Journal of System Simulation

Volume 33 | Issue 11 Article 7

11-17-2021

A Natural Computing Method Based on Spatial Division Search Strategy

Xiaoqing Sun
College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China;

Cheng Hao
College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China;

Luyao Zhang
College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China;

Weidong Ji
College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China;

See next page for additional authors

A Natural Computing Method Based on Spatial Division Search Strategy

Abstract

Abstract: A natural computing method based on spatial division search strategy is proposed. *The strategy can map the high-dimensional space to the three-dimensional Cartesian coordinate system by grouping the dimensional space into a group of three dimensions. The individual after spatial segmentation is numbered into subindividual, to increases the particle number while reducing the dimension, thus the individual is distributed over wider search space to effectively increases the diversity of the population. The algorithm iterates to a certain extent and can synthesize the individual into the original individual through the numbered index. By calculating the fitness value, some poor individuals can be deleted to balance the time efficiency and speed up the running time*. At the end of the iteration, the global optimal position of individual in the group can be found through the numbered index to synthesize the fitness value of the optimal individual output, which makes the algorithm have a better ability to search for the optimization. *The convergence of the strategy is analyzed by Markov chain*. The spatial division search strategy is applied to PSO, CA and DE, and its performance is verified in the standard test functions. Experimental results show that the proposed strategy can improve the convergence speed and optimization ability obviously.

Keywords

space division, numbered particle, natural calculation, diversity preservation

Authors

Xiaoqing Sun, Cheng Hao, Luyao Zhang, Weidong Ji, and Wang Xu

Recommended Citation

Sun Xiaoqing, Cheng Hao, Zhang Luyao, Ji Weidong, Wang Xu. A Natural Computing Method Based on Spatial Division Search Strategy[J]. Journal of System Simulation, 2021, 33(11): 2589-2605.

第 33 卷第 11 期 系统仿真学报© Vol. 33 No. 11
2021 年 11 月 Journal of System Simulation Nov. 2021

一种基于空间分割搜索策略的自然计算方法

孙小晴，程昊，张潞瑶，季伟东*，王旭
(哈尔滨师范大学 计算机科学与信息工程学院， 黑龙江 哈尔滨 150025)

摘要：在传统自然计算方法的基础上提出了基于空间分割搜索策略的自然计算方法，该策略利用对维数空间进行3维为一组的分组方式，可以使得高维空间映射到直观的三维空间直角坐标系中，同时对空间分割后的个体进行编号形成分个体，在减少维数的基础上间接增加了粒子数的规模，使个体分布于更广阔的搜索空间，有效增加了种群多样性。算法迭代到一定程度，可通过编号索引将分个体合成原个体，通过适应度值的计算，删除部分较差个体，平衡时间效能，加快运行时间。迭代最后可通过编号索引寻找组别中分个体的全局最优位置，合成最优个体输出适应度值，使得算法有更好的寻优能力。利用马尔可夫链对该策略进行收敛性分析。将空间分割搜索策略应用于粒子群算法、布谷鸟算法和差分进化算法中，并在标准测试函数中验证其性能。实验表明：该策略在收敛速度和寻优能力上均有明显的提升。

关键词：空间分割；编号分粒子；自然计算；多样性保持
中图分类号：TP301 文献标志码：A 文章编号：1004-731X (2021) 11-2589-17
DOI: 10.16182/j.issn1004731x.joss.21-FZ0696

A Natural Computing Method Based on Spatial Division Search Strategy

Sun Xiaoqing, *Cheng Hao*, *Zhang Luyao*, *Ji Weidong**, *Wang Xu*
(College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China)

Abstract: A natural computing method based on spatial division search strategy is proposed. *The strategy can map the high-dimensional space to the three-dimensional Cartesian coordinate system by grouping the dimensional space into a group of three dimensions. The individual after spatial segmentation is numbered into subindividual, to increases the particle number while reducing the dimension, thus the individual is distributed over wider search space to effectively increases the diversity of the population. The algorithm iterates to a certain extent and can synthesize the individual into the original individual through the numbered index. By calculating the fitness value, some poor individuals can be deleted to balance the time efficiency and speed up the running time*. At the end of the iteration, the global optimal position of individual in the group can be found through the numbered index to synthesize the fitness value of the optimal individual output, which makes the algorithm have a better ability to search for the optimization. *The convergence of the strategy is analyzed by Markov chain*. The spatial division search strategy is applied to PSO, CA and DE, and its performance is verified in the standard test functions. Experimental results show that the proposed strategy can improve the convergence speed and optimization ability obviously.

Keywords: space division; numbered particle; natural calculation; diversity preservation

收稿日期：2021-05-18 修回日期：2021-08-09
基金项目：国家自然科学基金(31971015)；黑龙江省自然科学基金(LH2021F037)；哈尔滨师范大学硕士研究生学术创新基金(HSDSSCX2019-08)
第一作者：孙小晴(1994-)，女，硕士生，研究方向为群体智能。E-mail：sunxiaoqing2649@163.com
通讯作者：季伟东(1978-)，男，博士，教授，研究方向为大数据、群体智能。E-mail：kingjwd@126.com

引言

自然计算通常是指受自然现象启发而发展起来的智能算法，具有模仿自然界生物进化的特点，是一类具有自适应、自组织、自学习能力的模型与算法，能够解决传统计算方法难于解决的各种复杂问题[1]。如人工蜂群算法模拟蜜蜂繁殖、采蜜的机制，根据不同分工实现蜂群信息的共享和交流，从而寻找最优解[2]；蚁群算法通过个体间的信息传递，利用正反馈和分布式协作寻找最优路径，解决TSP(Traveling Salesman Problem)难题[3]；根据自然界狮子寻找生存资源的群体行为，提出的群搜索优化算法[4]等。同时，自然计算[5]在智能交通、智能控制、新能源与新材料、人工神经网络、网络安全与军事领域均得到了广泛应用[6]。虽然自然计算方法模拟的自然现象不尽相同，但是它们都是基于种群的优化方法[7]。所以在基于种群的自然计算方法中对种群个体规模和种群维数空间的分析一直是自然计算方面的重要问题。种群规模增大，可以有效扩充个体搜索的空间范围，提高寻找最优解的能力，增强种群多样性，但是会导致收敛速度变慢，运行时间过长；反之，当种群规模过小，尤其对于多峰函数，极易使个体陷入局部最优，降低了算法的性能。而对于种群的维数空间，高维数空间使得算法复杂多样，运行时间较长。所以，如何选择一种有效的策略来平衡算法种群个体规模和种群维数空间，成为当前自然计算方法中有待解决的关键问题。

针对该问题，已经出现了很多方面的改进，如对种群规模进行控制以及对个体间的拓扑结构、空间维数进行改进等。

关于改进个体间的种群规模，王蓉芳等[7]利用最优个体的更新与否对种群状态进行划分，然后对处于收敛的种群删除冗余个体，加快寻优速度；陷入局部最优则增加新个体，增加种群多样性。Sun等[8]提出了双种群的粒子群算法，分为主群和从群，主群和从群的工作有明显划分，在种群多样性和收敛速度上得到了平衡。Xu 等[9]同样利用双种群的思想，把种群分为两个子种群，一个负责局部搜索，一个负责全局搜索，提高了收敛精度和收敛速度。Wei 等[10]提出多种群自适应策略粒子群算法(Multiple Adaptive Particle Swarm Optimization，MAPSO)，将种群划分为多个种群且在进化过程中进行重新组合，同一种群的个体可以执行不同的搜索行为，同一粒子也可以执行不同的搜索行为。针对求解大规模问题，在动态多种群粒子群优化算法(Dynamic Multi-Particle Swarm Optimizer，DMS-PSO)[11]的基础上，使用了一种有效的分组策略，同时实现维数和种群的双分组的协同进化动态粒子群优化算法(Dynamic Multi-Swarm-Cooperative Coevolution，DMS-CC)，维数分组能有效解决决策变量多且互相关联的问题，种群分组则解决多模态和陷入局部最优的问题[12]。针对种群多样性和收敛性之间的矛盾，提出了具备反向学习和局部学习能力相结合的粒子群优化算法，利用较差种群位置信息引导部分种群进行反向学习，保证算法的全局勘探能力[13]。

在拓扑结构方面，邓先礼等通过融合两种常用的邻居拓扑结构，赋予个体更多的信息来源，同时，利用子种群的概念，引入加速因子组合，实现子种群间的协作与计算资源的合理分配[14]。通过环型拓扑结构和利用维度排序并比较适应度值，进而生成局部最优拓扑，拓扑结构的改变，可以有效利用种群中的信息[15]。同时，为了更好的平衡个体的搜索性能，在拓扑结构的基础上提出了一种基于高斯函数实现非线性递减惯性权重的算法[16]。喻飞等[17]提出了 LensPSO 算法，利用透镜成像原理对反向学习进行扩展，变异和反向学习增强后期开发能力。

对于空间个体维数的改进方面，为避免维数间的相互干扰，导致某些维度虽然变得更优，但却被维数较差的覆盖，运行效率低下，提出了一种最优粒子逐维变异的粒子群优化算法[18]。唐祎玲等[19]将最优个体在空间中的信息以二维数组的方式作为划分依据，通过速度分解的方式寻找最优粒子，

DOI: 10.16182/j.issn1004731x.joss.21-FZ0696

使群体向全局最优个体靠近，提高了算法的收敛速度和求解精度。基于维数空间的大规模求解问题，提出了社会学习粒子群优化算法(Social Learning Particle Swarm Optimization，SLPSO)[20]和竞争学习算法(Competitive Swarm Optimizer，CSO)[21]，社会学习粒子群优化方法采用了与维数相关的参数控制方法，减轻了参数设置的负担问题；竞争学习算法采用一种竞争机制，能够有效解决高维空间的问题。Zhou 等[22]对 CSO 算法进行了改进，提出了对立竞争学习的粒子群优化算法(Opposition-Based Learning Competitive Particle Swarm Optimizer，OBL-CPSO)，通过 3 个个体竞争，并引入反向学习机制，改变个体的学习模式，可有效解决高维问题。

