江西财经大学
JIANGXI UNIVERSITY OF FINANCE AND ECONOMICS

学校代码
密级
中图分类号
UDC

硕士学位论文
MASTER DISSERTATION

论文题目（中文）：萤火虫优化算法的研究及其应用

论文题目（英文）：Research on Firefly Algorithm and Its Application

作　　者　刘桂元　　导　　师　汪 靖　副教授
申请学位　专业学位硕士　　培养单位　江西财经大学
学科专业　软件工程　　研究方向　智能优化算法

二〇二〇 年 三 月

目录

CONTENTS

摘要

作为典型的群智能优化算法，萤火虫优化算法（Firefly Algorithm，FA）因其具有良好的寻优搜索能力，在计算机、工程、管理以及经济等领域都有广泛应用。FA 通过模拟萤火虫种群的生物特性而设计出来用于求解最优化问题的随机优化算法。其在具有实现简单，需调节参数少，实用性强的同时，也具有收敛慢，易陷于局部最优，离散优化弱等缺点。因此，该算法还有较大的改进空间。本文对 FA 改进的主要工作如下：

（1）针对萤火虫优化算法的吸引移动公式中的步长和吸引力部分，提出了自适应步长以及比例模型吸引的改进策略。在自适应步长策略中，增加了控制步长变化的控制体，通过当前与上一代的最优个体位置距离来确定下一代是否进行，当距离大于设定阈值时步长不变，反之则根据步长变化公式进行步长减小，以适应当前阶段对步长大小的需求。而在比例模型吸引力改进中，由于原标准萤火虫优化算法中吸引力部分的快速趋 1 性，极大地减小了种群在中后期进化的多样性，而改进后的 FA 将当前代的吸引力改为上一代的 0.4 倍，在满足进化需求的情况下，尽可能保留种群多样性。最后通过比较改进后的 FA 优化标准测试函数的实验结果数据，说明了以上两种改进策略确实有助于 FA 性能的提高。

（2）基于反向学习的萤火虫优化算法。尽管改进后的吸引力部分有助于保留种群的多样性，但出于对求解结果的精度的需求，算法在演化过程中难免会减小种群的多样性，陷入局部最优。因此，有必要引入增加多样性的机制以提高算法的全局搜索能力。为此，本文第四章中引入了反向学习策略，对每代种群中适应值最差的三个个体进行反向学习，使最差的三个个体以解空间的中心对称点为反向点进行反向变换，跳出局部最优，提高种群多样性。而实验表明，FA 在引入了反向学习策略后，提高了其全局搜索能力。

（3）因为萤火虫优化算法在初期主要是针对连续优化问题提出的，对离散问题的求解能力较弱，而为使其能应用于柔性作业车间调度的离散问题。文中提出了萤火虫个体离散化编码方案。通过将不同工件工序的加工顺序信息转化为萤火虫个体的位置信息以解决种群个体的离散化，并将目标优化时间性能赋予萤火虫个体适应值。在算法求得结果后再对个体位置信息进行解码成机器的加工信息，确定工件工序的加工机器和开始时间。最后，经过仿真实验的检验测试，说明了该改进后的编码设计算法对求解柔性作业车间调度问题的有效性。

关键词：群智能算法；萤火虫优化算法；反向学习；柔性作业车间调度

Abstract

As a typical swarm intelligence optimization algorithm, Firefly Algorithm (FA) has a wide range of applications in the fields of computer, engineering, management, and economics because of its good search ability. FA is a stochastic optimization algorithm designed to solve the optimization problem by simulating the biological characteristics of the firefly population. It has the advantages of simple implementation, few adjustment parameters, and strong practicability, but also has the disadvantages of slow convergence, easy fall into local optimization, and weak discrete optimization. Therefore, the algorithm still has much room for improvement. The main work of FA improvement in this paper is as follows:

(1) Aiming at the step size and attractive part of the attracting movement formula of the firefly optimization algorithm, an adaptive step size and an improved strategy of proportional model attraction are proposed. In the adaptive step strategy, a control body that controls the change of the step size is added to determine whether the next step size is changed based on the current optimal individual position distance from the previous generation. When the distance is greater than a set threshold, the step size does not change. On the contrary, the step size is reduced according to the step size change formula to meet the demand for the step size in the current stage. In the improvement of the attractiveness of the scale model, due to the rapid convergence of the attractive part of the original standard firefly optimization algorithm, the diversity of population evolution in the middle and late stages is greatly reduced, and the improved FA will reduce the attractiveness of the current generation. It was changed to 0.4 times of the previous generation, and the population diversity was preserved as much as possible while meeting the needs of evolution. Finally, by comparing the experimental results of the improved FA optimization standard test function, it is shown that the above two improvement strategies do help improve the performance of FA.

(2) Firefly algorithm based on opposition-based learning. Although the improved attractive part helps to preserve the diversity of the population, due to the demand for the accuracy of the solution results, the algorithm will inevitably reduce the diversity of the population during the evolution process and fall into a local optimum. Therefore, it is necessary to introduce a mechanism to increase diversity to improve the global search capability of the algorithm. To this end, the opposition-based learning strategy is introduced in Chapter 4 of this paper. The three individuals with the worst fitness values in each generation of the population are implemented opposition-based learning strategy, so that the worst three individuals use the central symmetry point of the solution space as the reverse point to jump out of the local optimal, and improve the diversity of the population. Experiments show that FA has

improved its global search ability after introducing opposition-based learning strategy.

(3) Because the firefly algorithm was mainly proposed for continuous optimization problems in the early stage, the ability to solve discrete problems is weak. In order to make it applicable to the discrete problem of flexible job shop scheduling, an individual discrete coding solution for fireflies is proposed. By transforming the processing sequence information of different workpiece processes into the position information of individual fireflies, the discreteness of the individual populations is resolved, and the target optimized time performance is given an adaptive value to the individual fireflies. After the result of the algorithm is obtained, the individual position information is decoded into the processing information of the machine, and the processing machine and start time of the workpiece process are determined. Finally, simulation tests verify the effectiveness of the improved coding design algorithm for solving flexible job shop scheduling problems.

Keyword: Optimization Algorithm; Firefly Algorithm; Opposition-based Learning; Flexible Job Shop Scheduling Problems

第 1 章 绪论

1.1 研究背景及意义

随着全球工业的高速发展，智能制造为主导的第四次工业革命对制造业的生产调度规模提出了新的挑战。鉴于现今世界中实际生产调度问题的大规模性、计算复杂性、动态随机性、多约束性和多目标性等特点，寻找更贴近实际大规模生产调度问题的新型智能优化方法已经成为相关领域理论界和工程界共同关注的热点方向。

业界早已证明生产调度问题为 NP-难问题，虽然对它的研究已有几十年的历史，但大部分研究都是集中在小规模调度问题上的应用上，而在激烈的市场竞争下，各行各业的生产规模日益增大。实际生产中，待解决的往往是数千台机器、每月数千个订单的大规模调度问题，传统的调度算法应用到大规模生产调度环境下时，计算时间将随着工件和机器数的增长成指数级倍数增加(n 个工件 m 台机器的问题包含(n!)m 种排列)， 这让实际需求已无法承受。

自从 20 世纪 80 年代以来，涌现了一批通过模拟自然现象或过程而建立起来的一类智能优化方法，如萤火虫优化算法、蚁群算法、粒子群优化算法、人工蜂群算法和人工鱼群算法等等，它们具有适于高度并行、自组织、自学习与自适应等特征，已被广大的研究人员成功建模并用于求解大量的复杂实际优化问题，并以其独特的优点和机制在生产调度领域得到广泛的应用[1-5]，逐渐成为计算智能领域的主要研究热点。国际著名期刊《InformationSciences》在 2015 年专门推出一期专刊，反映启发式智能优化算法领域在大规模优化领域里的最新进展[6]。智能优化算法的性能取决于对算法的探索(exploration)能力和开发(exploitation)能力的平衡，例如，在萤火虫优化算法中，亮度高的萤火虫吸引视野区域内亮度相对较弱的萤火虫向其移动从而构成萤火虫之间的搜索空间，同时在较弱的萤火虫向亮度高的萤火虫移动的过程中又加以由高斯分布、均匀分布或者莱维飞行[7]所得到随机步长进行扰动来平衡萤火虫的探索和开发能力。禁忌搜索算法以高概率来接受适应值好的解加强探索能力，同时又尽可能避开禁忌表中的一些点和允许一定的下山操作来增加算法开发能力；蚁群算法通过信息素加强来加强算法的探索能力，同时又通过蒸发因子来增加算法的开发性能。一旦探索能力强，开发能力弱，算法容易出现停滞现象，即在搜索的后期，易陷入局部最优；反之，算法虽然有更多的可能性找到全局最优，但是算法运行时间又过长。为此，学术界一直寻找解决这个矛盾方法。

在众多智能优化算法中，萤火虫优化算法是英国牛津大学学者 Xin-She

Yang 于 2008[8]年通过模拟自然界中真实萤火虫集体行为而提出了一种新型启发式优化算法，该算法简单高效，易于实现和操作，很快吸引了大量研究者的注意，并应用到很多实际工程优化问题中[9-12]。然而，萤火虫优化算法同样面临着与其他智能优化算法一样普遍存在的问题，即算法的时间计算复杂性随着问题规模的增大成指数倍增长。例如，申请者使用标准萤火虫优化算法，在 CPU 为酷睿 II T5200@1.6Ghz，内存为 2G 的硬件环境下，完成 CEC 2008 大规模优化竞赛上提出的 7 个问题[13]的测试大约需要近 8 个小时，完成 CEC 2010 大规模优化竞赛上提出的 20 个问题[14]的测试需要 137 个小时，而完成《Soft Computing Journal》大规模优化专刊上提出的 19 个问题[15]的测试大约需要 241 个小时。而生产调度问题本质上是一个离散组合优化问题，这类问题的解空间为离散点的集合，随着生产调度规模的增长, 解空间将指数增长而产生搜索的组合爆炸，当规模增长到一定程度，未经改进的标准萤火虫优化算法很难以求解。

而面对上述问题，则可以对萤火虫优化算法本身就行优化，如改进参数、混合策略，协同混合不同算法等，使得算法自身具备高度多样性的同时又能快速收敛。因此，本方向研究参数自适应的萤火虫优化算法在求解大规模生产调度问题具有重要意义，第一，对萤火虫优化算法的参数自适应调整展开研究可以更贴切地模拟萤火虫移动过程中的生物行为，完善萤火虫优化算法的理论，对提高萤火虫优化算法的普适性有着重大意义。第二，生产调度问题在数学上属于 NP-hard 问题，求解难度很大，大规模生产调度问题的计算复杂性更是成指数级增长，对其研究具有重要的学术意义；第三，生产调度是制造业的重要环节, 直接影响企业的效益和竞争力，高效的优化方法和调度技术对企业提高生产效率和经济效益至关重要，对其研究在工业界具有很重要的实际意义。

1.2 最优化问题及其求解方法

1.2.1 最优化问题

通常包含以下基本要素的问题称为最优化问题：

（1）一个目标函数 f：该目标函数即表示需要被寻求最大化或最小化的度量，且取目标函数的相反数的最小值时便可求得目标函数的最大值。然而，有些问题，如约束满足问题，是很难明确一个显式的目标函数，并且需要找寻一个满足其所有约束条件的解。

（2）一组未知数或变量：他们对目标函数解的优劣有直接或间接的影响。如当变量 x 代表影响因素时，f(x)则反映解 x 的优劣程度。

（3）一组约束条件：它们限制了影响因素的取值范围。大部分优化问题都具有一组约束条件，以限制相关变量的定义域。而有时甚至可以直接根据约束条

件来判断某侯选解是否为最优解。

而一个优化方法或算法的最终目的便是在解空间 S 中求得或试图求得一个满足所有约束条件的最优解。

1.2.2 最优化问题分类

根据最优化问题的不同特征可分为不同类别，如下列分类：

（1）根据影响目标函数的因素个数，只有一个影响因子的称为单变量最优化问题，多于一个影响因子的最优化问题则称作多变量最优化问题。

（2）根据变量类型分类，一个有着连续变量的最优化问题称为连续优化问题，而一个变量取值为整数的问题即为离散或整数问题，如歌一个问题既有取离散值的变量又有取连续值的变量称为混合整数问题。组合优化问题则是解是整数值变量的排列组合的问题。

（3）根据目标函数的非线性程度，当最优化问题的目标函数是线性函数时，问题为线性问题，当目标函数使用或包含二次函数及其他非线性函数时，该问题称为非线性问题。

（4）根据施加的约束条件，只进行定义域边界限制的问题为无约束的最优化问题，当其限制条件有或超过一个的等式以及非等式约束条件的称为约束最优化问题。

（5）根据最优解的个数，当问题解只有一个时，该问题为单模最优化问题，超过一个则为多模最优化问题。值得说明的是，有些迷惑最优化问题可能具有假最优解。

（6）根据优化准则的数目，如果一个目标函数就可以衡量被优化的问题，则称之为单目标问题。在问题有多于一个目标函数需要同时进行寻优的情况下，此时问题被称为多目标问题。

而对上述不同类型问题的求解，其方法或复杂或简单，或精确或近似，有着很大的不同。为此国内外大量的学者们花费巨大的精力，进行深入实验研究，在不断接受挑战的同时，也产生了很多优秀的成果并在下小节进行介绍。

1.2.3 最优化问题求解方法

在上小节中，阐述了不同最优化问题的分类，当然在用不同方法解决不同问题的同时也不排除存在同一方法可以解决不同最优化问题的情况。其中直接法，解析法和其他一些方法是常用于求解最优化问题的方法[16]。

（1）直接法，当问题不能显示描述的时候，无法用解析式求得相关条件，这时只能进行问题解的逐个探索，并通过迭代逼近问题最优解。

（2）解析法，如果在最优化问题模型中可以直接写出其解析表达式，则可以用数学解析工具进行求解。如当目标函数和约束条件都有明确的解析表达式时，

可以用数学中的微积分和拉格朗日乘子法等数学工具解决。

（3）其他方法，包括图论法以及各种智能进化优化算法，如神经网络算法（Neural Network Algorithm, NNA）、遗传算法（Genetic Algorithm, GA）、蚁群算法（Ant Colony Optimization, ACO）、禁忌搜索算法（Tabu Search, TS）、萤火虫优化算法（Firefly Algorithm, FA）和粒子群优化算法（Particle Swarm Optimization, PSO）等。

对于传统方法，其优点是以梯度为基础，求解效率高，理论性强，在早期具有广泛运用。而随着科学的发展，传统解决方法难以用于大规模高时效性的缺点也显露无疑。而随着学者们的不断努力，智能优化算法孕育而生，并较好的解决一些复杂的寻优问题。

1.3 智能优化算法

1.3.1 进化计算

当前经过国内外许多学者对最优化问题的不断深入研究，提出了大量的进化计算（EC）模型，而每个计算模型对具有对问题寻优的共同特性。算法 1.1 伪代码给出了进化算法（EA）一般的寻优过程。

从算法 1.1 伪代码中可以知道，该计算模型包含了达尔文进化理论的两个重要部分，一是物竞天择适者生存的自然选择理论，在算法中则体现为交叉选择操作。二是在一定环境下的基因突变理论，其在算法中体现为变异操作。尽管在生物进化过程中，物种并没有收敛趋势。但由于问题寻优的需要，模型中包含了一些收敛法则来确保算法的寻优的有效性，如限制迭代次数，在算法寻优的解满足问题要求的精度时终止算法。

算法1.1　一般进化算法

```
1  令迭代器t=0；
2  创建并初始化大小为N，维度为D的种群P；
3  repeat
4     计算种群P(t)中所有个体的适应度值f；
5     对种群执行交叉操作生成新的后代；
6     对种群执行变异操作生成变异个体；
7     执行种群个体选择，从而产生下一代群体；
8     迭代器加一；
9  until 满足终止条件；
```

1.3.2 智能优化算法

用于求优化问题求解或搜索过程的步骤的方法称为优化算法。优化算法一般可分为经典优化算法和智能优化算法两大类.由于经典优化算法具有过度依赖优化问题的数学特征，在求解具有较大问题的多种复杂问题时，具有计算复杂度高、求解时间长的缺点。实际优化问题很难得到最优解。为了在求解时间和求解质量之间实现良好的平衡，研究人员提出了一些启发式计算智能算法。智能优化算法以其快速有效的显著特征得到了广泛的关注和应用。群体智能是指一群简单的个体在当前现实环境中表现出来的集体智能。群体智能算法包括混沌蚁群算法，蚁群优化算法，蜂群算法，鱼群算法，萤火虫算法，杜鹃算法，粒子群优化算法，细菌觅食优化算法和烟花算法。群体智能算法是群体中的多个个体相互协作，采用概率传递方法，结合一定的启发式规则，在求解空间上进行并行搜索的方法。群体智能算法的特点是随机性、适应性和并行性。目前，群体智力的数学基础相对匮乏，许多研究人员正在不断探索。在性能改进和应用范围扩展方面，群智能算法得到了不断的改进。

