多子群的共生非均匀高斯变异樽海鞘群算法

陈忠云 [1] 张达敏 [1] 辛梓芸 [1]

摘 要 针对樽海鞘群算法求解精度不高和收敛速度慢等缺点, 提出一种多子群的共生非均匀高斯变异樽海鞘群算法. 根据不同适应度值将樽海鞘链群分为三个子种群, 各个子种群分别进行领导者位置更新、追随者共生策略和链尾者非均匀高斯变异等操作. 使用统计分析、收敛速度分析、Wilcoxon 检验、经典基准函数和 CEC 2014 函数的标准差来评估改进樽海鞘群算法的效率. 结果表明, 改进算法具有更好的寻优精度和收敛速度. 尤其在求解高维和多峰测试函数上, 改进算法拥有更好性能.

关键词 樽海鞘群算法, 多子群, 共生策略, 非均匀高斯变异, 函数优化

引用格式 陈忠云, 张达敏, 辛梓芸. 多子群的共生非均匀高斯变异樽海鞘群算法. 自动化学报, 2022, **48**(5): 1307−1317

DOI 10.16383/j.aas.c190684

Multi-subpopulation Based Symbiosis and Non-uniform Gaussian Mutation Salp Swarm Algorithm

CHEN Zhong-Yun[1] ZHANG Da-Min[1] XIN Zi-Yun[1]

Abstract In order to solve the problem that the standard salp swarm algorithm has a small convergent rate and low result precision in the evolutionary process, an improved algorithm called multi-subpopulation based symbiosis and non-uniform Gaussian mutation salp swarm algorithm (MSNSSA) is proposed in this paper. According to the different fitness values, the salp chain population is divided into three sub-populations, which perform the operations of leader position update, follower symbiosis strategy and chain tail non-uniform Gaussian mutation, respectively. The efficiency of the MSNSSA is evaluated using statistical analysis, convergence rate analysis, Wilcoxon′s test, standard deviations on classical benchmark functions and modern CEC 2014 functions. The results show that the MSNSSA has better optimization accuracy and convergence rate. Especially, in solving the high-dimension and multimodal function optimization problem, the improved algorithm has better performance.

Key words Salp swarm algorithm, multi-subpopulation, symbiosis strategy, non-uniform gaussian mutation, function optimization

Citation Chen Zhong-Yun, Zhang Da-Min, Xin Zi-Yun. Multi-subpopulation based symbiosis and non-uniform Gaussian mutation salp swarm algorithm. *Acta Automatica Sinica*, 2022, **48**(5): 1307−1317

近年, 元启发式算法作为一种有效的演化计算技术, 已受到众多学者的重视. 元启发式算法是指受到生物行为和物理现象的启发提出的一类算法, 其核心思想是实现搜索过程中随机性行为和局部搜索的平衡. 常用的元启发式算法包括粒子群算法 (Particle swarm optimization, PSO)[1]、正弦余弦算法 (Sine cosine algorithm, SCA)[2]、蝴蝶优化算法 (Butterfly optimization algorithm, BOA)[3]、飞蛾扑火优化算法 (Moth-flame optimization algorithm, MFO)[4]、大红斑蝶优化算法 (Monarch butterfly optimization, MBO)[5]、蚯蚓优化算法[6]、大象放牧优化 (Elephant herding optimization, EHO)[7] 等. 这些算法已成功应用于各种科学领域, 如过程控制、生物医学信号处理、图像处理以及许多其他工程设计问题.

樽海鞘群算法 (Salp swarm algorithm, SSA)[8] 提出的一种新型启发式智能算法. 樽海鞘群算法相对于粒子群算法等其他算法, 具有结构简单、参数少、容易实现等优点. 虽然樽海鞘群算法在求解大部分优化问题具有优越性, 但与其他群智能算法一样, 仍然存在求解精度低和收敛速度慢等缺陷. 文献 [9] 提出固定惯性权重, 可以加快搜索过程中的收敛速度, 并应用于特征选择问题. 文献 [10] 把樽海鞘群算法和混沌理论结合提出混沌樽海鞘群算法, 在解决特征提取问题时, 能发现最优特征子集,

收稿日期 2019-09-30 录用日期 2020-04-16
Manuscript received September 30, 2019; accepted April 16, 2020
贵州省自然科学基金 ([2017]1047) 资助
Supported by Natural Science Foundation of Guizhou Province ([2017]1047)
本文责任编委 孙秋野
Recommended by Associate Editor SUN Qiu-Ye
1. 贵州大学大数据与信息工程学院 贵阳 550025
1. School of Big Date and Information Engineering, Guizhou University, Guiyang 550025

最大程度地提高分类精度, 最小化所选特征的数目. 文献 [11] 提出采用子群规模调整, 每个子种群的大小随着进化的过程而逐渐增加, 有利于提高算法在初始阶段的探测能力和后期的开采能力. 文献 [12] 在算法中加入共享机制, 改进原始算法的随机追踪的位置更新公式, 降低搜索盲目性, 提高收敛速度. 文献 [13] 提出非均匀变异演化算法, 使个体能够跳出局部最优, 以克服早熟现象. 文献 [14] 通过高斯变异来增强蝙蝠算法种群的多样性.

为解决标准樽海鞘群算法存在的求解精度不高和收敛速度慢等问题, 本文提出了一种多子群的共生非均匀高斯变异樽海鞘群算法 (Multi-subpopulation based symbiosis and non-uniform Gaussian mutation salp swarm algorithm, MSNSSA). 根据适应度值大小, 将种群分为领导者、追随者和链尾者三个子群. 首先对领导者位置更新公式中参数 c_1 进行分析, 以更好平衡探索和开发能力; 然后对追随者位置更新公式采用共生策略, 增加与最优个体的交流, 增强局部开发能力; 最后链尾者更新使用非均匀高斯变异, 增强种群多样性. 通过求解 14 个典型测试函数和 CEC 2014 测试函数的最优解, 验证了改进算法 MSNSSA 的有效性和鲁棒性.

1 樽海鞘群算法

在樽海鞘群算法[8] 中, 樽海鞘链由领导者和追随者两种类型的樽海鞘组成. 领导者是位于樽海鞘链的最前面, 而其他个体则为追随者的角色. 随机初始化种群:

$$X_{N\times d} = \text{rand}(N,\, d) \times (ub - lb) + lb \tag{1}$$

其中, N 为樽海鞘群的规模, d 为空间维数.

在 SSA 中, 食物源的位置是所有樽海鞘个体的目标位置, 即探索过程中全局最优解, 影响着领导者位置更新. 领导者的位置更新公式如下所示:

$$x_j^i = \begin{cases} F_j + c_1((ub_j - lb_j)c_2 + lb_j),\ c_3 \geq 0.5 \\ F_j - c_1((ub_j - lb_j)c_2 + lb_j),\ c_3 < 0.5 \end{cases} \tag{2}$$

式中, x_j^i 为第 i 只樽海鞘 (领导者) 在第 j 维空间的位置; F_j 为食物源在第 j 维空间的位置, 即在搜索空间中整个樽海鞘群的搜索目标为食物源; ub_j 为在第 j 维空间的上限; lb_j 为在第 j 维空间的下限; c_2 和 c_3 为区间 [0, 1] 内产生的随机数.

由式 (2) 可知, 领导者位置更新主要受到食物源位置影响. 其中参数 c_1 定义如下:

$$c_1 = 2\mathrm{e}^{-(\frac{4t}{T_{\max}})^m} \tag{3}$$

式中, t 为当前迭代次数, $T_{\max}$ 为最大迭代次数, 幂系数 $m = 2$ [8]. 参数 c_1 在迭代过程中自适应降低, 当值较大时, 有助于提升探索能力. 而当值较小时, 则有助于具体开发能力. 系数 c_1 可以使 SSA 的探索能力和开发能力处于较好状态. 因而系数 c_1 是樽海鞘群算法中最重要的参数.

为更新追随者的位置, 使用以下公式:

$$x_j^i = \frac{at^2}{2} + v_0\Delta t + x_j^i \tag{4}$$

式中, $i \geq 2$, x_j^i 为第 i 只追随者在第 j 维空间的位置, Δt 为时间, v_0 为初速度, 加速度 $a = (v_t - v_0)/\Delta t$, 其中, $v_t = (x_j^{i-1} - x_j^i)/\Delta t$, x_j^{i-1} 为第 $i-1$ 只樽海鞘在 j 维空间的位置. 因为, 时间即为迭代次数之差, 故 $\Delta t = 1$, 并且初速度 $v_0 = 0$, 式 (4) 可表示为:

$$x_j^i := \frac{x_j^i + x_j^{i-1}}{2} \tag{5}$$

其中, 由式 (2) 和式 (5) 可以模拟樽海鞘链的行为机制.

2 多子群的共生非均匀高斯变异樽海鞘算法

智能算法的核心是为了寻求探索能力和开发能力的平衡, 为了实现这一点. 本文在标准樽海鞘群算法基础上, 以最小化问题为例, 保持原种群个体数目不变的条件下, 按照适应度值从小到大, 均匀地把原有种群分为领导者、追随者和链尾者三个子群体. 这三种子群体执行不同的更新策略, 分别侧重于平衡搜索、局部搜索和全局搜索. 并结合相应的进化策略, 改善算法的优化性能. 具体描述如下.

