05	2.33E-06	1.71E-05	**1.46E-07**
F_{12}	1000	1.04E+01	2.68E+01	9.21E+00	1.06E+01	3.18E+01	8.05E+02	**7.06E+00**

注: F_1 到 F_6 是单峰函数， F_7 到 F_{11} 是多峰函数， F_{12} 是离散函数。

表 3-4 30 维测试结果统计

Function	Iteration	PSO	SPSO	APSO	PSOCF	AIWPSO	SLFPSO	APSOIIM
F_1	1000	1.19E+02	6.50E+01	7.59E+00	4.93E+00	9.67E-07	8.77E-07	**6.71E-07**
F_2	1000	2.86E+00	3.05E+00	3.06E+00	2.74E+00	7.92E+00	**3.66E-02**	5.03E+00
F_3	1000	6.03E+01	6.19E+01	8.15E+01	4.56E+01	1.89E+01	2.96E+01	**1.02E+01**
F_4	1000	4.55E-02	3.49E-02	7.71E-05	1.52E-05	**3.68E-07**	3.01E-05	6.71E-04
F_5	1000	4.89E+00	4.38E+00	6.34E+00	4.71E+00	5.36E+00	7.00E-02	**1.63E-03**
F_6	1000	1.15E+00	1.09E+00	1.92E+00	1.56E+00	**7.21E-01**	7.05E+00	9.89E+00
F_7	1000	1.29E+02	3.25E+01	7.03E+01	3.39E+01	3.01E+01	8.35E+00	**5.31E+00**
F_8	1000	1.01E+02	8.81E+01	1.11E+02	9.16E+01	9.98E+01	5.84E+01	**3.06E+00**
F_9	1000	1.71E-01	1.93E-01	3.09E-01	2.07E-01	2.20E-01	1.60E-02	**3.98E-05**
F_{10}	1000	1.15E+00	1.28E+00	1.52E+00	1.36E+00	5.34E-01	5.00E+01	**3.46E-01**
F_{11}	1000	4.62E+01	3.16E+01	7.51E+01	4.00E+01	2.33E+00	1.71E+01	**7.03E-06**
F_{12}	1000	5.32E+01	7.04E+01	9.06E+01	7.32E+01	5.05E+01	4.05E+01	**3.18E+00**

注: F_1 到 F_6 是单峰函数， F_7 到 F_{11} 是多峰函数， F_{12} 是离散函数。

从表 3-3 中可以看出，当迭代次数固定为 1000 时，在 F_4 和 F_6 中，所提出的算法精度略低于对比算法。而在所有剩余函数上，APSOIIM 算法搜索精度表现优异。值得一提的是，对于像 F_3 、 F_8 和 F_9 这样具有多个局部最优的测试函数，

APSOIIM 仍然可以取得优异结果，这表明该算法作为交互信息机制和压缩因子策略优势的高效结合，有效地平衡了探索和开发。

此外，表 3-4 显示出 APSOIIM 算法将测试维度扩展为 30 维时的寻优性能。经分析，可以看出所提算法在多峰和离散函数中性能优越。众所周知，多峰函数具有多个局部最优，对算法要求更严格，往往需要鲁棒性强的算法，其优秀性能足以有效证明 APSOIIM 算法处理复杂问题的能力。

2）与目前主流算法的对比测试

为了验证所提算法的有效性，将 APSOIIM 算法与当前主流算法(蚂蚁狮优化器[42](ALO)、回溯搜索优化算法[43](BSA)、GSA[44]、全局引力搜索算法[45](GGSA)、改进引力搜索算法[46](IGSA)及协方差矩阵自适应进化策略算法[47](CMA-ES))进行对比实验，结果详见表 3-5。

表 3-5 30 维度下 APSOIIM 算法与当前主流算法对比结果统计

Function	Iteration	ALO	BSA	GSA	GGSA	IGSA	CMA-ES	APSOIIM
F_1	1000	6.21E-06	9.97E+00	2.97E-11	2.78E-18	1.60E-17	**1.42E-18**	6.71E-07
F_2	1000	6.32E+00	1.20E+00	5.09E-06	**9.53E-08**	1.57E-07	2.98E-07	5.03E+00
F_3	1000	2.88E+01	4.72E+02	4.35E+01	3.71E+01	5.22E+01	3.68E+01	**1.02E+01**
F_4	1000	1.23E-04	6.86E-04	3.68E-05	4.01E-06	**2.88E-06**	7.17E-05	6.71E-04
F_5	1000	4.49E-03	2.54E+02	1.36E-02	3.63E-03	7.30E-02	7.49E-03	**1.63E-03**
F_6	1000	3.21E+01	8.21E-01	2.12E-06	**3.43E-07**	7.12E-05	5.07E-05	9.89E+00
F_7	1000	2.17E+01	5.13E+01	8.75E+00	**3.51E+00**	4.33E+00	1.56E+01	5.31E+00
F_8	1000	7.71E+01	6.58E+01	8.93E+01	3.31E+01	3.54E+01	2.53E+01	**3.06E+00**
F_9	1000	2.50E-03	1.07E+00	4.52E+00	7.28E+00	1.49E-02	**5.76E-15**	3.98E-05
F_{10}	1000	9.98E+00	7.54E-01	1.57E+00	6.57E-01	2.73E+00	3.57E+00	**3.46E-01**
F_{11}	1000	1.08E+03	2.72E+03	3.24E+02	4.77E+02	1.35E+03	1.59E-05	**7.03E-06**
F_{12}	1000	3.15E+02	1.05E+02	7.11E+01	3.00E+01	1.51E+01	6.50E+01	**3.18E+00**

注: F_1 到 F_6 是单峰函数，F_7 到 F_{11} 是多峰函数，F_{12} 是离散函数。

从表 3-5 可知，与目前主流算法相比，APSOIIM 算法在求解复杂优化问题方面发挥出色，能够有效平衡算法勘探与开采能力，降低 PSO 算法陷入局部最优概率，验证所提算法的高效性。

3）与 CEC2013-CEC2018 获奖者的比较

在本节中，将 APSOIIM 算法与 CEC 系列的其他优秀算法进行比较，即 iCMAES-ILS[48]（IEEE CEC 2013 中排名第二）、SHADE[49]（IEEE CEC 2013 中排名第一的非 CMA-ES 变体）、LSHADE[50]（CEC 2014 竞赛获胜者）、

EBOwithCMAR[51]（IEEE CEC 2017 获胜者）、CJADE[52]（高级 DE 之一）、HSES[53]（IEEE CEC 2018 获胜者）和 iLSHADE-RSP[54]（先进的 DEs 之一）。种群大小和最大迭代次数分别设置为 40 和 1000，实验结果见表 3-6。

表 3-6 APSOIIM 与 CEC2013-CEC2018 获奖者的对比结果统计

Function	Metric	iCMAES-ILS	SHADE	LSHADE	EBO with CMAR	CJADE	HSES	iLSHADE-RSP	APSOIIM
F_1	Mean	7.02E-28	1.01E-28	1.53E-30	4.04E-18	1.31E-22	**2.31E-36**	3.56E-32	6.71E-07
	Rank	5	4	3	7	6	**1**	2	8
F_2	Mean	3.09E-08	4.51E-07	**3.78E-10**	2.02E-09	3.53E-08	3.21E-09	1.05E-09	5.03E+00
	Rank	5	7	**1**	3	6	4	2	8
F_3	Mean	**3.56E+00**	8.00E+01	3.21E+01	1.54E+01	1.20E+01	5.66E+00	3.91E+01	1.02E+01
	Rank	**1**	8	6	5	4	2	7	3
F_4	Mean	3.21E-06	4.54E-05	**1.54E-06**	1.92E-06	3.41E-04	4.75E-04	5.01E-06	6.71E-04
	Rank	3	5	**1**	2	6	7	4	8
F_5	Mean	**2.45E-04**	6.15E-03	1.33E-02	3.36E-02	5.79E-03	3.99E-03	1.77E-03	1.63E-03
	Rank	**1**	6	7	8	5	4	3	2
F_6	Mean	1.50E+01	2.13E+01	3.08E+00	**4.12E-07**	3.13E-03	7.67E-02	4.15E+01	9.89E+00
	Rank	6	7	4	**1**	2	3	8	5
F_7	Mean	1.43E-02	2.03E-04	7.51E-04	3.73E-05	2.94E-03	**1.33E-08**	8.24E-03	5.31E+00
	Rank	7	3	4	2	5	**1**	6	8
F_8	Mean	6.31E+01	1.61E+01	1.43E+01	1.57E+01	9.08E+00	3.87E+01	2.37E+01	**3.06E+00**
	Rank	8	5	3	4	2	7	6	**1**
F_9	Mean	1.17E-03	1.53E-04	6.98E-05	3.40E-03	8.04E+00	**3.78E-08**	1.02E-02	3.98E-05
	Rank	5	4	3	6	8	**1**	7	2
F_{10}	Mean	4.02E-01	6.09E+00	3.90E-01	7.46E-01	4.77E-02	6.02E-01	2.01E+00	3.46E-01
	Rank	4	8	3	6	1	5	7	2
F_{11}	Mean	3.29E-05	2.11E-02	1.83E-04	8.93E-01	2.31E-04	1.05E-04	1.26E-05	**7.03E-06**
	Rank	3	7	5	8	6	4	2	**1**
F_{12}	Mean	4.25E+00	7.51E+01	6.44E+00	5.39E+00	9.87E+00	1.17E+01	3.32E+00	**3.18E+00**
	Rank	3	8	5	4	6	7	2	**1**
/	AvgRank1	5.00	5.83	3.83	5.00	4.67	4.17	5.00	**2.51**
	Rank	5	6	2	5	4	3	5	**1**

注：AvgRank1 表示多峰函数的排名。

表 3-6 结果显示，除了 F_7 外的所有多峰函数，APSOIIM 均排名第一，说明所提算法具有优越的抗干扰能力，能够有效解决复杂优化问题。

4）成功概率测试

针对精度对比测试仅能反映算法在大量测试样本下的求优能力，无法精确显示单次情况的问题，引入了成功概率测试。该方法能够有效评估算法求解优化问题的能力及算法随机性程度，验证算法在复杂环境下是否具有抗干扰能力，表

3-7 中给出成功概率的具体符号。

表 3-7 成功概率符号列表

Probability of success(P)	Representative Symbols
P >80%	#S
50%<P <80%	#PS
P <50%	#NS

表 3-8 呈现出所有对比算法的成功概率评估结果。

表 3-8 成功概率实验结果

Function	Iteration	Category	SPSO	APSO	PSOCF	AIWPSO	SLFPSO	APSOIIM
F_1	1000	1	#PS	#S	#S	#S	#S	#S
		2	#NS	#S	#S	#S	#S	#S
F_2	1000	1	#PS	#NS	#NS	#S	#S	#PS
		2	#S	#S	#S	#S	#PS	#S
F_3	1000	1	#S	#S	#S	#S	#PS	#S
		2	#PS	#PS	#PS	#PS	#NS	#S
F_4	1000	1	#S	#S	#S	#S	#S	#S
		2	#S	#PS	#S	#PS	#PS	#S
F_5	1000	1	#S	#S	#S	#S	#PS	#S
		2	#NS	#S	#S	#PS	#NS	#PS
F_6	1000	1	#PS	#PS	#S	#S	#PS	#S
		2	#NS	#NS	#S	#NS	#PS	#PS
F_7	1000	1	#S	#S	#S	#PS	#PS	#S
		2	#NS	#PS	#PS	#NS	#NS	#S
F_8	1000	1	#NS	#PS	#PS	#NS	#NS	#S
		2	#NS	#NS	#NS	#NS	#NS	#S
F_9	1000	1	#S	#S	#S	#S	#S	#S
		2	#NS	#S	#S	#PS	#PS	#S
F_{10}	1000	1	#NS	#NS	#NS	#S	#S	#S
		2	#NS	#PS	#NS	#PS	#NS	#PS
F_{11}	1000	1	#S	#S	#S	#S	#S	#S
		2	#NS	#S	#S	#S	#PS	#S
F_{12}	1000	1	#PS	#PS	#S	#NS	#NS	#S
		2	#NS	#NS	#NS	#NS	#NS	#PS

注：可接受的精度如表 3-1 所示，符号表示详见表 3-7

采用各种算法分别获得成功概率统计结果如表 3-9 所示。

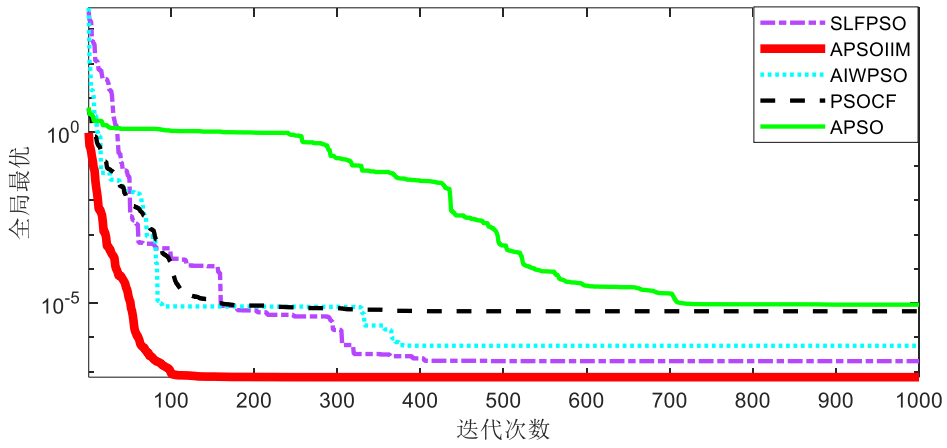
表 3-9 成功概率统计结果

Symbols	SPSO	APSO	PSOCF	AIWPSO	SLFPSO	APSOIIM
#NS	11	5	5	6	8	**0**
#PS	**5**	7	3	6	9	**5**
#S	8	12	16	12	7	**19**

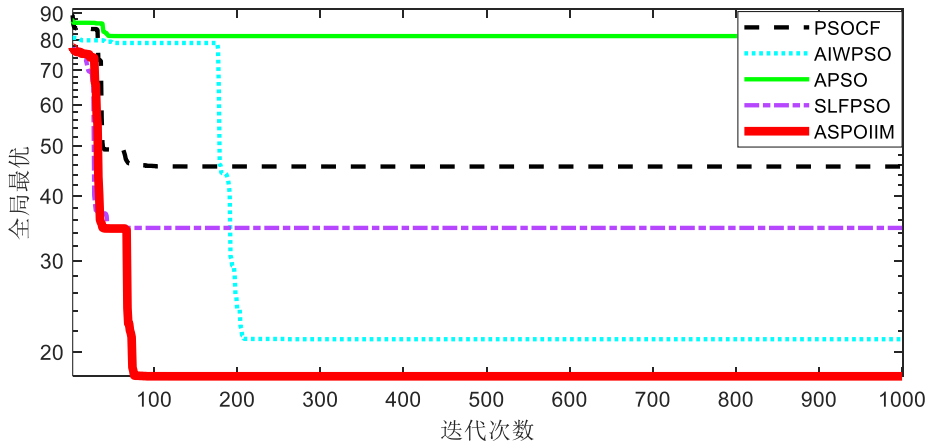
通过分析表 3-9 中的数据，可以明显看出，APSOIIM 共获得 19 次#S 类别，0 次#NS 类别。相比之下，其他对比算法似乎都略显“劣质”。此外，对于#PS 类别，提高可接受精度标准意味着对算法的更高要求，本质上要求启发式算法在指定数量的试验中出现更多优秀测试结果，即使条件严格，APSOIIM 仍表现出优越性能。

5）收敛曲线对比

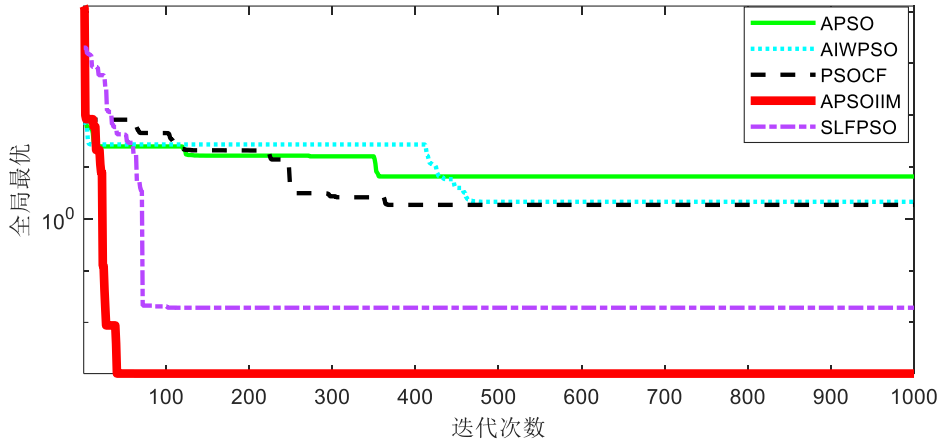
为验证 APSOIIM 算法在单峰函数上的有效性，绘制出所有算法的适应度值迭代曲线，具体如图 3-4 所示。



(a) 各算法函数 F_1 收敛曲线



(b) 各算法函数 F3 收敛曲线

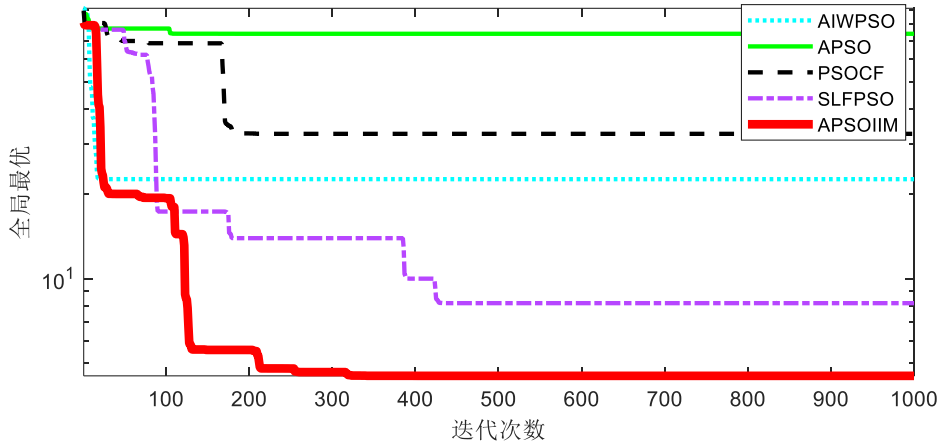


(c) 各算法函数 F5 收敛曲线

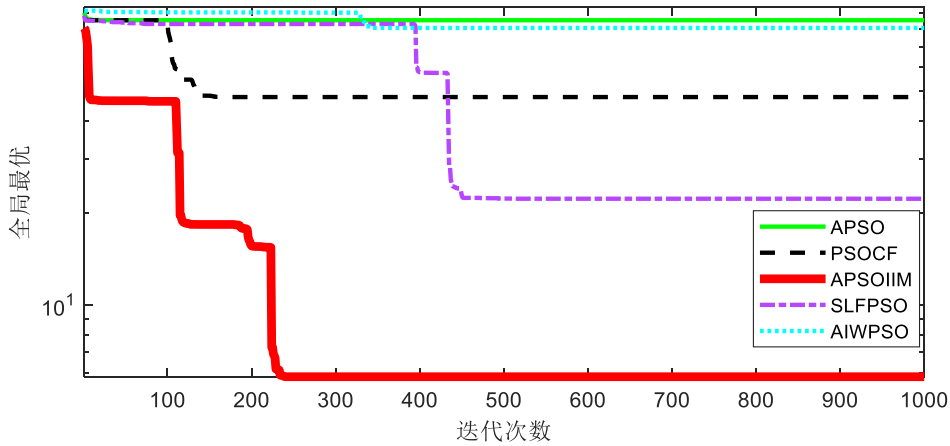
图 3-4 单峰函数的收敛曲线

从图 3-4 中可知，对于 F_1、F_3、F_5，APSOIIM 算法在迭代初期得到的最优解优于其他算法，说明初始化策略在一定程度上提高了算法的寻优能力，此外所提算法在单峰函数中始终实现最高精度，显示出其强大的搜索性能。

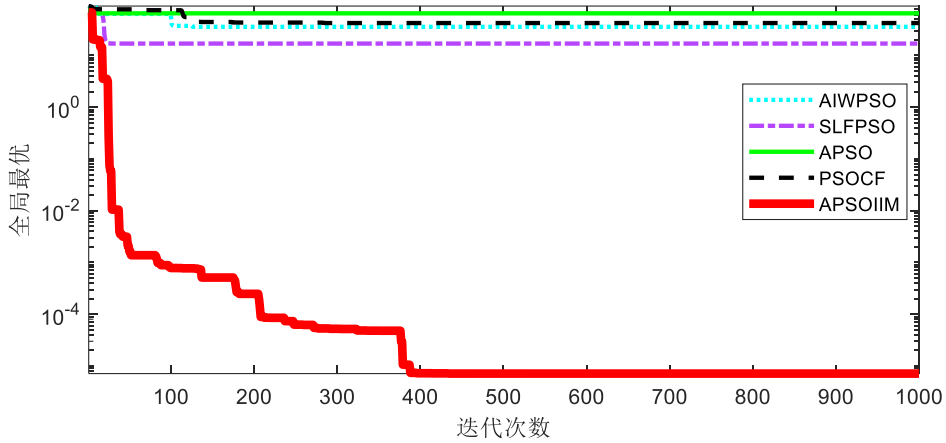
APSOIIM 算法在多峰函数下的寻优效果如图 3-5 所示：



(a) 各算法函数 F_7 收敛曲线



(b) 各算法函数 F8 收敛曲线



(c) 各算法函数 F_{11} 收敛曲线

图 3-5 多峰函数的收敛曲线

通过分析图 3-5，可以发现对于 F_8、F_{11}，在迭代过程中观察到，APSO 和 PSOCF 的收敛速度和搜索精度似乎有所欠缺，对比算法在 50 次迭代时收敛曲线较慢明显突出了此点。相比之下，该阶段中 APSOIIM 算法收敛曲线几乎垂直，有效地证明所提算法具有较强的勘探能力。对于 F_7，虽然 APSOIIM 算法前期寻优速度并不占优，但其始终实现最高收敛精度，显示出其强大的搜索性能。

此外，为了验证所提算法对于离散函数的寻优性能，将 APSOIIM 与 APSO、PSOCF、AIWPSO 及 SLFPSO 算法进行对比实验，效果如图 3-6 所示：

