融合均值榜样的反向互学习水母搜索算法

段艳明[a],肖辉辉[a,b],谭黔林[a]

(河池学院 a.大数据与计算机学院; b.广西蚕桑生态学与智能化技术应用重点实验室,广西 河池 546300)

摘 要:为解决水母搜索算法(jellyfish search algorithm,JS)的洋流运动缺乏多样性、群内运动缺乏引导性、种群间信息无交流,造成搜索速度慢、稳定性差及易早熟的问题,构建了一种融合均值榜样的反向互学习水母搜索算法(oppositional-mutual learning jellyfish search algorithm based on mean-value example,OMLJS).首先在水母跟随洋流运动(全局搜索)部分,利用前两代水母的平均位置代替只考虑上一代水母的平均位置来引导水母个体的位置更新,提高算法的全局搜索能力;其次在水母的群内主动运动(局部搜索)部分,利用最优个体代替随机个体来引导水母进行更有效的搜索,加快算法的收敛速度;然后在水母进入下一次迭代前增加对水母种群进行动态反向互学习步骤,增加种群多样性及增强种群间的信息交流,达到互补另外两个策略,提高算法的整体优化性能.选用 12 个经典的基准测试优化函数,将 OMLJS 与 5 个对比算法从解的平均值、最优值及方差进行对比分析,并用于求解最小生成树问题,OMLJS 能够更快地找到最小生成树.实验结果表明,OMLJS 的收敛速度、求解精度明显提高.

关键词:水母搜索算法;均值榜样学习;反向互学习;时间控制机制;最小生成树问题

中图分类号:TP18 **文献标志码**:A **文章编号**:1000-2367(2024)04-0111-09

受自然界演化规律和生物群体的智能行为启发,不断涌现出求解复杂优化问题的群智能优化算法.如粒子群算法(particle swarm optimization,PSO)、遗传算法(genetic algorithm,GA)、差分算法(difference algorithm,DE)、变色龙群算法(chameleon swarm algorithm,CSA)[1]、蜉蝣算法(mayflies algorithm,MA)[2]等.群智能优化算法可以在缺乏全局信息的条件下,对各种复杂优化问题进行有效求解,为解决当前复杂的工程实践等优化问题开辟了一条新途径.这促进了群智能优化算法在函数优化、图像处理、无线传感器网络、聚类[3]、数据挖掘、大数据、云计算、逆运动学[4]等各个领域中的广泛应用.

CHOU 等[5]于 2020 年提出了一种新型群智能优化算法——人工水母搜索算法(artificial jellyfish search algorithm,JS).该算法具有较高优化性能,且实现比较简单.但与所有的群智能算法类似,在种群迭代后期,随着种群多样性的丢失,使算法较易陷入局部最优,同时,若对种群没有采取有效措施来加快向最优解的逼近,算法收敛速度较慢.为此,许多学者对其进行多方面的改进研究.

TRUONG 等[6]为了更好地平衡 JS 算法的洋流运动(全局搜索)和群内运动(局部搜索),采用模糊自适应策略对 JS 算法的时间控制机制进行改进,加强了算法的勘探能力和开发能力的平衡.MANITA 等[7]在算法后期对水母个体执行正交学习策略,增强解的多样性,有效提高算法的全局搜索能力.ABDEL-BASSET 等[8]利用基于排序更新方法对最优解周围的解空间进行广泛搜索,提高算法的搜索速度和解的精度,并运用自适应策略使当前解在最优个体的周围进行两种不同的自适应步长位置更新,提高算法的开发能力和避免算法早熟,有效提高算法的优化能力.王秋萍等[9]提出了基于改进双种群水母搜索算法,在两个种群之间利用

收稿日期:2023-04-25;**修回日期**:2023-05-30.

基金项目:国家自然科学基金(61972184;61562032);河池学院高层次人才科研启动项目(2019GCC012).

作者简介:段艳明(1978—),女,江西永新人,河池学院副教授,研究方向为智能计算.

通信作者:肖辉辉,教授,博士, E-mail:gxhcxyxzy@126.com.

引用本文:段艳明,肖辉辉,谭黔林.融合均值榜样的反向互学习水母搜索算法[J].河南师范大学学报(自然科学版),2024,52(4):111-119.(Duan Yanming, Xiao Huihui, Tan Qianlin. Oppositional-mutual learning jellyfish search algorithm based on mean-value example[J]. Journal of Henan Normal University(Natural Science Edition),2024,52(4):111-119.DOI:10.16366/j.cnki.1000-2367.2023.04.25.0001.)

相互学习交流机制,以提升算法的优化速度,并将改进算法用于多阈值图像分割问题,得到较好的分割效果.上述学者提出的改进措施,能在一定程度上有效提高基本 JS 算法的寻优性能,但仍存在不足之处:首先,没有针对性地对 JS 算法的洋流运动部分和群内运动提出改进策略;其次,只限于两个种群间的学习交流,减少了个体的交流机会,不利于算法优化速度的提升;最后,以上改进对算法的全局搜索能力没有得到较大提高及未能完全跳出局部最优解,求解复杂多峰函数时依然易陷入局部最优,导致解的精度和优化速度有待提高.

为此,本文提出了一种融合均值榜样的反向互学习水母搜索算法(OMLJS).OMLJS 算法在基本 JS 算法的基础上主要提出了 3 个改进策略:(1)均值榜样策略:在水母跟随洋流运动(全局搜索)部分,利用前两代水母的平均位置来引导水母个体的位置更新,进而提高算法的全局搜索能力;(2)最优个体引导策略:在水母的群内主动运动(局部搜索)部分,利用最优个体来引导水母进行更有效的搜索,使水母个体有迹可循,从而加快算法的收敛速度;(3)动态反向互学习策略:对水母种群进行动态反向学习来增加种群多样性,并通过个体间的互学习来增加种群的信息交流,从而达到互补前面两个策略,提高算法的整体优化性能.通过 12 个经典测试函数及 5 个对比算法与本文算法对比分析,并对本文所提出的 3 种改进策略的有效性分别进行验证.实验结果表明本文算法的优化能力表现突出,具有明显的优势,充分体现了本文算法的可行性和优越性.同时把本文算法用于求解最小生成树问题,通过 4 个算例的验证,同样表明本文算法的有效性及较强的优化性能.