这些算法虽然在一定程度上平衡了算法的有效性，但是针对算法的问题，实现过程依赖于具体步骤，适用性较弱。为了解决以上问题，受种群分组的启发，提出了基于空间分割搜索(Space Division Search，SDS)的自然计算方法，对维数空间进行 3 维分组映射至三维空间中，同时对空间分割后的个体进行编号形成子个体。子个体在父个体的基础上大大减少了维数，也减少了个体的计算时间，加快了子个体的寻优速度。同时间接的增加了种群规模，有效增加了种群的多样性，让子个体按编号分布于三维空间中，增加了寻找到全局最优解的概率。该方法同时实现了种群分组和普适性，适用于任何自然计算方法。将该策略应用于自然计算领域的粒子群算法和遗传算法中，并与目前主流算法进行对比，验证空间分割搜索策略的有效性和鲁棒性。

1 空间分割搜索算法在自然计算方法中的应用

1.1 空间分割搜索策略

在传统的自然计算方法中，为更好地搜寻算法的最优值，在种群规模和维数方面做了诸多改进。如算法迭代过程中随机增加种群规模[23]、利用 Logistic 模型自适应控制种群规模等来增加种群的有效性和多样性[7]。逐维学习[24]、逐维变异[18]，以及将维数随机进行重组[12]等算法在维数方面的改进，也有效地增强了算法的性能。但是对于维数这个概念来说是比较抽象的，难以想象种群在多维数中的构造方式，使算法变得更加复杂。所以，本文提出的空间分割搜索策略主要解决 3 个问题：

(1) 如何解决多维数空间的抽象复杂问题；

(2) 如何增加种群规模，增强种群多样性；

(3) 如何解决算法迭代过程中的时间性能问题。

基于空间分割搜索策略的基本思想：初始化种群规模 N 和维数 D，随机生成粒子的速度和位置，生成一个 $N\times D$ 的矩阵。对空间维数 D 进行分割，以 3 维为一组，即 $P=D/3$，这样，就把复杂的高维空间转换到所熟知的三维空间(x,y,z 轴的空间坐标系)中，其中，P 为分割的组别数，$P=(1,2,\cdots,R)$，若分割的组别数 $P=D/3$ 无法整除，则 $P=[D/3]+1$，使得抽象的高维空间函数转换为可观测问题。

维数空间进行分割后，由原来的一个 $1\times D$ 维粒子会分出多个分粒子，为了让分粒子存在一一对应的关系，对空间分割后的粒子进行编号，则 $N\times D$ 维矩阵表示为 $\boldsymbol{A}_{QR}$ 的形式，其中 Q 为第 i 个粒子，i 的取值为 $(1,2,\cdots,N)$，R 为第 j 个空间维数，j 的取值为 $(1,2,\cdots,P)$，进行空间分割后，相当于将原有的种群规模数 N 分割成 $N\times P$ 个分粒子，将分粒子初始化的位置放置于三维空间直角坐标系中，如图 1 所示。图 1 只是简单列出几个分粒子的示意图(A_{ij} 表示第 i 个粒子第 j 组的三维空间)。该方法在减少维数的基础上间接增加了种群的规模，兼顾了种群的多样性。

对分粒子分别进行编号后，放置于三维空间中迭代寻优，迭代结束后，提取对应的编号分粒子中适应度值最小时所对应的位置取值，合成 g_{best}，g_{best} 为合成的一个 D 维全局最优粒子。这种对分粒子的合成方式，虽然很大程度结合了各个空间的最优解，但是种群规模的增加，会导致时间性能变差。所以，选择一种对算法精度影响不大的方式用

于平衡时间效能，加快运行时间是一种可行的方法。即在空间分割搜索策略中，对分粒子编号后，迭代一定次数 K，利用编号一一对应的关系，进行粒子的整合，合成原来的 $N\times D$ 维矩阵，计算其适应度值，根据适应度值进行排序，删除部分较差种群的个体。然后根据空间分割策略继续打散粒子，形成分粒子，迭代 K 次，继续整合粒子，根据适应度值，再删除部分较差种群的个体，然后继续把空间维数分组打散粒子，直至迭代结束，提取对应的编号分粒子中适应度值最小时所对应的位置取值，合成 D 维的全局最优粒子，输出适应度值。

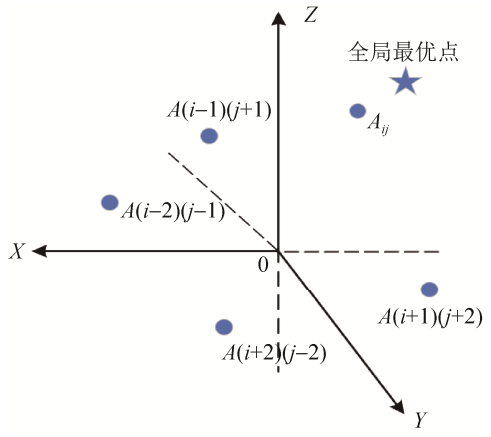


图 1 三维空间示意图

Fig. 1 Three dimensional space schematic diagram

1.2 空间分割搜索的算法流程

算法流程：

step 1：初始化随机生成 N 个 D 维的粒子形成初始种群 pop；

step 2：将空间维数进行分割，每三维为一组，计算 $P=D/3$ 的取值；

step 3：对粒子数进行编号，表示为 $\boldsymbol{A}_{QR}$，将原有的粒子数分割成 $(N-nS)\cdot P$ 个分粒子，其中 S 表示后期删除较差的粒子个数，n 的取值为(0，1，…，n)，Q 的取值表示 i 个粒子，R 的取值表示第 j 个空间分割的组别数；

step 4：将新生成的 $(N-nS)\cdot P$ 个三维的粒子放置于三维空间直角坐标系中，形成新的种群；

step 5：计算新的种群 pop_{new} 中每个分粒子的适应度值 $fitness$，确定个体最优值 p_{best} 和全局最优值 g_{best}；

step 6：根据标准粒子群公式更新种群 pop_{new} 中每个粒子的速度和位置，计算新的种群 pop_{new} 中每个分粒子的适应度值 $fitness$，并更新 p_{best} 和 g_{best}；

step 7：判断当前的迭代次数是否为 K 的整数 n 倍，如果是则转 step 9，否则，则返回 step 5；

step 8：判断当前迭代次数是否达到结束条件，如果是则转 step 11，否则转 step 5；

step 9：根据编号索引 $A_{11},A_{12},\cdots,A_{1P}$；$A_{21},A_{22},\cdots,A_{2P}$；…；$A_{(N-nS)1}$，$A_{(N-nS)2}$，…，$A_{(N-nS)P}$ 合成 $(N-nS)\times D$ 维新种群 pop'_{new}，计算适应度值 $fitness$，并根据 $fitness$ 从小到大排序，删除 S 个较差的种群个体；

step 10：根据空间分割搜索策略，重复 step 4；

step 11：分别提取编号分粒子 $A_{11},A_{21},\cdots,A_{(N\text{-}nS)1}$；$A_{12},A_{22},\cdots,A_{(N\text{-}nS)2}$；…，$A_{1P},A_{2P},\cdots,A_{(N\text{-}nS)P}$ 中的适应度值最小时所对应的位置取值，合成 g_{best}，$g_{\text{best}}=(g_{\text{best}_1},g_{\text{best}_2},\cdots,g_{\text{best}_P})$，$g_{\text{best}}$ 为合成的一个 D 维全局最优粒子；

step 12：计算 $fitness(g_{\text{best}})$ 的适应度值，输出最终结果。

空间分割搜索策略示意图如图 2 所示。

2 SDS 算法收敛性及时间复杂度分析

2.1 SDS 算法收敛性分析

定义 1(最小优化问题)令 $\min\{f(x)\mid\forall x\in S,\ 0<f(x)<+\infty\}$ 为最小优化问题。在可行域 S 内的所有元素对应的适应度值集合中的最小元素即为最小优化问题的解。

定义 2(个体状态和个体状态空间)个体状态由个体在 d 维空间下的运动过程中的位置构成，记为 x_d，其中 $x_d\in S_d$，S_d 表示可行解空间，d 表示可行解空间维度。每个个体在迭代过程中的所有可能维度状态组成个体的维度状态空间，记为

$$X_d=\{x_d\mid x_d\in S_d,d=1,2,\cdots,D\}。\tag{1}$$

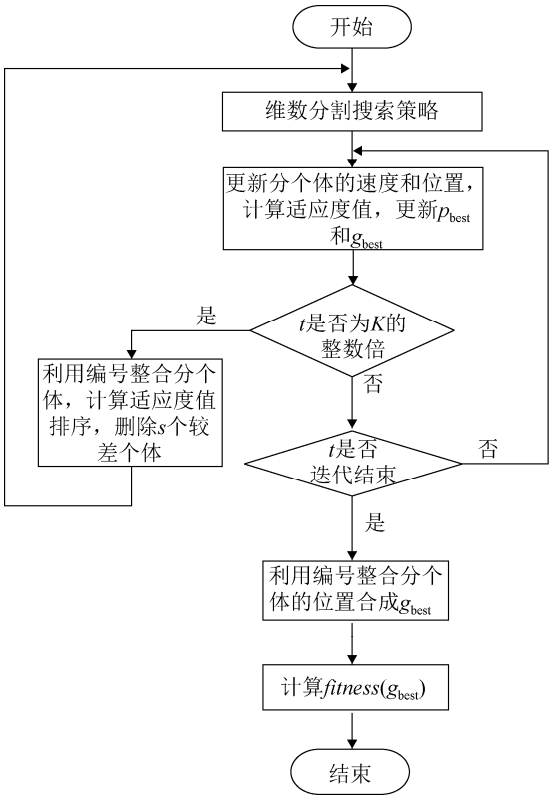


图 2 空间分割搜索策略示意图

Fig. 2 Spatial segmentation search strategy schematic diagram

定义 3(种群状态和种群状态空间)在一次迭代中，所有个体在 d 维空间下的状态构成本次迭代在 d 维空间下的种群状态，记为 $\varphi=(x_{1,d},x_{2,d},\cdots,x_{i,d})$, $i=1,2,\cdots,N,d=1,2,\cdots,D$ 其中 $x_{i,d}$ 为本次迭代中第 i 个体在第 d 维的状态。种群的所有维度状态组成该种群的维度状态空间[25]，记为

$$\phi=\{(x_{1,d},x_{2,d},\cdots,x_{i,d})\mid x_{i,d}\in X_d,i=1,2,\cdots,N,d=1,2,\cdots,D\} \quad (2)$$