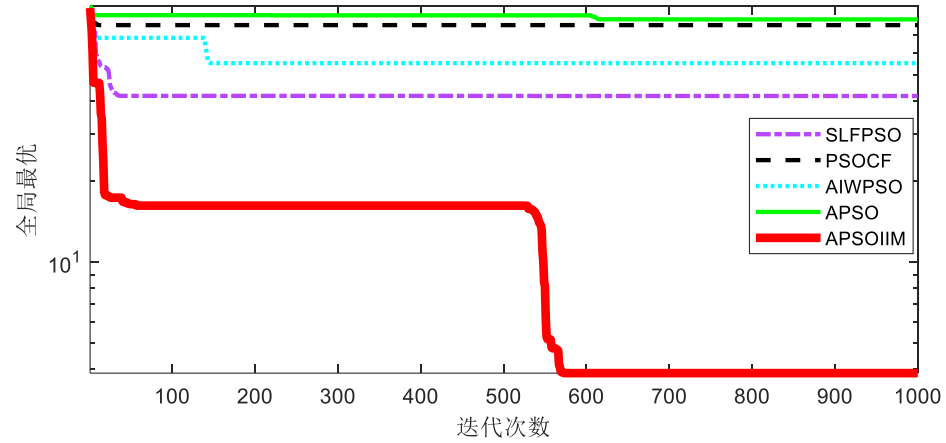


图 3-6 离散函数的收敛曲线

从图 3-6 中可知，APSOIIM 算法在离散函数寻优中表现优异。从前期迭代角度来看，与 APSO、PSOCF、AIWPSO 及 SLFPSO 对比算法相比，所提算法收敛曲线几乎垂直下降，展现其优越的收敛速度；其次从图 3-6 中可以发现在迭代过程中，所提算法能够有效跳出局部最优；最后在迭代后期 APSOIIM 算法收敛精度遥遥领先，验证其优越的抗干扰能力及鲁棒性。

6）勘探-开发分析

为了进一步提高所提算法的说服力，检验 APSOIIM 算法是否保持临界平衡，引入了勘探-开发实验，在数学上，其分析准则如下所示：

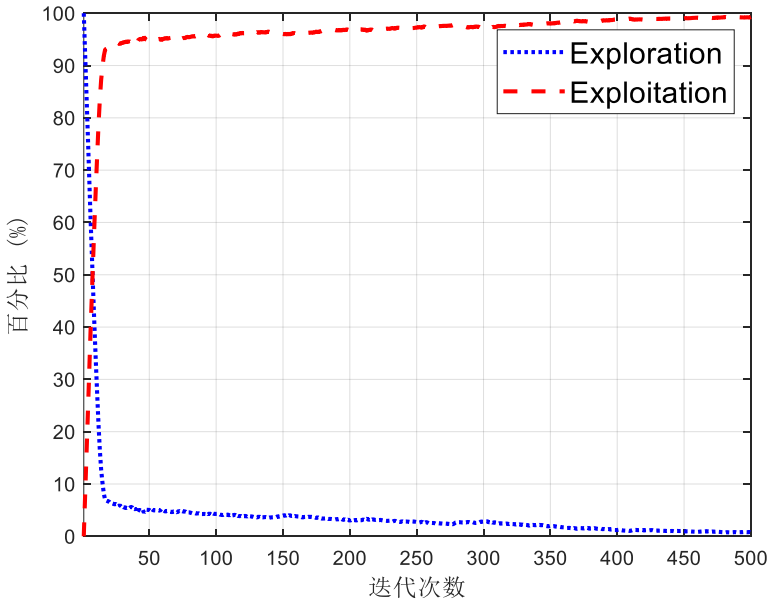
$$Exploration(\%)=\frac{Div(t)}{Div_{\max}}\times 100 \tag{3.35}$$

$$Exploitation(\%)=\frac{\left|Div(t)-Div_{\max}\right|}{Div_{\max}}\times 100 \tag{3.36}$$

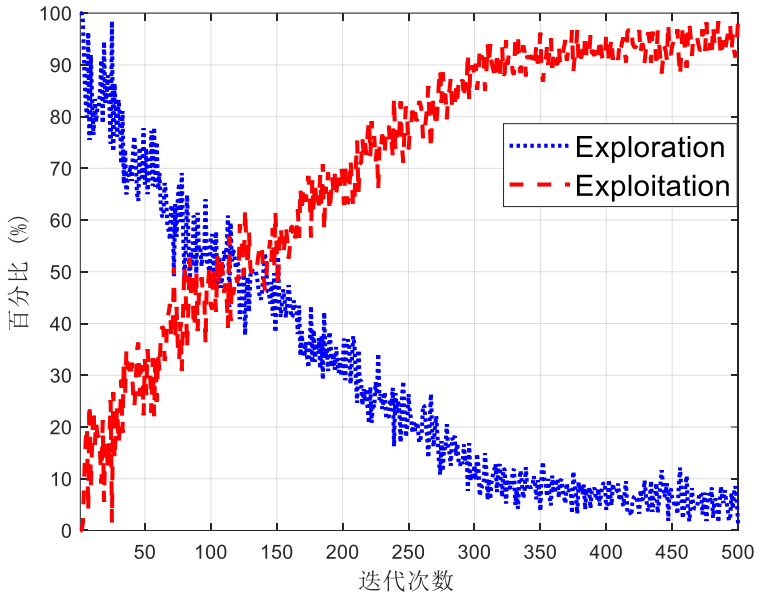
$$Div(t)=\frac{1}{D}\sum_{d=1}^{D}\frac{1}{N}\sum_{i=1}^{N}\left|median(x_d(t))-x_{id}(t)\right| \tag{3.37}$$

式中：$Div_{\max}$ 为整个迭代种群的最大多样性，参数 x_{id} 表示第 t 次迭代时第 i 个个体的第 d 维的值。

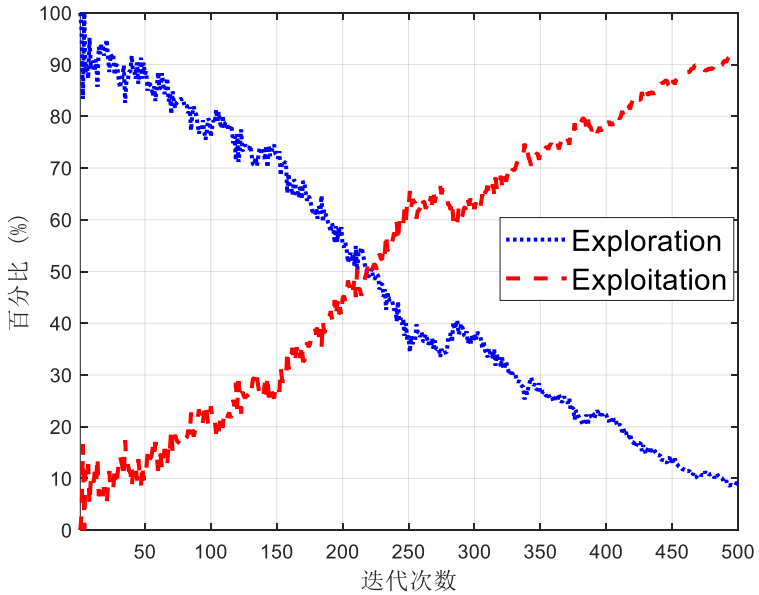
为了检验所提出的算法是否在勘探和开发之间保持合理平衡，从测试函数中选择了三种类型的优化问题，具体来说是 F_3，F_9 和 F_{12}。如图 3-7 所示，其中百分比表示种群的勘探和开发水平，从单峰函数 F_3 可以看出，开发比例迅速增加，说明算法具有较好的局部搜索能力，而从多峰函数 F_9 可以看出，勘探比例下降缓慢，说明算法具有较好的全局探索能力。



(a) 各算法函数 F_3 勘探-开采曲线



(b) 各算法函数 F_9 勘探-开采曲线



(c) 各算法函数 F_{12} 勘探-开采曲线

图 3-7 单峰函数 F_3，多峰函数 F_9 和离散函数 F_{12} 的勘探开发曲线

3.3.2 实际应用

为了验证所提算法在实际中的可行性，应用一种基于约束违反的简单约束处理技术集中解决含约束的实际工程优化问题。该技术主要基于约束违反对两种解决方案进行比较，并利用 Deb 可行性规则选择两种解决方案。全部实验均在系统配置为 Windows 11 处理器英特尔酷睿 i7-12700 CPU、@2.10 GHz、16 GB RAM 和 MATLAB（R2021a）的软件上进行。

对于正则优化问题，对应于解 x 的约束违反 $viol_x$ 可以通过式(3.38)计算：

$$viol_{\hat{x}} = \sum_{j=1}^{J} G_j(x) + \sum_{k=1}^{K} H_k(x) \tag{3.38}$$

$$G_j(x) = \begin{cases} g_j(x) & if\ \ G_j(x) > 0 \\ 0 & else \end{cases} \tag{3.39}$$

$$H_k(x) = \begin{cases} |h_k(x)| & if\ \ |h_k(x)| - \in > 0 \\ 0 & else \end{cases} \tag{3.40}$$

式中：$\in$是预定义的公差参数，在以下工程应用中采用固定值10^{-4}。

1）齿轮系设计问题

齿轮系设计问题是一个典型的离散优化问题，示意图如图 3-8 所示。该问题共涉及四个决策变量x_1、x_2、x_3和x_4，这些决策变量代表一列火车中四个齿轮的齿数，找到最优齿数（决策变量），然后最小化齿数比即为该问题的目标。在数学层面上的表述如下：

$$\begin{aligned} Min\quad & f_1(x) = \left(\frac{1}{6.931} - \frac{x_2 x_3}{x_1 x_4}\right)^2 \\ s.t.\quad & x = (x_1, x_2, x_3, x_4) \\ & 12 \le x_1, x_2, x_3, x_4 \le 60 \end{aligned} \tag{3.41}$$

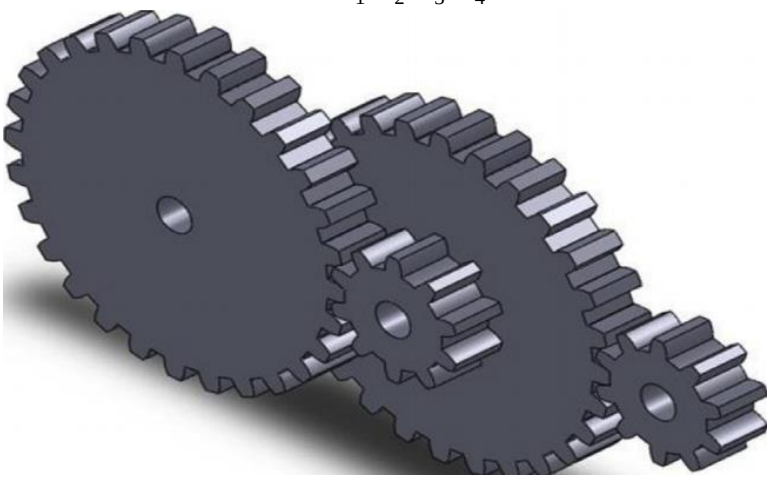


图 3-8 齿轮系设计问题

表 3-10 齿轮系设计问题的结果比较

Algorithm	Optimal decision variables				Optimum value	Max function evaluations
	X_1	X_2	X_3	X_4		
ISCA	43	16	19	49	**2.7009E-12**	700
SCA	52	25	12	40	2.3576E-09	700
MBA	43	16	19	49	**2.7009E-12**	10000
CS	43	16	19	49	**2.7009E-12**	5000
APSOIIM	43	16	19	49	**2.7009E-12**	**100**

为了验证所提算法的高效性，将 APSOIIM 算法与正弦余弦算法、布谷鸟搜索算法、地雷爆破算法和改进正弦余弦算法进行对比实验。根据表 3-10 所示，APSOIIM 算法能够通过最少的迭代次数实现与 CS、MBA 及 ISCA 相同的寻优结果 2.7009E-12，说明 APSOIIM 算法收敛速度快。

2）伸/缩弹簧设计问题

该问题目标是寻找受剪切应力、挠度、浪涌频率、外径和设计变量约束的伸/缩弹簧的最小值。此类问题通常被认为是含约束的最优问题，其中共涉及线圈直径 x_1、平均线圈直径 x_2 及有效线圈匝数 x_3 三个决策变量，其数学描述为：

$$\begin{aligned} &Min\ \ f_2(x)=(x_3+2)x_2{x_1}^2 \\ &s.t.\ \ g_1(x)=1-\frac{{x_2}^3x_3}{71785{x_1}^4}\le 0 \\ &\qquad g_2(x)=\frac{4{x_2}^2-x_1x_2}{12566({x_1}^3x_2-{x_1}^4)}+\frac{1}{5108{x_1}^2}-1\le 0 \\ &\qquad g_3(x)=1-\frac{140.45x_1}{{x_2}^2x_3}\le 0 \\ &\qquad g_4(x)=\frac{x_1+x_2}{1.5}-1\le 0 \\ &\qquad 0.05\le x_1\le 2, 0.25\le x_2\le 1.30, 2\le x_3\le 15 \end{aligned} \tag{3.42}$$

表 3-11 压缩管柱设计问题的结果比较

Algorithm	Optimal decision variables			Optimum value
	X_1	X_2	X_3	
GA	0.15148	0.35166	11.63220	0.012705
GSA	0.05028	0.32368	13.52541	0.012702
PSO	0.05388	0.35119	13.96804	0.014241
APSOIIM	0.05165	0.35615	11.34035	**0.012675**

为了验证各算法的收敛精度，对伸/缩弹簧设计实际问题进行求解。该问题已通过一系列元启发式算法 GA、GSA 得到解决，对于涉及到的所有算法，最大迭代次数均取 1000。由表 3-11 可知，APSOIIM 算法寻优最值为 0.012675，与 GSA、GA 等算法相比寻优精度显著提高，表明所提算法的准确性。

3）悬臂梁设计问题

该问题通常被视为具有约束的实际工程问题，其目的是最小化带有方形横截面悬臂梁的重量。如图 3-9 所示，梁被支撑在节点 1 处，并施加力作用在节点 5 上。该问题共涉及五个决策变量，分别表示不同光束的高度或宽度，其数学表达

式具体见式(3.43)：

$$\begin{aligned} Min \quad & f_3(x) = 0.0624 \times \sum_{i=1}^{5} x_i \\ s.t. \quad & \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \le 1 \\ & 0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100 \end{aligned} \tag{3.43}$$

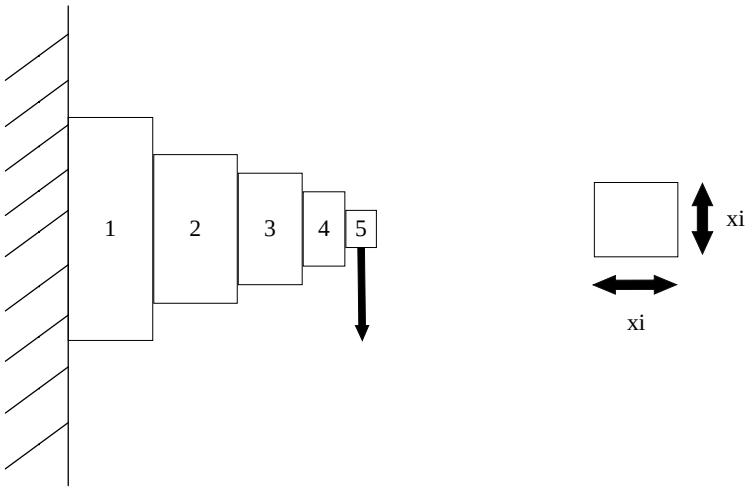


图 3-9 悬臂梁设计问题

表 3-12 悬臂梁设计问题的结果比较

Algorithm	Optimal decision variables					Optimum value
	X_1	X_2	X_3	X_4	X_5	
m-SCA	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999
SCA	6.0100	5.3000	4.4900	3.4900	2.1500	1.34000
CS	6.0089	5.3049	4.5023	3.5077	2.1504	1.33999
GCA(II)	6.0100	5.3000	4.4900	3.4900	2.1500	1.34000
CONLIN	6.0100	5.3000	4.4900	3.4900	2.1500	1.34000
APSOIIM	6.0131	5.3008	4.4929	3.5097	2.1573	**1.33997**

为了最小化悬臂梁重量，在相同数量的评估次数 1000×40=40000 条件下，将 APSOIIM 与 m-SCA、SCA、CS、GCA(II)及 CONLIN 算法进行对比，所获得的最佳结果见表 3-12。从表中数据可知，APSOIIM 算法最优值为 1.33997，寻优精度最高，能够有效解决悬臂梁设计问题。

4）Himmelblau’s 非线性优化问题

Himmelbla 的非线性优化问题被视为具有约束的实际工程问题，它包含 5 个优化变量和 6 个非线性约束。在限制范围内搜索最佳值是本算法的目标，数学模型如下：

$$
\begin{aligned}
Min\quad & f_4(x)=5.3578547X_3^{\,2}+0.8356891X_1X_5+37.293239X_1-40792.141\\
s.t.\quad & g_1(x)=85.334407+0.0056858X_2X_5+0.0006262X_1X_4-0.0022053X_3X_5\\
& g_2(x)=80.51249+0.0071317X_2X_5+0.0029955X_1X_2-0.0021813X_3^{\,2}\\
& g_3(x)=9.300961+0.0047026X_3X_5+0.0012547X_1X_3+0.0019085X_3X_4\\
& 78\le X_1\le 102\\
& 33\le X_2\le 45\\
& 27\le X_3,X_4,X_5\le 45\\
& 0\le g_1(x)\le 92\\
& 90\le g_2(x)\le 110\\
& 20\le g_3(x)\le 25
\end{aligned}
\tag{3.44}
$$

表 3-13 Himmelblau 非线性优化问题的比较结果

Algorithms	Optimal decision variables					Optimum value
	X_1	X_2	X_3	X_4	X_5	
GPSO	78.01	33.01	30.03	44.94	36.72	-30658.18
OPSO	78.76	33.37	30.81	44.56	34.91	-30472.52
NOA	78.05	33.01	30.02	44.98	36.73	-30658.72
APSOIIM	78.00	33.00	29.99	45.00	36.78	**-30666.95**

为了验证 APSOIIM 算法收敛精度，如表 3-13 所示，将所提算法与高斯粒子群优化、正交粒子群优化和胡桃夹子优化算法等算法进行对比实验。根据表 3-13 可以发现，与其他算法相比，APSOIIM 算法的最优值为-30666.95，更接近实际最优值且变量满足参数约束。

5）尺蠖机器人逆运动学

尺蠖机器人逆运动学研究属于新颖的约束型实际工程应用，旨在实现获得位置与期望位置之间的距离最小，实现精确拟合，其参数表示每个模块的关节角。尺蠖机器人的链接长度为 0.091，模拟的预期点为(0.5,0.5)。该问题具体数学模型表达式如下：

$$
\begin{gathered}
y=[y_1,y_2,y_3,y_4]=[\theta_1,\theta_2,\theta_3,\theta_4]\\
f(y)=\sqrt{((x_d-\sum_{i=1}^{n}L_i\cos(\sum_{j=1}^{i}\theta_j)))^2+(y_d-(\sum_{i=1}^{n}L_i\sin(\sum_{j=1}^{i}\theta_j)))^2}\\
where\quad 0\le y_1,y_2,y_3,y_4\le 360.
\end{gathered}
\tag{3.45}
$$

式中：(x_d,y_d) 是所需点，L_i 是链接的长度。

表 3-14 尺蠖机器人逆运动学对比结果

Algorithms	Optimal decision variables				Optimum value	Mean	STD
	θ_1	θ_2	θ_3	θ_4			
Rao	45	360	0	0	**3.4311E-01**	3.8456E-01	3.5383E-02
CLRAO	45	360	0	0	**3.4311E-01**	3.5232E-01	1.6861E-02
APSOIIM	45	360	0	0	**3.4311E-01**	**3.4410E-01**	**5.6601E-03**

为了验证 APSOIIM 算法寻优稳定性，对尺蠖机器人逆运动学问题进行求解。从表 3-14 可以看出，几种算法找到的最优值均为 3.4311E-01，一定程度上表明算法寻求最优值的能力是相似的，但通过比较“Mean”和“STD”，发现所提算法均方差最小，足以说明 APSOIIM 具有优越的抗干扰能力。

3.4 基于 APSOIIM 算法的微电网系统单目标调度

如上所述，在第二章中已分别构建了两种优化调度模型，以实现微电网系统运行费用最小或污染物治理成本最低。各模型所涉及到的分布式电源参数包括出力功率、运维系数及维护成本等，具体如表 3-15 所示[33]。

表 3-15 各分布式电源的参数设置

电源类型	MT	DE	WT	PV
额定功率(kw)	1000	800	500	500
出力功率(kw)	0-1000	0-800	0-480	0-480
运维系数	0.0473	0.0836	0.2098	0.0092
维护成本(元/kWh)	0.06	0.04	0.12	0.08

其它参数设定为：与电网交互功率为$\pm 150kW/h$；蓄电池容量为$600kWh$，其充放电效率为90%，充放电功率为$80kW$。

1）系统运行费用最小的优化调度

优化调度是指通过协调各分布式电源的输出，以实现减少运行成本、提高设备利用率的目的。当仅考虑系统运行费用最小时，综合考虑分布式电源的发电成本和与大电网的交互情况，优化 DE、MT、WT、PV、BESS 及 grid 出力至关重要，具体如图 3-10 所示。

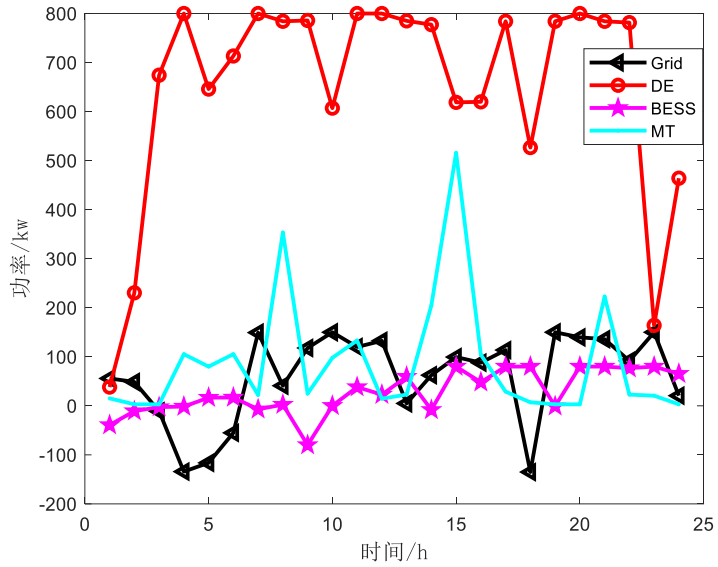


图 3-10 考虑系统运行费用最小的出力调度图

由图 3-10 可知，该情况下发电成本较低的 DE 是“主力军”。在 1:00-3:00 时段，鉴于此时电价较低且负荷需求不高的情况，除了 DE 及 WT 正常出力以外，选择从大电网进行购电从而避免 MT 发电，且蓄电池进行充电操作，以备不时之需。在 4:00-6:00 时段，此时 MT、DE 及 BESS 共同出力，剩余电量进行售电操作，补贴收益。从 7:00 开始，太阳出现，PV 开始陆续地进行出力，整个系统处于相对平衡状态。上午 10:00 至 12:00，居民开始正常的生产、生活，电力需求量显著上升，在 12:00 时，电力需求达到高峰，DE 出力已趋于饱和，此时 WT、PV、BESS、MT 及 Grid 纷纷加入，以满足负荷需求。在 12:00-16:00，负载量显著减少，DE 逐渐下降出力概率，该举有助于延长设备寿命，减少维修费用。在 16:00-18:00，PV、DE、MT、BESS、WT 共同出力以满足负荷需求，并剩余有大量电量，此时进行售电操作，补贴收益。在 19:00-22:00 时段，晚间用电高峰来临，多个分布式电源同时出力，此时为确保微电网安全稳定运行，微电网必须向大电网进行购电操作。22:00 后，用电需求量出现显著下降，这时电价降低，DE 和 MT 降低输出功率，进入维护期，WT、BESS 及 Grid 三者协同工作，确保负荷的稳定供应。

