

单位代码: 10359
学 号: 2019H11266

密 级: 公开
分 类 号: TP18



合肥工业大学
Hefei University of Technology

硕士学位论文

MASTER'S DISSERTATION

(学术硕士)

论文题目: 改进粒子群优化算法及应用

学科专业: 数 学

作者姓名: 吴浩然

导师姓名: 霍星 教授

完成时间: 2022 年5月



单位代码： 10359

学 号： 2019111266

密 级： 公开

分类号： TP18

合肥工业大学

Hefei University of Technology

硕士学位论文

MASTER'S DISSERTATION

论文题目： 改进粒子群优化算法及应用

学位类别： 学历硕士

专业名称： 数学

作者姓名： 吴浩然

导师姓名： 霍星 教授

完成时间： 2022 年 5 月

合 肥 工 业 大 学

学历硕士学位论文

改进粒子群优化算法及应用

作者姓名：_____吴浩然_____

指导教师：_____霍星 教授_____

学科专业：_____数学_____

研究方向：_____计算数学_____

2022 年 5 月

A Dissertation Submitted for the Degree of Master

**Improved Particle swarm optimization algorithm and
its application**

By

Wu Haoran


Hefei University of Technology
Hefei, Anhui, P.R.China
May, 2022

合 肥 工 业 大 学


本论文经答辩委员会全体委员审查,确认符合合肥工业大学学历硕士学位论文质量要求。

答辩委员会签名(工作单位、职称、姓名)


主席: 中国科学技术大学 教授 

委员: 合肥工业大学 教授 

合肥工业大学 教授 

合肥工业大学 副教授 

合肥工业大学 教授 

导师: 合肥工业大学 教授 

学位论文独创性声明

本人郑重声明:所呈交的学位论文是本人在导师指导下进行独立研究工作所取得的成果。据我所知,除了文中特别加以标注和致谢的内容外,论文中不包含其他人已经发表或撰写过的研究成果,也不包含为获得合肥工业大学或其他教育机构的学位或证书而使用过的材料。对本文成果做出贡献的个人和集体,本人已在论文中作了明确的说明,并表示谢意。

学位论文中表达的观点纯属作者本人观点,与合肥工业大学无关。

学位论文作者签名:吴浩然

签名日期:2022年5月29日

学位论文版权使用授权书

本学位论文作者完全了解合肥工业大学有关保留、使用学位论文的规定,即:除保密期内的涉密学位论文外,学校有权保存并向国家有关部门或机构送交论文的复印件和电子光盘,允许论文被查阅或借阅。本人授权合肥工业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,允许采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密的学位论文在解密后适用本授权书)

学位论文作者签名:吴浩然

指导教师签名:雷

签名日期:2022年5月29日

签名日期:2022年5月29日

论文作者毕业去向

工作单位:

联系电话:

通讯地址:

E-mail:

邮政编码:

致 谢

时光荏苒，岁月如梭，在合肥工业大学的三年的时间如同白驹过隙，却给我留下了深深的影响。自从 2020 年新冠疫情爆发以来，我的学习生活受到了较大的影响，但在老师、朋友和同学的帮助下，我度过了充实的研究生生活。在这毕业之际，我要感谢所有在这里给予我帮助和关心的老师、朋友和同学！

首先要感谢我的导师霍星教授，在这三年的时间里，她给了我深刻的教诲与莫大的鼓舞。从小论文的选题到最终的定稿，从内容到格式，老师都不厌其烦的给了我指导。在老师的辛勤栽培与孜孜不倦的教诲下，我收获巨大，完成了小论文与毕业论文。

感谢邵堃老师对于我小论文的帮助，邵老师多次不厌其烦地给我的小论文提出了宝贵意见，在他的指导下，我学到了许多写作技巧。

感谢崔光鹏，张飞，杨鹏，谢瑞，王浩，邹韵，陈影，张坤，邓银平，李昊等课题组的兄弟姐妹们在生活和学习上给我的帮助。他们勤奋苦学，踏实敬业的精神不断鼓舞着我努力学习奋勇前进。

感谢数学学院全体教师在我研究生期间所给予的无私帮助。

最后感谢我的父母，是他们在我陷入低谷时给了我无限的关心，他们教会了我要热爱生活直面挫折，我的每一点进步都离不开父母在背后的支持。

摘 要

由于标准粒子群优化算法容易陷入局部最优,因此其收敛精度较低。针对这个问题,提出了两种通过改进粒子学习方式在低维条件下避免算法陷入局部最优的改进粒子群优化算法。

在这两种改进粒子群优化算法的迭代过程中,每个粒子不再固定的向自身历史最优位置与全局历史最优位置进行学习,而是根据代码中随机生成的粒子矩阵的行顺序以及两种不同的选择规则,使粒子仅向自身历史最优位置或位于其上方的某个粒子的历史最优位置学习,在尽量避开对全局历史最优位置的学习的同时增加了对其它粒子信息的利用。这种学习方式减少了全局历史最优位置的吸引,增加了粒子更新位置时信息来源的多样性,从而避免算法陷入局部最优。

本文在 5 个测试函数上对两种改进粒子群优化算法进行了测试,并将测试结果与标准粒子群优化算法,一个粒子群优化算法变体:量子粒子群优化算法以及其它两种元启发式算法:遗传算法,鲸鱼优化算法进行了比较。

最后,将两个改进粒子群优化算法与二维 Tsallis 熵结合对数据集 BSDS500 中的 100 张图像进行分割,并将分割结果与元启发式算法:粒子群优化算法,量子粒子群优化算法,遗传算法,鲸鱼优化算法以及神经网络 Unet 的分割结果进行对比展示,然后将这两种改进粒子群优化算法与这些元启发式算法的分割结果进行量化对比。

两次实验的结果表明本文所提出的改进的粒子群优化算法,通过减少粒子对全局最优值的学习以及增加对其它粒子信息的应用,可以在低维环境下避免算法陷入局部最优。

关键词: 粒子群优化算法; Tsallis 熵; 图像分割; 局部搜索; 改进粒子群优化算法

ABSTRACT

Because the standard particle swarm optimization algorithm is easy to fall into local optimization, its convergence accuracy is low. To solve this problem, two improved particle swarm optimization algorithms are proposed to avoid falling into local optimization under low dimensional conditions by improving the learning method of particles.

In the iterative process of these two improved particle swarm optimization algorithms, each particle no longer learns from its own historical optimal position and global historical optimal position, but makes the particle learn only from its own historical optimal position or the historical optimal position of a particle above it according to the row order of the randomly generated particle matrix in the code and two different selection rules. While avoiding the learning of the global historical optimal position as much as possible, it increases the utilization of other particle information. This learning method reduces the attraction of the global historical optimal position and increases the diversity of information sources when updating the position of particles, so as to avoid the algorithm falling into local optimization.

In this thesis, two improved particle swarm optimization algorithms are tested on five test functions, and the test results are compared with the Standard Particle Swarm Optimization Algorithm, a variant of particle swarm optimization algorithm: Quantum Inspired Particle Swarm Optimization Algorithm and two other meta heuristic algorithms: Genetic Algorithm and The Whale Optimization Algorithm.

Finally, two improved particle swarm optimization algorithms and two-dimensional Tsallis entropy are combined to segment 100 images in the dataset BSDS500, and the segmentation results are compared with the segmentation results of meta heuristic algorithms: Standard Particle Swarm Optimization Algorithm, Quantum Inspired Particle Swarm Optimization Algorithm, Genetic Algorithm, The Whale Optimization Algorithm and neural network Unet, and then the segmentation results of the two improved particle swarm optimization algorithms and these meta heuristic algorithms are quantitatively compared.

The results of two experiments show that the improved particle swarm optimization algorithm proposed in this thesis can avoid the algorithm falling into local optimization in low-dimensional environment by reducing the learning of particles to the global optimal position and increasing the application of other particle information.

KEYWORDS: Particle swarm optimization; Tsallis entropy; Image segmentation; Local search; Improved particle swarm optimization algorithm

目 录

第一章 绪论.....	1
1.1 元启发式算法.....	1
1.2 粒子群优化算法的研究现状.....	1
1.3 基于熵阈值的图像分割法.....	3
1.4 本文主要工作及结构安排.....	4
第二章 粒子群优化算法及相关实验准备.....	6
2.1 粒子群优化算法.....	6
2.2 二维 Tsallis 熵阈值分割.....	7
2.3 测试函数.....	8
2.4 本章小结.....	9
第三章 改进粒子群优化算法.....	10
3.1 引言.....	10
3.2 改进粒子群优化算法原理.....	10
3.3 确定参数 m 的实验.....	12
3.4 对比实验结果与分析.....	13
第四章 综合学习改进粒子群优化算法.....	16
4.1 引言.....	16
4.2 综合学习策略.....	16
4.3 综合学习改进粒子群优化算法原理.....	16
4.4 确定学习概率的实验.....	18
4.5 对比实验结果与分析.....	19
4.6 两种改进粒子群优化算法的收敛速度分析.....	21
第五章 基于改进粒子群优化算法的二维 Tsallis 熵阈值分割.....	22
5.1 引言.....	22
5.2 改进粒子群优化算法结合二维 Tsallis 熵具体步骤.....	22
5.3 评价指标.....	23
5.4 实验结果与分析.....	24
第六章 结论.....	27
参考文献.....	29

插图清单

图 2.1 二维直方图.....	7
图 4.1 IPSO, CLIPSO 收敛速度分析.....	21
图 5.1 算法流程图.....	23
图 5.2 各算法在样图(a)上的分割结果.....	25
图 5.3 各算法在样图(b)上的分割结果.....	25
图 5.4 各算法在样图(c)上的分割结果.....	25

表格清单

表 3.1 参数 m 实验.....	12
表 3.2 各算法参数设置.....	14
表 3.3 各算法比较.....	14
表 4.1 参数 P_c 实验.....	18
表 4.2 各算法比较.....	19
表 5.1 各元启发式算法分割结果评价.....	26

第一章 绪论

1.1 元启发式算法

启发式算法(heuristic algorithm)是一种基于人的直观经验或自然规律所构造的一种算法,它能够以较小的代价给出目标组合优化问题的一个可行解,但该可行解与目标问题的最优解之间的误差一般无法预计^[1]。元启发式算法(meta-heuristic algorithm)是启发式算法的改进,其是用来构造启发式算法的一般框架。相比于启发式算法,元启发式算法不具备被求解问题的特征,从而较少依赖算法本身组织结构,因此元启发式算法泛用性更好。

元启发式算法通常可以分为三类:第一类是基于进化的元启发式算法,该类元启发式算法是受自然进化理论启发的,如遗传算法(Genetic Algorithm, GA)^[2]。遗传算法模拟了生物种群的进化机制,在迭代过程中根据不同搜索代理的表现,“选择”表现较好的一部分搜索代理不进行任何更新直接遗传到下一代,然后再将搜索代理两两配对进行“交叉”,形成新的搜索代理,最后对部分搜索代理依照一定的概率进行“变异”。在求解复杂的问题时,遗传算法能以较快的速度取得较好的结果,因此其得到了广泛的应用。第二类是受生物种群的社会行为启发的元启发式算法,该类元启发式算法主要受动物群体的社会行为启发,如粒子群优化算法(Particle Swarm Optimization, PSO)^[3],鲸鱼优化算法(The Whale Optimization Algorithm, WOA)^[4]和天牛须搜索算法(Beetle Antennae Search Algorithm, BAS)^[5]等。其中鲸鱼优化算法是模拟鲸鱼捕食行为提出的算法,在 WOA 中,鲸鱼的捕食行为被抽象为两个步骤:收缩包围机制与螺旋吐泡机制,在算法迭代过程中,根据不同情况鲸鱼会以不同的方式进行移动。第三类则是基于物理现象的元启发式算法,该类元启发式算法主要是模仿宇宙的物理规律,如黑洞算法^[6],基于星系的搜索算法^[7]等^[8]。而在这么多的元启发式算法中粒子群优化算法又因需要操作的参数少,寻优效率高而得到许多研究者的青睐。

