控制与决策
Control and Decision

基于模拟退火机制的自适应粘性粒子群算法

孙一凡, 张纪会

引用本文:
孙一凡,张纪会. 基于模拟退火机制的自适应粘性粒子群算法[J]. 控制与决策, 2023, 38(10): 2764−2772.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2022.0291

基于模拟退火机制的自适应粘性粒子群算法

孙一凡[1,2]，张纪会[1,2†]

(1. 青岛大学 自动化学院，山东 青岛 266071；2. 山东省工业控制技术重点实验室，山东 青岛 266071)

摘 要: 为了进一步提升粒子群算法在离散优化问题中的性能,针对粘性二进制粒子群算法缺乏全局搜索能力、容易陷入局部最优和收敛速度慢的缺点,提出一种新的自适应参数策略和粒子散度指标,并结合模拟退火机制改善该算法的寻优能力. 为了检验算法性能,通过选取不同维数的背包问题算例库以及不同规模的UCI特征选择问题算例库进行仿真实验,并对实验数据进行统计分析. 实验以及分析结果表明,所提算法在寻优精度、算法稳定性和收敛速度上均优于对比算法.

关键词: 二进制粒子群算法；自适应策略；粒子散度；模拟退火；背包问题；特征选择

中图分类号: TP301 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2022.0291

引用格式: 孙一凡,张纪会. 基于模拟退火机制的自适应粘性粒子群算法[J]. 控制与决策, 2023, 38(10): 2764-2772.

Adaptive stickiness particle swarm optimization algorithm based on simulated annealing mechanism

SUN Yi-fan[1,2], *ZHANG Ji-hui*[1,2†]

(1. School of Automation，Qingdao University，Qingdao 266071，China；2. Shandong Key Laboratory of Industrial Control Technology，Qingdao 266071，China)

Abstract: In order to further improve the performance of particle swarm optimization in discrete optimization problems, aiming at the shortcomings of stickiness binary particle swarm optimization, such as lack of global search ability, easy to fall into local optimum and slow convergence speed, a new adaptive parameter strategy and a particle divergence index are proposed, which are combined with simulated annealing mechanism to improve the optimization ability of the algorithm. In order to test the performance of the algorithm, simulation experiments are carried out by selecting the knapsack problem case library with different dimensions and the UCI feature selection problem case library with different scales, and the experimental data are statistically analyzed. The experimental and analytical results show that the proposed algorithm is superior to the comparison algorithm in optimization accuracy, algorithm stability and convergence speed.

Keywords: binary particle swarm optimization；adaptive strategy；particle divergence；simulated annealing；backpack problem；feature selection

0 引 言

粒子群算法(particle swarm optimization, PSO)是一种常用的群体智能优化算法[1],具有概念简单、参数较少、易于实现等优势,主要用来解决连续函数优化问题. 由于标准粒子群算法中位置向量和速度向量定义的连续性,无法直接用来解决离散优化问题. 为了解决这一问题,国内外学者先后提出了很多离散粒子群算法,这些方法大致可分为3类:一是保持粒子运动方程不变,通过转换函数将粒子的速度和位置离散化,使其适用于离散空间,如Kennedy等[2]提出的二进制粒子群算法(binary particle swarm optimization, BPSO),首次将粒子群算法应用于解决一类离散优化问题;二是根据待求解离散优化问题的特点,对算法迭代公式中的符号和运算规则重新定义,使其能够解决相应的离散优化问题[3-7];三是从粒子群算法的基本原理出发,在离散域中重新定义粒子的位置和速度属性以及运动方程,如文献[8]提出的粘性二进制粒子群算法(stickiness binary particle swarm optimization, SBPSO),其对粒子速度、位置和运动方程的定义更适用于二进制空间,寻优效果较

收稿日期: 2022-02-24；**录用日期:** 2022-04-27.
基金项目: 国家自然科学基金项目(61673228, 62072260)；青岛市科技计划项目(21-1-2-16-zhz).
责任编委: 冯俊娥.
†通讯作者. E-mail: zhangjihui@qdu.edu.cn.

好. 同连续PSO一样,这些离散粒子群算法都存在早熟收敛等问题[9],究其原因,主要是由于搜索过程中粒子间快速的信息流动使粒子聚集在一起,使群体多样性下降过快,导致寻优停滞,易陷入局部最优,算法前期的探索能力和后期开发能力有待进一步改善[10-11]. 针对这一问题,文献[12]引入直觉模糊熵作为衡量粒子群状态和速度变化的基本参数,并提出一种基于直觉模糊熵的改进二进制粒子群算法,在解决大规模背包问题时表现较好;文献[13]提出混合粒子群-遗传算法,将遗传操作并入粒子群算法中,算法收敛速度得到提高,但算法后期的开发能力依然不足;文献[14-15]提出基于模拟退火的粒子群算法,在一定程度上降低了算法陷入局部最优的可能;文献[16]使用一个称为粒子年龄的指数来引导粒子走向搜索空间中更有前途的区域,防止粒子群算法在处理多模态优化问题时过早收敛;文献[17]将模拟退火思想和反向操作引入离散粒子群算法,虽然可以帮助粒子摆脱局部最优,但算法收敛速度较慢;文献[18]针对组合排列问题提出一种新的编码方案和邻域扰动方式,提高了算法的寻优能力.

以上这些研究对粒子群算法作了不同程度的完善和改进,但在解决离散优化问题时,算法的性能表现还有待提升. 本文在粘性二进制粒子群算法的基础上,针对其容易陷入局部最优、收敛速度过慢和寻优过程拖延等问题,引入自适应策略和模拟退火机制,提出基于模拟退火机制的自适应粘性粒子群算法(simulated annealing stickiness binary particle swarm optimization, SA-SBPSO),使其探索和开发能力得到平衡. 为了检验改进算法的效果,分别在背包问题和特征选择问题算例上进行仿真实验,并与多种进化算法进行比较. 实验结果表明,SA-SBPSO算法可以在加快收敛速度的同时避免早熟,并且具有较强的跳出局部最优的能力,算法寻优能力得到显著提升.