2）污染物治理成本最低的优化调度

为了降低微电网系统的污染物治理成本，可以通过调度各设备的出力情况来降低污染气体治理费用。具体如图 3-11 所示。

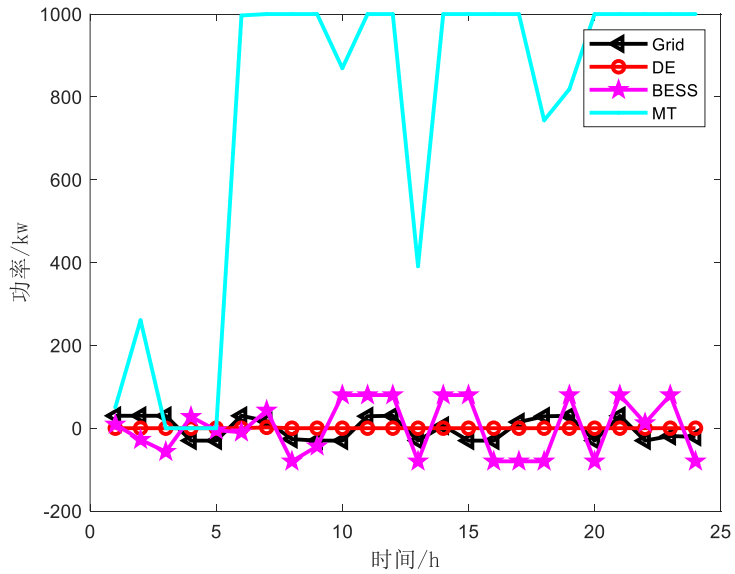


图 3-11 考虑污染物治理成本最低的出力调度图

在 0:00-5:00，WT 率先进行出力，为减少不必要的 MT 启停成本，不足的电量通过 BESS 及 Grid 相互协调得到满足，且对环境极其不友好的柴油发电机不出力。在 6:00-9:00，MT 接近满发状态，DE 依旧保持不出力以减少环境污染，此时对于逐渐增加的负荷需求，WT、PV 及 MT 保持满发，BESS 尽可能进入充电模式，以备后续不时之需，Grid 承担辅助角色以保证居民生产生活。在 10:00-12:00，临近中午时段，负荷量出现大幅度增长，此时 MT、PV、WT 及 BESS 处于满发状态，考虑到环境保护，Grid 相对于 DE 更加有利，不足电量进行大电网购电操作。在 13：00，此时系统负荷出现急剧下降，MT 减少发电以维护设备，BESS 进入充电模式，剩余电量进行售电操作，补贴收益。在 14:00-16:00，除了 MT、PV、WT 稳定出力外，BESS 尽可能进行放电操作，在满足负荷需求的情况下，补贴收益，保护环境的同时降低成本。在 17:00-21:00，进入晚高峰时段，此时 MT、PV、WT 及 BESS 大幅度出力，不足电量部分同样采用大电网购电的方式维护生态环境，减少污染物排放量。在 22:00-0:00，负荷明显降低，BESS 将剩余电量放至大电网，实现一定的经济补偿后进入充电状态。

3.5 本章小结

本章首先利用混沌初始化策略产生均匀分布的粒子，加快收敛速度，然后，引入交互信息机制增加种群的多样性，实现与邻近粒子个体最优的有效相互，在探索和开发之间取得平衡；其次，对所提 APSOIIM 算法进行理论收敛证明；之

后，将所提算法与 PSO、SPSO、APSO、PSOCF 及 SAIW-PSO 等算法进行对比实验，包括 CEC2014 和 CEC2017 基准函数以及著名的工程优化问题。实验结果表明，APSOIIM 的综合性能优于对比算法；最后，基于新型 PSO 算法对微电网单目标优化问题进行调度，结果表明 APSOIIM 算法能够有效寻找最优解，降低能耗。未来，APSOIIM 将用于解决不同研究方向的优化问题，如约束优化问题、整数规划问题等。

第 4 章 基于 HPSOFF 算法的微电网多目标优化调度

在实际微电网系统中，仅考虑单目标优化调度是不合适的，常常“顾此失彼”。微电网系统优化调度问题通常表现为多属性且不同优化目标之间可能存在相互关联或矛盾的情况，因此研究微电网多目标优化调度具有重大意义。针对多目标优化问题的复杂性，本章探讨了一种高效的混合粒子群优化与萤火虫算法，其中结合非线性惯性权重粒子群算法(Particle Swarm Optimization with Nonlinear Inertia Weight and Attenuation Factor, PSO-NIWAF)和动态种群萤火虫算法(Firefly Algorithm of Dynamic Population, FFA-DP)来克服其各自算法的局限性以实现精准寻优，其流程图如图 4-1 所示。

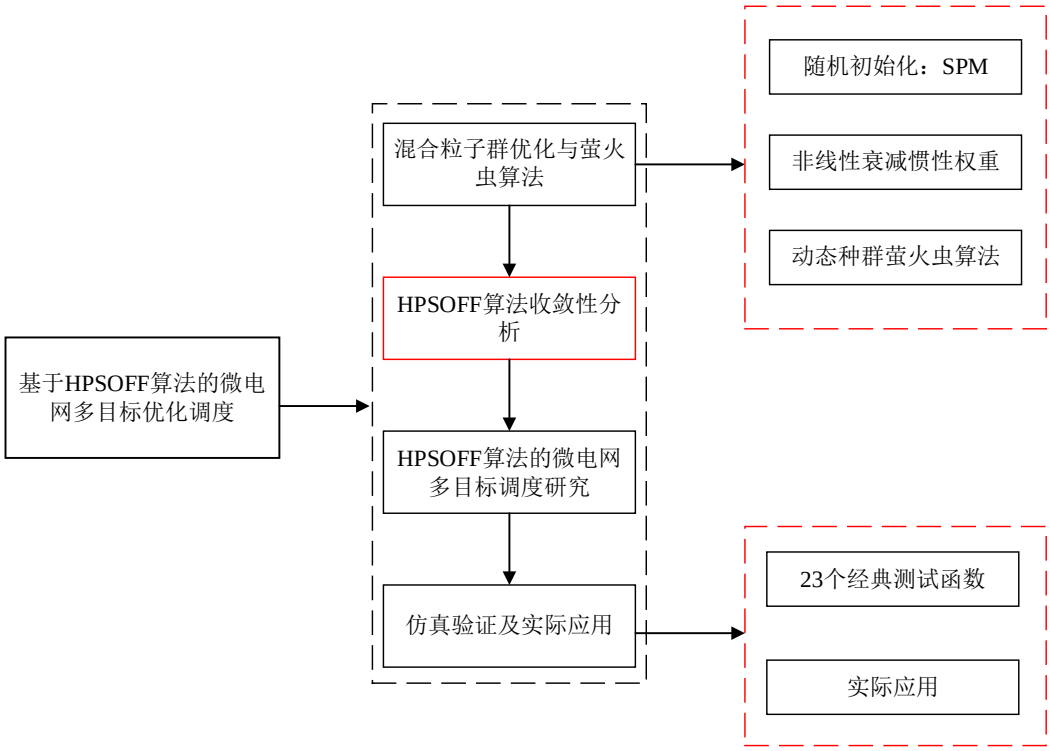


图 4-1 HPSOFF 算法流程图

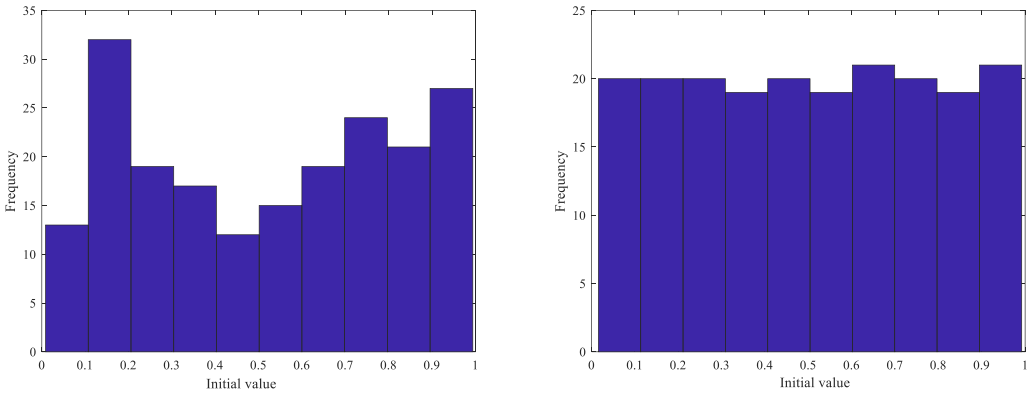
4.1 混合粒子群优化和萤火虫算法

4.1.1 具有非线性衰减因子惯性权重的粒子群算法

针对不同的初始化方式会影响算法优化性能[55]的问题，引入了一种称为随机参数映射(SPM)的混沌初始化方法，其数学模型如式(4.1)所示。

$$x(t+1)=\begin{cases} mod(\dfrac{x(t)}{\eta}+\mu sin(\pi x(t))+r,1) & 0\le x(t)<\eta \\ mod(\dfrac{x(t)/\eta}{0.5-\eta}+\mu sin(\pi x(t))+r,1) & \eta\le x(t)<0.5 \\ mod(\dfrac{(1-x(t))/\eta}{0.5-\eta}+\mu sin(\pi(1-x(t)))+r,1) & 0.5\le x(t)<1-\eta \\ mod(\dfrac{(1-x(t))}{\eta}+\mu sin(\pi(1-x(t)))+r,1) & 1-\eta\ \le x(t)\le 1 \end{cases} \tag{4.1}$$

式中：$u=0.3$，$\eta=0.4$；r 是混沌系统的扰动系数；$x(t)$ 是第 t 个粒子的位置；$x(t+1)$ 表示经混沌操作后的第 t 个粒子的位置。



(a).随机初始化　　(b).混沌初始化

图 4-2 粒子分布直方图

图 4-2 表明，SPM 方法产生的粒子分布均匀性优于随机方法，证明了混沌初始化方法的有效性。另一方面，惯性权重 w 表示粒子与前一刻相比保持其运动状态的能力。在现有文献中，惯性权重通常采用线性递减的方式更新。但在实际的优化过程中，前期粒子需要较大的 w 来实现全局勘探，后期粒子需要较小的 w 来进行精细开发。显然，线性迭代法无法满足实际需求。为此，提出了一种基于非线性衰减因子惯性权重的粒子群优化（PSO-NIWAF），其中采用非线性衰减因子惯性权重（NIWAF）策略来平衡算法探索和开发，具体如式(4.2)和(4.3)所示[56]。

$$k(t)=\begin{cases} 1 & t=1 \\ \dfrac{Stdfit(t)}{Stdfit(t-1)} & t>1 \end{cases} \tag{4.2}$$

$$w=w_{\max}+(w_{\min}-w_{\max})\times\frac{1}{1+\exp[-10b(\dfrac{2t}{k(t)\cdot T}-1)]} \tag{4.3}$$

式中：w 为惯性权重；c_1 为加速度因子；t 为当前迭代次数；T 为最大迭代次数；

b 为[0,1]之间的随机数； $Stdfit(t)$ 为第 t 代种群适应度值的标准差； $v_{i,j}^{t}$ 、 $x_{i,j}^{t}$ 、 $pbest_{i,j}^{t}$ 、及 $gbest_{i,j}^{t}$ 分别是速度分量、位置分量、局部最优以及全局最优。

从图 4-3 中可以看出，与线性递减法相比，所提算法可以始终保持较高的惯性权值，从而高效率地实现全局优化，为后续萤火虫算法进行精细搜索奠定基础。

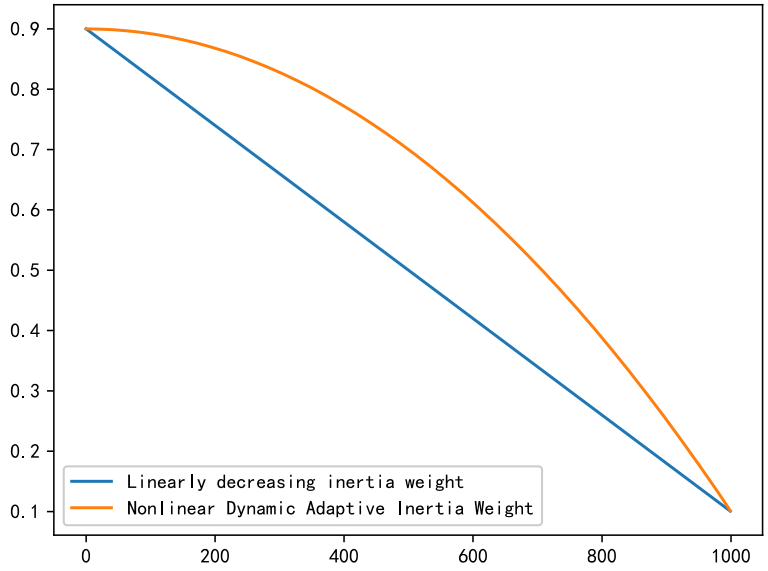


图 4-3 惯性权重趋势图

4.1.2 动态种群萤火虫算法

与 PSO 相比，FFA 算法不受速度限制且局部开发效果良好，但其挣脱局部最优能力弱。为了解决这一缺陷，研究了一种动态种群萤火虫算法(FFA-DP)，采用动态种群策略更新种群粒子并构建临时种群，达到丰富种群多样性效果，具体见式(4.4)至式(4.7)。

$$r_{ij}=\left\|X_i^{t}-X_j^{t}\right\|=\sqrt{(x_i^{t}-x_j^{t})^2-(y_i^{t}-y_j^{t})^2} \tag{4.4}$$

$$I=I_0e^{-\gamma r_{ij}^{2}} \tag{4.5}$$

$$\beta=\beta_0e^{-\gamma r_{ij}^{2}} \tag{4.6}$$

$$X_i^{t+1}=X_i^{t}+\beta_0e^{-\gamma r_{ij}^{2}}(x_j^{t}-x_i^{t})+\alpha\varepsilon_i^{t} \tag{4.7}$$

式中： r_{ij} 为两个粒子之间的范式距离； t 为当前迭代次数； I_0 为自身亮度； γ 为吸收系数； β_0 为最大吸引度； $\alpha\varepsilon_i^{t}$ 为随机项。

如图 4-4 所示，在产生新一代后，将候选种群 X2 和原始种群 X1 组合形成临时新种群 X3，并对种群粒子的适应度值进行排序；然后，选择具有最佳适应度值的数量等同于原始种群的粒子，构成下一次迭代的初始种群。此方式有效利用 FFA 算法的局部开发能力，避免陷入“局部陷阱”。

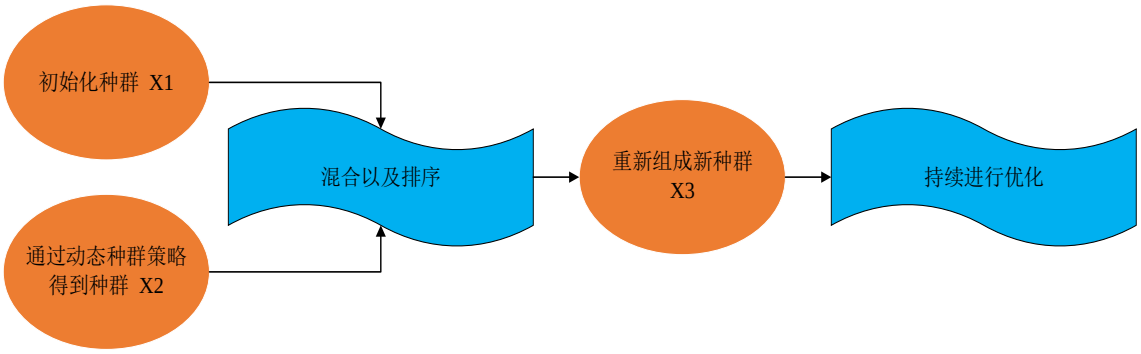


图 4-4 动态种群策略流程图

4.1.3 HPSOFF 算法混合优化步骤

粒子群算法在全局探索中表现出优异的性能，但容易陷入局部最优。相反，FFA 算法没有速度特性，可以提供出色的局部开发，但通常难以跳出局部搜索。因此本章提出了 HPSOFF 算法。在迭代的前半部分，利用 PSO-NIWAF 的高效全局探索来确定最优解的近似范围；随后，在临时库内比较每次迭代所得到的最优解，并选择全局最优解为作为 FFA 的初始种群。同时，寻优进程转换为 FFA 阶段，利用其卓越的局部开发最终获得最优解，其系统流程图如图 4-5 所示：

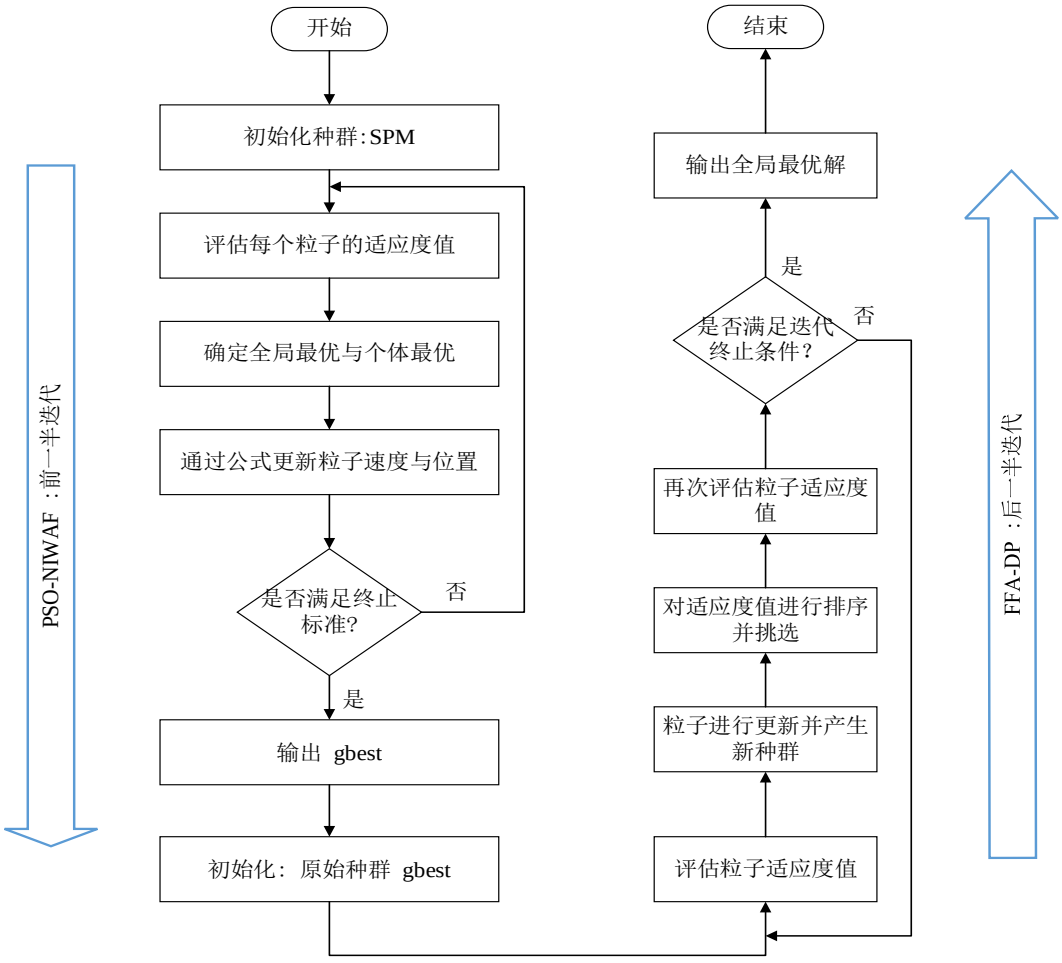


图 4-5 HPSOFF 系统流程图

HPSOFF 的伪代码在算法 1 中给出：

Algorithm 1: Pseudocode of the proposed HPSOFF

Require: r_{ij} β_0 I_0 γ // Control parameters of FFA-DP
Require: w // Nonlinear adaptive inertia weight
Require: $Iter_{\max}$ // Maximal generation number
Require: N // Size of the swarm
Require: D // Dimension of solutions
// Use PSO-NIWAF for optimization
// Initialization
$Iter$ =0
Initialize the population x by Eq (4.1)
Initialize the velocity v randomly
Calculate the fitness value *fitness*
//main loop
while $Iter < Iter_{\max} / 2$ **do**
  for i =1 to N **do**
    Update particle velocity by Eqs
    Update particle position by Eqs
    Evaluate $particle_{i,j}^t$
      if $fitness(particle_{i,j}^t) \le fitness(pbest_{i,j}^t)$
        $pbest_{i,j}^t = particle_{i,j}^t$
      end if
      if $fitness(pbest_{i,j}^t) \le fitness(gbest_{i,j}^t)$
        $gbest_{i,j}^t = pbest_{i,j}^t$
      end if
      Check position and velocity range limitations
      Calculate the fitness value *fitness*
      Output $gbest_{i,j}^t$ and *fitness*
  end for
end while
// Use FFA-DP for optimization
// Initialization
Initialize the population x_{new} =previous $gbest_{i,j}^t$
Calculate the fitness value *fitness*
//main loop
while $Iter_{\max} / 2 < Iter < Iter_{\max}$ **do**
  for i =1 to N **do**
    Update particle position by Eq (4.4) to Eq (4.7)
    Obtain new population x_{new2}
    Rank x_{new} and x_{new2}
    Reconstruct the population X_i^t
    Evaluate X_i^t
      if $fitness(X_i^t) \le fitness(pbest_{i,j}^t)$
        $pbest_{i,j}^t = X_i^t$
      end if
      if $fitness(pbest_{i,j}^t) \le fitness(gbest_{i,j}^t)$
        $gbest_{i,j}^t = pbest_{i,j}^t$
      end if
      Calculate the fitness value *fitness* and output results
  end for
end while

4.2 收敛性分析

HPSOFF 算法是一种混合优化算法，即增强型 PSO 算法在进行一定迭代次数的全局搜索后，将所寻到的最优解作为 FFA 算法的初始种群并接着进行局部开发。此时，相较于 FFA 算法而言 HPSOFF 算法仅仅是初始种群产生了改变，不会影响算法的收敛性。紧接着，构建了萤火虫算法的马尔科夫链数学模型，并证明萤火虫种群位置状态序列为有限齐次 Markov 链；通过分析马尔科夫链的相关性质，得出 HPSOFF 算法群体状态序列必将进入最优状态集的结论；之后证明 HPSOFF 算法满足随机优化算法收敛准则的两个条件 C_1 和 C_2，最终验证 HPSOFF 算法全局收敛。