1.2 粒子群优化算法研究现状

尽管粒子群优化算法在过去的几十年里发展十分迅速,并成功地应用于许多领域,但当优化问题存在大量局部最优时,其仍旧存在着早熟收敛的问题。也就是说在迭代后期,有很大可能大部分粒子都会聚集在某个局部最优的小范围区域内,从而导致算法最后收敛不到全局最优解。为了提高经典粒子群算法的搜索性能,人们提出了大量的粒子群算法变体,这些变体大致可分为以下四类^[9]:

(1)基于修改参数的方法:如 PSO 中的惯性权重,在文[10]中作为常数引入,但 Shi 和 Eberhart 则提出将惯性权重作为自适应参数引入^[11]。与常数相比,根

据迭代次数进行自适应变化的惯性权重可以调节算法在不同阶段的速度,使算法从全局搜索逐渐转为局部搜索。除此之外还有 PSO 中的加速度系数,一般直接取一个常数,但 Cheng 等人则提出了加速度系数的自适应更新策略^[12]。通过引入加速度系数的自适应策略,可以改变粒子在不同阶段向不同信息来源获取信息的程度。

(2)基于不同邻域拓扑的方法:为了增强种群的多样性,人们提出了多种邻域拓扑结构的 PSO 变体,例如环拓扑和冯诺依曼拓扑。Rui 提出了一种完全知情的粒子群优化算法(The Fully Informed Particle Swarm, FIPS)^[13],该方法中每个粒子是根据多个相邻粒子的历史最优位置来更新自身位置,而不是根据粒子自身历史最优位置或全局历史最优位置。Liang 等人提出了动态多群粒子群优化算法(Dynamic Multi Swarm Particle Swarm Optimizer, DMSPSO)^[14]。在动态多群粒子群优化算法中,粒子群会被分为几个小的种群,然后这些小种群会各自互不干扰的进行搜索。在迭代一定次数后,算法会将各个小种群随机进行重组以将各个小种群获得的信息进行交流,然后再以新的小种群配置进行迭代。动态多群粒子群优化算法通过将大种群分为数个小种群,保持了粒子群在迭代过程中的多样性。

(3)基于不同学习策略的方法:为了更好地收敛,人们提出了许多不同学习策略的 PSO 变体。如基于适应度-距离比的粒子群优化算法(Fitness Distance Ratio based Particle Swarm Optimizer, FDRPSO)^[15]。在 FDRPSO 中,粒子获取信息的对象除了自身历史最优位置与全局历史最优位置外,还多了一个邻域内最优粒子的历史最优位置。而粒子邻域内最优粒子的选择取决于两个粒子的适应度与距离之比。FDRPSO 通过增加粒子的学习对象,使粒子在更新时可以获取更多的信息。除此之外还有综合学习粒子群优化算法(Comprehensive Learning Particle Swarm Optimizer, CLPSO)^[16],该方法中每个粒子的学习对象并不是某个固定的粒子,而是不同粒子的组合。在综合学习粒子群优化算法中,粒子的学习对象的每个维度的值要么来自于随机选择的其它粒子的历史最优位置对应维度的值,要么取粒子自身历史最优位置对应维度的值,而两者的取舍规则取决于一个叫学习概率的随机数。

(4)混合方法:除了对 PSO 自身的改进,另一个活跃的研究方向是将 PSO 算法与其他搜索技术相结合。每一种搜索方法都有各自的优点,因此将 PSO 算法与其他进化算法相结合,取长补短,可以提高 PSO 的性能。例如 Juang 等将遗传算法与 PSO 结合提出了一种新的混合算法(Hybrid of Genetic Algorithm and Particle Swarm Algorithm, HGAPSO)^[17]。在 HGAPSO 中,粒子群在更新位置之前会先按照适应值进行排序,接下来与遗传算法不同的是排序后的前一半的粒子会按照粒子

群优化算法进行迭代,然后迭代后的前一半粒子会作为交叉与变异的对象以生成剩下的一半粒子,至此整个种群都完成了更新。HGAPSO 通过将遗传算法与粒子群优化算法结合,更加精确的模拟了自然界生物种群的进化模式,得到了更好的结果。除此之外,Shen 等人则将天牛须搜索算法与粒子群优化算法结合提出了自适应变异的天牛群优化算法(Beetle Swarm Algorithm with Adaptive Mutation, BSOAM)^[18]。天牛须搜索算法是单粒子优化算法,在 BSOAM 中,每个粒子都会被视为一个天牛,然后按照新的天牛须搜索算法的公式进行迭代。除此之外,每次迭代之后,算法会计算变异概率,再根据结果决定是否对目前的全局最优位置进行变异扰动。BSOAM 在粒子群的迭代方式中插入了天牛的迭代法则,在高维环境下取得了较好的结果。

1.3 基于熵阈值的图像分割法

图像分割就是把图像分成若干个特定的、具有独特性质的区域并提出人们感兴趣目标的技术和过程。图像分割在实际生活中的应用十分广泛,如在文字识别领域中,图像分割可以将文字与背景分离,对提取古文献的内容有很大的帮助。在遥感领域中,图像分割可以将遥感图像中的道路森林等目标提取出来,除此以外图像分割还在工业,医学等领域有着十分重要的作用。

现有的图像分割方法主要分以下几类:基于阈值的分割方法、基于区域的分割方法以及基于边缘的分割方法。其中基于阈值的分割方法是最常见的方法,该方法根据图像中目标与背景的灰度值的差异,通过设置一个或几个阈值来将图像分割成多个区域^[19]。目前比较成功的阈值方法有聚类方差法,熵方法等。在这之中熵阈值分割方法又因为实现简单且性能稳定而得到了广泛的应用。20 世纪 80 年代,学者们就开始将熵的概念应用在阈值选择上。Pun 首先将熵的概念引入图像的阈值分割,提出了最大后验熵法^[20],后续又有学者提出了一维最大 Shannon 熵阈值法^[21],除了最大熵阈值法外,也有学者提出了最小交叉熵阈值法^[22]。

上述基于熵的阈值分割法中一般采用的都是一维 Shannon 熵,但 Shannon 熵具有广延性。通常来讲,就是在图像分割领域中图像总的 Shannon 熵等于背景的 Shannon 熵与目标的 Shannon 熵之和。但通常情况下图像具有非广延性,使用 Shannon 会忽略目标和背景之间的相关性^[19]。为了解决这个问题,Tsallis 在 Shannon 熵的基础上提出了 Tsallis 熵^[23],相比于 Shannon 熵,Tsallis 熵具有非广延性,使用 Tsallis 熵进行图像分割可以取得更好的结果。除此之外由于一维熵阈值分割法是利用图像的一维直方图进行计算的,这在得到最佳分割阈值的过程中没有使用到像素点的邻域信息。因此图像分割结果容易受到噪声的干扰,而利用像素点的邻域信息可以在一定程度上改善分割结果^[24],有学者据此提出了多种基于图像二维直方图的熵阈值分割方法,如二维 Renyi 熵阈值法^[25],二维 Tsallis 熵阈值法^[26]

等。

1.4 本文主要工作及结构安排

由上文所述可知，当目标问题存在大量局部最优时，粒子群优化算法有较大概率无法收敛到全局最优，为了避免这种情况，本文提出了两种在低维环境下可以避免局部最优的改进粒子群优化算法：

由于在标准粒子群优化算法中，每个粒子在更新位置时都只会从当前自身历史最优位置以及全体历史最优位置获取信息，粒子在更新位置时没有直接使用其他粒子的位置信息，因此粒子在更新位置时对其它粒子的信息使用不足^[27]。除此之外，由于全体历史最优位置的存在，每个粒子在更新自身位置时都会向全局历史最优位置学习，因此一旦粒子群中的某个粒子陷入了局部最优，其余粒子也大概率会随着陷入局部最优。本文提出的方法对这两方面进行了改进，首先针对粒子对其余粒子信息使用不足的缺点，增加了粒子在更新位置时对其余粒子的历史最优位置信息的利用；其次针对全局历史最优位置容易导致粒子群陷入局部最优的问题，本文提出的方法极大减少了粒子向全局历史最优位置获取信息的频率。通过改进以上两个方面的问题，本文提出的改进粒子群优化算法在低维的环境下可以有效的避免粒子陷入局部最优。

由于图像分割的重要意义，因此精确且快速的分割图像十分重要。与 Shannon 熵相比，应用 Tsallis 熵的熵阈值分割法更进一步的考虑了图像目标与背景之间的相互关系，因此相对于 Shannon 熵使用 Tsallis 熵可以更精确的分割图像。除此之外与一维熵阈值分割法相比，二维熵阈值分割法利用了图像邻域的相关信息，其可以有效地避免噪声的干扰，从而实现更高精度的图像分割。虽然利用二维熵阈值法进行图像分割可以得到更好的结果，但同时分割步骤也变得更加繁琐，计算量也变得更大，从而运算时间也变得更长^[28]。因此为了提高分割效率，有学者提出了利用元启发式算法结合二维熵来进行阈值选择。本文则将提出的改进粒子群优化算法与二维 Tsallis 熵结合进行图像分割，取得了较好的效果。

本文的结构安排如下：

第一章是绪论，简单介绍了元启发式算法的分类；主要介绍了粒子群优化算法的研究现状；简单介绍了图像分割的研究意义以及熵阈值分割算法；最后对本文的内容以及结构进行了简要介绍。

第二章详细介绍了标准粒子群优化算法的定义及公式；详细介绍了二维 Tsallis 熵阈值分割法的步骤与公式；详细介绍了后文实验所需的测试函数。

第三章首先详细介绍了本文提出的改进粒子群优化算法的具体步骤与公式，然后将改进粒子群优化算法在测试函数上进行实验以确定算法中的最优参数值，最后将改进粒子群优化算法与其它部分元启发式算法在测试函数上进行对比实验。

第四章首先详细介绍了本文提出的另一种改进粒子群优化算法的具体步骤与公式，然后将该算法在测试函数上进行实验以确定算法中的最优参数值，最后将该算法与其它部分元启发式算法在测试函数上进行对比实验。

第五章首先介绍了本文提出的改进粒子群优化算法与二维 Tsallis 熵结合进行图像分割的具体步骤，然后介绍了评价图像分割结果的三种评价指标，最后将本文提出的算法结合二维 Tsallis 熵进行图像分割，并将分割结果与其它部分元启发式算法结合二维 Tsallis 熵的分割结果进行量化对比。