2.1 SSA 算法参数分析

领导者位置更新式 (2) 中, 参数 c_1 可以使 SSA 的探索能力和开发能力处于较好状态. 因而系数 c_1 是樽海鞘群算法中最重要的参数. 式中的幂系数 m 的取值大小对算法探索能力和开发能力起到至关重要的作用. 本文选取 m 从 1.0 到 3.5 之间分析对算法性能的影响. 图 1 为 m 取不同数值时, 随迭代次数增加参数 c_1 从 2 递减到 0 的变化曲线.

选取 Schaffer 测试函数对参数进行分析, 该函数在距全局最优点大约 3.14 范围内存在无穷多个局部极小将其包围, 并且函数强烈振荡. 实验时, 设定最大迭代次数为 1000, 改变幂系数 m 后对 Schaffer 测试函数进行 100 次寻优计算, 则 m 对算法性能的影响如表 1 所示.

如表 1 所示, 随着幂系数的增加, SSA 算法寻优的结果先增加后降低, 当 $m = 2.5$ 是寻优最优值

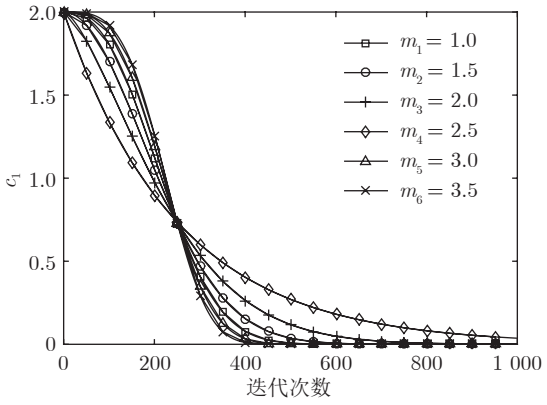

图 1　c_1 变化曲线

Fig. 1　c_1 change curve

表 1　参数 m 对 SSA 的影响

Table 1　Influence of parameter m on SSA

m	最佳值	平均值	标准差	平均收敛代数
1.0	2.98×10^{-8}	1.40×10^{-3}	2.51×10^{-3}	883
1.5	4.67×10^{-9}	2.68×10^{-3}	4.34×10^{-3}	937
2.0	1.20×10^{-14}	3.41×10^{-3}	4.64×10^{-3}	975
2.5	0	0	0	847
3.0	9.72×10^{-3}	9.72×10^{-3}	2.08×10^{-17}	719
3.5	9.72×10^{-3}	9.72×10^{-3}	1.99×10^{-17}	621

达到 0. 无论是平均值和标准差, $m = 2.5$ 时都为最佳. 随着 m 增加, 平均收敛代数在降低. 虽然此时 $m = 2.5$ 的平均收敛代数没有 $m = 3.0$ 和 3.5 时的效果好, 但是 $m = 2.5$ 的寻优结果要比其余都要好. 结合图 1 分析可知, 当 $m = 2.5$ 时, 迭代前期参数 c_1 变化较快, 能较好维持算法的全局搜索能力, 迭代后期 c_1 变化较慢, 能有效提高算法的局部寻优能力, 从而取得良好的寻优效果. 综上可知, 当 $m = 2.5$ 时, 参数 c_1 能使 SSA 算法的探索能力和开发能力处于更好的平衡状态.

2.2　共生策略

由式 (5) 追随者位置更新公式产生的新个体直接替换原个体, 此更新方式存在以下缺点: 1) 更新后的追随者个体不管适应度值好与坏都直接替换原有个体, 具有一定盲目性, 会导致收敛速度降低; 2) 第 i 只樽海鞘位置会根据第 i 和 $i-1$ 只樽海鞘位置进行更新, 对先前个体依懒性较强, 缺乏与其他个体学习的部分. 若追随者的位置是局部最优解, 则会容易陷入局部最优, 出现停滞. 为增强樽海鞘群算法的开发能力. 本文提出一种共生策略对追随者位置进行更新, 公式如下:

$$x_j^i = x_j^i + \text{rand}(0, 1) \times (F_j - C \times R) \tag{6}$$

式中, rand(0, 1) 是 (0, 1) 之间的随机数; F_j 为食物源在第 j 维空间的位置; $C = (x_j^i + x_j^{i-1})/2$ 称为共生量, 代表樽海鞘链中第 i 和 $i-1$ 只樽海鞘的关系特征; R 的作用解释如下: 在自然界中, 某些互惠关系可能给一个生物带来比另一个生物更大的有益优势. 换句话说, 樽海鞘 i 与樽海鞘 $i-1$ 相互作用时可能会获得巨大的利益. 同时, 樽海鞘 $i-1$ 与樽海鞘 i 相互作用时可能只会获得足够的利益或没有那么显着的利益. 式中受益因子 R 是随机确定的 1 或 2. 这些因素代表樽海鞘个体是部分或全部受益于相互作用. 经式 (6) 新产生的追随者需判断适应度值是否优劣后再替换原有个体.

在式 (6) 追随者位置更新策略中, 增加了社会部分 $\text{rand}(0, 1) \times (F_j - C \times R)$, 使种群中的最优个体参与追随者的进化. 与原来追随者更新式 (5) 只使用第 i 和 $i-1$ 只樽海鞘位置进行信息交流的方式相比, 社会部分引入更多组合模式, 使其不再单一围绕前一个追随者附近搜索, 即将追随者从原个体位置指引到食物源位置所在附近. 从而提高算法的开发能力.

2.3　非均匀高斯变异

处于樽海鞘链尾端个体的适应度值较差, 被分类为链尾者子群. 对于这类樽海鞘个体可以很好地保留自身信息, 而不是一味地朝着领导者移动. 为增强种群多样性, 本文为适应度值较差的链尾者提出一种非均匀高斯变异策略, 其更新如下:

$$x_j^i = x_j^i + \Delta(t, GD_t^i) \tag{7}$$

$$\Delta(t, y) = y(1 - r^{(\frac{1-t}{T_{\max}})^b}) \tag{8}$$

$$GD_t^i = N((F_j - x_j^i), \sigma) \tag{9}$$

式中, $\Delta(t, GD_t^t)$ 为非均匀变异步长, 是一种通过高斯分布 (GD_t^t) 自适应调节步长的变异算子; r 为 [0, 1] 区间均匀产生的随机数; b 为系统参数, 决定变异计算的非均匀程度, 取值为 $b = 2$ [13]; F_j 为食物源在第 j 维空间的位置; σ 为高斯分布标准差.

非均匀高斯变异策略有如下特色: 1) 其更新对象为种群中适应度最差的樽海鞘个体, 而不是当前种群中的全部个体, 降低变异计算复杂度. 2) 由式 (7) 可知, 对链尾者更新公式以个体自身为基准, 选择食物源位置与当前个体进行高斯分布后自适应调整变异步长的学习策略. 该更新方式有利于保持种群多样性, 进一步增强算法全局搜索能力.

综上所述, 通过把原有算法种群分类为领导者、

追随者和链尾者三个子群, 分别对其执行不同的搜索任务. 领导者子群负责平衡算法探索能力和开发能力. 追随者子群负责增强算法开发能力. 链尾者子群负责增强算法探索能力. 进化过程中每一个子群有针对性地进行搜索, 更适合自身的进化需求, 增强算法求解性能.

3 仿真实验与结果分析

为验证本文提出的 MSNSSA 在求解优化问题上的有效性和鲁棒性, 将 MSNSA 算法与加入共生策略的樽海鞘群算法 (记为 SSSA)、加入非均匀高斯变异的樽海鞘算法 (记为 NSSA)、SSA、PSO[1]、SCA[2]、BOA[3] 和 MFO[4] 在 14 个典型的标准测试函数[15] 最优值求解上独立进行 50 次对比实验.

实验 1. 采用如表 2 所示 14 个复杂函数作为适应度函数. 选取的测试函数中包含单峰、多峰、可分和不可分等不同特征的测试函数. 表 2中测试函数维度从 2 维到 200 维, 可以更加全面地验证算法性能.