1　标准粒子群算法、二进制粒子群算法和粘性二进制粒子群算法

首先对标准粒子群、二进制粒子群、粘性二进制粒子群算法的原理进行简要介绍,然后分析粘性二进制粒子群算法的缺点.

1.1　标准粒子群算法

受鸟群捕食行为的启发,标准粒子群算法将鸟群抽象为没有质量的粒子,每个粒子均具备两个属性:位置向量和速度向量. 其中:速度向量表示粒子移动的快慢和方向;位置向量表示粒子在解空间的位置. 每个粒子可以根据目标函数来计算当前位置的适应值,并计算自身历史最优解P,所有粒子通过信息共享得到全局最优解G,随机初始化粒子,并按照以下公式更新粒子速度和位置:

$$V_i^{t+1} = \omega \times V_i^t + c_1 r_1 \times (P_i - X_i^t) + c_2 r_2 \times (G - X_i^t), \quad (1)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}. \quad (2)$$

其中:t是迭代次数,V_i和X_i分别是粒子i的速度和位置,ω是惯性权重,c_1和c_2是学习因子,r_1和r_2是$[0,1]$上均匀分布的随机数,P_i是粒子i的个体历史最优位置,G是粒子群全局最优位置.

1.2　二进制粒子群算法

二进制粒子群算法(BPSO)是在标准粒子群算法的基础上,通过转换函数将速度V的每个分量映射至$[0,1]$区间,表示某位置取为1或0的概率值,从而将粒子群算法应用于离散域来解决0-1离散问题. 粒子的位置和速度更新公式如下:

$$V_{id}^{t+1} = \omega \times V_{id}^t + c_1 r_1 \times (P_{id} - X_{id}^t) + c_2 r_2 \times (G_d - X_{id}^t). \quad (3)$$

$$S(V_{id}^{t+1}) = \frac{1}{1 + \mathrm{e}^{-V_{id}^{t+1}}}. \quad (4)$$

$$X_{id}^{t+1} = \begin{cases} 1, & r < S(V_{id}^{t+1}); \\ 0, & \text{otherwise}. \end{cases} \quad (5)$$

其中:X_{id}取值为0或1,t为迭代次数,d为速度和位置分量,$S(\cdot)$为转换函数,r为$[0,1]$区间均匀分布的随机数.

虽然转换函数的引入使粒子群算法能够解决离散问题,但将连续域中速度和位置的概念直接应用于离散域是不合适的,因为其对离散空间位置和速度的描述并不直观.

1.3　粘性二进制粒子群算法

粘性二进制粒子群算法(SBPSO)[8]针对离散搜索空间定义了两个概念:粘性(stickiness)和翻转(flip). 粘性是指在离散域中,粒子保持当前比特位不变的倾向;翻转是指粒子比特位的0-1互换. 因此,将粒子的速度重新定义为翻转概率来更好地描述粒子在离散域中的速度和位置变化情况,在解决离散问题时效果较BPSO得到一定提升. 其速度和位置更新公式如下:

$$V_{id}^{t+1} = i_s \times (1 - \mathrm{stk}_{id}^t) + i_p \times |P_{id} - X_{id}^t| + i_g \times |G_d - X_{id}^t|. \quad (6)$$

$$X_{id}^{t+1}=\begin{cases}1-X_{id}^{t},\ r<V_{id}^{t+1};\\ X_{id}^{t},\ \text{otherwise}.\end{cases}\tag{7}$$

其中: X_{id}取值为0或1; stk_{id}^{t}表示在第t次迭代时,第i个粒子第d维的粘性,当粒子i第d维刚发生翻转时,粒子i在该维的粘性为1,然后粘性值随着迭代进行不断衰减,直到该维粘性为0或再次翻转; i_s表示粘性权重; i_p表示认知权重; i_g表示社会权重, $i_s+i_p+i_g=1, \alpha=\dfrac{i_p}{i_g}$($\alpha$为常数); $|\cdot|$表示汉明距离. 为了平衡算法的探索和开发能力,采用以下动态参数控制策略:

$$i_s=i_{s_u}-(i_{s_u}-i_{s_l})\times\frac{\text{iter}}{\text{maxiter}}.\tag{8}$$

$$i_p=\alpha\times\frac{1-i_s}{1+\alpha}.\tag{9}$$

$$i_g=\frac{1-i_s}{1+\alpha}.\tag{10}$$

$$\text{ustk}^{t}=\text{ustk}^{l}+(\text{ustk}^{u}-\text{ustk}^{l})\times\frac{\text{iter}}{\text{maxiter}}.\tag{11}$$

$$\text{stk}_{id}^{t+1}=\begin{cases}1,\ d_{\text{th}}\text{ bit just flipped};\\ \max\left\{\text{stk}_{id}^{t}-\dfrac{1}{\text{utsk}},0\right\},\ \text{otherwise}.\end{cases}\tag{12}$$

其中: i_{s_u}和i_{s_l}分别表示i_s的上界和下界, iter和maxiter分别表示当前迭代次数和最大迭代次数, ustk表示粒子粘性值从1衰减到0所需的迭代次数, ustk^u和ustk^l分别表示ustk的上界和下界.

粘性概念的提出,是为了保证当P和G没有改变,粒子经过大量迭代而比特位没有翻转时,会有一定的翻转倾向来跳出当前位置,从而在一定程度上避免陷入局部最优. 然而,笔者通过实验发现该算法容易陷入局部最优,究其原因,从式(6)可以看出,当粒子位于全局最优位置时,翻转倾向仅依靠动量和粘性衰减获得,而通过大量迭代使粘性衰减来获得翻转倾向不仅拖延了算法寻优进程,也不利于算法实现探索和开发上的平衡. 因此,有必要通过适当方式来增强算法的寻优效率和跳出局部最优的能力. 再者,作为粒子群算法在离散域中的应用,仅对动量权重系数i_s采取动态调整策略,而没有考虑算法进行过程中i_p和i_g比例的动态调整,无法保持较好的搜索效果和收敛速度,也需要进一步改进.