最后是结论部分，该部分总结了论文的工作，并对本文提出的改进粒子群优化算法的未来改进方向进行了展望。

第二章 粒子群优化算法及相关实验准备

2.1 粒子群优化算法

粒子群优化算法是由 Kennedy 和 Eberhart 于 1995 年提出的一种元启发式算法，是一种受生物种群的社会行为启发的一种元启发式算法。粒子群优化算法首先会根据目标问题随机生成遍布解空间的若干粒子，然后粒子之间通过相互协同合作在解空间中进行移动并在移动过程中寻找最优解。粒子群优化算法简单，实用，收敛效率高，因此被广泛应用于各个方面，比如图像分割，神经网络的训练等。在粒子群优化算法中，每个粒子都代表目标问题解空间中一个潜在的候选解。假设目标问题解空间的最大维数为 D ，粒子群所包含粒子的个数为 N 。则在算法迭代过程中第 t 代时第 i 个粒子的位置可以表示为 $X_i(t) = \{x_i^1, x_i^2, \dots, x_i^D\}$ ，其中 $i = \{1, 2, \dots, N\}$ ，同时粒子的速度可以表示为 $V_i(t) = \{v_i^1, v_i^2, \dots, v_i^D\}$ 。在粒子搜索全局最优解过程中，算法会记录每个粒子在搜索过程中发现的最佳位置即个体历史最优位置 $pbest_i$ 。除此之外算法也会记录整个粒子群所发现的最佳位置即全局历史最优位置 $gbest$ 。然后每个粒子在更新时都会根据自身的历史最优位置 $pbest_i$ 和全局历史最优位置 $gbest$ 来调整自身位置。经典粒子群优化算法的具体迭代公式如下：

$$V_i^d(t+1) = V_i^d(t) + c_1 R_1(t) (pbest_i^d(t) - X_i^d(t)) + c_2 R_2(t) (gbest^d(t) - X_i^d(t)) \quad (1)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (2)$$

式(1)中右边的第一部分称为惯性分量，它有助于确保粒子的收敛轨迹并防止其在搜索方向上出现过度的振荡。式(1)右边第二部分称为认知成分，表示粒子当前位置与其自身迄今为止最佳位置之间的距离，它表示了粒子返回其历史最优位置的趋势。式(1)右边第三部分称为社会成分，反映了粒子当前位置和全局最优位置之间的距离，它表示了粒子跟随整个群体历史最优位置的趋势^[29]。上式中， d 为粒子的某一个维度。 t 为当前迭代次数。 c_1 和 c_2 为加速度系数，它们可以决定粒子的自我认知和社会认知的影响，并反映不同群体之间的信息交换。较大的 c_1 将导致粒子被吸引在在局部范围内难以对解空间进行探索，而较大的 c_2 将导致粒子过多被全局历史最优位置吸引，从而导致粒子容易陷入局部最优^[30]，二者通常情况下都取 2。 $R_1(t)$ 与 $R_2(t)$ 为 $[0,1]$ 之间的随机数。

由上文可知，粒子群优化算法存在早熟收敛等问题，而这在经典粒子群优化算法上则更为明显。因此 1998 年，shi^[10] 等对经典粒子群优化算法进行了修改，提出了目前广泛应用的标准粒子群优化算法：

$$V_i^d(t+1) = \omega V_i^d(t) + c_1 R_1(t) (pbest_i^d(t) - X_i^d(t)) + c_2 R_2(t) (gbest^d(t) - X_i^d(t)) \quad (3)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (4)$$

与经典粒子群优化算法相比,标准粒子群优化算法增加了一个惯性权重 ω 。而文[11]则将惯性权重改为随迭代次数改变的自适应参数。自适应的惯性权重可以用来调节粒子在不同阶段速度,从而达到平衡算法的全局寻优能力与局部寻优能力的目的^[31]。当 ω 较大时,式(3)左边也会相对变得更大,粒子迭代时的步长也就更大,此时粒子可以快速地遍历解空间全局,因此这时的粒子全局寻优能力较强。当 ω 较小时,式(3)左边也会相对变得更小,粒子迭代时的步长也就更小,此时粒子可以较为详细的地遍历解空间的局部区域,因此这时的粒子局部寻优能力较强。但当粒子的速度过大时,粒子每次更新时的步长也更大,在搜索过程中可能会直接略过最优解。当粒子速度过低时,粒子的更新步长也会更小,粒子可能会在陷入局部最优时难以跳出局部最优^[32]。因此在设置中,惯性权重 ω 会随着迭代的进行而逐渐减小,这样粒子在迭代前期便具有较大的更新速度,粒子群可以更好的开发整个解空间。而在迭代后期,随着惯性权重的减小,粒子的更新速度也逐渐减小,这时每个粒子的寻优范围也会逐渐减小,所有粒子都只会“细致”的探索自身所处的邻域。 ω 的迭代公式如下所示:

$$\omega = \omega_{\max} + \frac{(\omega_{\min} - \omega_{\max})(t-1)}{\text{Max_iter} - 1} \quad (5)$$

上式中 Max_iter 为最大迭代次数, $\omega_{\min} = 0.4$, $\omega_{\max} = 0.9$ 。

2.2 二维 Tsallis 熵阈值分割

设 f 为大小 $M \times N$, 灰度级为 L 的图像, 用 $f(x,y)$ 表示图像 (x,y) 处的像素点的灰度值, $g(x,y)$ 表示以 (x,y) 为中心的 3×3 的像素邻域的平均灰度值, 则基于 $f(x,y)$ 与 $g(x,y)$ 可构造出二维直方图, 如图 2.1 所示:

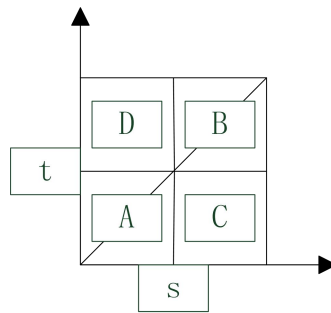


图 2.1 二维直方图

Fig 2.1 Two-dimensional histogram

在图 2.1 中, (s,t) 为假设的阈值向量, 可以看出, 向量 (s,t) 将上图分割为四个部分。因为目标与背景内像素点的灰度值都比较相近, 因此位于目标与背景区域的

像素点的灰度值与平均灰度值会比较相近。所以在图 2.1 中位于对角线周围的区域 A 和 B 表示目标或背景，C 和 D 则表示边界点或噪声。虽然在图 2.1 中四个区域看起来一样大，但在实际情况中图像中的边界点和噪声点相对于目标点和背景点来说数量非常少，因此可以忽略不计。则由上图可以得到目标和背景的两类概率分别为：

$$A: \left\{ \frac{p_{ij}}{P_0}; i=1,2,\dots,s; j=1,2,\dots,t \right\} \quad (6)$$

$$B: \left\{ \frac{p_{ij}}{1-P_0}; i=s+1,\dots,L; j=t+1,\dots,L \right\} \quad (7)$$

其中 p_{ij} 为联合概率：

$$p_{ij} = \frac{r_{ij}}{M \times N} \quad (8)$$

$$P_0 = \sum_{i=1}^s \sum_{j=1}^t p_{ij} \quad (9)$$

上式中 r_{ij} 表示 $f(x,y)=i$ 以及 $g(x,y)=j$ 的灰度值对 (i,j) 出现的次数， M, N 为图像的大小。由上述公式可得图像目标和背景的二维 Tsallis 熵分别为：

$$S_q^A(s,t) = \frac{1}{q-1} \left[1 - \sum_{i=1}^s \sum_{j=1}^t \left(\frac{p_{ij}}{P_0} \right)^q \right] \quad (10)$$

$$S_q^B(s,t) = \frac{1}{q-1} \left[1 - \sum_{i=s+1}^L \sum_{j=t+1}^L \left(\frac{p_{ij}}{1-P_0} \right)^q \right] \quad (11)$$

上式中 q 为待定系数，其代表了 Tsallis 熵的非广延性，本文中 q 取 0.8。则总的二维 Tsallis 熵为：

$$S_q(s,t) = S_q^A(s,t) + S_q^B(s,t) + (1-q)S_q^A(s,t)S_q^B(s,t) \quad (12)$$

最后只需将上式作为目标函数求出上式的最大解，就能得到最优分割阈值 (s^*, t^*) 。以上公式为文[26]和文[33]的工作。

2.3 测试函数

为了验证改进粒子群优化算法的寻优能力以及跳出局部最优的能力，使用了 5 个经典的测试函数来验证两种改进粒子群优化算法的性能。这 5 个测试函数中， f_1 为单峰函数，其余为多峰函数。具体公式如下所示：

Sum of Different Powers Functions:

$$f_1(x) = \sum_{i=1}^d |x_i|^{i+1} \quad (13)$$

在本文的实验中该函数解空间维度设置为 2，自变量范围为[-1,1]，最优值为 0。

Eggholder Function:

$$f_2(x) = -(x_2 + 47) \sin \left(\sqrt{\left| x_2 + \frac{x_1}{2} + 47 \right|} \right) - x_1 \sin \left(\sqrt{|x_1 - (x_2 + 47)|} \right) \quad (14)$$

该函数解空间维度固定为 2，自变量范围为[-512,512]，最优值为-959.6407。

Griewank Function:

$$f_3(x) = \sum_{i=1}^d \frac{x_i^2}{4000} - \prod_{i=1}^d \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1 \quad (15)$$

在本文的实验中该函数解空间维度被设置为 2，自变量范围为[-600,600]，最优值为 0。

Holder Table Function:

$$f_4(x) = - \left| \sin(x_1) \cos(x_2) \exp \left(\left| 1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \right| \right) \right| \quad (16)$$

该函数解空间的维度固定为 2，自变量范围为[-10,10]，最优值为-19.2085。

Langermann Function:

$$f_5(x) = \sum_{i=1}^m c_i \exp \left(-\frac{1}{\pi} \sum_{j=1}^d (x_j - A_{ij})^2 \right) \cos \left(\pi \sum_{j=1}^d (x_j - A_{ij})^2 \right) \quad (17)$$

该函数解空间的维度固定为 2，自变量范围为[0,10]，最优值为-5.1621259 且：

$$m = 5, c = (1, 2, 5, 2, 3), A = \begin{pmatrix} 3 & 5 \\ 5 & 2 \\ 2 & 1 \\ 1 & 4 \\ 7 & 9 \end{pmatrix}$$

2.4 本章小结

本章详细介绍了经典粒子群优化算法的公式以及算法中各参数的意义。然后介绍了标准粒子群优化算法的公式及其新增的参数：惯性权重的意义。之后又详细介绍了二维 Tsallis 熵阈值分割法的原理与公式。最后列出了后文实验所需的测试函数的公式及对应参数值。

第三章 改进粒子群优化算法

3.1 引言

由式(1)(2)可知,粒子在更新自身位置时的学习对象仅有自身的历史最优位置 $pbest_i$ 与所有粒子发现的全局历史最优位置 $gbest$,这使得更新位置的粒子对其他粒子的信息使用不足。且由于全局历史最优位置 $gbest$ 的存在,一旦粒子群中的某个粒子陷入了解空间中的某个局部最优点,该粒子的个体历史最优位置就会成为全局历史最优位置,从而其余所有粒子在更新位置时都会从局部最优位置获取信息,整个算法也会因此陷入局部最优。

为了解决这个问题,研究人员提出了大量粒子群优化算法的变体,如文[12]则通过修改加速度系数来避免粒子陷入局部最优,该方法提出了时变加速度系数,使加速度系数随着迭代次数的增加而改变,从而在不同阶段粒子从全局最优位置获取信息的趋势也不同。文[13]则通过修改粒子群的拓扑结构来使得粒子可以从自身邻域中的粒子获取信息,而不仅仅从全局最优位置获取信息,这拓宽了粒子获取信息的渠道,从而避免局部最优。文[16]则通过修改粒子的学习策略来避免算法陷入局部最优,该方法中粒子的每个维度都会向不同的粒子进行学习,而学习的对象则由随机选择的两个粒子竞争后得出。文[16]通过这种方式大大增加了粒子从其余粒子获取信息的渠道,因此可以有效地避免算法陷入局部最优。文[18]则将天牛须搜索算法与粒子群优化算法结合,通过将粒子群优化算法与其它元启发式算法结合取长补短,从而避免算法陷入局部最优。

