DOI:10.16451/j.cnki.issn1003-6059.2013.11.009



# 求解连续函数优化问题的合作协同进化布谷鸟搜索算法*

胡欣欣[1] 尹义龙[2]

[1](福建农林大学 计算机与信息学院 福州 350002)
[2](山东大学 计算机科学与技术学院 济南 250101)

**摘 要** 为改善布谷鸟搜索算法求解连续函数优化问题的性能,提出合作协同进化的布谷鸟搜索算法.改进算法通过应用合作协同进化框架,将种群的解向量分解成若干子向量,并构成相应子群体.利用标准布谷鸟算法更新各子群体的解向量.各子群体为其它子群体提供最优个体,组合成问题解向量并完成子群体评价.经 10 个测试函数实验仿真,结果说明改进算法能有效改善求解连续函数优化问题的性能.同时,针对连续函数优化问题,该算法与其它算法相比是有竞争力的优化算法.

**关键词** 布谷鸟搜索算法,合作协同进化,子群体,函数优化问题,分解
**中图法分类号** TP 181

# Cooperative Co-Evolutionary Cuckoo Search Algorithm for Continuous Function Optimization Problems

HU Xin-Xin[1], YIN Yi-Long[2]
[1](*School of Computer and Information Science*, *Fujian Agriculture and Forestry University*, *Fuzhou* 350002)
[2](*School of Computer Science and Technology*, *Shandong University*, *Jinan* 250101)

## ABSTRACT

To improve the performance of cuckoo search algorithm for continuous function optimization problems, a cooperative co-evolutionary cuckoo search algorithm is proposed. Through the framework of cooperative co-evolutionary, the improved algorithm divides the solution vectors of population into several sub-vectors and constructs the corresponding sub-swarms. The solution vectors of each sub-population are updated by the standard cuckoo search algorithm. Each sub-population provides the vectors of the best solution, which are combined with solution vectors of other sub-populations, and the combined solution vectors are evaluated. The simulation experiments on 10 benchmark functions show that the proposed algorithm efficiently improves the performances on contnuous function optimization problems and it is a competitive optimization algorithm for the problems compared with other algorithms.

* NSFC-广东省联合基金重点支持项目(No. U1201258)、教育部新世纪优秀人才支持计划项目(No. NCET-11-0315)、山东省自然科学杰出青年基金项目(No. JQ201316)、福建省自然科学基金项目(No. 2011J05044,2013J01216)资助
收稿日期:2013-05-13
作者简介 胡欣欣,女,1982 年生,博士,实验师,主要研究方向为智能算法及应用. E-mail: xinxinhu@fafu.edu.cn. 尹义龙(通讯作者),男,1972 年生,教授,博士生导师,主要研究方向为机器学习、数据挖掘、图像处理. E-mail: ylyin@sdu.edu.cn.

**Key Words** Cuckoo Search Algorithm, Cooperative Co-Evolutionary, Sub-Swarm, Function Optimization Problems, Decomposition

## 1 引 言

优化问题是普遍存在于工程设计、科学研究及经济管理等领域. 由于其目标函数可具有不同的特征,如连续性、多峰、变换、旋转及噪声等[1-2],使得传统的数值优化算法在解的质量、收敛性及计算速度等方面远不能满足要求. 随着仿生学学科的发展,出现具有适合高度并行及自组织、自适应和自学习等特征的仿生优化算法,如遗传算法[3]、粒子群算法[4-5]、蚁群算法[6]和差分演化算法[7]等,并被成功用于求解优化问题. 近几年,借助于模拟自然界规律与过程的思想及手段,出现蜂群优化算法[8]、生物地理优化算法[9]和布谷鸟搜索(Cuckoo Search, CS)算法[10-11]等新颖的仿生优化算法.

Yang 和 Deb[10]源于布谷鸟特殊的繁殖策略,抽象出 CS 算法,并用于求解工程设计问题及连续函数优化问题. 文献[10~12]比较分析 CS 算法与遗传算法、粒子群算法、人工蜂群算法及差分演化算法的性能及参数,确定 CS 算法不仅具有简单、参数少、易于实现等优点,且性能接近于标准的粒子群优化算法和差分演化算法等算法. 目前,CS 算法已在工程设计[11,13-14]、多目标优化问题[15]、测试数据自动生成[16]、参数估算[17]及连续函数优化问题[18-23]等方面得到应用.

借助于 Lévy Flights 随机游动组件和偏好随机游动组件,CS 算法在求解优化问题时得到较好的性能. 然而,CS 算法作为一种新的随机搜索算法,理论研究需要完善及其收敛速度和解的质量需进一步提高. 因此,众多学者对 CS 算法进行相关研究,特别针对连续函数优化问题提出相关改进版本,包括两组件中的步长改进[18-19]、参数自适应[20]及与其它算法混合[21-23]等方面. 但根据没有免费午餐理论[24],各种改进算法都有其固有的缺陷,如适应能力不强、搜索效果不够理想,对复杂问题求解能力有限等,不可能满足所有要求,因此,有必要探索新的改进方法或策略.

另一方面,Potter 和 de Jong[25]提出求解函数优化问题的合作协同进化(Cooperative Co-Evolutionary, CC)通用框架,并成功应用于求解复杂的优化问题[25-31],同时验证具有勘探潜在解并加快收敛的能力. 由于框架具有通用性,其它优化算法可作为子组件,相继出现合作协同进化遗传算法[25]、合作协同进化(1+1)演化算法[32]、合作粒子群优化算法[33]和合作协同进化差分算法[34-35]等.

因此,本文应用 CC 通用框架,将 CS 算法作为框架的子组件,提出合作协同进化的布谷鸟搜索算法(Cooperative Co-evolutionary Cuckoo Search, Co2CS),以改善 CS 算法求解连续函数优化问题的性能.

## 2 布谷鸟搜索算法

Yang 和 Deb 通过模拟布谷鸟寄生育雏行为及鸟类或果蝇 Lévy flights 行为,抽象出布谷鸟搜索算法. 该算法基于 3 条理想的规则[10-11].

