跳跃海豚群算法

王艳娇†, 史新梦

(东北电力大学 电气工程学院, 吉林 吉林 132000)

摘要: 针对海豚群算法收敛速度慢、容易陷入局部最优等缺陷, 提出一种跳跃海豚群算法. 增加跳跃步骤, 保留更多优秀解, 加快收敛; 提出声波长度随迭代自适应变化的策略以及含有变异因子的动态位置更新策略, 满足算法不同时期对进化的需求; 并加入早熟收敛机制, 降低陷入局部最优的机率. 最后, 分析了参数对算法性能的影响, 并与四种算法进行对比, 实验表明其在收敛速度、收敛精度以及鲁棒性上优势明显.

关键词: 海豚群算法; 函数优化; 长度自适应策略; 跳跃步骤; 早熟收敛机制

引用格式: 王艳娇, 史新梦. 跳跃海豚群算法. 控制理论与应用, 2019, 36(10): 1755 – 1767

DOI: 10.7641/CTA.2019.80474

Jumping dolphin swarm algorithm

WANG Yan-jiao†, SHI Xin-meng

(School of Information Engineering, Northeast Electric Power University, Jilin Jilin 132000, China)

Abstract: Aiming at the shortages of dolphin swarm algorithm (DSA) with slow convergence speed and easy to fall into local optimum, a jumping dolphin swarm algorithm (JDSA) was proposed. By adding jumping steps, more excellent solutions are retained to accelerate convergence speed. A strategy of making the sound wave-length adaptive to the change of iteration and a strategy of renewing the dynamic position containing the variation factor are proposed to meet the requirements of different periods of the algorithm. The premature convergence mechanism is added to reduce the probability of falling into local optimum. Finally, the influence of parameters on the performance of the algorithm is analyzed and compared with four algorithms. Experiments show that the algorithm has obvious advantages in convergence speed, convergence accuracy and robustness.

Key words: dolphin swarm algorithm; function optimization; length adaptive strategy; jumping steps; premature convergence mechanism

Citation: WANG Yanjiao, SHI Xinmeng. Jumping dolphin swarm algorithm. *Control Theory & Applications*, 2019, 36(10): 1755 – 1767

1 引言

工程实践中存在许多传统数学方法无法解决的复杂优化问题, 学者受自然界的生物行为启发, 设计出了群智能算法进行求解, 如2010年由Yang模拟蝙蝠捕猎行为提出的蝙蝠算法(bat-inspired algorithm, BA)[1]、2010年Yang模拟萤火虫发光特性设计的萤火虫算法(firefly algorithm, FA)[2]、2012年由Pan模拟果蝇觅食行为提出的果蝇优化算法(fruit fly optimization algorithm, FOA)[3]、2014年由Mirjalili等人模拟灰狼群的捕猎行为和领导层级机制提出的灰狼算法(gray wolf optimization algorithm, GWO)[4]、2015年由Meng等人模拟鸟群的群体行为设计出的鸟群算法(bird swarm alogrithm, BSA)[5]等. 这些群智能算法具有全局搜索能力强和鲁棒性好等特点, 对于解决工程上的复杂优化问题具有较强的实用性.

在此背景下, 2016年伍天琪等人模拟海豚捕食行为提出了海豚群算法(dolphin swarm algorithm, DSA)[6]. 选取 10 个典型测试函数对粒子群算法[7]、遗传算法[8]、人工蜂群算法[9]进行实验, 结果表明DSA算法无论在收敛速度还是收敛精度方面较其他算法都有

收稿日期: 2018−06−27; 录用日期: 2019−02−27.

†通信作者. E-mail: wangyanjiao1028@126.com; Tel.: +86 432-64806383.

本文责任编委: 侯忠生.

国家自然科学基金项目(61501107), 吉林省教育厅“十三五”科学技术研究(吉教科合字[2016]第95号), 吉林市科技创新发展计划项目(201750219)资助.

Supported by the National Natural Science Foundation of China (61501107), the Research on Science and Technology in the Thirteenth Five-Year Plan of the Education Department of Jilin Province ([2016] Ninety-Fifth) and the Jilin City Science and Technology Innovation Development Project (201750219).

明显优势, 然而DSA算法也存在一定的缺陷, 如算法局部探索能力不强、鲁棒性和种群多样性差等. 为此, 2018年李志鹏等人提出了混沌海豚群算法(chaotic dolphin swarm algorithm, CDSA)[10], 该算法的初始种群由混沌序列生成, 在捕猎阶段后加入动态分群策略, 对适应度值不同的个体进行不同的更新策略, 并且加入了早熟优化机制, 结果表明, CDSA算法在低维多模态、低维单模态测试函数上收敛速度和收敛精度都有明显的提高, 但仍存在搜索后期收敛速度缓慢、处理高维单模态函数、高维多模态函数等复杂问题时, 容易陷入局部最优等缺陷.

为进一步提升DSA算法的收敛速度和收敛精度, 本文提出了一种跳跃海豚群算法(jumping dolphin swarm algorithm, JDSA), 主要创新点如下:

1) 在搜寻阶段, 为弥补固定长度的声波在算法后期不能进行精细搜索的缺陷, 将固定长度的声波参数改为声波自适应策略, 前期采用较大的声波长度增强算法的全局搜索能力; 随着迭代搜索的进行, 声波长度随之减小, 海豚将在更小的区域进行精细搜索, 平衡了算法的全局搜索和局部开发能力.

2) 在搜寻阶段后加入跳跃步骤, 个体直接跳到邻域最优解, 加快了算法收敛速度.

3) 在捕猎阶段, 个体更新公式过度依赖邻域最优解, 导致种群多样性下降, 算法一旦陷入局部最优, 没有有效的机制跳出. 为此, 对海豚的位置更新公式进行改进, 使海豚前期偏向自身学习, 后期偏向邻域最优解学习, 在保证收敛速度的同时维持了种群的多样性; 在位置更新公式末端加入变异扰动因子, 降低了种群陷入局部最优的可能性.

4) 在捕猎阶段后加入早熟收敛机制, 利用种群适应度方差和个体间的平均距离评判种群是否早熟, 并对早熟种群进行处理, 有效的避免了早熟现象的发生.

最后, 本文分析了不同参数对算法性能的影响并将DSA算法、改进的CDSA算法、JDSA算法以及其余4种目前性能较优的算法在16个性能各异的标准测试函数上进行对比, 结果表明, JDSA算法在收敛速度、收敛精度以及鲁棒性上都有明显的优势.

2 标准海豚群算法

2.1 标准海豚群算法原理

在生物学上, 海豚通过回声定位[11]搜寻周围区域的食物, 回波强度可以帮助海豚预估猎物的位置、大小. 海豚搜寻到较大猎物后, 将利用不同频率的声波与同伴进行信息交流, 以此来获取更优位置, 随后进入捕猎阶段, 海豚向最优位置移动, 引导个体逐渐进化. 通过模拟上述海豚实际捕食过程提出的DSA算法主要包括搜寻、呼叫、接收、捕猎4个关键阶段, 具体如下.

2.1.1 种群初始化

在优化问题中, 每个海豚个体i代表一个可行解, 表示为$\mathrm{Dol}_i=[x_1\ \ x_2\ \ \cdots\ \ x_D]^{\mathrm{T}}(i=1,2,\cdots,N)$, 其中$N$为种群数目, D为所优化问题的维度, $x_j(j=1,2,\cdots,D)$表示第i个海豚个体在第j维的取值, 按照式(1)随机产生N个海豚个体Dol_i:

$$\mathrm{Dol}_{i,j}=F_j+\mathrm{rand}\times(H_j-F_j),\ j=1,2,\cdots,D, \tag{1}$$

式中: H_j和F_j分别表示第j维变量搜索范围的上限和下限, rand为[0,1]之间的随机数.

2.1.2 搜寻

海豚通过声波搜寻其附近区域. 海豚Dol_i随机向M个方向发射声波, 记海豚Dol_i在时间t搜索到的新位置为X_{ijt}, 可用式(2)表示:

$$X_{ijt}=\mathrm{Dol}_i+V_j\times t, \tag{2}$$

式中: $t=1,2,3,\cdots,T_1$, $V_j=[v_1\ \ v_2\ \ \cdots\ \ v_D]^{\mathrm{T}}$ $(j=1,2,\cdots,M)$表示海豚在第j个方向发射的一段声波, 满足$\|V_j\|=\mathrm{speed}$, speed为速度常量.