在证明 HPSOFF 算法全局收敛之前，首先分析为何算法满足条件 C_1 和 C_2，即可表明算法全局收敛。

条件 C_1：若 $f(A(x,\xi)) \le f(x)$，且 $\xi \in S$，则 $f(A(x,\xi)) \le f(\xi)$。

条件 C_2：对 $\forall D \in S$，当 $v(D) > 0$ 时，有：

$$\prod_{k=0}^{\infty}(1-u_k(D)) = 0$$

式中：$u_k(D)$ 为算法 A 第 k 次迭代搜索解在集合 D 上的概率测度。

引理 1[57]（全局收敛）：设 f 是可测的，可测空间 S 是 R^n 上的可测子集，算法 A 满足条件 C_1 和 C_2，$\left\{x^k\right\}_{k=0}^{\infty}$ 是算法 A 产生的解序列，则有 $\lim\limits_{k\to\infty} P(x_k \in R_{\varepsilon,M}) = 1$，此时算法 A 全局收敛，$P(x_k \in R_{\varepsilon,M})$ 是算法 A 第 k 步的搜索结果 x_k 在 $R_{\varepsilon,M}$ 中的概率测度。

证明[57]：由条件 C_1 得，$x^k \notin R_{\varepsilon,M} \Rightarrow x^l \notin R_{\varepsilon,M}, \forall l < k$ 即为：

$$P(x^k \in S / R_{\varepsilon,M}) \le \prod_{l=0}^{k-1}(1-u_l(R_{\varepsilon,M})) \tag{4.8}$$

$$P(x^k \in R_{\varepsilon,M}) \le 1-P(x^k \in S / R_{\varepsilon,M}) \ge 1-\prod_{l=0}^{k-1}(1-u_l(R_{\varepsilon,M})) \tag{4.9}$$

由条件 C_2 得，$\prod\limits_{l=0}^{k-1}(1-u_l(R_{\varepsilon,M})) = 0$，则有：

$$1 \ge \lim_{k\to\infty} P(x^k \in R_{\varepsilon,M}) \ge 1-\lim_{k\to\infty}\prod_{l=0}^{k-1}(1-u_l(R_{\varepsilon,M})) = 1 \tag{4.10}$$

即证 $\lim\limits_{k\to\infty} P(x^k \in R_{\varepsilon,M}) = 1$。

紧接着构建所提算法的马尔科夫模型。首先，定义 $y = (x, pb)$ 为萤火虫位置状态，则萤火虫所有可能位置状态构成萤火虫位置状态空间，记为：

$$Y=\{y=(x,pb)/x,pb\in S,f(pb)\le f(x)\} \tag{4.11}$$

式中：x 为萤火虫位置；pb 为最佳位置。

同理，定义 $q=(y_1,y_2,y_3,\cdots y_N)$ 为萤火虫位置的群体状态，则萤火虫种群位置状态空间为 $Q=\{q=(y_1,y_2,y_3,\cdots y_N),y_i\in Y,1\le i\le N\}$。

定义 1[58]（萤火虫位置的状态转移）：在萤火虫算法迭代中，对 $\forall y_1=(x_1,pb_1)\in Y$，$\forall y_2=(x_2,pb_2)\in Y$，萤火虫位置状态由 y_1 一步转移到 y_2，记为 $T_y(y_1)=y_2$。

定理 4-1：在 HPSOFF 算法中，萤火虫位置状态由 y_1 转移至 y_2 的转移概率 $P(T_y(y_1)=y_2)$ 表达式如下：

$$P(T_y(y_1)=y_2)=P(x_1\to x_1')P(pb_1\to pb_1')P(x_1'\to x_2)P(pb_1'\to pb_2) \tag{4.12}$$

证明：HPSOFF 算法种群是超空间中的一组点，因此各粒子的更新过程即为超空间中一组点之间的位置过渡。这意味着 $x_1\to x_1'$，$pb_1\to pb_1'$，$x_1'\to x_2$，$pb_1'\to pb_2$ 需同时成立，此时萤火虫位置状态从 y_1 转移至 y_2 的概率为：

$$P(T_y(y_1)=y_2)=P(x_1\to x_1')P(pb_1\to pb_1')P(x_1'\to x_2)P(pb_1'\to pb_2) \tag{4.13}$$

式中：萤火虫位置状态由 x_1 转移至 x_1' 的概率为：

$$P(x_1\to x_1')=\begin{cases}1/|gb-x_1| & x_1'\in[x_1,x_1+(x_1-gb)]\\ 0 & x_1'\notin[x_1,x_1+(x_1-gb)]\end{cases} \tag{4.14}$$

萤火虫历史最佳位置由 pb_1 转移至 pb_1' 的概率为：

$$P(pb_1\to pb_1')=\begin{cases}1 & f(x_1')\le f(pb_1)\\ 0 & f(x_1')>f(pb_1)\end{cases} \tag{4.15}$$

萤火虫位置状态由 x_1' 转移至 x_2 的概率为：

$$P(x_1'\to x_2)=\begin{cases}1-p_{ij} & r>p_{ij}\\ p_{ij} & \mathrm{r}<p_{ij}\end{cases} \tag{4.16}$$

萤火虫历史最佳位置由 pb_1' 转移至 pb_2 的概率为：

$$P(pb_1'\to pb_2)=\begin{cases}1 & f(x_2)\le f(pb_1')\\ 0 & f(x_2)>f(pb_1')\end{cases} \tag{4.17}$$

定义 2[58]（萤火虫位置的群体状态转移）：在萤火虫算法迭代中，对 $\forall q_i=(y_{i1},y_{i2},y_{i3},\cdots y_{iN})\in Q$，$\forall q_j=(y_{j1},y_{j2},y_{j3},\cdots y_{jN})\in Q$，萤火虫群体位置状态由 q_i 一步转移到 q_j，记作 $T_q(q_i)=q_j$。

定理 4-2：在 HPSOFF 算法中，萤火虫群体位置状态由 q_i 转移至 q_j 的转移概率 $P(T_q(q_i)=q_j)$ 表达式如下：

$$P(T_q(q_i)=q_j)=\prod_{k=1}^{N}P(T_y(y_{ik})=y_{jk}) \tag{4.18}$$

证明：为实现萤火虫位置群体状态从q_i一步转移到q_j，根据定义 1 可知萤火虫群体状态q_i中全部的萤火虫位置状态需同时转移至q_j中相应位置状态，即$T_y(y_{i1})=y_{j1}$，$T_y(y_{i2})=y_{j2}$，…，$T_y(y_{iN})=y_{jN}$同时成立，则萤火虫位置的群体状态转移概率表达式为：

$$P(T_q(q_i)=q_j)=P(T_y(y_{i1})=y_{j1})P(T_y(y_{i2})=y_{j2})\cdots P(T_y(y_{iN})=y_{jN})=\prod_{k=1}^{N}P(T_y(y_{ik})=y_{jk})$$

故定理 4-2 得证。

定理 4-3：在 HPSOFF 算法中，萤火虫种群的位置状态序列$\{q(t),t\geq 0\}$是有限齐次 Markov 链。

证明：有限性：由于任何优化算法的搜索空间均是有限的，则所有萤火虫位置状态y是有限的，所构成的位置状态空间Y亦是有限空间；同理，萤火虫群体位置状态空间Q由有限个数的萤火虫群体状态q构成，故萤火虫群体位置空间具有有限性。

Markov 性：由定理 4-2 知，在萤火虫群体位置状态序列$\{q(t),t\geq 0\}$中，对于$\forall q(t-1)\in Q,\forall q(t)\in Q$，其状态转移概率$P(T_q(t-1))=q(t))$与群体中全部萤火虫的位置转移概率$P((T_y(y_i(t-1))=y_i(t))$，$1\leq i\leq N$相关，其转移概率表达式如下：

$$\begin{aligned}P((T_y(y\ (t-1))=y\ (t))=&P(x(t-1)\rightarrow x'(t-1))P(pb(t-1)\rightarrow pb'(t-1))\\&P(x'(t-1)\rightarrow x(t))P(pb'(t-1)\rightarrow pb(t))\end{aligned} \tag{4.19}$$

由式(4.19)可知，任意t时刻萤火虫位置状态的转移概率仅与$t-1$时刻状态有关，同理萤火虫位置群体的状态转移概率$P(T_q(t-1))=q(t))$也仅与前一时刻群体状态有关，群体状态序列$\{q(t),t\geq 0\}$的 Markov 性得证。

齐次性：根据定理 4-2 可知，$P(x(t-1)\rightarrow x'(t-1))$，$P(pb(t-1)\rightarrow pb'(t-1))$，$P(x'(t-1)\rightarrow x(t))$，$P(pb'(t-1)\rightarrow pb(t))$仅与前一时刻的状态相关而与时间无关，则由式(4.19)可知$P((T_y(y\ (t-1))=y\ (t))$与时间t无关，同理得出$P((T_q(q\ (t-1))=q\ (t))$与时间t无关，即证萤火虫位置群体状态序列的齐次性。

定理 4-4：在 HPSOFF 算法中，最优萤火虫位置的状态集合W是其状态空间Y上的一个闭集。

证明：对$\forall y_i\in W,\forall y_j\notin W$，根据定理 4-1 可知$T_y(y_j)=y_i$的概率为$P(T_y(y_j)=y_i)=P(x_j\rightarrow x_j')P(pb_j\rightarrow pb_j')\ P(x_j'\rightarrow x_i)P(pb_j'\rightarrow pb_i)$，又因为对

$\forall y_i \in W, \forall y_j \notin W$ ， $f(pb_j) \geq f(pb_i) = f(gb) = \inf(f(a)), a \in S$ ， 由 式 (4.12) 知 $P(pb_j \to pb_j^{'})P(pb_j^{'} \to pb_i) = 0$，则 $P(T_y(y_j) = y_i) = 0$，故定理 4-4 得证。

定义 3[58]：设优化问题 $\langle S, f \rangle$ 的全局最优解为 gb，最优萤火虫位置的群体状态集为 $R = \{q = (y_1, y_2, \cdots y_N) / \exists y_i \in W, 1 \leq i \leq N\}$。

引理 2[58]：假定 Markov 链有一个非空集合 C，且不存在另一个非空闭集 B，使 $C \cap B = \varnothing$，则当 $j \in C$ 时，$\lim\limits_{n \to \infty} P(x_n = j) = \pi_j$，且当 $j \notin C$ 时，$\lim\limits_{n \to \infty} P(x_n = j) = 0$。

定理 4-5：在 HPSOFF 算法中，对于萤火虫群体位置序列 $\{q(t), t \geq 0\}$，最优萤火虫位置群体的状态集合 R 是其状态空间 Q 上的一个闭集。

证明：根据定理 4-2 可知 $\forall q_i \in R, \forall q_j \notin Q$ 时，$T_q(q_j) = q_i$ 的概率为：

$$P(T_q(q_j) = q_i) = \prod_{k=1}^{N} P(T_y(y_{jk}) = y_{ik}) \tag{4.20}$$

对 $\forall q_i \in R, \forall q_j \notin Q$，$T_q(q_j) = q_i$ 意为至少存在一个萤火虫的位置状态从 W 的内部转移至外部，即 $\exists T_y(y_{jk}) = y_{ik}, y_{jk} \in W, y_{ik} \notin W, 1 \leq i \leq N$。根据定理 4-4 可知 W 是 Y 的一个闭集且 $P(T_y(y_{jk}) = y_{ik}) = 0$，则：

$$P(T_q(q_j) = q_i) = P(T_y(y_{jk}) = y_{ik}) = 0 \tag{4.21}$$

故定理 4-5 得证。

定理 4-6：萤火虫群体位置状态空间 Q 中不存在 R 之外的非空闭集 L，使得 $R \cap L = \varnothing$。

证明：假设萤火虫群体位置状态空间 Q 中存在非空闭集 L，且 $R \cap L = \varnothing$，设 $q_i = (gb, gb, \cdots gb) \in R$，则 $\forall q_j = (y_{j1}, y_{j2}, \cdots y_{jN}) \in L$，有 $f(pb_j) > f(pb)$，又因为 $P(T_q(q_j) = q_i) = P(T_y(y_{jk}) = y_{ik})$，故 $P(T_y(y_j) = y_i)$ 都有：

$$P(T_y(y_j) = y_i) = P(x_j \to x_j^{'})P(pb_j \to pb_j^{'})P(x_j^{'} \to x_i)P(pb_j^{'} \to pb_i) \tag{4.22}$$

因 为 $P(pb_j \to pb_j^{'})P(pb_j^{'} \to pb_i) = 1$ ， $P(x_j \to x_j^{'})P(x_j^{'} \to x_i) > 0$ ， 则 $P(T_y(y_j) = y_i) \neq 0$，所以 L 不是非空闭集，与假设矛盾，定理 4-6 证毕。

定理 4-7：HPSOFF 算法必将收敛于全局最优。

证明：在 HPSOFF 算法中，采用精英保留策略，即当前粒子适应度值优于全局最佳适应度值时，全局最优进行替换以保证算法非增性，满足条件 C_1。此外通过定理 4-5，定理 4-6 及引理 2 可知当 $n \to \infty$ 时，萤火虫位置群体状态序列必将进入最优状态集 R，此时算法搜索到全局最优解，满足条件 C_2。故 HPSOFF 算法必将收敛于全局最优。

4.3 仿真验证及实际应用

4.3.1 仿真验证

为获得更精准的局部最优和全局最优，提升搜索精度和速度，提出了含混合优化机制的 HPSOFF 算法。为验证其性能，应用所提算法和其他 4 种算法，即 PSO[36]、线性惯性权重粒子群优化[37]（LPSO）、FFA[59]和正弦余弦算法[60]（SCA），对 23 个经典基准函数和著名的工程应用进行求解，所选函数的特性如表 4-1 所示。为保证对比实验的准确性及公平性，测试函数均固定为 30 维，所有算法均采用推荐参数值，具体如表 4-2 所示。种群规模和最大迭代次数分别设置为 50 和 1000，平均值是通过 30 次计算获得的。全部实验均在系统配置为 Windows 11 处理器英特尔酷睿 i7-12700 CPU、@2.10 GHz、16 GB RAM 和 MATLAB（R2021a）的软件上进行。

表 4-1 23 个经典测试函数的详细信息

NO.	Function	Dim	Range	$f_{\min}$	Type
1	$F_1(x)=\sum_{i=1}^{N}x_i^2$	30	[-100,100]	0	
2	$F_2(x)=\sum_{i=1}^{N}\lvert x_i\rvert+\prod_{i=1}^{N}\lvert x_i\rvert$	30	[-10,10]	0	
3	$F_3(x)=\sum_{i=1}^{N}(\sum_{j=1}^{i}x_j)^2$	30	[-100,100]	0	
4	$F_4(x)=\max\{\lvert x_i\rvert, 1\le i\le N\}$	30	[-100,100]	0	Unimodal
5	$F_5(x)=\sum_{i=1}^{N-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	[-30,30]	0	
6	$F_6(x)=\sum_{i=1}^{N}([x_i+0.5])^2$	30	[-100,100]	0	
7	$F_7(x)=\sum_{i=1}^{N}ix_i^4+random[0,1)$	30	[-1.28,1.28]	0	
8	$F_8(x)=\sum_{i=1}^{N}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	[-500,500]	-12567	
9	$F_9(x)=\sum_{i=1}^{N}[x_i^2-10\cos(2\pi x_i)+10]$	30	[-5.12,5.12]	0	
10	$F_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2})-\exp(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i))+20+e$	30	[-32,32]	0	
11	$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}})+1$	30	[-600,600]	0	Multimodal
12	$F_{12}(x)=\frac{\pi}{N}\{10\sin^2(\pi y_i)+\sum_{i=1}^{N-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_N-1)^2\}+\sum_{i=1}^{N-1}u(x_i,10,100,4)$，$y_i=1+\frac{x_i+1}{4}$，$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	30	[-50,50]	0	

续表 4-1：

13	$F_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{N-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_N-1)[1+\sin^2(2\pi x_N)]\}+\sum_{i=1}^{N}u(x_i,5,100,4)$	30	[-50,50]	0	
14	$F_{14}(x)=[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}]^{-1}$	2	[-65.536,65.536]	1	
15	$F_{15}(x)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	4	[-5,5]	0.0003075	
16	$F_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	[-5,5]	-1.0316285	
17	$F_{17}(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	2	[-5,10] * [0,15]	0.398	Fixed
18	$F_{18}(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_1^2)]\times[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[-2,2]	3	
19	$F_{19}(x)=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2]$	3	[0,1]	-3.86	
20	$F_{20}(x)=-\sum_{i=1}^{4}c_i\exp[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2]$	6	[0,1]	-3.32	
21	$F_{21}(x)=-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	[0,10]	-10.1532	
22	$F_{22}(x)=-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	[0,10]	-10.4028	
23	$F_{23}(x)=-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	[-100,100]	-10.5363	

表 4-2 算法参数设置

Algorithms	Parameter setting	References
PSO	$N=50,\ c_1=2,\ c_2=2$	文献[36]
LPSO	$N=50,\ c_1=2,\ c_2=2,\ w_{\max}=0.9,\ w_{\min}=0.4$	文献[37]
FFA	$N=50,\ \gamma=1,\ I_0=\beta_0=2,\ \varepsilon=0.2$	文献[59]
SCA	$N=50,\ \alpha$=2	文献[60]
HPSOFF	$N=50,\ c_1=c_2=2,\ w_{\max}=0.9,\ w_{\min}=0.1,\ \gamma=1,\ I_0=\beta_0=2,\ \varepsilon=0.2$	本文方法

1）收敛精度测试

为评估各算法复杂条件下的寻优能力，引入收敛精度测试。表 4-3 列出每种算法搜索精度的测试结果。结果表明，HPSOFF 算法在整体排名中稳居第一，始终保持竞争力，充分证明所提算法高效寻优性能。

此外，在表 4-3 中，符号“0”表示 HPSOFF 与对比算法的优化性能没有明显差异，而“1”和“-1”表示 HPSOFF 分别优于或劣于对比算法。表 4-3 中的结果表明，在 23 个函数中与 LSA[61]、DSA[62]、BSA[43]、FFA[59]、HS[63]、PSO[36]、LPSO[37]、GSA[44]及 SCA[60]相比，HPSOFF 算法分别占优 14、15、13、11、22、14、22、

22 和 21，其他测试函数结果包括相等和劣势，其中相等分别占 7、4、6、6、1、8、1、5 和 0，而劣势分别占 2、4、4、6、0、1、0、2 和 2。以上数据证明，所提算法具有较强的抗干扰能力。

表 4-3 30 维度下各算法收敛精度统计结果

Function	Item	LSA	DSA	BSA	FFA	HS	LPSO	SCA	GSA	PSO	HPSOFF
F_1	Mean	4.71E-08	1.23E+01	9.97E+00	1.65E+01	2.32E+01	4.36E-02	4.61E-04	2.77E-11	2.13E+00	**4.03E-13**
	Median	**1.85E−15**	9.40E+00	7.06E+00	2.31E+01	2.39E+00	4.56E-03	2.76E−05	3.12E−12	2.454E+00	3.30E-13
	Rank	3	8	7	9	10	5	4	2	6	1
	W-test	1	1	1	1	1	1	1	1	1	/
F_2	Mean	3.51E-02	1.01E+00	1.21E+00	1.83E+00	1.43E+00	6.00E-01	6.42E-06	5.09E-06	1.04E+01	**7.18E-07**
	Median	1.03E−03	7.21E−01	2.04E+00	3.26E+00	2.05E+00	5.59E-01	1.24E−06	5.19E−06	6.09E+00	**5.88E-07**
	Rank	4	6	7	9	8	5	3	2	10	1
	W-test	1	1	1	1	1	1	1	1	1	/
F_3	Mean	4.23E+01	2.09E+05	3.05E+03	3.75E+01	6.26E+04	3.91E+00	2.70E+03	3.26E+02	3.31E+01	**1.54E-01**
	Median	3.48E+01	2.16E+04	2.51E+03	3.77E+01	7.13E+03	4.16E+00	2.25E+03	3.02E+02	3.19E+01	**1.35E-01**
	Rank	5	10	8	4	9	2	7	6	3	1
	W-test	1	1	1	1	1	1	1	1	1	/
F_4	Mean	1.48E+00	2.68E+01	9.85E+00	1.58E+00	7.37E+00	1.03E+00	2.54E+01	**9.46E-02**	8.54E-01	2.43E-01
	Median	3.25E-01	2.70E+01	9.26E+00	2.16E+00	5.12E+00	1.09E+00	7.53E+00	**5.67E-06**	6.08E-01	1.56E-01
	Rank	5	10	8	6	7	4	9	1	3	2
	W-test	1	1	1	1	1	1	1	-1	1	/
F_5	Mean	6.42E+01	1.09E+03	4.71E+02	1.88E+02	8.29E+02	6.72E+01	4.29E+01	4.24E+01	9.59E+02	**4.26E+01**
	Median	6.36E+00	1.96E+03	3.85E+02	2.21E+02	4.17E+02	3.25E+01	4.91E+01	3.66E+01	6.04E+02	**2.68E+01**
	Rank	4	10	7	6	8	5	2	3	9	1
	W-test	1	1	1	1	1	1	1	0	1	/
F_6	Mean	3.34E+00	1.46E+01	1.29E+01	1.69E+01	2.52E+01	2.85E-02	4.36E+00	3.45E-10	2.39E+00	**0.00E+00**
	Median	2.59E+00	2.05E+01	1.15E+00	1.64E+01	4.02E+00	2.81E-02	2.48E+00	5.44E-11	2.30E+00	**0.00E+00**
	Rank	5	8	7	9	10	3	6	2	4	1
	W-test	1	1	1	1	1	1	1	1	1	/
F_7	Mean	2.36E-02	1.12E-01	5.46E-02	**1.18E-03**	4.68E-01	2.32E-01	3.68E-01	4.69E-01	2.45E+00	5.59E-03
	Median	2.28E-02	1.06E-01	5.19E-02	**1.34E-04**	2.65E-02	3.69E-01	2.58E-01	5.47E-01	2.48E+00	4.63E-03
	Rank	3	5	4	1	8	6	7	9	10	2
	W-test	1	1	1	-1	1	1	1	1	1	/
F_8	Mean	-8.00E+03	-1.05E+04	-9.59E+03	**-1.23E+04**	-7.39E+03	-7.76E+03	-4.08E+03	-2.78E+03	-7.57E+03	-2.03E+04
	Median	-5.18E+03	-2.36E+04	-7.65E+03	**-1.33E+04**	-7.25E+03	-7.70E+03	-4.01E+03	-1.17E+03	-7.64E+03	-3.77E+04
	Rank	4	1	2	3	8	5	9	10	6	7
	W-test	0	-1	-1	-1	0	0	1	1	0	/
F_9	Mean	6.26E+01	4.70E+01	6.48E+01	5.51E+01	8.70E+01	4.98E+01	1.17E+01	1.93E+01	1.97E+02	**1.13E+01**
	Median	7.95E+01	3.67E+01	5.06E+01	5.45E+01	7.03E+01	4.66E+01	**2.38E-01**	1.89E+01	1.99E+02	1.17E+01
	Rank	7	4	8	6	9	5	2	3	10	1
	W-test	1	1	1	1	1	1	0	1	1	/
F_{10}	Mean	2.56E+00	4.04E+00	3.17E+00	2.33E+00	8.87E+00	4.88E-01	1.18E+01	3.22E-06	2.56E+00	**3.26E-07**
	Median	3.48E+00	6.16E+00	2.79E+00	2.32E+00	8.11E+00	5.16E-01	1.94E+01	2.11E-06	3.05E+00	**2.86E-07**
	Rank	5	8	7	4	9	3	10	2	6	1
	W-test	1	1	1	1	1	1	1	1	1	/
F_{11}	Mean	**7.23E-03**	1.16E+00	1.08E+00	1.23E+00	1.29E+00	1.14E-02	1.48E-01	4.82E+00	1.83E-01	1.13E-02
	Median	**7.42E-04**	1.07E+00	1.03E+00	1.16E+00	1.27E+00	1.03E-03	2.68E-02	5.04E+00	2.33E-01	1.11E-02
	Rank	1	7	6	8	9	3	4	10	5	2
	W-test	-1	1	1	1	1	0	1	1	1	/
F_{12}	Mean	3.58E-01	2.96E-01	1.36E-01	6.76E-02	1.84E-01	2.78E+00	4.85E+01	1.28E-01	9.06E+00	**2.07E-02**
	Median	1.05E-01	1.69E-01	7.59E-02	6.87E-02	3.45E-01	2.81E+00	5.45E-01	3.21E-03	8.37E+00	**6.07E-15**
	Rank	7	6	4	2	5	8	10	3	9	1
	W-test	1	1	1	1	1	1	1	1	1	/
F_{13}	Mean	2.69E-02	1.27E+00	7.57E-01	6.58E-01	1.85E+00	2.05E-02	4.25E+00	3.31E-02	5.11E-01	**4.03E-03**
	Median	5.21E-01	2.56E+00	2.73E-01	3.45E-00	1.71E+00	1.99E-02	2.98E+00	1.92E-11	4.01E-01	**2.02E-13**
	Rank	3	8	7	6	9	2	10	4	5	1
	W-test	1	1	1	1	1	1	1	1	1	/
F_{14}	Mean	1.077447	**0.998004**	**0.998004**	**0.998004**	1.605938	**0.998004**	1.295645	4.285704	**0.998004**	1.083574
	Median	**0.998004**	**0.998004**	**0.998004**	**0.998004**	**0.998004**	**0.998004**	0.998014	2.982127	**0.998004**	0.998003
	Best	**0.998004**	**0.998004**	**0.998004**	**0.998004**	**0.998004**	**0.998004**	**0.998004**	1.891408	**0.998004**	0.898003
	Worst	2.982105	**0.998004**	**0.998004**	**0.998004**	5.988898	**0.998004**	2.982105	10.770589	**0.998004**	3.928845
	Rank	2	1	1	1	5	1	4	6	1	3
	W-test	0	0	0	0	1	0	1	1	0	/