2 改进的粘性二进制粒子群算法

2.1 自适应策略

2.1.1 学习权重调整

标准的二进制粒子群算法和粘性二进制粒子群算法对于粒子自我认知和社会认知权重系数的设置,一般是通过经验或实验方式获得,在算法迭代过程中保持不变,这一方式虽然简易可行,但是在算法寻优的不同时期无法实时调整寻优重点,无法兼顾算法的前期探索能力和后期开发能力. 在算法寻优前期,希望粒子有较强的自我认知能力,运动偏向个体最优位置,来提高前期的搜索能力,随着迭代的进行,希望粒子不断均衡自我认知和社会认知能力,加强粒子间交流,以此达到自适应调整的目的. 通过测试发现,反正切函数的变动趋势对该算法参数调整比较契合,因此将其应用于参数自适应调整,具体计算方式为

$$\alpha=-\mathcal{H}\times\arctan\frac{\text{iter}}{\text{maxiter}}+\Theta.\tag{13}$$

其中: iter为算法当前迭代次数, maxiter为预先设定的最大迭代次数, $\mathcal{H}$和Θ为调节系数.

2.1.2 散度指标

针对二进制空间搜索的特殊性,综合考虑粒子的外部表现(适应度值)和内部结构(二进制序列),提出一种新的散度指标来衡量粒子群进化过程中的粒子多样性,具体计算方式为

$$d_{x\_\text{mean}}^{t}=\frac{1}{N}\sum_{i=1}^{N}|X_i^t-G|,\tag{14}$$

$$d_{x\_\text{min}}^{t}=\min\{|X_i^t-G|,\ i=1,\dots,N\},\tag{15}$$

$$d_{x\_\text{max}}^{t}=\max\{|X_i^t-G|,\ i=1,\dots,N\},\tag{16}$$

$$d_{f\_\text{mean}}^{t}=\frac{1}{N}\sum_{i=1}^{N}|f(X_i^t)-f(G)|,\tag{17}$$

$$d_{f\_\text{min}}^{t}=\min\{|f(X_i^t)-f(G)|,\ i=1,\dots,N\},\tag{18}$$

$$d_{f\_\text{max}}^{t}=\max\{|f(X_i^t)-f(G)|,\ i=1,\dots,N\},\tag{19}$$

$$\text{Div\_xf}^{t}=\frac{d_{x\_\text{mean}}^{t}-d_{x\_\text{min}}^{t}}{d_{x\_\text{max}}^{t}-d_{x\_\text{min}}^{t}}\times\frac{d_{f\_\text{mean}}^{t}-d_{f\_\text{min}}^{t}}{d_{f\_\text{max}}^{t}-d_{f\_\text{min}}^{t}}.\tag{20}$$

其中: N表示粒子数; $d_{x\_\text{mean}}^{t}$、$d_{x\_\text{min}}^{t}$、$d_{x_\text{max}}^{t}$分别表示算法在第t次迭代时,所有粒子的当前位置到全局最优位置的平均、最小和最大距离(汉明距离); $d_{f\_\text{mean}}^{t}$、$d_{f\_\text{min}}^{t}$、$d_{f_\text{max}}^{t}$分别表示在第t次迭代时,所有粒子的当前适应度值到全局最优值的平均、最小和最大距离. 由式(20)可见:当$d_{x_\text{mean}}^{t}=d_{x_\text{min}}^{t}$或$d_{f_\text{mean}}^{t}=d_{f_\text{min}}^{t}$时, $\text{Div\_xf}^{t}=0$,此时粒子群距离全局最优粒子整体偏近;当$d_{x_\text{mean}}^{t}=d_{x_\text{max}}^{t}$且$d_{f_\text{mean}}^{t}=d_{f_\text{max}}^{t}$时, $\text{Div\_xf}^{t}=1$,此时粒子群距离全局最优粒子整体偏远. 然而随着算法的进行,粒子多样性势必呈下降趋势,为了在粒子多样性和聚集性上取得一致从而平衡算法的探索和开发能力,当$\text{Div_xf}^{t}-\xi<0$(ξ为常数)时对粒子群采取相应措施,增加粒子群多样性和寻优能力,为此,本文采取模拟退火机制来引导粒子群进化.

2.2　模拟退火机制

模拟退火算法(simulated annealing, SA)是一种启发式随机寻优算法, 它模拟了物理退火过程, 从给定的初始高温开始, 利用具有概率突跳特性的Metropolis准则在解空间中进行随机搜索, 经过重复降温及热平衡达到, 最终得到全局最优解. 其中, Metropolis准则是模拟退火算法收敛于全局最优解的关键所在, Metropolis准则以一定概率接受差解, 可以有效避免陷入局部最优和克服初值依赖性. 关于接受概率的Metropolis准则为

$$p=\begin{cases}1, & E(X_{\text{new}})-E(X_{\text{old}})\leqslant 0;\\ \mathrm{e}^{-\frac{E(X_{\text{new}})-E(X_{\text{old}})}{T}}, & E(X_{\text{new}})-E(X_{\text{old}})>0.\end{cases}\tag{21}$$

其中: $E(X)$ 表示解 X 对应的能量值, X_{old} 和 X_{new} 分别表示扰动前的旧解和扰动后的新解, T 表示温度参数.