求出在最大搜索时间T_1内海豚Dol_i搜寻到的个体最优解L_i和邻域最优解K_i. 其中L_i满足

$$L_i=\min_{j=1,2,\cdots,M;\ t=1,2,\cdots,T_1}\mathrm{fitness}(X_{ijt}),$$

最大搜索半径$R_1=T_1\times\mathrm{speed}$. K_i代表海豚Dol_i自身与其他相邻个体发现的最优位置, 在本阶段K_i按式(3)进行更新.

$$K_i=\begin{cases}L_i, & \mathrm{fitness}(L_i)<\mathrm{fitness}(K_i),\\ K_i, & \text{其他}.\end{cases} \tag{3}$$

2.1.3 呼叫和接收

呼叫和接收阶段同时进行, 用以更新海豚Dol_i的邻域最优解K_i. 声波传输需要时间, 定义$N\times N$阶传输时间矩阵 TS (transmission time matrix), $\mathrm{TS}_{i,j}$表示声波从海豚Dol_j到海豚Dol_i剩余传播时间, 初始值均为最大传输时间T_2(人为设定), 算法每迭代一次$\mathrm{TS}_{i,j}$减1, 代表声波在一个单位时间内传播. 各次迭代中, 当Dol_j的邻域最优解K_j优于Dol_i的邻域最优解K_i并且$\mathrm{TS}_{i,j}$大于两者之间的声波传播时间$\left\lceil\dfrac{\mathrm{DD}_{i,j}}{A\times\mathrm{speed}}\right\rceil$时, 按式(4)对$\mathrm{TS}_{i,j}$进行更新,

$$\mathrm{TS}_{i,j}=\begin{cases}\left\lceil\dfrac{\mathrm{DD}_{i,j}}{A\times\mathrm{speed}}\right\rceil, & \mathrm{fitness}(K_j)<\mathrm{fitness}(K_i),\\ & \mathrm{TS}_{i,j}>\left\lceil\dfrac{\mathrm{DD}_{i,j}}{A\times\mathrm{speed}}\right\rceil,\\ \mathrm{TS}_{i,j}, & \text{其他},\end{cases} \tag{4}$$

式中: $\mathrm{DD}_{i,j}$表示海豚Dol_i与海豚Dol_j间的距离,

$$\mathrm{DD}_{i,j}=\|\mathrm{Dol}_i-\mathrm{Dol}_j\|,\ i,j=1,2,\cdots,N,\ i\neq j,$$

A是调节声波传播速度的加速度常数.

$\mathrm{TS}_{i,j}$完成更新后, 判断是否满足$\mathrm{TS}_{i,j}=0$, 若满足条件, 代表海豚Dol_j发出的声波已经被海豚Dol_i接收, 此时$\mathrm{TS}_{i,j}$重新赋值为T_2, 选取K_i和K_j中较优者更新K_i, 具体如式(5)所示:

$$K_i=\begin{cases}K_j, & \mathrm{TS}_{i,j}=0,\\ & \mathrm{fitness}(K_j)<\mathrm{fitness}(K_i),\\ K_i, & \text{其他}.\end{cases}\tag{5}$$

2.1.4 捕猎

根据海豚Dol_i, L_i, K_i三者之间的位置关系和R_1的大小(其中: Dol_i与K_i间以及K_i与L_i之间的距离分别为$\mathrm{DK}_i=\|\mathrm{Dol}_i-K_i\|$和$\mathrm{DKL}_i=\|L_i-K_i\|$), 分为以下3种情况进行捕猎, 获得海豚的新位置NewDol_i, 将其与Dol_i的邻域最优解K_i进行比较, 如果

$$\mathrm{fitness}(\mathrm{NewDol}_i)<\mathrm{fitness}(K_i),$$

则更新K_i, 即$K_i=\mathrm{NewDol}_i$.

1) $\mathrm{DK}_i<R_1$, 如图1所示, 表示海豚Dol_i的邻域最优解K_i在搜索范围之内, 此时认为$K_i=L_i$, 海豚Dol_i获得的新位置如式(6)所示:

$$\begin{cases}\mathrm{NewDol}_i=K_i+\dfrac{\mathrm{Dol}_i-K_i}{\mathrm{DK}_i}\times R_2,\\ R_2=(1-\dfrac{2}{e})\times\mathrm{DK}_i,\end{cases}\tag{6}$$

式中$e>2$为半径衰减系数, 通常取3或4.

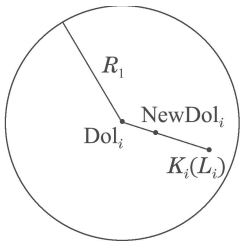


图 1 海豚Dol_i的邻域最优解K_i在搜索范围之内

Fig. 1 The optimal neighborhood solution K_i of dolphin Dol_i is within the search range

2) $\mathrm{DK}_i>R_1$且$\mathrm{DK}_i>\mathrm{DKL}_i$, 如图2所示, 表示海豚$\mathrm{Dol}_i$的邻域最优解$K_i$在搜索范围之外, 且$L_i$比$\mathrm{Dol}_i$更接近$K_i$, 海豚$\mathrm{Dol}_i$获得的新位置如式(7)所示.

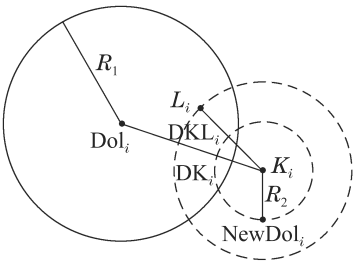


图 2 海豚Dol_i的邻域最优解K_i在搜索范围之外, 且L_i距离K_i较近

Fig. 2 The optimal neighborhood solution K_i of dolphin Dol_i is outside the search area and L_i is close to K_i

$$\begin{cases}\mathrm{NewDol}_i=K_i+\dfrac{\mathrm{random}}{\|\mathrm{random}\|}\times R_2,\\ R_2=\Big(1-\dfrac{\dfrac{\mathrm{DK}_i}{\mathrm{fitness}(K_i)}+\dfrac{\mathrm{DK}_i-\mathrm{DKL}_i}{\mathrm{fitness}(L_i)}}{e\times\mathrm{DK}_i\times\dfrac{1}{\mathrm{fitness}(K_i)}}\Big)\times\mathrm{DK}_i.\end{cases}\tag{7}$$

3) $\mathrm{DK}_i>R_1$且$\mathrm{DK}_i<\mathrm{DKL}_i$, 如图3所示, 表示海豚$\mathrm{Dol}_i$的邻域最优解$K_i$在搜索范围之外, 且$\mathrm{Dol}_i$比$L_i$更接近$K_i$, 海豚$\mathrm{Dol}_i$获得的新位置如式(8)所示:

$$\begin{cases}\mathrm{NewDol}_i=K_i+\dfrac{\mathrm{random}}{\|\mathrm{random}\|}\times R_2,\\ R_2=\Big(1-\dfrac{\dfrac{\mathrm{DK}_i}{\mathrm{fitness}(K_i)}-\dfrac{\mathrm{DKL}_i-\mathrm{DK}_i}{\mathrm{fitness}(L_i)}}{e\times\mathrm{DK}_i\times\dfrac{1}{\mathrm{fitness}(K_i)}}\Big)\times\mathrm{DK}_i.\end{cases}\tag{8}$$

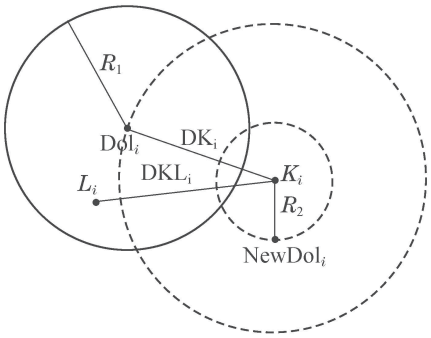


图 3 海豚Dol_i的邻域最优解K_i在搜索范围之外且Dol_i距K_i较近

Fig. 3 The optimal neighborhood solution K_i of dolphin Dol_i is outside the search area and Dol_i is close to K_i

3 跳跃海豚群算法

本文主要从4个方面对基本DSA算法进行改进.

3.1 声波长度自适应策略

在搜寻阶段, 海豚通过发射声波搜索周围区域,可描述为: 新位置=基向量+声波×搜索时间, 其中声波方向随机, 大小为一个常数, 搜索时间是一个正整数常数. 声波及搜索时间对新位置更新的影响如下:1) 搜索时间对位置更新的影响. 图4给出了搜索时间$T_1=2$、声波个数$M=3$时的搜索场景, 如果海豚在$T_1=1$时没有搜索到更优位置, 表明该方向搜索到更优解的可能性较低, 继续增加搜索时间可能造成无效搜索, 进而降低搜索效率. 可见固定的搜索时间较为合理; 2) 固定长度声波对位置更新的影响. 由式(2)可见, 声波长度决定基向量自身的搜索步长, 由于算法在不同进化时期对个体的进化策略要求不同, 固定长度的搜索步长已经不能满足算法的要求. 综上可知, 搜索时间对位置更新影响不大, 而声波长度对位置更新的影响较大, 故JDSA算法主要对声波长度进行改进.

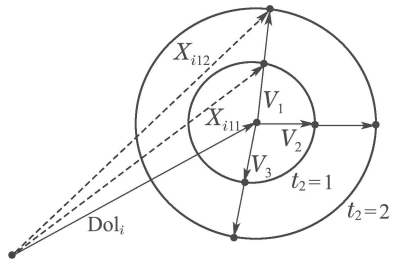


图 4 搜索时间$T_1=2$、声波个数$M=3$时的搜索场景

Fig. 4 Search scene with the search time $T_1=2$ and the number of sound waves $M=3$

进化时期对DSA算法的影响如下: 在进化前期, 每个海豚个体相距较远, 为增强算法的全局搜索能力, 可采用较大的声波长度, 使海豚可以在较大的区域进行搜索; 在搜索后期, 海豚个体相距较近, 都处于较优位置附近, 此时如果搜索范围过大, 可能导致个体跳过最优解, 造成无效搜索. 故声波长度应随迭代次数的增加逐渐减小. 基于上述分析, 提出如式(9)所示的声波长度自适应策略:

$$\begin{cases} \|V(r)\| = V_{\min} + \|V(r-1)\| \times \alpha, \\ \alpha = \exp(-\beta \times (\dfrac{r}{R_{\max}})^s), \end{cases} \tag{9}$$

式中: $V_{\min}$是声波长度的最小值, 由上述分析可知, 个体间的距离影响声波长度的取值, 故$V_{\min}$的取值是通过多次实验后, 分析算法后期个体间的距离确定的, r为当前迭代次数, $R_{\max}$为最大迭代次数, 声波长度的初始值$\|V(1)\|$设为$\max\limits_{j=1,2,\cdots,D}\dfrac{(H_j-F_j)}{4}$, α为非线性时变函数, β和s取值均为常数(β取值范围为[10, 100], s取值范围为[1, 10]), 可根据具体问题人为设定, 如无特殊要求, 一般当β和s分别取30和5时, 即可取得较优效果(具体可参见第3.1部分实验).