1 水母搜索算法(JS)

水母搜索算法(JS)是模拟水母在海洋中的捕食行为,水母追随食物可以在海洋中随意运动,也可以通过自己收缩向后推水来向前运动,或以其他水母运动作为其运动参照.

1.1 JS 算法的洋流运动(全局搜索策略)

当洋流富含食物时,水母会跟随洋流朝食物丰富的方向移动.JS 算法的洋流运动如式(1)所示:

$$\eta = X^* - \beta \times \mathrm{rand}(0,1) \times \mu, \tag{1}$$

其中,X^* 表示目前最优水母位置,μ 为所有水母种群的平均位置,β 为分布因子,其取值通常为 3.

则水母的位置可由式(2)更新:

$$X_i(t+1) = X_i(t) + \mathrm{rand}(0,1) \times \eta. \tag{2}$$

1.2 JS 算法的群内运动(局部搜索策略)

水母种群内的运动即为优化问题的局部搜索,又分为被动运动和主动运动两种.

1)被动运动:水母群在形成初期,无需其他水母的信息去寻找更好的食物,其运行形式表现为被动运动,其位置的更新用式(3)进行:

$$X_i(t+1) = X_i(t) + \gamma \times \mathrm{rand}(0,1) \times (U_b - L_b), \tag{3}$$

其中,U_b 和 L_b 分别是搜索空间的上限值和下限值,γ 是大于零的运动系数.

2)主动运动:主动运动是当前水母个体 i 借助另一水母个体 j 实现搜索食物的运动形式.利用水母 i 到水母 j 的向量来确定当前水母 i 的运动方向,当水母 j 位置的食物溶度比水母 i 位置的食物溶度高时,则水母 i 向水母 j 移动,否则,水母 i 远离水母 j .其位置更新如式(4)所示:

$$X_i(t+1) = X_i(t) + \mathrm{rand}(0,1) \times L, \tag{4}$$

这里,L 是群内运动方向,由式(5)决定:

$$L = \begin{cases} X_j(t) - X_i(t), f(X_i) \geq f(X_j) \\ X_i(t) - X_j(t), f(X_i) < f(X_j). \end{cases} \tag{5}$$

1.3 时间控制机制

水母的运动形式随着时间的推移在不断变化.在算法初期为洋流运动,随着算法的迭代,水母进行群内运动,群内运动根据食物的溶度又分为主动运动和被动运动,为了调控水母的这 3 种运动形式,算法利用了时间控制机制.时间控制机制分为时间控制函数 $c(t)$ 和常数 c_0.时间控制函数用搜索迭代次数来模拟,如

式(6)所示:

$$c(t)=\left|\left(1-\frac{t}{M}\right)\times(2\times\mathrm{rand}(0,1)-1)\right|,\tag{6}$$

其中,t 是算法目前进化的代数,M 是算法进化的最大代数.显然,$c(t)\in[0,1]$,且其值与迭代次数相关.常数 c_0 设为 0.5,当 $c(t)\geq c_0$ 时,水母进行全局搜索(洋流运动).否则,水母执行局部搜索(群内运动):当 $\mathrm{rand}(0,1)>(1-c(t))$ 时,水母进行被动运动,反之为主动运动.由于$(1-c(t))$ 随时间的推移从 0 增加到 1,因此,水母在种群中也逐渐从被动运动向主动运动过渡.

2　融合均值榜样的反向互学习水母搜索算法

2.1　均值榜样策略

通过分析可知,式(2)中的 η(洋流运动)利用了上一代种群的平均值来引导水母个体的位置更新,即考虑了个体间的相互协作有利于团队目标的实现,但若能把相互协作推广到整个种群中,通过获取种群中的所有个体信息,达到扩大协作范围,则在很大程度上能提高算法的全局搜索能力.根据此思路,为了避免偶然性和扩充团队的协助范围,在全局搜索部分水母个体的位置更新中利用前两代的种群均值信息来引导个体的运动,即用式(7)代替式(2)来更新水母位置:

$$X_i(t+1)=X_i(t)+(a(t-1)-a(t-2))\times\eta,\tag{7}$$

其中,$a(t-1)$ 和 $a(t-2)$ 为种群中所有水母在 $t-1$ 代和 $t-2$ 代的位置均值.这样,通过广泛获取种群在前两代中所有水母的经验来提高搜索区域的有效性,避免水母盲目搜索,进而提高算法的勘探能力.

同时用式(8)更新式(7)的 η(洋流运动),使水母个体跟随最优个体 X^* 运动,加快搜索进度,

$$\eta=X^*-X_i(t).\tag{8}$$

这样,利用前两代的种群均值信息和当前最优个体来指导当前水母的运动方向,使水母朝着更有效的方向搜索目标,从而使水母更快地找到有效搜索区域,提高算法的全局搜索能力.

2.2　最优个体引导策略

在 JS 算法的局部搜索的水母主动运动部分,水母个体通过向任意其他个体学习,来达到朝着食物多的方向运动.虽然水母间的信息交流,能很好地在较大范围进行搜索食物,但这样加大了搜索的范围,增加了水母搜索的盲目性,即出现偏航现象,降低了搜索速度及算法的不稳定性.为此,可以利用最优个体来引导水母进行更有效的搜索,使水母个体有迹可循,从而加快算法的收敛速度.因此,可以用式(9)替换式(4)来实现水母的主动运动:

$$X_i(t+1)=X^*+\mathrm{rand}(0,1)\times L,\tag{9}$$

其中,X^* 为当前最优个体,L 同式(4).当算法迭代后半期,即 $|c(t)|<0.5$ 且 $\mathrm{rand}(0,1)>1-c(t)$ 时,算法主要进行局部开发,利用式(9)中的最优个体的引导作用可以使水母更快地收敛到最优解,进而提高算法的开发能力.

为了证明最优个体在水母主动运动中的引导作用,利用非凸病态优化函数 Rosenbrock 对该改进策略进行验证(后面实验部分将进行更全面验证),OMLJS1 为原始水母主动运动,OMLJS2 为采用了最优个体引导的水母主动运动,图 1 为这两种算法在独立运行 20 次的全局最优适应度值对比图,从图 1 中可以看出,OMLJS1 在 20 次中只有 5 次能找到最优值 0,其他 15 次存在反复跳动,而 OMLJS2 在 20 次中能找到最优值 0.因此,进一步证明了在水母主动运动部分利用最优个体引导策略的有效性,算法更稳定,收敛速度更快.