3　SA-SBPSO算法

本文将模拟退火算法与粘性二进制粒子群算法相结合, 并采取自适应策略, 提出SA-SBPSO算法, 基本流程如下.

step 1: 初始化粒子群, 为每个粒子赋予随机的初始位置和速度, 初始参数设定.

step 2: 计算初代粒子最优值和次优值差 Δ, 得到模拟退火初始温度 T_0.

step 3: 按式(13)对 α 采取自适应调整策略.

step 4: 计算个体适应值, 更新粒子个体历史最优位置和全局最优位置.

step 5: 按式(14)～(20)计算粒子群散度指标Div_xf, 若指标条件满足, 则执行step 6, 反之执行step 7.

step 6: 进入循环. 随机选择一种扰动方式, 对全局最优解进行邻域扰动, 记录较优解, 判定是否达到扰动上限, 如果达到, 则将扰动得到的最优解作为新解, 根据Metropolis准则计算接受概率并更新全局最优解.

step 7: 按式(8)～(12)更新参数, 降温.

step 8: 按式(6)和(7)更新粒子速度和位置.

step 9: 判断是否达到停止条件, 达到则输出最优解, 算法结束, 反之则返回step 3.

针对二进制序列所采取的邻域扰动方式为, 对当前解的随机互异两位进行交换操作或者对随机选中的一位进行翻转操作来产生新解(如图1所示), 这一过程重复 L 次(L 为马尔可夫链长度).

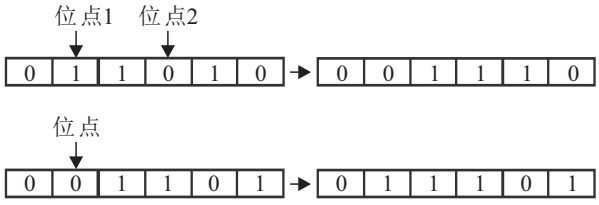

图 1　邻域扰动(上图是交换操作, 下图是翻转操作)

综上, 本文提出的SA-SBPSO算法伪代码如下.

表 1　SA-SBPSO算法伪代码

```
begin
N = min{n, 100}
for i ← 1 to N do
  randomly initialize x_i^0 and v_i^0
end for
acquire f_0^best and f_0^sub_best
Δ = |f_0^best − f_0^sub_best|, T_0 = −|Δ| / ln P_0
parameter setting
for t ← 1 to maxiter do
  adaptive adjustment α according to (13)
  for i ← 1 to N do
    evaluate f(x_i^t)
    update P_i
  end for
  update G
  compute Div_xf^t according to (14)～(20)
  if Div_xf^t − ξ < 0 then
    for j ← 1 to L do
      Neighbor_Dist(G)
      Merit()
    end for
    Metropolis()
    update G
  end if
  T = max{T × β, T_min}
  parameter update according to (8)～(12)
  modify x_i^{t+1} and v_i^{t+1} according to (6)、(7)
end for
end
```

4　实验设计与结果分析

4.1　实验环境设置

算法测试环境为Win10操作系统, Intel(R) Core (TM) i5-9500 CPU处理器, 8 GB内存, Python3. 8编程语言, PyCharm2021.1.3开发环境.

4.2　实验方案

为了检验改进算法的性能, 采用0-1背包问题和特征选择问题数据集对遗传算法、二进制粒子群算法、粘性二进制粒子群算法和本文提出的SA-SBPSO算法进行测试并比较分析. 对每一算例, 粒子数(种群规模) N 设置为 $\min\{n, 100\}$, n 为问题维数, 初始温度设置为 $T_0=\dfrac{-|\Delta|}{\ln P_0}$, Δ 表示初代粒子最优值和次优值的差. 算法相关参数设置如表2所示. 其中: P_0 为

初始概率,β为降温系数,maxiter为算法最大迭代次数,L为马尔可夫链长度,$[\cdot]$为取整运算,$\mathcal{H}$、Θ、ξ为常数. 对于每个算例,算法独立运行30次并统计相应指标.

表 2 参数设置

参数	P_0	β	maxiter	L	$\mathcal{H}$	Θ	ξ
背包问题	0.9	0.99	1 000	$[0.8\times n]$	1	$(\pi+1)/2$	1
特征选择问题	0.85	0.9	100				

0-1背包问题是组合优化领域的一个典型问题,并且已被证明是NP完全问题. 0-1背包问题有着广泛的理论和实际应用背景,如投资决策、货物装载、材料切割问题等. 特征选择问题是从一组特征中挑选出一些最有效的特征以降低特征空间维数的过程,是模式识别的关键问题之一,相较于背包问题处理过程更加复杂. 上述二者均为典型的离散二进制优化问题.

4.3 实验1

4.3.1 0-1多维背包问题(MKP)

0-1多维背包问题可以简单描述为:给定n个物品和一个背包,每个物品有其价值和对m项资源的占用,背包对每项资源有不同的限制. 目标是在满足背包对不同资源限制的同时,使放入物品的总价值最大. 其数学模型为

$$\max f=\sum_{i=1}^{n} p_i\times x_i.$$

$$\text{s.t.}\ \sum_{i=1}^{n} w_{ij}\times x_i\leqslant C_j,\ j=1,\dots,m;$$

$$x_i=\begin{cases}0,\ \text{物品}\,i\,\text{未被放入},\\ 1,\ \text{物品}\,i\,\text{被放入},\end{cases}\quad i=1,\dots,n.$$

其中:x_i表示物品i是否被放入背包,p_i表示物品i的价值,w_{ij}表示物品i对资源j的占用,C_j表示背包对资源j的约束.

4.3.2 实验1数据信息

针对背包问题,用两个著名的背包问题测试集进行测试:第1个是SAC-94基准测试集,包含6个数据集,每个数据集附有若干算例,物品个数从10到105不等,约束条件数从2到30不等. 第2个是GK基准测试集,包含一个GK数据集,附有11个算例,物品数从100到2 500不等,约束条件数从15到1 500不等. 背包问题数据集基本信息如表3所示.