1.4 主要研究内容及文章结构

本文主要阐述萤火虫优化算法的改进及应用，在参考国内外学者已发表的论文的基础上，针对萤火虫优化算法现有的不足进行改进。并将优化后的算法应用于车间调度问题的解决。本文共 6 个章节，相关结构如下：

第 1 章为绪论，主要阐述了萤火虫优化算法的研究背景及意义，最优化问题的分类及求解，以及智能优化算法的相关概念。

第 2 章介绍了萤火虫优化算法的基本概念、公式和标准萤火虫优化算法的算法框架。然后分析了萤火虫优化算法的相关参数，最后总结了萤火虫优化算法的研究现状。

第 3 章则针对萤火虫优化算法现有的不足进行分析改进，分别讲述了该算法中步长和吸引力的不足及改进思想。然后进行了实验分析，且实验数据表明该改进方法是行之有效的。

第 4 章主要通过反向学习的策略改进萤火虫优化算法种群多样性不足问题，章节中，先是介绍了反向学习的基本概念，然后讲解了算法与反向学习策略的融合思路，并结合实验结果数据进行分析。

第 5 章将改进的萤火虫优化算法运用于解决车间调度问题，其中阐述了车间调度问题的基本概念及问题特点和分类，接着描述用户从优化算法解决柔性车间调度的难点，然后提出了离散化编码解决方案，并运用改进的优化算法成功解决了该问题。

第 6 章对本文做出总结，提出当前研究存在的不足，并对未来可能的研究方向进行归纳展望。

第 2 章 萤火虫优化算法概述

2.1 萤火虫优化算法生物学原理

萤火虫优化算法是一种元启发式的全局优化算法，其动机是萤火虫的闪烁行为。它最初是由 Yang 于 2008 年提出的，他解释了萤火虫的闪烁行为如何抽象出一种优化算法。萤火虫是生物发光的，而且以一定的节奏进行闪光。它们要么是为了吸引潜在的伴侣，要么是为了保护自己不受捕食者的侵害。并且，萤火虫根据光的强度相互移动。萤火虫的光强度随着萤火虫之间距离的增加而减小[17]。因此，萤火虫算法遵循以下三个理想化规则：

（1）萤火虫不分性别根据光线的强弱相互吸引。

（2）每个萤火虫的吸引力和亮度是相关的，吸引力较小的萤火虫向具有更多吸引力的萤火虫移动。

（3）萤火虫亮度与目标函数相关。

2.2 标准萤火虫优化算法

2.2.1 算法相关参数分析

在萤火虫算法中，每个萤火虫的位置表示要解决的问题的解决方案。萤火虫的亮度取决于要解决的问题的目标函数值。目标函数越好，萤火虫的亮度越强。随着迭代过程的进行，种群中的弱萤萤离自己的萤火虫越来越近，大多数萤火虫会聚集在最亮的萤火虫附近，最亮的萤火虫是问题的最优解。萤火虫算法有四个非常重要的概念：光强、吸引力、距离和运动。

亮度：

$$I(r)=I_0e^{-\gamma r^2} \tag{2.1}$$

其中，初始亮度用变量 I_0 表示，r 为两个萤火虫之间的笛卡尔距离，亮度的损失因子则用γ表示。

吸引力：

$$\beta(r)=\beta_0e^{-\gamma r^2} \tag{2.2}$$

其中，吸引力可以定义为另一只萤火虫观察到的光强，被定义为β，如式 2.2 中所给出的那样。这里β_0是当距离为零时的吸引力。

距离：

$$r_{ij}=\left\|x_i-x_j\right\|=\sqrt{\sum_{d=1}^{D}\left(x_{id}-x_{jd}\right)^2} \tag{2.3}$$

在式(2.3)中给出了萤火虫之间距离的计算方式，并被分配在笛卡尔坐标。其中 d 是维度的数目。

移动：

$$x_i\left(t+1\right)=x_i\left(t\right)+\beta_0 e^{-\gamma r^2}\left(x_j\left(t\right)-x_i\left(t\right)\right)+\alpha\epsilon_i \tag{2.4}$$

萤火虫相互吸引，萤火虫的运动作为吸引力的结果可以用方程 2.4 来计算。这里萤火虫 i 被 j 吸引，并以一定的随机系数α向其方向移动。

方程 2.4 解决了整个算法中最关键的部分，即萤火虫向最优解的移动。光吸收系数γ可以有一个小的值或一个大的值。如果选择光吸收系数为零，则萤火虫的吸引力与萤火虫之间的距离为零的情况相同。这意味着，具有这种亮度的萤火虫可以从任何位置看到，使它成为全球最好的。如果系数γ太大，由于引入随机步长参数α，萤火虫的运动就会变得随机。通过考虑这两种渐近行为，可以实现萤火虫算法。

而萤火虫算法中，其主要参数有光吸收系数γ、最大吸引力β_0和随机性参数α，且该算法对这三个参数是非常敏感的。如前所述，萤火虫算法是一种元启发式算法，当问题是非线性和非凸时，它给出了最好的结果[18]。

2.2.2 算法基本框架及优点描述

Yang 提出标准萤火虫优化算法如算法 2.1 所示。

算法2.1　标准萤火虫算法框架

```
在解空间中随机实例化N个D维萤火虫个体种群 (Xi | i=1,2 ... N)
设置最大迭代数MAX_GEN、当前迭代数t=0
While t<MAX_GEN
    For i = 1 to N
        For j = 1 to N
            If f(Xi) <f (Xj)
                根据公式2.4移动萤火虫Xi
                更新萤火虫Xi亮度
                t + +
            end if
        end for
    end for
end While
```

在萤火虫优化算法寻优时，种群个体的适应值一般被赋值为目标函数在该个体坐标点的计算值。算法首先对参数及萤火虫个体进行初始化，然后循环迭代评估种群个体的相对亮度，并根据萤火虫个体相对亮度来确定是否移动当前萤火虫个体的位置及更新亮度，如此直至迭代结束。

而在众多的优化算法中，FA 算法的优点有：FA 能够有效地处理高度非线性、非凸的优化问题；与其他经典方法相比，FA 的收敛速度快，不易陷入局部最优，FA 算法容易与其他优化算法相结合[19-21]；一个好的初始解决方案是不需要启动优化过程，即，它导致相同的最优，而不考虑初始解。

2.2 萤火虫优化算法研究现状

萤火虫优化算法同其他智能优化算法相比，萤火虫优化算法参数简单，参数设置较少，易于实现和操作，适合于并行实现，很快吸引了大量研究者的注意，并应用到很多实际工程优化问题中。在学者 Xin-She Yang2008 年最开始提出的萤火虫优化算法中，种群个体凭借着个体之间的吸引力与随机干扰步长进行位置的移动更新。而为了平衡吸引力与步长的影响作用，国内外的学者们在调节算法中的吸引力与步长部分做了大量的改进优化。

Liu[22]等提出了设置步长上下限的线性减小的改进方案，在其改进的萤火虫优化算法中，使得迭代初期的步长较大从而避免群体过早陷入局部最优，而迭代后期则减小步长以提高算法求解精度，并有着良好效果。同样，Goel 和 Panchal[23]也提出了一种类似的优化策略，只是其改进算法中的线性减小策略比 Liu 等提出的步长下降趋势更快。

Wang[24]提出了一种只依赖于最大值的α改进方案，α与迭代次数平方成反比，这意味着 会迅速减小，萤火虫的随机移动在少量迭代后几乎消失。Yu[25]等提出了一种自适应的α，该算法在一定程度上使用了上一代最好解的值自适应的调节步长随机移动参数的大小，有效地避免了算法过早出现停滞现象，然而算法的机制限制了α只会在(0, 1)之间变化，因此，除非增加比例因子，否则当需要大区域搜索时，α的调节功能可能会太小。控制萤火虫移动的另一个重要参数是吸引力步长参数β，在萤火虫优化算法中，吸引力被定义为$\beta_0 e^{-\gamma r^2}(x_j - x_i) = \beta(x_j - x_i)$，$\beta 0$ 为光源处即$r = 0$ 的吸引力。吸引力步长参数β依赖于初始时吸引力$\beta 0$、萤火虫之间的距离r和光强吸收系数γ。近年来对吸引力步长参数β的研究主要集中在对γ和$\beta 0$ 的调节上。Selvarasu 等[26]指定了最大的β_{max}和最小的β_{min}，重定义了基于最大的β_{max}和最小的β_{min} 的β更新机制，这种更新机制使得吸引力步长参数β一直在控制在$[\beta_{min}, \beta_{max}]$之间变化，取得了较好的效果。在 Selvarasu 和 Kalavathi[27]提出的改进方案中，先是把β_{max}设在(0,1)的区间内，然后把β_{min}、α和γ放在解空间中使得解增加三个新的维度，并随着算法的迭代进化一起进化以

达到自适应目的从而在一定程度上避免陷入局部最优，且取得较好的成果。Gandomi 等[28]对萤火虫优化算法引入了混沌理论，通过 12 种不同的混沌图来自适应的调节γ和β，仿真结果表面混沌优化的萤火虫优化算法的效果比标准萤火虫优化算法要好。同时，Jansi 和 AL-Wagih 也进行了相似的研究，Jansi[29]和 AL-Wagih[30]分别采用 Chebyshev 策略和 Sinusoidal 策略来测试步长数值的优劣性。Wang 等[31]提出的改进萤火虫优化算法则通过步长的自适应和变化的β_0来满足算法收敛及对求解结果精度的要求，且其算法改进方案对于大部分测试函数都取得较为满意的结果。

而对于以上提出的改进萤火虫算法其主要原理为在算法求解过程中，随着群体迭代的进行而动态改变萤火虫优化算法的步长和吸引力参数，使得在求解前期步长相对较大，以强化种群对解空间的勘探能力，而后期步长较小以提高在种群的局部解空间的开发能力，最终提高算法求解的效果和精度。但这对种群在演化过程中所形成的“知识”的利用则不过充分，缺乏相应的实验研究。虽然文献[25]在对算法种群在演化过程中所形成的“知识”的利用有一定的研究，但其同时也对步长的变化做了一定的限制，而这也同样限制步长对算法的调节功能。

第 3 章 基于萤火虫优化算法自身参数的改进

3.1 萤火虫优化算法步长改进

3.1.1 萤火虫优化算法步长改进分析

分析萤火虫优化算法的改进思想前，首先来了解下该算法的一些基本特征：

（1）种群性：进化算法由一定数量的个体组成一个种群，并通过种群中的个体间简单的根据适应值互相“吸引”的行为来找最佳适应值，从而体现出群体智能。

（2）迭代性：进化过程以代为基础，使得每代中种群个体探索的新适应度与历史最优值进行比较，确保寻优结果向着更优的结果单向发展。最终找到（全局或局部）最优解，或找到近似（全局或局部）最优解。

（3）策略性：主要指种群中个体向“更优区域”的移动方式（或公式，或策略）。

（4）随机性：主要指群体在迭代过程中，为了靠近最优值，探索出“新路径”或“新点”所采取的随机性机制，（此时的随机一般不具有使种群增加全局多样性的能力）。

（5）多样性：由于移动策略的影响及求解精度的要求，进化中的种群不可避免要失去多样性（这里主要指逐渐失去初始化时全局的多样性）。因此，进化算法有必要增加理论上能保持全局多样性的策略。（为提高种群的全局多样性，可采取反向学习、初始化为全局点等策略。）

这里要说明的是，也正是“ 策略性 + 随机性 ”确保进化算法中的每代的所有探索的新适应度中有大概率出现比历史代最优值更优的情况。从而通过每代探索的更优的新适应度对历史最优适应度进行替换，确保寻优结果向着更优的结果单向发展。

因为在标准萤火虫优化算法中，由于移动公式中吸引力部分随着迭代次数快速趋于 1，如果萤火虫步长不减小，那么该算法中演化后期便有很大概率出现萤火虫个体在最优个体周围振荡现象。如此导致该种群在往后的迭代中新适应值很难比历史最优适应值更优。如图 3.1 所示。

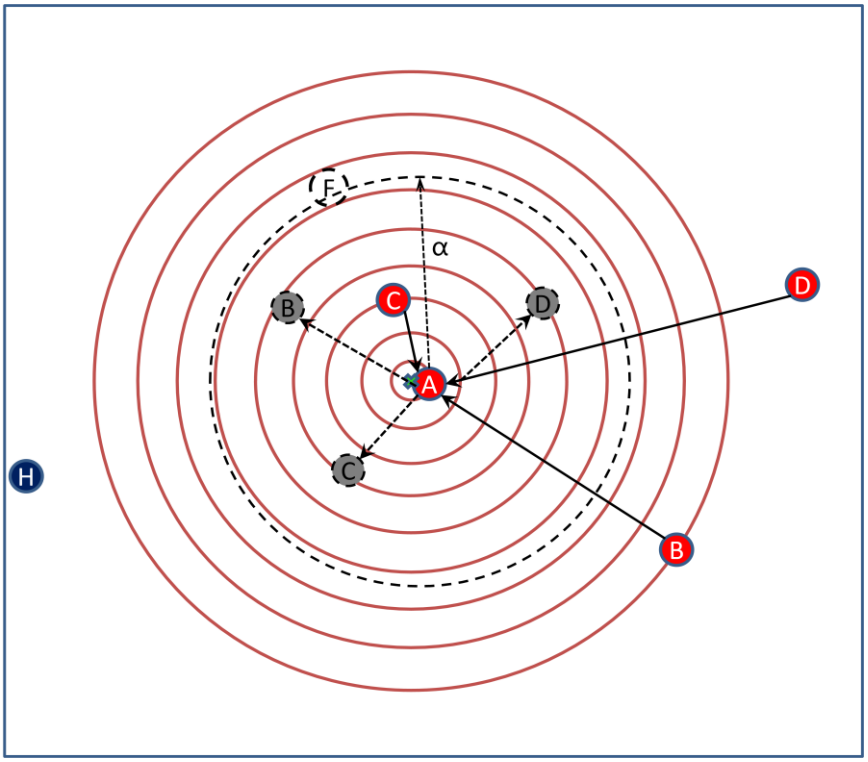


图 3.1　二维 sphere 函数探索振荡模型图

图 3.1 中，红色实线圆点 B、C、D 被更优点 A 吸引，但因为在进化后期步长α 的不变，以致难以探索到实线圆 A 内，或者说需要更多的探索次数才能探索出更优的新适应值，从而在以后迭代中更加难以探索到有效的新适应值点。因此，可以得出结论，为了使历史最优点随着迭代次数能有效的接近全局或局部最优适应值点，所以算法应该具有一种从理论上可以让种群的个体随迭代次数无限接近最优值的可行机制（或策略），如在萤火虫算法中步长随迭代次数的增加而进行一定的减小。

而在减小步长的萤火虫算法版本中，如在 MFA[32]中，步长随迭代次数的增加呈现指数级递减。然而由于不同函数有着不同的形状，以致进化算法求解不同目标函数达到相同精度解所需探索次数不同。而这意味着进化算法在求解不同目标函数时，在同一策略下，对种群接近最优值模型有不同的需求。而这在萤火虫算法求解不同目标函数时表现为对步长的减小有不同的需求。相关理论模型图如图 3.2 所示。

在图 3.2 中，点 A、B、C、D、H 为萤火虫个体，中心绿叉为 Sphere 函数的全局最优点，红色实线圆为该函数等适应点集合。同样由于移动公式中吸引力部分随着迭代次数快速趋于 1，当图中 B 点被 A 点吸引而移动时，由移动公式 1 可知，又因为β 的趋于 1，所以点 B 在此时的移动可以看成，点 B 先执行β 部分移动到达点 A，然后执行α部分移动向点 A 周围进行探索。