表 2 基准函数

Table 2 Benchmark function

函数	维度	特征	定义域	最佳值
f_1 Sphere	10	US	[−100, 100]	0
f_2 Schwefel 1.2	50	UN	[−100, 100]	0
f_3 Schwefel 2.21	50	US	[−100, 100]	0
f_4 Quartic	100	US	[−1.28, 1.28]	0
f_5 Rosenbrock	100	UN	[−30, 30]	0
f_6 Step	200	US	[−100, 100]	0
f_7 Schaffer	2	MN	[−100, 100]	0
f_8 Foxholes	2	MN	[−65.56, 65.56]	≈1
f_9 Kowalik	4	MN	[−5, 5]	3.075×10^{-4}
f_{10} Rastrigin	10	MS	[−5.12, 5.12]	0
f_{11} Ackley	50	MN	[−32, 32]	0
f_{12} Griewank	100	MN	[−600, 600]	0
f_{13} Penalized1	100	MN	[−50, 50]	0
f_{14} Penalized2	200	MN	[−50, 50]	0

实验最大迭代次数为 1 000, 种群个数为 30, 各算法其余的参数设置如表 3 所示:

通过 50 次独立试验后从每种算法获得结果的最佳值、平均值和标准偏差以及求解成功率 SR% 和平均耗时 T 等多个实验对比数据如表 4 所示. 其中求解成功率为计算成功次数除以本次实验的求解次数. 判断一次求解是否成功按照下式:

表 3 参数设置

Table 3 Parameter settings

算法	主要参数
MSNSSA	$m = 2.5$, $R = 1$ 或 2
SSSA	$m = 2.0$, $R = 1$ 或 2
NSSA	$m = 2.0$
BOA	$p = 0.8$, $c = 0.01$, $a = 0.1$
MFO	—
SCA	$a = 2$
PSO	$c_1 = c_2 = 1.5$

$$\begin{cases} \dfrac{|F_A - F_T|}{F_T} < 10^{-5}, & F_T \neq 0 \\ |F_A - F_T| < 10^{-5}, & F_T = 0 \end{cases} \tag{10}$$

式中, F_A 为每次实际求解最佳值, F_T 为测试函数理论最佳值.

首先, 对于表 4 中的最优值、平均值都可以反应算法的收敛精度和寻优能力. 对于 6 个单峰函数 $f_1 \sim f_6$, MSNSSA 在求解 f_5 和 f_6 函数时, 寻优精度达到理论最优值 0. 同时, 随着搜索空间维度的增加, 寻优收敛精度 f_1 至 f_4 函数有所下降, 因为伴随求解维度的增加, 对于算法求解难度也呈指数级别递增, 所以算法的收敛精度有所降低. 随着维度增加 SSA、SCA、MFO 算法求解精度较差, 其中 SCA 算法求解 100 维的 Rosenbrock 时与理论最优值存在 1.0×10^7 级的误差. 对于只引入共生策略的 SSSA 和引入非均匀高斯变异的 NSSA 两种算法的寻优精度较和标准差都比原始 SSA 要好, 这说明引入不同策略可增强算法性能. 而 MSNSSA 相对与其他几种算法寻优精度和标准差都要好, 且求解部分单峰函数达到理论最优值. 对于 8 个多峰函数 $f_7 \sim f_{14}$, 算法求解精度相对于单峰函数要低一些. 同样, 随着维度增加算法求解精度也有所降低. 当维度增加到 200 维时, SSA 算法与理论最优值存在 1.0×10^2 级的误差. MSNSSA 算法在求解 8 个多峰函数中, 有 5 个都达到了理论最优值. 在求解函数 f_7 和 f_{10} 时, 包括 MSNSSA 在内的其余 4 种算法都达到了理论最优值, 但 MSNSSA 的平均值比其他几种算法都要好. 在求解函数 f_8 时, 8 种算法都达到了理论最优值, 但 MSNSSA、SSSA 和 NSSA 三种算法的平均值比其他算法好. 求解其他函数时, MSNSSA算法比其他算法的精度都较高. 另外 SSSA 和 NSSA 比标准 SSA 寻优结果基本较好, 进一步说明加入不同策略对算法性能有所提升. 可见 MSNSSA算法在求解单峰、多峰、可分、不可分以及高维的基准函数时都有明显的优势.

其次, 表 4 中标准差和成功率可以反映算法的

表 4　基准函数结果对比

Table 4　Comparison of benchmark function results

函数	算法	最佳值	平均值	标准差	SR (%)	T (s)
f_1	MSNSSA	$\mathbf{1.06 \times 10^{-34}}$	$\mathbf{5.01 \times 10^{-30}}$	$\mathbf{1.30 \times 10^{-29}}$	100	0.80
	SSSA	1.53×10^{-18}	1.18×10^{-11}	3.54×10^{-11}	100	0.73
	NSSA	3.46×10^{-24}	7.94×10^{-15}	2.36×10^{-14}	100	0.56
	SSA	2.52×10^{-10}	6.55×10^{-10}	2.35×10^{-10}	100	0.28
	BOA	1.04×10^{-14}	1.50×10^{-14}	1.83×10^{-15}	100	0.44
	MFO	1.06×10^{-32}	5.64×10^{-29}	2.35×10^{-28}	100	0.26
	SCA	7.48×10^{-32}	2.18×10^{-24}	1.02×10^{-23}	100	0.23
	PSO	4.39×10^{-5}	2.66×10^{-4}	2.10×10^{-4}	0	0.25
f_2	MSNSSA	$\mathbf{3.31 \times 10^{-30}}$	$\mathbf{2.84 \times 10^{-27}}$	$\mathbf{2.32 \times 10^{-27}}$	100	3.62
	SSSA	4.95×10^{-14}	1.75×10^{-9}	3.26×10^{-9}	100	4.79
	NSSA	1.91×10^{-21}	1.00×10^{-10}	4.20×10^{-10}	100	3.32
	SSA	1.29×10^{3}	4.81×10^{3}	2.25×10^{3}	0	2.98
	BOA	1.59×10^{-14}	1.85×10^{-14}	1.00×10^{-15}	100	5.67
	MFO	2.21×10^{4}	5.52×10^{4}	2.49×10^{4}	0	3.09
	SCA	1.08×10^{4}	3.41×10^{4}	1.46×10^{4}	0	3.05
	PSO	7.10×10^{1}	1.68×10^{2}	6.46×10^{1}	0	2.89
f_3	MSNSSA	$\mathbf{1.15 \times 10^{-17}}$	$\mathbf{2.71 \times 10^{-16}}$	$\mathbf{1.65 \times 10^{-16}}$	100	1.03
	SSSA	9.96×10^{-9}	3.87×10^{-6}	5.10×10^{-6}	82	0.99
	NSSA	3.74×10^{-13}	1.55×10^{-8}	3.20×10^{-8}	100	0.78
	SSA	1.11×10^{1}	1.82×10^{1}	3.61×10^{0}	0	0.47
	BOA	1.04×10^{-11}	1.21×10^{-11}	8.36×10^{-13}	100	0.65
	MFO	6.77×10^{1}	8.19×10^{1}	4.86×10^{0}	0	0.60
	SCA	3.65×10^{1}	5.85×10^{1}	8.18×10^{0}	0	0.55
	PSO	4.06×10^{0}	5.90×10^{0}	1.11×10^{0}	0	0.40
f_4	MSNSSA	$\mathbf{1.66 \times 10^{-6}}$	$\mathbf{1.16 \times 10^{-5}}$	$\mathbf{5.52 \times 10^{-6}}$	38	1.78
	SSSA	2.58×10^{-5}	1.66×10^{-4}	1.38×10^{-4}	0	2.03
	NSSA	2.09×10^{-5}	2.84×10^{-4}	2.90×10^{-4}	0	1.53
	SSA	8.23×10^{-1}	1.36×10^{0}	3.27×10^{-1}	0	1.18
	BOA	1.36×10^{-4}	6.89×10^{-4}	3.33×10^{-4}	0	2.13
	MFO	2.37×10^{1}	1.78×10^{2}	1.20×10^{2}	0	1.52
	SCA	3.37×10^{0}	5.93×10^{1}	4.20×10^{1}	0	1.43
	PSO	2.10×10^{0}	4.48×10^{0}	2.24×10^{0}	0	1.01
f_5	MSNSSA	**0**	$\mathbf{3.99 \times 10^{-28}}$	$\mathbf{3.80 \times 10^{-28}}$	100	1.33
	SSSA	9.77×10^{1}	9.78×10^{1}	8.27×10^{-2}	0	1.30
	NSSA	4.44×10^{-27}	2.48×10^{-12}	4.33×10^{-12}	100	1.05
	SSA	6.60×10^{2}	2.20×10^{3}	2.71×10^{3}	0	0.68
	BOA	9.89×10^{1}	9.89×10^{1}	2.86×10^{-2}	0	1.23
	MFO	3.03×10^{6}	7.18×10^{7}	5.39×10^{7}	0	1.02
	SCA	2.19×10^{7}	6.79×10^{7}	3.34×10^{7}	0	0.93
	PSO	6.47×10^{2}	1.12×10^{3}	3.10×10^{2}	0	0.52
f_6	MSNSSA	**0**	$\mathbf{4.56 \times 10^{-29}}$	$\mathbf{3.92 \times 10^{-29}}$	100	1.58
	SSSA	3.39×10^{0}	4.93×10^{0}	6.31×10^{-1}	0	1.48
	NSSA	2.85×10^{-27}	2.27×10^{-13}	7.95×10^{-13}	100	1.30
	SSA	2.16×10^{3}	3.34×10^{3}	6.74×10^{2}	0	0.87
	BOA	4.58×10^{1}	4.83×10^{1}	8.80×10^{-1}	0	1.06
	MFO	1.27×10^{5}	1.82×10^{5}	2.02×10^{4}	0	1.61
	SCA	5.91×10^{3}	2.88×10^{4}	1.53×10^{4}	0	1.44

表 4　基准函数结果对比 (续表)

Table 4　Comparison of benchmark function results (continued table)