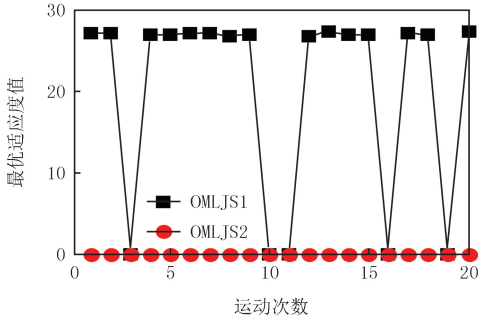


图1　最优个体引导策略对比图

Fig.1　Comparison of optimal individual guidance strategies

2.3 动态反向互学习策略

在上面两个策略的作用下,虽然水母较集中地向最优个体靠拢,但这样易使算法“早熟”.为此,本小节采用动态反向互学习策略,对水母种群进行动态反向学习来增加种群多样性,并通过种群间的互学习来增加种群间的信息交流[10],从而达到互补前面两个策略,提高算法的整体优化性能.动态反向互学习策略是使水母个体向动态反向个体或随机其他个体进行学习的一种有效学习策略.

1)动态反向学习.动态反向学习能引导水母个体在具有不对称性和动态变化特征的搜索空间中从其对立位置进行学习,增加了个体之间的差异性,这将在很大程度上提高最优解的概率,从而提升了算法的勘探能力.如式(10)所示:

$$X^{dol}=X-r_1(r_2X^0-X), \tag{10}$$

其中,$X^0=U_b+L_b-X$,U_b、L_b 分别为搜索空间的下界和上界,则 X^0 为 X 的反向值,r_1 和 r_2 为[0,1]的随机数.

2)互学习.通常,优化算法在迭代的每一代,都是保存适应度最好的最优个体作为下一代搜索的参考,但有时适应度差的个体在某些维度上可能有更好的优化信息.因此,个体间可以通过相互学习来提高信息的交流,如式(11)所示:

$$X^{ml}=X+r^3(X^{\mathrm{rand}}-X), \tag{11}$$

其中,$r^3\in\mathrm{rand}(0,1)$,X^{rand} 为不同于 X 的随机个体,则 $X^{\mathrm{rand}}-X$ 为当前个体与随机个体间的差异,也正是因为这个差异使算法进行个体间的信息交流,从而提高算法的开发能力.

3)动态反向互学习.为了充分利用动态反向学习和互学习的各自优势,提高算法的勘探能力和开发能力,为此通过式(12)来实现水母的动态反向互学习:

$$\begin{cases}X_i=X_i+r_1(r_2(U_b+L_b-X_i)-X_i),\mathrm{rand}(0,1)<0.5,\\ X_i=X_i+r_3(X_m-X_i),\text{其他}.\end{cases} \tag{12}$$

这里,r_1,r_2 和 r_3 为[0,1]的随机数,$X_m(m\neq i)$ 为当前代不同于 X_i 的随机水母个体.通过式(12),水母种群在一定概率下进行动态反向学习,增加了个体之间的差异性,同时通过互学习增加了种群间的信息交流,达到提升算法勘探和开发能力的效果.

算法在进入下一迭代前,通过对水母种群 P_1 实施动态反向互学习,得到一组新的种群 P_2,把新种群 P_2 与之前的种群 P_1 合并,种群数翻了一倍($2*N$).然后,根据适应度值从中选取排在前 N 个水母更换种群 P_1 进入下一次的迭代.

2.4 OMLJS 算法流程

基于上述提出的 3 个改进策略,改进的水母搜索算法(OMLJS 算法)的步骤如下:

步骤 1 对参数赋值:种群数 N,空间维度 D,迭代次数 M.

步骤 2 利用 Logistic 混沌映射初始化种群.

步骤 3 计算水母个体的适应度值及所有水母的平均位置,并保存最优个体及最优值.

步骤 4 利用式(6)计算时间控制函数值 $c(t)$.

步骤 5 若 $|c(t)|\geq0.5$,算法进行全局搜索(水母跟随洋流运动),即利用式(7)进行位置更新,随即进行越界处理,更新最优位置及最优值.

步骤 6 若 $|c(t)|<0.5$,算法进行局部搜索(水母群内运动):并当 $\mathrm{rand}(0,1)>1-c(t)$ 时,利用式(4)进行位置更新,否则利用式(9)进行位置更新.随即进行越界处理,更新最优值及最优位置.

步骤 7 水母进行动态反向互学习:利用式(12)更新所有水母的位置得到一组新种群,然后与原种群合并为 $2*N$ 的种群,再计算其适应度值,取前 N 的水母作为下一次迭代的种群.

步骤 8 若达到算法最大进化代数,则进化结束,并输出最优解和最优值,不然执行步骤 3 至步骤 8.

2.5 OMLJS 算法时间复杂度分析

通过对 OMLJS 算法分析可知,OMLJS 算法主要包括种群初始化、跟随洋流运动、群内主动运动、群内被动运动、动态反向互学习 5 个阶段.设种群规模为 N,问题空间为 D,M 为算法最大进化代数,算法的参数初始化时间设为 t_0,每一维生成随机数的时间设为 t_1,越界处理的时间设为 t_2,求解适应度值的时间频度为 $f(D)$.