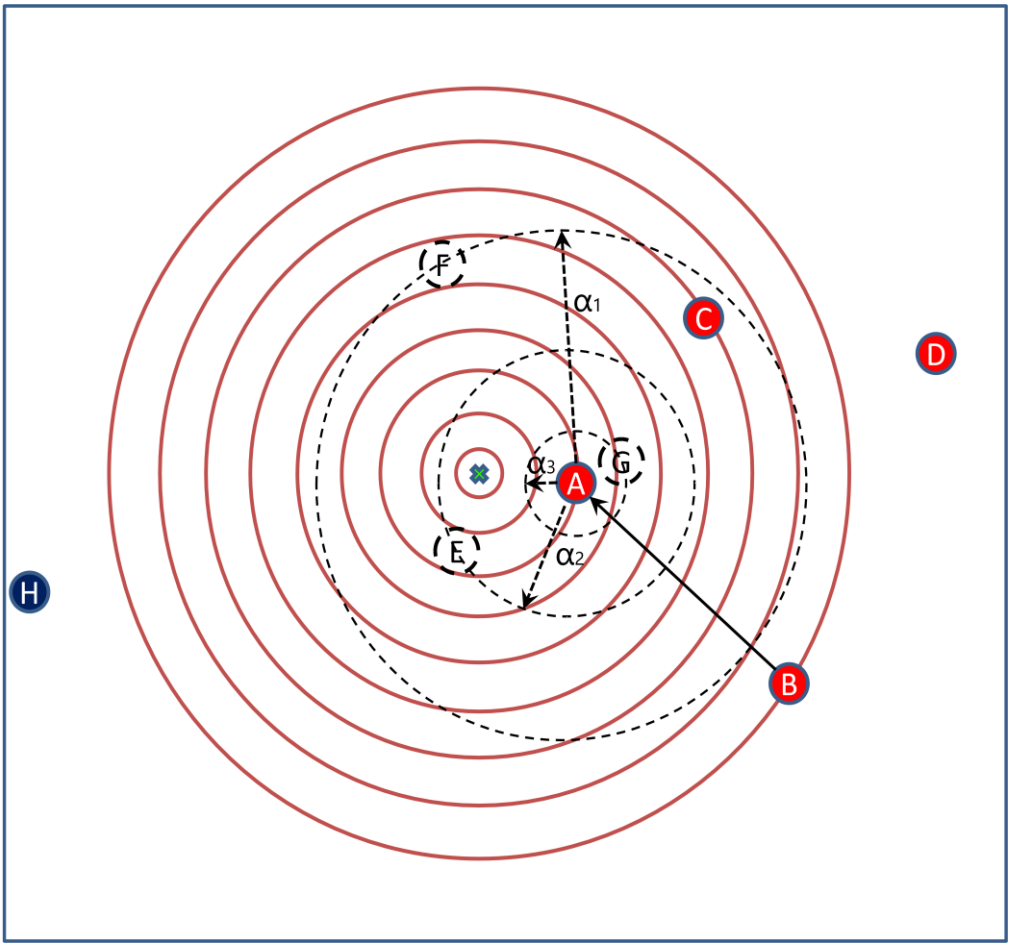


图 3.2　二维 sphere 函数某阶段探索模型图

而由于步长α的大小不同，则点 B 在该次探索得到有效（或更优）的新适应值概率不同。从图 2 中可以看出，在向周围随机探点的情况下；当$\alpha=\alpha_1$时，点 B 探索到有效的新适应值点的概率 $P\alpha_1=SA/SF$（实线圆 A 的面积除以虚线圆 F 的面积）；当$\alpha=\alpha_2$时，B 探索到有效的新适应值点的概率 $P_{\alpha2}=(S_A \cap S_E)/ S_E$ (实线圆 A 的面积与虚线圆 E 的面积的交集除以虚线圆 E 的面积)；当$\alpha=\alpha_3$时，B 探索到有效的新适应值点的概率 $P_{\alpha3}=(S_A \cap S_G)/ S_G$（实线圆实线圆 A 的面积与虚线圆 G 的面积的交集除以虚线圆 G 的面积)。并从图 2 中相关圆的位置及半径的大小关系可知，$P_{\alpha1}< P_{\alpha2}< P_{\alpha3}$ ，且当步长趋于 0 时，概率 $P_{\alpha1}$、 $P_{\alpha2}$、 $P_{\alpha3}$ 将趋于 0.5。

或许从概率角度上，求得有效的新适应点的概率 P 越大越好。然而，在算法探索得到有效（或更优）的新适应值概率上，并不是概率越大越好。因为从图 2 中可以知道，当萤火虫步长α越小时概率 P 越大，并在步长趋于 0 时 P 趋于极限 0.5，但此时也意味着此时探索得到的更优的新适应值向最优值移动的距离是趋于 0 的。因此，可以得出，在萤火虫算法求解目标的函数时，不同阶段有着不同的最适步长，而在演化过程中步长太长导致大量无效探索，步长太小则导致种群难以有效接近全局（或局部）最优值。因此，设计良好的策略使得算法在不同阶

段有着最适的步长对萤火虫算法性能有着至关重要的作用。

因此，改进算法的其中之一的可行方向是，在吸引部分不变的情况下，设计一个适应于不同目标函数的步长变化模型，使得步长适应萤火虫算法的不同的演化迭代阶段。

3.1.2 自适应步长策略设计

在标准萤火虫算法中，当萤火虫 x 移动到另一个更亮的萤火虫时，随机移动可以帮助算法更好地探索更亮萤火虫周围的解。 随机运动的影响取决于参数α。如果α得很大，萤火虫 x 将跳出邻域搜索范围，探索一个新的解空间。 相反，如果设置较小，算法将在邻域周围搜索，应该指出，α并不完全代表随机运动。 从公式 2.4 中，我们可以看到α也在（-0.5,0.5）之间乘一个随机向量。在算法迭代的早期阶段，较大的随机运动可以帮助算法探索更多的解空间，提高算法的全局搜索能力。 在后期，萤火虫可能更接近最优解。 此时，较小的随机运动可以使算法在较好的解周围尽快收敛，因此，许多文献提出，随着迭代次数的增加，α将呈指数下降。

这看起来是个好策略。然而实验表明，α指数的快速递减会造成算法在演化后期步长变得较小，从而不利于群体向全局最优点移动。所以一个好的步长变化策略对于不同的函数来说变化时机应该是不确定的，且简单地让α随着迭代次数以指数级递减的方式对于某些函数也是不适应的。于此，本文提出了一种基于距离控制的自适应步长策略，α被重新定义为下公式：

$$\alpha(t+1)=(\frac{8}{9}+\frac{1}{10(x_d^{\max}-x_d^{\min})^t})\alpha(t) \tag{3.1}$$

公式 3.1 中 r 表示问题的定义域的区间，t 为迭代次数。并且在改进后的算法中引入α减小的控制变量，当某次迭代的最优个体移动距离小于控制变量时，然后步长按公式 3.1 进行变化。而对于控制变量的设置，首先我们计算出每代最优个体的移动距离，然后根据距离与阈值的大小判断α与阈值是否需要改变。如果需要改变，则根据公式 3.1 更新参数α，以及使用如下公式更新阈值：

$$\eta(t+1)=\eta_0(\frac{8}{9}+\frac{1}{10(x_d^{\max}-x_d^{\min})^t})\eta(t) \tag{3.2}$$

公式 3.2 中 t 为迭代次数，且 η_0 被设为定义域距离长度的百分之一。

基于以上思考，本文提出了一种带有控制体的自适应步长的萤火虫优化算法(Self-adaptive Step Strategy Firefly Algorithm, SASFA)，并较好的解决了标准萤火

虫优化算法中步长过早或过慢收敛以及后期在在极值点周围振荡问题。在算法 3.1 中列出了所提出的改进萤火虫优化算法的框架，可以看到，该算法不增加循环次数，因此该算法与标准 FA 算法具有相同的算法复杂度。与标准 FA 算法相比，算法 3.1 对这三个地方进行了修改。 首先，在第 17 行中使用方程 3.1 更新步长α。其次，在第 15 行引入距离控制变量。第三，在第 16 行通过公式 3.2 更新距离控制变量。

算法3.1　改进萤火虫优化算法SASFA框架

```
1   在解空间中随机实例化N个D维萤火虫个体种群 (Xi | i=1,2 ... N)
2   初始化萤火虫个体亮度
3   设置最大迭代数MAX_GEN、当前迭代数t=0
4   While t<MAX_GEN
5     For i = 1 to N
6       For j = 1 to N
7         If f(Xi) <f (Xj)
8               根据公式2.4移动萤火虫Xi
9               更新萤火虫Xi亮度
10              t + +
11        end if
12      end for
13    end for
14    计算最佳个体的移动距离
15    if distance < η(t)
16          根据公式3.2更新控制变量;
17          根据公式3.1更新步长;
18    end if
19  end While
```

3.1.3 实验设置

为了验证我们提出的改进的萤火虫优化算法，在下一个实验中，使用了 13 个 benchmark 函数[33]来测试算法的性能。 在这些函数中，f_1-f_5 是单峰函数，f_6 是一个具有最小值的跳跃函数，并且不连续，f_7 是一个具有噪声的二次函数，以及具有许多局部极小值的 f_8-f_{13} 多模态函数。 所有函数都是最小化问题且全局最优为 0，它们具体数学表达式如表 3.1 所示，其中 D 为问题的维度大小。且在实验中，上述改进萤火虫优化算法将与标准 FA 和其他三个最近提出的 FA 进行性能比较，所涉及的算法集的所有参数的细节如表 3.2 所示。

为了保证算法的公平性，将所有算法的下列参数设置为相同的，并列出如下：

- Population size: 20
- Max iterations: 5.0E+05
- Run times: 30

表3.1 实验使用的benchmark函数

Name	Function	Search Range
Sphere	$f_1(x)=\sum_{i=1}^{D}x^2$	[-100, 100]
Schwefel 2.22	$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}x_i$	[-10, 10]
Schwefel 1.2	$f_3(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	[-100, 100]
Schwefel 2.21	$f_4(x)=\max\{\lvert x_i\rvert, 1\le i\le D\}$	[-100, 100]
Rosenbrock	$f_5(x)=\sum_{i=1}^{D}[100(x_{i+1}-x_i^2)^2+(1-x_i^2)^2]$	[-30, 30]
Step	$f_6(x)=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor$	[-100, 100]
Quartic with noise	$f_7(x)=\sum_{i=1}^{D}ix_i^4+random[0,1)$	[-1.28, 1.28]
Schwefel 2.26	$f_8(x)=\sum_{i=1}^{D}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)+418.9829D$	[-500, 500]
Rastrigin	$f_9(x)=\sum_{i=1}^{D}[x_i^2-10\cos 2\pi x_i+10]$	[-5.12, 5.12]
Ackley	$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$	[-32, 32]
Griewank	$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]
Penalized 1	$f_{12}(x)=\frac{\pi}{D}\left\{\sum_{i=1}^{D-1}(y_i-1)^2[1+\sin(\pi y_{i+1})]+(y_D-1)^2+10\sin^2(\pi y_1)\right\}+\sum_{i=1}^{D}u(x_i,10,100,4), y_i=1+(x_i+1)/4$, $u(x_i,a,k,m)=\begin{cases}u(x_i,a,k,m), & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	[-50, 50]
Penalized 2	$f_{13}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_D-1)^2[1+\sin^2(2\pi x_D)]\right\}+\sum_{i=1}^{D}u(x_i,5,100,4)$	[-50, 50]

表3.2 对比算法参数设置

	α	$\alpha_{\min}$	α_0	γ	β	$\beta_{\min}$
FA(Yang 2010)	0.2	-	-	1.0 or $1/\Gamma^2$	$\beta_0=1.0$	-
MFA(Fister et al. 2012)	-	-	0.5	$1/\Gamma^2$	$\beta_0=1.0$	0.2
WSSFA(Yu et al. 2014)	-	0.04	-	1.0	$\beta_0=1.0$	-
VSSFA(Yu et al. 2015)	-	-	-	1.0	$\beta_0=1.0$	-
SASFA	-	-	0.5	$1/\Gamma^2$	$\beta_0=$ Eq. 3.1	-

3.1.4 结果对比与分析

表 3.3 记录了 SASFA 在 13 个 benchmark 函数上的结果。 如表 3.3 所示，除了f_5，f_8以及f_9为局部最优解之外，其他函数大多找到近似最优解，特别是f_6，每次都找到全局最优解。

Yang（Yang, 2010）建议$\gamma=\frac{1}{\Gamma^m}, m\ge 1$，$\Gamma$是给定问题的给定长度尺度，本文将 m 设为 2。 因此，对 SASFA 和 FA 的两个版本进行了比较。 比较结果列于表 3.4。 “mean”表示 30 次运行的平均最优值，“stddev”是标准差。 从表 3.4 中我们可以看到，当$\gamma=1.0$时，在大多数基准函数中，标准 FA 很难得到全局最

优解，但当γ变为$1/\Gamma^2$时，标准 FA 的性能显著提高。f_1, f_4, f_7和f_{11}得到了近似最优解，其余函数的质量也得到了提高，对于 benchmark 函数 SASFA 比两个版本要好得多，只有f_5和f_9除外。

表3.3　算法SASFA计算结果

Function	Worst	Best	Mean	Std Dev
f_1	3.70E-44	9.17E-46	8.64E-45	1.59E-44
f_2	1.10E-23	2.01E-24	7.11E-24	3.49E-24
f_3	1.13E-29	5.06E-32	4.61E-30	4.78E-30
f_4	4.73E-24	1.89E-24	3.02E-24	1.18E-24
f_5	1.59E+01	7.81E+01	3.05E+01	2.70E+01
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	2.86E-02	7.08E-02	4.42E-02	1.58E-02
f_8	-4.63E+03	-6.71E+03	-5.58E+03	8.92E+02
f_9	7.06E+01	3.48E+01	5.27E+01	1.65E+01
f_{10}	3.95E-14	2.18E-14	2.89E-14	7.54E-15
f_{11}	3.19E-02	0.00E+00	1.03E-02	1.32E-02
f_{12}	2.83E-32	1.47E-32	2.06E-32	5.26E-33
f_{13}	2.07E-01	1.57E-32	1.24E-01	8.67E-02

表3.4　标准FA和SASFA实验计算结果

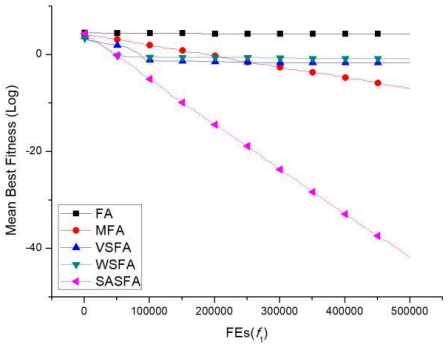
Function	FA($\gamma = 1.0$)		FA($\gamma = 1/\Gamma^2$)		SASFA	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
f_1	6.67E+04	1.83E+04	5.14E-02	1.36E-02	8.64E-45	1.59E-44
f_2	5.19E+02	1.42E+02	1.07E+00	2.65E-01	7.11E-24	3.49E-24
f_3	2.43E+05	4.85E+04	1.26E-01	1.86E-01	4.61E-30	4.78E-30
f_4	8.35E+01	3.16E+01	9.98E−02	2.34E−02	3.02E-24	1.18E-24
f_5	2.69E+08	6.21E+07	3.41E+01	6.23E+00	3.05E+01	2.70E+01
f_6	7.69E+04	3.38E+03	5.24E+03	1.08E+03	0.00E+00	0.00E+00
f_7	5.16E+01	2.46E+01	7.55E−02	1.42E−02	4.42E-02	1.58E-02
f_8	1.10E+04	3.77E+03	9.16E+03	1.78E+03	-5.58E+03	8.92E+02
f_9	3.33E+02	6.28E+01	4.95E+01	2.39E+01	5.27E+01	1.65E+01
f_{10}	2.03E+01	2.23E−01	1.21E+01	1.96E+00	2.89E-14	7.54E-15
f_{11}	6.54E+02	1.69E+02	2.13E−02	1.47E−02	1.03E-02	1.32E-02
f_{12}	7.16E+08	1.82E+08	6.24E+00	4.62E+00	2.06E-32	5.26E-33
f_{13}	1.31E+09	4.76E+08	5.11E+01	1.28E+01	1.24E-01	8.67E-02

表 3.5 列出了 VSSFA、WSSFA、MFA 和 SASFA 之间的比较结果。 如 3.1.3 实验设置所示，算法运行的所有参数都设置为相同的参数。如表 3.5 所示，VSSFA 和 WSSFA 似乎很难找到全局最优解。MFA 和 SASFA 优于 VSSFA 和 WSSFA，特别是 F6，MFA 和 SASFA 都实现了全局最优解。 MFA 实现了两个最优解(f_6，f_{11})，而 SASFA 实现了十个最优解(f_1~f_4、f_5、f_6、f_9~f_{10}、f_{12})，大多数情况下 SASFA 的性能优于 MFA。

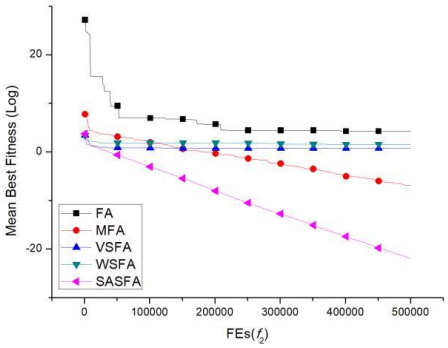
表3.5 VSSFA, WSSFA, MFA 和 SASFA 的平均最优实验结果