定义 4(渐进收敛算法的收敛准则)对于一般的优化问题 $<Y,f>$，采用随机优化算法 Z，第 t 次迭代的结果为 x_t，下一次迭代结果则为 $x_{t+1}=Z(x_t,\zeta)$，其中，Y 为问题的可行解空间，f 为该算法的适应度函数，ζ 为该算法曾经搜索过的解[26]，在 Lebesgue 测度空间中，搜索的下界被定义为：

$$R_{\xi,M}\begin{cases}\{x\in Y\mid f(x)<\sigma+\xi\}, & \sigma\text{有限}\\ \{x\in Y\mid f(x)<-c\}, & \sigma=-\infty\end{cases} \quad (3)$$

其中 $\xi>0,c$ 为充分大的正数，如果算法 Z 在迭代中寻找到了 $R_{\xi,M}$ 中的一个点，即认为算法 Z 找到了可以接受的全局最优点或近似全局最优点[27]。

条件 1 $f(Z(x,\zeta))\leqslant f(x)$，且若 $\zeta\in Y$, $f(Z(x,\zeta))\leqslant f(\zeta)$。

条件 2 对 $\forall A\in Y,\text{s.t.}\,v(A)>0$，有：$\prod_{t=0}^{\infty}(1-v_t(A))=0$，其中 $v_t(A)$ 为算法 Z 在第 t 次迭代搜索到的解在空间 A 上的概率测度。

定理 1 SDS 算法在使种群放置三维空间后，新生成的种群依然遵循收敛准则。

证明 降维前种群矩阵为 $\boldsymbol{m}_D=\{x_{1,D},x_{2,D},\cdots,x_{N,D}\}$，记该种群中的最佳个体为 $x^t{}_{g_{\text{best}},D}$。一个算法收敛的充要条件是符合条件 1 与条件 2，任意渐进收敛算法一定符合上述 2 个条件。

根据条件 1，最佳个体适应度值序列 $(x^1_{g_{\text{best}},D},\ x^2_{g_{\text{best}},D},\cdots,\ x^t_{g_{\text{best}},D})$ 为单调非递增序列，所以对 $\forall i>j$ 有 $f(x^i_{g_{\text{best}},D})\leqslant f(x^j_{g_{\text{best}},D})$ 成立。

根据条件 2，有 $\lim_{t\to\infty}x^t_{g_{\text{best}},D}=X_{g_{\text{best}},D}$ 成立，其中 $X_{g_{\text{best}},D}$ 为该问题的最优解。

通过 SDS 算法将种群矩阵由 $\boldsymbol{m}_D=\{x_{1,D},x_{2,D},\cdots,x_{N,D}\}$ 降维至 $m_3=\{x_{1,3},x_{2,3},\cdots,x_{N\cdot P,3}\}$ 后，会产生新的最优个体 $x^t{}_{g_{\text{best}},3}$，映射至 3 维空间后的种群依然遵循渐进收敛算法进行收敛操作，那么对于降维后的种群矩阵 $\boldsymbol{m}_3=\{x_{1,3},x_{2,3},\cdots,x_{N\cdot P,3}\}$，有 $\forall i>j\ f(x^i_{g_{\text{best}},3})\leqslant f(x^j_{g_{\text{best}},3})$，即降维后的种群符合条件 1。生成的最佳个体适应度序列 $(x^1_{g_{\text{best}},3},\ x^2_{g_{\text{best}},3},\cdots,\ x^t_{g_{\text{best}},3})$ 为单调非递增序列，随着迭代次数的增加，$f(x^t_{g_{\text{best}},3})$ 逐渐靠近最优值，所以有 $\lim_{t\to\infty}P(x^t_{g_{\text{best}},3}=X_{g_{\text{best}},3})=1$，即降维后的种群符合条件 2，所以降维后的种群在迭代过程中依然收敛。证毕。

定理 2 任意渐进收敛算法在经过 SDS 算法降维处理后，以概率为 1 收敛。

证明 种群矩阵由 D 维降低至三维后，会生成 k 个子种群，这 k 个子种群在同一解空间中迭代寻

优。根据定理 1，种群在降维后得到的新种群渐进收敛。

因为降维前种群是由这 k 个三维子种群按维度组合而成，所以根据子种群按维度拼接得到的原始种群的 g_{best} 为：

$$X_{g_{\text{best},D}} = X_{g_{\text{best},1,3}} + X_{g_{\text{best},2,3}} + \cdots + X_{g_{\text{best},k,3}} \tag{4}$$

式中：‘+’表示连接。

那么根据降维后的新种群的收敛概率得到降维前的原始种群的收敛概率有：

$$\prod_{k=1}^{K} \lim_{t\to\infty} P(m_{k,3}^{t} \in M,\ x_{g_{\text{best},k,3}}^{t} = X_{g_{\text{best},k,3}}) = 1^{K} = 1 \tag{5}$$

其中 $X_{g_{\text{best},k,3}}$ 为降维后第 k 个子种群最佳个体。根据式(5)，SDS 算法以概率为 1 收敛，定理 2 得证。

2.2 时间复杂度分析

SDS 每一次所需时间为 $T_{\text{EO}} + T_{\text{SDS}}$，其中，$T_{\text{EO}}$ 是当前种群中所有个体进行一步操作的时间，T_{SDS} 是执行 SDS 的时间。由于 T_{EO} 的计算必须依据不同的算法进行不同的具体操作，所以本文只讨论 SDS 算法的时间复杂度。

从图 2 可以看出，每代执行 SDS 共有 2 种情况，执行了删除个体操作的时间 T_{DEC} 和执行了将分散的个体合成为新个体的合成操作的时间 T_{TOG}；只执行了合成时间的 T_{TOG}。根据文中对这 2 个步骤的描述，通过符号 O 运算规则，推导出各自所需时间分别为 $T_{\text{DEC}} = O\left(\frac{T}{300} \times sizepop \times \frac{D}{3} \times 0.7\right)$ 和 $T_{\text{TOG}} = O\left(sizepop \times D/3 + D/3\right)$，最后可得：

$$T_{\text{SDS}} = T_{\text{C}} + \max(T_{\text{DEC}}, T_{\text{TOG}}) \tag{6}$$

式中：T_{C} 为算法中的标准触发时间，$T_{\text{C}} = O(1)$，将 T_{C}，T_{DEC}，T_{TOG} 带入式(6)，得 $T_{\text{SDS}} = O(1) + O(T \times sizepop \times D)$，化简，得到空间分割搜索算法策略的时间复杂度最差为 $O(T \times sizepop \times D)$。

3 实验结果与分析

实验的仿真平台选用 Windows10，Matlab2018a。本实验将空间 SDS 这一策略应用于自然计算领域中的布谷鸟算法(Cuckoo Search，CS)[28]、粒子群算法(PSO)[29]和差分进化算法(Differential Evolution，DE)[30]中，得到 SDS+CS 算法、SDS+PSO 算法、SDS+DE 算法，并与 CS 算法、PSO 算法、DE 算法进行对比。同时，将 SDS+PSO 算法与目前主流的几个算法进行对比，选取高斯分布衰减惯性权重粒子群优化算法(Decreasing Inertia Weight based Gaussian Function PSO，GDIWPSO)[16]、具备反向学习和局部学习能力的粒子群优化算法(Reverse-learning and Local-learning PSO，RLPSO)[13]、基于多种群的自适应迁移粒子的粒子群算法(Multi-population based Self-adaptive Migration PSO, MSMPSO)[14]这 3 个对比算法，这些算法分别从种群的规模、拓扑结构和维数方面进行了改进。

本实验采用 12 个经典测试函数，如表 1 所示。其中 F_1~F_4，F_{11}，F_{12} 为高维单峰函数，仅存在一个全局最优点；F_5~F_{10} 是高维多峰函数，存在多个局部最优点，函数的全局最优值较难寻找，并且在寻优的过程中极易陷入局部最优值。多峰函数的复杂度较高，增加寻优难度，这些测试函数的全局最优值为 0。

3.1 参数设置

实验过程中，PSO 算法和 SDS+PSO 算法的参数设置为：学习因子 $c_1 = c_2 = 2$，惯性权重 $\omega_{\max} = 0.9$，$\omega_{\min} = 0.4$，进化迭代次数 FEs=1000，种群规模 N =20，维数 D=30，K=300，S=0.3N。GA 算法和 SDS+GA 算法的参数设置为：交叉概率 P_{c} =0.70，变异概率 P_{m} =0.05，进化迭代次数 FEs=500，种群规模 N=20，维数 D =30。各算法对表 1 中的 12 个测试函数分别执行 20 次。越界粒子本文采用通用的越界处理办法，将其重新设置为边界值，如表 1 所示。在对比算法中，种群规模 N =20，维数 D=30，进化迭代次数 FEs =1000，测试次数 20 次，其余参数设置与文献[13-14,16]保持一致，参数表示如表 2 所示。

DOI: 10.16182/j.issn1004731x.joss.21-FZ0696

表 1 测试函数
Tab. 1 Test function

函数名称	测试函数	维数	取值范围
Sphere F_1	$F_2=\sum_{i=1}^{D}x_i^2$	30	$[-100,100]^n$
Rosenbrock F_2	$F_2=\sum_{i=1}^{D-1}[100\times(x_{i+1}-x_i^2)^2+(x_{i-1})^2]$	30	$[-5,10]^n$
Dixon-price F_3	$F_3=(x_1-1)^2+\sum_{i=1}^{D}i(2x_i^2-x_{i-1})^2$	30	$[-10,10]^n$
Sum of different powers F_4	$F_4=\sum_{i=1}^{D}\lvert x_i\rvert^{i+1}$	30	$[-1,1]^n$
Ackley F_5	$F_5=-a\cdot\exp\left(-b\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(cx_i)\right)a+a+\exp(1);$ $a=20,b=0.2,c=2\pi$	30	$[-32.768,32.768]^n$
Rastrigin F_6	$F_6=\sum_{i=1}^{D}(x_i^2-10\times\cos(2\pi x_i)+10)$	30	$[-5.12,5.12]^n$
Griewank F_7	$F_7=\sum_{i=1}^{D}\frac{x_i^2}{4\,000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]^n$
Powell F_8	$F_8=\sum_{i=1}^{D/4}(x_{4i-3}+10x_{4i-2})^2+5\sum_{i=1}^{D}(x_{4i-1}-x_{4i})^2+\sum_{i=1}^{D}(x_{4i-1}-2x_{4i-1})^4+10\sum_{i=1}^{D}(x_{4i-1}-x_{4i})^4$	30	$[-4,5]^n$
Levy F_9	$F_9=\sin^2(\pi\omega_1)+\sum_{i=1}^{D-1}(\omega_i-1)^2\left[1+10\sin^2(\pi\omega_i+1)\right]+$ $(\omega_D-1)^2[1+\sin^2(2\pi\omega_D)];\ \ \omega_i=1+(\omega_i-1)/4$	30	$[-10,10]^n$
Schwefel F_{10}	$F_{10}=418.982\,9D-\sum_{i=1}^{D}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$[-500,500]^n$
Sum squares F_{11}	$F_{11}=\sum_{i=1}^{D}ix_i^2$	30	$[-10,10]^n$
Zakharov F_{12}	$F_{12}=\sum_{i=1}^{D}x_i^2+(0.5\sum_{i=1}^{D}ix_i^2)^2+(0.5\sum_{i=1}^{D}ix_i^2)^4$	30	$[-5,10]^n$