表 3 背包问题数据集基本信息

数据集	算例个数	物品数	约束个数
hb	2	28～35	2～4
pb	6	27～37	4～30
pet	6	10～50	5～10
sento	2	60	30
weing	8	28～105	2
weish	30	30～90	5
GK	11	100～2 500	15～100

4.3.3 实验1结果

对于SAC-94基准测试集,问题规模较小,统计算法准确率,即找到最优解的能力;对于GK基准测试集,由于物品个数和约束个数较多,问题规模较大,统计算法找到最优解的平均值、最优值和标准差来衡量算法的寻优能力和稳定性. 实验结果见表4和表5(表中最优求解准确率用下划线标明,最优求解平均值、最优值、标准差用加粗标明).

表 4 算法求解准确率对比(SAC-94基准测试集)

数据集	GA	BPSO	SBPSO	SA-SBPSO
hb	<u>0.53</u>	0.13	0.40	0.47
pb	<u>0.53</u>	0.23	0.30	0.50
pet	0.67	0.37	0.67	<u>0.83</u>
sento	0.30	0.23	0.33	<u>0.60</u>
weing	0.50	0.40	0.63	<u>0.87</u>
weish	0.53	0.50	0.67	<u>0.87</u>

表 5 算法求解最优值、平均值和标准差对比(GK基准测试集)

案例	GA			BPSO			SBPSO			SA-SBPSO		
	opt	avg	sd	opt	avg	sd	opt	avg	sd	opt	avg	sd
GK01	3 690	3 678.11	**8.25**	3 722	3 702.34	8.91	3 735	3 722.07	10.16	**3 751**	**3 732.67**	8.90
GK02	3 846	3 814.16	10.04	3 899	3 874.91	10.65	3 920	3 908.43	14.32	**3 944**	**3 928.71**	**9.41**
GK03	5 478	5 450.60	10.28	5 534	5 488.36	11.32	5 578	5 557.60	13.61	**5 608**	**5 595.20**	**9.73**
GK04	5 610	5 580.93	11.40	5 657	5 601.72	13.24	5 692	5 661.50	15.10	**5 726**	**5 705.83**	**10.57**
GK05	7 415	7 398.13	12.83	7 458	7 412.58	16.68	7 446	7 423.87	20.21	**7 502**	**7 485.58**	**11.18**
GK06	7 460	7 435.78	**8.75**	7 568	7 540.38	11.65	7 568	7 529.73	19.22	**7 626**	**7 607.01**	11.22
GK07	18 528	18 481.47	**18.36**	18 810	18 713.42	23.47	18 832	18 785.87	29.27	**19 066**	**19 032.47**	19.48
GK08	18 276	18 234.50	**14.53**	18 454	18 425.17	14.67	18 456	18 425.91	28.05	**18 640**	**18 612.46**	17.95
GK09	56 001	55 934.43	30.18	56 549	56 490.73	**25.01**	56 636	56 494.93	62.06	**57 552**	**57 476.39**	30.21
GK10	55 724	55 663.30	**22.40**	56 121	56 039.22	27.50	56 134	56 045.28	55.13	**56 730**	**56 672.76**	24.26
GK11	93 205	93 121.97	**25.74**	93 532	93 407.21	57.19	93 574	93 476.27	56.77	**94 274**	**94 176.43**	45.27

4.3.4　实验 1 结果分析

由表 4 所列数据可以发现: 改进后的 SA-SBPSO 算法在 SAC-94 基准测试集上较 SBPSO 算法准确性得到提升, 求解效果整体优于对比算法; 遗传算法在 SAC-94 基准测试集中的 hb 和 pb 数据集上准确率较高, 而在其他数据集上表现较差, 说明遗传算法在物品个数和约束条件较少时寻优能力较好; 传统二进制粒子群算法相较于其他算法准确率较低, 寻优精确性较差. 由表 5 所列数据可以发现, 在 GK 基准测试集上, 改进后的 SA-SBPSO 算法较其他算法具有更好的寻优精度、收敛速率和稳定性. 在高维背包问题上, SA-SBPSO 算法性能明显优于其他算法, 如图 2 所示, 对于 GK08 算例, 改进算法在 200 代前后就已经找到了原算法的最优解, 并且继续保持较好的探索和开发能力.

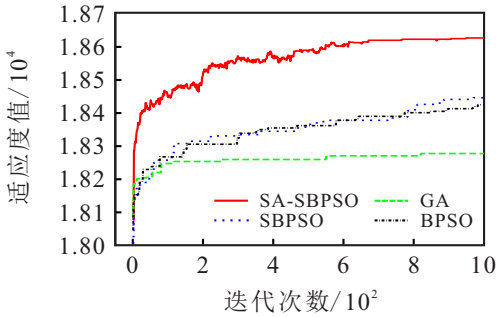

图 2　4 种算法在 GK08 问题上的求解曲线

当物品个数和约束条件较多时, 遗传算法的寻优能力受到限制, 同时从相应标准差及最优值可以看出其容易早熟和陷入局部最优, 寻优能力低下. 从图 3 可以看出, 传统二进制粒子群算法较遗传算法寻优能力有一定程度提升, 但仍差于其他两种算法.

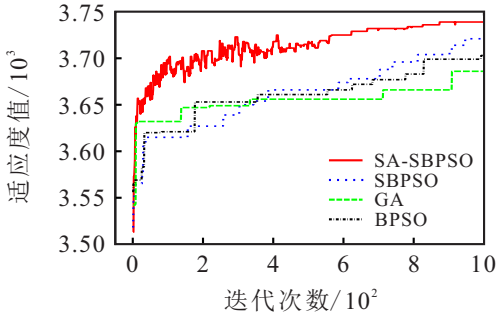

图 3　4 种算法在 GK01 问题上的求解曲线

由图 4 可以看出, 在算法迭代前中期, SA-SBPSO 算法的 Div_xf 值总体上大于原算法的 Div_xf 值, 说明在这一阶段改进算法的粒子多样性、算法效果优于原算法, 具有更好的探索能力. 随着算法的进行, 在迭代后期 SA-SBPSO 算法的 Div_xf 值呈下降趋势, 并总体上小于原算法的 Div_xf 值, 说明在这一阶段改进算法的收敛效果较好, 具有较强的开发能力.