DSA算法原有的搜索式(2)改为如式(10)所示新的搜索方式:

$$X_{ijt} = \mathrm{Dol}_i + V_j(r) \times t. \tag{10}$$

由式(9)–(10)可见, 随着迭代次数的增加, 声波长度$\|V\|$以非线性的形式逐渐减小, 且最大搜索半径也会随之减小, 使JDSA算法可以在更小的区域进行精细搜索, 与理论分析相符.

3.2 跳跃步骤

搜寻阶段完成后, 海豚Dol_i将得到邻域最优解K_i, 经过呼叫和接收两个阶段的信息交流, 若海豚Dol_j的邻域最优解K_j优于K_i并满足式(5)的情况下, K_i将被替换为K_j, 原来的邻域最优解K_i被舍弃, 否则, 保持K_i不变. 显然, 这种方式利于种群迅速聚集, 即DSA算法具有较快的收敛速度, 但是也极有可能造成多个海豚个体具有相同邻域最优解, 而后续搜索是基于该邻域最优解的, 若处理多峰函数问题, 算法极有可能迅速收敛到某个局部最优值, 而被舍弃的K_i很有可能代表了其他峰的局部信息, 对算法避免陷入局部最优有一定的指导作用.

综上所述, 被舍弃的K_i含有海豚Dol_i原来的位置信息, 对维持种群的多样性有一定贡献, 本节在搜寻阶段后加入跳跃步骤, 用以更新海豚个体Dol_i的位置, 具体更新方式如式(11)所示:

$$\mathrm{Dol}_i = K_i. \tag{11}$$

式(11)表示在搜寻阶段得到海豚个体Dol_i的邻域最优解后, 海豚个体Dol_i直接跳跃到邻域最优解K_i的位置, 结合第2.3节的海豚位置更新公式可见, 被遗漏掉的邻域最优解K_i参与到海豚的位置更新中, 在保证算法收敛速度的同时, 对种群的多样性也做出了一定的贡献.

3.3 捕猎阶段位置更新方式的改进

由第2.1.4节捕猎阶段可见, 海豚的位置更新方式按R_2, DK_i和DKL_i的关系分为式(6)(7)(8)3种, 各位置更新方式对算法的影响如下: 式(6)中, 海豚以自身的邻域最优解K_i为基向量, 学习因子$\dfrac{\mathrm{Dol}_i-K_i}{\mathrm{DK}_i}$为自身与邻域最优解之间差向量的单位向量, 海豚从原来的位置向邻域最优解靠近, 显然, 这种方式可以使算法获得较快的收敛速度; 而式(7)–(8)是在基向量K_i的基础上, 加了一个随机单位向量$\dfrac{\mathrm{random}}{\|\mathrm{random}\|}$, 显然, 这种方式利于增加种群多样性. 由于式(6)过度向最优解学习, 导致通过式(6)进行捕猎后的海豚适应度值远优于其他海豚个体, 其他海豚将迅速向其靠拢, 使得由式(7)–(8)得到的海豚位置也会被其代替, 种群多样性骤减, 如果该位置是某一局部最优点, 则DSA算法将陷入局部最优, 出现早熟现象. 由此可知, 式(7)–(8)虽在一定程度上增加了种群多样性, 但由于式(6)过度向最优解学习, 个体趋于一致, 种群多样性骤减, K_i容易陷入局部最优. 综上说明, 式(6)无法有效维持种群多样性是DSA算法陷入局部最优的一个重要因素.

针对上述问题, 提出如式(12)所示的含有变异因子的新型海豚位置更新策略代替式(6).

$$\begin{aligned}\mathrm{NewDol}_i = (1-\frac{r}{rR_{\max}})\times\mathrm{Dol}_i + (\frac{r}{rR_{\max}})\times K_i + \\ c(r_1-0.5)\times(r_2\times K_i-\mathrm{Dol}_i),\end{aligned} \tag{12}$$

式中: r_1和r_2是服从$U(0,1)$分布的随机数, r为当前迭代次数, $R_{\max}$为最大迭代次数, c为常数, $c(r_1-0.5)\times(r_2\times K_i-\mathrm{Dol}_i)$为变异扰动部分.

由式(12)可见, 在进化前期, 海豚位置更新公式主要由$(1-\dfrac{r}{R_{\max}})\times\mathrm{Dol}_i + c(r_1-0.5)\times(r_2\times K_i-\mathrm{Dol}_i)$决定, 与原来的搜索模式(6)相比, 基向量的改变

以及变异扰动因子的增加使个体不再紧紧围绕K_i搜索, 变异扰动围绕个体自身进行, 维持了种群的多样性; 随着迭代的进行, Dol_i的权重逐渐减小, K_i的权重逐渐增加, 加快了算法的收敛速度, 而变异扰动中心逐渐转移到K_i上, 又可以防止DSA算法陷入局部最优. 此时, 位置更新的基向量, 由单一的K_i变为个体自身与其邻域最优解的组合, 这样既保证了算法的收敛速度, 又维持了种群的多样性.

3.4　早熟收敛机制

通过上述3部分的改进, 在处理单模态函数、低维多模态函数以及简单的高维多模态函数时, 算法的收敛速度和收敛精度都有很大提升, 但在优化较为复杂的有多个局部最优峰值的高维多模态函数时, 仍然存在陷入局部最优的风险, 为此, 在算法的最后阶段引入早熟收敛机制[12]. 该机制主要分为早熟判断和早熟处理两部分.

3.4.1　早熟判断

随着种群的进化, 个体越来越集中, 即种群适应度方差σ^2和个体平均距离$\bar{D}$将越来越小, 当两者小于相应的临界值时, 认为算法出现早熟, 即$\sigma^2<\mu$或$\bar{D}<\lambda$. 其中$\bar{D}$和σ^2的计算方式分别如式(13)–(14)所示:

$$\bar{D}=\frac{\sum\limits_{i=1}^{N}\sqrt{\sum\limits_{j=1}^{D}(\mathrm{Dol}_{ij}-\overline{\mathrm{Dol}_j})^2}}{N\times L},\tag{13}$$

式中: L为搜索空间的最大长度, Dol_{ij}表示第i个海豚的第j维, $\overline{\mathrm{Dol}_j}$表示所有海豚个体的第$j$维的均值, $\bar{D}$值越小, 表示海豚个体聚集程度越高.

$$\sigma^2=\sum_{i=1}^{N}\frac{\mathrm{fitness}(\mathrm{Dol}_i)-\overline{\mathrm{fitness}}}{\mathrm{fitness}},\tag{14}$$

式中: $\mathrm{fitness}(\mathrm{Dol}_i)$表示每个海豚个体的适应度值, $\overline{\mathrm{fitness}}$表示海豚群的平均适应度值, fitness为归一化因子, 具体计算如式(15).

$$\mathrm{fitness}=\begin{cases}\max\left|\mathrm{fitness}(\mathrm{Dol}_i)-\overline{\mathrm{fitness}}\right|,\\ \quad\max\left|\mathrm{fitness}(\mathrm{Dol}_i)-\overline{\mathrm{fitness}}\right|>1,\\ 1,\ \text{其他}.\end{cases}\tag{15}$$

由式(14)–(15)可见, σ^2反映了全体海豚个体的离散程度, σ^2越小, 表示种群越趋于收敛, 若收敛的位置不是全局最优解, 此时算法将出现早熟.

3.4.2　早熟处理

当满足早熟条件时, 对其进行早熟处理, 由于个体处于较聚集的状态, 彼此之间差异性小, 故采用式(1)随机产生一个解来替换种群中的任意一个个体, 否则跳出早熟收敛机制.

对早熟处理分析发现, 对个体进行初始化处理可以增加种群的多样性, 帮助算法跳出早熟状态. 虽然当前种群出现早熟, 但种群聚集在较优位置的概率较大, 若抛弃过多个体并对其初始化, 可能导致种群向次于当前位置的方向进化, 造成资源浪费. 故本文随机选择一个个体抛弃并对其进行初始化操作, 这样不仅提高了种群的多样性, 帮助算法跳出早熟状态, 还可以避免资源浪费.