表 2 对比算法的参数设置
Tab. 2 Parameter settings of compared algorithm

对比算法	文献[13-14,16]原文设置参数
GDIWPSO	$c_1=c_2=2.0$； $k=0.2$
RLPSO	$c_1=1.85$, $c_2=2.0$; c_3=0.7 , c_4=0.3 ; $w\in[0.4,0.9]$; $g=20$; $L=20$; $M=N/4$; $n=0.8\times N$
MSMPSO	$c_{11}=2.0$ $c_{12}=1.0$ $c_{13}=0.2$; $c_{21}=0.2$ $c_{22}=1.0$ $c_{23}=2.0$; $c_{31}=1.0$ $c_{32}=1.0$ $c_{33}=1.0$

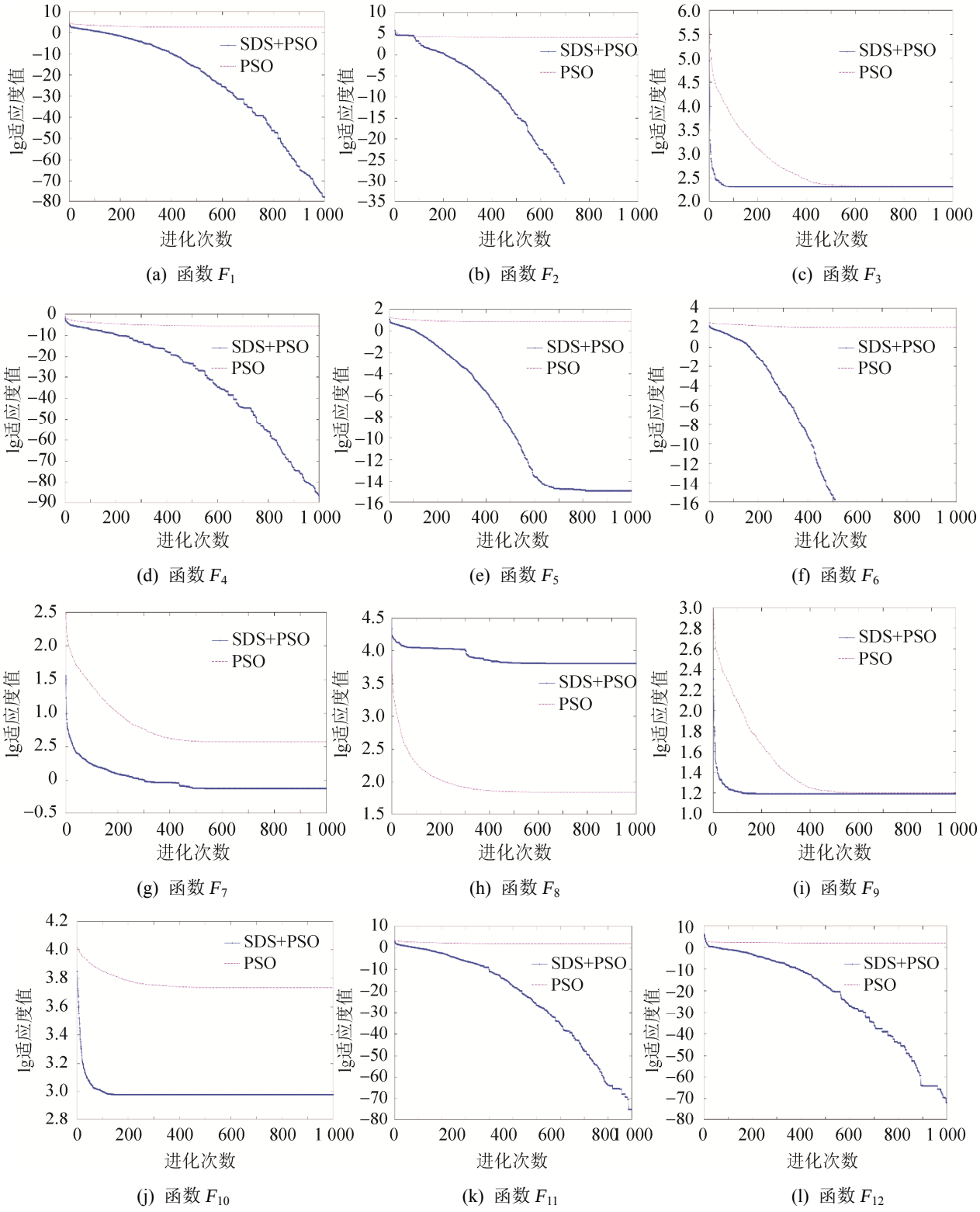
3.2 实验结果分析

3.2.1 空间分割搜索策略与 3 种标准算法对比

图 3~5 分别表示将空间分割搜索策略应用于粒子群算法、布谷鸟算法、差分进化算法中，得到 SDS+PSO，SDS+CS，SDS+DE 算法与 PSO，CS，DE 算法进行对比。

由图 3 的对比收敛图中可以看出，在单峰函数 F_1~F_4，F_{11}，F_{12} 中，SDS+PSO 算法均优于 PSO 算法，除 F_3 函数效果不是很明显外，其余函数有较大优势，尤其在函数 F_2 中在迭代大概 700 次时已经找到全局最优值，PSO 算法在迭代到一定次数时容易陷入局部最优，在函数 F_1，F_4，F_{11}，F_{12} 迭代 1 000 次后仍有下降的趋势。在多峰函数 F_5~F_{10} 中，SDS+PSO 算法在函数 F_8 的寻优能力弱于 PSO 算法，函数 F_9 中前期 SDS+PSO 算法寻优能力较好，后期与 PSO 算法持平，但是对于其他的多峰函数，

SDS+PSO 并没有陷入局部最优，并且寻找的最优值接近于全局最优，在函数 F_6 中迭代 500 次找到全局最优值，相较于粒子群算法，SDS+PSO 算法的寻优能力较强，精度较高。