本文则改进了粒子群优化算法中粒子的更新策略,通过增加粒子对其余粒子信息的利用以及减少粒子从全局最优位置获取信息,从而使算法可以在低维环境下可以有效地避免早熟收敛。

3.2 改进粒子群优化算法原理

在不失一般性的前提下,考虑以下最小化问题:

$$\min f = f(X) \quad (18)$$

当把粒子的位置 x_i^t 代入上式后便可得到粒子 i 在第 t 代时的适应值,而适应值则是用来评价粒子优劣的指标。

在本文提出的改进粒子群优化算法(Improved Particle Swarm Optimization, IPSO)中首先是减少粒子对全局最优位置的依赖,因此在具体的迭代过程中,每个粒子在迭代时不再次次都向全局最优位置学习。除此之外为了使粒子能在除全局历史最优位置与自身历史最优位置外的渠道获取信息,改进粒子群优化算法使粒子在更新时会根据一定的规则向其它粒子的历史最优值学习。

改进粒子群优化算法具体原理如下：

(1)首先，将粒子随机组为一个列向量，当粒子的维度大于一时，该列向量便可视作一个矩阵，其中每个粒子为一行，所有粒子的同一个维度则组成了该矩阵的一列。然后为了减少粒子对全局最优位置的依赖，以及拓宽粒子获取信息的渠道，每次迭代时，位于矩阵第一行的粒子会先按照标准粒子群优化算法的迭代法则更新自身位置，即该粒子的学习对象为自身历史最优位置 $pbest_i$ 与全局历史最优位置 $gbest$ 。

(2)然后剩下的粒子按照矩阵的行顺序从上往下依次更新位置，但这些粒子的学习对象与第一个粒子不同，它们在更新位置之前，先将自身当前的适应值与位于其上一行的粒子的当前适应值进行比较，若该粒子自身的适应值更优，则粒子只向自身的历史最优位置 $pbest_i$ 进行学习，若是上一个粒子的适应值更优，则该粒子就只向上一个粒子的历史最优位置 $pbest_{i-1}$ 进行学习，其中 $i = \{2, 3, \dots, N\}$ 。

(3)在上述 IPSO 的迭代步骤中，除了位于矩阵第一行的粒子会向全局历史最优位置 $gbest$ 外，其余粒子均不会向 $gbest$ 学习，虽然这会避免粒子被 $gbest$ 吸引而轻易地陷入局部最优，但由于粒子是随机分布的且在算法开始的时候粒子矩阵的行顺序也是随机确定的，因此粒子在迭代过程中可能会缺少全局信息，从而导致算法收敛速度变慢。因此，在迭代开始时，首先将所有粒子按标准粒子群优化算法的迭代规则迭代一次，然后每隔一定的迭代次数（设为 m 代），就对所有的粒子整体进行一次标准 PSO 迭代以获取全局最优信息。

IPSO 具体的迭代公式如下：

$$V_i^d(t+1) = \omega V_i^d(t) + c_1 R_1(t) (pbest_i^d(t) - X_i^d(t)) + c_2 R_2(t) (gbest^d(t) - X_i^d(t)), t=1 \quad (19)$$

$$\begin{cases} V_i^d(t+1) = \omega V_i^d(t) + c_1 R_1(t) (pbest_i^d(t) - X_i^d(t)), f(i) \leq f(i-1) \\ V_i^d(t+1) = \omega V_i^d(t) + c_1 R_1(t) (pbest_{i-1}^d(t) - X_i^d(t)), f(i) > f(i-1) \end{cases}, 2 \leq i \leq N \quad (20)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (21)$$

上式中， $f(i)$ 与 $f(i-1)$ 为粒子的适应值， N 为粒子数量，加速度系数 $c_1 = c_2 = 2$ ，惯性权重 ω 按(5)式计算，其中 $\omega_{\max} = 0.9$ ， $\omega_{\min} = 0.4$ ， t 为当前迭代次数。

IPSO 算法流程：

- 1 输入： $(D, N, \omega_{\max}, \omega_{\min}, c_1, c_2, v_{\max})$
- 2 输出： $(gbest)$ 。
- 3 初始化粒子群 $X^0 = \{X_1^0, \dots, X_N^0\}$ 与速度 $V^0 = \{V_1^0, \dots, V_N^0\}$ ；
- 4 计算 X^0 适应值，然后根据适应值得到 $pbest_i$ ， $gbest$ ；
- 5 根据式(3)(4)更新 X^1 ， V^1 ；

```

6    计算  $X^1$  适应值, 并更新  $pbest_i$  与  $gbest$ ;
7    令  $t=1$ ;
8    While (不满足停机条件)
9         $t=t+1$ ;
10       if mod( $t,m$ )==0
11           根据式(3)(4)更新  $X^t$ ,  $V^t$ ;
12           计算  $X^t$  适应值, 并更新  $pbest_i$  与  $gbest$ ;
13           continue;
14       end if
15       根据式(3)(4)更新  $X_1^t$ ,  $V_1^t$ ;
16       计算  $X_1^t$  适应值并更新  $pbest_1$  与  $gbest$ ;
17       根据式(19)(20)(21)更新  $X_i^t$ ,  $V_i^t$ , 同时更新  $pbest_i$  与  $gbest$  其中  $i=\{2,3,\dots,N\}$ ;
18   end While;
19   输出  $gbest$ 

```

3.3 确定参数 m 的实验

在这部分, 本文通过设置不同的 m 值, 让 IPSO 算法在 2.3 节介绍的 5 个测试函数上进行测试以确定参数 m 的最优值。除参数 m 外, 算法其余的参数设置与标准粒子群优化算法一样。粒子群数量设置为 40, 最大迭代次数设置为 7500, 算法在每个函数上测试 30 次, 然后对得到的 30 个结果取平均值。表格中加入了 NON 这一项, 表示所有粒子除了开始第一次时整体使用标准粒子群优化算法迭代一次, 之后一直按照粒子矩阵行顺序进行迭代而中间不再插入标准粒子群优化算法。实验环境为: Intel(R) Core(TM)i7-6700HQ CPU @2.60GHz。RAM8.00GB。Windows10 操作系统。MatlabR2016a。运行结果见表 3.1。

表 3.1 参数 m 实验

Table 3.1 Experiment of parameter m

函数	m	mean	min	max	std	time
f_1	NON	0.0000	0.0000	0.0000	0.0000	137.0433
	3	0.0000	0.0000	0.0000	0.0000	100.2875
	5	0.0000	0.0000	0.0000	0.0000	189.6072
	10	0.0000	0.0000	0.0000	0.0000	161.2814
	15	0.0000	0.0000	0.0000	0.0000	143.9742

	13	0.0000	0.0000	0.0000	0.0000	173.1174
f_2	NON	-889.3318	-959.6407	-716.6715	84.7606	132.6709
	3	-875.9687	-959.6407	-716.6679	74.9376	95.3313
	5	-903.1562	-959.6407	-716.6715	53.6167	191.6065
	10	-901.854	-959.6407	-753.0502	60.2684	165.7223
	15	-910.7552	-959.6407	-718.1675	51.7337	185.8682
	13	-913.1668	-959.6407	-716.6715	51.9377	161.8080
f_3	NON	0.0021	0.0000	0.0099	0.0035	185.7362
	3	0.0000	0.0000	0.0000	0.0000	132.7212
	5	0.0000	0.0000	0.0000	0.0000	234.0166
	10	0.0000	0.0000	0.0000	0.0000	202.4372
	15	0.0000	0.0000	0.0000	0.0000	241.1401
	13	0.0002	0.0000	0.0074	0.0014	198.6632
f_4	NON	-19.2085	-19.2085	-19.2085	7.136e-15	142.8881
	3	-19.0105	-19.2085	-18.0207	0.4502	102.6515
	5	-19.0897	-19.2085	-18.0207	0.3624	191.7907
	10	-19.1293	-19.2085	-18.0207	0.3014	160.2338
	15	-19.2085	-19.2085	-19.2085	6.9506e-15	168.5405
	13	-19.2085	-19.2085	-19.2085	7.0439e-15	152.1151
f_5	NON	-4.1365	-4.1558	-3.6776	0.0878	165.4448
	3	-4.1239	-4.1558	-3.6776	0.1213	103.9981
	5	-4.1558	-4.1558	-4.1558	9.0336e-16	194.7620
	10	-4.1399	-4.1558	-3.6776	0.0873	168.5055
	15	-4.1558	-4.1558	-4.1558	9.0336e-16	159.9480
	13	-4.1558	-4.1558	-4.1558	9.0336e-16	167.4198

从表 3.1 中可以看出在函数 f_1 上, 无论 m 取什么值, 算法均能找到函数的最优值; 在函数 f_2 上, 当 $m=13$ 时, 效果最好, 当 $m=15$ 时效果次之; 在函数 f_3 上, 仅当 m 取 13 以及 NON 时, 算法陷入了局部最优; 在函数 f_4 上, 当 m 取 3, 5, 10 时, 算法陷入了局部最优; 在函数 f_5 上, 当 m 取 NON, 3, 10 时, 算法陷入了局部最优。综合 IPSO 在五个测试函数上的实验结果, 在本文剩下的实验部分, 我们都取 $m=15$ 。

3.4 对比实验结果与分析

为了验证改进粒子群优化算法的有效性, 本文将 IPSO 与标准粒子群优化算法 (Particle Swarm Optimization, PSO) 以及量子粒子群优化算法 (Quantum-Inspired Particle Swarm Optimization, QPSO)^[34], 遗传算法 (Genetic Algorithm, GA), 鲸鱼优化算法 (The Whale Optimization Algorithm, WOA) 在五个测试函数上的测试结果进行比较。各算法的参数设置如表 3.2 所示。

表 3.2 各算法参数设置

Table 3.2 Parameter setting of each algorithm

算法	参数	参数值
IPSO	$c_1; c_2$	2;2
	$\omega_{\max}; \omega_{\min}; m$	0.9;0.4;15
PSO	$c_1; c_2$	2;2
	$\omega_{\max}; \omega_{\min}$	0.9;0.4
QPSO	$\alpha_{\max}; \alpha_{\min}$	1;0.5
	u	[0,1]
GA	交叉概率;	0.5;
	变异概率	0.1
WOA	\bar{a}	[2,0]
	\bar{r}	[0,1]
	$l; b$	[-1,1];1

所有测试函数的维度均设置为 2 维，所有算法均使用相同的种群大小 40 和最大迭代次数 7500。每个算法都在测试函数上运行 30 次，然后取平均值。实验环境与表 3.1 相同。实验结果如表 3.3 所示。