函数	算法	最佳值	平均值	标准差	SR (%)	T (s)
f_6	PSO	4.43×10^{1}	7.07×10^{1}	1.11×10^{1}	0	0.63
f_7	MSNSSA	**0**	**0**	**0**	100	0.82
	SSSA	**0**	8.61×10^{-13}	1.54×10^{-12}	100	0.80
	NSSA	**0**	1.60×10^{-14}	2.62×10^{-14}	100	0.59
	SSA	3.66×10^{-14}	5.25×10^{-3}	4.89×10^{-3}	46	0.33
	BOA	1.68×10^{-14}	1.04×10^{-2}	4.35×10^{-3}	2	0.93
	MFO	**0**	8.16×10^{-3}	3.60×10^{-3}	16	0.27
	SCA	**0**	4.92×10^{-7}	3.48×10^{-6}	98	0.25
	PSO	6.15×10^{-9}	3.30×10^{-3}	4.65×10^{-3}	62	0.30
f_8	MSNSSA	$\mathbf{9.98 \times 10^{-1}}$	$\mathbf{9.98 \times 10^{-1}}$	$\mathbf{6.34 \times 10^{-17}}$	100	2.68
	SSSA	$\mathbf{9.98 \times 10^{-1}}$	$\mathbf{9.98 \times 10^{-1}}$	2.21×10^{-16}	100	3.51
	NSSA	$\mathbf{9.98 \times 10^{-1}}$	$\mathbf{9.98 \times 10^{-1}}$	1.21×10^{-16}	100	2.40
	SSA	$\mathbf{9.98 \times 10^{-1}}$	1.22×10^{0}	6.11×10^{-1}	86	2.11
	BOA	$\mathbf{9.98 \times 10^{-1}}$	1.16×10^{0}	3.69×10^{-1}	26	4.63
	MFO	$\mathbf{9.98 \times 10^{-1}}$	2.28×10^{0}	2.05×10^{0}	52	2.05
	SCA	$\mathbf{9.98 \times 10^{-1}}$	1.51×10^{0}	8.79×10^{-1}	34	2.03
	PSO	$\mathbf{9.98 \times 10^{-1}}$	1.38×10^{0}	8.44×10^{-1}	66	2.08
f_9	MSNSSA	$\mathbf{3.07 \times 10^{-4}}$	$\mathbf{3.08 \times 10^{-4}}$	$\mathbf{1.47 \times 10^{-7}}$	8	0.96
	SSSA	$\mathbf{3.07 \times 10^{-4}}$	4.29×10^{-4}	1.56×10^{-4}	0	0.98
	NSSA	3.08×10^{-4}	4.48×10^{-4}	2.29×10^{-4}	0	0.71
	SSA	4.83×10^{-4}	3.22×10^{-3}	6.40×10^{-3}	0	0.45
	BOA	3.12×10^{-4}	3.66×10^{-4}	5.91×10^{-5}	0	1.17
	MFO	3.20×10^{-4}	9.25×10^{-4}	3.45×10^{-4}	0	0.39
	SCA	3.27×10^{-4}	8.72×10^{-4}	3.79×10^{-4}	0	0.37
	PSO	3.22×10^{-4}	1.15×10^{-3}	2.80×10^{-3}	0	0.41
f_{10}	MSNSSA	**0**	**0**	**0**	100	0.95
	SSSA	**0**	9.59×10^{-11}	1.35×10^{-10}	100	0.95
	NSSA	**0**	6.62×10^{-14}	1.25×10^{-13}	100	0.71
	SSA	1.99×10^{0}	5.33×10^{0}	1.27×10^{0}	0	0.43
	BOA	**0**	2.93×10^{1}	1.81×10^{1}	20	1.10
	MFO	5.97×10^{0}	2.32×10^{1}	1.22×10^{1}	0	0.40
	SCA	**0**	6.31×10^{-1}	3.16×10^{0}	90	0.38
	PSO	3.01×10^{0}	1.01×10^{1}	4.25×10^{0}	0	0.38
f_{11}	MSNSSA	$\mathbf{4.44 \times 10^{-15}}$	$\mathbf{2.01 \times 10^{-14}}$	$\mathbf{1.77 \times 10^{-14}}$	100	1.15
	SSSA	1.14×10^{-8}	1.46×10^{-6}	2.17×10^{-6}	100	1.16
	NSSA	$\mathbf{4.44 \times 10^{-15}}$	4.74×10^{-8}	7.31×10^{-8}	100	0.89
	SSA	1.56×10^{0}	1.97×10^{0}	1.58×10^{-1}	0	0.59
	BOA	1.06×10^{-11}	1.22×10^{-11}	5.97×10^{-13}	100	1.25
	MFO	1.09×10^{1}	1.91×10^{1}	1.81×10^{0}	0	0.72
	SCA	3.55×10^{-2}	1.70×10^{1}	7.24×10^{0}	0	0.69
	PSO	4.48×10^{0}	6.78×10^{0}	1.03×10^{0}	0	0.50
f_{12}	MSNSSA	**0**	**0**	**0**	100	1.50
	SSSA	1.19×10^{-14}	4.83×10^{-10}	1.18×10^{-9}	100	1.57
	NSSA	**0**	1.93×10^{-13}	4.31×10^{-13}	100	1.22
	SSA	2.11×10^{-1}	3.28×10^{-1}	3.39×10^{-2}	0	0.86
	BOA	3.11×10^{-15}	1.34×10^{-14}	6.68×10^{-15}	100	1.51

表 4 基准函数结果对比 (续表)
Table 4 Comparison of benchmark function results (continued table)

函数	算法	最佳值	平均值	标准差	SR (%)	T (s)
f_{12}	MFO	3.82×10^{1}	2.80×10^{2}	1.19×10^{2}	0	1.19
	SCA	9.68×10^{0}	5.36×10^{1}	4.38×10^{1}	0	1.12
	PSO	8.91×10^{1}	1.17×10^{2}	1.36×10^{1}	0	0.79
f_{13}	MSNSSA	$\mathbf{4.78 \times 10^{-33}}$	$\mathbf{4.97 \times 10^{-29}}$	$\mathbf{9.06 \times 10^{-29}}$	100	4.34
	SSSA	3.01×10^{-2}	5.40×10^{-2}	1.13×10^{-2}	0	5.63
	NSSA	4.83×10^{-28}	3.29×10^{-16}	5.56×10^{-16}	100	4.00
	SSA	3.04×10^{1}	4.57×10^{1}	9.46×10^{0}	0	3.61
	BOA	9.80×10^{-1}	1.11×10^{0}	5.44×10^{-2}	0	6.74
	MFO	7.21×10^{8}	1.31×10^{9}	2.86×10^{8}	0	4.38
	SCA	3.71×10^{8}	1.01×10^{9}	3.02×10^{8}	0	4.24
	PSO	3.75×10^{0}	5.75×10^{0}	1.01×10^{0}	0	3.30
f_{14}	MSNSSA	$\mathbf{1.35 \times 10^{-32}}$	$\mathbf{3.61 \times 10^{-27}}$	$\mathbf{7.51 \times 10^{-27}}$	100	2.88
	SSSA	5.36×10^{0}	9.72×10^{0}	8.22×10^{-1}	0	3.61
	NSSA	2.50×10^{-26}	1.18×10^{-14}	4.32×10^{-14}	100	2.56
	SSA	1.34×10^{2}	1.76×10^{2}	2.38×10^{1}	0	2.22
	BOA	9.98×10^{0}	9.99×10^{0}	4.50×10^{-3}	0	4.21
	MFO	8.43×10^{6}	2.38×10^{8}	1.98×10^{8}	0	2.51
	SCA	8.61×10^{7}	3.31×10^{8}	1.46×10^{8}	0	2.49
	PSO	1.07×10^{2}	1.46×10^{2}	2.28×10^{1}	0	1.99

稳定性和跳出局部最优的能力. MSNSSA 算法独立 50 次计算的都很接近理论最优值、标准差也较小. 说明 MSNSSA 的寻优求解有着一定的稳定性. 另外, MSNSSA 算法标准差始终都要比另外几种算法要优秀, 进一步验证了 MSNSSA 的有效性. 14 个基准函数中有单峰、多峰、低维和高维, 除了 f_4 和 f_9 函数, MSNSSA 在成功率基本上为 100%, 而标准 SSA 在 f_1 函数的成功率为 100% 以外, 其余基准函数成功率几乎为 0. 随着搜索维度的增加, 标准 SSA 在寻优求解能力上表现出很大不足. 特别是在求解多维函数时, 最优值和成功率都很差, 说明标准 SSA 在跳出局部最优的能力较弱. 而在 MSNSSA 和 SSSA 中都引入共生策略, 这对算法跳出局部搜索具有很大作用.

从平均耗时来看. 由表 4 可知, SCA 和 PSO 平均耗时最短, 改进的 MSNSSA、SSSA、NSSA 这 3 种算法相对于标准 SSA 的平均耗时都要大, 此种情况也在合理范围内. 因为算法中引入更多的算子, 使得算法能够搜索到更多解, 导致运行时间变长. 总体来看, MSNSSA 平均耗时比另外两种算法增加的并不是很大, 在允许范围内.