**规则 1** 每只布谷鸟一次只产一颗蛋,并随机选择一个鸟巢存放.

**规则 2** 蛋最好的鸟巢将会被保留至下一代.

**规则 3** 可用鸟巢的数量是固定的,且鸟巢中外来蛋被发现的概率是 $p_a \in [0,1]$.

基于上述的 3 条规则,结合文献[11]的实现代码,CS 算法的基本流程如下.

**算法 1** CS 算法

```
Begin 初始化种群 n host nests X_i(i=1,2,…,n);
计算适应值 F_i(i=1,2,…,n);
While(不满足停止条件)
    采用 Lévy flights 生成的新解 X_i
    计算新解 X_i 的适应值 F_i
    选择候选解 X_j
      If (F_i>F_j)
        用新的解替代候选解
      End
    按发现概率 p_a 丢弃差的解
    用偏好随机游动产生新的解替代丢弃的解
    保留最好解
    End
End
```

在 CS 算法中,Lévy Flights 随机游动和偏好随机游动是两个关键的组件,分担着局部搜索和全局搜索. 在迭代过程中,CS 算法首先在当前解的基础上以 Lévy Flights 随机游动方式生成新的解,评价后以贪婪方式选择较好的解; 其次,为增加多样性,以概率 $p_a$ 放弃部分解; 最后,采用偏好随机游动方式重新生成与被放弃解相同数量的新解,在评价并保留较好的解之后,完成一次迭代.

在 Lévy Flights 组件中,算法采用式(1)生成 F-化解 $X_{t+1,i}$:

$$X_{t+1,i}=X_{t,i}+\alpha\oplus Lévy(\beta)\ , \tag{1}$$

其中,$X_{t,i}$表示第 $t$ 代第 $i$ 个解,$\alpha$ 是步长信息,用于控制随机搜索的范围:

$$\alpha=\alpha_0(X_{t,i}-X_{best})\ ,$$

其中,$\alpha_0$是常数($\alpha_0=0.01$),$X_{best}$表示当前最优解.

式(1)中,$\oplus$是点乘积(Entry-Wise Multiplications),$Lévy(\beta)$是 Lévy 随机概率函数:

$$Lévy(\beta)\sim u=t^{-1-\beta},\quad 0<\beta\leqslant 2\ .$$

为便于计算,文献[11]和文献[15]认为:

$$Lévy(\beta)\sim\frac{\phi u}{|v|^{\frac{1}{\beta}}}\ .$$

其中,$u$,$v$ 服从标准正态分布,$\beta=1.5$,

$$\phi=\left(\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\left(\frac{1+\beta}{2}\right)\beta\times 2^{\frac{\beta-1}{2}}\right)}\right)^{\frac{1}{\beta}}\ .$$

整合上述公式,在 Lévy Flights 组件中,生成新的解 $X_i$的完整公式可归纳为

$$X_{t+1,i}=X_{t,i}+\alpha_0\frac{\phi u}{|v|^{\frac{1}{\beta}}}(X_{t,i}-X_{best})\ ,$$

在偏好随机游动组件中,算法以混合变异和交叉操作的方式重新生成若干个新解:

$$X_{t+1,i}=X_{t,i}+r(X_{t,j}-X_{t,k})\ ,$$

其中,$r$ 是缩放因子,是(0,1)区间的均匀分布随机数,$X_j$和 $X_k$是两个随机的解.

## 3　合作协同进化的布谷鸟搜索算法

合作协同进化为演化算法求解高维或复杂的优化问题提供通用框架,其采用"分而治之"策略. 在 CC 框架中,目标优化问题的解向量被分解成 $K$ 个子向量,构成 $K$ 个子群体,每个子群体单独利用演化算法进化. 在评价每个子群体时,其它子群体通过合作方式为该子群体提供相关信息,组合成完整种群. 求解高维函数优化问题的 CC 框架详细见文献[25]、[32]~[35].

根据 CC 思想,Co2CS 算法首先将种群中的解向量 $\boldsymbol{X}_i=[x_{i1},x_{i2},\cdots x_{iD}]\ (i=1,2,\cdots,N)$ 平均分解成 $K$ 个子向量 $\boldsymbol{S}_k\boldsymbol{X}_i=[x_{i1},x_{i2},\cdots x_{iD/K}]\ (k=1,2,\cdots,K;\ i=1,2,\cdots,n)$. 其次,在第 $k$ 个子群体中,利用 CS 算法中的 Lévy Flights 组件和偏好随机游动组件更新解向量 $\boldsymbol{S}_k\boldsymbol{X}_i$. 由于优化问题是 $D$ 维,而每个子群体却是 $D/K$ 维,所以在评价某个子群体时,需要其它子群体提供解向量重新组合成 $D$ 维解向量. 较为简单组合模式是其它子群体提供最优解向量作为被评价子群体中剩余的($D-D/K$)维,如第 $k$ 个子群体 $D$ 维解向量可表示为 $\boldsymbol{S}_1\boldsymbol{X}_{best},\boldsymbol{S}_2\boldsymbol{X}_{best},\cdots,\boldsymbol{S}_{k-1}\boldsymbol{X}_{best},\boldsymbol{S}_k\boldsymbol{X}_i,\boldsymbol{S}_{k+1}\boldsymbol{X}_{best},\cdots,\boldsymbol{S}_K\boldsymbol{X}_{best},\boldsymbol{S}_{k+1}\boldsymbol{X}_{best}$表示第 $k+1$ 个子群体的最优解. Co2CS 算法的流程如下.