则基本水母算法 JS 的种群初始化阶段的时间复杂度为:

$$T_1=O(t_0+N(t_1\times D+f(D)))=O(D+f(D)). \tag{13}$$

设基本水母算法 JS 跟随洋流运动阶段通过式(3)更新位置的时间为 t_3,则此阶段的时间复杂度为:

$$T_2=O(N((t_2+t_3)\times D+f(D)))=O(D+f(D)). \tag{14}$$

设基本水母算法 JS 群内主动运动阶段通过式(6)更新位置的时间为 t_4,则此阶段的时间复杂度为:

$$T_3=O(N((t_2+t_4)\times D+f(D)))=O(D+f(D)). \tag{15}$$

设基本水母算法 JS 群内被动运动阶段通过式(4)更新位置的时间为 t_5,则此阶段的时间复杂度为:

$$T_4=O(N((t_2+t_5)\times D+f(D)))=O(D+f(D)). \tag{16}$$

因此,基本水母算法 JS 的时间复杂度为:

$$T=T_1+M\times(T_2+T_3+T_4)=O(D+f(D)). \tag{17}$$

改进算法 OMLJS 有 3 个地方进行了改进,其中对水母跟随洋流运动阶段、水母群内主动运动阶段的更新位置进行的改进,并不增加时间复杂度,与基本 JS 算法一致.而动态反向互学习策略的又包括动态反向学习、互学习、越界处理、合并种群、计算适应度值及对水母个体的适应度值进行排序 6 个部分.假设动态反向学习的时间为 t_6,互学习的时间为 t_7,越界处理的时间及计算机适应度值的时间同上,合并种群的时间为 t_8,水母个体的适应度值进行排序的时间为 t_9.则动态反向互学习的时间复杂度为:

$$T_5=O(N((t_2+t_6+t_7)\times D+t_8+t_9+2f(D)))=O(D+f(D)). \tag{18}$$

则 OMLJS 算法的时间复杂度为:

$$T^*=T_1+M\times(T_2+T_3+T_4+T_5)=O(D+f(D)). \tag{19}$$

由此可见,改进的 OMLJS 算法与 JS 算法的时间复杂度属于同一数量级,并没有提高算法的时间复杂度,在算法的运行时间上是有效的.

3　仿真实验及结果分析

3.1　测试函数及对比算法

为了验证本文提出的 OMLJS 算法的性能,选取 12 个典型基准测试函数进行验证,如表 1 所示.

除了与基本水母算法 JS 进行对比外,本文选用了 4 个算法进行比较:MFCPSO 算法(PSO variant based on a multiple-input multiple-output Fuzzy logic controller,MFCPSO)[11]、异构综合学习 PSO 算法(heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation,HCLPSO)[12]、基于多种群变异策略集成的差分进化算法(differential evolution with multi-population based ensemble of mutation strategies,MPEDE)[13]、改进的 JS 算法(modified artificial jellyfish search optimizer,MJSO)[14].在实验中,种群规模 $N=50$,最大进化代数 $M=1\ 000$,对比算法中的其他参数来自对比算法相应的参考文献.

3.2　解的质量对比分析

为了检验算法优化性能的鲁棒性和正确性,避免算法的偶然性带来实验误差,用各测试函数独立运行 20 次的仿真结果的平均值、最优值和方差作为比较依据,平均值用来反映算法的求解精度,方差用来反映算法的稳定性.实验结果如附录表 S1 所示,其中加粗的实验结果为该函数在对比算法中优化结果最优.从附录表 S1 中可以看出:1)本文 OMLJS 算法的性能较对比算法有明显的提高,在 12 个测试函数中有 10 个函数能找到全局最优值,其他 2 个函数也是对比算法中最好效果的,其中 f_3 函数的平均值和最优值均达到 8.881 8e-16,方差为 0;2)基本水母搜索算法 JS 只有 2 个函数找到全局最优值,尤其是在带旋转的多峰多维函数(f_{10}-f_{12}),JS 算法的优化效果很不理想,因为 JS 算法的搜索方程是基于随机的,这种搜索方式效率低下,导致收敛速度慢,JS 算法的提出者在参考文献[5]的迭代次数为 10 000,才能在大部分包括 f_{10}-f_{12} 在内的测试函数上找到最优解,说明本文算法在水母的主动运动部分利用最优个体引导策略的有效性,能很大程度上提高收敛速度.3)对于非凸病态优化函数 f_2,全局最小值位于一个狭长的抛物线形平坦山谷内,收敛到全局最小值很困难,在迭代次数为 1 000 的情况下,对比算法 HCLPSO、MFCPSO、MPEDE 的优化结果与理

论值存在较大的差距,这 3 种对比算法在参考文献中的迭代次数都为 10 000 次,才能有相对较好的优化结果,而本文算法在迭代次数为 1 000 的情况下就能找到全局最优解,证明了本文算法改进策略是有效的.4)对于多峰多维函数 f_6 和带旋转的多峰多维函数 f_{10},在迭代次数为 1 000 的情况下,对比算法 HCLPSO、MFCPSO、MPEDE 的优化结果也不太理想,但本文算法亦能找到全局最优值,证明了本文算法的收敛精度得到很大的提升.5)对比算法 MJSO 是通过两个随机个体之间位置及最优个体和一个随机个体之间的位置差异来引导水母运动,加速向最优解的收敛,一定程度上提升了算法的收敛速度和收敛精度,但差于本文算法,证明了本文算法中的 3 种改进策略相辅相成,共同提升算法的优化性能.

因此,本文算法 OMLJS 的优化能力表现出良好的竞争力,验证了本文提出的改进策略一定程度上提高了算法的收敛精度、收敛速度及鲁棒性等优化性能.

表 1 测试函数

Tab. 1 Test functions

函数类型		函数名	维度	搜索范围	最优值
单峰多维	f_1	Sphere	30	[−100,100]	0
单峰多维	f_2	Rosenbrock	30	[−30,30]	0
多峰多维	f_3	Ackley	30	[−32,32]	0
多峰低维	f_4	Schaffer	2	[−10,10]	0
多峰多维	f_5	Griewank	30	[−600,600]	0
多峰多维	f_6	Rastrigrin	30	[−5.12,5.12]	0
多峰多维	f_7	Schwefe's2.2	30	[−10,10]	0
多峰多维	f_8	Zakharov	30	[−10,10]	0
单峰多维	f_9	Shifted Sphere	30	[−100,100]	−450
多峰多维	f_{10}	Shifted Schwefel's	30	[−100,100]	−450
多峰多维	f_{11}	Shifted Rosenbrock's	30	[−100,100]	390
多峰多维	f_{12}	Shifted Rotated Ackley's	30	[−32,32]	−140