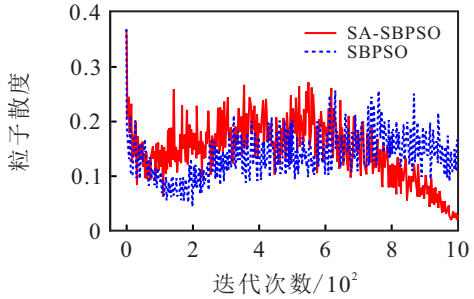

图 4　粒子群散度变化

4.3.5　测试结果统计分析

1) 为说明实验结果的科学性, 对本文所列算法在测试算例上的 30 次实验结果进行统计分析[19], 限于篇幅, 仅以 SA-SBPSO 算法求解 GK01 算例为例进行分析, SA-SBPSO 算法在 GK01 算例上的 30 次实验结果(升序排序后)如表 6 所示.

表 6　测试结果

测试结果 Y				
3 712	3 717	3 722	3 722	3 723
3 724	3 725	3 726	3 728	3 728
3 730	3 731	3 731	3 731	3 732
3 733	3 734	3 735	3 736	3 736
3 738	3 738	3 738	3 740	3 741
3 742	3 745	3 745	3 746	3 751

计算可得表 6 数据的统计参数: 样本均值 $\overline{Y}$ 为 3 732.67, 样本标准差 s 为 8.90. 取分布区间为 [3 710, 3 760], 取组距为 10, 得到概率分布直方图如图 5 所示.

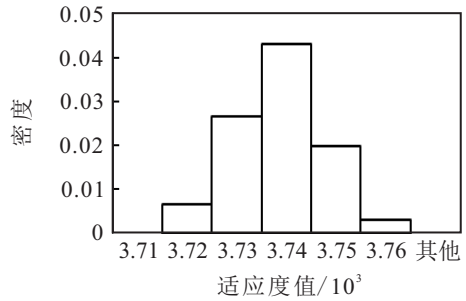

图 5　GK01 问题 30 次结果概率分布直方图

本次统计分析样本容量为 30, 还不足以精确描述数据的分布情况, 但易见图 5 平滑后的形状约呈钟形, 可大致判断表 6 数据 Y 近似服从正态分布, 即 $Y \sim N(\mu, \sigma^2)$, 参数 μ、σ^2 未知.

根据区间估计原理, 取显著性水平 $\tau = 0.05$, 置信水平为 $1-\tau = 0.95$, 可以求出表 6 数据相关参数的区间估计:

均值 μ 在置信水平为 0.95 的置信区间为 $\left(3\,732.67 \pm t_{0.025}(29) \times \dfrac{8.90}{\sqrt{30}}\right)$, 即 (3 729.35, 3 735.99), 说明对于 GK01 算例, SA-SBPSO 算法实验结果的均值在 3 729.35 与 3 735.99 之间, 这个估计的可信度为 95 %. 若以此区间的任意值作为 μ 的近似值, 其误差

不大于$\frac{8.90}{\sqrt{30}}\times 2.045\,2=3.32$,这个误差估计的可信程度为95%.

标准差σ在置信水平为0.95的置信区间为$\left(\frac{\sqrt{29}\times 8.90}{\sqrt{\chi^2_{1-0.025}(29)}},\ \frac{\sqrt{29}\times 8.90}{\sqrt{\chi^2_{1-0.025}(29)}}\right)$,即(7.09, 11.93),说明对于GK01算例,SA-SBPSO算法实验结果的标准差在7.09与11.93之间,估计的可信度为95%.

2) 为检验改进算法实验结果Y与原算法实验结果Y'是否存在显著性差异,对实验数据进行假设检验,以GK01为例,两个算法的30次实验结果差值如表7所示. 其中:"+"表示提升,"−"表示降低.计算可得表7数据的统计参数:样本均值$\overline{Y-Y'}$为10.60,样本标准差s_-为11.81.

表7 测试结果差值

测试结果差值$(Y-Y')$				
+26	−9	+7	+5	+9
+23	+12	+7	+5	−7
+34	+20	+2	+8	+12
+14	+20	+23	−13	+24
−15	+23	+26	+6	+5
+14	+17	+8	+13	−1

易知:$Y_i-Y_i'\ (i=1,2,\ldots,30)$是来自正态总体$N(\mu_-,\sigma_-^2)$的样本,参数$\mu_-$、$\sigma_-^2$未知.

根据假设检验原理,取显著性水平$\tau=0.05$,检验假设:$H_0:\mu_-\leqslant 0;H_1:\mu_->0$.

构造统计量

$$\frac{\overline{Y-Y'}}{s_-/\sqrt{30}}=\frac{10.60}{11.81/\sqrt{30}}=$$

$$4.916\,1>t_{0.05}(29)=1.699\,1,$$

故拒绝假设H_0,认为$\mu_->0$,即在显著性水平$\tau=0.05$下,认为对于GK01算例,改进算法的每次实验结果比原算法均有提升,且提升均值为10.60.

对其他算法及相应算例的实验结果也进行了同样的统计分析,结果类似,这里不再一一列举.

4.4 实验2

4.4.1 特征选择问题

特征选择是指从数据集对应的M个特征(属性)中选择N个特征(属性)使得系统的特定指标函数最优,同时也是删除冗余特征、无关特征和干扰特征,降低数据集维度的过程. 本文由所给算法得出特征子集对应解集,停止准则设定为算法最大迭代次数. 特征选择问题的基本流程如图6所示.