Function	VSSFA Mean	WSSFA Mean	MFA Mean	SASFA Mean
f_1	5.84E+04	6.34E+04	1.56E-05	8.64E-45
f_2	1.13E+02	1.35E+02	1.85E-03	7.11E-24
f_3	1.16E+05	1.10E+05	5.89E-05	4.61E-30
f_4	8.18E+01	7.59E+01	1.73E-03	3.02E-24
f_5	2.16E+08	2.29E+08	2.29E+01	3.05E+01
f_6	5.48E+04	6.18E+04	0.00E+00	0.00E+00
f_7	4.43E+01	3.24E-01	1.30E-01	4.42E-02
f_8	1.07E+04	1.06E+04	4.84E+03	-5.58E+03
f_9	3.12E+02	3.61E+02	6.47E+01	5.27E+01
f_{10}	2.03E+01	2.05E+01	4.23E-04	2.89E-14
f_{11}	5.47E+02	6.09E+02	9.86E-03	1.03E-02
f_{12}	3.99E+08	6.18E+08	5.04E-08	2.06E-32
f_{13}	8.12E+08	9.13E+08	6.06E-07	1.24E-01
w/t/l	13/0/0	13/0/0	9/1/3	-

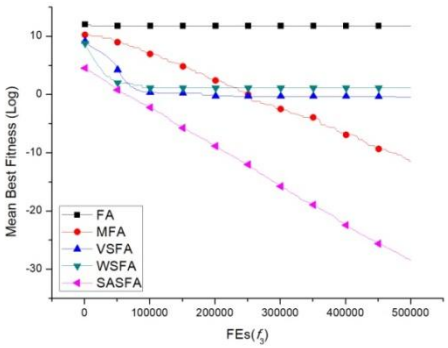
图 3.3 给出了 VSSFA、WSSFA、MFA 和 SASFA 之间的收敛曲线。 为了提高数值差异的灵敏度，我们对某些函数(f_1-f_4，f_7，f_9-f_{10}，f_{12}，f_{13})的适应度值进行了对数转换。 如图 3.3 所示，SASFA 比其他四种算法都要好得多，无论是在收敛速度上，还是在大多数函数上，除了 f_3，f_8 和 f_{13} 之外，SASFA 的质量都很好。



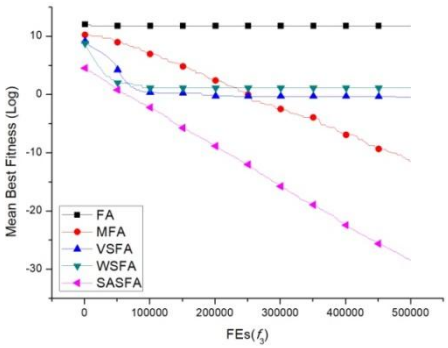
(a) Sphere (f_1)



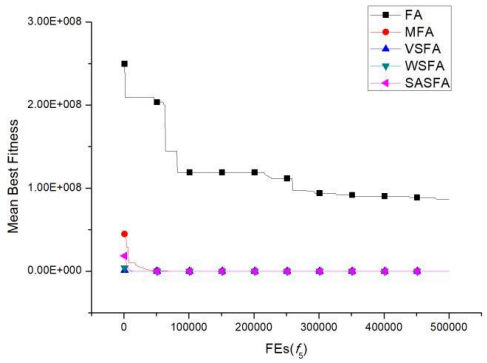
(b) Schwefel 2.22(f_2)



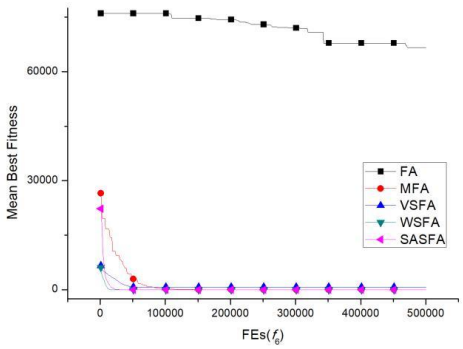
(c) Schwefel 1.2(f_3)



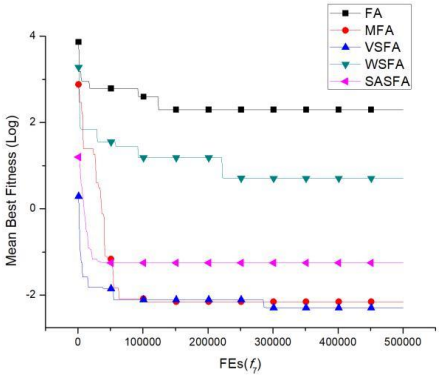
(d) Schwefel 2.21 (f_4)



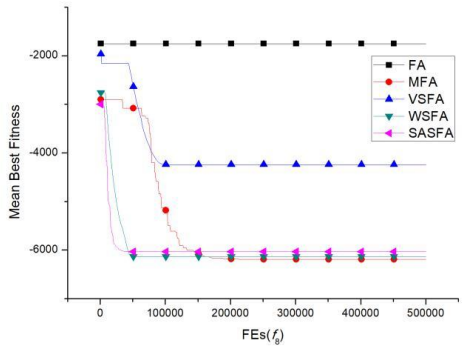
(e) Rosenbrock(f_5)



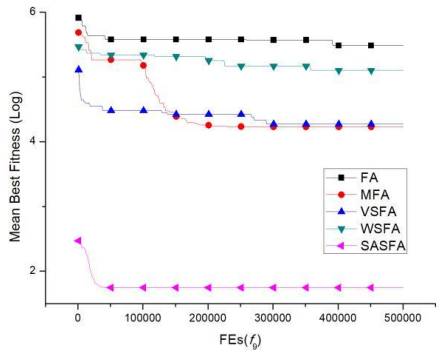
(f) Step(f_6)



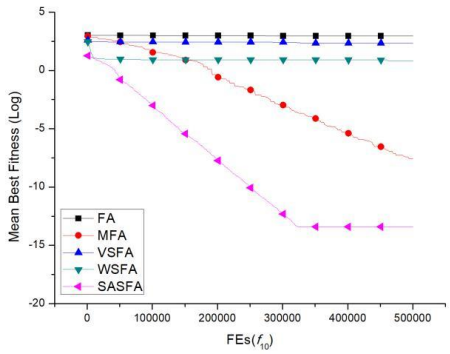
(g) Quartic with noise(f_7)



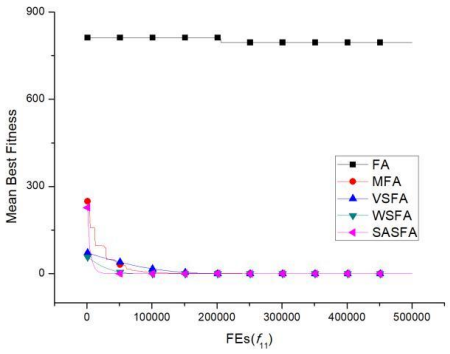
(h) Schwefel 2.26(f_8)



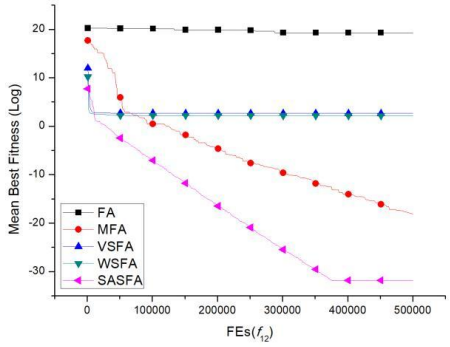
(i) Rastrigin(f_9)



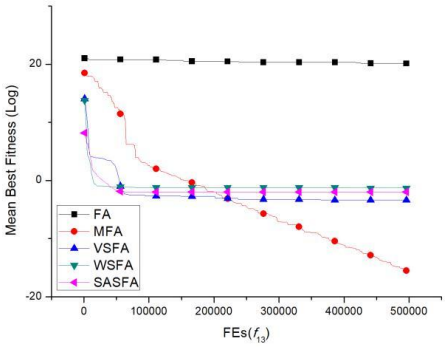
(j) Ackley(f_{10})



(k) Griewank(f_{11})



(l) Penalized 1(f_{12})



(m)Penalized 2(f_{13})

图3.3　VSSFA, WSSFA, MFA和SASFA算法收敛曲线图

3.2 萤火虫优化算法吸引力改进

3.2.1 萤火虫优化算法吸引力改进分析

根据算法的移动公式（2.4）可知，其中吸引力部分，即$\beta_0 e^{-\gamma r^2}$部分，在种群寻优过程中使得萤火虫个体以移动比例向着更优的个体移动。而图 3.4 刻画了萤火虫优化算法种群在寻优过程中，吸引力部分的变化趋势，其中横坐标为萤火虫个体的评估迭代次数（实验设置总迭代次数为 500000），纵坐标为吸引力部分的具体值。分析图 3.4 可知，当β_0=1 时，萤火虫优化算法中吸引力部分将随着迭代次数的增加而以较快的速度趋于 1。这也意味着当种群迭代到 50000 次数以后，种群中的个体将直接在当前迭代的最优个体周围进行探索，较大的减小了种群的多样性，易使群众陷入对目标的未包含状态，如图所示。图 3.5 中同心圆表示 Sphere 函数的等值线，三角形表示种群个体，中心的叉为优化算法所要寻找的目标解。可以看到在图 3.5 的 a 部分中，原种群包含目标点，多样性较高。在演化若干代后，由于种群中更优个体的高吸引性，使得种群演化成未包含状态，如图 3.5 的 b 部分所示，种群失去多样性。而在算法没有有效的多样性增加策略时，这将使种群难以从未包含状态变回包含状态，算法最终难以寻得全局最优解。因此，在改进 FA 算法的吸引力部分设计中应尽量避免算法在演化过程中出现种群未包含目标解的状态。

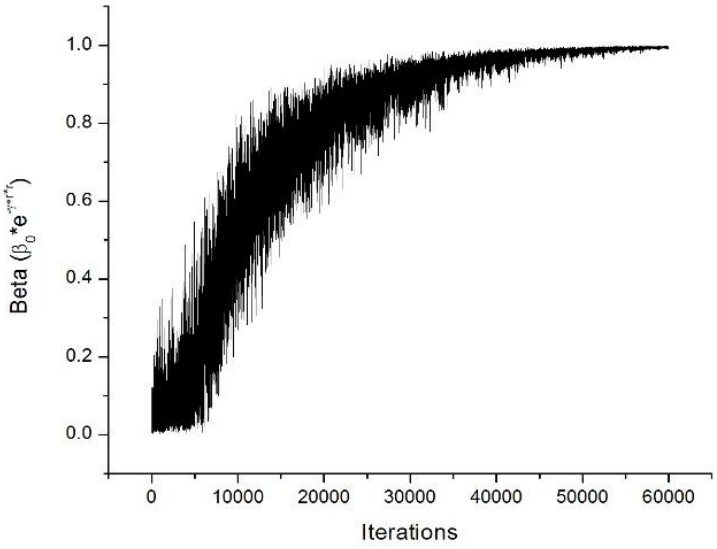
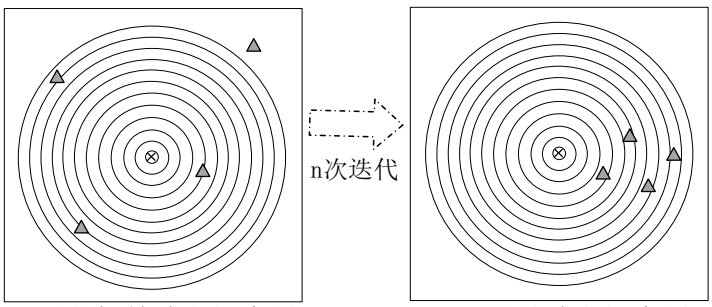


图 3.4 β_0=1 时萤火虫优化算法吸引力部分寻优变化趋势

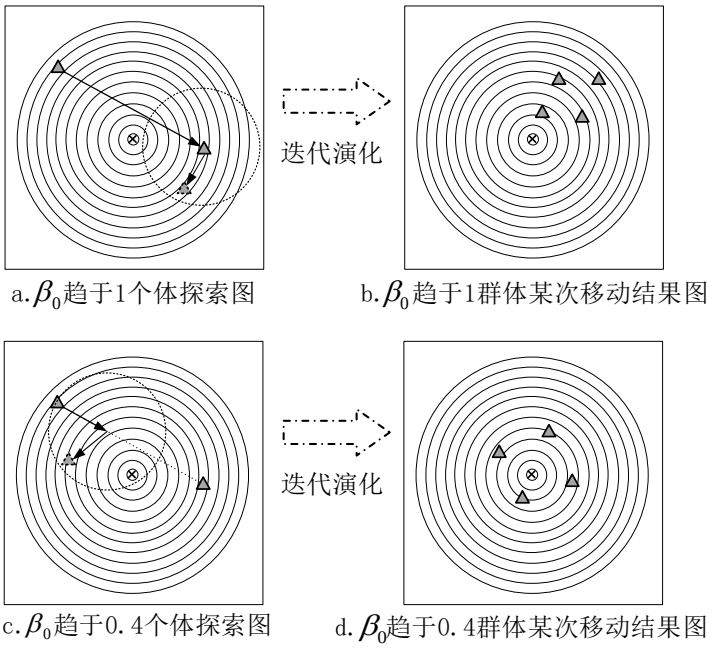


a. 群体初始阶段探索图　　b. 群体某阶段探索图

图 3.5 萤火虫群体在 Sphere 函数中探索状态演变图

3.2.2 基于比例模型的吸引力改进设计

基于上小节对 FA 吸引力部分的优化分析，可将吸引力部分的初始吸引力系数改为 0.4，即β_0=0.4，如此，萤火虫个体在吸引移动时，最终将不再趋向于 1 而是 0.4，从而使得个体在某代吸引移动时将以一定的比例移向更优个体，其在 FA 演化过程中的效果如图 3.6 所示。在图 3.6 中，实线同心圆表示空间中 Sphere 函数的等值线，三角形为寻优空间中萤火虫个体，中心叉表示该解空间中的所有求得的目标函数的最优解。而图 3.6 的 a、c 部分中，实线箭头为种群个体在某次移动时，更劣个体向更优个体的移动路径，虚线圆则为萤火虫个体因步长而引起的移动的可能的探索区域。从图 3.6 的 b、d 部分对比可知，在改变 FA 的初始吸引力后，种群将更有可能处于包含状态，更有利于保持种群多样性，提高 FA 的全局寻优能力。



a.β_0趋于1个体探索图　　b.β_0趋于1群体某次移动结果图

c.β_0趋于0.4个体探索图　　d.β_0趋于0.4群体某次移动结果图

图 3.6　算法不同初始吸引力在 Sphere 函数中探索移动对比图

至此，综合 3.1.2 小节对 FA 的步长的改进，将改进后的 FA 的计算模型刻画成图 3.7，其中实线箭头表示种群个体因吸引力而移动的部分，其长度表示为 *d*，*D* 则为某次移动中两个种群个体的笛卡尔距离。在图 3.7 中，虚线圆的含义则是种群个体因步长和随机变量而移动的可能的到达区域，其半径为 *B(t)*，即当前的步长大小。

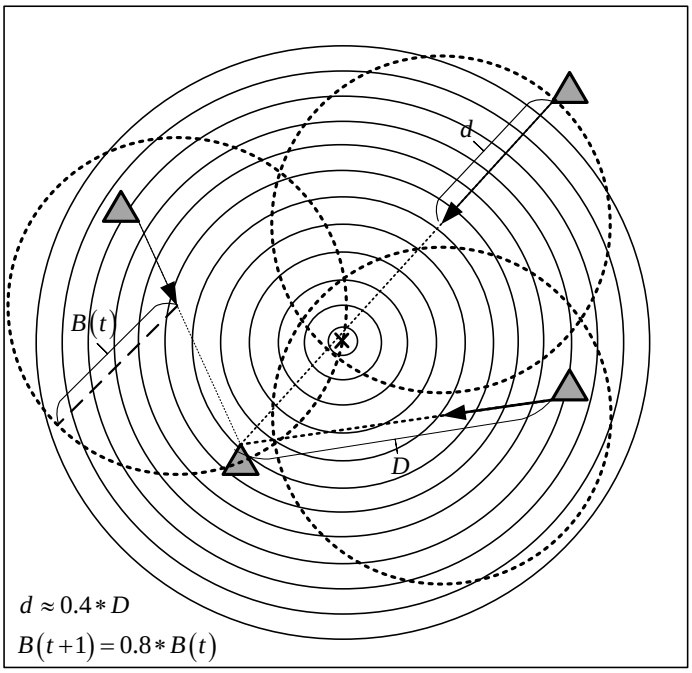


图 3.7　PFA 算法在 Sphere 函数中演化计算模型示意图

相比于标准萤火虫优化算法，改进后的 FA 有着对种群多样性更高的保留能

力，优化了 FA 对全局的寻优性能。如此，将提出的改进萤火虫优化算法命名为比例适应的萤火虫优化算法（Proportional Firefly Algorithm，PFA）。其中，算法 3.2 描述了 PFA 基本框架，相比于标准萤火虫算法框架，PFA 在第 8 行中公式 2.4 的β_0=0.4 进行了改变，以及第 14 至 18 行增加了对步长变化的控制体。因此 PFA 和 FA 的时间复杂度是相同的。