**算法 2**　Co2CS 算法

```
Begin
  初始化种群: X_{i,j}(i=1,2,…,N; j=1,2,…D);
  分解解向量为 K 个子向量,构成 K 子群体
  初始化各子群体的 bestnest
  While(不满足停止条件)
    Foreach 子群体 K
      采用 Lévy Flights 生成新的解 S_kX_i
      组合子群体的 m 维向量(S_1X_best, S_2X_best, …, S_{k-1}
      X_best, S_{k+1}X_best, …, S_KX_best)
        评价子群体,并保留较好的解
      按发现概率 p_a 丢弃差的解;
      采用偏好的随机游动生成新的解 S_kX_i
      组合子群体的 m 维向量(S_1X_best, S_2X_best, …, S_{k-1}
      X_best, S_{k+1}X_best, …, S_KX_best)
      评价子群体,并保留较好的解
      更新整个种群最优解
    End
  End
End
```

## 4　实验与结果分析

为验证 Co2CS 算法求解连续函数优化问题的性能,选用文献[1]中的前 10 个测试函数,其中,$F_1\sim F_5$ 是单峰函数,$F_6\sim F_{10}$多峰函数; $F_1$、$F_9$ 是变量间独立的函数,其它函数是变量间相关联的函数; $F_1$、$F_2$、$F_4$、$F_6$ 和 $F_9$ 函数具有变换的特点,$F_3$、$F_7$、$F_8$ 和 $F_{10}$函数具有变换且旋转的特点,而 $F_5$ 函数的最优解在边界,使得 10 个函数很难求解.

为评估算法性能,采用如下评价准则.

1) 适应值误差(Error)[1]. 如下式所示:

$$Error=f(X)-f(X^*)\ , \tag{2}$$

其中,$X$ 表示算法得到的解,$X^*$ 是函数全局最优解. 从式(2)可知,适应值误差越小,解的质量越好.

2) 函数评价次数(NFEs)[1]. 当算法在每次运行时,在当前函数评价次数没有达到最大函数评价次数且最优解的适应值误差小于指定误差阈值所消耗

的函数评价次数，其中，$F_1 \sim F_5$ 误差阈值是 $10^{-6}$，而 $F_6 \sim F_{10}$ 误差阈值 $10^{-2}$.

3) 成功运行次数(SR)[1]. 如果当前函数评价次数没有达到最大函数评价次数且最优解的适应值误差小于指定误差阈值的次数.

4) 收敛曲线图[1]. 算法在指定运行次数中最优解的平均适应值误差曲线图.

在本文中，收敛曲线图中的平均误差均是变换后的平均适应值误差，其变换函数是 lg.

### 4.1 与 CS 算法比较

表 1 列出 Co2CS 算法与 CS 算法优化 10 个函数适应值平均误差和标准差，其格式为“平均误差±标准差”. 其中，维数 $D=30$，种群规模 $N=D$，最大函数评价次数 $FES=10000D$，独立运行 50 次，发现概率 $p_a=0.25$，Co2CS 子群体个数为 5. “≈”表示 CS 算法的平均误差与 Co2CS 算法的平均误差在 0.05 水平下的双侧 $t$-检验是不显著的；“*”表示 CS 算法的平均误差与 Co2CS 算法的平均误差在 0.05 水平下的双侧 $t$-检验是显著的，解的质量较 Co2CS 算法差；“*”表示 CS 算法的平均误差与 Co2CS 算法的平均误差在 0.05 水平下的双侧 $t$-检验是显著的，解的质量较 Co2CS 算法好.

**表 1 CS 与 Co2CS 的平均误差($D=30$)**

Table 1 Mean errors of CS and Co2CS ($D=30$)

| | CS | Co2CS |
|---|---|---|
| $F_1$ | 1.12e-30±3.12e-30≈ | 4.10e-30±2.86e-29 |
| $F_2$ | 5.62e-03±6.01e-03* | 1.26e-09±2.11e-09 |
| $F_3$ | 1.97e+06±5.66e+05* | 6.63e+05±3.40e+05 |
| $F_4$ | 1.28e+03±7.09e+02* | 3.93e+02±1.15e+03 |
| $F_5$ | 2.98e+03±7.18e+02* | 7.64e+03±1.89e+03 |
| $F_6$ | 2.08e+01±2.00e+01* | 5.24e+00±3.22e+00 |
| $F_7$ | 2.08e-04±1.20e-03* | 1.56e-02±1.19e-02 |
| $F_8$ | 2.09e+01±4.77e-02* | 2.06e+01±2.06e-01 |
| $F_9$ | 2.67e+01±4.96e+00* | 9.68e+00±3.20e+00 |
| $F_{10}$ | 1.68e+02±3.48e+01≈ | 1.68e+02±5.03e+01 |

对于单峰函数而言，Co2CS 算法在 $F_1$ 和 $F_5$ 的平均误差逊色于 CS 算法，但在 $F_2$、$F_3$ 和 $F_4$ 函数上的平均误差明显优于 CS 算法；对于多峰函数而言，在 $F_7$ 和 $F_{10}$ 函数上，Co2CS 算法的平均误差劣于 CS 算法，但在 $F_6$、$F_8$ 和 $F_9$ 函数上的平均误差明显优于 CS 算法. 对于具有变换特点的函数而言，Co2CS 算法的平均误差明显优于 CS 算法；对于变换且旋转的函数而言，除 $F_7$ 函数，Co2CS 算法的平均误差明显优于 CS 算法.

表 2 给出两算法收敛于指定误差阈值的函数、所需平均函数评价次数、标准差及成功次数. 从表中可知，Co2CS 算法收敛于指定误差阈值的函数个数优于 CS 算法. 在 $F_1$ 函数上，CS 算法的成功次数与 Co2CS 算法相同，但借助于平均函数评价次数，Co2CS 算法明显优于 CS 算法，体现出较快的收敛速度，也反映在图 1 中. 在 $F_2$ 和 $F_6$ 函数上，CS 算法没有收敛于指定误差阈值，而 Co2CS 算法收敛于指定误差阈值，特别在 $F_2$ 函数上，其收敛是稳定的. 其中，$F_6$ 的收敛过程见图 2，说明 Co2CS 算法具有较快的收敛速度，且在后期具有较好的求解能力. 在 $F_7$ 函数上，借助于成功次数，Co2CS 算法收敛的稳定性弱于 CS 算法，但 Co2CS 算法收敛于指定误差阈值所需的平均函数评价次数优于 CS 算法，体现出在迭代早期具有较快的收敛速度. 图 3 是 $F_7$ 函数的收敛过程，验证上述结论.