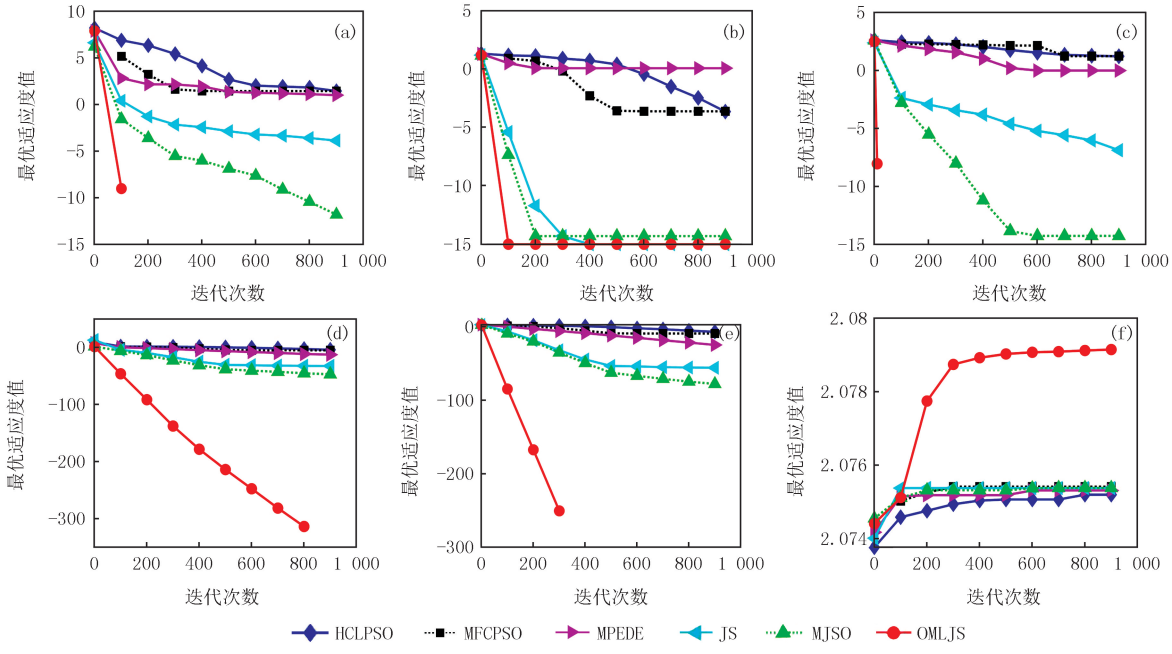
3.3 收敛曲线图及最优适应度方差图对比分析

为了更直观地对比本文算法与对比算法的优化性能,显示本文算法的寻优精度、收敛速度和算法跳出局部搜索空间的能力,本小节给出了如图 2 所示的收敛曲线图和如图 3 所示的最优适应度方差图,限于篇幅只给出部分测试函数的图.从图 2 中能很明显地看出,整体上本文算法较对比算法其收敛速度最快,在算法迭代前期本文算法的求解速度较于其他对比算法有着很大的提升,说明局部开发能力要好于其他算法,能够快速地对搜索空间进行遍历寻优,从而缩短算法前期的勘探时间.从图 2 中也能明显地看出本文算法的寻优精度明显要高于绝大部分对比算法.从而验证了本文算法的改进策略的有效性,很大程度上提高了算法的收敛速度.同时,从图 3 可以看出,本文算法的最优适应度方差几乎接近 0,而其他对比算法亦偏大,尤其是 HCLPSO 算法、MFCPSO 算法和 MPEDE 算法.因此,本文算法的稳定性及求解精度要好,其改进策略有效.

3.4 改进策略的有效性分析

本文算法对基本 JS 算法提出了 3 个改进策略:水母跟随洋流运动的均值榜样策略、水母群内主动运动的最优个体引导策略、种群的动态反向互学习策略,为了验证各个改进策略对本文算法的性能影响,本小节分别对其进行测试.设仅在水母跟随洋流运动部分采用均值榜样策略的算法为 JS1,仅在水母群内主动运动部分采用最优个体引导策略的算法为 JS2,仅在进入下一代迭代前对种群采用动态反向互学习策略的算法为 JS3,然后再与基本 JS 算法,及本文 OMLJS 算法在选用的 6 个测试函数上进行比较,实验结果如附录表 S2 所示.

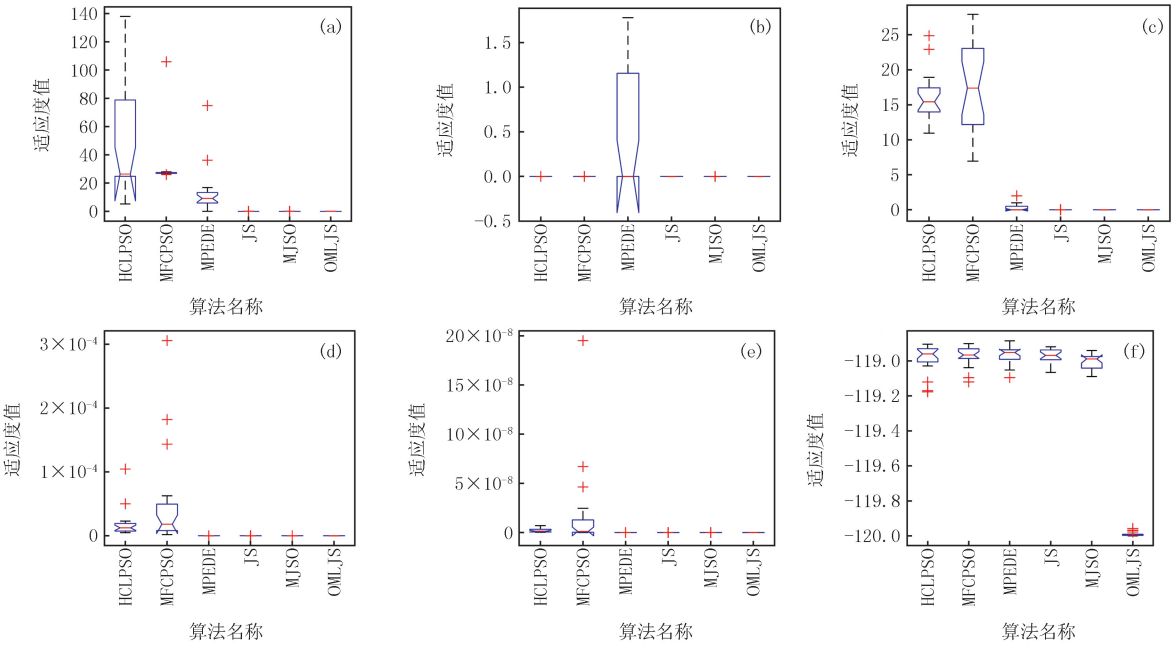
从附录表 S2 中可以看出,每种改进策略对基本 JS 算法都能提高优化性能,其中对 f_4 函数的 3 个评价指标都达到 0,而其他 5 个函数的实验结果表明,动态反向互学习的策略的效果要更好一点,若跟另外 2 种改进策略综合后效果更佳,测试函数中每个改进策略都表现出较强的寻优能力.因此,3 种单一策略改进的 JS 算法在寻优精度和收敛速度方面比基本 JS 算法都有所提升,但是次于综合策略改进的 OMLJS 算法,这更进一步验证了本文算法中的 3 种改进策略的有效性,相辅相成,共同作用于提高算法的优化能力.