算法3.2　改进萤火虫优化算法PFA框架

```
1   在解空间中随机实例化N个D维萤火虫个体种群 (X_i | i=1,2 ... N)
2   初始化萤火虫个体亮度
3   设置最大迭代数MAX_GEN、当前迭代数t=0
4   While t<MAX_GEN
5     For i = 1 to N
6       For j = 1 to N
7         If f(X_i) <f (X_j)
8             根据公式2.4移动萤火虫X_i
9             更新萤火虫X_i亮度
10            t + +
11        end if
12      end for
13    end for
14    计算最佳个体的移动距离
15    if distance < η(t)
16        根据公式3.2更新控制变量;
17        根据公式3.1更新步长;
18    end if
19  end While
```

3.2.3 结果分析

在实验环境设置同 3.1.1 小节所述的情况下，表 3.6 展示了改进后的 PFA 的独立运行 30 次时的相关实验数据。其中，Worst 数据列表示 30 次实验中最差的计算结果，反之，Best 数据列为最好的计算结果。而 Mean 为 30 次寻优结果的平均最优值，Std Dev 为其标准偏差。从表 3.6 中可以看出，PFA 除对 f_5、f_8、f_9 三个函数的寻优结果较差外，其它都有着较好的优化结果。尤其在优化 f6 函数时，PFA 都计算出了其全局最优解。

在表 3.7 中展示了与 Yang 提出的两个不同版本的萤火虫优化算法的比较数据，第一个 FA 为 Yang 在 2008 年文章[8]中提出的初始 FA，第二个 FA 为 Yang 在 2010 年文章[7]提出的改进 FA 并将γ 设置为 $1/\Gamma^2$（Γ 为目标函数的定义域）。而其两个版本 FA 相比中，PFA 大部分都有着较好的结果精度，如函数f_1~f_4、f_6、f_{10}~f_{13}。

表3.6 PFA算法实验结果数据

Function	Worst	Best	Mean	Std Dev
f_1	2.50E-70	2.75E-74	2.36E-71	4.97E-71
f_2	1.58E-36	6.89E-39	1.79E-37	3.24E-37
f_3	3.54E-15	2.73E-18	2.47E-16	6.55E-16
f_4	9.03E-39	1.65E-41	2.53E-39	3.13E-39
f_5	9.37E+01	2.11E+01	2.66E+01	1.29E+01
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	8.42E-02	1.02E-02	2.83E-02	1.46E-02
f_8	6.11E+03	2.67E+03	3.78E+03	1.08E+03
f_9	3.58E+01	1.29E+01	2.23E+01	6.20E+00
f_{10}	4.31E-14	1.47E-14	2.50E-14	6.28E-15
f_{11}	1.48E-02	0.00E+00	3.12E-03	5.01E-03
f_{12}	1.04E-01	1.57E-32	3.46E-03	1.89E-02
f_{13}	2.22E-32	1.57E-32	1.64E-32	1.58E-33

表3.7 FA和PFA算法实验结果数据对比

Function	FA(γ =1.0)		FA(γ =1/Γ^2)		PFA	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
f_1	6.67E+04	1.83E+04	5.14E-02	1.36E-02	2.36E-71	4.97E-71
f_2	5.19E+02	1.42E+02	1.07E+00	2.65E-01	1.79E-37	3.24E-37
f_3	2.43E+05	4.85E+04	1.26E-01	1.86E-01	2.47E-16	6.55E-16
f_4	8.35E+01	3.16E+01	9.98E−02	2.34E−02	2.53E-39	3.13E-39
f_5	2.69E+08	6.21E+07	3.41E+01	6.23E+00	2.66E+01	1.29E+01
f_6	7.69E+04	3.38E+03	5.24E+03	1.08E+03	0.00E+00	0.00E+00
f_7	5.16E+01	2.46E+01	7.55E−02	1.42E−02	2.83E-02	1.46E-02
f_8	1.10E+04	3.77E+03	9.16E+03	1.78E+03	3.78E+03	1.08E+03
f_9	3.33E+02	6.28E+01	4.95E+01	2.39E+01	2.23E+01	6.20E+00
f_{10}	2.03E+01	2.23E−01	1.21E+01	1.96E+00	2.50E-14	6.28E-15
f_{11}	6.54E+02	1.69E+02	2.13E−02	1.47E−02	3.12E-03	5.01E-03
f_{12}	7.16E+08	1.82E+08	6.24E+00	4.62E+00	3.46E-03	1.89E-02
f_{13}	1.31E+09	4.76E+08	5.11E+01	1.28E+01	1.64E-32	1.58E-33

在如表 3.8 所示的 PFA 与不同版本的萤火虫优化算法中，可以知道除 f_5、f_{12} 外，PFA 在其他的目标函数的寻优中都有一定的优势，尤其是在 f_1~f_4 目标函数的寻优结果上的优势更是显著。因此，综合表明 PFA 在对问题的解的精度、种群的收敛速率方面都更具优势，使得萤火虫优化算法的性能优劣进一步的提高。

表3.8 VSSFA、WSSFA、MFA和PFA算法寻优结果平均值对比

Function	VSSFA Mean	WSSFA Mean	MFA Mean	PFA Mean
f_1	5.84E+04	6.34E+04	1.56E-05	2.36E-71
f_2	1.13E+02	1.35E+02	1.85E-03	1.79E-37
f_3	1.16E+05	1.10E+05	5.89E-05	2.47E-16
f_4	8.18E+01	7.59E+01	1.73E-03	2.53E-39
f_5	2.16E+08	2.29E+08	2.29E+01	2.66E+01
f_6	5.48E+04	6.18E+04	0.00E+00	0.00E+00
f_7	4.43E+01	3.24E-01	1.30E-01	2.83E-02
f_8	1.07E+04	1.06E+04	4.84E+03	3.78E+03
f_9	3.12E+02	3.61E+02	6.47E+01	2.23E+01
f_{10}	2.03E+01	2.05E+01	4.23E-04	2.50E-14
f_{11}	5.47E+02	6.09E+02	9.86E-03	3.12E-03
f_{12}	3.99E+08	6.18E+08	5.04E-08	3.46E-03
f_{13}	8.12E+08	9.13E+08	6.06E-07	1.64E-32
w/t/l	13/0/0	13/0/0	9/1/3	-

第 4 章 基于学习策略改进的萤火虫优化算法

4.1 反向学习策略

反向学习（Opposition-based Learning，OBL）的概念在我们周围的世界中广泛存在，但它有时以不同的方式被理解。 例如，物理学中的相反粒子、语言中的反义词、概率事件的补充、模拟中的对立变量、文化中的相反词、集合理论中的补集、哲学中的主体和客体、万物有灵论中的善与恶、政治中的反对党以及宗教和哲学中的二元论[34,35]。

基于反向学习是近十年来备受关注的一个新的研究领域。许多软计算算法已经通过使用 OBL 的概念得到了增强，例如，强化学习（RL）、人工神经网络（ANN）、模糊系统和各种优化方法，例如遗传算法（GA）、差分进化（DE）、粒子群优化（PSO）、蚁群算法（ACS）、群搜索算法（GSA）、人工蜂群（ABC）、模拟退火（SA）等，提出了 OBL 的基本概念，它同时考虑了当前估计（猜测）及其对应的对立面，以有效地找到解。当一个算法的主要目标是寻找目标函数的最优解时，同时考虑一个估计及其对立面有利于提高算法的性能。

自 2005 年 1 月以来，已经出版了 400 多份关于 OBL 概念的出版物。这些研究成果已发表在机器学习或软计算领域的会议、期刊和书籍中。在这些论文中，60%是期刊论文，38%是会议论文，2%是书籍/论文。文献[36,37]对 OBL 进行了两次调查，共调查了 52 篇和 138 篇论文。一些研究工作集中在数学证明和理论定义上，以研究和使用 OBL 的好处；一些研究工作集中在机器学习方法中使用 OBL 的各种方案的特殊发展上；还有一些是关于 OBL 在电力系统、模式识别和图像处理、识别问题、生物信息学、医学等各种科学和工程应用中的不同应用。

在中国古代哲学中，最初的对立概念首先表现在阴阳符号中。这象征着二元概念，其中黑色和白色分别是阴和阳。此外，希腊古典自然元素模式描述了对立的概念，如火（热和干）与水（冷和湿）、土（冷和干）与空气（热和湿）。冷、热、湿和干表示自然实体及其相对实体[205]。似乎用对立概念来描述现实世界中许多实体或情境的概念。因此，计算对立概念[297]是从现实世界中的对立概念中得到启发的，相反数在[38]中的定义如：

$$x' = a + b - x \tag{4.1}$$

其中，$x \in [a, b]$，x'称为 x 的相反数。

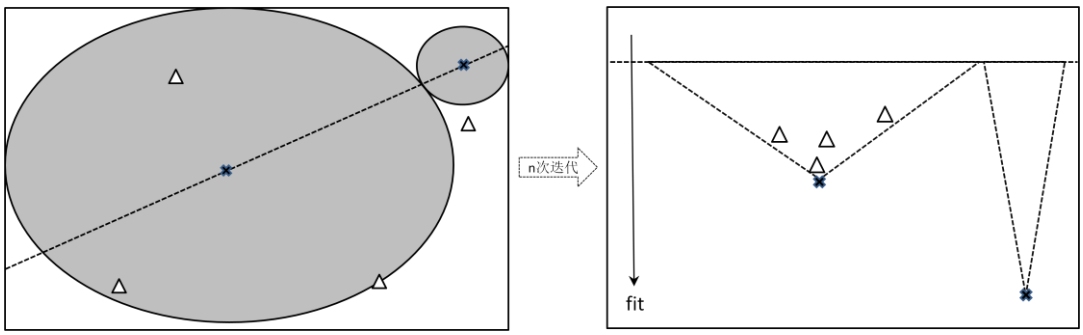
事实上，研究表明，在寻找未知的最优解时，同时搜索随机方向和其相反方

向可以提供更高的机会来寻找有希望的区域。如果当前的估计（猜测）离未知最优解很远，计算它们的对立面会导致向未知最优解的距离更近。

4.2 基于反向学习策略的算法改进设计

4.2.1 算法改进分析

萤火虫优化算法作为启发式进化算法，同样拥有容易陷入局部最优的缺点。尤其是在算法演化的后期，由于对结果精度的要求，算法需要强化对某局部解空间的探索，如此难以避免牺牲种群的多样性。如图 4.1 展示了种群在某二维空间探索时，陷入局部最优的过程，其中实线三角形表示种群个体，又表示某局部空间的适应值的最值，且大面积椭圆区域的最值要劣于小面积椭圆区域的最值。从图 4.1 可以看到，此时种群在经过 n 次迭代后，由图 a 个体相对分散状态演化到了图 b 个体相对集中于某一局部最值状态，使得种群多样性减小，陷入局部最优。



a. 种群某探索阶段图　　b. 种群在某探索阶段的空间切线的适应值

图4.1　种群在某二维空间中探索示意图

因此，为了增加其种群的多样性，有必要为萤火虫优化算法增加一种使种群个体能跳出局部最优的机制。而本章则试图将反向学习策略引入 FA 中以期解决算法过早收敛问题。如图 4.2 所示，其中大面积椭圆区域中心的最值要大于小面积椭圆区域中心的最值。当问题为求解全局最小值时，则要求种群能在小面积椭圆区域探索以求得全局最优值。而如图 4.2 中的虚线三角形（种群中的某个因反向学习策略而进入具有全局最优解的区域的个体）所示，通过反向学习使得种群中的个体有几率在小面积椭圆区域进行探索，极大的增加种群的多样性，跳出局部最优，提高算法求解问题全局最优的能力。

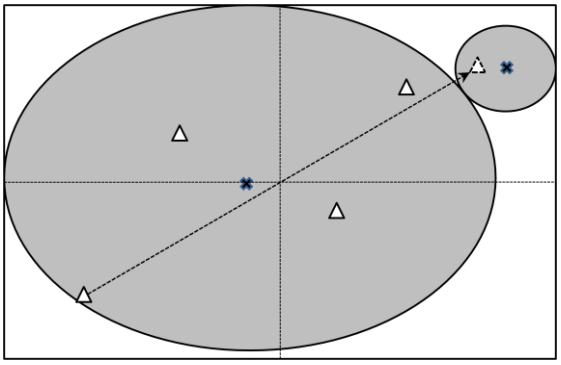


图 4.2　种群在某二维空间中反向学习探索示意图

4.2.2 算法与反向学习策略混合的设计

如前所述，陷入局部最优是进化算法的一种常见现象。 因此，为了提高萤火虫优化算法对求解空间的探索能力，我们在改进的萤火虫算法中将每次迭代的最差的三个个体进行反向学习，使其对解空间中心进行反转。

算法 4.1　POFA 基本框架

在解空间中随机实例化 N 个 D 维萤火虫个体种群 $(X_i \mid i=1,2 \ldots N)$
初始化萤火虫个体亮度
设置最大迭代数 MAX_GEN，当前迭代数 t=0
While$t<MAX_GEN$
  For i = 1 to N
    For j = 1 to N
      If $f(X_i) < f(X_j)$
        根据公式 2.4 移动萤火虫 X_i
        更新萤火虫 X_i 亮度
        t + +
      end if
    end for
  end for
  计算最佳个体的移动距离
  if distance < η(t)
    根据公式 3.2 更新控制变量;
    根据公式 3.1 更新步长;
  end if
    根据公式 4.1 反转当前代中适应值最差的三个个体;
end while

而空间中反向点的定义如公式 4.1 所示，其中，x_i和x_i'为种群演化中互为反向的两点的第 i 维的坐标值。 $x_i^{\max}$ 和x_i^{min} 是 i 维中两点坐标值的上限和下限。

$$x_i^{'} = \left(x_i^{max} + x_i^{min} \right) - x_i ; \quad x_i^{'}, x_i \in [x_i^{min}, x_i^{max}] \tag{4.1}$$

基于以上分析，本文提出了基于反向学习改进的萤火虫优化算法（Proportional and Opposition-based Learning Firefly Algorithm, POFA），POFA 框架如算法 4.1 中所示。从 POFA 框架可以看出，与 FA 算法框架相比，POFA 不增加计算周期的数量，因此 POFA 的复杂性和 FA 是相同的。

4.3 实验结果分析

在实验环境设置如 3.1.3 小节所示的情况下，在表 4.1 中，记录了 POFA 求解 13 个基准函数运行 30 次的实验相关数据，其中“Mean”是平均结果，“STD Dec”是标准差。 从表中可以看出，除在优化f_5、f_8函数算法陷入局部最优外，POFA 对其他目标函数的优化都有优异的表现，且从算法每次的求解结果的标准差可知，POFA 优化性能相对稳定。特别是对于f_6、f_9、f_{11}目标函数，在 30 次测试中的每一个目标函数都求得全局最优解。

表4.1　POFA算法实验结果数据

Function	Worst	Best	Mean	Std Dev
f_1	1.87E-74	5.65E-82	8.51E-76	3.45E-75
f_2	1.89E-39	1.61E-41	3.97E-40	5.24E-40
f_3	5.12E-91	3.51E-97	4.50E-92	1.67E-91
f_4	5.34E-41	1.22E-43	1.54E-41	2.16E-41
f_5	1.94E+01	1.77E+01	1.87E+01	4.00E-01
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	2.98E-05	2.10E-08	3.70E-06	5.60E-06
f_8	-3.78E+03	-2.46E+03	-3.11E+03	3.24E+02
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{10}	3.24E-14	7.55E-15	2.05E-14	7.49E-15
f_{11}	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{12}	2.60E-32	1.57E-32	1.57E-32	2.08E-33
f_{13}	1.10E-02	2.95E-32	3.66E-04	2.01E-03

同样，在与学者 Yang 提出的两个不同版本的 FA 的对比中，POFA 的求解结果也有着突出的优势，如表 4.2 所示。从表中的数据对比中，除f_5、f_8、f_{13}结果数据相差较小外，其它寻优结果精度都有质的提高。