图 2 给出 6 个基准函数的平均收敛曲线图, 各函数分图图例同图 2(f) 一致. 由于 MSNSSA 收敛精度较高, 为了便于观察收敛情况, 本文对寻优适应度值 (纵坐标) 取以 10 为底的对数. 由图 2可以看出, 在迭代前期, MSNSSA 和 SSSA 两个算法收敛曲线下降很快, 这说明引入共生策略, 增加种群局部探索能力, 使得算法一开始收敛速度就较快. 随着更迭次数的增加 MSNSSA 算法持续寻优, 在迭代后期也未出现停止状况, 收敛速度比其他算法都要快, 且寻优精度较高. 另外, 在迭代后期 MSNSSA 比只加入非均匀高斯变异的 NSSA 收敛速度要快, 说明非均匀高斯变异增强了种群多样, 验证了改进算法的有效性. MSNSSA 在函数 f_7 和 f_{10} 上寻优精度达到理论最优值 0. 而标准 SSA 算法收敛速度慢, 前期和后期都出现不同程度的停滞. 从图 2 可知, 其他算法在前期和后期都出现停滞现象.

无论单峰、多峰, 还是低维和高维, 对于每个函数 MSNSSA 比另外 7 种算法的收敛速度和寻优精度都要好. 由表 4 可知, 对于函数 f_7 和 f_{10}, MSNSSA 的最佳值为 0. 所以在图 2(d) 和图 2(e) 中, MSNSSA 曲线后面部分没有显示.

基于 50 次独立运行算法的平均值和标准差二者之间不相互对比每次运行结果. 文献 [16] 提出对于改进进化算法性能的评估, 应该进行统计检验. 换而言之, 仅基于平均值和标准偏差值来比较算法还不够. 需要进行统计检验以验证所提出的改进算法比其他现有算法具有显著的改进优势. 为了判断 MSNSSA 的每次结果是否统计上显著的与其他算法的最佳结果不同, 采用 Wilcoxon 秩和检验在 5% 的显著性水平下进行. 表 5 给出所有基准函数的 MSNSSA 和其他算法的 Wilcoxon 秩和检验中计算的 p 值. 例如如果最佳算法是 MSNSSA, 则在 MSNSSA 与 SSSA, MSNSSA 与 SSA 等之间进行成对比较. 由于最佳算法无法与自身进行比较, 因此, 针对每个函数中的最佳算法标记为 N/A, 表示不适用. 这意味着相应的算法可以在秩和检验中没有统计数据与自身进行比较. 符号 “+”、“−”和 “=”分别表示 MSNSSA 的性能要优于、劣于和相当于对比算法. 根据文献 [16], 当 $p < 0.05$ 时,可以被认为是拒绝零假设的有力验证.

由表 5 可知, MSNSSA 的 p 值基本小于 0.05. 只有在 f_7 的 MSNSSA 与 SSA 时, p 值大于 0.05. 这表明该算法的优越性在统计上是显著的. 即 MSNSSA 算法比其他算法具有更高的收敛精度.

所有算法的定量分析是基于 14 个基准函数的平均绝对误差 (Mean absolute error, MAE). 文献 [17] 提出对优化算法进行排序, MAE 是一种有

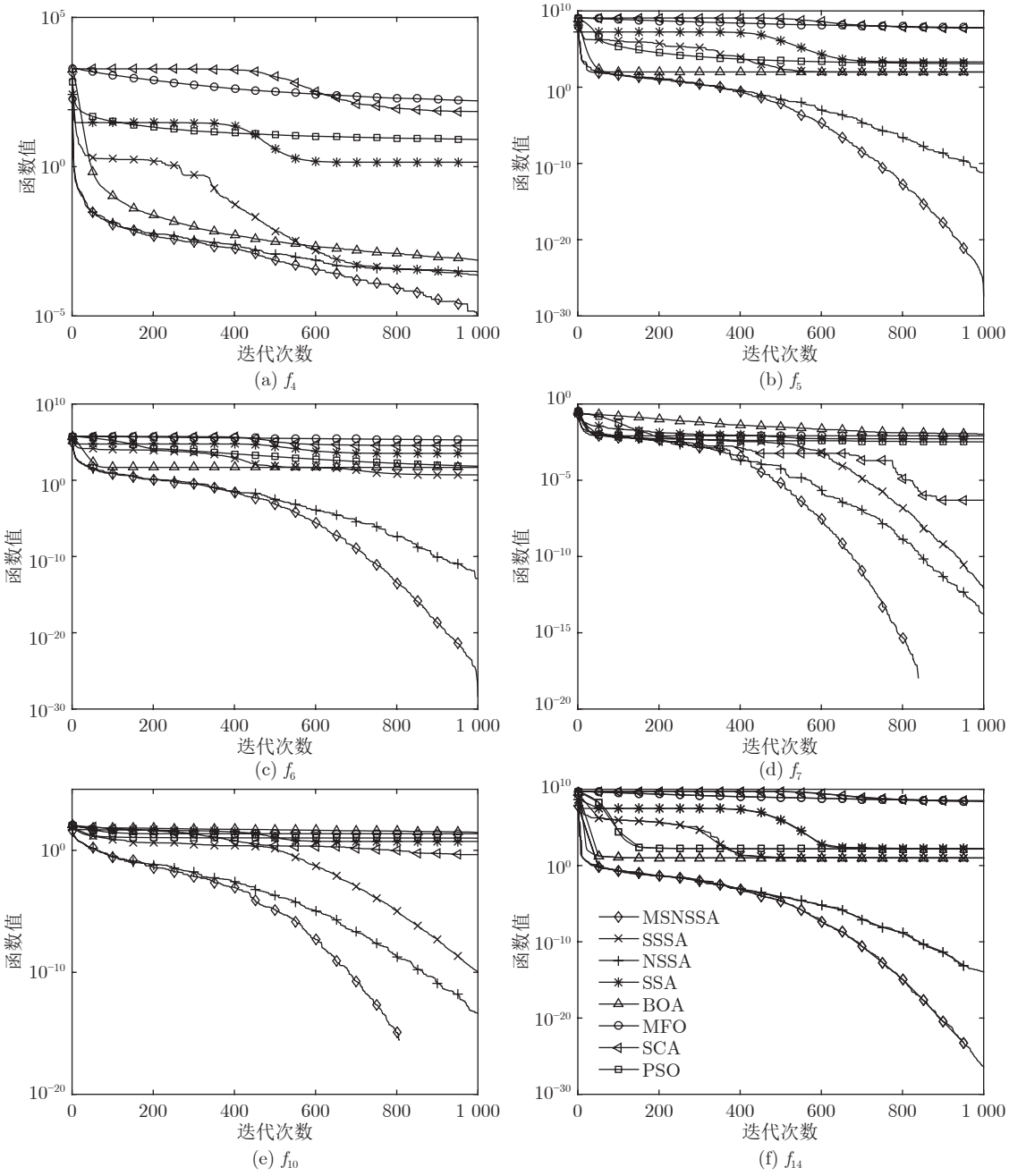

图 2　基准函数平均收敛曲线

Fig. 2　Function average convergence curves

效且可行的性能指标. 表 6 给出了这些基准函数的 MAE 排序, 计算公式如下:

$$MAE = \sum_{i=1}^{N_f} \frac{|m_i - o_i|}{N_f} \tag{11}$$

式中, m_i 为算法产生的最优结果的平均值, o_i 为相应基准函数的理论最优值, N_f 为基准函数个数. 计算值见表 6.

由表 6 可知, MSNSSA 算法排名为 1. 与另外 7 种算法相比, MSNSSA 算法提供最小的 MAE, 进一步说明 MSNSSA 算法的有效性. NSSA 和 SSSA 分别排第 2 名和第 3 名.

实验 2. 为更好评估 MSNSSA 的有效性和稳定性. 本文还在 CEC 2014 基准函数中选取部分单峰、多峰、混合和复合类型的函数进行优化求解, 如表 7 所示. 实验种群规模为 30, 最大迭代次数为 1 000, 维度为 30.