表 3.3 各算法比较

Table 3.3 Comparison between algorithms on test functions

Function	Algorithm	mean	min	max	std	time
f_1	IPSO	0.0000	0.0000	0.0000	0.0000	143.9742
	PSO	8.4485e-215	0.0000	2.5346e-213	0.0000	30.1029
	QPSO	0.0000	0.0000	0.0000	0.0000	22.9133
	GA	3.0114e-09	1.907e-11	1.4022e-08	4.4908e-09	47.0654
	WOA	0.0000	0.0000	0.0000	0.0000	34.9759
f_2	IPSO	-910.7552	-959.6407	-718.1675	51.7337	185.8682
	PSO	-834.0616	-959.6407	-582.3063	118.3031	33.3356
	QPSO	-955.3032	-959.6407	-894.5789	16.5067	22.5566
	GA	-720.0222	-888.4729	-458.8144	120.8527	74.1419
	WOA	-959.6407	-959.6407	-959.6407	5.8733e-13	35.5933
f_3	IPSO	0.0000	0.0000	0.0000	0.0000	241.1401
	PSO	0.0021	0.0000	0.0099	0.0035	47.3477
	QPSO	4.5862e-11	0.0000	1.3759e-09	2.512e-10	46.8559
	GA	1.6963e-09	4.6941e-12	1.3747e-08	2.5621e-09	108.8085

	WOA	0.0007	0.0000	0.0074	0.0023	58.1411
f_4	IPSO	-19.2085	-19.2085	-19.2085	6.9506e-15	168.5405
	PSO	-19.0105	-19.2085	-18.0207	0.4502	20.4662
	QPSO	-19.2085	-19.2085	-19.2085	2.2471e-07	22.8509
	GA	17.8123	-19.1834	-14.0085	1.5004	59.6464
	WOA	-19.2085	-19.2085	-19.2085	7.3533e-13	33.8082
f_5	IPSO	-4.1558	-4.1558	-4.1558	9.0336e-16	159.9480
	PSO	-3.8629	-4.1558	-2.1933	0.6764	24.3163
	QPSO	-4.0904	-4.1558	-2.1940	0.3582	26.4537
	GA	-3.8100	-4.0733	-3.6752	0.1152	62.1523
	WOA	-4.0971	-4.1558	-3.6776	0.1433	37.7265

在表 3.3 中, mean 表示算法运行 30 次所得结果的平均值。min 表示这 30 次结果中的最优值。max 表示这 30 次结果中的最差值。std 表示这 30 次结果的标准差。time 表示算法运行 30 次所花费的时间。由表 3.3 数据可以发现, IPSO 在这 5 个基准函数上的平均表现比较优异。在单峰函数 f_1 上, IPSO, QPSO, WOA 每次都可以发现函数的最优值, 同时 PSO 与 GA 则多次陷入了局部最优。在函数 f_2 上, IPSO 多次陷入了局部最优, 但在该函数上只有 WOA 算法每次都达到了全局最优。在函数 f_3 上, 只有 IPSO 每次都能准确找到函数的全局最优值, 其余算法均不同程度的陷入了局部最优。在函数 f_4 上, IPSO 与 QPSO, WOA 均能次次达到全局最优, PSO 与 GA 算法则多次陷入了局部最优。在函数 f_5 上, 所有算法均陷入了局部最优, 但 IPSO 所能达到的效果最接近全局最优值。由此可以得知, IPSO 可以在一定程度上避免局部最优。但从表中也可以看出, IPSO 的运行时间要长于其他算法, 这也是未来需要改进的方向。

第四章 综合学习改进粒子群优化算法

4.1 引言

由第三章可知,通过增加粒子对其余粒子信息的利用以及减少粒子从全局最优位置获取信息,改进粒子群优化算法在低维环境下可以有效地避免局部最优。但如 3.2 节详细介绍的算法流程可知,改进粒子群优化算法的粒子排列顺序是一开始就确定下来的,而在改进粒子群优化算法中粒子在更新时只会向自身历史最优位置与前一个粒子的历史最优位置学习,并间歇性的向全局历史最优位置学习。虽然这相对于标准粒子群优化算法来说增加了粒子更新时的信息获取渠道,但学习对象还是比较固定,信息来源的多样性较差。因此在这一章,在改进粒子群优化算法的基础上,再次引入了综合学习策略提出了综合学习改进粒子群优化算法(Comprehensive Learning Improved Particle Swarm Optimizer,CLIPSO)以获得更好的效果。

4.2 综合学习策略

综合学习策略是由文[16]提出的一种新的学习策略。在文[16]中,作者分析了标准粒子群优化算法的缺点,当粒子群的全局历史最优位置 $gbest$ 与粒子的个体历史最优位置 $pbest_i$ 位于粒子当前位置的两边时,可能会使粒子运动发生振荡。但相对于 $pbest_i$ 来说, $gbest$ 能提供更大的吸引力,因此即使 $gbest$ 位于远离全局最优位置的局部最优区域,其也会吸引着粒子向局部最优区域移动。如果 $gbest$ 与粒子的 $pbest_i$ 位于粒子当前位置的同一侧,则粒子将朝该方向移动,并且一旦其 $pbest_i$ 落入 $gbest$ 所在的相同局部最优区域,粒子就可能无法跳出局部最优区域。

因此文[16]的作者提出了一种综合学习策略以保证粒子从好的样本中学习并增加粒子群的多样性。在综合学习策略中,粒子在更新时,每个维度都会向不同的粒子学习。具体规则如下所述:在粒子更新时,粒子的每个维度都会对应生成一个随机数,然后将随机数与事先设置的常数学习概率 P_c 进行比较,若随机数较大,则粒子该维度将只从自身 $pbest$ 的对应维度进行学习。若学习概率 P_c 更大,粒子该维度则向其他粒子的 $pbest$ 对应的维度进行学习,而学习的对象则通过竞争选择而来。竞争选择的具体规则如下所述:首先随机选择两个粒子,然后比较这两个粒子的历史最优值(将两个粒子的 $pbest$ 带入评价函数求得的适应值),选择适应值较好的一个将其 $pbest$ 对应的维度当成粒子更新时对应维度的学习对象。

4.3 综合学习改进粒子群优化算法原理

受综合学习策略的启发,本文简化了 CLPSO 中的综合学习策略,并将其与前文提出的改进粒子群优化算法结合提出了综合学习改进粒子群优化算法。综合学

习改进粒子群优化算法总体步骤与改进粒子群优化算法相似，但在粒子进行链式迭代时引入了简化的综合学习策略，由此增加了粒子群的多样性并使粒子更新时获取信息的渠道范围大大增加。

综合学习改进粒子群优化算法的具体原理如下所示：

(1)首先，将粒子按行组为一个矩阵，位于矩阵第一行的粒子会先按照标准粒子群优化算法的迭代法则更新自身位置，即该粒子的学习对象为自身历史最优位置 $pbest_i$ 与全局历史最优位置 $gbest$ 。

(2)然后剩下的粒子按照矩阵的行顺序从上往下依次更新位置，但这些粒子的学习对象与改进粒子群优化算法不同，它们在更新位置之前会生成一个随机数，然后将该随机数与事先设定的学习概率 Pc 进行比较，若学习概率较大，则粒子只向自身的历史最优位置进行学习。反之若随机数较大，则在该粒子上方随机挑选两个粒子，再对这两个粒子的当前适应值进行对比，选择较优的粒子作为当前更新位置的粒子的学习对象。

(3)最后，因为该算法获取全局最优位置信息的方式与改进粒子群优化算法相同，所以在第一次迭代时，所有粒子都会按照标准粒子群优化算法的法则迭代一次，之后每隔一定的迭代次数（同设为 15 代），就对所有的粒子整体进行一次标准粒子群优化算法迭代以获取全局最优信息。

CLIPSO 具体迭代公式如下所示：

$$V_i^d(t+1) = \omega V_i^d(t) + c_1 R_1(t) (pbest_i^d(t) - X_i^d(t)) + c_2 R_2(t) (gbest^d(t) - X_i^d(t)) \quad t=1 \quad (22)$$

$$\begin{cases} V_i^d(t+1) = \omega V_i^d(t) + c_1 R_1(t) (pbest_i^d(t) - X_i^d(t)), rand_i \leq Pc \\ V_i^d(t+1) = \omega V_i^d(t) + c_1 R_1(t) (pbest_j^d(t) - X_i^d(t)), rand_i > Pc \end{cases}, 2 \leq i \leq N; 1 \leq j < i \quad (23)$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (24)$$

算法步骤如下：

CLIPSO 算法流程：

- 1 输入： $(D, N, \omega_{\max}, \omega_{\min}, c_1, c_2, v_{\max})$
- 2 输出： $(gbest)$.
- 3 初始化粒子群 $X^0 = \{X_1^0, \dots, X_N^0\}$ 与速度 $V^0 = \{V_1^0, \dots, V_N^0\}$ ；
- 4 计算 X^0 适应值，然后根据适应值得到 $pbest_i$ ， $gbest$ ；
- 5 根据式(3)(4)更新 X^1 ， V^1 ；
- 6 计算 X^1 适应值，并更新 $pbest_i$ 与 $gbest$ ；
- 7 令 $t=1$ ；
- 8 While（不满足停机条件）
- 9 | $t=t+1$ ；

10	if mod(t,m)==0
11	根据式(3)(4)更新 x^t , v^t ;
12	计算 x^t 适应值, 并更新 $pbest_i$ 与 $gbest$;
13	continue;
14	end if
15	根据式(3)(4)更新 x_1^t , v_1^t ;
16	计算 x_1^t 适应值并更新 $pbest_i$ 与 $gbest$;
17	根据式(22)(23)(24)更新 x_i^t , v_i^t , 同时更新 $pbest_i$ 与 $gbest$ 其中 $i=\{2,3,\cdots N\}$;
18	end While;
19	输出 $gbest$

4.4 确定学习概率的实验

在这部分, 本文通过设置不同的 Pc 值, 让 CLIPSO 算法在 2.3 节介绍的 5 个测试函数上进行测试以确定参数 Pc 的最优值。算法使用与标准粒子群优化算法一样的参数, 种群大小设置为 40, 最大迭代次数设置为 7500, 算法在每个函数上测试 30 次, 然后对取得的结果取平均值,实验环境与表 3.1 相同。运行结果见表 4.1。

表 4.1 参数 Pc 实验

Table 4.1 experiment of parameter Pc						
函数	Pc	mean	min	max	std	time
f_1	0.1	0.0000	0.0000	0.0000	0.0000	126.9032
	0.3	0.0000	0.0000	0.0000	0.0000	120.4235
	0.5	0.0000	0.0000	0.0000	0.0000	137.3035
	0.8	0.0000	0.0000	0.0000	0.0000	153.2298
	0.98	0.0000	0.0000	0.0000	0.0000	141.0404
	1	0.0000	0.0000	0.0000	0.0000	139.2825
f_2	0.1	-760.9716	-937.0407	-562.3048	106.4381	145.7724
	0.3	-826.0528	-934.0935	-676.8069	80.8615	142.2320
	0.5	-910.0918	-959.6407	-738.0371	59.8571	150.9975
	0.8	-929.7428	-959.6407	-888.9491	27.7938	302.0860
	0.98	-939.6031	-959.6407	-894.5789	24.9983	209.3457
	1	-913.6499	-959.6407	-752.0239	50.5258	284.5892
f_3	0.1	0.0000	0.0000	0.0000	0.0000	158.1899
	0.3	0.0000	0.0000	0.0000	0.0000	168.6532
	0.5	0.0000	0.0000	0.0000	0.0000	165.3337