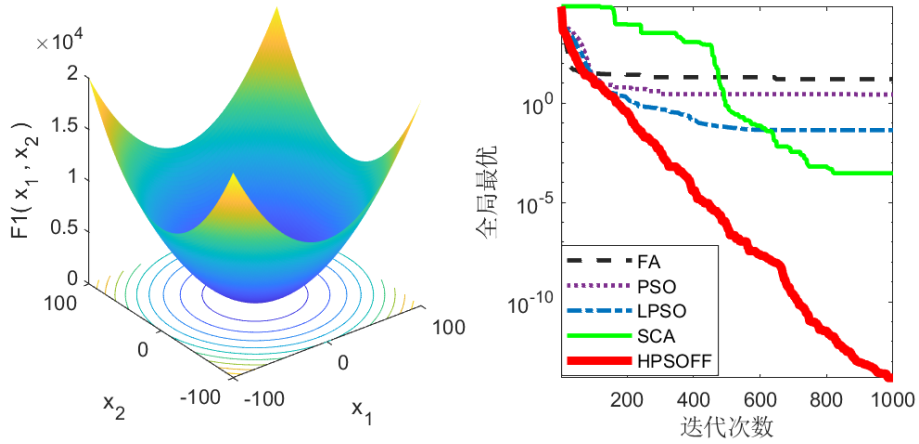
续表 4-3：

F_{15}	Mean	0.000535	0.00096	0.000626	0.000309	0.000963	**0.0003075**	0.00086	0.004355	0.000733	0.0002421
	Median	0.000309	0.000865	0.000632	0.000308	0.000934	**0.0003075**	0.000731	0.004105	0.000709	**0.0003075**
	Best	0.000307	0.000602	0.000379	0.000307	0.000739	**0.0003075**	0.000414	0.001686	0.000671	**0.0003075**
	Worst	0.001594	0.001954	0.000817	0.000309	0.0015	**0.0003075**	0.001452	0.008215	0.000851	0.002043
	Rank	3	7	4	2	8	1	6	10	5	9
	W-test	1	1	1	-1	1	-1	1	1	1	/
F_{16}	Mean	**-1.031628**	**-1.031628**	**-1.031628**	**-1.031628**	-0.996514	**-1.031628**	-1.031615	**-1.031628**	-1.031621	**-1.031628**
	Median	**-1.031628**	**-1.031628**	**-1.031628**	**-1.031628**	-1.02233	**-1.031628**	-1.031616	**-1.031628**	-1.031621	**-1.031628**
	Best	**-1.031628**	**-1.031628**	**-1.031628**	**-1.031628**	-1.031627	**-1.031628**	**-1.031628**	**-1.031628**	-1.031621	**-1.031628**
	Worst	**-1.031628**	**-1.031628**	**-1.031628**	**-1.031628**	-0.668653	**-1.031628**	-1.031588	**-1.031628**	-1.031621	**-1.031628**
	Rank	1	1	1	1	4	1	3	1	2	1
	W-test	0	0	0	0	1	0	1	0	1	/
F_{17}	Mean	**0.397887**	**0.397887**	**0.397887**	**0.397887**	0.407773	**0.397887**	0.398417	**0.397887**	**0.397887**	**0.397887**
	Median	**0.397887**	**0.397887**	**0.397887**	**0.397887**	0.39844	**0.397887**	0.398157	**0.397887**	**0.397887**	**0.397887**
	Best	**0.397887**	**0.397887**	**0.397887**	**0.397887**	**0.397887**	**0.397887**	0.397891	**0.397887**	**0.397887**	**0.397887**
	Worst	**0.397887**	**0.397887**	**0.397887**	**0.397887**	0.495021	**0.397887**	0.401634	**0.397887**	**0.397887**	**0.397887**
	Rank	1	1	1	1	3	1	2	1	1	1
	W-test	0	0	0	0	1	0	1	0	0	/
F_{18}	Mean	**3**	3.000000008	**3**	**3**	3.000052375	**3**	3.000015858	**3**	3.000484	**3**
	Median	**3**	3	**3**	**3**	3.000029069	**3**	3.000006115	**3**	3.000266	**3**
	Best	**3**	**3**	**3**	**3**	3.000000457	**3**	3.000000431	**3**	3.000216	**3**
	Worst	**3**	3.00000038	**3**	**3**	3.000258898	**3**	3.000102102	**3**	3.001372	**3**
	Rank	1	2	1	1	4	1	3	1	5	1
	W-test	0	1	0	0	1	0	1	0	1	/
F_{19}	Mean	**-3.862782**	**-3.862782**	**-3.862782**	**-3.862782**	-3.860211	**-3.862782**	-3.855231	-3.514778	-3.861302	**-3.862782**
	Median	**-3.862782**	**-3.862782**	**-3.862782**	**-3.862782**	-3.861372	**-3.862782**	-3.854523	-3.539936	-3.861553	**-3.862782**
	Best	**-3.862782**	**-3.862782**	**-3.862782**	**-3.862782**	-3.86772	**-3.862782**	-3.862132	**-3.862782**	-3.862307	**-3.862782**
	Worst	**-3.862782**	**-3.862782**	**-3.862782**	**-3.862782**	-3.843937	**-3.862782**	-3.853671	-3.08581	-3.859849	**-3.862782**
	Rank	1	1	1	1	2	1	4	5	3	1
	W-test	0	0	0	0	1	0	1	1	1	/
F_{20}	Mean	-3.27206	**-3.321995**	**-3.321995**	-3.322	-3.121637	-3.250659	-2.906018	-2.048445	-3.187518	-3.286327
	Median	**-3.321995**	**-3.321995**	**-3.321995**	-3.322	-3.155958	-3.203102	-3.011903	-2.019945	-3.146741	**-3.321995**
	Best	**-3.321995**	**-3.321995**	**-3.321995**	-3.322	-3.303125	**-3.321995**	-3.126322	**-3.321995**	-3.297050	**-3.321995**
	Worst	-3.203102	**-3.321995**	-3.321978	-3.322	-2.664424	-3.203102	-1.456282	-1.254658	-3.122148	**-3.321995**
	Rank	4	1	1	2	7	5	8	9	6	3
	W-test	1	0	0	0	1	1	1	1	1	/
F_{21}	Mean	-7.02732	-10.152834	**-10.15309**	-10.1521	-2.782671	-5.119264	-3.012834	-5.055198	-6.727041	-6.649343
	Median	-5.100772	-10.153195	**-10.15319**	-10.1521	-2.167447	-5.055198	-2.375601	-5.055198	-7.510957	-7.626986
	Best	**-10.1532**	**-10.1532**	**-10.1532**	-10.1521	-9.127235	**-10.1532**	-8.10853	-5.055198	-10.12140	**-10.1532**
	Worst	-2.630472	**-10.144972**	-10.14913	-10.1521	-1.141857	-2.630472	-0.498192	-5.055198	-2.62399	-2.630472
	Rank	4	2	1	3	10	7	9	8	5	6
	W-test	-1	-1	-1	-1	1	1	1	1	0	/
F_{22}	Mean	-7.136702	-10.393586	**-10.40292**	-10.4022	-3.045779	-8.680252	-3.846918	-8.542597	-9.038078	-8.345327
	Median	**-10.40294**	**-10.40294**	**-10.40294**	-10.4026	-2.485769	-10.4029	-4.658063	**-10.40294**	-10.29466	**-10.40294**
	Best	**-10.40294**	**-10.40294**	**-10.40294**	-10.4012	-9.367837	-10.4029	-6.587977	**-10.40294**	-10.32700	**-10.40294**
	Worst	-1.837593	-10.101115	**-10.40224**	-10.4032	-1.259142	-3.7243	-0.906982	-5.087672	-2.755298	-2.751934
	Rank	8	3	1	2	10	5	9	6	4	7
	W-test	1	-1	-1	-1	1	0	1	0	-1	/
F_{23}	Mean	-7.910438	-10.53317	**-10.53641**	-10.5359	-4.204342	-8.958416	-5.538952	**-10.53641**	-10.34972	-7.1044928
	Median	**-10.53641**	-10.536396	**-10.53641**	-10.5359	-2.682462	**-10.53641**	-5.060989	**-10.53641**	-10.37284	-7.832445
	Best	**-10.53641**	**-10.53641**	**-10.53641**	-10.5359	-10.151938	**-10.53641**	-7.998784	**-10.53641**	-10.47993	**-10.53641**
	Worst	-1.85948	-10.458195	-10.53598	-10.5359	-2.629258	-2.421734	-4.512439	**-10.53641**	-10.12205	-2.427335
	Rank	6	3	1	2	9	5	8	1	4	7
	W-test	0	-1	-1	-1	1	-1	1	-1	-1	/
/	**AvgRan**	**3.78**	**4.91**	**4.13**	**3.87**	**7.43**	**3.65**	**6.04**	**4.57**	**5.30**	**2.65**
	Rank	**3**	**7**	**5**	**4**	**10**	**2**	**9**	**6**	**8**	**1**
	W-test	**14/7/2**	**15/4/4**	**13/6/4**	**11/6/6**	**22/1/0**	**14/8/1**	**22/1/0**	**16/5/2**	**21/0/2**	**/**

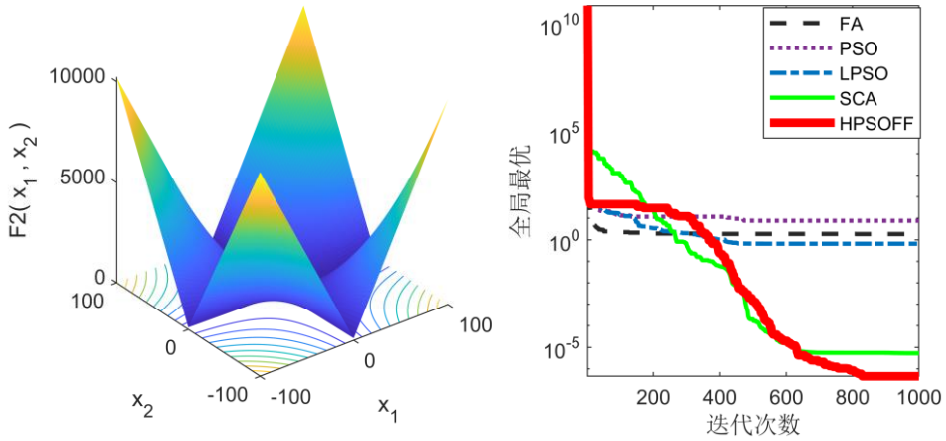
注:1/0/-1 表示 HPSOFF 性能优于、等于和劣于对比算法，14/7/2 表示 HPSOFF 算法与对比算法相比占优、相等、劣势分别为 14、7、2。

2）收敛曲线比较

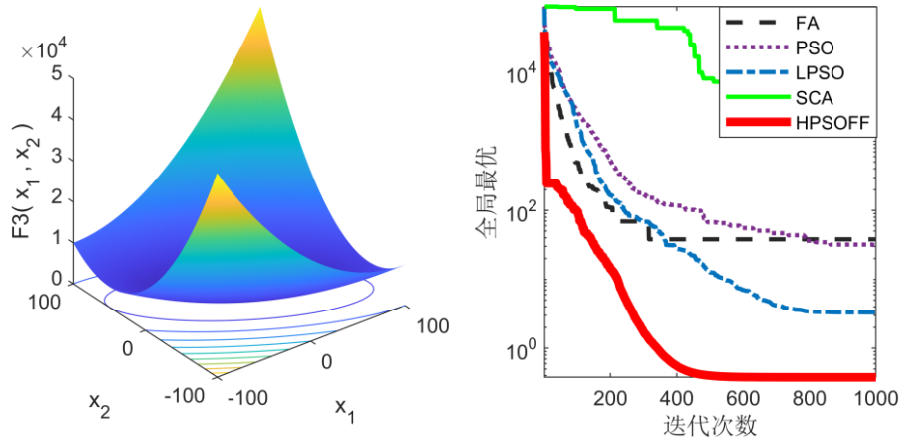
为验证 HPSOFF 算法在单峰函数上的有效性，绘制出所有函数的二维形状图形；同时展示出 30 维度下各算法收敛曲线，具体如图 4-6 所示。



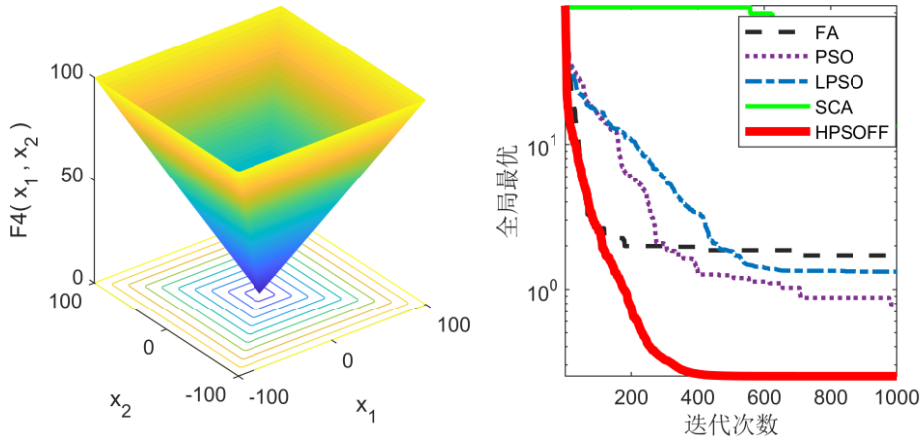
(a) 各算法函数 F_1 收敛曲线



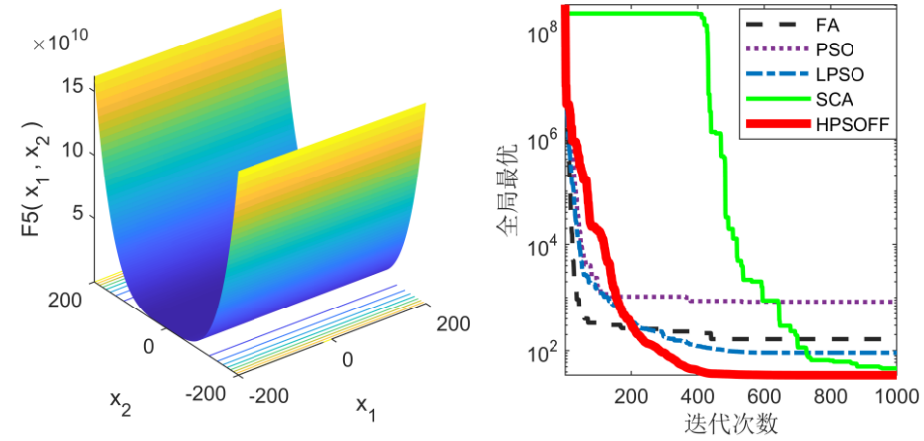
(b) 各算法函数 F_2 收敛曲线



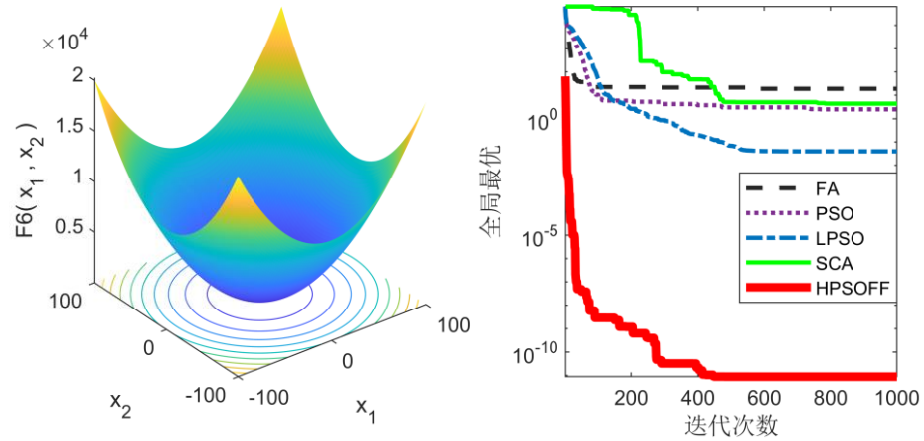
(c) 各算法函数 F_3 收敛曲线



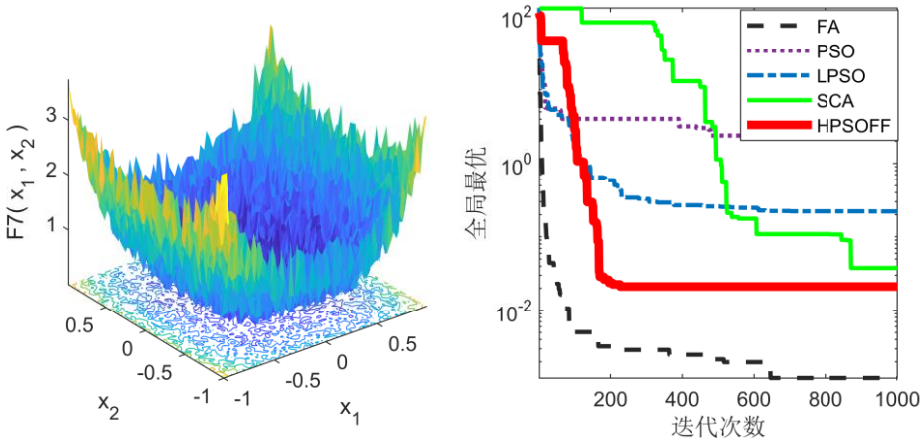
(d) 各算法函数 F_4 收敛曲线



(e) 各算法函数 F_5 收敛曲线



(f) 各算法函数 F_6 收敛曲线

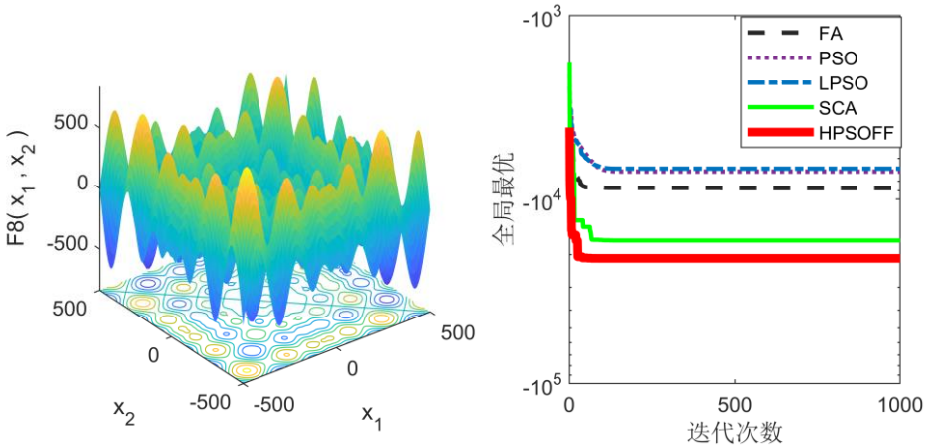


(g) 各算法函数 F_7 收敛曲线

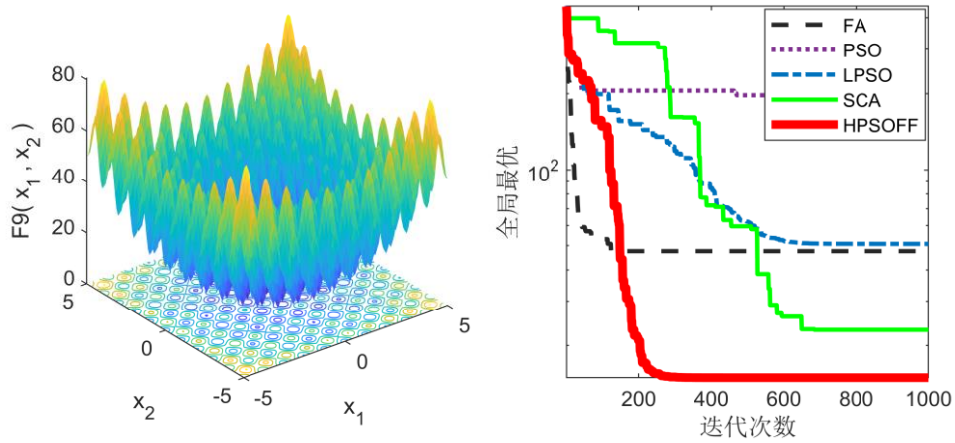
图 4-6 30 维度下单峰函数收敛曲线

如图 4-6 所示，对于 F_1、F_2、F_3、F_4 及 F_6，HPSOFF 算法得到的最优值在迭代初期优于其他算法，说明混沌初始化可以提高搜索效率，增强寻优能力；此外，对于除 F_4 之外的所有单峰函数，所提算法均排名第一，其收敛精度与对比算法相比大幅度提高，表明 HPSOFF 能够有效搜索解空间。