**表 2 CS 与 Co2CS 的平均函数评价次数($D=30$)**

Table 2 Mean NFEs of CS and Co2CS ($D=30$)

| | CS | Co2CS |
|---|---|---|
| $F_1$ | 91339.9±2410.7(50) | 49787.0±1189.9(50) |
| $F_2$ | — | 221110.2±15220.5(50) |
| $F_6$ | — | 192094.1±32284.4(9) |
| $F_7$ | 148660.8±26954.6(50) | 77478.9±13129.0(21) |

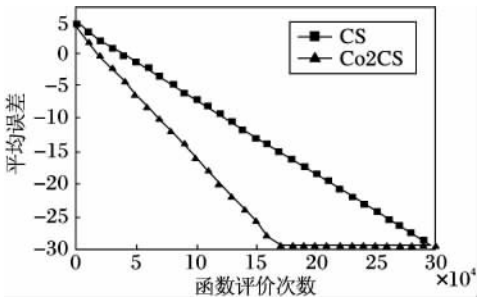


**图 1 CS 和 Co2CS 在 $F_1$ 函数上的收敛过程**

Fig. 1 Convergence process of CS and Co2CS on $F_1$

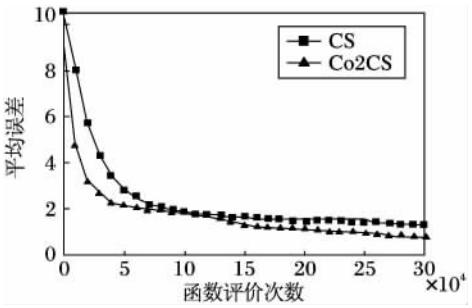


**图 2 CS 和 Co2CS 在 $F_6$ 函数上的收敛过程**

Fig. 2 Convergence process of CS and Co2CS on $F_6$

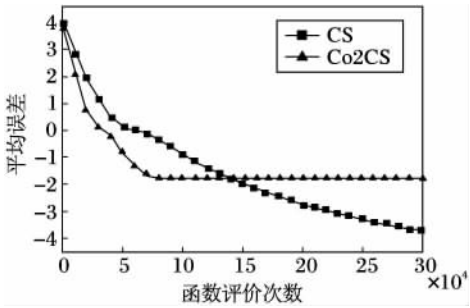


**图 3**　CS 和 Co2CS 在 $F_7$ 函数上的收敛过程

Fig. 3　Convergence process of CS and Co2CS on $F_7$

在 $F_5$ 函数上，虽然 Co2CS 算法的平均误差劣于 CS 算法，但图 4 的收敛过程说明 Co2CS 算法在迭代前期具有较快的收敛速度.

Co2CS 算法在 $F_{10}$ 函数上的平均误差没有明显优于 CS 算法，然而，图 5 说明 Co2CS 算法具有较快的收敛速度.

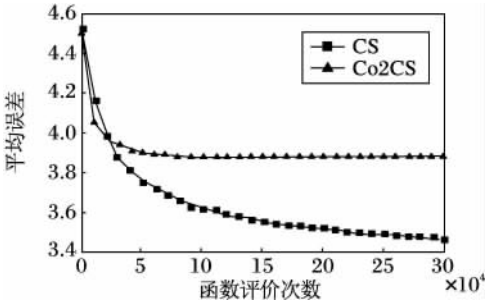


**图 4**　CS 和 Co2CS 在 $F_5$ 函数上的收敛过程

Fig. 4　Convergence process of CS and Co2CS on $F_5$

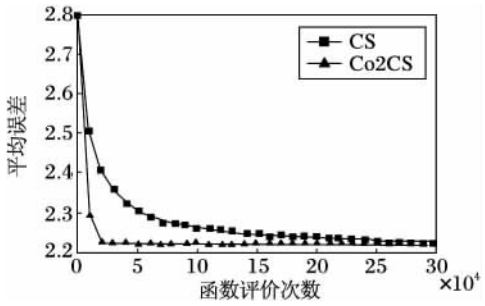


**图 5**　CS 和 Co2CS 在 $F_{10}$ 函数上的收敛过程

Fig. 5　Convergence process of CS and Co2CS on $F_{10}$

图 6 ~ 图 9 分别是 $F_3$、$F_4$、$F_8$ 和 $F_9$ 的收敛过程，皆说明 Co2CS 算法具有较快的收敛速度. 后期的收敛情况进一步证明表 1 中 Co2CS 算法具有更优的平均误差.