(a) 函数 F_1 (b) 函数 F_2 (c) 函数 F_3

(d) 函数 F_4 (e) 函数 F_5 (f) 函数 F_6

(g) 函数 F_7 (h) 函数 F_8 (i) 函数 F_9

(j) 函数 F_{10} (k) 函数 F_{11} (l) 函数 F_{12}

图 3 SDS+PSO, PSO 实验结果图

Fig. 3 SDS+PSO, PSO experimental results diagram

DOI: 10.16182/j.issn1004731x.joss.21-FZ0696

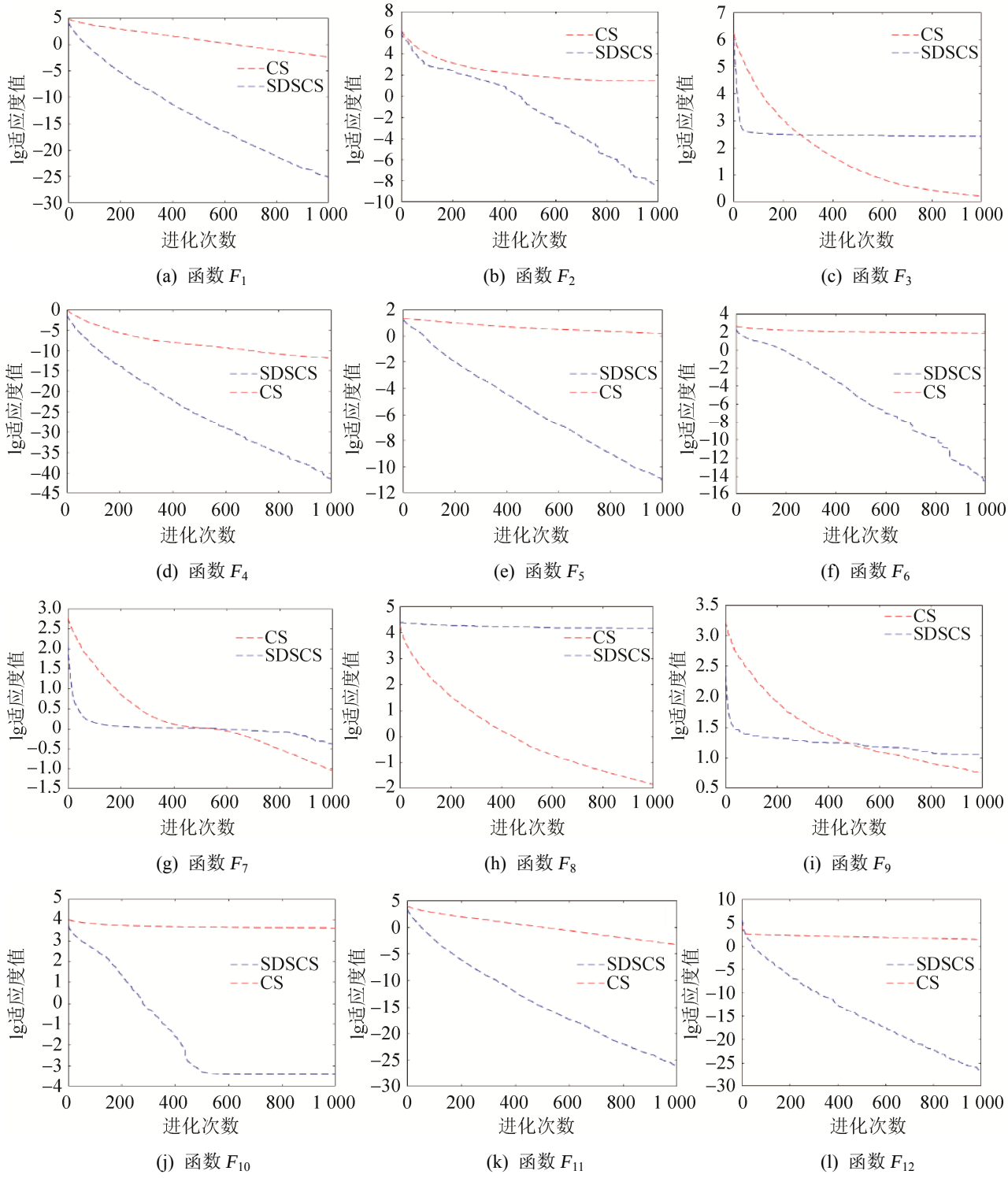


图 4　SDS+CS, CS 实验结果图

Fig. 4　SDS+CS, CS experimental results diagram

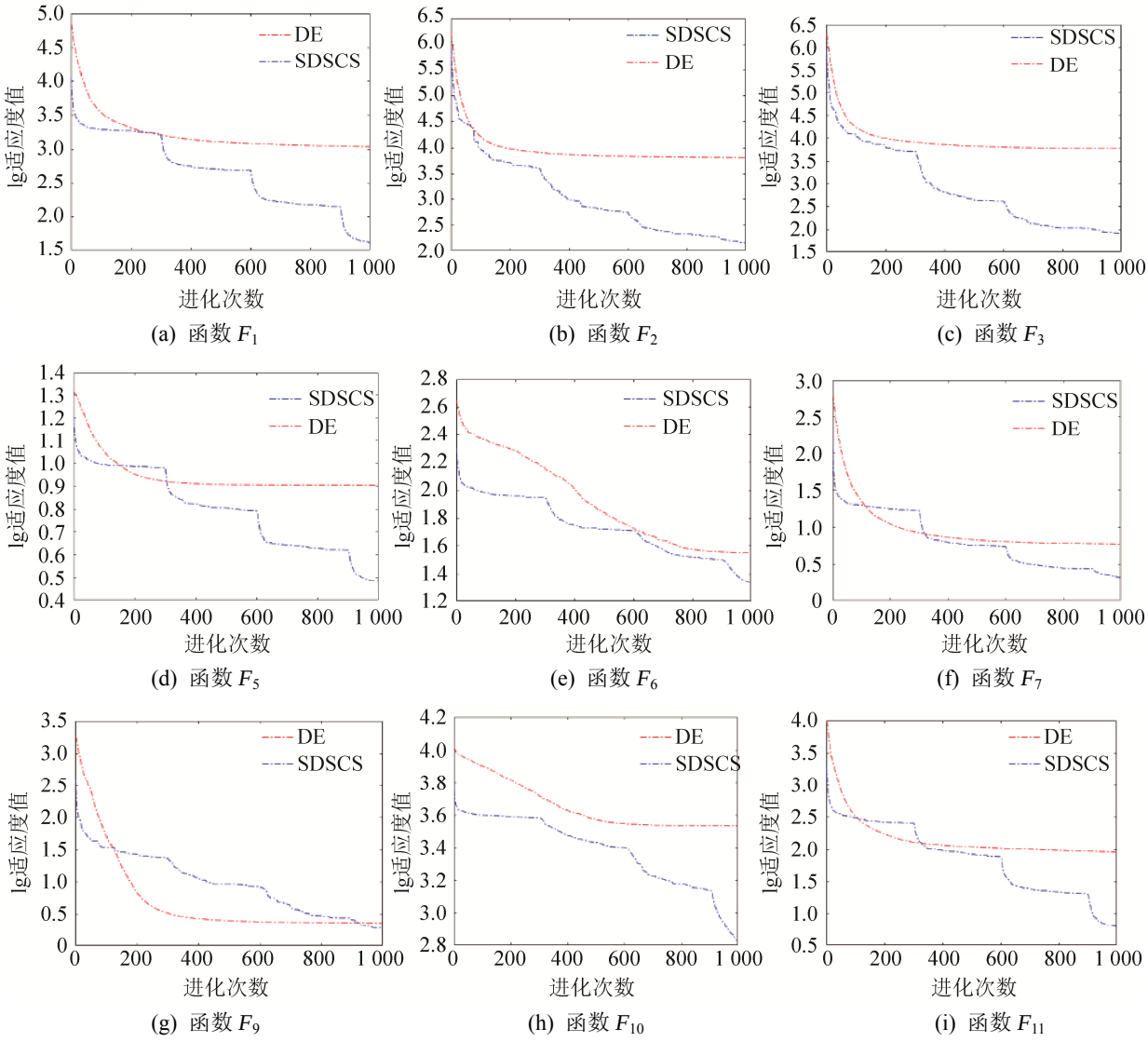


图 5 SDS+DE, DE 实验结果图

Fig. 5 SDS+DE, DE experimental results diagram

从图 4 中可以清晰地看出，使用 SDS+CS 算法策略比原布谷鸟算法 CS 有更强的全局搜索能力，但是在个别测试函数中存在一定差异。根据收敛曲线走势，SDS+CS 算法在函数 F_3，F_7，F_9 大概进化 50 次曲线保持水平，函数 F_{10} 在迭代 400 次之后曲线保持水平状态，不再有下降的趋势，说明该算法陷入局部最优的困境，这是由于未采取维数分组策略的 CS 的搜索能力更强，使用维数空间分组策略后的 SDS+CS 虽然避免了维数灾难，但没有跳出局部最优值。在函数 F_8 中 SDS+CS 算法表现并不优秀，这是因为 F_8 测试函数功能复杂，同时具有许多局部极值，导致 SDS+CS 算法早熟。在函数 F_1，F_2，F_4，F_5，F_6，F_{11}，F_{12} 进化 1 000 次之后，仍有继续下降的趋势，说明算法并没有陷入局部最优值陷阱，但是 CS 算法在进化 1 000 次之后陷入局部最优。综上所述，在 12 个测试函数中，多数函数在 SDS+CS 算法表现出较好的性能，证明了 SDS+CS 算法具有更强的搜索能力。

在 SDS+DE 算法中，收敛曲线几乎在每个测试函数进化 1 000 次之后仍有继续下降的趋势，说明算法充分发挥了维数空间分割搜索的优势，并且没有陷入局部最优值陷阱，但是 DE 算法在 1 000 次进化过程中，曲线呈现平直状态，陷入局部极值，没有找到最优解。这样也更好的证明，增加 SDS

https://dc-china-simulation.researchcommons.org/journal/vol33/iss11/7
DOI: 10.16182/j.issn1004731x.joss.21-FZ0696

策略的算法在差分进化算法中具有更强的全局搜索能力。但是在函数 F_7，F_9，F_{11} 中，虽然前期迭代过程中 DE 算法表现更加突出，这是由于 DE 算法具有较强的全局搜索能力，而在后期的迭代过程中 SDS+CS 算法超过 DE 算法，进而更接近全局最优值。验证了经过使用该维数空间分割策略后，提高了原算法寻优的精准度。收敛函数如图 5 所示。

通过图 3~5 可知，利用空间分割搜索策略得到的 SDS+PSO，SDS+CS，SDS+DE 算法实验效果整体优于 PSO，CS，DE 算法。数据结果如表 3 所示。

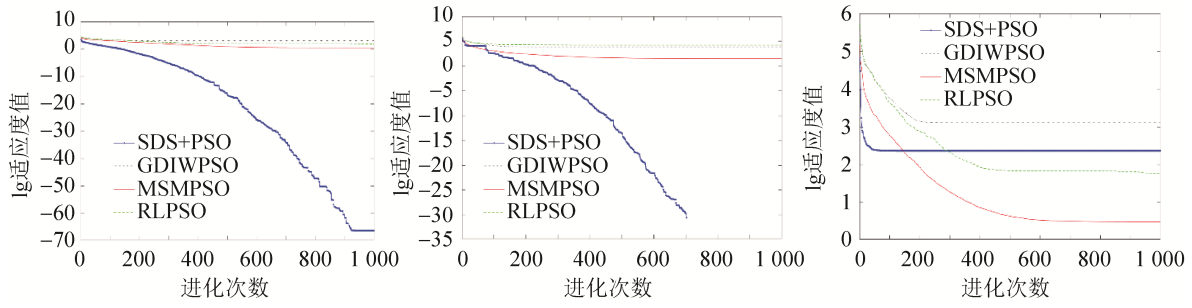
表 3 测试函数数据
Tab. 3 Test function data

测试函数		PSO	CS	DE	SDS+PSO	SDS+CS	SDS+DE
F_1	Avg	315.96	27.161 5	1 086.500 0	3.21e-74	8.19e-26	41.708 9
	Sd	107.74	14.771 0	980.331 5	1.40e-73	2.05e-25	14.613 1
	Best	77.45	11.211 8	73.342 0	2.25e-87	1.17e-27	22.327 7
F_2	Avg	2.34e+03	0.006 0	6 511.5	0	4.12e-09	147.433 0
	Sd	2.08e+04	0.005 0	9 803.6	0	1.52e-08	45.394 3
	Best	130.23	0.001 1	405.167 5	0	4.97e-13	83.436 8
F_3	Avg	232.67	1.598 3	6 009.4	242.46	269.005 3	83.265 4
	Sd	148.54	1.090 2	4925	101.59	103.499 7	32.820 9
	Best	12.41	0.676 7	380.965 6	116.06	62.546 9	30.699 6
F_4	Avg	2.49e-06	1.59e-12	0	4.97e-84	3.78e-42	1.09e-06
	Sd	3.53e-06	6.78e-12	0	2.16e-83	8.97e-42	1.04e-06
	Best	3.48e-08	9.36e-20	0	9.30e-98	1.17e-44	1.11e-07
F_5	Avg	7.15	1.505 3	8.010 2	1.24e-15	8.86e-12	3.053 5
	Sd	1.00	0.688 4	2.049 3	1.07e-15	9.37e-12	0.371 6
	Best	5.02	0.161 5	4.983 4	8.88e-16	1.67e-12	2.454 7
F_6	Avg	101.81	77.166 5	35.737 6	0	2.84e-15	21.113 2
	Sd	22.69	14.043 5	7.641 8	0	1.12e-14	3.998 8
	Best	66.05	46.399 8	22.206 3	0	0	15.747 1
F_7	Avg	4.06	0.089 8	5.921 2	0.89	0.422 8	2.106 4
	Sd	1.16	0.058 1	3.724 5	0.45	0.405 8	0.249 2
	Best	2.40	0.011 0	1.387 5	0	5.14e-06	1.721 2
F_8	Avg	58.12	0.014 4	50.515 7	5.30e+03	1.48e+04	-
	Sd	59.17	0.010 8	53.957 3	3.88e+03	2.80e+03	-
	Best	10.15	0.005 3	2.644 5	501.29	6.39e+03	-
F_9	Avg	16.71	5.712 5	2.266 8	10.65	11.290 6	1.834 1
	Sd	9.88	2.392 4	2.832 6	23.6	6.045 1	0.637 1
	Best	3.77	2.700 4	0.143 6	1.50e-32	0.611 1	0.814 7
F_{10}	Avg	5.45e+03	4 029.7	3 435.3	592.19	3.82e-04	677.927 2
	Sd	682.62	230.355 6	596.477 4	592.19	2.82e-12	496.793 7
	Best	4.43e+03	3518.2	2 485	3.82e-04	3.82e-04	144.119 0
F_{11}	Avg	118.20	0.000 6	91.800 4	1.99e-80	1.11e-26	6.169 2
	Sd	238.63	0.004 0	52.874 0	4.64e-80	1.72e-26	1.869 0
	Best	34.13	0.000 1	3.591 6	2.05e-86	4.19e-28	3.314 7
F_{12}	Avg	85.58	28.434 5	53.502 8	2.67e-75	2.56e-27	0.772 1
	Sd	37.69	6.902 4	44.184 8	1.16e-74	4.79e-27	0.245 1
	Best	34.10	15.664 1	12.897 7	2.23e-88	9.34e-29	0.270 1