3.4.3　算法执行步骤

综合以上4部分的改进, 本文提出的跳跃海豚群算法(JDSA)的步骤如下:

步骤 1　参数初始化: 给定种群规模N、优化问题的维度D、搜索问题的上下界H和F、函数最大调用次数Max-FEs、传输时间矩阵TS等;

步骤 2　根据式(1)随机生成初始化种群, 并计算个体适应度值;

步骤 3　利用式(10)产生海豚个体的新位置, 并更新个体最优解L_i和邻域最优解K_i, 根据式(11)执行跳跃步骤, 完成海豚个体Dol_i的位置更新;

步骤 4　按式(4)对传输时间矩阵TS进行更新, TS的每一个元素减1, 按式(5)完成K_i的更新;

步骤 5　根据DK_i, DKL_i, R_1的大小关系, 选择式(12)(7)(8)之一对海豚个体Dol_i的位置进行更新, 并完成邻域最优解K_i的更新;

步骤 6　执行早熟收敛机制;

步骤 7　判断是否满足终止条件, 若满足, 输出当前最优解, 算法终止, 否则返回步骤3, 继续搜索.

3.5　复杂度分析

群体智能算法的时间复杂度正比于种群规模N和个体维度D[13–14], DSA和JDSA各部分的时间复杂度如表1所示.

表 1　算法的时间复杂度分析

Table 1　Time complexity analysis of the algorithm

DSA算法步骤	复杂度	JDSA算法步骤	复杂度
种群初始化	$N\times D$	种群初始化	$N\times D$
计算适应度值	N	计算适应度值	N
搜寻新位置	$N\times T_1\times M$	搜寻新位置	$N\times T_1\times M+1$
计算适应度值	$N\times T_1\times M$	计算适应度值	$N\times T_1\times M$
更新K和L	N	更新K和L	N
更新TS	$2\times N\times N$	更新TS	$2\times N\times N$
更新K	N	更新K	N
更新海豚位置	$N\times D$	更新海豚位置	$N\times D$
计算适应度值	N	计算适应度值	N
更新K	N	更新K	N
		早熟收敛	3

由表1可以得出, DSA和JDSA寻优一次的时间复杂度均为$\mathrm{O}(N^2)$. JDSA主要增加了声波长度自适应策

略、跳跃步骤和早熟收敛机制并对捕猎阶段海豚位置更新方式进行了改进, 其中跳跃步骤和捕猎阶段的改进并不改变算法的时间复杂度, 而声波长度自适应策略和早熟收敛机制的时间复杂度均为一个常数, 故两算法的时间复杂度是相同的.

4 实验仿真与结果分析

为了充分验证本文提出的跳跃海豚群算法在求解优化问题上的有效性和先进性, 本节进行了一系列的实验, 所有试验均在CPU: Intel(R) Core(TM) i5-7300 HQ、8 G内存、2.50 GHz主频的计算机上实现, 程序采用MATLAB R14b语言实现. 实验分成2部分, 第一, 考察各个参数对JDSA算法的影响; 第二, 将JDSA算法与其他优化算法进行比较.

本文选取CEC2005中的16个基准测试函数进行测试, 其中f_1–f_3, f_9, f_{11}, f_{12}为高维单模态函数; f_4–f_8, f_{10}为高维多模态函数; f_{13}, f_{16}为低维单模态函数; f_{14}, f_{15}为低维多模态函数. 其中除f_9维度为32, 其余高维测试函数均为30维, 低维测试函数均为2维, 除函数f_{16}最优值为-1之外, 其他测试函数的最优值均为0.

4.1 参数对算法性能影响分析

本文提出的JDSA算法, 在DSA算法的基础上, 增加了参数β和s, 为考察这两个参数对算法性能的影响, 进行如下实验.

4.1.1 参数β的性能影响测试

为考察参数β对JDSA算法性能的影响, 保证JDSA算法其他参数不变, 测试β取值不同时, JDSA算法在16个测试函数上达到预设函数评价次数时收敛精度的均值. 除β外, JDSA算法的其他参数设置如下: $s=5$、种群数目为50、高维函数f_1–f_{12}的最大函数评价次数20000次、低维函数f_{13}–f_{16}的最大函数评价次数为2000次. 为避免算法单次运行的随机性对算法评价带来的不良影响, 不同取值的β对每个测试函数均独立运行30次, 其平均值的统计结果如表2所示.

表 2 参数β对算法的影响

Table 2 Influence of parameters β on the algorithm

函数		β的取值				
		10	30	50	70	90
f_1	Sphere	1.7199e−122	**2.5541e−127**	1.4729e−114	7.1141e−118	1.3429e−102
f_2	Schwefel 2.22	3.4177e−59	**1.7884e−63**	1.8346e−58	5.0094e−62	2.2859e−62
f_3	Schwefel 1.2	3.8086e−121	**1.4151e−127**	1.1527e−128	3.9482e−119	7.7295e−119
f_4	Rastrigin	0	0	0	0	0
f_5	Griwank	0	0	0	0	0
f_6	Ackley	8.8818e−16	8.8818e−16	8.8818e−16	8.8818e−16	8.8818e−16
f_7	Alpine	4.0332e−59	**5.0099e−66**	2.9495e−58	3.3902e−63	3.6410e−61
f_8	Salomon	0	0	0	0	0
f_9	Powell	1.2922e−119	**1.5834e−128**	3.4540e−128	1.2791e−124	2.9898e−126
f_{10}	Weierstrass	0	0	0	0	0
f_{11}	Zakharov	3.8392e−123	**9.4341e−130**	1.9874e−123	3.1757e−125	1.5801e−127
f_{12}	Elliptic	**1.3480e−124**	9.4175e−124	5.5526e−110	1.2871e−117	3.7264e−104
f_{13}	Matyas	**3.5583e−13**	6.2377e−12	2.4162e−12	14238e−12	2.5487e−12
f_{14}	Schaffer	6.3016e−04	**5.1928e−05**	6.2472e−04	5.0855e−04	8.8800e−04
f_{15}	Boachevsky3	**2.4645e−10**	1.1369e−09	6.9913e−06	5.5679e−06	2.4913e−06
f_{16}	Easom	**−0.7010**	−0.6147	−0.4817	−0.3752	−0.3825

由表2可以看出, 对同一测试函数, β取值不同, 优化结果不同. 当$\beta=30$时, 16个测试函数中有12个测试函数收敛精度最高, 而β取值为10时, 有8个测试函数收敛精度最高, 虽在函数Elliptic, Matyas, Boachevsky3, Easom上的收敛精度达到最高, 但β取值为30时也达到了次优值. 当β取值为50, 70, 90时, 各函数均未达到最高. 综上比较, β取值为30时优化效果较好.

4.1.2 参数s的性能影响测试

与第4.1.1部分测试方式类似, 当β取为30、其他参数设置同4.1.1, 改变s值, 在16个函数上进行30次独立实验, 统计JDSA算法在第4.1.1部分的预设函数评价次数内所得最优解的平均值, 具体实验结果如表3所示.

分析表3可知, 无论s取上述何值, 在函数Rastrigin, Griwank, Salomon, Weierstrass上均可达到理论最优值. 对于其他函数, 随着s值的增加, 收敛精度随之提高, 当s取值为5时收敛精度达到最高, 但继续增加s值收敛精度反而降低. 综上所述, s取5即可取得较为满意的函数优化效果.

表 3 参数s对算法的影响

Table 3 Influence of parameters s on the algorithm

	函 数	s的取值				
		1	3	5	7	9
f_1	Sphere	4.7876e−120	2.5136e−120	**2.3689e−125**	1.1396e−121	3.7845e−123
f_2	Schwefel 2.22	2.2366e−60	1.8594e−63	**1.2120e−65**	1.7142e−61	1.8078e−60
f_3	Schwefel 1.2	9.3693e−118	1.6412e−123	**6.5208e−128**	1.8692e−124	5.0183e−111
f_4	Rastrigin	0	0	0	0	0
f_5	Griwank	0	0	0	0	0
f_6	Ackley	8.8818e−16	8.8818e−16	8.8818e−16	8.8818e−16	8.8818e−16
f_7	Alpine	5.1699e−64	3.2453e−65	**2.3573e−65**	1.3960e−62	2.1752e−61
f_8	Salomon	0	0	0	0	0
f_9	Powell	2.6476e−113	1.7065e−128	**6.2377e−130**	1.0981e−128	8.2277e−122
f_{10}	Weierstrass	0	0	0	0	0
f_{11}	Zakharov	9.1640e−124	1.4869e−127	**1.6069e−130**	2.5739e−128	3.4217e−121
f_{12}	Elliptic	1.7518e−111	3.3848e−114	**2.5226e−118**	3.3987e−115	1.1636e−115
f_{13}	Matyas	2.3976e−11	4.7671e−12	**8.3487e−15**	5.5183e−13	2.7575e−13
f_{14}	Schaffer	1.4000e−03	9.6280e−04	**9.8712e−05**	9.4096e−04	1.6741e−04
f_{15}	Boachevsky3	2.2047e−04	8.8199e−05	**1.9949e−06**	4.1629e−06	7.5605e−05
f_{16}	Easom	−0.3059	−0.5440	**−0.8382**	−0.7018	−0.7916

4.2 与其他算法的比较

为了充分验证本文提出的JDSA算法的优势, 将JDSA算法与基本DSA算法、改进的CDSA算法及目前优化效果较好的其他4种算法在收敛精度、收敛速度、鲁棒性以及时间复杂度等方面进行比较, 对比的4种算法主要包括: 2014年由Xiangtao Li提出的Modified differential evolution with self-adaptive parameters method—MDE[15]、2015年由Ismail Babaoglu提出的Artificial bee colony algorithm with distribution-based update rule—distABC[16]、2016年由Jianhua Xiao提出的An improved gravitational search algorithm for green partner selection in virtual enterprises—IGSA/PSO[17]以及2017年由Wei Sun提出的All dimension neighborhood based particle swarm optimization with randomly selected neighbors—ADN–RSN–PSO[17].