(a)f_2函数, (b)f_3函数, (c)f_6函数, (d)f_7函数, (e)f_8函数, (f)f_{12}函数.

图2　部分测试函数收敛曲线对比图

Fig. 2　Comparison of some test functions convergence curve



(a)f_2函数, (b)f_3函数, (c)f_6函数, (d)f_7函数, (e)f_8函数, (f)f_{12}函数.

图3　部分测试函数最优适应度值方差图

Fig. 3　Comparison of variance of optimal fitness values of some test functions

4　求解最小生成树问题

为了进一步验证 OMLJS 算法的优化性能,利用 OMLJS 对最小生成树问题进行了测试,并与对比算法进行比较.最小生成树问题是组合优化最典型、最著名的问题之一,最小生成树有许多实际应用,如电路设计问题、城市电缆铺设问题、旅行商问题等.给定一个带权连通无向图 $G=(V,E)$,顶点的集合 $V=\{v_1,v_2,\cdots,v_n\}$,边的集合 $E=\{(v_i,v_j)\mid v_i\in V,v_j\in V,i\neq j\}$,对于每条边 $(v_i,v_j)\in E$,都有非负权重 w_{ij}. 连通图 G

的生成树 T 是一个非环子图,含有图 G 中全部 n 个顶点和构成一棵树的($n-1$) 条边.对于带权连通无向图 G,可能有多棵不同生成树,定义每棵生成树的所有边的权值之和为 $\omega(T)$,则最小的生成树称为图 G 的最小生成树.$\omega(T)$ 由式(20) 计算得到:

$$\omega(T)=\sum_{(v_i,v_j)\in E} w_{ij}, \tag{20}$$

其中,(v_i,v_j) 为生成树上的 $n-1$ 条边,w_{ij} 为边(v_i,v_j) 的权重.

将 OMLJS 应用于最小生成树问题,该问题可以描述如下:

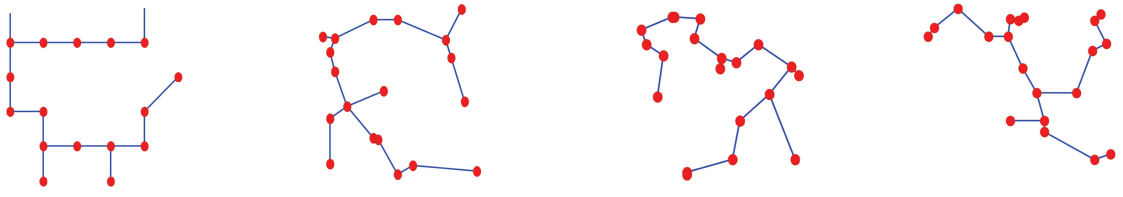
步骤 1　以二维矩阵的形式输入图 G.

步骤 2　利用 OMLJS 求 G 的最小生成树,将 OMLJS 中的每个种群(候选解)作为一棵生成树,将式(20)作为评估候选解的目标函数,则目标函数最小值的解为最佳解,即为图 G 的最小生成树.

步骤 3　输出最小生成树.

利用 4 个算例进行测试,算例来自参考文献[15],种群 $N=40$,最大进化代数 $M=500$,独立运行 20 次,实验结果见附录表 S3 所示.Mean、Best、Std 分别代表算法独立运行 20 次所得最小生成树的平均权重和、最优权重和及方差.从表 S3 可以看出,较对比算法,OMLJS 算法在每个算例上都能得到较好的平均权重和、最优权重和及方差,特别在算例 1 和算例 2 上,OMLJS 算法每次都能得到最佳最小生成树,其平均权重和、最优权重和都达到最佳值,方差均为 0.对于算例 3 和算例 4,OMLJS 算法的优化性能较对比算法也是最好的.虽然对比算法 MFCPSO 和 MJSO 在 4 个算例中也能找到最好的最优权重和,但平均权重和及方差要略差于 OMLJS 算法.因此,OMLJS 算法在求解最小生成树问题中也表现出很好的效果,进一步验证了 OMLJS 算法的有效性及较强的优化性能.

为了更直观地体现 OMLJS 算法求解最小生成树的结果,图 4 给出了 OMLJS 算法求解 4 个算例所得到的最小生成树.在每个实例中,每个顶点都由一个圆表示,n 个顶点和 $n-1$ 条选择的总权重最小的边构成了图 G 的最小生成树.



(a) 算例1　(b) 算例2　(c) 算例3　(d) 算例4

图4　OMLJS生成的最小生成树

Fig. 4　Minimum spanning trees obtained by OMLJS

5　结　论

水母搜索算法是一种新型群智能优化算法,具有较好的优化性能,但由于该算法的洋流运动搜索方式(全局搜索)、群内运动搜索方式(局部搜索)及时间控制机制仍存在一定问题,造成该算法搜索速度慢、稳定性差及易早熟等方面的问题.本文针对上述存在的不足,提出了均值榜样学习的动态反向互学习的改进水母搜索算法.为了检验改进算法的可行性和有效性,对 12 个基准测试优化函数进行优化,并用于求解最小生成树问题,本文算法的优化结果与对比算法相比,优化能力得到较明显的提高.

附录见电子版(DOI:10.16366/j.cnki.1000-2367.2023.04.25.0001).

参　考　文　献

[1] BRAIK M S.Chameleon Swarm Algorithm:a bio-inspired optimizer for solving engineering design problems[J].Expert Systems with Applications,2021,174:114685.