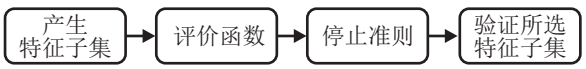

图6 特征选择流程

特征选择问题评价函数如下:

$$\min f=\lambda\times \text{ErrorRate}+(1-\lambda)\times\frac{\text{num}_{\text{select}}}{\text{num}_{\text{all}}}.$$

其中:λ表示权重系数(一般将λ设置为0.9来突出分类性能的重要性),ErrorRate表示分类错误率(通过KNN算法计算得出),$\text{num}_{\text{select}}$表示所选择的特征数量,$\text{num}_{\text{all}}$表示全部特征数量.

4.4.2 KNN算法

本文采用KNN分类算法计算分类错误率,对于所产生的特征子集进行评估. KNN算法即K近邻算法,具有逻辑简单、便于理解、实用性强、易于实现和无需估计参数等优点,被广泛应用于机器学习领域,是最简单的分类算法之一. KNN算法的核心思想是,如果一个样本在特征空间中的K个最相邻的样本中的大多数属于某一个类别,则该样本也属于这个类别,并且具有这个类别上样本的特性.

对于K值的选取,大量研究表明,选择合适的K值是建立精确KNN分类模型的必要条件,K值过小会导致估计误差变大,出现过拟合现象;K值过大则会导致近似误差变大,出现欠拟合现象. 本文取$K=5$来均衡近似误差和估计误差[8].

对于数据集的处理:导入数据后,对属性列进行归一化处理,避免因数值差异和奇异样本导致的不良影响;然后将数据顺序打乱重排(使得数据划分更有随机性),分别提取其标签数据和属性值数据待算法调用.

对于数据集的划分:采取常用规则,训练集数据比例设置为0.8(向上取整),测试集数据比例设置为0.2(向下取整).

4.4.3 实验2数据信息

针对特征选择问题:在UCI特征选择数据集中进行测试,数据集特征数从13到617不等,样本数从101到4 400不等,特征选择数据集基本信息如表8所示.

表8 特征选择数据集基本信息

数据集	特征数	样本数	类别数
wine	13	178	3
zoo	16	101	7
segmentation	19	2 310	7
german	24	1 000	2
WBCD	30	569	2
Ionosphere	34	351	2
sonar	60	208	2
movementlibras	90	360	15
hillvalley	100	606	2
musk1	166	476	2
madelon	500	4 400	2
isolet	617	1 559	26

4.4.4 实验2结果

对于特征选择问题,首先侧重的是分类性能,其次是选择特征数,因此,分别统计算法的分类成功率和选择特征个数来衡量算法寻优能力. 对每个数据集,算法独立运行30次并统计相应指标,实验结果见表9和表10(表中最优选择特征数用加粗标明,最优训练和测试准确率用下划线标明).

表9　选择特征数和训练准确率对比

数据集	GA		BPSO		SBPSO		SA-SBPSO	
	特征数	训练准确率/%	特征数	训练准确率/%	特征数	训练准确率/%	特征数	训练准确率/%
wine	4.10	96.43	4.41	96.57	**3.06**	96.53	**3.06**	96.53
zoo	6.50	97.50	6.11	97.34	5.62	97.29	**5.60**	97.50
segmentation	6.52	95.18	6.80	95.18	5.60	95.03	**5.38**	95.12
german	7.80	82.05	8.46	82.15	**6.36**	82.43	6.40	82.78
WBCD	4.20	95.86	3.88	96.21	3.71	96.25	**3.43**	96.30
Ionosphere	7.28	94.63	**5.30**	94.16	5.40	95.21	5.86	95.44
sonar	18.04	95.91	17.78	92.89	16.11	96.08	**15.20**	96.52
movementlibras	32.35	87.63	25.50	91.01	**23.22**	91.19	24.40	91.30
hillvalley	37.10	76.45	39.67	79.44	36.50	79.24	**30.27**	79.42
musk1	73.74	96.15	63.61	98.28	55.10	98.69	**48.78**	98.87
madelon	227.68	88.72	215.44	90.13	200.53	90.31	**190.40**	91.75
isolet	232.09	93.31	232.02	95.31	226.79	95.59	**215.21**	95.75

表10　测试准确率对比　　%

数据集	GA	BPSO	SBPSO	SA-SBPSO
wine	96.48	96.51	96.50	96.50
zoo	96.88	96.85	96.83	97.04
segmentation	93.17	93.22	93.26	93.26
german	68.94	69.13	70.74	72.61
WBCD	92.36	92.49	93.55	95.47
Ionosphere	92.48	93.09	93.83	94.59
sonar	81.67	82.58	82.11	84.09
movementlibras	78.33	80.26	82.66	83.38
hillvalley	54.80	57.79	57.85	58.24
musk1	85.46	86.33	86.31	87.56
madelon	75.84	78.60	79.96	81.97
isolet	91.05	94.68	94.77	94.89

4.4.5 实验2结果分析

从表9所列数据可以看出,改进后的SA-SBPSO算法在小型特征选择问题上(如wine、zoo等数据集),较原算法寻优能力具有小幅提升,在选择特征数上虽然会有个别数据集略差于对比算法,但在准确率上优于对比算法,这与评价函数中分类精度的权重有较大关系. 对于特征选择问题,算法的直接优化目标是在选择特征数和分类精度之间权衡,实现评价函数的最优化. 在较大型特征选择问题上(如sonar、movementlibras和isolet等数据集),无论是在选择特征数还是准确率上均优于对比算法,寻优能力得到较大提升.

遗传算法在zoo、segmentation等小型特征选择问题上表现良好,当问题规模变大时则表现较差,说明其探索和开发能力不足,容易早熟和陷入局部最优. 随着问题维数提升,传统二进制粒子群算法的表现从持平到逐渐优于遗传算法,说明其对中高维问题有较好的寻优能力.