4.2.1 收敛精度的比较

实验过程中对比算法的参数均设置为各自对应文献中的参数, 表4给出了各算法的实验参数设置.

表 4 各算法参数设置

Table 4 Parameter setting of each algorithm

算 法	参 数
MDE	$CR = 0.4$, $F =$随机数rand
IGSA/PSO	$G_0 =$100, $\alpha =$20
distABC	limit= $\dfrac{\text{种群数目} \times \text{维度}}{2}$
ADN–RSN–PSO	$w = 0.7298$, $c_1 = c_2 = 2.05$
DSA	$T_1 =$3, $T_2 =$1000, speed= 1, $A = 5$, $M = 3$, $e = 4$
CDSA	$T_1 =$3, $T_2 =$1000, speed= 1, $A = 5$, $M = 3$, $e = 4$, $\alpha = 0.5$, $\beta = 0.001$
JDSA	$T_1 = 3$, $T_2 = 1000$, speed=1, $A = 5$, $M = 3$, $e = 4$, $c = 2$ $r_1 =$ rand , $r_2 =$rand, $\mu =$0.5, $\lambda =$0.001, $V_{\min} =$0.000 2

为保证本次对比试验的公平性, 各算法的种群数目均为50, 最大函数评价次数均为80000. 为避免算法单次运行的随机性带来的不良影响, 各算法对每个测试函数分别独立运行30次, 并统计30次实验中的最小值、平均值、最大值、方差、算法在各测试函数中所得平均值的排序, 统计结果如表5所示. 为了证明JDSA算法的有效性, 本文采用Friedman和Wilcoxon两种非参数检验对各算法的性能作进一步分析. 实验在SPSS 19.0上进行, 计算各算法的Friedman统计量秩均值, 对于最小化问题, 秩均值越小表明算法越好; 将JDSA算法分别与其他6种算法两两进行Wilcoxon符号秩检验, 利用SPSS计算出各配对算法对应的概率值P, 如果P小于给定的显著性水平θ(一般取值为0.05), 则认为两配对算法有显著性差异, 检测结果如表6所示.

表 5 收敛精度比较

Table 5 Convergence accuracy comparison

函 数	统计量	MDE	IGSA/PSO	distABC	ADN–RSN–PSO	DSA	CDSA	JDSA
f_1 Sphere	最小值	2.1695e−13	5.0868e−14	1.4864e−12	4.3911e−65	2.0526e−11	2.7734e−06	**0**
	平均值	1.8134e−12	9.4093e−09	2.3970e−11	5.6141e−07	3.1843e−10	1.1120e−02	**0**
	最大值	8.1261e−12	1.6478e−07	1.8311e−10	1.6842e−05	1.981e−09	8.1502e−02	**0**
	方差	4.4107e−24	9.6054e−16	1.8716e−21	9.1402e−12	2.0739e−19	3.4200e−04	**0**
	排序	2	5	3	6	4	7	1
f_2 Schwefel2.2	最小值	7.4743e−07	3.0582e−04	7.6136e−10	1.0125e−29	8.0813e+02	2.3340e−01	**5.4676e−234**
	平均值	5.4778e−06	7.6349e−03	2.4966e−09	1.1409e−03	1.1109e+03	7.6080e−01	**1.8587e−215**
	最大值	1.9814e−05	5.7897e−02	1.1455e−08	1.8503e−02	1.4709e+03	1.8336e+00	**5.5759e−214**
	方差	2.0213e−11	1.4069e−04	4.6712e−18	1.3706e−05	2.4590e+04	1.4380e−01	**0**
	排序	3	5	2	4	7	6	1
f_3 Schwefel 1.2	最小值	6.8651e+03	2.2755e−01	4.9214e+03	7.6811e−57	5.0792e+00	4.8080e−01	**0**
	平均值	1.3902e+04	3.8629e+00	1.7987e+03	1.7608e−13	2.3724e+00	1.3221e+00	**0**
	最大值	2.0883e+04	3.0340e+01	3.3601e+04	5.2792e−12	6.8534e+00	2.8654e+00	**0**
	方差	1.4328e+07	3.0183e+01	4.2718e+07	8.9798e−25	2.0511e+00	3.1070e−01	**0**
	排序	7	5	6	2	4	3	1
f_4 Rastrigin	最小值	3.0444e+01	2.6864e+01	8.9208e+01	**0**	9.6511e+01	2.9863+01	**0**
	平均值	4.7238e+01	4.5669e+01	1.2093e+02	5.5305e−02	1.7810e+02	2.9877e+01	**0**
	最大值	6.2260e+01	7.1637e+01	1.4936e+02	8.3615e−01	2.6068e+02	2.9894e+01	**0**
	方差	6.8400e+01	1.4139e+02	2.2467e+02	3.9363e−02	1.4117e+03	6.9607e−05	**0**
	排序	5	4	6	2	7	3	1
f_5 Griwank	最小值	3.4417e−15	7.9393e−03	1.9106e−11	**0**	2.9104e−07	9.9216e−05	**0**
	平均值	2.9322e−13	8.9142e−01	2.6338e−07	1.7618e−11	1.2888e−02	4.4798e−03	**0**
	最大值	2.6972e−12	2.5679e+00	6.7439e−06	5.2839e−10	3.4392e−02	1.4300e−02	**0**
	方差	3.2413e−25	3.9929e−01	1.4617e−12	8.996e−21	1.0492e−04	1.7005e−05	**0**
	排序	2	7	4	3	6	5	1
f_6 Ackley	最小值	2.0455e+01	1.7503e−06	1.4085e−07	1.3603e−06	1.6462e+00	1.4091e+00	**8.8818e−16**
	平均值	2.0576e+01	2.9264e−05	9.6402e−07	7.9779e−03	9.1263e+00	2.1410e+00	**8.8818e−16**
	最大值	2.0645e+01	2.2219e−04	2.7938e−06	1.2497e−01	1.9348e+01	2.5448e+00	**8.8818e−16**
	方差	2.48745e−03	3.0938e−09	9.334e−14	5.6250e−04	6.4999e+01	6.7400e+00	**0**
	排序	7	3	2	4	6	5	1
f_7 Alpine	最小值	1.3062e−03	2.6672e−06	1.3053e−06	1.6016e−15	3.0934e+00	1.8200e−02	**4.4907e−238**
	平均值	6.9617e−03	3.0560e−03	2.2338e−04	7.2350e−04	1.0062e +01	5.9890e−01	**8.3764e−220**
	最大值	1.1370e−02	1.0314e−02	1. 0670e−03	1.6255e−02	2.5269e+01	2.3574e+00	**2.5126e−218**
	方差	8.5610e−06	8.4663e−06	8.4582e−08	8.5078e−06	2.2537e+01	7.0050e−01	**0**
	排序	5	4	2	3	7	6	1
f_8 Salomon	最小值	2.0036e−01	4.9987e−01	1.9988e−01	3.7748e−15	6.9987e−01	1.9990e−01	**0**
	平均值	2.8208e−01	9.1864e−01	2.0408e−01	2.5652e−04	1.2432e+00	2.8990e−01	**0**
	最大值	3.0122e−01	1.9999e+00	2.8740e−01	7.4484e−03	1.8999e+00	2.9990e−01	**0**
	方差	1.2091e−03	7.5273e−02	2.6866e−04	1.7851e−06	1.2046e−01	9.0000e−04	**0**
	排序	4	6	3	2	7	5	1
f_9 Powell	最小值	3.8895e−02	5.5815e−03	1.0344e−02	1.5994e−57	1.4161e−03	6.5975e−03	**0**
	平均值	1.2320e−01	2.0928e−01	1.7351e+01	6.0453e−30	3.2781e−03	1.9400e−02	**0**
	最大值	2.4069e−01	1.0374e+00	4.2430e+02	1.8084e−28	5.3440e−03	3.9000e−02	**0**
	方差	2.6478e−03	6.33812e−02	5.7411e+03	1.0535e−57	7.8394e−07	6.0133e−05	**0**
	排序	5	6	7	2	3	4	1

(转下页)