表 8 记录了 CEC 2014 中部分函数独立运行

表 5 基准函数 Wilcoxon 秩和检验的 p 值

Table 5 p-value for Wilcoxon′s rank-sum test on benchmark function

函数	SSSA	NSSA	SSA	BOA	MFO	SCA	PSO
f_1	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$3.92\times10^{-5}+$	$1.16\times10^{-13}+$	$7.05\times10^{-18}+$
f_2	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$
f_3	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$
f_4	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$
f_5	$4.26\times10^{-18}+$	$4.26\times10^{-18}+$	$4.26\times10^{-18}+$	$4.26\times10^{-18}+$	$4.26\times10^{-18}+$	$4.26\times10^{-18}+$	$4.26\times10^{-18}+$
f_6	$6.79\times10^{-18}+$	$6.79\times10^{-18}+$	$6.79\times10^{-18}+$	$6.79\times10^{-18}+$	$6.79\times10^{-18}+$	$6.79\times10^{-18}+$	$6.79\times10^{-18}+$
f_7	$1.26\times10^{-19}+$	$5.96\times10^{-18}+$	$3.23\times10^{-20}+$	$3.31\times10^{-20}+$	$2.61\times10^{-17}+$	$\mathbf{8.22\times10^{-2}}-$	$3.31\times10^{-20}+$
f_8	$3.51\times10^{-18}+$	$4.12\times10^{-19}+$	$1.25\times10^{-20}+$	$1.23\times10^{-19}+$	$1.86\times10^{-6}+$	$1.23\times10^{-19}+$	$1.23\times10^{-19}+$
f_9	$4.28\times10^{-11}+$	$9.53\times10^{-17}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$	$7.06\times10^{-18}+$	$7.07\times10^{-18}+$	$7.07\times10^{-18}+$
f_{10}	$4.67\times10^{-19}+$	$1.14\times10^{-12}+$	$3.31\times10^{-20}+$	$1.69\times10^{-18}+$	$3.30\times10^{-20}+$	$1.82\times10^{-3}+$	$3.31\times10^{-20}+$
f_{11}	$5.90\times10^{-18}+$	$6.96\times10^{-17}+$	$5.90\times10^{-18}+$	$5.90\times10^{-18}+$	$5.90\times10^{-18}+$	$5.90\times10^{-18}+$	$5.90\times10^{-18}+$
f_{12}	$3.31\times10^{-20}+$	$1.84\times10^{-10}+$	$3.31\times10^{-20}+$	$3.29\times10^{-20}+$	$3.31\times10^{-20}+$	$3.31\times10^{-20}+$	$3.31\times10^{-20}+$
f_{13}	$7.04\times10^{-18}+$	$7.48\times10^{-18}+$	$7.04\times10^{-18}+$	$7.04\times10^{-18}+$	$7.04\times10^{-18}+$	$7.04\times10^{-18}+$	$7.04\times10^{-18}+$
f_{14}	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$	$7.05\times10^{-18}+$
+/=/−	12 / 0 / 0	12 / 0 / 0	12 / 0 / 0	12 / 0 / 0	12 / 0 / 0	11 / 0 / 1	12 / 0 / 0

表 6 MAE 算法排名

Table 6 MAE algorithm ranking

算法	*MAE*	排名
MSNSSA	7.12641×10^{-2}	1
NSSA	7.12655×10^{-2}	2
SSSA	7.67841×10^{0}	3
BOA	1.11882×10^{1}	4
PSO	6.98046×10^{1}	5
SSA	3.06178×10^{2}	6
SCA	3.42235×10^{7}	7
MFO	5.23029×10^{7}	8

表 7 CEC 2014 基准函数

Table 7 CEC 2014 benchmark function

函数	维度	特征	定义域	最佳值
CEC03	30	UN	[−100, 100]	300
CEC05	30	MN	[−100, 100]	500
CEC18	30	HF	[−100, 100]	1800
CEC23	30	CF	[−100, 100]	2300
CEC24	30	CF	[−100, 100]	2400
CEC25	30	CF	[−100, 100]	2500

30 次后每种算法的平均值和标准差的结果. CEC 2014 函数具有复杂的特征, 因此所有算法都较难找到函数最优值. 根据表 8 中结果显示, MSNSSA 在 6 个基准函数上都求得比其他 5 个算法更优的结果. 验证 MSNSSA 具有较好的有效性和鲁棒性.

表 8 CEC 2014 优化结果对比

Table 8 Comparison of optimization results of CEC 2014

函数	指标	MSNSSA	SSA	BOA	MFO	SCA	PSO
CEC03	平均值	3.48173×10^{4}	7.13409×10^{4}	7.71258×10^{4}	1.05168×10^{5}	5.95030×10^{4}	4.93113×10^{4}
	标准差	3.98187×10^{3}	1.97053×10^{4}	7.95590×10^{3}	4.35140×10^{4}	1.31496×10^{4}	7.30314×10^{3}
CEC05	平均值	5.20018×10^{2}	5.20177×10^{2}	5.21049×10^{2}	5.20275×10^{2}	5.21035×10^{2}	5.20990×10^{2}
	标准差	6.58164×10^{-3}	1.35252×10^{-1}	6.03943×10^{-2}	1.73345×10^{-1}	4.83807×10^{-2}	9.88705×10^{-2}
CEC18	平均值	2.78893×10^{3}	1.21041×10^{4}	4.31418×10^{9}	2.18893×10^{7}	3.16304×10^{8}	5.78922×10^{3}
	标准差	7.12803×10^{2}	9.17792×10^{3}	2.03175×10^{9}	8.30950×10^{7}	1.92087×10^{8}	3.37197×10^{3}
CEC23	平均值	2.50000×10^{3}	2.63108×10^{3}	2.50000×10^{3}	2.67493×10^{3}	2.71333×10^{3}	2.61612×10^{3}
	标准差	0	7.36331×10^{0}	0	4.24126×10^{1}	2.39342×10^{1}	1.24951×10^{0}
CEC24	平均值	2.70000×10^{3}	2.71750×10^{3}	2.70000×10^{3}	2.71882×10^{3}	2.73876×10^{3}	2.72053×10^{3}
	标准差	0	5.83456×10^{0}	0	8.00616×10^{0}	7.90888×10^{0}	6.35266×10^{0}
CEC25	平均值	2.60000×10^{3}	2.64087×10^{3}	2.60000×10^{3}	2.68247×10^{3}	2.61048×10^{3}	2.63564×10^{3}
	标准差	0	7.27821×10^{0}	0	3.48329×10^{1}	1.81045×10^{1}	1.02263×10^{1}

为比较本文的多子群 MSNSSA 算法与其他多子群算法和改进樽海鞘群算法的优劣. 其中, MFOA-SQP (Multiple subgroups fruit fly optimization algorithm based on sequential quadratic programming local search)[18] 是基于局部搜索的多子群果蝇优化算法, 鸡群算法 (Chicken swarm optimization, CSO)[19] 是一种新型的天然多种群优化算法, HCPSO (Improved particle swarm optimization based on hierarchical autonomous learning)[20] 是基于分层自主学习的改进粒子群优化算法, DMS-PSO (Dynamic multi-Swarm PSO)[21] 是动态多子群粒子群优化算法, PSO-SMS (Self-adaptive multi-swarm PSO)[11] 是自适应多子群粒子群优化算法, CASSA (Crazy and adaptive salp swarm algorithm)[22] 是疯狂自适应的樽海鞘群算法, CESSA (Chaotic and elite centroid stretching mechanism salp swarm algorithm)[23] 是混沌精英质心拉伸机制樽海鞘群算法, ICMOABC (Interval cooperative multi-objective artificial bee colony algorithm)[24] 是区间合作多目标人工蜂群算法, HCMOIWO (Hybrid cooperative multi-objective optimization invasive weed optimization)[25] 是混合合作多目标优化入侵杂草优化, MPEA (Multi-population evolutionary algorithm)[26] 多子群进化算法. 各算法在种群数量为 50, 搜索维度为 30 维, 迭代次数为 2 000 的情况下, 将本文改进算法独立运行 50 次后与其他参考文献的几种多子群算法和改进樽海鞘群算法进行对比, 引用文献 [18] 和文献 [20] 的数据见表 9 和表 10.

由表 9 和表 10 可知, 在求解 f_1 和 f_2 函数时,

表 9　与参考文献中算法均值的对比

Table 9　Comparison of the mean with algorithm in references

算法	f_1	f_2	f_3	f_4	f_5	f_6	f_7
MSNSSA	7.35×10^{-36}	9.69×10^{-32}	$\mathbf{1.47\times10^{-21}}$	$\mathbf{3.53\times10^{-6}}$	**0**	**0**	**0**
MFOA-SQP[18]	**0**	5.62×10^{-11}	5.96×10^{-6}	5.71×10^{-3}	2.88×10^{1}	2.53×10^{-4}	**0**
CSO[19]	**0**	1.79×10^{-9}	1.63×10^{-5}	6.15×10^{-4}	1.65×10^{2}	6.06×10^{-3}	**0**
HCPSO[20]	8.71×10^{-28}	3.39×10^{-3}	1.38×10^{-2}	2.57×10^{-4}	3.14×10^{-5}	5.76×10^{-3}	3.68×10^{-10}
DMS-PSO[21]	4.29×10^{-12}	4.54×10^{-6}	2.06×10^{1}	1.07×10^{-2}	2.77×10^{1}	5.68×10^{-2}	7.31×10^{-4}
PSO-SMS[11]	3.55×10^{-20}	9.82×10^{-8}	1.53×10^{-5}	2.09×10^{-2}	2.59×10^{1}	3.54×10^{-4}	7.19×10^{-3}
CASSA[22]	9.35×10^{-147}	$\mathbf{2.84\times10^{-52}}$	8.66×10^{-6}	1.88×10^{-5}	2.77×10^{1}	9.81×10^{-2}	**0**
CESSA[23]	2.50×10^{-23}	4.22×10^{-3}	1.73×10^{-15}	5.90×10^{-5}	2.86×10^{1}	7.51×10^{-2}	**0**
EHO[5]	9.63×10^{-7}	4.71×10^{-4}	6.93×10^{-1}	1.25×10^{-5}	2.85×10^{1}	6.55×10^{0}	6.30×10^{-5}
EWA[6]	7.25×10^{1}	2.15×10^{0}	2.48×10^{-5}	1.10×10^{-1}	5.14×10^{3}	2.74×10^{3}	9.93×10^{-3}
MBO[7]	8.53×10^{-3}	4.17×10^{-4}	2.66×10^{-1}	4.46×10^{-1}	2.05×10^{-7}	1.42×10^{0}	1.52×10^{-2}
MABC[24]	6.02×10^{-4}	6.53×10^{-6}	9.55×10^{0}	1.07×10^{-2}	2.25×10^{-8}	5.98×10^{0}	2.91×10^{-1}
MIWO[25]	3.17×10^{-5}	5.41×10^{-10}	9.34×10^{-13}	8.41×10^{-3}	5.28×10^{-1}	7.68×10^{-2}	4.50×10^{-1}
MPEA[26]	2.70×10^{-11}	1.52×10^{-20}	1.04×10^{-2}	2.35×10^{-1}	6.74×10^{-12}	3.26×10^{-5}	8.74×10^{-3}