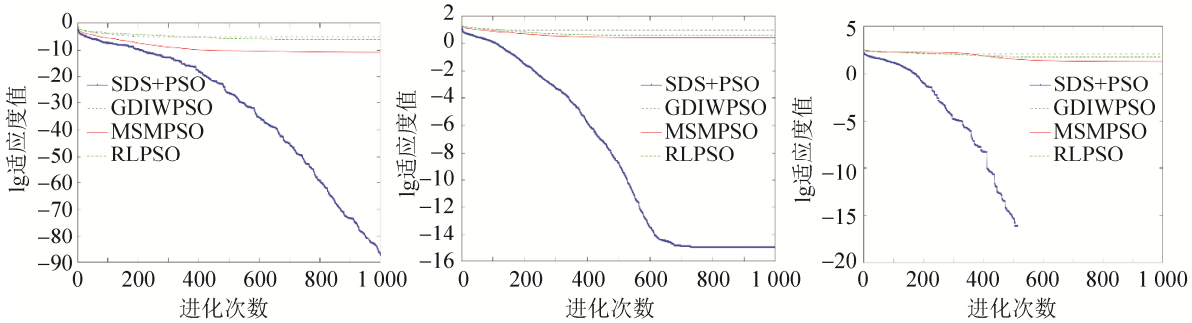
注：“-”代表无此函数数据

3.2.2 空间分割搜索策略在低维空间与改进的主流算法对比

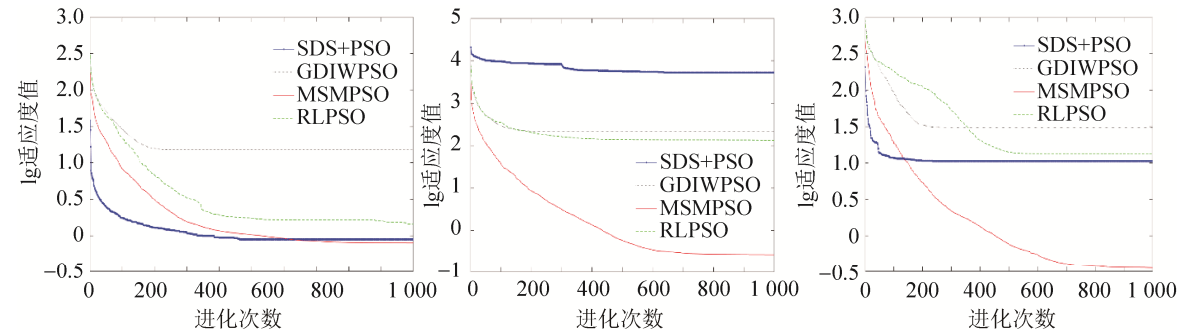
以粒子群算法为例，图 6 表示空间分割搜索策略 SDS+PSO 和其他 3 个改进的算法[13-14,16]进行对比，数据结果如表 4 所示。由收敛图可得，在单峰函数 F_1~F_4，F_{11}，F_{12} 中，除 F_3 函数的寻优能力弱于 MSMPSO，其余单峰函数在误差允许的范围内均接近全局最优值，尤其在函数 F_2 中已经寻优到全局最优值 0。其他改进算法在迭代到一定次数时陷入局部最优，而 SDS+PSO 策略没有被局部最优值所牵绊，全局搜索能力较强，在函数 F_1，F_4，F_{11}，F_{12} 中仍有下降的趋势。在多峰函数 F_5~F_{10} 中，SDS+PSO 算法在函数 F8 的寻优能力较其他算法结果较差，这是因为这个函数是一个较复杂的函数且存在许多局部极小值，导致算法容易早熟收敛。但是对于其他的多峰函数，SDS+PSO 并没有陷入局部最优，并且寻找的最优值接近于全局最优，在函数 F_6 中已经找到全局最优值，相较于 3 个改进算法，存在明显的优势。综合来说，SDS+PSO 策略利用对维数空间进行三维为一组的分组方式，同时对空间分割后的粒子编号形成分粒子，在减少维数的基础上间接增加了粒子数的规模，并利用编号索引删除部分较差粒子，有效增加了种群的多样性且运行时间变少，无论是在解决单峰还是多峰函数问题上，随着进化次数的进行，可以使粒子在求解精度和收敛速度上呈现出较为优越的性能。



(a) 函数 F_1 (b) 函数 F_2 (c) 函数 F_3

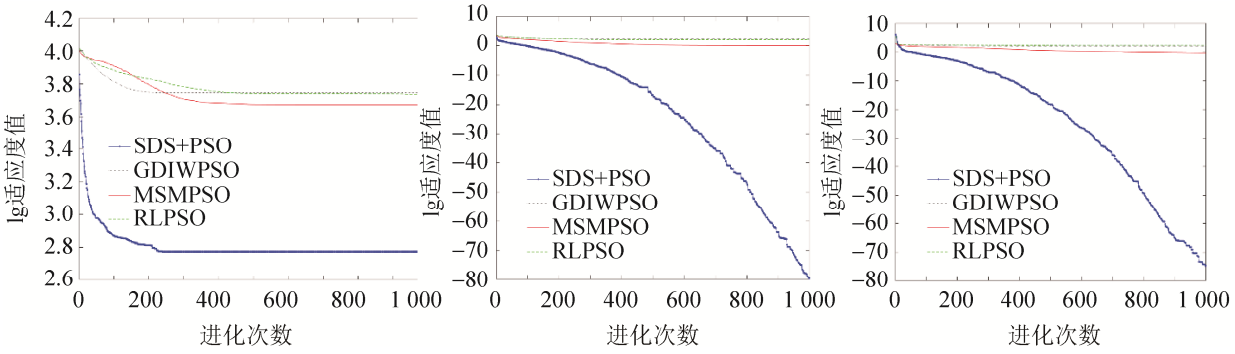


(d) 函数 F_4 (e) 函数 F_5 (f) 函数 F_6



(g) 函数 F_7 (h) 函数 F_8 (i) 函数 F_9

DOI: 10.16182/j.issn1004731x.joss.21-FZ0696



(j) 函数 F_{10} (k) 函数 F_{11} (l) 函数 F_{12}

图 6 SDS+PSO 与改进算法对比图

Fig. 6 Comparison of SDS+PSO and the improved algorithm

表 4 测试函数数据表

Tab. 4 Test function data table

测试函数		PSO	GDIWPSO	RLPSO	MSMPSO	SDS+PSO
F_1	Mean	315.96	1.32e+03	84.93	2.66	3.21e-74
	St	107.74	446.35	114.27	1.98	1.40e-73
	Best	77.45	378.87	0	0.13	2.25e-87
F_2	Mean	7.34e+03	6.88e+03	9.14e+03	39.59	0
	St	2.08e+04	1.69e+04	1.97e+04	23.27	0
	Best	130.23	459.68	71.96	28.37	0
F_3	Mean	232.67	1.39e+03	43.39	1.86	242.46
	St	148.54	999.37	57.79	1.36	101.59
	Best	12.41	68.53	0.67	0.68	116.06
F_4	Mean	2.49e-06	7.26e-06	**8.79e-07**	2.90e-12	4.97e-84
	St	3.53e-06	8.13e-06	**1.02e-06**	5.35e-12	2.16e-83
	Best	3.48e-08	1.51e-07	3.31e-14	3.05e-25	9.30e-98
F_5	Mean	7.15	9.16	2.54	**2.40**	**1.24e-15**
	St	1.00	1.45	2.26	**0.68**	**1.07e-15**
	Best	5.02	5.98	8.88e-16	**0**	**8.88e-16**
F_6	Mean	101.81	122.66	59.61	19.98	0
	St	22.69	23.33	28.91	9.16	0
	Best	66.05	78.84	13.76	10.08	0
F_7	Mean	4.06	15.04	1.44	0.80	0.89
	St	1.16	19.73	0.85	0.29	0.45
	Best	2.40	4.67	0	0.14	0
F_8	Mean	58.12	214.16	132.70	0.27	5.30e+03
	St	59.17	409.10	328.73	0.22	3.88e+03
	Best	10.15	11.63	5.82	6.01e-05	501.29
F_9	Mean	16.71	30.58	13.31	0.36	10.65
	St	9.88	12.33	9.41	0.30	23.60
	Best	3.77	11.55	1.90	0.04	1.50e-32
F_{10}	Mean	5.45e+03	5.58e+03	5.44e+03	4.71e+03	592.19
	St	682.62	785.89	1.42e+03	931.92	592.19
	Best	4.43e+03	4.50e+03	2.66e+03	3.08e+03	3.82e-04
F_{11}	Mean	118.20	241.80	120.17	1.18	1.99e-80
	St	238.63	258.20	211.05	0.66	4.64e-80
	Best	34.13	72.78	1.61	0.27	2.05e-86
F_{12}	Mean	85.58	114.44	142.30	0.50	2.67e-75
	St	37.69	41.99	115.71	0.47	1.16e-74
	Best	34.10	47.33	41.82	0.04	2.23e-88