(接上页)

f_{10} Weierstrass	最小值	1.3176e+01	2.3660e+00	1.8633e+01	1.4147e−01	1.0340e+01	1.6047e+01	**0**
	平均值	1.4510e+01	4.8553e+00	1.9966e+01	2.4524e+00	1.4083e+01	1.8108e+01	**0**
	最大值	1.5765e+01	8.2650e+00	2.0773e+01	6.5611e+00	1.9030e+01	1.9500e+01	**0**
	方差	3.9403e−01	1.6715e+00	2.4298e−01	2.8674e+00	3.9562e+00	8.2810e−01	**0**
	排序	5	3	7	2	4	6	1
f_{11} Zakharov	最小值	3.5527e+02	1.4177e−02	4.9931e+02	8.7304e−19	9.3270e−04	8.9763e−04	**0**
	平均值	9.8839e+02	4.9496e+00	1.4292e+03	7.9457e+01	2.1139e−03	4.7525e−03	**0**
	最大值	2.3497e+02	2.2036e+01	2.8557e+03	1.2354e+03	3.2850e−03	1.6611e−02	**0**
	方差	2.5574e+05	3.0039e+01	3.0428e+05	8.1302e+04	3.9141e−07	1.6118e−05	**0**
	排序	6	4	7	5	2	3	1
f_{12} Elliptic	最小值	6.7571e+04	5.8706e+01	8.2189e+04	5.4241e−12	2.3242e+06	6.9283e+02	**0**
	平均值	1.1291e+05	1.3396e+03	1.9245e+05	6.7136e+03	4.4578e+06	6.9115e+03	**0**
	最大值	2.2873e+05	3.9668e+03	3.9940e+05	4.3448e+04	8.4498e+06	2.0031e+04	**0**
	方差	1.6222e+09	1.1591e+06	7.0885e+09	1.8638e+08	1.8993e+12	1.6993e+07	**0**
	排序	5	2	6	3	7	4	1
f_{13} Matyas	最小值	1.5629e−123	3.5276e−21	2.3808e−283	2.3516e−37	3.0893e−122	4.8607e−55	**0**
	平均值	1.2105e−115	2.6321e−19	1.7933e−246	1.8952e−08	5.9885e−116	1.0251e−38	**0**
	最大值	3.4846e−114	1.0228e−18	5.3800e−245	3.9011e−07	7.0809e−115	1.5372e−37	**0**
	方差	3.9053e−229	8.3973e−38	**0**	5.3275e−15	3.0788e−230	8.4614e−76	**0**
	排序	4	6	2	7	3	5	1
f_{14} Schaffer	最小值	**0**	**0**	**0**	5.5084e−05	**0**	**0**	**0**
	平均值	6.4773e−04	5.8869e−03	2.1223e−04	2.4033e−02	3.8864e−03	**0**	**0**
	最大值	9.7159e−03	7.7159e−03	9.0774e−04	7.8661e−02	9.7159e−03	**0**	**0**
	方差	5.8737e−06	1.3325e−05	4.8199e−08	5.6758e−04	2.2656e−05	**0**	**0**
	排序	3	5	2	6	4	1	1
f_{15} Boachevsky3	最小值	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	平均值	**0**	1.2953e−17	**0**	7.8043e−06	**0**	1.8504e−18	**0**
	最大值	**0**	1.1102e−16	**0**	7.9729e−04	**0**	5.5511e−17	**0**
	方差	**0**	7.5668e−34	**0**	3.7512e−08	**0**	9.9292e−35	**0**
	排序	1	3	1	4	1	2	1
f_{16} Easom	最小值	**−1**	**−1**	−0.99991	−0.7509	−0.99015	**−1**	**−1**
	平均值	**−1**	−0.96667	−0.99702	−0.07373	−0.86336	**−1**	**−1**
	最大值	**−1**	−2.575e−162	−0.99157	−1.4963e−36	−0.71718	**−1**	**−1**
	方差	**0**	0.032222	5.8711e−06	0.0290	0.0058124	**0**	**0**
	排序	2	3	4	6	5	1	1

对表5–6中的数据进行分析可以得到如下结论:

1) JDSA算法性能分析.

由表5中JDSA算法在各测试函数上的结果可以发现:

第一, 只有一个全局峰值点的单模态函数包括f_1, f_2, f_3, f_9, f_{11}, f_{12}, f_{13}, f_{16}, 一般用于测试算法的寻优精度. JDSA算法在大多数单模态函数上都达到了理论最优值, 主要原因是JDSA算法改进了声波长度, 增加了算法的局部搜索能力; 而函数f_2并未达到理论最优值, 主要原因是实验给的函数评价次数不够, JDSA算法在函数f_2上收敛速度较其他函数慢, 而导致算法在该函数上收敛速度慢的原因是函数f_2包含一个连续多变量, 导致函数f_2的形状过于陡峭, 而JDSA算法围绕邻域最优解移动, 如果邻域最优解附近的适应度函数的形状过于陡峭, 则捕食阶段更可能移动到随机位置.

第二, 含有多个局部极值点或局部最优点的多模态函数包括f_4, f_5, f_6, f_7, f_8, f_{10}, f_{14}, f_{15}, 通常可用来检验算法的全局搜索能力和局部开发能力. JDSA算法在典型的多模态函数f_4, f_5, f_6, f_8, f_{10}, f_{14}, f_{15}上表现优异, 这是因为JDSA算法的跳跃步骤以及捕猎阶段加入了变异扰动因子, 增加了算法的多样性,

而早熟收敛机制保证了算法可以跳出局部最优, 防止了算法出现早熟现象.

表 6 非参数检验结果

Table 6 Non-parametric test results

Friedman		Wilcoxon	
算法	秩均值	算法	P值
MDE	4.19	MDE	0.001
IGSA/PSO	4.81	IGSA/PSO	0.000
distABC	4.19	distABC	0.001
ADN–RSN–PSO	4.13	ADN–RSN–PSO	0.000
DSA	5.00	DSA	0.001
CDSA	4.14	CDSA	0.001
JDSA	1.28		

综合分析单模态函数和多模态函数的实验结果可知, JDSA算法在单模态和多模态函数搜索中性能相对均衡.

2) 与其他算法的对比.

对于高维测试函数f_1–f_{12}, MDE, IGSA/PSO, distABC和ADN–RSN–PSO均未取得理论最优, 而对于DSA系列的算法, 改进后的CDSA算法结果也并不理想, 而JDSA算法在f_1, f_3, f_4, f_5, f_8, f_9, f_{10}, f_{11}, f_{12}上均可获得理论最优值; 与其他算法相比, JDSA算法在函数f_2, f_7上得到最优解的平均值更接近理论最优值, 并且在30次实验中有一定机会收敛至理论最优值; 在函数f_6上虽然JDSA算法只达到了理论次优值, 但其结果明显优于其他算法. JDSA算法在测试各函数所得到的方差全部为0, 与其他6种算法相比, 所获方差最小, 说明JDSA算法较其他6种算法稳定性更好. 综上所述, JDSA算法在处理高维函数上可获得更优的性能.

对于低维函数来说, JDSA算法均达到了理论最优值, 而CDSA算法在函数f_{14}, f_{16}上可以收敛至理论最优值, 但在函数f_{13}上收敛效果不太理想, 在函数f_{15}上表现也不太稳定. MDE, distABC均可以在函数f_{15}上收敛到理论最优值, 在其他函数上都无法收敛, 而IGSA/PSO算法在任何低维函数上都没有收敛. 由此可知, 在低维函数上, 与其他6种进化算法相比, JDSA算法求解精度也最好.

3) 各算法的显著性分析.

由表6中Friedman检验的结果可以看出, JDSA算法的秩均值最小, 即本文所提算法效果最好; 由Wilcoxon符号秩检验的结果可知, JDSA算法与其他算法比较的P值均小于0.05, 说明JDSA算法与其他算法有显著性差异.

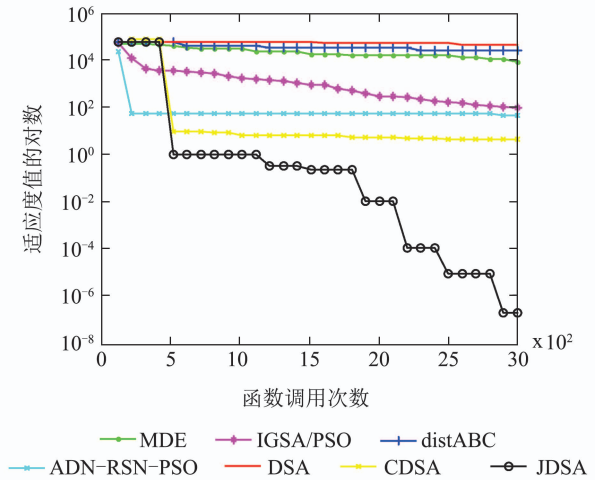
综上说明, JDSA算法在求解精度方面明显优于其他6种算法, 性能更为优异.

4.2.2 收敛速度的比较

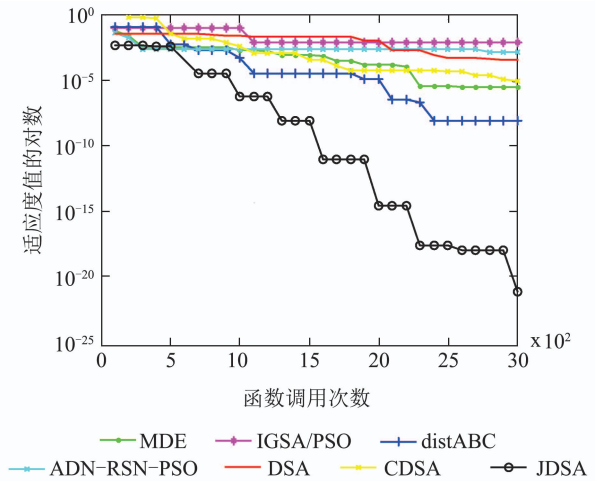
本节考察JDSA算法的收敛速度, 进行如下实验, 各算法采用如表4的参数设置, 均独立运行30次, 分别记录各算法在相同评价次数下获得的最优结果. 其中对于高维函数f_1–f_{12}以及低维函数f_{13}–f_{16}, 分别测试它们在函数评价次数为40000, 50000, 60000以及4000, 6000, 8000时所获最优解的平均值. 测试结果如表7所示.