算法	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
MSNSSA	$\mathbf{9.98\times10^{-1}}$	$\mathbf{3.08\times10^{-4}}$	**0**	$\mathbf{8.88\times10^{-16}}$	**0**	$\mathbf{1.39\times10^{-34}}$	$\mathbf{3.27\times10^{-30}}$
MFOA-SQP[18]	$\mathbf{9.98\times10^{-1}}$	1.06×10^{-3}	**0**	3.55×10^{-15}	**0**	3.71×10^{-6}	7.53×10^{-10}
CSO[19]	$\mathbf{9.98\times10^{-1}}$	6.03×10^{-4}	1.12×10^{-7}	1.24×10^{-12}	**0**	1.64×10^{-7}	2.24×10^{-1}
HCPSO[20]	$\mathbf{9.98\times10^{-1}}$	1.40×10^{-2}	2.49×10^{-5}	2.26×10^{-4}	8.67×10^{-5}	2.69×10^{-13}	4.18×10^{-3}
DMS-PSO[21]	2.13×10^{0}	5.68×10^{-1}	3.88×10^{1}	1.88×10^{0}	2.24×10^{-2}	2.87×10^{-3}	6.88×10^{-1}
PSO-SMS[11]	$\mathbf{9.98\times10^{-1}}$	2.09×10^{-2}	1.53×10^{1}	2.99×10^{0}	7.23×10^{-2}	1.12×10^{-5}	1.76×10^{-8}
CASSA[22]	$\mathbf{9.98\times10^{-1}}$	4.81×10^{-3}	**0**	$\mathbf{8.88\times10^{-16}}$	**0**	2.33×10^{-20}	1.68×10^{-2}
CESSA[23]	$\mathbf{9.98\times10^{-1}}$	2.59×10^{-3}	1.48×10^{1}	1.06×10^{-2}	2.88×10^{-1}	5.68×10^{-18}	2.62×10^{1}
EHO[5]	1.67×10^{0}	1.27×10^{-1}	1.21×10^{-6}	2.39×10^{-4}	1.89×10^{-6}	1.35×10^{-1}	6.86×10^{1}
EWA[6]	1.50×10^{0}	1.76×10^{-3}	3.10×10^{1}	3.05×10^{0}	1.53×10^{0}	5.77×10^{-1}	1.35×10^{-3}
MBO[7]	$\mathbf{9.98\times10^{-1}}$	1.78×10^{-1}	5.86×10^{-1}	1.13×10^{-1}	8.05×10^{-1}	7.20×10^{-15}	7.12×10^{-1}
MABC[24]	1.41×10^{0}	4.32×10^{-4}	4.15×10^{-2}	2.05×10^{-1}	5.63×10^{-2}	8.05×10^{-9}	1.18×10^{-2}
MIWO[25]	1.89×10^{0}	1.01×10^{-2}	4.62×10^{-1}	1.86×10^{-1}	3.29×10^{-2}	1.98×10^{-1}	1.35×10^{-3}
MPEA[26]	$\mathbf{9.98\times10^{-1}}$	1.46×10^{-3}	6.38×10^{-6}	2.84×10^{-1}	4.38×10^{-7}	2.01×10^{-6}	2.27×10^{-5}

表 10 与参考文献中算法标准差的对比

Table 10 Comparison of the standard deviation with algorithms in reference

算法	f_1	f_2	f_3	f_4	f_5	f_6	f_7
MSNSSA	1.04×10^{-35}	4.52×10^{-32}	$\mathbf{4.75 \times 10^{-21}}$	$\mathbf{1.75 \times 10^{-6}}$	**0**	**0**	**0**
MFOA-SQP[18]	**0**	2.20×10^{-11}	2.03×10^{-6}	4.12×10^{-3}	5.10×10^{-2}	2.17×10^{-4}	**0**
CSO[19]	**0**	1.04×10^{-9}	4.73×10^{-6}	3.12×10^{-2}	7.39×10^{2}	4.75×10^{-3}	**0**
HCPSO[20]	3.55×10^{-28}	2.04×10^{-3}	5.80×10^{-3}	1.88×10^{-5}	1.07×10^{-4}	3.48×10^{-3}	5.97×10^{-10}
DMS-PSO[21]	3.00×10^{-11}	2.23×10^{-5}	7.48×10^{0}	1.03×10^{-3}	2.69×10^{0}	4.87×10^{-2}	3.81×10^{-1}
PSO-SMS[11]	4.61×10^{-20}	1.47×10^{-7}	4.65×10^{-6}	2.50×10^{-3}	2.19×10^{0}	2.21×10^{-4}	1.57×10^{-4}
CASSA[22]	2.32×10^{-147}	$\mathbf{2.27 \times 10^{-50}}$	4.15×10^{-6}	1.21×10^{-5}	1.16×10^{-1}	4.08×10^{-2}	**0**
CESSA[23]	1.84×10^{-23}	1.51×10^{-2}	1.25×10^{-13}	5.08×10^{-5}	4.89×10^{-2}	3.11×10^{-2}	**0**
EHO[5]	1.26×10^{-7}	8.23×10^{-4}	8.44×10^{-1}	1.26×10^{-5}	1.83×10^{-2}	7.56×10^{0}	6.21×10^{-5}
EWA[6]	7.43×10^{1}	1.54×10^{0}	7.37×10^{-6}	8.73×10^{-2}	8.93×10^{3}	2.58×10^{-3}	1.93×10^{-4}
MBO[7]	1.28×10^{-4}	1.83×10^{-4}	3.00×10^{0}	3.89×10^{-1}	3.54×10^{-7}	3.65×10^{-1}	1.10×10^{-2}
MABC[24]	7.23×10^{-3}	3.63×10^{-3}	1.18×10^{0}	1.77×10^{-1}	3.87×10^{-7}	1.21×10^{-1}	1.55×10^{-1}
MIWO[25]	4.32×10^{-6}	1.28×10^{-5}	2.21×10^{-12}	2.63×10^{-3}	7.54×10^{-1}	1.82×10^{-2}	2.09×10^{-2}
MPEA[26]	5.74×10^{-10}	4.13×10^{-18}	1.80×10^{-1}	5.22×10^{-2}	3.19×10^{-10}	2.63^{-6}	3.56×10^{-1}

算法	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
MSNSSA	$\mathbf{2.95 \times 10^{-23}}$	$\mathbf{3.56 \times 10^{-8}}$	**0**	**0**	**0**	$\mathbf{3.56 \times 10^{-34}}$	$\mathbf{7.31 \times 10^{-30}}$
MFOA-SQP[18]	1.13×10^{-1}	4.47×10^{-4}	**0**	1.32×10^{-12}	**0**	1.76×10^{-6}	5.56×10^{-10}
CSO[19]	8.01×10^{0}	9.92×10^{-4}	3.16×10^{-5}	1.01×10^{-11}	**0**	4.74×10^{-7}	1.17×10^{-1}
HCPSO[20]	2.96×10^{0}	6.22×10^{-2}	1.05×10^{-5}	2.52×10^{-4}	2.79×10^{-6}	5.95×10^{-11}	1.84×10^{-5}
DMS-PSO[21]	5.94×10^{-1}	8.04×10^{-1}	2.80×10^{0}	2.46×10^{-1}	1.77×10^{-2}	7.54×10^{-1}	5.02×10^{-1}
PSO-SMS[11]	2.77×10^{-1}	5.76×10^{-3}	1.29×10^{0}	3.87×10^{-1}	6.36×10^{-2}	3.35×10^{-6}	9.20×10^{-9}
CASSA[22]	2.82×10^{-1}	2.56×10^{-5}	**0**	9.86×10^{-32}	**0**	8.12×10^{-18}	1.35×10^{-2}
CESSA[23]	9.19×10^{1}	8.90×10^{-1}	2.15×10^{1}	5.31×10^{-2}	3.41×10^{-1}	1.08×10^{-18}	6.14×10^{0}
EHO[5]	8.37×10^{-1}	1.62×10^{0}	2.30×10^{-7}	1.37×10^{-5}	3.17×10^{-7}	2.20×10^{0}	3.37×10^{1}
EWA[6]	2.28×10^{-1}	4.22×10^{-2}	1.87×10^{1}	1.24×10^{0}	4.99×10^{-1}	3.22×10^{-1}	2.04×10^{-3}
MBO[7]	3.95×10^{0}	4.65×10^{0}	4.17×10^{-1}	7.19×10^{0}	8.32×10^{-1}	1.27×10^{-12}	2.27×10^{-1}
MABC[24]	6.58×10^{0}	8.28×10^{-1}	2.41×10^{-1}	1.65×10^{-1}	5.66×10^{-1}	6.77×10^{-1}	1.67×10^{-1}
MIWO[25]	1.92×10^{1}	2.07×10^{-2}	1.30×10^{-1}	2.68×10^{0}	5.31×10^{-1}	2.30×10^{-1}	1.42×10^{-2}
MPEA[26]	5.37×10^{-1}	7.43×10^{-3}	5.35×10^{-6}	3.91×10^{-1}	7.48×10^{-3}	5.94×10^{-8}	6.36×10^{-4}

MSNSSA 未达到最优, 而求解其余几个函数在平均值和标准差较其余几种算法都能达到最优. 因此, 更进一步说明本文提出的 MSNSSA 算法比其他多子群算法具有更大优越性.