	0.8	0.0000	0.0000	0.0000	0.0000	167.8978
	0.98	0.0002	0.0000	0.0074	0.0014	184.6312
	1	0.0030	0.0000	0.0074	0.0037	172.1381
f_4	0.1	-18.9314	-19.2085	-18.0207	0.5110	133.4562
	0.3	-19.2085	-19.2085	-19.2085	1.0113e-14	137.5323
	0.5	-19.2085	-19.2085	-19.2085	6.4908e-14	137.8413
	0.8	-19.2085	-19.2085	-19.2085	1.1157e-14	142.0838
	0.98	-19.2085	-19.2085	-19.2085	9.3299e-15	142.8586
	1	-19.2085	-19.2085	-19.2085	6.1535e-15	147.6276
f_5	0.1	-4.0585	-4.1558	-2.1940	0.3724	136.3331
	0.3	-4.1399	-4.1558	-3.6776	0.0873	143.1094
	0.5	-4.1558	-4.1558	-4.1558	9.0336e-16	146.0862
	0.8	-4.1558	-4.1558	-4.1558	9.0336e-16	154.5934
	0.98	-4.1558	-4.1558	-4.1558	9.0336e-16	150.5782
	1	-4.1558	-4.1558	-4.1558	9.0336e-16	151.4938

从表 4.1 中可以看出在函数 f_1 上, 无论 Pc 值为多少, 算法都能发现全局最优值。在函数 f_2 上, 当 Pc=0.98 时, 算法的表现最好, 当 Pc=0.8 时, 效果次之。在函数 f_3 上, 只有当 Pc 等于 0.98 和 1 时, 算法陷入了局部最优。在函数 f_4 上只有当 Pc 等于 0.1 时, 算法才陷入了局部最优。在函数 f_5 上, 当 Pc 等于 0.1 和 0.3 时, 算法陷入了局部最优。因此结合上表的数据, 本文后续的 CLIPSO 中, 学习概率 Pc 均取 0.8。

4.5 对比实验结果与分析

在这部分, 本文将综合改进粒子群优化算法与 3.4 节提到几种元启发式算法在五个测试函数上进行对比实验。所有算法的参数设置与表 3.2 相同, 实验使用与表 3.3 相同的测试函数, 参数设置及设备环境。每个算法都在测试函数上运行 30 次, 然后取平均值。实验结果如表 4.2 所示。

表 4.2 各算法比较

Table 4.2 Comparison between algorithms on test functions

Function	Algorithm	mean	min	max	std	time
f_1	IPSO	0.0000	0.0000	0.0000	0.0000	143.9742
	CLIPSO	0.0000	0.0000	0.0000	0.0000	153.2298
	PSO	8.4485e-215	0.0000	2.5346e-213	0.0000	30.1029
	QPSO	0.0000	0.0000	0.0000	0.0000	22.9133
	GA	3.0114e-09	1.907e-11	1.4022e-08	4.4908e-09	47.0654
	WOA	0.0000	0.0000	0.0000	0.0000	34.9759
f_2	IPSO	-910.7552	-959.6407	-718.1675	51.7337	185.8682

	CLIPSO	-929.7428	-959.6407	-888.9491	27.7938	302.0860
	PSO	-834.0616	-959.6407	-582.3063	118.3031	33.3356
	QPSO	-955.3032	-959.6407	-894.5789	16.5067	22.5566
	GA	-720.0222	-888.4729	-458.8144	120.8527	74.1419
	WOA	-959.6407	-959.6407	-959.6407	5.8733e-13	35.5933
f_3	IPSO	0.0000	0.0000	0.0000	0.0000	241.1401
	CLIPSO	0.0000	0.0000	0.0000	0.0000	167.8978
	PSO	0.0021	0.0000	0.0099	0.0035	47.3477
	QPSO	4.5862e-11	0.0000	1.3759e-09	2.512e-10	46.8559
	GA	1.6963e-09	4.6941e-12	1.3747e-08	2.5621e-09	108.8085
	WOA	0.0007	0.0000	0.0074	0.0023	58.1411
f_4	IPSO	-19.2085	-19.2085	-19.2085	6.9506e-15	168.5405
	CLIPSO	-19.2085	-19.2085	-19.2085	1.1157e-14	142.0838
	PSO	-19.0105	-19.2085	-18.0207	0.4502	20.4662
	QPSO	-19.2085	-19.2085	-19.2085	2.2471e-07	22.8509
	GA	17.8123	-19.1834	-14.0085	1.5004	59.6464
	WOA	-19.2085	-19.2085	-19.2085	7.3533e-13	33.8082
f_5	IPSO	-4.1558	-4.1558	-4.1558	9.0336e-16	159.9480
	CLIPSO	-4.1558	-4.1558	-4.1558	9.0336e-16	154.5934
	PSO	-3.8629	-4.1558	-2.1933	0.6764	24.3163
	QPSO	-4.0904	-4.1558	-2.1940	0.3582	26.4537
	GA	-3.8100	-4.0733	-3.6752	0.1152	62.1523
	WOA	-4.0971	-4.1558	-3.6776	0.1433	37.7265

上表中, mean 表示算法运行 30 次所得结果的平均值, min 表示这 30 次结果中的最优值, max 表示这 30 次结果中的最差值, std 表示这 30 次结果的标准差, time 表示算法运行的时间。由表 4.2 数据可以发现, CLIPSO 在这 5 个基准函数上的平均表现比较优异。在单峰函数 f_1 上, IPSO, CLIPSO, QPSO, WOA 每次都可以发现函数的最优值,同时 PSO 与 GA 则多次陷入了局部最优。在函数 f_2 上, IPSO 与 CLIPSO 多次陷入了局部最优,但在该函数上只有 WOA 算法每次都达到了全局最优且 CLIPSO 的平均表现要优于 IPSO。在函数 f_3 上,只有 IPSO 与 CLIPSO 每次都能收敛到全局最优值,其余算法均不同程度的陷入了局部最优。在函数 f_4 上, IPSO, CLIPSO 与 QPSO, WOA 均能次次达到全局最优, PSO 与 GA 算法则多次陷入了局部最优。在函数 f_5 上,所有算法均陷入了局部最优,但有 IPSO 与 CLIPSO

的平均表现均优于其它算法，且二者所能达到的效果最接近全局最优值。由此可以得知，CLIPSO 与 IPSO 一样也可以在一定程度上避免局部最优且在函数 f_2 上，CLIPSO 的表现要稍微优于 IPSO。

4.6 两种改进粒子群优化算法的收敛速度分析

在本节中，将 IPSO，CLIPSO 与上文提到的其余四种元启发式算法在 5 个测试函数上收敛速度进行比较。算法的参数，运行环境均与上文相同，每个算法在函数上只运行一次，每次设置种群数为 40，迭代次数为 7500。算法会记录每次迭代时的全局历史最优值，并在算法结束时生成折线图，以此直观地比较各个算法的收敛速度。具体结果如图 4.1 所示：

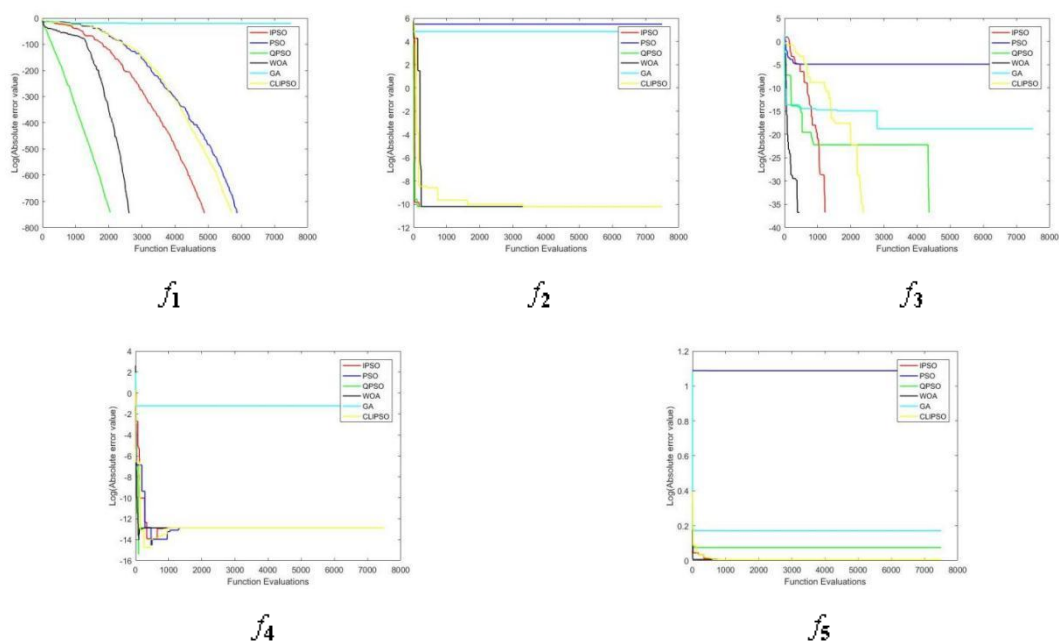


图 4.1 IPSO，CLIPSO 收敛速度分析

Fig 4.1 IPSO, CLIPSO convergence rate analysis

由图 4.1 可知，在函数 f_1 上 IPSO 与 CLIPSO 的收敛速度要慢于 WOA 与 QPSO，但快于 PSO 以及 GA。在函数 f_2 上，CLIPSO 与 IPSO 的收敛速度要快于 PSO，GA，WOA，但与 QPSO 的收敛速度不相上下。在函数 f_3 上，在迭代前期，CLIPSO 与 IPSO 较慢，但二者一直在不断地下降，到了迭代后期，二者的收敛速度仅次于 WOA，要快于 PSO，GA，QPSO。在函数 f_4 上，除 GA 陷入了局部最优外，其余算法的收敛速度不相上下。在函数 f_5 上，所有算法在前期的收敛速度都基本相同，但到了迭代后期，PSO，QPSO，GA 均陷入了局部最优，IPSO，CLIPSO 与 WOA 则在继续收敛。总体来说，IPSO 的收敛速度要快于 CLIPSO，二者在单峰函数上的收敛速度要慢于 QPSO 与 WOA 算法，在多峰函数上的收敛速度则基本位于所有算法的前列

第五章 基于改进粒子群优化算法的二维 Tsallis 熵阈值分割

5.1 引言

由上文所述,采用二维 Tsallis 熵进行图像分割可以较为精确的分割图像。但二维熵阈值分割法也会导致分割图像的计算量增大从而使分割时间变得更长。因此有学者提出了使用元启发式算法结合二维熵进行阈值选择。如文[35]则提出了一种新的基于 Tsallis 熵和 Renyi 熵的二维熵阈值分割方法并将其与遗传算法结合进行图像分割,文章结果表明,与经典的阈值分割方法相比获得了更好的图像分割质量。文[36]则提出了一种基于灰度和局部平均灰度直方图和 Tsallis-Havrda-Charvát 熵的 RGB 彩色图像多级阈值模型,并使用粒子群优化算法进行优化,获得了三个分割的分量图像,获得了较好的分割结果。

为了验证改进粒子群优化算法(Improved Particle Swarm Optimizer, IPSO)以及综合学习改进粒子群优化算法(Comprehensive Learning Improved Particle Swarm Optimizer, CLIPSO)的分割精度,本章将二者与二维 Tsallis 熵结合对数据集 BSD500 中的 100 张图片进行了分割,并将分割结果与标准粒子群优化算法(Particle Swarm Optimization, PSO),量子粒子群优化算法(Quantum-Inspired Particle Swarm Optimization, QPSO),遗传算法(Genetic Algorithm, GA),鲸鱼优化算法(The Whale Optimization Algorithm, WOA)以及神经网络 Unet^[37]的分割结果进行对比。