由表7可知, 在相同的函数评价次数下, 除MDE, IGSA/PSO, CDSA算法在函数f_{16}上与JDSA算法取得的求解精度相同, 对于其他测试函数, 在相同的函数评价次数下, JDSA算法的收敛精度明显优于其他6种算法. 由此可知, 与其余各算法相比, JDSA算法在收敛速度方面具有明显的优势.

此外, 为直观考察各算法的收敛过程, 图5给出了各算法的部分进化过程曲线, 其中横坐标为函数评价次数, 纵坐标为所获得的最优解适应度值的对数值.



(a) f_1收敛曲线图



(b) f_{13}收敛曲线图

图 5 部分函数收敛曲线图

Fig. 5 The graph of partial function convergence

表 7 收敛速度比较

Table 7 Convergence rate comparison

函数	函数评价次数	MDE	IGSA/PSO	distABC	ADN–RSN–PSO	DSA	CDSA	JDSA
f_1 Sphere	40 000	5.2249e−08	2.5655e−06	8.2687e−03	1.1273e−18	8.3161e−03	7.3667e−04	**0**
	50 000	1.6434e−10	2.6158e−07	3.1325e−04	1.0212e−24	6.1702e−05	6.5276e−04	**0**
	60 000	7.2647e−13	2.8889e−08	4.1292e−06	2.0143e−22	5.0713e−08	4.7572e−05	**0**
f_2 Schwefel2.2	40 000	7.6759e−04	3.7400e−02	3.5740e−03	6.0988e−05	2.8527e+00	2.2400e−01	**1.6687e−136**
	50 000	9.5221e−05	7.6217e−03	2.8626e−04	1.5345e−05	1.0212e+00	3.3570e−01	**1.7312e−169**
	60 000	1.7558e−06	1.3722e−03	2.5375e−05	1.5940e−10	3.2850e−01	2.5430e−01	**1.4773e−188**
f_3 Schwefel 1.2	40 000	1.4282e+04	3.8076e+01	1.9455e+04	7.49918e−15	3.4456e+02	1.1965e+00	**0**
	50 000	1.2664e+04	5.5167e+00	1.8570e+04	8.4960e−21	1.6118e+02	1.3504e+00	**0**
	60 000	1.4534e+04	8.7890e−01	1.4632e+04	2.5495e−33	4.6363e+01	7.2360e−01	**0**
f_4 Rastrigin	40 000	5.5446e+01	2.4881e+01	1.0677e+02	6.7248e−07	9.5516e+01	2.9849e+01	**0**
	50 000	4.3156e+01	2.6865e+01	1.1714e+02	1.2996e−08	1.2636e+02	2.9984e+01	**0**
	60 000	3.4729e+01	2.6865e+01	7.4717e+01	7.8296e−11	1.0149e+02	2.9849e+01	**0**
f_5 Griwank	40 000	7.9024e−05	1.7963e+00	1.2840e−01	6.1985e+01	1.2340e−01	1.3985e−03	**0**
	50 000	1.0183e−06	1.3634e+00	2.2787e−03	4.5860e+01	7.8311e−03	3.0460e−03	**0**
	60 000	6.0049e−09	1.0932e+00	5.0625e−05	4.5044e+01	1.2008e−03	2.1376e−03	**0**
f_6 Ackley	40 000	1.7201e−03	2.7322e−03	1.9800e−02	4.6153e−07	1.9034e+00	1.9166e+00	**8.8818e−16**
	50 000	1.3352e−04	4.3944e−04	2.9716e−03	1.4582e−06	1.3404e+00	2.0619e+00	**8.8818e−16**
	60 000	9.1992e−06	4.0438e−05	5.6463e−04	1.4556e−06	2.8141e+00	2.0172e+00	**8.8818e−16**
f_7 Alpine	40 000	1.0310e−02	5.2374e−03	4.5800e−02	1.3981e−07	5.7424e+00	1.9600e−02	**2.2829e−141**
	50 000	6.8523e−03	3.8147e−03	2.0700e−02	3.0233e−08	6.8837e+00	1.9400e−02	**4.9912e−171**
	60 000	4.0367e−03	1.7550e−04	1.2100e−02	2.2117e−08	6.8837e+00	2.3100e−02	**6.4253e−195**
f_8 Salomon	40 000	5.0074e−01	1.2999e+00	6.1470e−01	3.0353e−07	5.3179e+00	2.9990e−01	**0**
	50 000	4.0000e−01	1.0999e+00	4.4600e−01	2.0520e−11	3.9990e−01	2.9990e−01	**0**
	60 000	2.9987e−01	7.9990e−01	3.0360e−01	2.0242e−07	3.9990e−01	2.9990e−01	**0**
f_9 Powell	40 000	1.4890e+00	2.0292e+00	9.4219e−03	1.1187e−33	4.0500e−02	5.7000e−02	**0**
	50 000	3.2960e−01	9.9300e−02	9.2245e+0	1.0102e−35	1.4200e−02	5.2500e−02	**0**
	60 000	2.7660e−01	1.2600e−01	4.2325e+02	1.4769e−41	8.3000e−03	1.5000e−02	**0**
f_{10} Weierstrass	40 000	1.2534e+01	4.3605e+00	1.7857e+01	4.1400e−02	8.6853e+00	1.6731e+01	**0**
	50 000	1.3039e+01	4.0249e+00	2.0246e+01	2.0019e−03	1.1199e+01	1.9326e+01	**0**
	60 000	1.3404e+01	3.0588e+00	1.9532e+01	8.3870e−01	1.3296e+01	1.8910e+01	**0**
f_{11} Zakharov	40 000	5.9396e+02	1.3305e+01	1.4479e+03	1.4934e−21	1.0869e+00	6.3430e−01	**0**
	50 000	3.7975e+02	1.4988e+01	1.1438e+03	2.9748e−12	1.5810e−01	1.7310e−01	**0**
	60 000	4.6587e+02	4.9333e+00	6.1443e+02	6.9658e−10	9.5000e−03	1.0400e−02	**0**
f_{12} Elliptic	40 000	4.6727e+04	3.8.48e+03	1.2226e+05	1.7204e−04	9.4294e+06	1.0897e+04	**0**
	50 000	8.0023e+04	1.2910e+03	8.0505e+04	9.9840e−10	2.8706e+06	4.7305e+03	**0**
	60 000	6.1573e+04	2.3709e+03	1.7998e+05	1.3683e−14	4.2102e+06	5.7781e+03	**0**
f_{13} Matyas	4000	1.4420e−09	2.2849e−15	1.1503e−10	1.5263e−08	1.5887e−07	5.1920e−09	**0**
	6000	8.2767e−13	1.0848e−17	8.3303e−15	1.2288e−10	2.4950e−10	3.2648e−11	**0**
	8000	5.4780e−17	4.2125e−21	4.0735e−17	6.3884e−14	2.9188e−12	8.9800e−14	**0**
f_{14} Schaffer	4000	2.8846e−03	1.3567e−10	3.9020e−04	1.5332e−03	1.7874e−04	7.0146e−08	**0**
	6000	2.4797e−04	8.1046e−15	9.4996e−05	2.8203e−05	4.7124e−05	5.6732e−09	**0**
	8000	3.2898e−04	**0**	1.9289e−04	9.2482e−14	4.4616e−08	6.3222e−13	**0**
f_{15} Boachevsky3	4000	2.2348e−06	6.4243e−12	2.7524e−10	6.7804e−05	6.8096e−04	4.8667e−08	**0**
	6000	2.6129e−10	9.4369e−16	1.0706e−12	3.5742e−09	1.6054e−05	5.6630e−11	**0**
	8000	4.1556e−13	**0**	1.1102e−16	8.6981e−08	8.1928e−09	2.9021e−132	**0**
f_{16} Easom	4000	−0.9990	**−1**	−0.9935	−0.5696	−0.9500	**−1**	**−1**
	6000	**−1**	**−1**	−0.9913	−0.6532	−0.9998	**−1**	**−1**
	8000	**−1**	**−1**	−0.9987	−0.8766	**−1**	**−1**	**−1**

4.2.3 鲁棒性的比较

为评价各算法的鲁棒性, 本文进行了一系列的实验, 所有参数设置均与第4.2.1节一致, 分别记录函数f_1–f_{15}收敛至10^{-10}和函数f_{16}收敛至$10^{-10}-1$的成功率, 成功率越高说明算法的鲁棒性越好, 具体结果如表8所示.