为了更全面地比较 POFA 的优化性能，我们与近年来其他改进的萤火虫算法 VSSFA、WSSFA、MFA 进行了比较。具体数据的描述于表 4.3。 如表所示，POFA 对f_1~f_4、f_6、f_7和f_9~f_{12}的性能优于其他改进的萤火虫优化算法，尤其是f_6、f_9、

f_{11}三个目标函数都找到了全局最优解。而这也主要归因于 POFA 在融合反向学习后，算法对种群的多样性有更高的保留能力，提高全局最优解的搜索能力，同时结合自适应步长策略，使得算法在求解的精度上也有很大的提高。

表4.2 FA和POFA算法实验结果数据对比

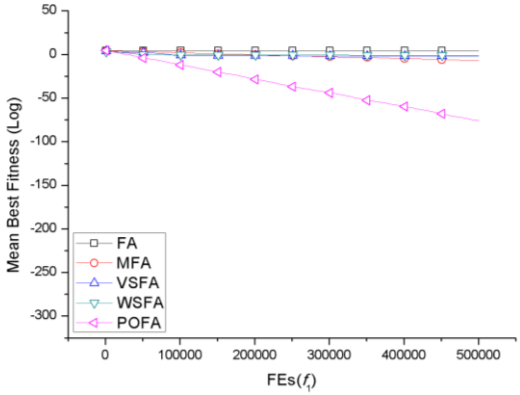
Function	FA($\gamma = 1.0$)		FA($\gamma = 1/\Gamma^2$)		POFA	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
f_1	6.67E+04	1.83E+04	5.14E-02	1.36E-02	8.51E-76	3.45E-75
f_2	5.19E+02	1.42E+02	1.07E+00	2.65E-01	3.97E-40	5.24E-40
f_3	2.43E+05	4.85E+04	1.26E-01	1.86E-01	4.50E-92	1.67E-91
f_4	8.35E+01	3.16E+01	9.98E−02	2.34E−02	1.54E-41	2.16E-41
f_5	2.69E+08	6.21E+07	3.41E+01	6.23E+00	1.87E+01	4.00E-01
f_6	7.69E+04	3.38E+03	5.24E+03	1.08E+03	0.00E+00	0.00E+00
f_7	5.16E+01	2.46E+01	7.55E−02	1.42E−02	3.70E-06	5.60E-06
f_8	1.10E+04	3.77E+03	9.16E+03	1.78E+03	-3.11E+03	3.24E+02
f_9	3.33E+02	6.28E+01	4.95E+01	2.39E+01	0.00E+00	0.00E+00
f_{10}	2.03E+01	2.23E−01	1.21E+01	1.96E+00	2.05E-14	7.49E-15
f_{11}	6.54E+02	1.69E+02	2.13E−02	1.47E−02	0.00E+00	0.00E+00
f_{12}	7.16E+08	1.82E+08	6.24E+00	4.62E+00	1.57E-32	2.08E-33
f_{13}	1.31E+09	4.76E+08	5.11E+01	1.28E+01	3.66E-04	2.01E-03

表4.3 VSSFA、WSSFA、MFA和POFA算法寻优结果平均值对比

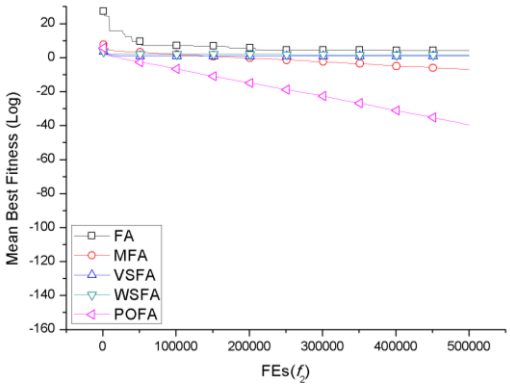
Function	VSSFA Mean	WSSFA Mean	MFA Mean	POFA Mean
f_1	5.84E+04	6.34E+04	1.56E-05	8.51E-76
f_2	1.13E+02	1.35E+02	1.85E-03	3.97E-40
f_3	1.16E+05	1.10E+05	5.89E-05	4.50E-92
f_4	8.18E+01	7.59E+01	1.73E-03	1.54E-41
f_5	2.16E+08	2.29E+08	2.29E+01	1.87E+01
f_6	5.48E+04	6.18E+04	0.00E+00	0.00E+00
f_7	4.43E+01	3.24E-01	1.30E-01	3.70E-06
f_8	1.07E+04	1.06E+04	4.84E+03	3.11E+03
f_9	3.12E+02	3.61E+02	6.47E+01	0.00E+00
f_{10}	2.03E+01	2.05E+01	4.23E-04	2.05E-14
f_{11}	5.47E+02	6.09E+02	9.86E-03	0.00E+00
f_{12}	3.99E+08	6.18E+08	5.04E-08	1.57E-32
f_{13}	8.12E+08	9.13E+08	6.06E-07	3.66E-04

图 4.1 给出了 FA、MFA、VSFA、WSFA 和 POFA 之间在求解 13 个 benchmark 函数时的收敛曲线。并且为了方便图中比较观察，对函数f_1~f_4、f_7、f_{10}~f_{13}进行了对数转化，其中横坐标为迭代次数，纵坐标为适应值（越小越好）。从图 4.1

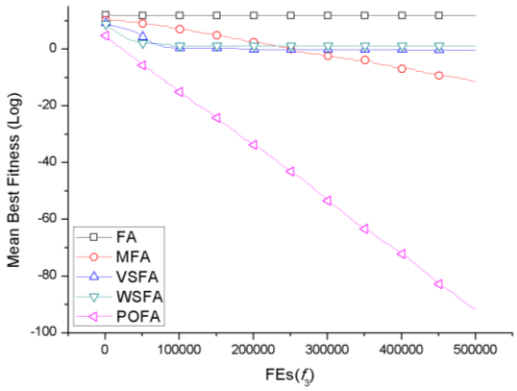
中可以看出，与其他四种算法相比，除了 f_8 和 f_13 外，POFA 在收敛速度和求解质量上都有很好的性能。



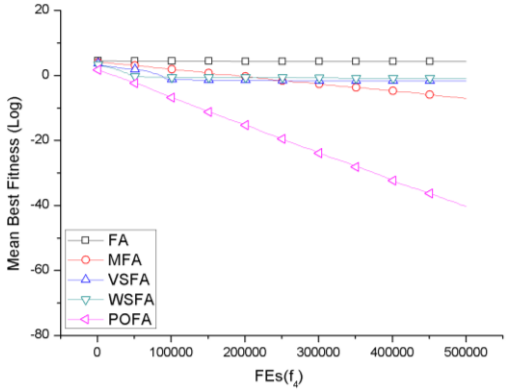
(a)Sphere (f_1)



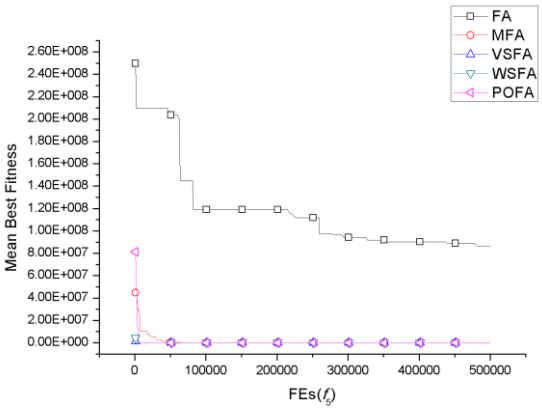
(b)Schwefel 2.22(f_2)



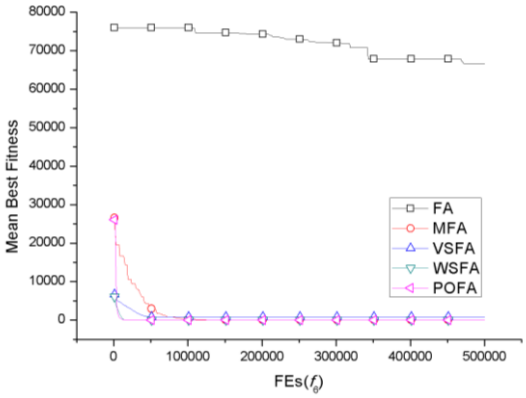
(c) Schwefel 1.2(f_3)



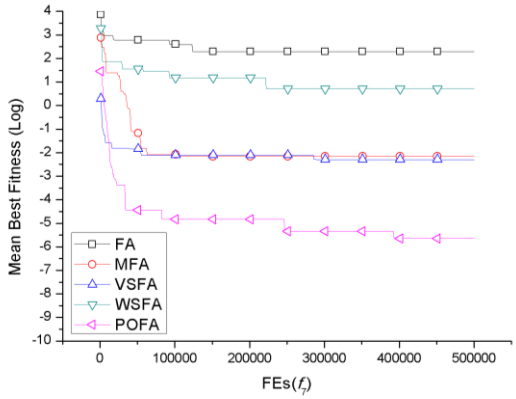
(d) Schwefel 2.21 (f_4)



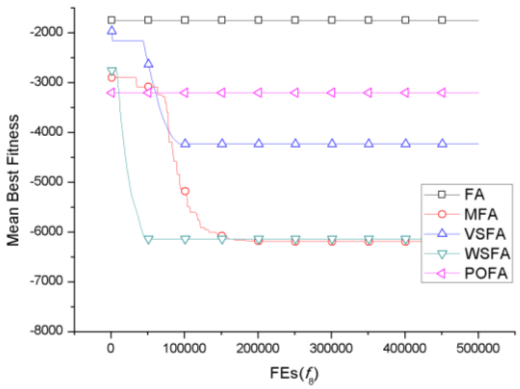
(e)Rosenbrock(f_5)



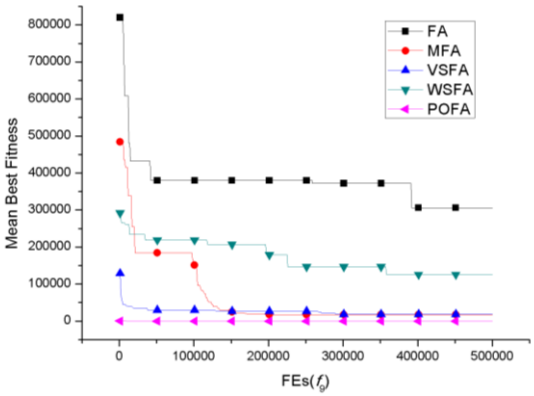
(f)Step(f_6)



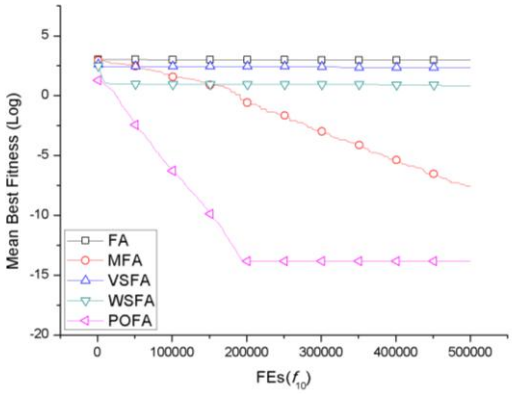
(g)Quartic with noise(f_7)



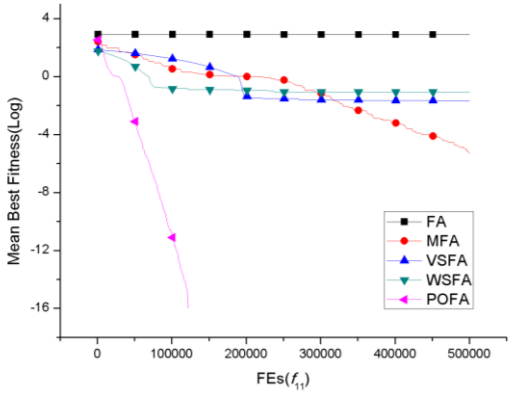
(h)Schwefel 2.26(f_8)



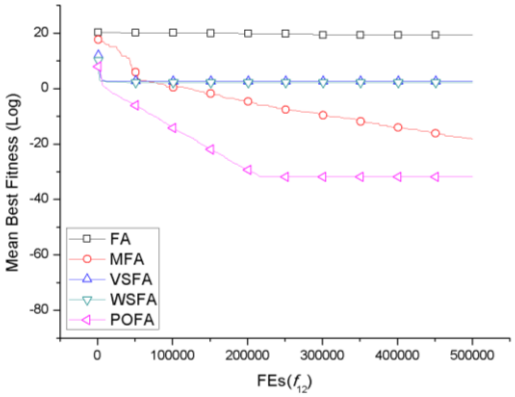
(i)Rastrigin(f_9)



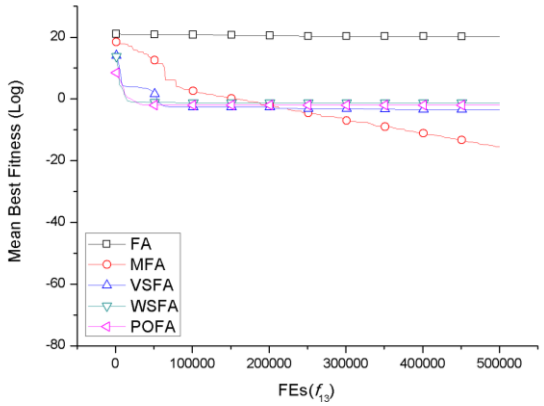
(j) Ackley(f_{10})



(k)Griewank(f_{11})



(l)Penalized 1(f_{12})



(m)Penalized 2(f_{13})

图4.1 FA，MFA，VSFA，WSFA，POFA算法收敛图

第 5 章 改进萤火虫算法在车间调度中的应用

5.1 车间调度作业基本概念

5.1.1 车间调度问题的描述与分类

首先对于调度问题一般定义为：将有限资源在满足一定的约束条下分配给若干项目，使得单个或多个目标尽可能最优化。通常将每个项目都按照最早时间执行，则该任务执行排序即为一个调度方案，而所有项目执行完成后，所需优化的目标最优时该调度方案即为最优调度方案。

在各种生产制造中，车间调度问题定义为：将 n 个需要经过多道工序的工件安排在 m 太机器上加工，并且相同工件的工序需要按一些可行的顺序进行加工。而调度的目标是通过合理的调度方案将若干工件工序安排的若干机器上加工，使得时间、成本的目标性能最优化。

对于车间调度问题的分类，其方式较多，而根据工件和车间机器构成可以将问题大致分为五类。

（1）单机调度问题（single machine scheduling problem，SMP），即工件只有一个加工工序且所有工件只在一台机器上加工。一般当车间机器只有一台运行时，工件工序的调度即为该调度。

（2）并行机调度问题（parallel machine scheduling problem，PMP），在该问题中工件同样只有一道工序，但可以任意在有若干台功能相同的机器上加工。

（3）开放车间调度问题（open shop scheduling problem，OSP），每个工件有一道或多道工序且同一工件工序没有先后约束关系，即机器可以从任何一个工序开始，任何一个工序结束，工序之间没有等待时间。

（4）流水车间调度问题（flow shop scheduling problem，FSP），车间有多台用于加工不同工序的机器，而工件有多道工序且工序在每台机器上的加工的顺序是固定的，且所有工件的工序顺序是相同的。

（5）作业车间调度问题（job shop scheduling problem，JSP），车间有多台加工不同工序的机器，每个工件也有多道工序且工序在每台机器上的加工的顺序是固定的，但每个工件的工序顺序是不同的。

综合各类调度问题可以得到，它们具有：多约束性，离散性，计算复杂性，不确定性，多目标性等特点。而这些问题特性也吸引着中众多学者对调度问题的求解进行不懈的努力。多年来，国内外学者也提出了不同的解决方法来满足人们生活生产对求解该类似问题要求日益增高的需求。

5.1.2 车间调度问题的研究概况

生产调度是制造系统中最关键的问题之一，在文献[39]中得到了广泛的研究。生产调度是指将可用的生产资源分配给任务，并决定操作的顺序，以便满足所有约束条件并优化目标性能[40]。最著名的产品调度问题之一是作业车间调度问题（JSP），它是 NP-hard[41]。在 JSP 中，一组作业将在有限的机器集中处理。根据其生产流程，每个作业在给定加工时间的机器上加工，每个机器只能处理一个作业[42]。Flexible JSP（FJSP）是 JSP 的一个扩展，也是 NP-hard[43]，与 JSP 不同，FJSP 中每个操作都有一组可供选择的机器，这增加了调度的灵活性和复杂性[6]。因此，FJSP 比 JSP 更复杂。