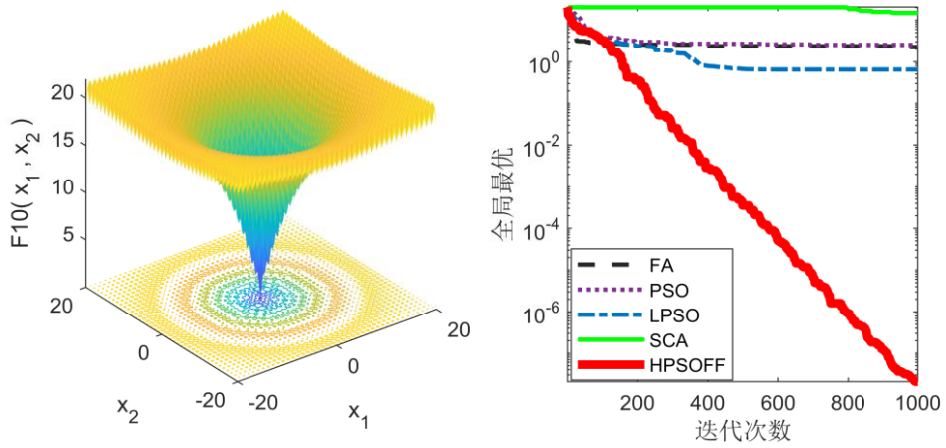
多模态函数($F_8 \sim F_{13}$)的特征是其具有多个局部最优，通常被认为是极其复杂的，主要验证算法是否具有优秀的抗干扰能力，具体寻优效果如图 4-7 所示。



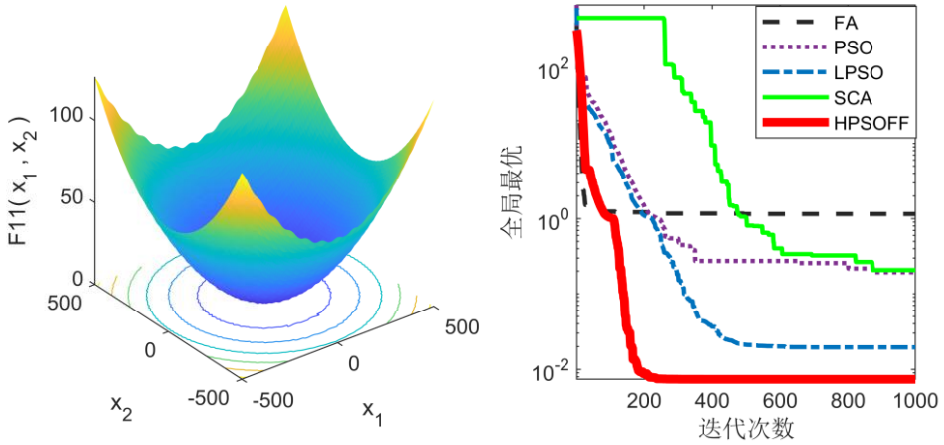
(a) 各算法函数 F_8 收敛曲线



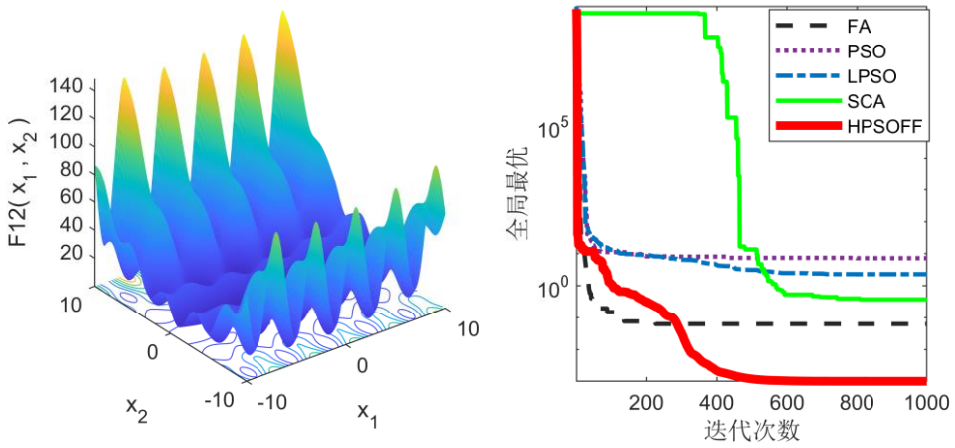
(b) 各算法函数 F_9 收敛曲线



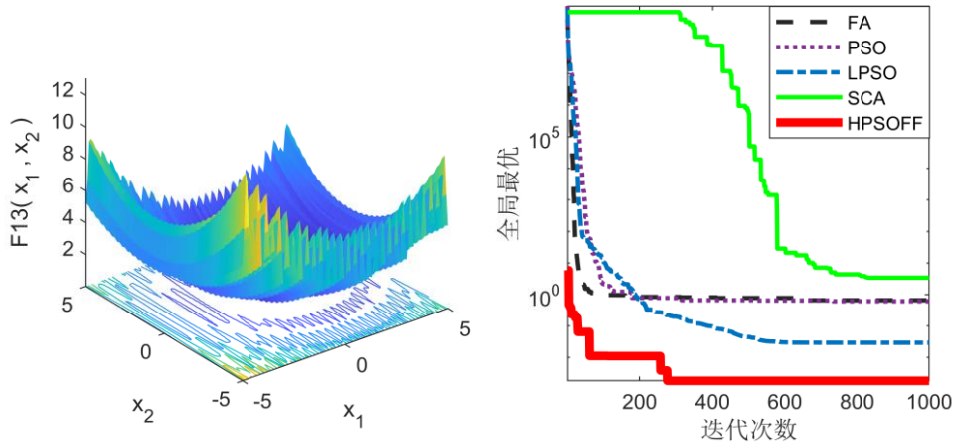
(c) 各算法函数 F_{10} 收敛曲线



(d) 各算法函数 F_{11} 收敛曲线



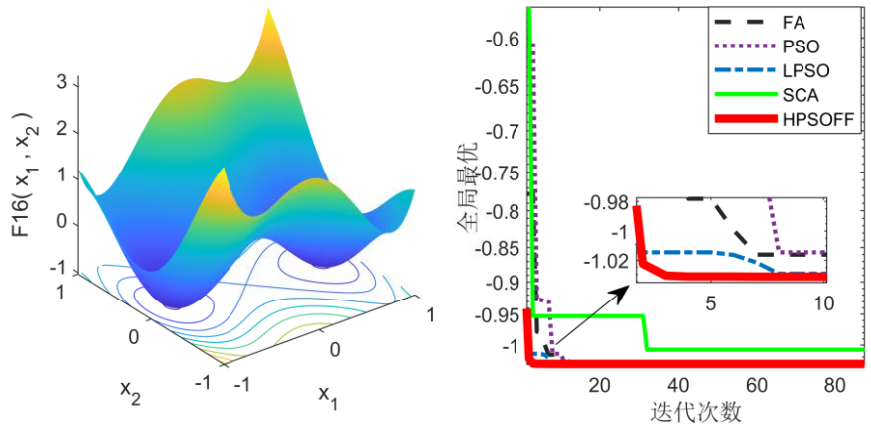
(e) 各算法函数 F_{12} 收敛曲线



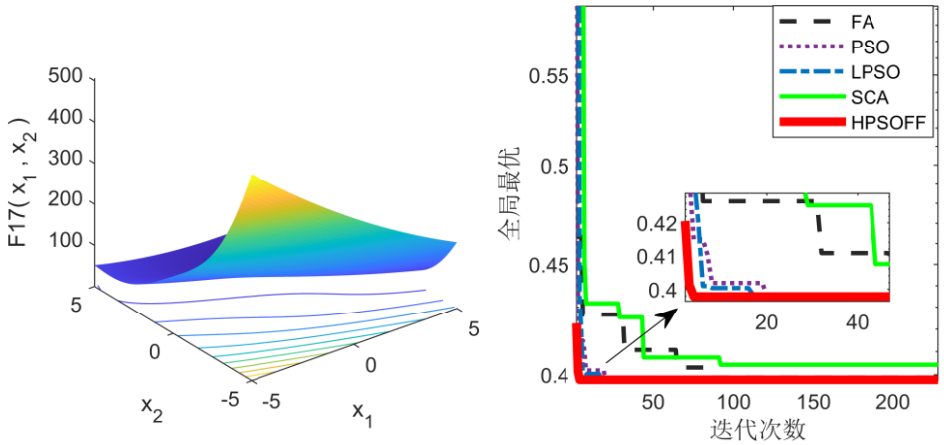
(f) 各算法函数 F_{13} 收敛曲线

图 4-7 30 维度下多峰函数收敛曲线

从图 4-7 可以看出，对于 F_8、F_{10}、F_{11}、F_{12} 及 F_{13}，HPSOFF 算法的收敛曲线几乎是垂直的，可以突出展示 HPSOFF 算法勘探和开发能力。此外，所提算法在固定维度函数下的寻优性能如图 4-8 所示：



(a) 各算法函数 F_{16} 收敛曲线



(b) 各算法函数 F_{17} 收敛曲线

图 4-8 固定维度下各函数收敛曲线

图 4-8 呈现出 HPSOFF 算法在固定维度函数下的优化性能，从图中可知与 FFA、PSO、LPSO 及 SCA 相比，所提算法能够快速找到全局最优，提高效率。

3）与目前先进算法的比较

为了验证所提算法的优越性，引入了总体排名和 Wilcoxon 排名测试并将 HPSOFF 算法与当前主流算法进行对比实验，如表 4-4 所示，HPSOFF 算法在大多数测试函数中始终保持较高的精度和稳定性。

表 4-4 30 维度下 HPSOFF 算法与目前主流算法对比结果统计

Function	Item	GA	SSA	CS	MCS	ES	ALO	GGSA	HPSOFF
F_1	Mean	5.55E-01	1.58E-07	7.53E+01	1.11E+00	2.14E+03	7.09E-06	**2.78E-18**	4.03E-13
	STD	2.32E+00	2.07E-07	2.62E+01	3.15E-01	1.66E+03	2.63E-05	**1.03E-18**	2.77E-13
	Rank	5	3	7	6	8	4	1	2
	W-test	1	1	1	1	1	1	-1	/
F_2	Mean	5.66E-03	2.36E+00	1.01E+01	3.24E-01	1.17E+01	4.78E+00	**9.54E-09**	7.16E-07
	STD	2.04E-02	1.76E+01	3.3E+00	3.13E-01	3.66E+00	5.31E+01	**7.39E-09**	4.31E-07
	Rank	3	5	7	4	8	6	1	2
	W-test	1	1	1	1	1	1	-1	/
F_3	Mean	8.54E+02	1.73E+03	6.05E+03	2.11E+02	5.08E+04	1.66E+03	4.67E+02	**1.51E-01**
	STD	2.10E+02	1.12E+04	2.51E+02	3.23E+02	2.51E+03	5.15E+02	3.01E+02	**8.33E-01**
	Rank	4	6	7	2	8	5	3	1
	W-test	1	1	1	1	1	1	1	/
F_4	Mean	4.56E+00	1.17E+01	6.98E+00	2.14E+00	6.53E+01	2.48E+01	7.08E-01	**2.33E-01**
	STD	5.92E-01	4.18E+00	2.05E-01	3.66E-01	2.58E+00	4.55E+00	4.89E-01	**1.58E-02**
	Rank	4	6	5	3	8	7	2	1
	W-test	1	1	1	1	1	1	1	/
F_5	Mean	2.67E+02	2.95E+02	**1.13E+00**	9.66E+01	1.03E+05	3.04E+01	4.23E+01	4.35E+01
	STD	5.01E+02	5.09E+02	**2.05E+00**	2.56E+01	3.68E+05	3.51E+02	3.11E+01	3.62E+01
	Rank	6	7	1	5	8	2	3	4
	W-test	1	1	-1	1	1	-1	-1	/
F_6	Mean	5.63E-01	1.80E-07	9.78E+04	5.12E+03	3.43E+03	3.09E-06	1.17E-18	**0**
	STD	2.17E+00	3.00E-07	3.56E+03	6.17E+02	3.02E+03	1.02E-06	4.77E-18	**0**
	Rank	5	3	8	7	6	4	2	1
	W-test	1	1	1	1	1	1	1	/
F_7	Mean	4.29E-02	1.57E-01	2.43E-02	9.11E-03	9.77E+01	8.78E-02	7.03E-01	**3.61E-03**
	STD	5.94E-03	6.27E-02	5.03E-03	**2.12E-04**	3.12E+01	4.56E-02	2.36E+00	3.05E-03
	Rank	4	6	3	2	8	5	7	1
	W-test	1	1	1	1	1	1	1	/
F_8	Mean	**-1.06E+04**	-7.42E+03	-8.97E+03	-9.80E+03	-8.45E+03	-5.56E+03	-2.91E+03	-7.53E+03
	STD	3.53E+03	7.22E+02	**2.55E+02**	6.06E+02	4.33E+02	2.06E+03	5.01E+02	2.31E+03

续表 4-4：

F_9	Rank	1	6	3	2	4	7	8	5
	W-test	-1	0	-1	-1	-1	1	1	/
	Mean	**3.08E+01**	6.01E+01	3.01E+02	1.35E+02	2.44E+02	7.70E+01	3.23E+02	4.28E+01
	STD	**1.01E+00**	3.21E+00	3.13E+01	3.04E+01	3.16E+01	2.26E+02	1.19E+01	2.01E+00
	Rank	1	3	7	5	6	4	8	2
	W-test	-1	1	1	1	1	1	1	/
F_{10}	Mean	1.64E+00	2.68E+00	1.94E+01	1.23E+01	7.27E+00	2.33E+00	**1.18E-09**	3.27E-07
	STD	5.23E-01	8.28E-00	4.05E-01	6.89E-02	1.65E+00	1.17E+00	**4.68E-10**	3.22E-07
	Rank	3	5	8	7	6	4	1	2
	W-test	1	1	1	1	1	1	-1	/
F_{11}	Mean	5.61E-01	1.62E-02	6.51E+02	7.21E+00	7.61E+01	**2.51E-03**	7.27E+00	1.14E-02
	STD	3.06E-01	2.12E-02	2.53E+00	1.53E+01	2.92E+00	**6.09E-04**	4.22E-02	2.00E-02
	Rank	4	3	8	6	7	1	5	2
	W-test	1	1	1	1	1	-1	1	/
F_{12}	Mean	3.09E-02	6.97E+00	2.33E+00	2.01E-01	3.18E+05	1.56E+01	2.15E-01	**2.07E-02**
	STD	4.09E-02	4.42E+00	2.55E-01	3.55E-01	1.32E+05	2.90E+00	2.95E-01	**4.33E-02**
	Rank	2	6	5	3	8	7	4	1
	W-test	1	1	1	1	1	1	1	/
F_{13}	Mean	3.63E-01	1.58E+01	7.64E-01	9.01E-03	7.34E+05	1.21E-02	2.02E-03	**1.27E-03**
	STD	3.11E-01	1.62E+01	2.13E-01	5.30E-03	1.12E+05	1.29E+01	**5.01E-04**	7.06E-03
	Rank	5	7	6	3	8	4	2	1
	W-test	1	1	1	1	1	1	1	/
F_{14}	Mean	**0.998004**	1.1965	0.998	2.344	8.06397	**0.998004**	3.59199	2.08357
	STD	**4.23E-12**	0.5467	1.466E-06	2.1186	5.877871	1.306659	2.780351	1.944989
	Rank	1	3	2	5	7	1	6	4
	W-test	-1	-1	-1	1	1	-1	1	/
F_{15}	Mean	0.005206	0.000886	0.000556	**0.0005487**	0.003	0.000782	0.002066	0.002415
	STD	0.007031	0.000257	**0.000075**	0.000154	0.005902	0.005973	0.000910	0.005990
	Rank	8	4	2	1	7	3	5	6
	W-test	1	-1	-1	-1	-1	-1	0	/
F_{16}	Mean	-1.03162	-1.03163	-1.0316	-1.0316	-1.03163	-1.03163	-1.03163	**-1.0316285**
	STD	0.00000134	6.13E-14	7.56E-12	3.65E-10	6.71E-16	5.63E-14	5.68E-16	**0**
	Rank	3	2	4	4	2	2	2	1
	W-test	1	1	1	1	1	1	1	/
F_{17}	Mean	0.39789	0.397887	0.3979	0.3979	55.6	**0.398**	0.397887	**0.398**
	STD	0.0000108	3.41E-14	1.12E-08	3.7E-09	2.89E-14	5.92E-14	0	**0**
	Rank	3	3	2	2	4	1	3	1
	W-test	1	1	1	1	1	0	1	/
F_{18}	Mean	3.000002	**3**	**3**	**3**	**3**	**3**	**3**	**3**
	STD	0.00000406	2.2E-13	2.99E-09	7.85E-08	8.61E-16	2.75E-13	2.83E-15	**0**
	Rank	2	1	1	1	1	1	1	1
	W-test	1	0	0	0	0	0	0	/
F_{19}	Mean	-3.86278	-3.86278	-3.86281	-3.86281	-3.86278	-3.86278	**-3.86147**	-3.86278
	STD	1.63E-07	1.47E-10	0.00000869	9.71E-08	2.71E-15	3.29E-14	0.002987	**0**
	Rank	2	2	3	3	2	2	1	2
	W-test	0	0	1	1	0	0	-1	/
F_{20}	Mean	-3.2744	-3.2304	**-3.3213**	-3.2744	-3.2236	-3.3219	-3.0986	-3.2863
	STD	0.05924	0.0616	**0.0004875**	0.0614	0.050751	0.058277	0.498156	0.054483
	Rank	4	5	1	4	6	2	7	3
	W-test	1	1	-1	1	1	-1	1	/
F_{21}	Mean	-5.72536	-9.6334	**-10.143**	-3.4299	-4.99059	-5.10077	-4.7287	-6.6493
	STD	3.32622	1.8104	**0.0132**	2.3623	3.465861	2.702862	1.667901	3.571439
	Rank	4	2	1	8	6	5	7	3
	W-test	1	-1	-1	1	1	1	1	/
F_{22}	Mean	-6.94349	-9.0295	-10.391	-8.3021	-7.07562	**-10.4029**	-7.92248	-8.3453
	STD	3.56118	2.3911	**0.0082**	3.393	3.64907	3.424583	2.697054	3.202411
	Rank	8	3	2	5	7	1	6	4
	W-test	1	-1	-1	1	1	-1	1	/
F_{23}	Mean	-7.0208	-9.0333	-10.4927	-5.9976	-7.02909	**-10.5364**	-10.3561	-7.1044
	STD	3.85233	2.9645	**0.0293**	3.916	3.838	2.939	0.987348	3.535084
	Rank	7	4	2	8	6	1	3	5
	W-test	1	-1	-1	1	1	-1	-1	/
/	**AvgRank**	**3.87**	**4.13**	**4.14**	**4.17**	**6.13**	**3.43**	**3.82**	**2.39**
	Rank	**4**	**5**	**6**	**7**	**8**	**2**	**3**	**1**
	W-test	**18/1/3**	**15/3/5**	**14/1/8**	**20/1/2**	**19/2/2**	**13/3/7**	**14/2/7**	**/**

注:1/0/-1 表示 HPSOFF 性能优于、等于和劣于对比算法，18/1/3 表示 HPSOFF 算法与对比算法相比占优、相等、劣势分别为 18、1、3。

4）高维度性能比较

为了验证算法解决复杂问题的能力，引入了高维性能优化实验，采用 FFA、PSO、LPSO、SCA 及 HPSOFF 五种方法求解 100 维度下 13 个变维函数，具体结果如表 4-5 所示。

根据表 4-5 可以发现，HPSOFF 算法在 100 维度单模态/多模态函数上均排名第一，充分体现其强大的寻优能力。此外，数据表明，在求解过于复杂的问题时，所有算法的优化性能都会降低，值得注意的是，与其他对比算法相比，HPSOFF 算法的下降幅度较小，反映出其优异的鲁棒性和抗干扰水平。

表 4-5 100 维度下各算法寻优结果统计

Function	Method	Mean	STD	W-test	Rank
F_1	FFA	3.39E+02	2.64E+01	1	4
	PSO	4.03E+01	6.40E+00	1	3
	LPSO	7.86E+00	1.23E+00	1	2
	SCA	4.01E+03	4.17E+03	1	5
	HPSOFF	**1.51E-03**	**2.79E-04**	/	1
F_2	FFA	1.44E+01	5.45E-01	1	3
	PSO	6.17E+01	4.21E+00	1	5
	LPSO	2.23E+01	4.92E+00	1	4
	SCA	1.18E+00	1.20E+00	1	2
	HPSOFF	**1.12E-01**	**6.82E-02**	/	1
F_3	FFA	8.14E+03	1.27E+03	-1	2
	PSO	8.57E+03	9.31E+02	-1	3
	LPSO	**4.11E+03**	**4.36E+02**	-1	1
	SCA	1.67E+05	3.91E+04	1	5
	HPSOFF	1.41E+04	7.29E+03	/	4
F_4	FFA	**7.96E+00**	**8.08E-01**	-1	1
	PSO	2.53E+01	3.25E+00	1	4
	LPSO	3.46E+01	3.17E+00	1	3
	SCA	8.01E+01	3.37E+00	1	5
	HPSOFF	1.91E+01	3.02E+00	/	2
F_5	FFA	7.43E+03	5.56E+02	1	3
	PSO	1.37E+04	3.09E+03	1	4
	LPSO	2.92E+03	9.22E+02	1	2
	SCA	3.81E+07	2.30E+07	1	5
	HPSOFF	**3.02E+02**	**8.29E+01**	/	1
F_6	FFA	5.25E+02	3.21E+00	1	4
	PSO	4.33E+01	2.02E+01	1	3
	LPSO	**5.68E+00**	**1.09E+00**	-1	1
	SCA	4.17E+03	4.67E+03	0	5
	HPSOFF	4.17E+01	2.23E+01	/	2
F_7	FFA	**5.47E-02**	**9.10E-03**	-1	1
	PSO	1.56E+02	4.10E+01	1	5
	LPSO	1.08E+01	3.59E+00	1	3
	SCA	5.03E+01	3.83E+01	1	4
	HPSOFF	1.92E-01	2.51E-02	/	2
F_8	FFA	-2.54E+04	1.22E+03	1	4
	PSO	**-2.21E+04**	1.44E+03	-1	1
	LPSO	-2.38E+04	1.97E+03	1	3
	SCA	-7.36E+03	**5.73E+02**	1	5