表 8 鲁棒性的比较

Table 8 Robustness comparison

函数		成功率/%						
		MDE	IGSA/PSO	distABC	ADN–RSN–PSO	DSA	CDSA	JDSA
f_1	Sphere	**100**	43.3	93.3	96.7	36.7	0	**100**
f_2	Schwefel 2.22	0	0	3	20	0	0	**100**
f_3	Schwefel 1.2	0	0	0	**100**	0	0	**100**
f_4	Rastrigin	0	0	0	26.7	0	0	**100**
f_5	Griwank	**100**	0	10	96.7	0	0	**100**
f_6	Ackley	0	33.3	**100**	13.3	0	0	**100**
f_7	Alpine	0	0	0	10	0	0	**100**
f_8	Salomon	0	0	0	56.7	0	0	**100**
f_9	Powell	0	0	0	**100**	0	0	**100**
f_{10}	Weierstrass	0	0	0	0	0	0	**100**
f_{11}	Zakharov	0	0	0	6.7	0	0	**100**
f_{12}	Elliptic	0	0	0	6.7	0	0	**100**
f_{13}	Matyas	**100**	**100**	**100**	70	**100**	**100**	**100**
f_{14}	Schaffer	93.3	3	6.7	0	60	**100**	**100**
f_{15}	Boachevsky3	**100**	**100**	**100**	16.7	**100**	**100**	**100**
f_{16}	Easom	**100**	3.3	0	0	0	**100**	**100**

由表8可知, 对所测各函数, JDSA算法独立运行30次成功率都达到了100%, 说明JDSA算法具有较好的鲁棒性, 相较于JDSA算法, 其他6种算法成功率较低, 甚至在复杂的多模态函数Weierstrss上成功率为0. 由此可见, 与其他算法相比, 本文所提JDSA算法的鲁棒性更佳.

4.2.4 时间复杂度的比较

本节考察JDSA的时间复杂度, 进行如下实验,各算法的参数设置如表3所示, 均独立运行30次, 分别记录各算法在相同评价次数下每次所用的平均时间. 其中高维函数f_1–f_{12}评价次数为40000、低维函数f_{13}–f_{16}评价次数为4000, 测试结果如表9所示.

表 9 时间复杂度的比较

Table 9 Time complexity comparison

函数		平均时间/s						
		MDE	IGSA/PSO	distABC	ADN–RSN–PSO	DSA	CDSA	JDSA
f_1	Sphere	0.75	5.98	1.03	0.85	4.69	4.16	4.87
f_2	Schwefel 2.22	0.94	6.18	1.17	0.96	4.91	4.11	5.07
f_3	Schwefel 1.2	0.88	6.38	1.28	1.10	6.25	6.31	6.97
f_4	Rastrigin	1.13	6.78	1.64	1.36	5.03	5.47	3.17
f_5	Griwank	1.56	6.96	1.58	1.36	4.80	5.05	3.65
f_6	Ackley	1.08	6.73	1.50	1.45	5.22	3.92	3.61
f_7	Alpine	0.92	6.42	1.33	1.64	5.17	4.27	5.63
f_8	Salomon	0.94	6.48	1.39	1.82	3.58	3.11	3.48
f_9	Powell	1.28	7.33	1.74	1.52	5.47	4.86	5.74
f_{10}	Weierstrass	59.35	159.40	93.05	84.53	57.69	57.16	56.74
f_{11}	Zakharov	1.88	8.53	2.94	2.25	5.32	5.05	6.12
f_{12}	Elliptic	1.43	7.37	2.06	1.64	6.94	6.71	7.63
f_{13}	Matyas	0.19	0.55	0.24	0.24	0.48	0.46	0.89
f_{14}	Schaffer	0.19	0.55	0.25	0.25	0.46	0.46	0.44
f_{15}	Boachevsky3	0.20	0.55	0.25	0.24	0.48	0.52	0.48
f_{16}	Easom	0.19	0.55	0.25	0.24	0.41	0.38	0.90

由表9知, 与MDE, distABC, ADN–RSN–PSO相比DSA系列所用时间较长, 主要原因是呼叫阶段TS的更新. 对于单模态函数f_1, f_2, f_3, f_9, f_{11}, f_{12}, f_{13}, f_{16}, JDSA 有略大一些的时间消耗, 但在同等条件下, JDSA时间复杂度的阶并没太大改变. CDSA用时比DSA略短, 主要原因是CDSA在初始化阶段具有良好的初始值, 避免了随机搜索的耗时; 对于含有多个局部极值点或局部最优点的多模态函数f_4, f_5, f_6, f_7, f_8, f_{10}, f_{14}, f_{15}, JDSA用时相对较短, 这也表明JDSA的全局搜索和局部开发能力较强. 综上可知, JDSA并没有以消耗较多运行时间来提升算法的收敛性能, 时间复杂度与DSA和CDSA在数量级上相当.

5 结论

针对海豚群算法优化单目标问题时, 存在的收敛速度慢和收敛精度不高等缺陷, 提出JDSA算法, 本算法有4个特点: 1) 在搜寻阶段加入声波长度自适应策略, 声波长度随迭代次数的增加而减小, 平衡了算法的全局搜索和局部开发能力; 2) 在搜寻阶段后加入跳跃步骤, 使海豚个体直接跳跃到相应的邻域最优解, 保留了部分较优解, 在保证算法的收敛速度的同时增加了种群的多样性; 3) 在捕猎阶段, 改进个体位置更新公式, 在其中加入变异扰动因子, 提升算法收敛速度的同时, 也维持了种群多样性; 4) 在捕猎阶段后执行早熟收敛机制, 一定程度上可以避免算法陷入局部最优. 最后, 本文分析了参数对算法性能的影响并在16个标准测试函数上进行测试, 结果表明, 与DSA算法、CDSA算法以及目前应用效果较好的4种算法相比, JDSA算法在收敛精度、收敛速度、鲁棒性上都有明显优势, 有一定的应用价值.

参考文献:

[1] YANG X S. A new metaheuristic bat-inspired algorithm. *Computer Knowledge & Technology*, 2010, 284(12): 65 – 74.

[2] YANG X S. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2010, 2(2): 78 – 84.

[3] PAN W T. A new fruit fly optimization algorithm: taking the financial distress model as an example. *Knowledge-Based Systems*, 2012, 26(2): 69 – 74.

[4] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer. *Advances in Engineering Software*, 2014, 69(3): 46 – 61.

[5] MENG X B, GAO X Z, LU L, et al. A new bio-inspired optimisation algorithm: Bird swarm algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, 2016, 28(4): 673 – 687.

[6] WU T Q, YAO M, YANG J H. Dolphin swarm algorithm. *Frontiers of Information Technology & Electronic Engineering*, 2016, 17(8): 717 – 729.

[7] CHANG J, LIU D S. Structural modal parameter identification based on quantum-behaved particle swarm optimization. *Journal of Vibration & Shock*, 2014, 33(23): 42 – 46.

[8] UEDA Y, HORIO K, KUBOTA R. A modified real-coded genetic algorithm considering with fitness-based variability. *International Journal of Innovative Computing Information & Control Ijicic*, 2014, 10(4): 1509 – 1518.

[9] LI X, YANG G. Artificial bee colony algorithm with memory. *Applied Soft Computing*, 2016, 41(1): 362 – 372.

[10] LI Zhipeng, LI Weizhong, WANG Yang, et al. Air-targets threat assessment using grey neural network optimized by chaotic dolphin swarm algorithm. *Control and Decision*, 2018, 33(11): 1 – 6.
(李志鹏, 李卫忠, 江洋, 等. 混沌海豚群优化灰色神经网络的空中目标威胁评估. 控制与决策, 2018, 33(11): 1 – 6.)

[11] WATERS D A. Echolocation in air: Biological systems, technical challenges, and transducer design. *ARCHIVE Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2007, 221(10): 1165 – 1175.

[12] ZHANG Na, ZHAO Zedan, BAO Xiaoan, et al. Based on modified Tent chaotic gravity search algorithm. *Control and Decision*, 2019: 1 – 8 [2019–01–12]. https: //doi.org/10.13195/j.kzyjc.2018.0795.
(张娜, 赵泽丹, 包晓安, 等. 基于改进的Tent混沌万有引力搜索算法. 控制与决策, 2019: 1 – 8 [2019–01–12]. https: //doi.org/10.13195/j.kzyjc.2018.0795.)

[13] LIU Zhijun, GAO Yakui, ZHANG Weiguo, et al. Decomposition with ensemble neighborhood size multi-objective adaptive differential evolutionary algorithm. *Control Theory & Applications*, 2014, 31(11): 1492 – 1501.
(刘志君, 高亚奎, 章卫国, 等. 变邻域分解多目标自适应差分进化算法. 控制理论与应用, 2014, 31(11): 1492 – 1501.)

[14] WU Yali, JIN Xiaoyi, LIU Ge. Chainlike multi-population multi-agent evolutionary algorithm. *Control Theory & Applications*, 2013, 30(1): 37 – 53.
(吴亚丽, 靳笑一, 刘格. 链式多种群多智能体进化算法. 控制理论与应用, 2013, 30(1): 37 – 53.)

[15] LI X, YIN M. *Modified Differential Evolution with Self-adaptive Parameters Method*. New York: Springer-Verlag Inc., 2016.

[16] BABAOGLU, ISMAIL. Artificial bee colony algorithm with distribution-based update rule. *Applied Soft Computing*, 2015, 34(1): 851 – 861.

[17] XIAO J, NIU Y, CHEN P, et al. An improved gravitational search algorithm for green partner selection in virtual enterprises. *Neurocomputing*, 2016, 217(18): 103 – 109.

[18] SUN W, LIN A, YU H, et al. All-dimension neighborhood based particle swarm optimization with randomly selected neighbors. *Information Sciences*, 2017, 405(C): 141 – 156.

作者简介:

王艳娇 副教授, 博士, 主要研究方向为进化计算、智能信息处理, E-mail: wangyanjiao1028@126.com;

史新梦 硕士研究生, 主要研究方向为进化计算、智能信息处理, E-mail: m13131702191_1@163.com.