3.2.3 空间分割搜索策略在高维空间与主流进化算法对比

将本文提出的 DMS-CC[12]、CSO[21]、SLPSO[20]以及 DMS-PSO[11]进行对比，这些算法均为针对求解大规模高维复杂问题而提出的算法，而文中提出的空间分割搜索策略也是针对高维问题做出的改进。空间分割搜索策略采用的维数为 1 200 维，对比函数的维数选取均为 1 000 维，其余参数规模的选取与文献[17]中均保持一致。为了与文献中测试保持一致，因此，采用 F_1，F_4，F_5，F_6，F_7 和 F_{10} 这 6 个测试函数，如表 5 所示，算法收敛图如图 7 所示。

从表 5 中可以直观地看到，SDS+PSO 在这 6 个测试函数中的寻优结果均优于其他 4 个算法，并且采用该策略的维数高于对比算法，说明实验结果更优。在收敛图中可以发现，F_7 在迭代 600 次的时候找到全局最优解，停止迭代。利用空间分割搜索策略，可以使高维间接转换为求解低维问题的算法，维数降低的同时，利用编号索引导致种群规模增多，来增加种群多样性。同时，通过计算适应度值，进行排序，删除部分较差个体，用于平衡时间效能，这样就使得种群在不丧失多样性的前提下降低了维数，避免了种群陷入局部最优，提升了算法的效率。而其他 4 个算法是针对大规模高维复杂问题中维数本身做了相应的改进，其适用性较弱。对比这 4 个算法，其实验结果表明，本文提出的空间分割搜索策略有很好的寻优效果。

3.3 显著性检验

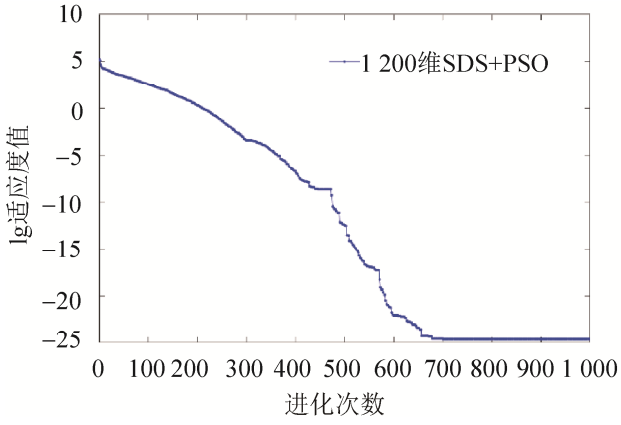
为了对空间搜索策略算法结果有一个更加全面的了解，以粒子群算法为例，对 SDS+PSO 和其他算法在 D=30 时得到的结果逐一进行显著性检验。由于不能够确定各算法多次优化同一函数的结果服从分布的类型，因此采用非参数检验的方法，这里选用 Mann Whitney 检验方法[31]，显著水平 α=0.05 。

表 5 高维空间中对比函数数据表

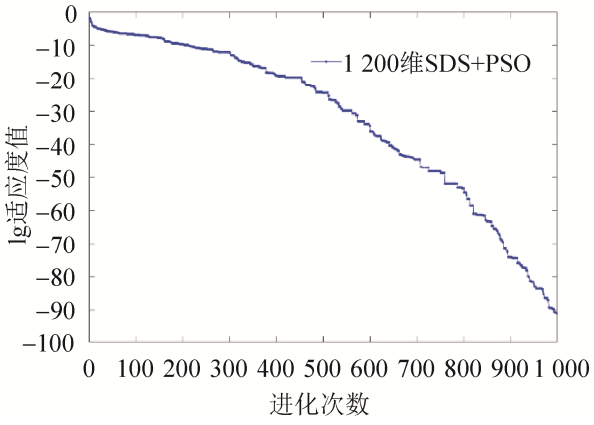
Tab. 5 Comparison function data table in high dimensional space

算法		F_1	F_4	F_5	F_6	F_7	F_{10}
DMS-CC	Best	13.2	5.88e+05	1.60e+14	2.30e+06	2.20e+04	2.00e+11
SLPSO	Best	4.78e-14	3.82e+06	1.90e+14	5.15e+09	9.42e+06	-
DMS-PSO	Best	1.66e+01	1.14e+08	4.36e+07	1.68e+09	9.23e+04	4.51e+10
CSO	Best	2.43e-09	3.71e+06	4.61e+14	3.25e+09	9.68e+06	2.13e+11
SDS+PSO	Best	**3.16e-78**	**1.26e-111**	**8.88e-16**	**0**	**0**	**0.02**

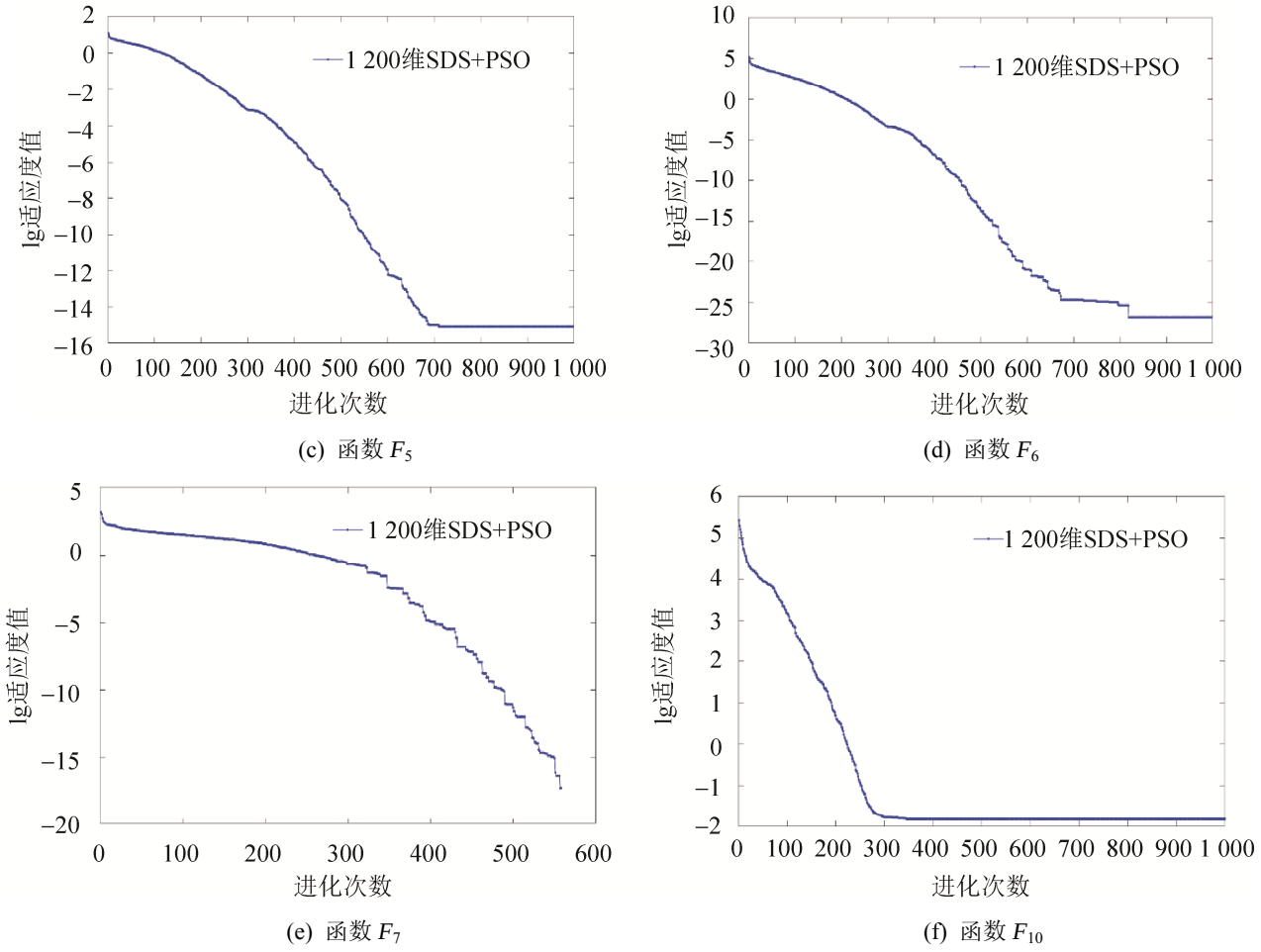
注：“-”代表无此函数数据



(a) 函数 F_1



(b) 函数 F_4



(c) 函数 F_5 (d) 函数 F_6

(e) 函数 F_7 (f) 函数 F_{10}

图 7 SDS+PSO 在高维空间中与改进算法对比图

Fig. 7 Comparison of SDS+PSO with the improved algorithm in high dimensional space

表 6 详细列出了 4 种算法的检验数据，这里以 PSO 列为例说明表中各数据的意义：PSO 列与 F_1 行交叉的数据为“0.000↑”，其中“0.000”为 Mann Whitney 检验的 p 值；“↑”表示 PSO 在 F_1 上的运行数据的秩均值大于 SDS+PSO 的运行数据的秩均值；“↓”表示 PSO 在此函数上的运行数据的秩均值小于 SDS+PSO 的运行数据的秩均值；“→”表示 2 种算法的运行数据的秩均值相等。表中的 better 行、same 行和 worse 行分别表示在显著水平为 0.05 下，12 个测试函数中 SDS+PSO 优于 PSO、与 PSO 无差和劣于 PSO 的函数的个数。从表 6 中可以得出，在 12 个测试函数中，SDS+PSO 和其他算法的综合性能相比，表现出较好的性能。

表 6 D=30，Mann Whitney 检验数据

Tab. 6 D=30，Mann Whitney test data

函数	PSO	MSMPSO	RLPSO	GDOIWPSO
F_1	0.000↑	0.000↑	0.000↑	0.000↑
F_2	0.000↑	0.000↑	0.000↑	0.000↑
F_3	0.659↓	0.000↓	0.000↓	0.000↑
F_4	0.000↑	0.000↑	0.000↑	0.000↑
F_5	0.000↑	0.000↑	0.000↑	0.000↑
F_6	0.000↑	0.000↑	0.000↑	0.000↑
F_7	0.000↑	0.158↓	0.003↑	0.000↑
F_8	0.000↓	0.000↓	0.000↓	0.000↓
F_9	0.014↑	0.023↓	0.011↑	0.000↑
F_{10}	0.000↑	0.000↑	0.000↑	0.000↑
F_{11}	0.000↑	0.000↑	0.000↑	0.000↑
F_{12}	0.000↑	0.000↑	0.000↑	0.000↑
better	10	8	10	11
same	0	0	0	0
worse	2	4	2	1

4 结论

本文提出了一种基于空间分割搜索(SDS)策略的自然计算方法，采用维数空间进行分组的方式，对空间分割后的粒子编号形成分粒子，不仅能将高维空间的复杂维数空间转化为三维空间的可观测问题，并且在减少维数的基础上间接增加了粒子数的规模，提高种群多样性的同时也找到了精度更高的解。同时，通过适应度值的计算，删除部分较差个体，相较于逐维变异和逐维学习的方式，空间分割搜索算法在时间上存在较大的优势，并且使用马尔可夫的证明过程分析了 SDS 算法的收敛性。通过实验证明，该策略的使用比标准算法有更好的全局搜索能力和求解精度。但本策略的不足之处是在平衡时间效能方面，删除粒子采用的是随机删除适应度较差的粒子，缺乏一种行之有效的删除机制，这个问题在以后的研究中可考虑结合 Logistic 模型创新种群删除机制，进一步提高算法的有效性。

参考文献：

[1] Martín-Vide C, Vega-Rodríguez M A. Theory and Practice of Natural Computing: Fifth Edition[J]. Soft Computing - A Fusion of Foundations, Methodologies and Applications (S1432-7643), 2019, 23(5): 1421.