柔性作业车间调度问题（FJSP）一般定义为：一组 n 个作业 $J = \{J_1, J_2, \cdots, J_n\}$将在一组 m 机器 $M = \{M_1, M_2, \cdots, M_m\}$上进行处理。每个作业 J_i 的操作排列$\{O_{i1}, O_{i2}, \cdots, O_{ipi}\}$根据给定的顺序进行处理。 每个操作可以在 M 的多台机器上进行处理。 FJSP 将确定每种操作最合适的机器（称为机器选择）以及机器上的操作顺序（称为操作顺序）。 FJSP 的优化目标是最小化某些指标，例如，加工时间，最大延迟和总流动时间。此外，FJSP 还存在如下一些约束：

（1）所有机器在初始可用。

（2）每个工序操作一次只能在一台机器上处理。

（3）每台机器一次只能执行一项工序操作。

（4）在处理过程中不能中断每个工序操作。

（5）在不同工件作业的工序操作之间没有优先级限制，工件作业是彼此独立的。

（6）对于每个工件作业，工序操作顺序是固定的。

根据是否可以在所有机器上处理所有操作，FJSP 可以分为两类，即总 FJSP（T-FJSP）和部分 FJSP（P-FJSP）[44]。它们描述如下：（i）T-FJSP， 每个操作都可以在所有机器上处理；（ii）P-FJSP， 至少有一项操作只能在所有计算机上进行。 T-FJSP 可以视为 P-FJSP 的特殊情况。FJSP 在现实世界中有着重要的应用，由于其复杂性和重要性，人们对解决这一问题进行了大量的关注。具有各种优化目标的 FJSP 也已得到广泛研究。

基于种群的元启发式算法（Population-based meta-heuristics）已经广泛应用于求解 FJSP 问题，其中遗传算法（GA）可能是应用最广泛的方法。该遗传算法不仅有效地解决了单目标模糊综合评判问题，而且有效地解决了多目标模糊综合评判问题。

首先介绍单目标模糊综合评判问题的求解方法，其中最常用的目标是 makespan。为了解决这一问题，人们提出了许多基于遗传算法的元启发式算法。例如，Pezzella 和 Ciaschetti[43]为 FJSP 提供了一个 GA。该算法综合了初始种群

生成、个体选择和新解生成的不同策略。Kacem 等人[44]提出了两种求解 FJSP 的方法。第一个是定位，第二个是遗传算法，他们应用遗传操作来提高解的质量。Gao 等人[45]开发了一种可变邻域下降的混合遗传算法，以实现最小制造时间、最大机器工作负荷和总工作负荷三个目标，提出了一种新颖的双矢量表示方案和一种有效的解码方法，将每个染色体解释为一个活动的序列。Zhang 等人[46]提出了一种有效的遗传算法，用于求解使完工时间最小化的 FJSP 问题。在该算法中，提出了一种改进的染色体表示方案，并设计了一种有效的解码方法，将每个染色体分解为一个可行的活动调度表。

基于遗传算法和分组遗传算法，Chen 等人[47] 开发了一个求解 FJSP 的调度算法，其目标是最小化多个性能指标，包括总延误、总机器空闲时间和制造时间。Chang 等人[48]提出了一种在交配后嵌入田口方法的遗传算法，以提高求解 FJSP 的效率。Nouri 等人[49]提出了一个求解 FJSP 的 holonic 多代理模型中两个元启发式的混合。调度代理采用基于邻域的遗传算法进行全局搜索，集群代理采用局部搜索技术。Jiang 和 Du[50]提出了一种改进的遗传算法，采用新的初始化方法来提高初始种群的质量，加快算法的收敛速度。同样的问题，Huang 等人[51]还提出了一种改进的遗传算法，设计了两种有效的交叉方法和两种变异方法。随后，他们又提出了一种改进的遗传算法，在交配过程中具有新的自适应交叉和变异概率[52]，从而大大提高了收敛速度。Driss 等人[53]提出了一个遗传算法来解决 FJSP 问题，以最小化制造时间，采用新的染色体表示法方便地表示解，并相应地设计了特殊的交叉和变异算子。Purnomo[54]开发了一个基于知识的改进遗传算法来解决 FJSP 问题，为了提高算法的性能，在遗传算法中嵌入了一些启发式算法。Morinaga 等人[55]开发了一个遗传算法，它利用启发式调度规则中包含的知识来求解 FJSP，同时执行机器选择和作业选择，以减轻对解决方案空间的搜索不足。为了提高遗传算法的开发能力，许多研究者将遗传算法与局部搜索启发式算法相结合。Li 和 Gao[56]提出了一种混合算法，将遗传算法和 TS 算法混合用于 FJSP。Gu 等人[57]提出了一种改进的模拟退火遗传算法（ISAGA）来求解 FJSP 问题。在 ISAGA 中，利用云模型理论中的 X 条件云发生器产生变异概率，并针对结果的变异性设计了模拟退火（SA）操作。Zhang 等人[58]开发了一个基于遗传算法的可变邻域搜索（VNS）来处理 FJSP。在他们提出的算法中，一些简单的局部搜索方法被用来平衡勘探和开发。Ma 等人[59]提出了一种求解 FJSP 的模因算法（MA）。MA 是一种混合遗传算法，结合两个有效的局部搜索来利用搜索区域中的信息。Cinar 等人[60]提出了一种基于优先级表示的遗传算法来求解 FJSP。每个操作的优先级由染色体上的一个基因表示。为了得到更好的解，迭代局部搜索（ILS）被嵌入到算法中。

5.2 改进萤火虫算法解决单目标柔性车间调度问题

5.2.1 问题与算法设计

因为萤火虫优化算法适应于连续优化问题，而生产调度属于离散组合优化问题。且萤火虫优化算法中个体的位置为连续值矢量，无法实现工件排序的更新，因此，在参数自适应调节的萤火虫优化算法基础上，构造从个体位置矢量到工件排序的恰当映射是用萤火虫优化算法解决调度问题的另一个关键科学问题。而解决该问题的关键为构造从个体位置矢量到工件排序的恰当映射是用萤火虫优化算法解决调度问题。因此本文在现有调度算法的基础上，建立了一套高效编码和解码方法，通过随机键排列实现从萤火虫个体的连续位置矢量到工件排序的转换，然后计算该个体携带信息的映射的调用方案所需时间性能赋予萤火虫个体的亮度值，进而使得萤火虫优化算法适合于求解生产调度问题。

虽然个体的位置矢量 $X_i = [x_{i,1}, x_{i,2}, ..., x_{i,n}]$本身无法表示工件的加工排序，但各个位置分量的值有大小次序关系。文献[5]就是利用这种次序关系，设计了基于SPV(smallest position value)规则离散萤火虫优化算法，并用来成功求解混合流水线车间调度问题。但这种规则的计算时间复杂性高，不是很适合大规模的调度问题。基于此，本文设计了更简单高效的规则，例如，考虑将 3 个且工序都为 2 的工件设置于 3 台机器上加工的流水线调度问题，将萤火虫优化算法中的个体位置设置为 6 维矢量，假设随机位置矢量 X_i = [0.07，2.81，1.86，3.71，1.89，0.51]，则首先赋予值最小 $x_{i,1}$ 的分量为值 1，接下来赋予 $x_{i,6}$ 对应的分量位置为值 2，然后分别赋予 $x_{i,3}$，$x_{i,5}$，$x_{i,2}$ 和 $x_{i,4}$ 对应分量位置值为 3，4，5 和 6，从而得到工件的加工次序，即π = (1,2,0,0,1,2)，其关系如表 5.1 所示，其中 $J(a,b)$表示工件 a 的第 b 道工序。而当确定工件工序后，同时根据个体工序信息随机确定可行的机器编码序列，并保存于进化种群个体中，如设可行的机器编码：M = (1, 0, 2,1,1,0)。而在种群演化时，机器编码序列也将向着更优个体的机器码进行移动，以及进行随机变异。

表5.1 个体的位置及其对应的排列值

分量位置	1	2	3	4	5	6
位置分量值	0.07	2.81	1.86	3.71	1.89	0.51
排列值	1	5	3	6	4	2
个体编码	1	2	0	0	1	2
个体解码	J(1,1)	J(2,1)	J(0,1)	J(0,2)	J(1,2)	J(2,1)

5.2.2 实验设置

为验证 5.2.1 小节中的设计算法的有效性，这里设计将用《Matlab 智能算法 30 个案例分析》第二版中的第 11 章[61]中的实验用例进行验证。实验中将 6 个工序都为 6 的工件分配到 10 台功能不同的机器上生产调度。每个工序加工使用的机器编号如表 5.2 所示，其中每道工件工序有 1 或多台机器可以用于加工，并且每个工件工序的生产顺序须按表中工序 1 至工序 6 的顺序进行加工生产。对应的各工件工序在相应机器的加工时间如表 5.3 所示。

表5.2　工序可选机器

	工件1	工件2	工件3	工件4	工件5	工件6
工序1	3,10	2	3,9	4	5	2
工序2	1	3	4,7	1,9	2,7	4,7
工序3	2	5,8	6,8	3,7	3,10	6,9
工序4	4,7	6,7	1	2,8	6,9	1
工序5	6,8	1	2,10	5	1	5,8
工序6	5	4,10	5	6	4,8	3

表5.3　工序加工时间

	工件1	工件2	工件3	工件4	工件5	工件6
工序1	3,5	6	1,4	7	6	2
工序2	10	8	5,7	4,3	10,12	4,7
工序3	9	1,4	5,6	4,6	7,9	6,9
工序4	5,4	5,6	5	3,5	8,8	1
工序5	3,3	3	9,11	1	5	5,8
工序6	10	3,3	1	3	4,7	3

在实验中，算法的基本参数为：种群大小为 20，最大评估次数为 5.0E+05，运行次数为 30。

5.2.3 结果与分析

表 5.4 记录了算法对 5.2.2 中实验设置的问题的求解结果，从表中可以看出，经过算法优化求得该 FJSP 问题中全部工件完成的最短时间为 48 秒，平均最终优化结果为 48.3 秒。算法优化过程如图 5.1 所示，图中横坐标 1 个单位表示算法 30 次独立运行中，对应每 1000 次评估时当前种群中最优个体适应度值的平均值。并从图 5.1 中可看到，经过算法搜索优化最终调度平均时间从初始的 50.4 秒减小到了 48.3 秒，即经 5.2.1 小节中的设计的优化算法将测试的 FJSP 调度问题平均

减小 2.1 秒，从而论证该算法编码设计对 FJSP 问题求解的有效性。

表5.4 算法求解FJSP实验结果数据

Worst	Best	Mean	Std Dev
54	48	48.3	1.18

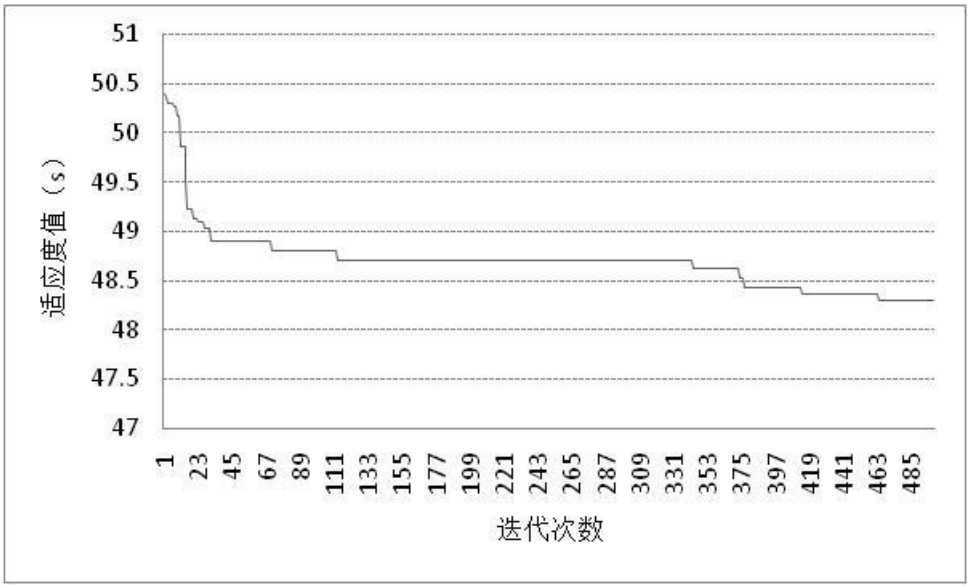


图 5.1 算法 30 次优化平均最优适应值变化趋势

在实验算法 30 次优化中，最优个体对应的工件加工甘特图如图 5.2 所示，其加工完所有工件工序的最短时间为 47s，个体原始编码为（6, 2, 6, 5, 3, 3, 1, 1, 4, 4, 2, 5, 3, 4, 2, 1, 5, 3, 6, 3, 6, 5, 1, 4, 6, 1, 6, 4, 2, 2, 5, 5, 1, 3, 4, 2）。图 5.2 中，横轴表示机器加工工件的时间，纵轴表示加工机器编号，在时间块标签中第一位表示工件序号，第二位表示该工件工序的序号，如标签“61”表示第 6 个工件的第 1 道工序。

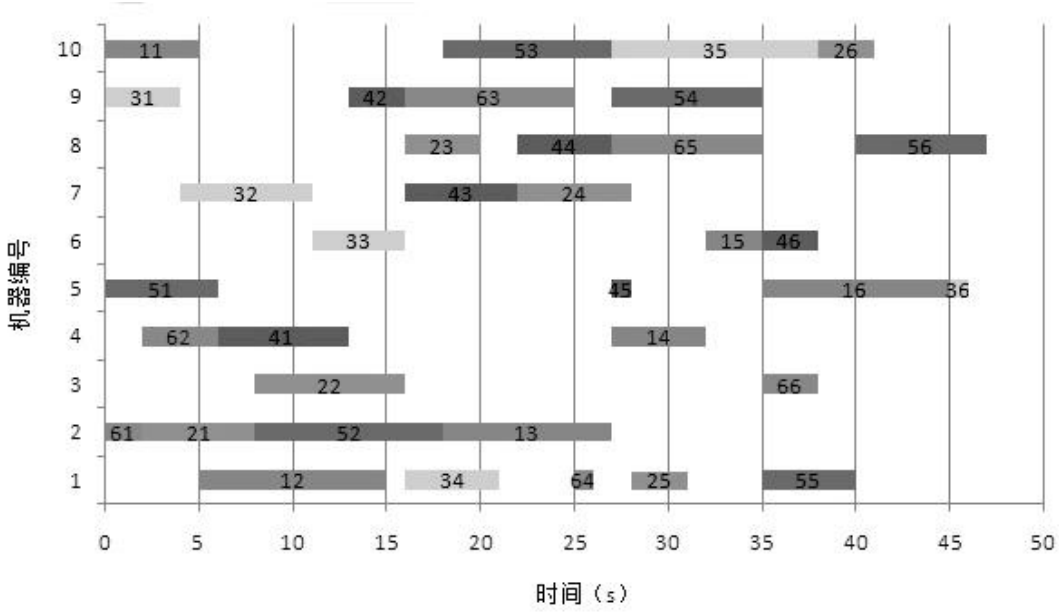

机器编号
10
9
8
7
6
5
4
3
2
1
11
53
35
26
31
42
63
54
23
44
65
56
32
43
24
33
15
46
51
45
16
36
62
41
14
22
66
61
21
52
13
12
34
64
25
55
0
5
10
15
20
25
30
35
40
45
50
时间（s）

图 5.2　工件加工甘特图

第 6 章 总结与展望

论文主要对萤火虫优化算法自身改进及在车间调度问题上的运用进行研究。前两章介绍了选题研究意义，以及智能算法和 FA 相关的基本概念。第三章优化了 FA 的步长和吸引力参数，而第四章则通过与反向学习策略的融合改进了 FA 的寻优性能。第五章将改进的 FA 运用于柔性作业车间调度问题的求解，并取得良好效果。

6.1 本文主要工作及结论

萤火虫优化算法是基于萤火虫物种特性提出的仿生算法，属于典型的进化算法，同时也具有算法后期收敛慢，易陷于局部最优的缺点。现从以下三个方面对本文的主要研究工作进行总结：

（1）论文首先介绍了选题的研究背景及意义，然后阐述了最优化问题的概念、分类，以及简单列出了问题现有的求解方法。接着讨论了进化计算，智能优化算法的优缺点，并在第一章的最后小节总结了本文的行文结构。而在第二中，文章同样先对萤火虫优化算法的仿生原理、基本概念等进行介绍，并详细讲解 FA 中个体的亮度、吸引力、距离和运动公式。最后论述了 FA 的研究现状。

（2）对于 FA 的改进研究，本文先是通过对算法步长的分析提出了自适应步长策略。为防止 FA 过早收敛，算法增加了对步长的控制结构，使得步长自适应算法在解空间所搜最优解时，不同阶段对步长大小不同的需求情形。接着 FA 吸引力的分析，论文提出算法基于比例模型的改进。改进后，算法后期不再以趋于 1 的比例进行移动。取而代之的时在个体相互吸引力时，将初始吸引力设为 0.4 使得种群在演化过程中近可能保留种群多样性。而在上述改进 FA 进行 benchmark 函数测试后，实验结果表明相比于改进前的 FA，改进后的 FA 在求解测试函数方面具有突出优势。同样在第四章中，为增加萤火虫种群的多样性，通过 FA 中引入反向学习策略，使得 FA 具有的更强的跳出局部最优解的能力，且实验数据表明，该改进也使得 FA 对最优化问题的求解能力得到了提高。

（3）由于最初萤火虫优化算法是主要针对连续优化问题的求解提出的，所以其自然对于离散优化问题的求解能力较弱。为了使 FA 具有对柔性作业车间调度问题（FJSP）求解的能力，文章中设计了一套将 FA 应用于求解 FJSP 的编码与解码方案。方案中，将工件工序在机器中的加工序列转换至萤火虫个体在解空间的位置信息中，并使 FJSP 中的时间目标性能对应于萤火虫个体亮度，从而通过算法演化求得工件加工的最短时间。并经实验论证，该算法编码对求解柔性作业车间调度问题的有效性。

6.2 未来工作展望

萤火虫优化算法作为在进化计算领域较新颖的算法之一，然后经过国内外许多学者的研究改进，但其仍然具有很大的改进空间。如接下来可以从一下几个方面进行更深入学习研究：

（1）作为元启发式算法，FA 同样具有数学理论薄弱，算法不够严谨的缺点，因此，尚需广大学者对其进行严谨的数学理论推导研究，以证明其收敛性。

（2）本文主要针对的是单目标优化低维问题的求解，而对现实的工程优化问题往往需要算法具有对高维多目标的求解能力。因此，下一步可以在将 FA 用于解决高维甚至超高维和多目标的复杂优化问题方面进行研究，提高 FA 求解优化问题的能力。

（3）如前所述，因为 FA 具有较低的求解离散优化问题（如 FJSP）的能力，所以设置合理有效及普适的离散编码方案，以提高 FA 对该类问题的处理能力同样也是值得研究的领域。

参考文献

[1] Sobeyko and L. Mönch, Integrated process planning and scheduling for large-scale flexible job shops using metaheuristics[J]. International Journal of Production Research, 2017, vol. 55, pp. 392-409.