[2] ZERVOUDAKIS K,TSAFARAKIS S.A mayfly optimization algorithm[J].Computers & Industrial Engineering,2020,145:106559.

[3] ZHANG X M,LIN Q Y.Three-learning strategy particle swarm algorithm for global optimization problems[J].Information Sciences,2022,593:289-313.

[4] 黄华娟，闵峰.求解逆运动学的多策略蜻蜓算法[J].河南师范大学学报(自然科学版)，2023，51(5)：46-58.
HUANG H J，MIN F.Multi-strategy dragonfly algorithm for solving inverse kinematics[J].Journal of Henan Normal University(Natural Science Edition)，2023，51(5)：46-58.
[5] CHOU J S，TRUONG D N.A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean[J].Applied Mathematics and Computation，2021，389：125535.
[6] TRUONG D N，CHOU J S.Fuzzy adaptive jellyfish search-optimized stacking machine learning for engineering planning and design[J].Automation in Construction，2022，143：104579.
[7] MANITA G，ZERMANI A.A modified jellyfish search optimizer with orthogonal learning strategy[J].Procedia Computer Science，2021，192：697-708.
[8] ABDEL-BASSET M，MOHAMED R，ABOUHAWWASH M，et al.An improved jellyfish algorithm for multilevel thresholding of magnetic resonance brain image segmentations[J].Computers，Materials & Continua，2021，68(3)：2961-2977.
[9] 王秋萍，李晓丹，戴芳，等.基于改进双种群水母搜索算法的多阈值图像分割[J].纯粹数学与应用数学，2022，38(3)：392-402.
WANG Q P，LI X D，DAI F，et al.Multi-threshold image segmentation based on improved double population jellyfsh search algorithm[J].Pure and Applied Mathematics，2022，38(3)：392-402.
[10] XU Y L，YANG X F，YANG Z L，et al.An enhanced differential evolution algorithm with a new oppositional-mutual learning strategy[J].Neurocomputing，2021，435：162-175.
[11] XIA X W，SONG H J，ZHANG Y L，et al.A particle swarm optimization with adaptive learning weights tuned by a multiple-input multiple-output fuzzy logic controller[J].IEEE Transactions on Fuzzy Systems，2023，31(7)：2464-2478.
[12] LYNN N，SUGANTHAN P N.Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation[J].Swarm and Evolutionary Computation，2015，24：11-24.
[13] WU G H，MALLIPEDDI R，SUGANTHAN P N，et al.Differential evolution with multi-population based ensemble of mutation strategies[J].Information Sciences，2016，329：329-345.
[14] ABDEL-BASSET M，MOHAMED R，CHAKRABORTTY R K，et al.An improved artificial jellyfish search optimizer for parameter identification of photovoltaic models[J].Energies，2021，14(7)：1867.
[15] ZHANG X M，KANG Q，WANG X.Hybrid biogeography-based optimization with shuffled frog leaping algorithm and its application to minimum spanning tree problems[J].Swarm and Evolutionary Computation，2019，49：245-265.

Oppositional-mutual learning jellyfish search algorithm based on mean-value example

Duan Yanming[a]，Xiao Huihui[a,b]，Tan Qianlin[a]

(a. College of Big Data and Computer Science；b. Guangxi Key Laboratory of Sericulture Ecology and Intelligent Technology Application，Hechi University，Hechi 546300，China)

Abstract：In order to solve the problems that the lack of divercity of the jellyfish search algorithm(JS) in ocean current motion，the lack of guidance in intra-swarm motion，and no exchange of information between populations，which result in slow search speed，poor stability，and susceptibility to precocious maturation，an oppositional-mutual learning jellyfish search algorithm based on mean-value example(OMLJS) is constructed. Firstly，in the part of the jellyfish following ocean currents(global search)，the average position of the previous two generations of jellyfish is used instead of only considering the average position of the previous generation to guide the position update of individual jellyfish，which improves the global search ability of the algorithm；secondly，in the part of jellyfish's intra-swarm active movement(local search)，the optimal individual is used instead of random individual to guide the jellyfish to perform more effective search，which accelerates the convergence speed of the algorithm；Then，before the jellyfish enters the next iteration，a dynamic reverse mutual learning step is added to the jellyfish population to increase the diversity of the population and enhance the information exchange between the populations，so as to complement the other two strategies and improve the overall optimization performance of the algorithm. Twelve classical benchmark test optimization functions are selected to compare and analyze OMLJS with five comparative algorithms in terms of the mean value，optimal value and variance of the solutions，and are used to solve the minimum spanning tree problem，where OMLJS is able to find the minimum spanning tree faster. The experimental results show that the convergence speed and solution accuracy of OMLJS are significantly improved.

Keywords：jellyfish search algorithm；mean-value example learning；oppositional-mutual learning；time control mechanism；minimum spanning tree problems

[责任编校　陈留院　赵晓华]