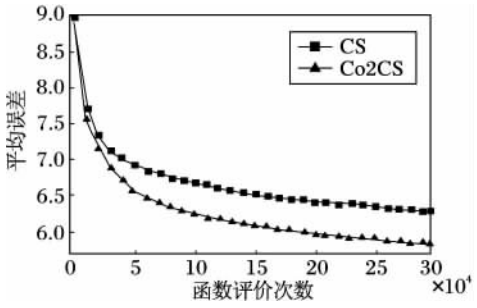


**图 6**　CS 和 Co2CS 在 $F_3$ 函数上的收敛过程

Fig. 6　Convergence process of CS and Co2CS on $F_3$

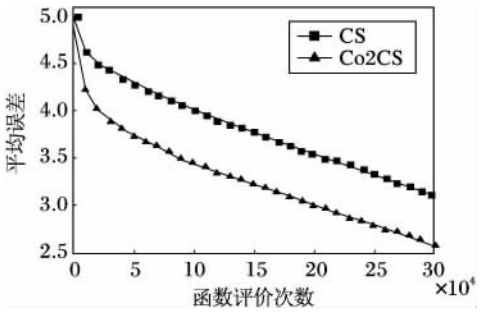


**图 7**　CS 和 Co2CS 在 $F_4$ 函数上的收敛过程

Fig. 7　Convergence process of CS and Co2CS on $F_4$

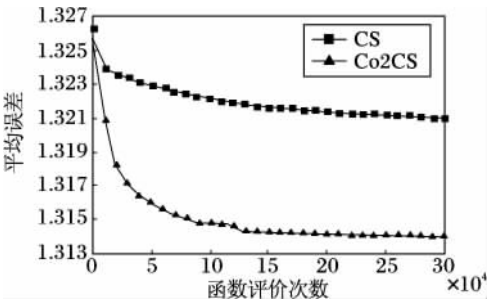


**图 8**　CS 和 Co2CS 在 $F_8$ 函数上的收敛过程

Fig. 8　Convergence process of CS and Co2CS on $F_8$

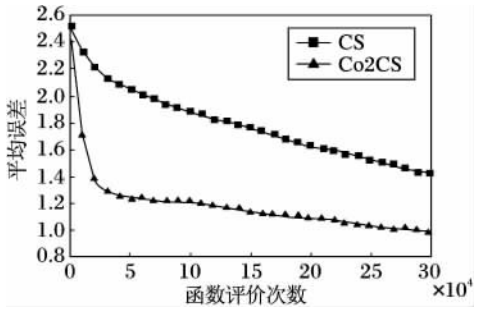


**图 9**　CS 和 Co2CS 在 $F_9$ 函数上的收敛过程

Fig. 9　Convergence process of CS and Co2CS on $F_9$

综合平均误差、成功次数、平均函数评价次数及收敛过程的比较结果，Co2CS 算法在 $F_1$ 和 $F_9$ 函数上

获得较好的性能，其原因是两函数变量间独立，使得 Co2CS 算法充分利用子群体的独立进化引导整个群体进行，加快收敛速度，并获得较优的平均误差. 然而，在其它变量间相关联的函数上，Co2CS 算法的性能优于 CS 算法，其原因是改进算法采用子群体数目较少，在一定程度上未破坏变量间的相关性，使得算法能够利用子群体的进化并协作引导整个种群进化. 针对 $F_5$ 和 $F_7$ 函数，Co2CS 算法未能取得较好的性能，其原因是这两个函数最优解分别在边界和边界之外，且函数的变量间是相关联的，使得各子群体较难独立进化，并引导 Co2CS 算法获得较好的性能.

### 4.2 子群体数目的影响分析

Co2CS 算法采用多个子群体的独立进化并合作完成整个种群的进化，为分析子群体数目对算法的影响，在 30 维的基础上，分别比较子群体数目为 $K$ 取 2、3、5、6、10、15 及 30 的实验结果，相应算法用 $Co2CS_K$ 表示，如表 3 所示.

表 3 说明在函数 $F_1$ 和 $F_9$ 上，随着子群体数目增加，Co2CS 算法的性能得到进一步改善，在最大子群体数目时收敛于全局最优. 究其原因是两函数的变量间是对立的，能够使 Co2CS 算法充分利用子群体的进化引导种群进化. 在 $F_2$、$F_3$、$F_4$、$F_5$、$F_6$ 和 $F_{10}$ 函数上，Co2CS 算法随着子群体数目增加，其性能进一步恶化. 此类函数的变量间是相关联的，子群体数目增加将进一步破坏其关联性，不利于函数的优化. 虽然 $F_7$、$F_8$ 函数的变量间不独立，Co2CS 算法却随着子群体增加，其性能分别稳定和得到进一步改善.

综合实验结果，在不同的函数上，不同的子群体数目，Co2CS 算法的性能各异. 然而，从表 3 中可知，当子群体数目 $K=5$ 或 $K=6$ 时，对变量独立的函数而言，Co2CS 算法能较好地利用子群体的进化与合作引导整个群体的进化；对变量相关联的函数而言，较少的子群体数目仍保留变量间的关联性，Co2CS 算法也能较好地利用子群体进化与合作引导整个群体的进化.

**表 3 不同子群体数目的平均误差（$D=30$）**

Table 3 Mean errors with different number of sub-swarms ($D=30$)

| | $Co2CS_2$ | $Co2CS_3$ | $Co2CS_5$ | $Co2CS_6$ | $Co2CS_{10}$ | $Co2CS_{15}$ | $Co2CS_{30}$ |
|---|---|---|---|---|---|---|---|
| $F_1$ | 1.68e-28± 2.04e-28 | 1.62e-29± 5.34e-29 | 4.10e-30± 2.86e-29 | 0.00e+00± 0.00e+00 | 0.00e+00± 0.00e+00 | 0.00e+00± 0.00e+00 | 0.00e+00± 0.00e+00 |
| $F_2$ | 1.02e-09± 3.65e-09 | 2.69e-07± 9.40e-07 | 1.26e-09± 2.11e-09 | 9.84e-08± 1.64e-07 | 2.28e-04± 2.30e-04 | 1.26e-01± 1.05e-01 | 4.44e+01± 2.51e+01 |
| $F_3$ | 3.96e+05± 2.15e+05 | 6.69e+05± 3.03e+05 | 6.63e+05± 3.40e+05 | 8.10e+05± 4.03e+05 | 1.02e+06± 5.15e+05 | 1.32e+06± 7.77e+05 | 2.18e+06± 9.02e+05 |
| $F_4$ | 7.59e+02± 2.07e+03 | 1.46e+03± 2.41e+03 | 3.93e+02± 1.15e+03 | 1.71e+02± 3.62e+02 | 1.17e+03± 2.22e+03 | 5.89e+03± 3.71e+03 | 1.37e+04± 6.66e+03 |
| $F_5$ | 5.05e+03± 1.58e+03 | 6.10e+03± 2.00e+03 | 7.64e+03± 1.89e+03 | 7.82e+03± 1.92e+03 | 1.08e+04± 2.99e+03 | 1.38e+04± 3.71e+03 | 2.08e+04± 4.69e+03 |
| $F_6$ | 1.53e+00± 1.67e+00 | 1.23e+01± 7.97e+00 | 5.24e+00± 3.22e+00 | 6.44e+00± 3.83e+00 | 6.99e+00± 6.66e+00 | 5.35e+00± 6.79e+00 | 5.06e+01± 4.36e+01 |
| $F_7$ | 1.96e-02± 1.56e-02 | 2.76e-02± 2.85e-02 | 1.56e-02± 1.19e-02 | 2.38e-02± 2.23e-02 | 2.27e-02± 1.81e-02 | 1.93e-02± 1.70e-02 | 1.57e-02± 1.26e-02 |
| $F_8$ | 2.05e+01± 2.10e-01 | 2.06e+01± 2.24e-01 | 2.06e+01± 2.06e-01 | 2.05e+01± 1.71e-01 | 2.02e+01± 9.52e-02 | 2.01e+01± 5.13e-02 | 2.01e+01± 5.96e-02 |
| $F_9$ | 6.64e+01± 1.56e+01 | 3.49e+01± 9.78e+00 | 9.68e+00± 3.20e+00 | 6.84e+00± 2.74e+00 | 3.94e-01± 6.91e-01 | 1.36e-09± 9.60e-09 | 0.00e+00± 0.00e+00 |
| $F_{10}$ | 1.82e+02± 5.83e+01 | 1.92e+02± 4.66e+01 | 1.68e+02± 5.03e+01 | 1.80e+02± 5.48e+01 | 1.70e+02± 4.98e+01 | 1.84e+02± 5.10e+01 | 3.49e+02± 1.19e+02 |