[2] P. Smet and G. Vanden Berghe, Large neighbourhood search for large-scale shift scheduling problems with multiple tasks[C]. in Proceedings of the 11th international conference on the practice and theory of automated timetabling, 2016.

[3] A. T. S. Al-Obaidi and S. A. Hussein, Two Improved Cuckoo Search Algorithm to Solve Flexible Job-Shop Scheduling Problem[J]. International Journal on Perceptive and Cognitive Computing, 2016, vol. 2.

[4] J. Q. Li and Q. K. Pan, Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm[J]. Information Sciences, 2015, vol. 316, pp. 487-502.

[5] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems[J]. IEEE transactions on evolutionary computation, 2014, vol. 18, pp. 301-305.

[6] X. Li, K. Tang, P. N. Suganthan, and Z. Yang, Editorial for the special issue of Information Sciences Journal (ISJ) on "Nature-inspired algorithms for large scale global optimization" [J]. Information Sciences, 2015, vol. 316, pp. 437-439.

[7] X.-S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications: Wiley Publishing, 2010.

[8] X.-S. Yang, Nature-Inspired Metaheuristic Algorithms: Luniver Press, 2008.

[9] J. S. Chou and A. D. Pham, Nature-inspired metaheuristic optimization in least squares support vector regression for obtaining bridge scour information[J]. Information Sciences2017, vol. 399,.

[10] P. Baumgartner, T. Bauernfeind, O. Biro, A. Hackl, C. Magele, W. Renhart, et al., Multi-Objective Optimization of Yagi-Uda Antenna Applying Enhanced Firefly Algorithm With Adaptive Cost Function[J]. IEEE Transactions on Magnetics, 2017, vol. PP, pp. 1-4.

[11] L. Xiao, W. Shao, T. Liang, and C. Wang, A combined model based on multiple seasonal patterns and modified firefly algorithm for electrical load forecasting[J].Applied Energy, 2016, vol. 167, pp. 135-153.

[12] M. Alb, P. Alotto, C. Magele, W. Renhart, K. Preis, and B. Trapp, Firefly Algorithm for Finding Optimal Shapes of Electromagnetic Devices[J]. IEEE Transactions on Magnetics, 2016, vol. 52, pp. 1-4.

[13] J. Wang, Z. Wu, and H. Wang, Hybrid differential evolution algorithm with chaos and generalized opposition-based learning[J]. in Advances in Computation and Intelligence, ed: Springer, 2010, pp. 103-111.

[14] H. Wang, Z. Wu, S. Y. Zeng, D. Jiang, Y. Liu, J. Wang, et al., A Simple and Fast Particle Swarm Optimization[J]. Multiple-Valued Logic and Soft Computing, 2010, vol. 16, pp. 611-629.

[15] H. Wang, Z. Wu, S. Rahnamayan, and J. Wang, Diversity Analysis of Opposition-Based Differential Evolution—An Experimental Study[J]. in Advances in Computation and Intelligence, ed: Springer, 2010, pp. 95-102.

[16] 张立卫. 最优化方法[M]. 北京: 科学出版社, 2010.

[17] X.-S. Yang, Nature-inspired Metaheuristic algorithms, 2nd edn. Luniver Press, 2010.

[18] Kumar D, Gandhi B G R, Bhattacharjya R K. Firefly Algorithm and Its Applications in Engineering Optimization[M], Nature-Inspired Methods for Metaheuristics Optimization. Springer, Cham, 2020: 93-103.

[19] Farahani S, Abshouri A, Nasiri B, Meybodi M Some hybrid models to improve firefly algorithm performance[J]. Int J Artif Intell, 2012, 8(S12):97–117.

[20] Farook S, Raju P Evolutionary hybrid genetic-firefly algorithm for global optimization[J]. Int J Comput Eng Manag, 2013, 16(3):37–45.

[21] Rizk-Allah R, Zaki E, El-Sawy A Hybridizing ant colony optimization with firefly algorithm

for unconstrained optimization problems[J]. Appl Math Comput, 2013, 224(3):473–483
[22] C. Liu, Y. Zhao, F. Gao, and L. Liu, Three-Dimensional Path Planning Method for Autonomous Underwater Vehicle Based on Modified Firefly Algorithm[J]. Mathematical Problems in Engineering,2015,(2015-10-12), vol. 2015, pp. 1-10, 2015.
[23] S. Goel and V. K. Panchal, Performance evaluation of a new modified firefly algorithm," in International Conference on Reliability[J]. INFOCOM Technologies and Optimization, 2015, pp. 1-6.
[24] G. Wang, L. Guo, D. Hong, L. Luo, and H. Wang, A Modified Firefly Algorithm for UCAV Path Planning[J]. International Journal of Hybrid Information Technology, vol. 5, 2012.
[25] S. Yu, S. Yang, and S. Su, Self-Adaptive Step Firefly Algorithm[J]. Journal of Applied Mathematics,2013,(2013-11-2), vol. 2013, pp. 610-614, 2013.
[26] R. Selvarasu, M. S. Kalavathi, and C. C. A. Rajan, SVC placement for voltage constrained loss minimization using self-adaptive Firefly algorithm[J]. Archives of Electrical Engineering, vol. 62, pp. 649-661, 2013.
[27] R. Selvarasu and M. S. Kalavathi, TCSC Placement for Loss Minimization using Self Adaptive Firefly Algorithm[J]. Journal of Engineering Science & Technology, vol. 10, pp. 291-306, 2015.
[28] H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, Firefly algorithm with chaos[J]. Communications in Nonlinear Science & Numerical Simulation, vol. 18, pp. 89-98, 2013.
[29] S. Jansi and P. Subashini, A Novel Fuzzy Clustering based Modified Firefly Algorithm with Chaotic Map for MRI Brain Tissue Segmentation.
[30] K. Al-Wagih, Improved Firefly Algorithm for Unconstrained Optimization Problems[J]. International Journal of Computer Applications Technology & Research, vol. 4, pp. 77-81, 2014.
[31] H. Wang, X. Zhou, H. Sun, X. Yu, J. Zhao, H. Zhang, et al., Firefly algorithm with adaptive control parameters[J]. Soft Computing, 2016.
[32] Fister Jr I, Yang X S, Fister I, et al. Memetic firefly algorithm for combinatorial optimization[J]. arXiv preprint arXiv: 2012, 1204.5165.
[33] P. Baumgartner, T. Bauernfeind, O. Biro, A. Hackl, C. Magele, W. Renhart, R. Torchio, Multi-Objective Optimization of Yagi-Uda Antenna Applying Enhanced Firefly Algorithm With Adaptive Cost Function[J],IEEE Transactions on Magnetics.2018, 54(3):1-4.
[34] S. Rahnamayan, G.G. Wang, M. Ventresca, An intuitive distance-based explanation of opposition-based sampling[J]. Appl. Soft Comput. 2012 12 (9) 2828–2839.
[35] S. Rahnamayan, Opposition-based differential evolution. 2007.
[36] B.E. Emilio, E. Cuevas, et al. Optimal power flow solution using modified flower pollination algorithm[C]. in 2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), IEEE, 2015, pp. 1–6.
[37] H. Dhahri, A.M. Alimi, Opposition-based differential evolution for beta basis function neural network[C], in Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE, 2010, pp. 1–8.
[38] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in null, pages 695-701. IEEE, 2005.
[39] Xia, W., Wu, Z. An effective hybrid optimization approach for multiobjective flexible job-shop scheduling problems[J]. Comput. Ind. Eng., 2005, 48, (2), pp. 409–425.
[40] Akyol, D.E., Bayhan, G.M. A review on evolution of production scheduling with neural networks[J]. Comput. Ind. Eng., 2007, 53, (1), pp. 95–122.
[41] Gonçalves, J.F., de Magalhães Mendes, J.J., Resende, M.G. A hybrid genetic algorithm for the job shop scheduling problem[J]. Eur. J. Oper. Res., 2005, 167, (1), pp. 77–95.
[42] Çaliş, B., Bulkan, S. A research survey: review of AI solution strategies of job shop scheduling problem[J]. J. Intell. Manuf., 2015, 26, (5), pp. 961–973.
[43] Pezzella, G.M., Ciaschetti, G. A genetic algorithm for the flexible job-shop scheduling problem[J]. Comput. Oper. Res., 2008, 35, (10), pp. 3202–3212.
[44] Kacem, I., Hammadi, S., Borne, P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems[J]. IEEE Trans. Syst. Man Cybern. C, Appl. Rev., 2002, 32, (1), pp. 1–13.
[45] Gao, J., Sun, L., Gen, M. A hybrid genetic and variable neighborhood descent algorithm for

flexible job shop scheduling problems[J]. Comput. Oper. Res., 2008, 35, (9), pp. 2892–2907.
[46] Zhang, G., Gao, L., Shi, Y. An effective genetic algorithm for the flexible job-shop scheduling problem[J]. Expert Syst. Appl., 2011, 38, (4), pp. 3563–3573.
[47] Chen, J.C., Wu, C.-C., Chen, C.-W., et al. Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm[J]. Expert Syst. Appl., 2012, 39, (11), pp. 10016–10021.
[48] Chang, H.-C., Chen, Y.-P., Liu, T.-K., et al. Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchigenetic algorithm[J]. IEEE. Access., 2015, 3, pp. 1740–1754.
[49] Nouri, H.E., Driss, O.B., Ghédira, K. Hybrid metaheuristics within a holonic multiagent model for the flexible job shop problem[J]. Procedia Comput. Sci., 2015, 60, pp. 83–92.
[50] Jiang, L., Du, Z. An improved genetic algorithm for flexible job shop scheduling problem[C]. 2015 2nd Int. Conf. on Information Science and Control Engineering (ICISCE), Shanghai, China, 2015, pp. 127–131.
[51] Huang, M., Mingxu, W., Xu, L. An improved genetic algorithm using opposition-based learning for flexible job-shop scheduling problem[C] 2016 2nd Int. Conf. on Cloud Computing and Internet of Things (CCIOT), Dalian, China, 2016, pp. 8–15.
[52] Huang, M., Wang, L.-M., Liang, X. An improved adaptive genetic algorithm in flexible job shop scheduling[C] 2016 2nd Int. Conf. on Cloud Computing and Internet of Things (CCIOT), Dalian, China, 2016, pp. 177–184.
[53] Driss, I., Mouss, K.N., Laggoun, A. An effective genetic algorithm for the flexible job shop scheduling problems[C] 11th Congres Int. de Genine Industriel – CIGI2015, Québec, Canada, 2015, pp. 26–28.
[54] Purnomo, M.R.A. A knowledge-based genetic algorithm for solving flexible job shop scheduling problem[J]. Int. Bus. Manag., 2016, 10, (19), pp. 4708– 4712.
[55] Morinaga, E., Sakaguchi, Y., Wakamatsu, H., et al. A method for flexible job-shop scheduling using genetic algorithm[J]. J. Adv. Manuf. Technol., 2017, 11, (2), pp. 79–86.
[56] Li, X., Gao, L. An effective hybrid genetic algorithm and TS for flexible job shop scheduling problem[J]. Int. J. Prod. Econ., 2016, 174, pp. 93–110.
[57] Gu, X., Huang, M., Liang, X. The improved simulated annealing genetic algorithm for flexible job-shop scheduling problem[C] 2017 6th Int. Conf. on Computer Science and Network Technology (ICCSNT), Dalian, China, 2017, pp. 22–27.
[58] Zhang, G., Zhang, L., Song, X., et al. A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem[J]. Cluster Comput., 2018, pp. 1–12, doi: 10.1007/s10586-018-2328-3.
[59] Ma, W., Zuo, Y., Zeng, J., et al. A memetic algorithm for solving flexible job-shop scheduling problems[C] 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 2014, pp. 66–73.
[60] Cinar, D., Oliveira, J.A., Topcu, Y.I., et al. A priority-based genetic algorithm for a flexible job shop scheduling problem' [J]. J. Ind. Manag. Optim., 2016, 12, (4), pp. 1391–1415.
[61] 郁磊，史峰，王辉等. Matlab 智能算法 30 个案例分析[M]. 北京：北京航空航天大学出版社, 2015: 108-117.

攻读硕士学位期间的学术活动及成果情况

（1）参加的科研项目

[1] 面向大规模调度问题的并行参数自适应萤火虫优化算法研究（编号：6186020260），国家自然科学基金，2019 年-- 2022 年。

（2）发表的学术论文

[1] J. Wang， G. Liu， A novel firefly algorithm with self-adaptive step strategy[J]. International Journal of Innovative Computing and Applications 10.1 (2019): 18-26.（对应本文 3.1 小节）

[2] J. Wang，G. Liu，W.W. Song，Firefly Algorithm with Proportional Adjustment Strategy[C]. International Workshop on Data Quality and Trust in Big Data， Springer， 2018: 78-93.（对应本文第 4 章）

[3] 汪靖，刘桂元. 基于动态步长变化的萤火虫算法[J]. 计算机工程与设计，2019，40(04):1001-1007. （对应本文 3.2 小节）

致谢

时光不与千秋老，岁月不与万年长。在江西财经大学的研究生三年转眼翻之即过，其间有喜有忧，回想起当初考研时激动而来，而今却将不舍离去，心中感慨不已。万情于心，欲诉无言，唯有深藏之，相信在江西财经大学的这三年学习生活将会是我一生中宝贵的财富。而在这三年读研间期，不管从学术知识上还是做人道理上，都让我成长了许多，受益终生。在此，我衷心向所有帮助，指引，激励我的人们表示最真心的感谢。

首先，我向我的导师汪靖老师致以诚挚的敬意和由衷的感谢。从论文的选题到论文的完成，他给予了我悉心的指导，严格的审查，使我获益颇多。在求学方面，汪老师严谨态度治学、循循善诱教学；在生活方面，老师于我亦师亦友，教会我许多做人处事的道理，相信这些对我今后方方面面的发展产生深深的影响。在此，再次向老师表示深深的敬意。

其次，我要感谢我所有的老师、同学以及母校。三年里，与同学朝夕相处，学习中，大家彼此勉励、共同进步；生活中，大家和睦相处，分享喜悦。我们共创了这难忘的三年研究生时光。同时由衷感谢江西财经大学这所美丽的校园，是她为我提供了学习知识的土壤，使我在这里茁壮成长；是她为我留下了这三年最美好的回忆。

最后，也特别感谢百忙之中审阅论文的各位老师。