附　录

表 S1　测试函数的实验结果比较

Tab. S1　Comparison of test function results

函数	统计值	对比算法					
		HCLPSO	MFCPSO	MPEDE	JS	MJSO	OMLJS
f_1	平均值	4.864 6e-09	2.049 0e-06	3.113 9e-28	1.132 1e-64	1.059 7e-99	**0**
	最优值	9.978 5e-10	6.489 4e-12	5.545 9e-31	4.936 9e-75	5.554 3e-110	**0**
	方差	4.063 1e-09	8.323 7e-06	6.612 9e-28	4.316 4e-64	4.735 8e-99	**0**
f_2	平均值	46.114 3	31.076 5	13.432 1	0.013 2	1.203 2e-12	**0**
	最优值	5.228 6	26.008 1	0.041 1	4.140 7e-05	3.571 5e-15	**0**
	方差	34.834 4	17.618 8	16.309 1	0.029 0	2.970 6e-12	**0**
f_3	平均值	2.218 8e-05	8.072 2e-05	0.563 2	3.197 4e-15	3.730 3e-15	**8.881 8e-16**
	最优值	1.199 8e-05	2.432 0e-06	7.993 6e-15	**8.881 8e-16**	**8.881 8e-16**	**8.881 8e-16**
	方差	9.493 3e-06	1.541 2e-04	0.664 9	1.738 6e-15	1.458 0e-15	**0**
f_4	平均值	**0**	**0**	**0**	6.966 6e-16	**0**	**0**
	最优值	**0**	**0**	**0**	**0**	**0**	**0**
	方差	**0**	**0**	**0**	3.115 6e-15	**0**	**0**
f_5	平均值	2.668 3e-10	8.9121 9e-10	2.220 4e-16	**0**	**0**	**0**
	最优值	2.078 5e-11	2.561 6e-12	2.220 4e-16	**0**	**0**	**0**
	方差	3.033 8e-10	1.122 2e-09	**0**	**0**	**0**	**0**
f_6	平均值	15.930 6	17.700 6	0.348 2	5.679 9e-05	3.375 1e-15	**0**
	最优值	10.944 8	6.948 6	**0**	5.061 4e-10	**0**	**0**
	方差	3.581 9	6.132 7	0.667 4	9.869 5e-04	2.817 5e-15	**0**
f_7	平均值	1.8978e-05	4.9152e-05	5.3214e-14	6.8357e-30	9.6771e-48	**0**
	最优值	4.7975e-06	1.7488e-06	6.0608e-16	2.6777e-36	1.0464e-52	**0**
	方差	2.254 6e-05	7.665 9e-05	9.788 9e-14	3.089 7e-29	2.831 2e-47	**0**
f_8	平均值	2.018 2e-09	1.885 1e-08	4.535 3e-27	1.512 9e-38	6.408 9e-78	**0**
	最优值	2.226 3e-10	1.145 2e-11	7.484 2e-30	2.191 4e-57	4.171 3e-86	**0**
	方差	1.744 9e-09	4.508 5e-08	1.080 4e-26	6.534 5e-38	1.886 3e-77	**0**
f_9	平均值	**−450.000 0**	**−450.000 0**	**−450.000 0**	−499.983 1	**−450.000 0**	**−450.000 0**
	最优值	**−450.000 0**	**−450.000 0**	**−450.000 0**	−499.999 0	**−450.000 0**	**−450.000 0**
	方差	5.283 9e-09	1.427 7e-06	5.376 8e-14	0.017 2	9.931 6e-14	**3.703 4e-12**
f_{10}	平均值	−319.938 2	−53.000 0	−449.984 9	1.134 5e+03	**−450.000 0**	**−450.000 0**
	最优值	−402.496 8	−324.028 2	−449.999 4	308.592 5	**−450.000 0**	**−450.000 0**
	方差	72.364 1	185.283 9	0.053 1	762.785 0	2.519 7e-05	**2.891 8e-06**
f_{11}	平均值	536.349 8	3.173 7e+03	410.130 6	6.262 6e+03	403.218 8	**392.409 4**
	最优值	411.248 0	415.870 4	390.001 7	908.915 9	**390.000 0**	**390.000 0**
	方差	174.475 6	1.170 2e+04	21.106 7	6.800 3e+03	23.043 5	**3.279 5**
f_{12}	平均值	−118.986 8	−118.976 1	−118.965 1	−118.972 0	−119.005 8	**−119.990 0**
	最优值	−119.177 2	−119.120 8	−119.095 9	−119.066 3	−119.089 8	**−119.999 6**
	方差	0.084 1	0.056 8	0.049 5	0.037 8	0.043 2	**0.011 9**

表 S2　改进策略有效性实验结果比较

Tab. S2　Comparison of improved strategy effectiveness results

函数	统计值	对比算法				
		JS	JS1	JS2	JS3	OMLJS
f_1	平均值	9.147 1e-66	1.605 3e-85	8.107 1e-125	**0**	**0**
	最优值	4.571 2e-77	1.047 0e-94	6.049 8e-133	**0**	**0**
	方差	4.013 2e-65	7.034 3e-85	3.396 7e-124	**0**	**0**
f_2	平均值	0.001 7	5.859 9e-13	4.117 0e-10	3.430 3e-05	**0**
	最优值	9.466 8e-06	**0**	108 341e-13	6.963 1e-08	**0**
	方差	0.002 5	2.620 6e-12	1.072 3e-09	3.368 7e-05	**0**
f_4	平均值	**0**	**0**	**0**	**0**	**0**
	最优值	**0**	**0**	**0**	**0**	**0**
	方差	**0**	**0**	**0**	**0**	**0**
f_6	平均值	6.402 1e-05	**0**	3.481 7e-14	**0**	**0**
	最优值	707 484e-10	**0**	8.881 8e-15	**0**	**0**
	方差	2.011 8e-04	**0**	4.196 3e-14	**0**	**0**
f_7	平均值	1.283 6e-28	1.824 8e-42	6.771 3e-56	3.711 1e-271	**0**
	最优值	3.475 1e-37	9.122 1e-49	9.142 5e-61	2.175 4e-280	**0**
	方差	5.501 0e-28	4.895 7e-42	2.628 1e-55	**0**	**0**
f_8	平均值	6.834 3e-31	1.337 8e-53	6.769 3e-102	**0**	**0**
	最优值	2.643 3e-56	6.624 8e-81	6.258 2e-117	**0**	**0**
	方差	3.056 4e-30	5.982 8e-53	3.027 3e-101	**0**	**0**

表 S3　最小生成树问题的结果比较

Tab. S3　Result comparisons for minimum spanning tree problems

例子	统计值	对比算法			
		MFCPSO	JS	M JSO	OMLJS
例 1	平均值	797.120 6	931.661 4	**776.568 5**	**776.568 5**
	最优值	**776.568 5**	826.274 2	**776.568 5**	**776.568 5**
	方差	24.149 1	51.323 7	2.332 8e-13	**0**
例 2	平均值	671.426 2	756.016 2	664.213 0	**663.938 8**
	最优值	**663.938 8**	683.484 4	**663.938 8**	**663.938 8**
	方差	11.254 5	45.241 0	1.226 2	**0**
例 3	平均值	541.532 8	654.549 9	539.169 4	**536.941 8**
	最优值	**534.304 3**	540.486 5	**534.304 3**	**534.304 3**
	方差	9.983 1	63.645 8	4.986 5	**4.134 7**
例 4	平均值	532.430 3	651.646 7	525.327 0	**519.421 1**
	最优值	**514.364 1**	537.923 3	**514.364 1**	**514.364 1**
	方差	12.102 6	61.307 8	7.793 5	**7.128 1**