从表10测试集实验结果可以看出:改进后的SA-SBPSO算法在绝大部分数据集上可以保持较高的测试准确率,寻优效果明显好于其他算法,泛化能力较好;对于hillvalley和madelon数据集,4种算法均出现过拟合现象,在测试集表现较差,说明当数据集中干扰属性影响过大以及各属性之间存在复杂关系时,所列算法无法较好处理,寻优能力有待进一步改善.

5　结　论

本文提出的SA-SBPSO算法在学习参数设定上采取了自适应调整方案,使算法收敛效果得到改善. 为避免算法陷入局部最优带来的停滞现象,提出一种新的粒子散度指标,结合模拟退火机制使得算法的探索和开发能力得到平衡,提高了算法的寻优能力. 通过实验结果看出,改进算法对于SAC-94基准测试集仍然不能有很高的准确率,对于特征选择问题也存在一定的改进空间. 下一步工作是要进一步完善算法的相关参数设置、退火温度控制和邻域扰动方式,扩展该算法的应用领域,使其不仅能有效解决0-1离散问题,也可以有效解决其他的离散优化问题.

参考文献(References)

[1] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proceedings of ICNN'95-International Conference on Neural Networks. Perth, 1995: 1942-1948.

[2] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]. International Conference on Systems, Man, and Cybernetics. Orlando, 1997: 4104-4109.

[3] Zheng R Z. An improved discrete particle swarm optimization for airline crew rostering problem[C]. IEEE Congress on Evolutionary Computation. Glasgow, 2020: 1-7.

[4] Bansal J C, Deep K. A modified binary particle swarm optimization for knapsack problems[J]. Applied Mathematics and Computation, 2012, 218(22): 11042-11061.

[5] Zhu W, Kai Z. A novel discrete particle swarm optimization algorithm for solving graph coloring problem[J]. Journal of Computational and Theoretical Nanoscience, 2015, 13(6): 3588-3594.

[6] 陈恩修, 刘希玉. 一种简便高效的二元离散粒子群算法[J]. 控制与决策, 2010, 25(2): 255-258.
(Chen E X, Liu X Y. Fast and easy binary discrete particle swarm optimization algorithm[J]. Control and Decision, 2010, 25(2): 255-258.)

[7] Chen W N, Zhang J, Chung H S H, et al. A novel set-based particle swarm optimization method for discrete optimization problems[J]. IEEE Transactions on Evolutionary Computation, 2010, 14(2): 278-300.

[8] Nguyen B H, Xue B, Andreae P, et al. A new binary particle swarm optimization approach: Momentum and dynamic balance between exploration and exploitation[J]. IEEE Transactions on Cybernetics, 2021, 51(2): 589-603.

[9] 王皓, 欧阳海滨, 高立群. 一种改进的全局粒子群优化算法[J]. 控制与决策, 2016, 31(7): 1161-1168.
(Wang H, Ouyang H B, Gao L Q. An improved global particle swarm optimization[J]. Control and Decision, 2016, 31(7): 1161-1168.)

[10] 刘建华, 杨荣华, 孙水华. 离散二进制粒子群算法分析[J]. 南京大学学报: 自然科学版, 2011, 47(5): 504-514.
(Liu J H, Yang R H, Sun S H. The analysis of binary particle swarm optimization[J]. Journal of Nanjing University: Natural Sciences, 2011, 47(5): 504-514.)

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

[12] 汪禹喆, 雷英杰, 周林, 等. 直觉模糊离散粒子群算法[J]. 控制与决策, 2012, 27(11): 1735-1739.
(Wang Y Z, Lei Y J, Zhou L, et al. Intuitionistic fuzzy discrete particle swarm algorithm[J]. Control and Decision, 2012, 27(11): 1735-1739.)

[13] Katiyar S. A comparative study of genetic algorithm and the particle swarm optimization[J]. ABES institute of technology, 2016, 9(2): 215-223.

[14] 闫群民, 马瑞卿, 马永翔, 等. 一种自适应模拟退火粒子群优化算法[J]. 西安电子科技大学学报, 2021, 48(4): 120-127.
(Yan Q M, Ma R Q, Ma Y X, et al. Adaptive simulated annealing particle swarm optimization algorithm[J]. Journal of Xidian University, 2021, 48(4): 120-127.)

[15] 姚若侠, 薛丹, 谢娟英, 等. 求解0-1背包问题的混合粒子群改进算法研究[J]. 华东师范大学学报: 自然科学版, 2020(6): 90-98.
(Yao R X, Xue D, Xie J Y, et al. Study of an improved hybrid particle swarm optimization algorithm for solving 0-1 knapsack problems[J]. Journal of East China Normal University: Natural Science, 2020(6): 90-98.)

[16] Jiang B, Wang N, Li X D. Particle swarm optimizer with aging operator for multimodal function optimization[J]. International Journal of Computational Intelligence Systems, 2013, 6(5): 862-880.

[17] Goodman R, Thornton M, Strasser S, et al. MICPSO: A method for incorporating dependencies into discrete particle swarm optimization[C]. IEEE Symposium Series on Computational Intelligence. Athens, 2016: 1-8.

[18] Du Z G, Pan J S, Chu S C, et al. Multi-group discrete symbiotic organisms search applied in traveling salesman problems[J]. Soft Computing, 2022, 26(9): 4363-4373.

[19] 岳嵚, 冯珊. 遗传算法的计算性能的统计分析[J]. 计算机学报, 2009, 32(12): 2389-2392.
(Yue Q, Feng S. The statistical analyses for computational performance of the genetic algorithms[J]. Chinese Journal of Computers, 2009, 32(12): 2389-2392.)

作者简介

孙一凡(1998–), 男, 硕士生, 从事智能优化算法的研究, E-mail: qd_sunyifan@126.com;

张纪会(1969–), 男, 教授, 博士, 从事复杂系统建模、智能优化算法、物流系统工程等研究, E-mail: zhangjihui@qdu.edu.cn.