续表 4-5：

	HPSOFF	-2.22E+04	2.18E+03	/	2
	FFA	5.10E+02	6.40E+02	1	4
	PSO	9.17E+02	**2.76E+01**	1	5
F_9	LPSO	4.47E+03	3.73E+01	1	3
	SCA	2.27E+02	1.32E+03	1	2
	HPSOFF	**1.71E+02**	5.32E+01	/	1
	FFA	3.96E+00	**6.64E-02**	1	2
	PSO	4.40E+00	2.53E-01	1	4
F_{10}	LPSO	4.03E+00	6.94E-01	1	3
	SCA	1.92E+01	4.35E+00	1	5
	HPSOFF	**3.05E+00**	3.57E-01	/	1
	FFA	4.14E+00	1.98E-01	1	4
	PSO	7.63E-01	2.72E-02	1	3
F_{11}	LPSO	4.33E-01	8.31E-02	1	2
	SCA	4.54E+01	4.12E+00	1	5
	HPSOFF	**5.07E-02**	**3.13E-02**	/	1
	FFA	**9.93E-01**	**1.01E-01**	-1	1
	PSO	2.31E+01	6.76E+00	1	4
F_{12}	LPSO	1.35E+01	3.13E+00	1	3
	SCA	1.61E+08	9.74E+07	1	5
	HPSOFF	2.15E+00	8.23E-01	/	2
	FFA	1.38E+01	9.91E-01	-1	1
	PSO	5.99E+01	5.96E+01	-1	2
F_{13}	LPSO	1.55E+02	2.95E+01	1	4
	SCA	1.84E+08	8.99E+07	1	5
	HPSOFF	6.19E+01	1.01E+01	/	3
Algorithms	**PSO**	**LPSO**	**FFA**	**SCA**	**HPSOFF**
AvgRank	**3.54**	**2.61**	**2.62**	**4.46**	**1.77**
Rank	**4**	**2**	**3**	**5**	**1**
W-test	**10/0/3**	**11/0/2**	**8/0/5**	**12/1/0**	**/**

注:1/0/-1 表示 HPSOFF 性能优于、等于和劣于对比算法，8/0/5 表示 HPSOFF 算法与对比算法相比占优、相等、劣势分别为 8、0、5。

4.3.2 实际应用

1）调频(FM)声波的参数估计

FM 声波合成在现代音乐系统中起着重要作用。该工程应用被认为是一个典型的六维优化问题，总共涉及到六个决策变量，即 $x=(a_1,w_1,a_2,w_2,a_3,w_3)$。其目的是产生一个类似于声音 2 的声音 1。该问题是一个高度复杂的多模态问题，具有很强的上位性，最小误差为 0，常被用来评估算法寻优性能，估计声音和目标声音的表达式如下：

$$y_0(t)=sin(5t\theta-1.5sin(4.8t\theta+2sin(4.9t\theta))) \tag{4.23}$$

$$y(t)=a_1sin(w_1t\theta+a_2sin(w_2t\theta+a_3sin(w_3t\theta))) \tag{4.24}$$

式中： $\theta=2\pi/100$，参数范围为[-6.4 6.35]，适应度函数定义为：

$$Min \quad f_1(x)=\sum_{t=0}^{100}(y(t)-y_0(t))^2 \tag{4.25}$$

表 4-6 调频问题的结果比较

Algorithms	Error in objective function			
	Min	Max	Mean	Std
JADE	**0**	**13.92**	7.55	26.18
FIPS	**0**	15.11	5.93	25.75
G-CMA-ES	3.326	55.09	38.75	16.77
HPSOFF	**0**	16.54	**2.66**	**3.32**

为了验证 HPSOFF 算法的准确性及稳定性，将所提算法与 JADE、FIPS 及 G-CMA-ES 算法进行对比实验。由表 4-6 可以看出，在 4 种算法中，只有 HPSOFF、JADE 和 FIPS 有可能找到最小值 0，并且在这 3 种算法中，HPSOFF 算法的平均值和标准差具有显著优势，证明 HPSOFF 算法可以较好的解决实际问题。

2）三杆桁架设计问题

该问题被认为是一个致力于最小化重量的实际工程问题，优化目标是找到两个参数的最优值。数学模型如下：

$$\begin{aligned} Min\quad & f_2(x) = L\times(2\sqrt{2}x_1 + x_2) \\ s.t.\quad & \frac{\sqrt{2}x_1 + x_2}{2x_1x_2 + \sqrt{2}x_1^{\,2}}P \le \delta \\ & \frac{x_2}{2x_1x_2 + \sqrt{2}x_1^{\,2}}P \le \delta \\ & \frac{1}{x_1 + \sqrt{2}x_2}P \le \delta \\ & 0.01 \le x_1, x_2 \le 1 \\ & L = 100,\ P=2,\ \delta=2 \end{aligned} \tag{4.26}$$

表 4-7 三杆桁架设计问题的结果比较

Algorithm	X_1	X_2	Optimal cost
GOA	0.78889	0.40762	263.8959
SCA	0.78669	0.41426	263.9348
HPSOFF	**0.78866365**	**0.408280786**	**263.895844**

为了验证算法寻优性能，采用多算法对三杆桁架设计问题进行求解，HPSOFF 和其他一般元启发式算法得到的最优解见表 4-7。根据表 4-7 中数据可以看出，所提算法能够找到最合适的决策变量，得出最优适应度值 263.895844，体现了 HPSOFF 算法平衡粒子勘探和开发的能力。

3）压力容器设计问题

压力容器设计问题是日常生活中经常遇到的一类实际工程问题，其通常被抽象地理解为寻找最优值的问题，原理如图 4-9 所示。该问题涉及四个决策变量：壳体厚度、封头厚度、内半径和圆柱壳长度，前两个决策变量是离散的且为 0.0625 的倍数，后两个决策变量是连续的。因此，该问题通常被视为混合整数优化问题，其数学模型如下：

$$
\begin{aligned}
&Min\ \ f_3(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^{\,2} + 19.84x_1^{\,2}x_3 + 3.1661x_1^{\,2}x_4 \\
&s.t.\ \ g_1(x) = 0.0193x_3 - x_1 \le 0 \\
&\quad x = (x_1, x_2, x_3, x_4) = (T_s, T_h, R, L) \\
&\quad g_2(x) = 0.00954x_3 - x_2 \le 0 \\
&\quad g_3(x) = 1296000 - \frac{4}{3}\pi x_3^{\,3} - \pi x_3^{\,2}x_4 \le 0 \\
&\quad g_4(x) = x_4 - 240 \le 0 \\
&\quad 1\times 0.0625 \le x_1, x_2 \le 99\times 0.0625 \\
&\quad 10 \le x_3, x_4 \le 200
\end{aligned}
\tag{4.27}
$$

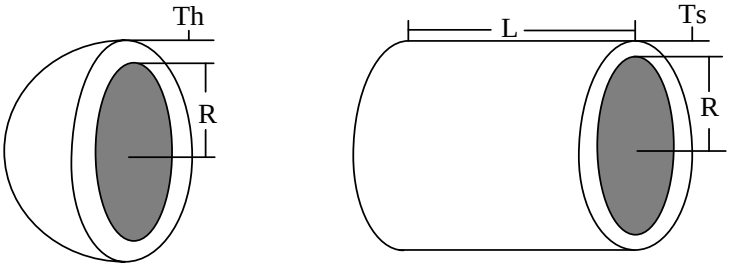


图 4-9 压力容器设计图

表 4-8 压力容器设计问题的结果比较

Algorithm	Optimal decision variables				Optimum cost
	X_1	X_2	X_3	X_4	
ISCA	0.8125	0.4375	42.09842	176.6382	6059.7457
GSA	1.1250	0.6250	55.9887	84.4542	8538.8360
PSO	0.8125	0.4375	42.0913	176.7465	6061.0780
Lagrangian multiplier	1.1250	0.6250	58.2910	43.6900	7198.0430
HPSOFF	0.8125	0.4345	42.0533	176.0036	**6027.9168**

为了探求 HPSOFF 算法解决混合整数优化问题的能力，将所提算法与 ISCA、PSO、GSA 及 Lagrangian multiplier 等算法进行对比实验。如表 4-8 所示，HPSOFF 算法在约束条件下可以准确识别出最优解，显著提高收敛精度。

4.4 基于 HPSOFF 算法的微电网系统多目标调度

1）基于 IPSO-BP 算法的风光、负荷预测

为了实现微电网系统的有效调度，需要提前对所在地区的负荷进行预测，所涉及到的模型相关参数有：典型夏季日时用户对负荷的需求情况及光电和风电的产量情况。

① 改进的粒子群算法：IPSO

基本粒子群算法是一种群体协作的搜索算法，其速度和位置更新公式为：

$$v_{i,j}^{t+1} = w \cdot v_{i,j}^{t} + c_1 r_1 (pbest_{i,j}^{t} - x_{i,j}^{t}) + c_2 r_2 (gbest_{i,j}^{t} - x_{i,j}^{t}) \tag{4.28}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1} \tag{4.29}$$

式中：w 为惯性权重；c_1 和 c_2 为学习因子；r_1 和 r_2 为两个分布在[0,1]之间的随机数；$v_{i,j}^{t}$、$x_{i,j}^{t}$、$pbest_{i,j}^{t}$ 及 $gbest_{i,j}^{t}$ 分别表示在第 t 次迭代中第 i 粒子在第 j 维的速度分量、位置分量、个体最优和全局最优。

为了降低种群陷入局部最优的风险，采用非线性惯性权重和自适应学习因子来提高 PSO 算法寻找全局最优解的能力，引入的非线性惯性权重具体如下：

$$w = w_{inital} - \sin(\frac{\pi t}{2T}) + m(\frac{t}{2T})^2 + n(\frac{t}{2T}) + \varepsilon \tag{4.30}$$

式中：w_{inital} 为惯性权重初值，一般取值为 1；t 为当前的迭代次数；T 是最大迭代次数；m、n、ε 代表的是极小的随机数，最多不能超过粒子速度上限的 0.01。

式(4.30)中所提出的非线性惯性权重能够在算法迭代早期快速地达到相应的惯性权重，有助于加强粒子群算法的全局寻优能力，减少迭代时间。此外，随着寻优的不断进行，粒子的速度会逐渐减慢，为了能够让粒子速度不会为 0，式(4.30)中的后半部分可以给定粒子一个很小的速度，在算法前期几乎可以忽略不计，但随着算法的进行，在后期能够保证粒子依旧具有一定的速度向着全局最优前进，降低粒子陷入局部最优的可能性。

此外，引入了一种反正弦调整因子策略，由于反正弦函数的值域区间取值为 $[-\pi/2, \pi/2]$，可以通过式(4.31)和(4.32)来达到更新学习因子的目的。具体如下：

$$c_1 = c_{1s} + (\frac{\pi}{2} - \arcsin(\frac{-2t+1}{T})) \times (c_{1e} - c_{1s}) \tag{4.31}$$

$$c_2 = c_{2s} + (\frac{\pi}{2} - \arcsin(\frac{-2t+1}{T})) \times (c_{2e} - c_{2s}) \tag{4.32}$$

式中：学习因子 c_1 的初始值 $c_{1s} = 2.5$；终值 $c_{1e} = 0.5$；学习因子 c_2 的初始值 $c_{2s} = 0.5$；终值 $c_{2e} = 2.5$。学习因子的不断更新与进化，能够有效保证种群多样

性，提升粒子跳出局部最优的能力。

② IPSO 优化 BP 步骤

对于 BP 神经网络而言，利用 IPSO-BP 寻优的结合方式为：将 BP 模型的关键参数（权重和阈值）作为粒子在各个维度上的寻优结果，通过上式对粒子的速度以及位置进行不断的更新，直至满足误差阈值为止。流程图如图 4-10 所示：

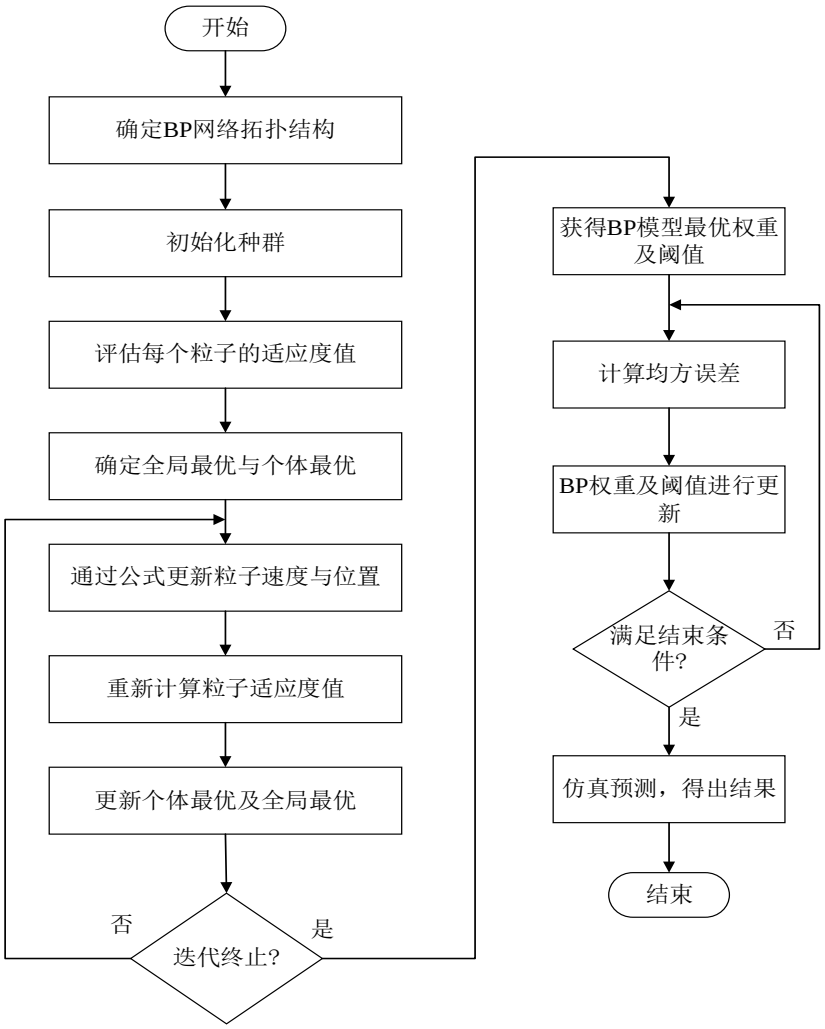


图 4-10 IPSO 寻找 BP 最优参数流程图

③ 实验仿真及分析

本节以某小区一户家庭每天用电量作为数据集，具体是将 2018 年 01 月 01 日-2018 年 03 月 20 日的最高温度、最低温度、平均温度、相对湿度、下雨量的多少以及最终的用电负荷分别作为训练样本的输入和输出，来对训练样本进行训练，真实考虑了气象因素对于负荷预测精度的影响。此外，选取 2018 年 03 月 21 日~2018 年 04 月 15 日的数据为测试样本（即预测样本），通过比较适应度值和实际输出值的均方差来评价预测模型的优劣。最后分别选用随机森林、BP、

GA-BP 以及 IPSO-BP 分别进行负荷预测，通过对比实验验证 IPSO 算法的有效性。值得一提的是，为了保证数据的公平，每组算法均运行了 30 次并取其平均值，具体如图 4-11 至 4-13 所示：

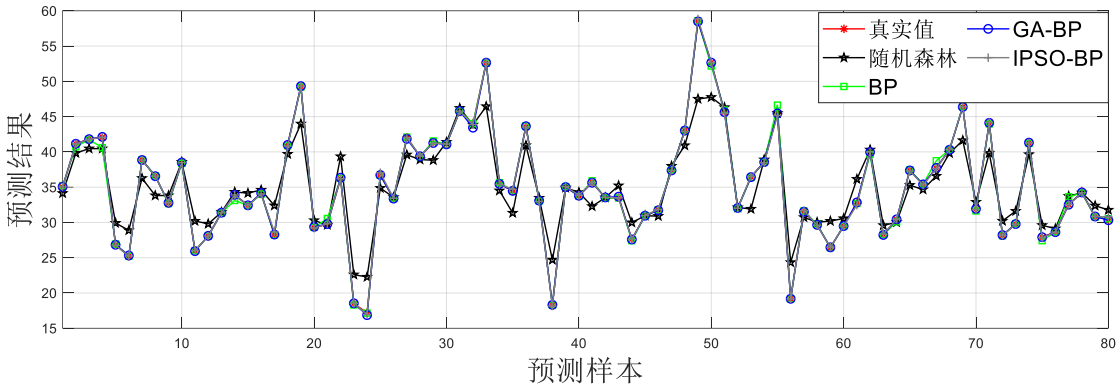


图 4-11 训练集预测结果对比

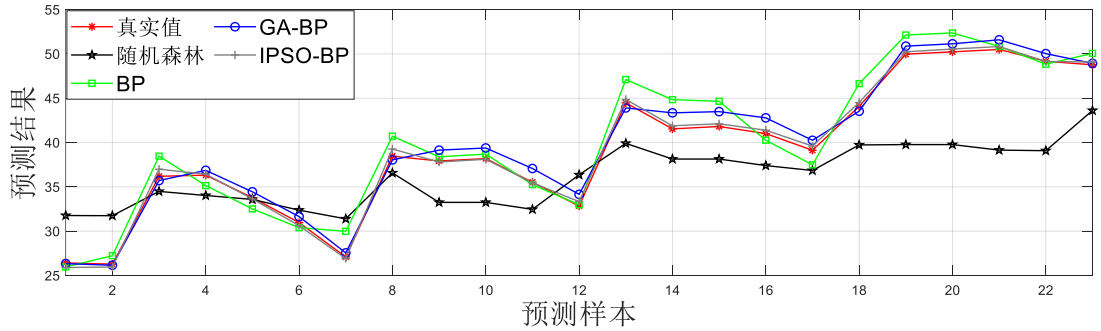


图 4-12 测试集预测结果对比

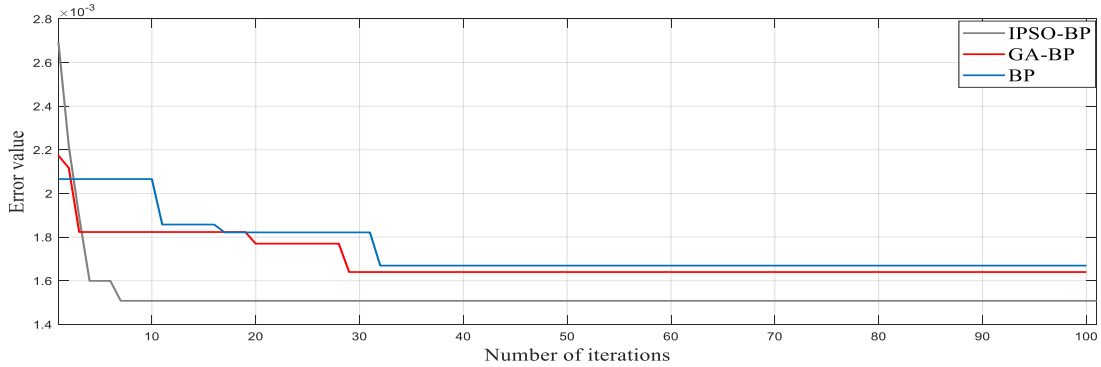


图 4-13 测试集迭代误差结果对比

从图中可以看出，面对相同的数据集，四种算法得出的实际值与预测值之间的误差却是不同的，其中随机森林的预测效果最差，预测的曲线有着明显的波动，与实际曲线拟合的效果不稳定。BP、GA-BP 较随机森林而言，预测精度得到改善，均方根误差达到了较低数值且预测曲线表现出较好的跟随性。而 IPSO-BP 算法预测效果最优，其拟合曲线几乎能够和实际值曲线贴合，跟随性能表现强，

较随机森林、BP、GA-BP 而言，预测精度分别有 93.23%、77.9%、61.9%的提高。

此外，通过图 4-13 可以看出 IPSO-BP 能够在保证预测精度的同时减小迭代次数，实现准确性和快速性的综合平衡。随着迭代进行可以观察出，IPSO-BP 能够快速达到最低误差并保持不变，且较 BP、GA-BP 相比误差更低。综上，说明 IPSO 能够有效的找出 BP 模型的最优权重和阈值，切实解决负荷预测低效的问题，具体评价指标如表 4-9 与 4-10 所示：

表 4-9 各算法均方误差统计结果

	RF	BP	GA-BP	IPSO-BP
训练集 RMSE	2.7961	0.3304	0.0907	**0.0900**
测试集 RMSE	5.6968	1.7507	1.0137	**0.3855**

表 4-10 各算法评价指标统计结果

	RF	BP	GA-BP	IPSO-BP
测试集 R2	0.4132	0.9446	0.9814	**0.9973**
迭代次数	87	80	42	**26**
测试集 MAE	4.7955	1.4441	0.8767	**0.3282**
测试集 MBE	-3.1306	0.8829	0.6831	**0.1740**

通过对比表 4-10 中随机森林、BP、GA-BP、IPSO-BP 算法的各项指标，可知随机森林算法的优化效果最差，所提方法效果最优，其相关系数几乎达到 1，且迭代次数最少，显著提高 BP 预测模型的预测精度。类似于负荷预测，该方法对于风光预测有着指导性的意义，同样适用于风光预测，其预测结果见图 4-14。