5.2 改进粒子群优化算法结合二维 Tsallis 熵具体步骤

将二维最大 Tsallis 熵的函数即式(12)作为改进粒子群优化算法的目标函数,每一对分割阈值(s, t)视为 IPSO 中的的一个粒子,最后根据 IPSO 的规则进行迭代并获得最优分割阈值。CLIPSO 结合二维 Tsallis 熵的分割过程与 IPSO 类似,因此本文只给出 IPSO 结合二维 Tsallis 熵进行图像分割的流程图。具体的流程如图 5.1 所示。

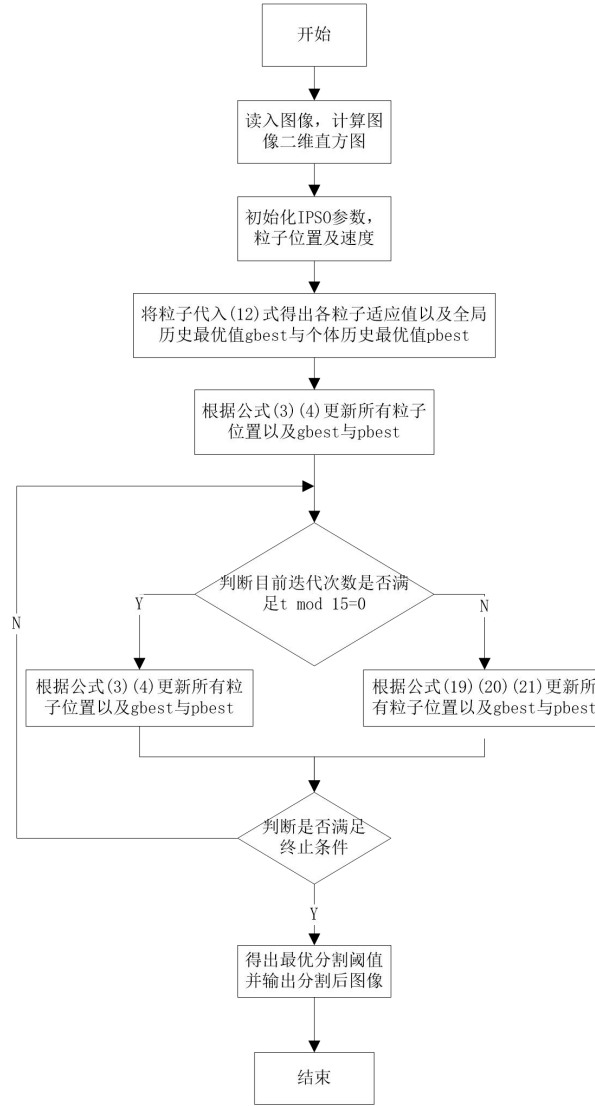


图 5.1 算法流程图

Fig 5.1 Algorithm flow chart

5.3 评价指标

为了对各元启发式算法分割后的图像进行量化对比, 引入了三种评价指标: 第一种指标为 F1-Measure^[38]简称 F 值, F1-Measure 是一种统计量, 常用于评价图像分割结果的好坏, F 值越高则说明实验方法越有效。其中 F 的公式如下所示:

$$F = \frac{2 * PR}{P + R} \quad (25)$$

上式中 P 表示精准率 (Precision), R 表示召回率 (Recall)。二者的公式如下:

$$P = \frac{TP}{TP + FP} \quad (26)$$

$$R = \frac{TP}{TP + FN} \quad (27)$$

第二种指标为 PA^[39], PA 即像素准确性 (pixel accuracy), 表示正确分类的像

素占总像素的百分比，PA 值越高，则说明分割结果越准确，其公式如下：

$$PA = \frac{TP + TN}{TP + TN + FP + FN} \quad (28)$$

引入的第三种指标为 IOU^[40]，IOU 即交并比，表示两个区域重叠的部分除以两个区域的集合，IOU 值越高，则说明分割结果越准确，具体公式如下：

$$IOU = \frac{TP}{TP + FP + FN} \quad (29)$$

上式中 TP 即 True Positive，表示将正类预测为正类的数目。在图像分割这种二元分类中即表示分割后的图像与 GT 图像（即 ground truth 图像，为人工分割标记的图像）相比，在图像的同一位置二者的灰度级都为 255 的像素点的个数。 TN 即 True Negative，表示将负类预测为负类的数目。在图像分割中即表示在图像的同一位置，GT 图像灰度为 0 而分割出来的图像灰度为 0，这些像素点的个数。 FP 即 False Positive，表示将负类预测为正类的误报数目。在图像分割中即表示在图像的同一位置，GT 图像的灰度为 0 而分割出来的图像灰度为 255，这些像素点的个数。 FN 即 False Negative，表示将正类预测为负类的数目。在图像分割中即表示在图像的同一位置，GT 图像灰度为 255 而分割出来的图像灰度为 0，这些像素点的个数。

5.4 实验结果与分析

各元启发式算法参数设置如表 3.2 所示，种群数设置为 20，最大迭代次数设置为 100。实验环境为：Intel(R) Core(TM)i7-6700HQ CPU @2.60GHz, RAM8.00GB, Windows10 操作系统，MatlabR2016a。神经网络 Unet 的实验环境为：NVIDIA GeForce RTX2070 SUPER，Intel(R) Core(TM)i7-5930K CPU @3.50GHz，RAM16.0GB, Windows10 操作系统，python3.6.5。神经网络训练时间为 3 天，直接在上面挑选的 100 张图片上进行训练，并取训练的结果进行展示。图 5.2，5.3，5.4 是从分割结果中挑出的对比较为明显的三张样图：

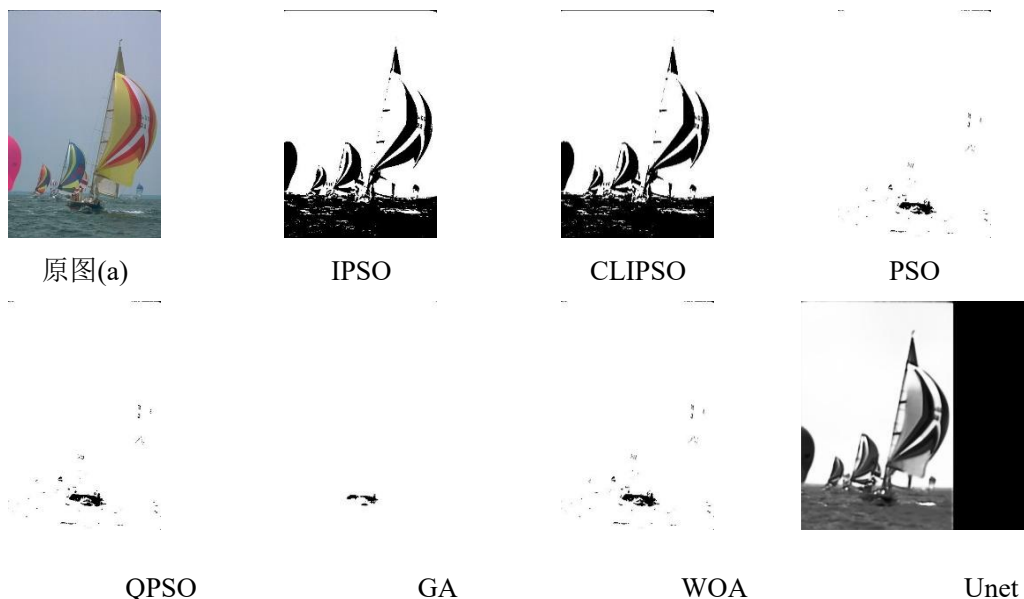


图 5.2 各算法在样图(a)上的分割结果

Fig 5.2 Segmentation results of each algorithm on sample image (a)

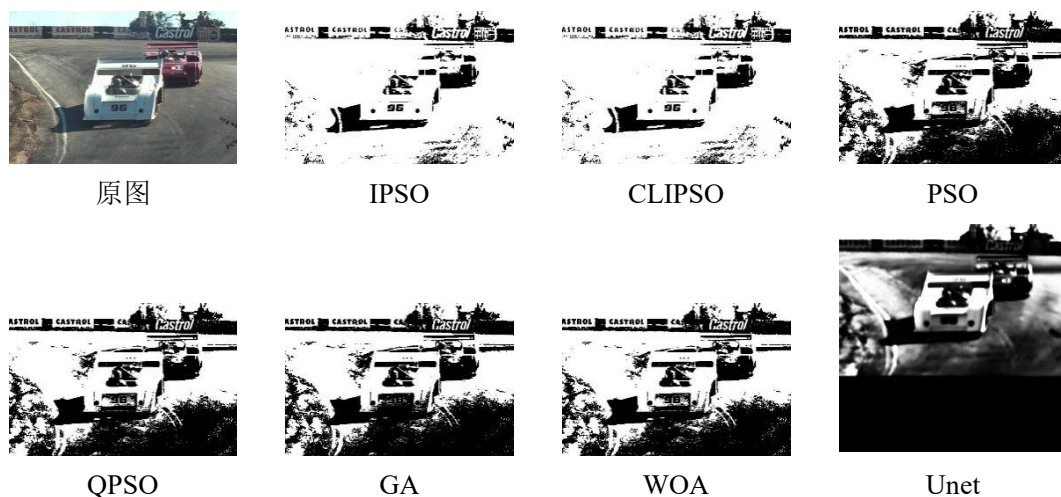


图 5.3 各算法在样图(b)上的分割结果

Fig 5.3 Segmentation results of each algorithm on sample image (b)

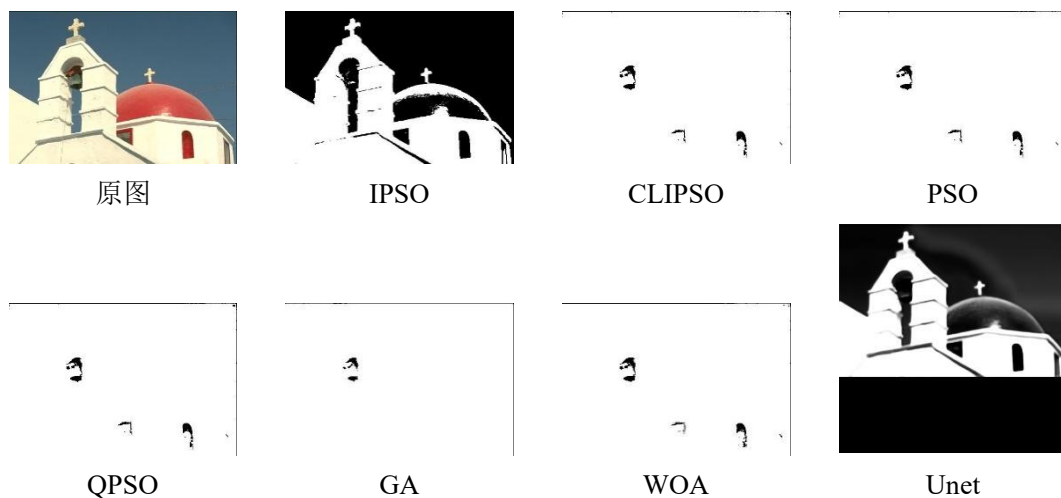


图 5.4 各算法在样图(c)上的分割结果

Fig 5.4 Segmentation results of each algorithm on sample image (c)