### 4.3 与其它改进 CS 算法比较

为分析 Co2CS 算法与其它改进 CS 算法的性能差异，表 4 列出 Co2CS 算法与 ICS 算法[20]、CSPSO 算法[22] 和 OLCS 算法[17,23] 在 $D=30$ 维空间上的性能比较结果.

分析表 4 可知，针对单峰函数，各算法的性能各异. 在 $F_1$ 函数上，Co2CS 算法优于 CSPSO 算法和 OLCS 算法；ICS 算法的平均误差值优于 Co2CS 算法，但在 0.05 水平下不显著. 在 $F_2$ 函数上，CSPSO 算法性能最优，其次 Co2CS 算法. 而在 $F_3$ 函数上，Co2CS 算法的性能仅次于 ICS 算法. Co2CS 算法在 $F_5$ 函数上的性能最弱，但在 $F_4$ 函数上的性能最优. 针对多峰函数而言，在 $F_6$、$F_8$ 和 $F_9$ 函数上，Co2CS 算法的性能明显优于其它 CS 算法. 在 $F_7$ 函数上，

Co2CS 算法的性能与 OLCS 算法性能相似,但弱于其它 CS 算法. 在 $F_{10}$函数上,Co2CS 算法优于 CSPSO 算法,但逊于 ICS 算法. 总体上,根据表 4 中针对平均误差检验统计结果,Co2CS 算法优于其它改进的 CS 算法.

**表 4　Co2CS 与其它 CS 的平均误差($D=30$)**

Table 4　Mean errors of Co2CS and others ($D=30$)

| | ICS | CSPSO | OLCS | Co2CS |
|---|---|---|---|---|
| $F_1$ | 0.00e+00± 0.00e+00≈ | 2.65e-28± 2.67e-28* | 2.41e-26± 6.21e-26* | 4.10e-30± 2.86e-29 |
| $F_2$ | 1.67e-03± 2.87e-03* | 1.41e-11± 2.68e-10＊ | 5.70e-02± 4.79e-02* | 1.26e-09± 2.11e-09 |
| $F_3$ | 3.85e+05± 1.78e+05＊ | 8.01e+05± 6.49e+05≈ | 2.57e+06± 7.13e+05* | 6.63e+05± 3.40e+05 |
| $F_4$ | 4.81e+02± 3.92e+02≈ | 5.93e+01± 4.39e+01＊ | 2.37e+03± 1.23e+03* | 3.93e+02± 1.15e+03 |
| $F_5$ | 1.63e+03± 5.45e+02＊ | 3.25e+03± 9.33e+02＊ | 2.44e+03± 7.31e+02＊ | 7.64e+03± 1.89e+03 |
| $F_6$ | 1.26e+01± 9.73e+00* | 6.56e+00± 1.78e+01≈ | 2.45e+01± 1.99e+01* | 5.24e+00± 3.22e+00 |
| $F_7$ | 2.09e-03± 2.49e-03＊ | 2.22e-02± 1.20e-15* | 4.72e-04± 1.13e-03＊ | 1.56e-02± 1.19e-02 |
| $F_8$ | 2.09e+01± 2.06e-02* | 2.09e+01± 5.62e-02* | 2.09e+01± 5.31e-02* | 2.06e+01± 2.06e-01 |
| $F_9$ | 1.61e+01± 4.15e+00* | 1.57e+02± 2.21e+01* | 3.54e+01± 6.64e+00* | 9.68e+00± 3.20e+00 |
| $F_{10}$ | 7.65e+01± 1.05e+01＊ | 2.52e+02± 5.86e+01* | 1.54e+02± 3.74e+01≈ | 1.68e+02± 5.03e+01 |
| * | 4 | 4 | 7 | |
| ≈ | 2 | 3 | 1 | |
| ＊ | 4 | 3 | 2 | |

**表 5　Co2CS 与其它 CS 的平均函数评价次数($D=30$)**

Table 5　Mean NFEs of Co2CS and others ($D=30$)

| | ICS | CSPSO | OLCS | Co2CS |
|---|---|---|---|---|
| $F_1$ | 51914± 923(50) | 95845± 2940(50) | 195993± 2554(50) | 49787± 1190(50) |
| $F_2$ | — | 206075± 13386(50) | — | 221110± 15221(50) |
| $F_6$ | — | 270272± 20778(25) | — | 192094± 32284(9) |
| $F_7$ | 193959± 48311(50) | 2244± 15645 (50) | 110102± 20000(50) | 77479± 13129 (21) |

表 5 给出 Co2CS 算法与其它 CS 算法收敛于指定误差阈值的函数、所需平均函数评价次数、标准差及成功次数. 借助于收敛指定阈值的函数数目,Co2CS 算法和 CSPSO 算法最优. 在 $F_1$ 函数上,各算法的成功次数相同,稳定收敛于指定误差阈值,但借助于平均函数评价次数,Co2CS 算法最优. 在 $F_2$ 函数上, Co2CS 算法稳定收敛于指定误差阈值,但收敛速度弱于 CSPSO 算法. 在 $F_6$ 函数上, 借助于成功次数,Co2CS 算法稍劣于 CSPSO 算法,然而借助于平均函数评价次数,前者优于后者. 针对 $F_7$ 函数,Co2CS 算法收敛于指定误差阈值的次数最少,但平均评价次数却优于 ICS 算法和 OLCS 算法.