综上可知, 多子群的共生非均匀高斯变异樽海鞘群算法对于本文多种基准函数都有很好的寻优结果. 特别是对于高维、多峰的函数, 具有较好的稳定性和寻优能力.

4 结束语

本文在标准樽海鞘群算法的基础上, 分析领导者位置更新公式中的幂系数以平衡探索能力和开发能力, 共生策略增强局部探索能力, 非均匀高斯变异增加种群多样性, 提出一种改进的多子群的共生非均匀高斯变异樽海鞘群算法. 并将樽海鞘群算法应用于经典和 CEC 2014 基准函数的寻优问题中. 不仅使用最优值、标准差等指标对算法进行检验, 本文还提出使用 Wilcoxon 秩和检验对算法显著性水平进行验证. 研究表明: 多子群的共生非均匀高斯变异樽海鞘群算法可以获得更好的全局搜索和局部搜索能力, 且收敛到质量更好的最优解, 算法的有效性和鲁棒性也得到验证. 在后续的研究中, 考虑将改进的樽海鞘群算法应用到工程实践问题中, 以进一步验证算法的性能.

References

1 Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the International Conference on Neural Networks. Perth, Australia: IEEE, 1995. 1942−1948

2 Mirjalili S. SCA: A sine cosine algorithm for solving optimization problems. *Knowledge Based Systems*, 2016, **96**(96): 120−133

3 Arora S, Singh S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Computing*, 2019, **23**(3): 715−734

4 Mirjalili S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, 2015, **89**(11): 228−249

5 Wang G G, D S, Cui Z H. Monarch butterfly optimization. *Neural Computing and Applications*, 2019, **31**(7): 1−20

6 Wang G G, Suash D, Santos C L D. Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global op-

timization problems. *International Journal of Bio-Inspired Computation*, 2018, **12**(1): 1−22

7 Wang G G, Deb S, Coelho L D S. Elephant herding optimization. In: Proceedings of the International Symposium on Computational and Business Intelligence. Bali, Indonesia: IEEE, 2015. 1−5

8 Mirjalili S, Gandomi A H, Mirjalili S Z, Saremi S, Faris H, Mirjalili S M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, 2017, **114**(6): 163−191

9 Hegazy A E, Makhlouf M A, Eltawel G S. Improved salp swarm algorithm for feature selection. *Journal of King Saud University–Computer and Information Sciences*, 2018, **6**(3): 1−10

10 Sayed G I, Khoriba G, Haggag M H. A novel chaotic salp swarm algorithm for global optimization and feature selection. *Applied Intelligence*, 2018, **48**(3): 1−20

11 Zeng Hui, Wang Qian, Xia Xue-Wen, Fang Xia. Particle swarm optimization algorithm based on self-adaptive multi-swarm. *Computer Engineering and Applications*, 2018, **54**(10): 59−65
(曾辉, 王倩, 夏学文, 方霞. 基于自适应多种群的粒子群优化算法. 计算机工程与应用, 2018, **54**(10): 59−65)

12 Xin Zi-Yun, Zhang Da-Min, Chen Zhong-Yun, Zhang Hui-Juan, Yan Wei. Shared crow algorithm using multi-segment perturbation. *Computer Engineering and Applications*, 2020, **56**(2): 55−61
(辛梓芸, 张达敏, 陈忠云, 张绘娟, 闫威. 多段扰动的共享型乌鸦算法. 计算机工程与应用, 2020, **56**(2): 55−61)

13 Zhao X C, Gao X S, Hu Z C. Evolutionary programming based on non-uniform mutation. *Applied Mathematics and Computation*, 2007, **192**(1): 1−11

14 He X S, Ding W J, Yang X S. Bat algorithm based on simulated annealing and Gaussian perturbations. *Neural Computing and Applications*, 2014, **25**(2): 459−468

15 Mirjalili S, Mirjalili S M, Yang X S. Binary bat algorithm. *Neural Computing and Applications*, 2014, **25**(3): 663−681

16 Derrac J, Garcia S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm & Evolutionary Computation*, 2011, **1**(1): 3−18

17 Emad N. A modified flower pollination algorithm for global optimization. *Expert Systems with Applications*, 2016, **57**(9): 192−203

18 Wang Ying-Bo, Wang Yi-Xing. A multiple subgroups fruit fly optimization algorithm based on sequential quadratic programming local search. *Computer Engineering & Science*, 2018, **40**(5): 906−915
(王英博, 王艺星. 基于SQP局部搜索的多子群果蝇优化算法. 计算机工程与科学, 2018, **40**(5): 906−915)

19 Meng X B, Liu Y, Gao X Z, Zhang H Z. A new bio-inspired algorithm: Chicken swarm optimization. In: Proceedings of the International Conference in Swarm Intelligence. Hefei, China: Springer, 2014. 86−94

20 Yuan Xiao-Ping, Jiang Shuo. Improved particle swarm optimization algorithm based on hierarchical autonomous learning. *Journal of Computer Applications*, 2019, **39**(1): 148−153
(袁小平, 蒋硕. 基于分层自主学习的改进粒子群优化算法. 计算机应用, 2019, **39**(1): 148−153)

21 Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer. In: Proceedings of the Swarm Intelligence Symposium. Pasadena, USA: IEEE, 2005. 124−129

22 Zhang Da-Min, Chen Zhong-Yun, Xin Zi-Yun, Zhang Hui-Juan, Yan Wei. Salp swarm algorithm based on craziness and adaptive. *Control and Decision*, 2020, **35**(9): 2112−2120
(张达敏, 陈忠云, 辛梓芸, 张绘娟, 闫威. 基于疯狂自适应的樽海鞘群算法. 控制与决策, 2020, **35**(9): 2112−2120)

23 Chen Zhong-Yun, Zhang Da-Min, Xin Zi-Yun, Zhang Hui-Juan, Yan Wei. Salp swarm algorithm using chaotic and elite centroid-stretching mechanism. *Computer Engineering and Applications*, 2020, **56**(10): 44−50
(陈忠云, 张达敏, 辛梓芸, 张绘娟, 闫威. 混沌精英质心拉伸机制的樽海鞘群算法. 计算机工程与应用, 2020, **56**(10): 44−50)

24 Zhang L M, Wang S S, Zhang K, Zhang X Q, Sun Z X, Zhang H, et al. Cooperative artificial bee colony algorithm with multiple populations for interval multiobjective optimization problems. *IEEE Transactions onFuzzy Systems*, 2019, **27**(5): 1052−1065

25 Naidu Y R, Ojha A K. Solving multiobjective optimization problems using hybrid cooperative invasive weed optimization with multiple populations. *Systems man and Cybernetics*, 2018, **48**(6): 821−832

26 Liu H T, Du W, Guo Z X. A multi-population evolutionary algorithm with single-objective guide for many-objective. *Optimization. Information Sciences*, 2019, **503**(9): 39−60

陈忠云 贵州大学大数据与信息工程学院硕士研究生. 主要研究方向为智能优化算法和认知无线网络.
E-mail: chenzhongyun315@hotmail.com
(**CHEN Zhong-Yun** Master student at the School of Big Date and Information Engineering, Guizhou University. His research interest covers intelligent optimization algorithm and cognitive wireless network.)

张达敏 贵州大学大数据与信息工程学院教授. 主要研究方向为智能优化算法和认知无线网络. 本文通信作者.
E-mail: dmzhang@gzu.edu.cn
(**ZHANG Da-Min** Professor at the School of Big Date and Information Engineering, Guizhou University. His research interest covers intelligent optimization algorithm and cognitive wireless network. Corresponding author of this paper.)

辛梓芸 贵州大学大数据与信息工程学院硕士研究生. 主要研究方向为智能优化算法和认知无线网络.
E-mail: muz_e@sina.cn
(**XIN Zi-Yun** Master student at the School of Big Date and Information Engineering, Guizhou University. Her research interest covers intelligent optimization algorithm and cognitive wireless networkreless network.)