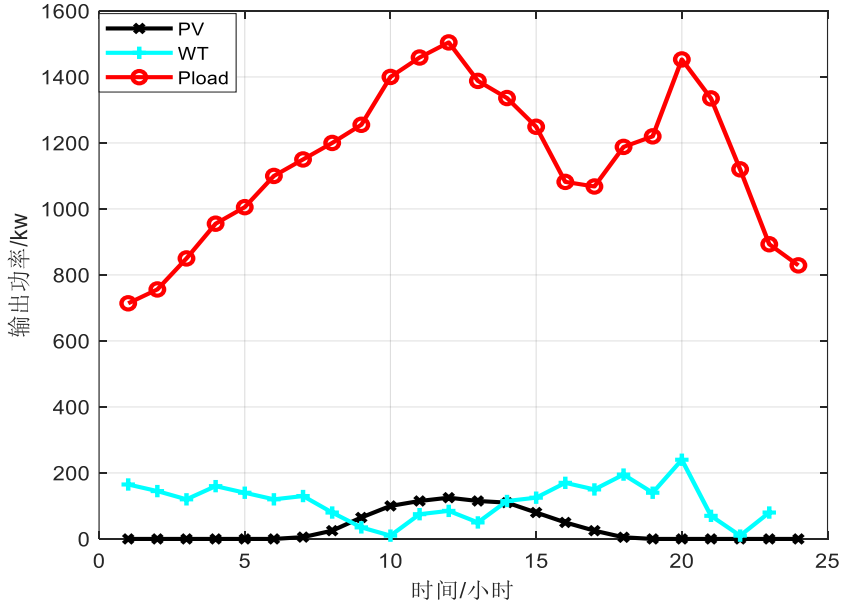


图 4-14 风光发电及负荷功率预测图

2）系统综合效益最优的优化调度

本节将对含有电动汽车灵活负载的微电网多目标优化调度模型进行探索，其流程图如图 4-15 所示。

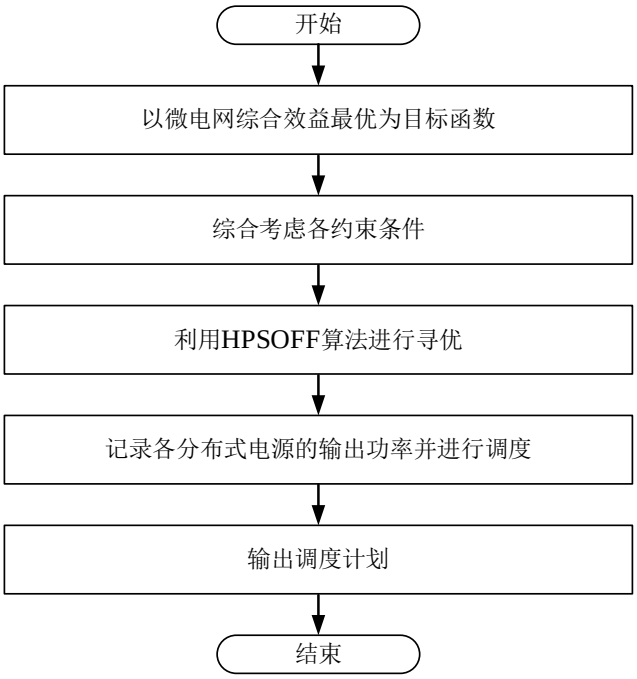


图 4-15 多目标优化模型流程图

基于 HPSOFF 算法得到微电网系统各设备的出力情况，如下图 4-16 所示。

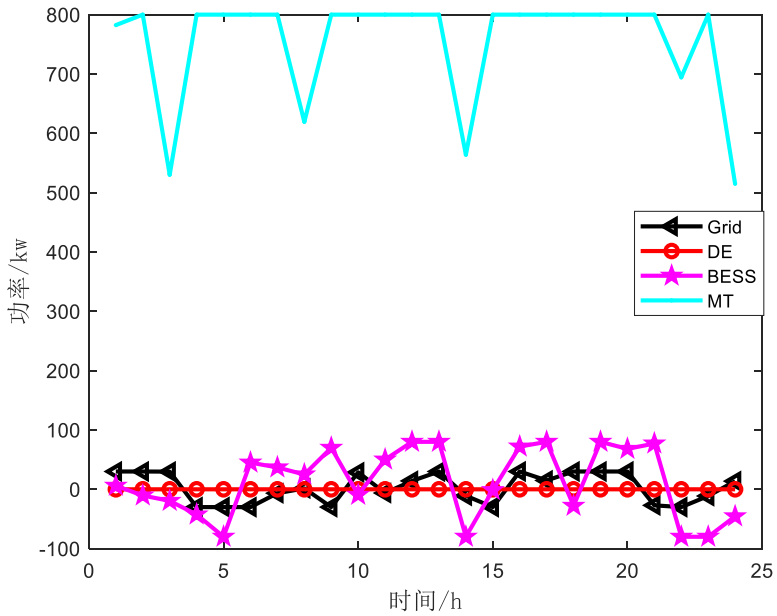


图 4-16 考虑综合效益最优的出力调度图

如前所述，MT 运行成本相对于 DE 有轻微提升，但污染物排放量以及治理费用却得到大幅度的下降，因此在 PV、WT 满发状态下，尽可能保持 MT 优先，

其扮演着环境友好型分布式电源的角色。如图 5-8 所示，MT 在整个调度过程中几乎持续保持较满发状态，在 0:00-5:00，负荷需求呈上升走向，此时电价低廉，系统从 Grid 购买电能，此举有益于同时减少运行费用及污染物治理成本，此时系统的总输出功率足以满足用电需求，多余电能储存于蓄电池。6:00-10:00 时段，用户负荷需求递增，为了满足居民的生产和生活需求，在 5:00 时储能系统的存储容量达到最大值，考虑此时电价上涨，系统减少向 Grid 进行购电操作，BESS 开始放电以满足生产生活需求，PV、WT 及 MT 也在不同程度稳定出力，剩余电量进行售电操作，补贴收益。10:00-13:00 时段，在负载激增的情况下，由于 MT 装置既能保证稳定输出又可以大幅度减少污染物治理成本，故始终以最大限度进行出力；此外，为了降低系统运行费用，BESS 不间断执行放电操作并与 PV、WT 及 MT 协作出力，共同保证系统的安全运行，其余不足电量则由大电网进行补给；最后，鉴于 DE 装置对环境极其不友好，因此在 PV、WT、BESS 及 MT 均接近满发时，无论此时电价高低，首选大电网进行补给，进而提高综合效益。在 14:00，负荷下降，PV、WT 和 MT 足以保证系统平稳运行且剩有余量，此时 BESS 进入充电状态，以备不时之需，系统将部分电能出售给大电网以获取利润。在 15:00-18:00，MT 作为绝对主力进行调度，此时系统较为平衡。在 19:00-21:00，考虑到此时进入晚高峰阶段，居民用电量出现大幅度提升，此时 PV、WT 及 MT 处于几乎满发状态，BESS 持续大功率放电，不足部分通过向大电网购电进行补给，所有分布式电源共同出力保持系统平稳运行。在 22:00-24:00，负荷减少，MT 迅速降低出力功率以减少运行成本，Grid 进入售电状态，降低污染物治理费用的同时增加收益，BESS 则进入充电状态。

4.5 本章小结

为了避免出现过早收敛现象，探究了一种融合 PSO 和 FFA 的混合算法。首先将 PSO-NIWAF 算法与 FFA-DP 相结合，既保留了两者的优点，同时避免陷入局部最优；之后，通过五种算法进行对比实验充分验证了 HPSOFF 算法的探索、开发性能；最后采用所提算法求解 23 个经典函数、实际工程应用及考虑综合经济效益最优的多目标优化调度模型，数值结果表明 HPSOFF 算法能够准确寻优，有效解决复杂多目标优化问题。

第 5 章 总结与展望

5.1 总结

为了推动新能源系统的发展，实现分布式电源的高效利用，本文建立了含电动汽车柔性负荷的并网微电网系统并提出两类目标函数；同时，为克服 PSO 算法后期迭代时间长及易陷入局部最优等缺点，探究了两种新型 PSO 算法，并将其分别用于运行费用最小、污染物治理成本最低的单目标优化调度及综合效益最优的多目标优化调度，从而实现最优能源分配。具体研究内容如下：

1）构建了含电动汽车灵活负载的微电网并网优化调度模型，阐明其运行机理和数学模型；之后，基于各类约束条件建立系统运行费用最小、污染物治理成本最低的单目标优化及系统综合效益最优的多目标优化函数。

2）研究了 APSOIIM 算法，并给出其详细全局收敛性证明。此外，通过 CEC2014、CEC2017 测试集及实际工程应用验证了所提算法的有效性，并将其应用于微网系统的单目标优化问题中，实验结果表明 APSOIIM 算法能够有效解决单目标优化问题。

3）探求了一种基于动态种群策略和衰减因子惯性权重机制的 HPSOFF 混合优化算法。通过 23 个数值仿真验证了所提算法优异的收敛精度及迭代速度，并利用 HPSOFF 算法对微电网进行多目标优化，最终实现系统综合效益最大化。

5.2 展望

本文以含电动汽车灵活负荷的微电网调度模型为研究对象，对两类单目标问题及多目标优化进行深入研究并根据各分布式电源出力情况实现经济调度，目前已取得阶段性进展，但仍存在若干问题有待深入探索，如：

1）相比于传统 PSO 算法，基于动态种群策略及非线性惯性权重机制的新型 HPSOFF 算法能够显著提高寻优性能，但所提算法的勘探-开发能力及种群变化曲线值得进一步深入研究。

2）目前仅进行了单个微电网系统的经济优化调度，今后可以研究微电网群智能优化调度，并采取不同的优化调度模型，探求微电网群之间的相互作用并加

入博弈论思想。

3）设计微电网系统模型时，未考虑居民满意度、模型复杂度等因素，今后有条件可以对含电动汽车的微电网系统模型进行深入探索。

参考文献

[1] 宋璇坤,韩柳,鞠黄培等. 中国智能电网技术发展实践综述[J].电力建设, 2016,37(07):1-11.

[2] 苗中磊,蔡逢煌,王群兴等. 分布式直流微电网分级控制技术综述[J].电源学报, 2019,17(06):115-127.

[3] Shi R, Li S, Sun C, et al. Adjustable robust optimization algorithm for residential microgrid multi-dispatch strategy with consideration of wind power and electric vehicles[J]. Energies, 2018, 11(8): 2050-2072.

[4] 马跃,孟润泉,魏斌等. 考虑阶梯式碳交易机制的微电网两阶段鲁棒优化调度[J].电力系统保护与控制, 2023,51(10):22-33.

[5] Wu X, Cao W, Wang D, et al. A Multi-objective optimization dispatch method for microgrid energy management considering the power loss of converters[J]. Energies, 2019, 12(11): 2160-2175.

[6] Kumar A, Sharma S, Verma A. Optimal sizing and multi-energy management strategy for PV-biofuel-based off-grid systems[J]. IET Smart Grid, 2019, 3(1): 83-97.

[7] Leng C, Yang H, Song Y, et al. Expected value model of microgrid economic dispatching considering wind power uncertainty[J]. Energy Reports, 2023, 9(10): 291-298.

[8] Wang H, Wu X, Sun K, et al. Research on the optimal economic power dispatching of a multi-microgrid cooperative operation[J]. Energies, 2022, 15(21): 8194-8207.

[9] Jiang K, Wu F, Zong X, et al. Distributed dynamic economic dispatch of an isolated AC/DC hybrid microgrid based on a finite-step consensus algorithm[J]. Energies, 2019, 12(24): 4637-4655.

[10] Mu Y, Jia H, et al. Dynamic economic dispatch of a hybrid energy microgrid considering building based virtual energy storage system[J]. Applied energy, 2017, 194(1): 386-398.

[11] Hou H, Xue M, Xu Y, et al. Multi-objective economic dispatch of a microgrid considering electric vehicle and transferable load[J]. Applied Energy, 2020, 262(8): 114489-114492.

[12] Du Y, Pei W, Chen N, et al. Real-time microgrid economic dispatch based on model predictive control strategy[J]. Journal of modern power systems and clean energy, 2017, 5(5): 787-796.

[13] Zhang G, Wang W, Du J, et al. Multi-objective Economic Optimal Dispatch for the Island Isolated Microgrid under Uncertainty Based on Interval Optimization[J]. Mathematical Problems in Engineering, 2021, 21(2): 1-14.

[14] Yeh W C, He M F, Huang C L, et al. New genetic algorithm for economic dispatch of stand-alone three-modular microgrid in DongAo Island[J]. Applied Energy, 2020, 263(1): 114508-114528.

[15] Zhang M, Gan M, Li L. Sizing and siting of distributed generators and energy storage in a

microgrid considering plug-in electric vehicles[J]. Energies, 2019, 12(12): 2293-2315.

[16] Xu J, Yan F, Yun K, et al. Noninferior solution grey wolf optimizer with an independent local search mechanism for solving economic load dispatch problems[J]. Energies, 2019, 12(12): 2274-2290.

[17] Guo F, Li G, Wen C, et al. An accelerated distributed gradient-based algorithm for constrained optimization with application to economic dispatch in a large-scale power system[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 51(4): 2041-2053.

[18] Sakthivel V P, Suman M, Sathya P D. Squirrel search algorithm for economic dispatch with valve-point effects and multiple fuels[J]. Energy Sources, Part B: Economics, Planning, and Policy, 2020, 15(6): 351-382.

[19] Wang S, Fan X, Han L, et al. Improved interval optimization method based on differential evolution for microgrid economic dispatch[J]. Electric Power Components and Systems, 2015, 43(16): 1882-1890.

[20] Peng Y, Jiang W, Wei X, et al. Microgrid Optimal Dispatch Based on Distributed Economic Model Predictive Control Algorithm[J]. Energies, 2023, 16(12): 4658-4675.

[21] Zhang Z, Wang Z, Cao R, et al. Research on two-level energy optimized dispatching strategy of microgrid cluster based on IPSO algorithm[J]. IEEE Access, 2021, 9(3): 120492-120501.

[22] Zhao F, Yuan J, Wang N. Dynamic economic dispatch model of microgrid containing energy storage components based on a variant of NSGA-II algorithm[J]. Energies, 2019, 12(5): 871-883.

[23] Du X, Wang L, Zhao J, et al. Power dispatching of multi-microgrid based on improved CS aiming at economic optimization on source-network-load-storage[J]. Electronics, 2022, 11(17): 2742-2755.

[24] Gao R, Wu J, Hu W, et al. An improved ABC algorithm for energy management of microgrid[J]. International Journal of Computers Communications & Control, 2018, 13(4): 477-491.

[25] Zhang Y, Lv Y, Zhou Y. Research on Economic Optimal Dispatching of Microgrid Based on an Improved Bacteria Foraging Optimization[J]. Biomimetics, 2023, 8(2): 150-173.

[26] Basak S, Dey B, Bhattacharyya B. Demand side management for solving environment constrained economic dispatch of a microgrid system using hybrid MGWOSCACSA algorithm[J]. CAAI Transactions on Intelligence Technology, 2022, 7(2): 256-267.

[27] Du X, Wang L, Zhao J, et al. Power dispatching of multi-microgrid based on improved CS aiming at economic optimization on source-network-load-storage[J]. Electronics, 2022, 11(17): 2742-2760.

[28] Elattar E E. Modified harmony search algorithm for combined economic emission dispatch of microgrid incorporating renewable sources[J]. Energy, 2018, 159(3): 496-507.

[29] Bentouati B, El-Sehiemy R A, Chettih S, et al. A Chaotic Krill Herd Optimizer for Efficient

Combination of Renewable Energy Sources in Isolated Microgrid Mode[J]. Electric Power Components and Systems, 2022, 49(18-19): 1445-1462.

[30] He Y, Wang W, Wu X. Multi-agent based fully distributed economic dispatch in microgrid using exact diffusion strategy[J]. IEEE Access, 2019, 8(1): 7020-7031.

[31] 周辉,张玉,肖烈禧等. 基于改进秃鹰算法的微电网群经济优化调度研究[J].太阳能学报, 2024,45(02):328-335.

[32] 付小懿,华运韬,马文来等. 混合-拟蒙特卡洛法在地球辐射外热流计算中的效率评估[J]. 红外与激光工程, 2023,52(11):149-163.

[33] 魏利胜,杨奔奔,孙瑞霞. 基于新型 BBO 算法的微电网优化调度研究[J].系统仿真学报, 2023,35(5):1075-1085.

[34] Kardani N, Bardhan A, Samui P, et al. Predicting the thermal conductivity of soils using integrated approach of ANN and PSO with adaptive and time-varying acceleration coefficients[J]. International Journal of Thermal Sciences, 2022, 173(6): 107427-107441.

[35] 潘峰,周倩,李位星. 标准粒子群优化算法的马尔科夫链分析[J].自动化学报, 2013,39(04):381-389.

[36] Kennedy J, Eberhart R. Particle swarm optimization (PSO)[C]//Proc. IEEE International Conference on Neural Networks, Perth, Australia. 1995: 1942-1948.

[37] Özsoy V S. The determination of the most suitable inertia weight strategy for particle swarm optimization via the minimax mixed-integer linear programming model[J]. Engineering Computations, 2021, 38(4): 1933-1954.

[38] Nabi S, Ahmad M, Ibrahim M, et al. AdPSO: adaptive PSO-based task scheduling approach for cloud computing[J]. Sensors, 2022, 22(3): 920-931.

[39] Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1): 58-73.

[40] Li M, Chen H, Wang X, et al. An improved particle swarm optimization algorithm with adaptive inertia weights[J]. International Journal of Information Technology & Decision Making, 2019, 18(03): 833-866.

[41] Xu H Q, Gu S, Fan Y C, et al. A strategy learning framework for particle swarm optimization algorithm[J]. Information Sciences, 2023, 619(1): 126-152.

[42] Kılıç H, Yüzgeç U. Tournament selection based antlion optimization algorithm for solving quadratic assignment problem[J]. Engineering Science and Technology, An International Journal, 2019, 22(2): 673-691.

[43] Yang H, Chen T, Huang N. An adaptive bird swarm algorithm with irregular random flight and its application[J]. Journal of Computational Science, 2019, 35(2): 57-65.

[44] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm[J]. Information Sciences, 2009, 179(13): 2232-2248.

[45] Huang L, Qin C. A novel modified gravitational search algorithm for the real world

optimization problem[J]. International Journal of Machine Learning and Cybernetics, 2019, 10(5): 2993-3002.

[46] Culley J M, Donevant S, Craig J, et al. Validation of a novel irritant gas syndrome triage algorithm[J]. American Journal of Disaster Medicine, 2018, 13(1): 13-26.

[47] Igel C, Hansen N, Roth S. Covariance matrix adaptation for multi-objective optimization[J]. Evolutionary Computation, 2007, 15(1): 1-28.

[48] Zhang H G, Tang M Z, Liu Y A, et al. Sardine Optimization Algorithm with Agile Locality and Globality Strategies for Real Optimization Problems[J]. Arabian Journal for Science and Engineering, 2022, 48(8): 9787-9825.

[49] 张忠伟,王金玉,张建波等. 考虑概率潮流的分布式电源优化配置[J].重庆大学学报, 2018,41(12):83-91.

[50] 秦海勤,赵杰,任立坤等. 基于改进 L-SHADE 算法的航空发动机性能退化评估[J].航空学报, 2023,44(14):169-182.

[51] Kumar A, Misra R K, Singh D. Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase[C]//2017 IEEE Congress on Evolutionary Computation (CEC). IEEE, Donostia-San Sebastian, Spain, 5-8 June, 2017: 1835-1842.

[52] Xu Z, Gao S, Yang H, et al. SCJADE: Yet Another State-of-the-Art Differential Evolution Algorithm[J]. IEEE Transactions on Electrical and Electronic Engineering, 2021, 16(4): 644-646.

[53] Choi T J, Ahn C W. An improved LSHADE-RSP algorithm with the Cauchy perturbation: iLSHADE-RSP[J]. Knowledge-Based Systems, 2021, 215(1): 106628-106648.

[54] Aranguren I, Valdivia A, Morales-Castañeda B, et al. Improving the segmentation of magnetic resonance brain images using the LSHADE optimization algorithm[J]. Biomedical Signal Processing and Control, 2021, 64(9): 102259-102274.

[55] Agushaka J O, Ezugwu A E, Abualigah L, et al. Efficient initialization methods for population-based metaheuristic algorithms: A comparative study[J]. Archives of Computational Methods in Engineering, 2023, 30(3): 1727-1787.

[56] Minh H L, Khatir S, Rao R V, et al. A variable velocity strategy particle swarm optimization algorithm (VVS-PSO) for damage assessment in structures[J]. Engineering with Computers, 2023, 39(2): 1055-1084.

[57] 张孟健,龙道银,王霄等. 基于马尔科夫链的灰狼优化算法收敛性研究[J].电子学报, 2020,48(08):1587-1595.

[58] 胡婷婷,贺兴时,杨新社. 基于萤火虫算法的 Markov 模型及收敛性分析[J].纺织高校基础科学学报, 2014,27(04):496-501.

[59] Kumar V, Kumar D. A systematic review on firefly algorithm: past, present, and future[J]. Archives of Computational Methods in Engineering, 2021, 28(4): 3269-3291.

[60] Abualigah L, Diabat A. Advances in sine cosine algorithm: a comprehensive survey[J].

Artificial Intelligence Review, 2021, 54(4): 2567-2608.

[61] Abualigah L, Elaziz M A, Hussien A G, et al. Lightning search algorithm: a comprehensive survey[J]. Applied Intelligence, 2021, 51(4): 2353-2376.

[62] Chu X, Gao D, Chen J, et al. Adaptive differential search algorithm with multi-strategies for global optimization problems[J]. Neural Computing and Applications, 2019, 31(12): 8423-8440.

[63] Gholami J, Ghany K K A, Zawbaa H M. A novel global harmony search algorithm for solving numerical optimizations[J]. Soft Computing, 2021, 25(4): 2837-2849.

攻读学位期间发表的学术论文

[1] ***Liu Rui***, Wei Lisheng, Zhang Pinggai. An Adaptive Particle Swarm Optimization with Information Interaction Mechanism[J]. Machine Learning: Science and Technology, 2023. **(SCI 二区,三审小修)**

[2] ***Liu Rui***, Wei Lisheng, Zhang Pinggai. An Efficient Hybrid Particle Swarm Optimization and Firefly Algorithm[J]. The Journal of Supercomputing, 2023. **(SCI 二区，已投稿:8672a9a1-d01c-4496-94d5-4a1314f645fc)**

[3] ***Liu Rui***, Wei Lisheng, Wang Ruiren. Brief-term Electricity Load Forecasting Based on PSO-optimized BP Neural Network[C]//2023 3nd International Conference on Computer, Control and Robotics (ICCCR2023). IEEE, ShangHai, China, 24-26 March, 2023: 404-408. **(EI: 20233414619610)**

[4] ***Liu Rui***, Wei Lisheng, Zhang Pinggai. Extreme Learning Machine Classifier Based on Nover Particle Swarm Optimization Algorithm[C]//The 38th Youth Academic Annual Conference of Chinese Association of Automation (YAC2023), Hefei, China, 27-29 August, 2023: 1005-1009. **(EI: 20240815614423)**

[5] Penxian Zhang, Zhang Yan, ***Liu Rui***. Stability Analysis of Time-varying Delay Systems Based on Generalized Free Weight Integral Inequality[C]//The 38th Youth Academic Annual Conference of Chinese Association of Automation (YAC2023), Hefei, China, 27-29 August, 2023. **(EI: 20240815614590)**

攻读学位期间获得的奖励

[1] 获得安徽工程大学**校级优秀毕业生奖项**

[2] 获得 2021-2022 学年安徽工程大学研究生**二等奖学金**

[3] 获得 2022-2023 学年安徽工程大学研究生**一等奖学金**

[4] 获得 2023-2024 学年安徽工程大学研究生**一等奖学金**

[5] 获得 2022 年校第八届“互联网+”大学生创新创业大赛**研究生创意组三等奖**

致　谢

东逝流水，叶落知秋。硕士生涯不知不觉已经走到尾声，之前总觉来日方长，没想到毕业却也悄然来临。在我不再那么热烈的二十五岁，回首望去，皆是回忆，是结束，也是开始。

师恩难忘，山高水长。真诚感谢魏利胜老师三年来对我的关心和指导，从一开始的讨论选题、小论文的稿件修改及毕业论文的撰写，魏老师一直耐心的对我进行指导，总能在我迷茫的时候用简单的话语道出深刻的道理。在生活中，魏老师十分关注学生的身心健康，经常带领实验室伙伴们聚餐交流，为学生实际解决生活上的困难。十年树木，百年树人，魏老师的教诲使我受益终生。

桃李不言，下自成跃。衷心感谢张平改老师在学术、生活上提供的帮助和指导。与张老师于研二初相识，在为数不多的时间里，其严谨的学术态度、扎实的理论知识及耐心的指导令我折服。祝愿张老师桃李天下。

山水一程，三生有幸。很幸运读研期间能够遇到 509 实验室的伙伴们，三年时光转瞬即逝，大家在一起吃饭聊天的快乐时光却还历历在目。非常有幸能够与你们共同经历人生中宝贵的一段旅程，感谢大家给予的帮助与支持，前路漫漫亦灿灿，祝愿所有人前程似锦。同时衷心感谢室友胡宇杨、王皖北、黄志远同学在平日里的关心和帮助。

读研不易，父母的支持与鼓励一直是我最大的动力，每当迷茫困惑时，和家人的几通电话，有时烦恼便也烟消云散了。感谢你们在我求学生涯中默默站在背后，做我坚强的后盾。

感谢我的女朋友张佩娴，七年来—路扶持相伴，互助成长，愿未来少年郎如愿迎娶他的新娘。

最后，向参加论文评阅、答辩及提出宝贵意见的专家、老师致以衷心的感谢！

刘瑞

安徽工程大学电气工程学院

二 O 二四年四月

尚德教学 唯实惟新