在图 5.2, 5.3, 5.4 中, 由于 BSDS500 中的图像大小不一致, 因此在将图像输入 Unet 前需要将图像经过 mask 以及等比缩放处理, 因此得到的图像大小与原图有差异。在上面展示的三张图片中, IPSO 的表现要优于其它元启发式算法, 在图(a)上, IPSO 与 CLIPSO 均可以明显的分割出目标与背景, 其它元启发式算法分割出来的几乎是一片空白, 在图(b)上, 所有算法都分割出了目标与背景, 但与 IPSO 以

及 CLIPSO 相比, 其它元启发式算法将更多的背景划入了目标之中, 在图(c)上, 除了 IPSO 以外, 包括 CLIPSO 在内的所有算法均未分割出目标与背景。与神经网络 Unet 的分割结果相比, Unet 的分割结果比 IPSO 与 CLIPSO 的分割结果要清晰明显, 在图(a)上, 二者表现相差不大, 在图(b)与图(c)上 Unet 的分割结果则要更明显一点。但与 Unet 相比 IPSO 与 CLIPSO 结合二维 Tsallis 熵的图像分割方法可以更广泛的应用在各种图像上, 且前期不需要花费时间与算力进行训练。总的来说, IPSO 与 CLIPSO 结合二维 Tsallis 熵的图像分割精度不如神经网络 Unet, 但其胜在可以花费较小的时间与算力而取得较高的效果。

虽然挑选的样图对比明显, 但在大多数情况下, 不同元启发式算法得到的图片的分割结果用肉眼难以辨别, 因此为了量化描述图像分割结果的优劣, 利用上文引入的评价指标来对各元启发式算法的分割结果进行评价。评价数据如表 5.1 所示。

表 5.1 各元启发式算法分割结果评价

Table 5.1 Evaluation of segmentation results of meta heuristic algorithms

Function	IPSO	CLIPSO	PSO	QPSO	GA	WOA
F1-measure	0.8846	0.8846	0.8782	0.8803	0.8260	0.8788
PA	0.8997	0.8991	0.8904	0.8929	0.8348	0.8903
IOU	0.8195	0.8196	0.8088	0.8123	0.7491	0.8105

由表 5.1 数据可以看出 IPSO 与 CLIPSO 的 F1-Measure 值相同, 为所有算法中最高。IPSO 的 PA 值最高, CLIPSO 次之, 二者均比其他算法的 PA 值高。CLIPSO 的 IOU 值最高, IPSO 次之。由此可以得出两种改进粒子群优化算法的分割效果要优于其他算法的分割效果。

第六章 结 论

由于粒子群优化算法的简单性以及有效性，其被广泛应用于生活中的各个方面，其中就包括图像分割。但粒子群优化算法又存在着早熟收敛的问题，这会导致粒子群优化算法寻找不到全局最优值，在图像分割中就会导致图像分割结果的不准确。本文针对粒子群优化算法早熟收敛的问题，通过修改粒子群优化算法中粒子的学习方式提出了两种改进的粒子群优化算法：

(1)粒子群优化算法中粒子在更新位置时过于依赖全局历史最优位置且其它粒子的信息对当前更新位置的粒子几乎没有任何直接的帮助。为此，提出了减少对全局历史最优位置的依赖且可以直接利用其它粒子位置信息的改进粒子群优化算法。在改进粒子群优化算法中，粒子的学习对象为自身的历史最优位置以及种群中某个其它粒子的历史最优位置，并且在算法每迭代一定次数后所有粒子才会从全局历史最优位置获取信息。通过这种方式，改进的粒子群优化算法可以在低维环境下在一定程度上避免陷入局部最优。

(2)在第一种改进粒子群优化算法中，粒子绝大部分时候的学习对象仍然比较固定，为了进一步扩展粒子更新时的信息来源，在第一种方法的基础上引入了综合学习策略提出了综合学习改进粒子群优化算法。在综合学习改进粒子群优化算法中，粒子的学习对象为随机选择的另外两个粒子通过竞争产生，且粒子是向自身的历史最优位置学习还是其它粒子的历史最优位置学习需要通过一个学习概率进行判断。通过这种方式，综合学习改进粒子群优化算法在低维环境上的表现要稍微优于第一种方法。

为了验证算法避免早熟收敛的能力，将改进粒子群优化算法与综合学习改进粒子群优化算法在 5 个测试函数上与标准粒子群优化算法，量子粒子群优化算法，遗传算法以及鲸鱼优化算法进行了比较。结果表明，这两种算法在低维环境下能有效地避免局部最优。

最后，将改进粒子群优化算法与综合学习改进粒子群优化算法结合二维 Tsallis 熵进行图像分割，并将分割结果与标准粒子群优化算法，量子粒子群优化算法，遗传算法，鲸鱼优化算法结合二维 Tsallis 熵以及神经网络 Unet 的分割结果进行对比。结果显示，改进粒子群优化算法与综合学习改进粒子群优化算法的分割结果要优于其它元启发式算法，但稍逊于神经网络 Unet 的分割结果。

虽然两种改进的粒子群优化算法相比于其它元启发式算法可以更精确的分割图像，但从二者与其它算法在测试函数的表现上来看，这两种改进算法的运算时间要长于其它算法。由于这两种改进算法在迭代过程中需要穿插标准粒子群优化算法，因此在二者后续的改进中可以考虑将标准粒子群优化算法替换为其它表现

更快更好的粒子群优化算法变体。

参考文献

- [1] 尚正阳, 顾寄南, 唐仕喜, 孙晓红. 针对几种元启发式算法的应用性能对比研究[J]. 机械设计与制造, 2021(04): 34-38.
- [2] Holland J H. Genetic Algorithms[J]. Scientific American, 1992, 267(1): 66-72.
- [3] Kennedy J, Eberhart R. Particle swarm optimization[C]//Proceedings of ICNN'95-international conference on neural networks. IEEE, 1995, 4: 1942-1948.
- [4] Mirjalili S, Lewis A. The whale optimization algorithm[J]. Advances in engineering software, 2016, 95: 51-67.
- [5] Jiang X and Li S. BAS: Beetle Antennae Search Algorithm for Optimization Problems[J]. International Journal of Robotics and Control, 2017, 1(1).
- [6] Hatamlou A. Black hole: A new heuristic optimization approach for data clustering, Inf. Sci. (2013) 222: 175-184.
- [7] Shah-Hosseini H. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation[J]. International Journal of Computational Science and Engineering, 2011, 6(1-2): 132-140.
- [8] Wang T and Yang L. Beetle swarm optimization algorithm: Theory and application[J]. arXiv preprint arXiv:1808.00206, 2018.
- [9] Zhang Q, Liu W and Meng X, et al. Vector coevolving particle swarm optimization algorithm[J]. Information sciences, 2017, 394: 273-298
- [10] Shi Y and Eberhart R. A modified particle swarm optimizer[C] //1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). Anchorage: IEEE, 1998: 69-73.
- [11] Shi Y and Eberhart R C. Empirical study of particle swarm optimization[C] //Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). Washington, DC: IEEE Press, 1999, 3: 1945-1950.
- [12] Cheng R and Yao M. Particle swarm optimizer with time-varying parameters based on a novel operator[J]. Applied Mathematics & Information Sciences, 2011, 5:33S-38S.
- [13] Rui M, Member, IEEE, et al. The Fully Informed Particle Swarm: Simpler, Maybe Better[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3):204-210.
- [14] Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer[C]//Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. IEEE, 2005: 124-129.
- [15] Peram T, Veeramachaneni K and Mohan C K. Fitness-distance-ratio based particle swarm optimization[C] //Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat.

No. 03EX706). Indianapolis: IEEE, 2003: 174-181.

- [16] Liang J J, Qin A K and Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(3):281-295.
- [17] Juang C F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004, 34(2): 997-1006.
- [18] 都海波, 沈涵, 周俊,等. 基于自适应变异的天牛群优化算法的模板匹配方法[J]. 计算机应用, 2020, 40(S02):7.
- [19] 邸秋艳. 基于 Tsallis 熵的阈值图像分割方法研究[D]. 秦皇岛: 燕山大学.
- [20] Pun T . Entropic thresholding, a new approach[J]. Computer Graphics & Image Processing, 1981, 16(3):210-239.
- [21] Kapur J N, Sahoo P K, Wong A K C. A new method for gray-level picture thresholding using the entropy of the histogram[J]. Computer vision, graphics, and image processing, 1985, 29(3): 273-285.
- [22] Li C H, Lee C K. Minimum cross entropy thresholding[J]. Pattern recognition, 1993, 26(4): 617-625.
- [23] Tsallis C. Possible generalization of Boltzmann-Gibbs statistics[J]. Journal of Statistical Physics, 1988, 52(1):479-487.
- [24] 张莉, 叶志伟, 王明威. 基于差分进化的二维熵图像分割[J]. 应用科学学报, 2016, 34(1):9.
- [25] Sahoo P K, Arora G. A thresholding method based on two-dimensional Renyi's entropy[J]. Pattern Recognition, 2004, 37(6):1149-1161.
- [26] Sahoo P K, Arora G. Image thresholding using two-dimensional Tsallis-Havrda-Charvát entropy[J]. Pattern recognition letters, 2006, 27(6): 520-528.
- [27] Meng Z, Zhong Y, Mao G, et al. PSO-sono: A novel PSO variant for single-objective numerical optimization[J]. Information Sciences, 2022, 586: 176-191.
- [28] 伍莹芮, 张志勇. 基于改进粒子群人工鱼群算法的二维熵多阈值快速图像分割[J]. 现代计算机, 2021(1):6.
- [29] Ali A B, Luque G, Alba E. An efficient discrete PSO coupled with a fast local search heuristic for the DNA fragment assembly problem[J]. Information Sciences, 2020, 512: 880-908.
- [30] Deng W, Xu J, Zhao H, et al. A novel gate resource allocation method using improved PSO-based QEA[J]. IEEE Transactions on Intelligent Transportation Systems, 2020.
- [31] Shi YH, Eberhart RC. Parameter selection in particle swarm optimization. *Evolutionary Programming VII: Proc. EP 98*, 1998,7(25):591-600.

- [32] 夏学文, 刘经南, 高柯夫, 等. 具备反向学习和局部学习能力的粒子群算法[J]. 计算机学报, 2015(7) : 93-103.
- [33] M. Portes De Albuquerque M P, Esquef I A, Mello A R G. Image thresholding using Tsallis entropy[J]. Pattern Recognition Letters, 2004, 25(9): 1059-1065.
- [34] Fang W, Sun J, Chen H, et al. A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population[J]. Information Sciences, 2016, 330: 19-48.
- [35] Abdel-Khalek S, Ishak A B, Omer O A, et al. A two-dimensional image segmentation method based on genetic algorithm and entropy[J]. Optik, 2017, 131: 414-422.
- [36] Borjigin S, Sahoo P K. Color image segmentation based on multi-level Tsallis–Havrda–Charvát entropy and 2D histogram using PSO algorithms[J]. Pattern Recognition, 2019, 92: 107-118.
- [37] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation[C]//International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015: 234-241.
- [38] Borji A, Cheng M M, Jiang H, et al. Salient Object Detection: A Benchmark[J]. IEEE Transactions on Image Processing, 2015.
- [39] 梁新宇, 罗晨, 权冀川, 肖铠鸿, 高伟嘉. 基于深度学习的图像语义分割技术研究进展[J]. 计算机工程与应用, 2020, 56(02): 18-28.
- [40] Everingham M, Eslami S, Gool L V, et al. The Pascal Visual Object Classes Challenge: A Retrospective[J]. International Journal of Computer Vision, 2015, 111(1):98-136.

厚德 笃学 崇实 尚新