### 4.4　与其它 CC 框架的算法性能比较

表 6 列出 Co2CS 算法与其它基于 CC 框架算法的性能比较结果,其中,测试函数是文献 [33] 中的 5 个函数. 为能正确比较,各函数的维数 $D=30$,Co2CS 算法的种群大小 $N=20$,最大函数评价次数 $FES=200000$,子群体数目 $K=6$,独立运行 50 次. CCGA 算法、CPSO-S 算法和 CPSO-H 算法的实验结果引用文献 [33] .

从表 6 可看出,在 $f_3$函数上,Co2CS 算法的平均误差最差,在 $f_1$ 函数上,Co2CS 算法的平均误差一般,但在 $f_0$、$f_2$和 $f_4$函数上,Co2CS 算法的平均误差明显最优. 表中的“* ”、“≈”和“＊”统计结果说明 Co2CS 算法总体上最优.

**表 6　Co2CS 与其它 CC 框架算法的平均误差**

Table 6　Mean errors of Co2CS and CC based algorithms

| | CCGA | CPSO-S | CPSO-H | Co2CS |
|---|---|---|---|---|
| $f_0$ | 3.80e+00± 1.93e-01* | 1.59e+00± 5.03e-01* | 4.21e-01± 3.21e-01* | 1.75e-01± 1.04e-01 |
| $f_1$ | 1.38e+02± 9.20e+01* | 1.20e-04± 8.99e-05≈ | 1.40e-29± 1.15e-29＊ | 1.38e-04± 9.07e-04 |
| $f_2$ | 9.51e-02± 3.39e-02* | 5.42e-05± 1.66e-05* | 2.73e-12± 2.03e-12* | 3.24e-14± 7.13e-15 |
| $f_3$ | 1.22e+00± 2.35e-01＊ | 1.46e-01± 1.03e-01＊ | 7.78e-01± 1.87e-01＊ | 1.11e+01± 3.35e+00 |
| $f_4$ | 2.20e-01± 6.57e-02* | 8.59e-02± 1.68e-02* | 5.24e-02± 1.19e-02≈ | 6.35e-02± 5.20e-02 |
| * | 4 | 3 | 2 | |
| ≈ | 0 | 1 | 1 | |
| ＊ | 1 | 1 | 2 | |

### 4.5　算法复杂性分析

在进化算法中,大部分的计算量都集中在目标函数评估过程中,其复杂性往往体现目标函数的评价次数. 记算法的最大迭代次数为 $M$,从算法 1 和算法 2 可知,CS 算法和 Co2CS 算法的复杂度分别为 $O(M\times N)$ 和 $O(M\times K\times N)$ . 在固定的迭代次数条件下,Co2CS 算法函数评价次数将与子群体的数目成正比. 而在相同函数评价次数条件下,Co2CS 算法与 CS 算法有相同的复杂度 $O(FES)$ . 但由于 Co2CS 算法将问题分解成子问题,需要额外存储空间,同时在空间操作及问题分解与合并上将消耗一定时间,导致 Co2CS 算法的时间复杂度和空间复杂度都较 CS 高.

为定量评价算法的复杂性，本文采用式(3)度量算法的复杂性[1]：

$$C=\frac{T_2-T_1}{T_0}, \tag{3}$$

其中，$T_0$表示执行特定的测试程序[1]所需时间；$T_1$表示在200 000次函数评价条件下，算法优化$F_3$函数所需时间；$T_2$表示在200 000次函数评价条件下，算法累计5次优化$F_3$函数所需的平均时间.

表7给出CS算法和Co2CS算法在不同搜索空间上的计算复杂性，其中Co2CS算法的子群体数目为5. 从表7可知，Co2CS算法具有较高的计算复杂度，主要由于Co2CS算法将问题分解成子问题消耗部分时间. 然而，随着维数增加，这部分时间将逐步弱化，以致Co2CS算法的复杂度与CS算法的复杂度差距缩小.

表7 计算复杂性

Table 7 Computational complexity

| | CS | | | | Co2CS | | |
|---|---|---|---|---|---|---|---|
| | $T_0$ | $T_1$ | $T_2$ | $C$ | $T_1$ | $T_2$ | $C$ |
| $D=10$ | — | 5.27 | 15.98 | 2.92 | 8.82 | 25.90 | 4.66 |
| $D=30$ | 3.67 | 4.26 | 12.79 | 2.33 | 5.65 | 16.78 | 3.04 |
| $D=50$ | — | 4.65 | 13.90 | 2.52 | 5.02 | 14.89 | 2.69 |

图10给出在30维搜索空间上，Co2CS算法受不同子群体数目影响的复杂度情况. 图10说明Co2CS算法的复杂度随子群体数目的增加而提高. 然而，分析复杂度提高的幅度及表3中的性能可知，当子群体数目为5或6时，Co2CS算法的效率是最优的.