[2] Karaboga D, Ozturk C. A Novel Clustering Approach: Artificial Bee Colony (ABC) Algorithm[J]. Applied Soft Computing (S1568-4946), 2011, 11(1): 652-657.

[3] 段海滨. 蚁群算法原理及其应用[M]. 北京: 科学出版社, 2005.
Duan Haibin. Principle and Application of Ant Colony Algorithm [M]. Beijing: Science Press, 2005.

[4] He S, Wu Q H, Saunders J R, et al. Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior[J]. IEEE Trans Evolutionary Computation (S1089-778X), 2009, 13(5): 973-990.

[5] El-Toukhy Y M, Heikal A M, Hameed M F O, et al. Optimization of Nanoantenna for Solar Energy Harvesting based on Particle Swarm Technique[C]// 2016 IEEE/ACES International Conference on Wireless Information Technology and Systems (ICWITS) and Applied Computational Electromagnetics (ACES), Honolulu, HI, 2016: 1-2. doi: 10.1109/ROPACES.2016.7465458.

[6] 康琦, 安静, 汪镭, 等. 自然计算的研究综述[J]. 电子学报, 2012, 40(3): 548-558
Kang Qi, An Jing, Wang Lei, et al. Research Review of Natural Computing [J]. Acta Electronica Sinica, 2012, 40(3): 548-558.

[7] 王蓉芳, 焦李成, 刘芳, 等. 自适应动态控制种群规模的自然计算方法[J]. 软件学报, 2012(7): 130-142.
Wang Rongfang, Jiao Licheng, Liu Fang, et al. Natural Calculation Method for Adaptive Dynamic Control of Population Size [J]. Journal of Software, 2012(7): 130-142 .

[8] Sun S, Li J. A Two-swarm Cooperative Particle Swarms Optimization[J]. Swarm & Evolutionary Computation (S2210-6502), 2014, 15: 1-18.

[9] Xu G, Cui Q, Shi X, et al. Particle Swarm Optimization based on Dimensional Learning Strategy[J]. Swarm and Evolutionary Computation (S2210-6502), 2019, 45: 33-51.

[10] Wei B, Xia X W, Yu F, et al. Multiple Adaptive Strategies based Particle Swarm Optimization Algorithm[J]. Swarm and Evolutionary Computation (S2210-6502), 2020, 57: 100731.

[11] Sabar N R, Abawajy J, Yearwood J, et al. Heterogeneous Cooperative Co-evolution Memetic Differential Evolution Algorithm for Big Data Optimization Problems [J]. IEEE Transactions on Evolutionary Computation (S1089-778X), 2017, 21(2): 315-327.

[12] 梁静, 刘睿, 于坤杰, 等. 求解大规模问题协同进化动态粒子群优化算法[J]. 软件学报, 2018, 29(9): 2595-2605.
Liang Jing, Liu Rui, Yu Kunjie, et al. Coevolutionary Dynamic Particle Swarm Optimization Algorithm for Solving Large-scale Problems [J]. Journal of Software, 2018, 29(9): 2595-2605.

[13] 夏学文, 刘经南, 高柯夫, 等. 具备反向学习和局部学习能力的粒子群算法[J]. 计算机学报, 2015, 38(7): 1397-1407.
Xia Xuewen, Liu Jingnan, Gao Kefu, et al. Particle Swarm Optimization with Reverse Learning and Local Learning [J]. Acta Computerica Sinica, 2015, 38(7): 1397-1407.

[14] 邓先礼, 魏波, 曾辉, 等. 基于多种群的自适应迁移 PSO 算法[J]. 电子学报, 2018, 46(8): 1858-1865.
Deng Xianli, Wei Bo, Zeng Hui, et al. Adaptive Migration PSO Algorithm based on Multiple Populations [J]. Acta Electronica Sinica, 2018, 46(8): 1858-1865.

[15] Kai Z, Qiu J H, Yi M Z. Enhancing Comprehensive

https://dc-china-simulation.researchcommons.org/journal/vol33/iss11/7
DOI: 10.16182/j.issn1004731x.joss.21-FZ0696

Learning Particle Swarm Optimization with Local Optima Topology[J]. Information Sciences (S0020-0255), 2019, 471: 1-18.

[16] 张迅, 王平, 邢建春. 基于高斯函数递减惯性权重的粒子群优化算法[J]. 计算机应用研究, 2012, 29(10): 3710-3712, 3724.
Zhang Xun, Wang Ping, Xing Jianchun. Particle Swarm Optimization Algorithm based on Decreasing Inertia Weight of Gaussian Function [J]. Journal of Computer Applications, 2012, 29(10): 3710-3712, 3724.

[17] 喻飞, 李元香, 魏波, 等. 透镜成像反学习策略在粒子群算法中的应用[J]. 电子学报, 2014, 42(2): 230-235.
Yu Fei, Li Yuanxiang，Wei Bo, et al. Application of Anti-Learning Strategy for Lens Imaging in Particle Swarm Optimization [J]. Acta Electronica Sinica, 2014, 42(2): 230-235.

[18] 罗强, 季伟东, 徐浩天, 等. 一种最优粒子逐维变异的粒子群优化算法[J]. 小型微型计算机系统, 2020, 41(2): 259-263.
Luo Qiang, Ji Weidong, Xu Haotian, et al. A Particle Swarm Optimization Algorithm with Optimal Particle Dimensional Variation [J]. Miniature Microcomputer System, 2020, 41(2): 259-263.

[19] 唐祎玲, 江顺亮, 叶发茂, 等. 最优粒子增强探索粒子群算法[J]. 计算机工程与应用, 2017, 53(4): 25-32.
Tang Yiling, Jiang Shunliang, Ye Famao, et al. Optimal Particle Enhancement Exploration Particle Swarm Optimization Algorithm [J]. Computer Engineering and Application, 2017, 53(4): 25-32.

[20] Cheng R, Jin Y C. A Social Learning Particle Swarm Optimization Algorithm for Scalable Optimization [J]. Information Sciences (S0020-0255), 2015, 291(6): 43-60.

[21] Cheng R, Jin Y C. A Competitive Swarm Optimizer for Large Scale Optimization [J]. IEEE Transactions on Cybernetics (S2168-2267), 2014, 45(2): 191-204.

[22] Zhou J, Fang W, Wu X, et al. An Opposition-based Learning Competitive Particle Swarm Optimizer[C]// 2016 IEEE Congress on Evolutionary Computation (CEC 2016). Vancouver，BC，Canada: IEEE, 2016: 515-521.

[23] 张雯雾, 王刚, 朱朝晖, 等. 粒子群优化算法种群规模的选择[J]. 计算机系统应用, 2010, 19(5): 125-128.
Zhang Wenwu, Wang Gang, Zhu Chaohui, et al. Selection of Particle Swarm Optimization Algorithm Population Size [J]. Computer System Application, 2010, 19(5): 125-128.

[24] Xu G, Cui Q, Shi X, et al. Particle Swarm Optimization based on Dimensional Learning Strategy[J]. Swarm and Evolutionary Computation (S2210-6502), 2019, 45: 33-51.

[25] 张孟健, 龙道银, 王霄, 等. 基于马尔科夫链的灰狼优化算法收敛性研究[J]. 电子学报, 2020, 48(8): 1587-1595.
Zhang Mengjian, Long Daoyin, Wang Xiao, et al. Convergence of Grey Wolf Optimization Algorithm Based on Markov Chain [J]. Acta Electronica Sinica, 2020, 48(8): 1587-1595.

[26] Solis F J, Wets J B. Minimization by Random Search Techniques [J]. Mathematics of Operations Research (S0364-765X), 1981, 6(1): 19-30

[27] 潘峰, 周倩, 李位星, 等. 标准粒子群优化算法的马尔科夫链分析[J]. 自动化学报, 2013, 39(4): 381-389.
Pan Feng, Zhou Qian, Li Weixing, et al. Markov Chain Analysis of Standard Particle Swarm Optimization Algorithm [J]. Acta Automatica Sinica, 2013, 39(4): 381-389.

[28] 孟凡超, 初佃辉, 李克秋, 等. 基于混合遗传模拟退火算法的 SaaS 构件优化放置[J]. 软件学报, 2016, 27(4): 916-932.
Meng Fanchao, Chu Dihui, Li Keqiu, et al. Solving SaaS Components Optimization Placement Problem with Hybrid Genetic and Simulated Annealing Algorithm[J]. Journal of Software, 2016, 27(4): 916-932.

[29] Kennedy J, Eberhart R. Particle Swarm Optimization [C]//IEEE International Conference on Neural Networks. Piscataway: IEEE Service Center, 1995: 1942-1948.

[30] Price K, Storn R, Lampinen J. Differential Evolution: a Practical Approach to Global Optimization[M]. Berlin, Germany: Springer-Verlag, 2005.

[31] 王东风, 孟丽, 赵文杰. 基于自适应搜索中心的骨干粒子群算法[J]. 计算机学报, 2016, 39(12): 2652-2667.
Wang Dongfeng, Meng Li, Zhao Wenjie. Improved Bare Bones Particle Swarm Optimization with Adaptive Search Center[J]. Chinese Journal of Computers, 2016, 39(12): 2652-2667.