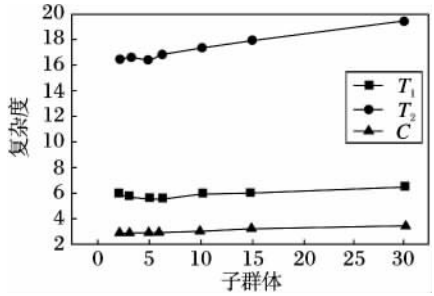


图10 不同子群体的复杂度

Fig. 10 Complexities with different sub-swarm sizes

## 5 结束语

本文提出一种基于合作协同进化的布谷鸟搜索算法，通过多个子群体的进化及合作引导整个群体的进化，以提高CS算法求解连续函数优化问题的性能. 通过选取若干不同属性的函数优化问题，将改进算法与CS算法进行对比实验，证实算法改进的有效性. 分析不同子群体数目对改进算法性能的影响，得出一些有意义的结论，旨在揭示子群体数目对改进算法性能影响的规律. 对比分析改进算法与其它改进的CS算法及其它基于CC框架的算法的性能，得出改进算法求解连续函数优化问题具有竞争力. 虽然实验结果说明改进算法具有较好收敛效果，但理论上收敛性分析工作需要今后进一步研究与探讨. 同时，论文从运行时间角度度量算法的复杂度，后续的工作将从理论上分析算法的复杂度. 另外，如何提高变量间高度关联函数的性能，值得进一步研究.

## 参 考 文 献

[1] Suganthan P N, Hansen N, Liang J J, *et al*. Problem Definitions and Evaluation Criteria for the CEC2005 Special Session on Real-Parameter Optimization. Technical Report, 2005005. Singapore, Singapore: Nanyang Technological University, 2005

[2] Qing A Y. Differential Evolution: Fundamentals and Applications in Electrical Engineering. Singapore, Singapore: John Wiley & Sons (Asia) Pte Ltd, 2009

[3] Holland J H. Adaptation in Natural and Artificial Systems. Ann Arbor, USA: University of Michigan Press, 1975

[4] Kennedy J, Eberhart R. Particle Swarm Optimization // Proc of the IEEE International Conference on Neural Networks. Perth, Australia 1995, IV: 1942-1948

[5] Eberhart R, Kennedy J. A New Optimizer Using Particle Swarm Theory // Proc of the 6th International Symposium on Micro Machine and Human Science. Nagoya, Japan, 1995: 39-43

[6] Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by a Colony of Cooperating Agents. IEEE Trans on Systems, Man and Cybernetics, 1996, 26(1): 29-41

[7] Storn R, Price K. Differenial Evolution: A Simple and Efficient Heuristic for Global Optimal over Continuous Spaces. Journal of Global Optimal, 1997, 11(4): 341-359

[8] Karaboga D. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report, TR06. Kayseri, Turkey: Erciyes University, 2005

[9] Simon D. Biogeography-Based Optimization. IEEE Trans on Evolutionary Computation, 2008, 12(6): 702-713

[10] Yang Xinshe, Deb S. Cuckoo Search via Lévy Flights // Proc of the World Congress on Nature & Biologically Inspired Computing. Coimbatore, India, 2009: 210-214

[11] Yang Xinshe, Deb S. Engineering Optimization by Cuckoo Search. International Journal of Mathematical Modeling and Numerical Optimization, 2010, 1(4): 330-343

[12] Civicioglu P, Besdok E. A Conceptual Comparison of the Cuckoo Search, Particle Swarm Optimization, Differential Evolution and

Artificial Bee Colony Algorithms. Artificial Intelligence Review, 2013, 39(4): 315-346

[13] Yang Xinshe. Cuckoo Search for Inverse Problems and Simulated-Driven Shape Optimization. Journal of Computational Methods in Sciences and Engineering, 2011, 12(1/2): 129-137

[14] Layeb A, Boussalia S R. A Novel Quantum Inspired Cuckoo Search Algorithm for Bin Packing Problem. International Journal of Information Technology and Computer Science, 2012, 4(5): 58-67

[15] Yang Xinshe, Deb S. Multi-Objective Cuckoo Search for Design Optimization. Computers & Operations Research, 2013, 40(6): 1616-1624

[16] Srivastava P R, Khandelwal R, Khandelwal S, *et al*. Automatic Test Data Generation Using Cuckoo Search and Tabu Search Algorithm. Journal of Intelligent Systems, 2012, 21(2): 195-224

[17] Li Xiangtao, Yin Minghao. Parameter Estimation for Chaotic Systems Using the Cuckoo Search Algorithm with an Orthogonal Learning Method. Chinese Physics B, 2012, 21(5): 113-118

[18] Walton S, Hassan O, Morgan K, *et al*. Modified Cuckoo Search: A New Gradient free Optimization Algorithm. Chaos, Solitons & Fractals, 2011, 44(9): 710-718

[19] Tuba M, Subotic M, Stanarevic N. Modified Cuckoo Search Algorithm for Unconstrained Optimization Problems // Proc of the 5th European Conference on Computing. Athens, Greece, 2011: 263-268

[20] Valian E, Mohanna S, Tavakoli S. Improved Cuckoo Search Algorithm for Global Optimization. International Journal of Communications and Information Technology, 2011, 1(1): 31-44

[21] Ghodrati A, Lotfi S. A Hybrid CS/PSO Algorithm for Global Optimization // Proc of the 4th Asian Conference on Intelligence information and Database Systems. Kaohsiung, China, 2012: 89-98

[22] Wang Fan, He Xingshi, Luo Ligui, *et al*. Hybrid Optimization Algorithm of PSO and Cuckoo Search // Proc of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce. Zhengzhou, China, 2011: 1172-1175

[23] Li Xiangtao, Wang Jianan, Yin Minghao. Enhancing the Performance of Cuckoo Search Algorithm Using Orthogonal Learning Method. Neural Computing and Applications, 2013. DOI: 10.1007/s00521-013-1354-6

[24] Wolpert D H, Macready W G. No Free Lunch Theorems for Optimization. IEEE Trans on Evolutionary Computation, 1997, 1(1): 67-82

[25] Potter M A, de Jong K A. A Cooperative Coevolutionary Approach to Function Optimization // Proc of the 3rd International Conference on Parallel Problem Solving from Nature. Jerusalem, Israel, 1994: 249-257

[26] Amaya J E, Cotta C, Fernndez L A J. A Memetic Cooperative Optimization Schema and Its Application to the Tool Switching Problem // Proc of the 11th International Conference on Parallel Problem Solving from Nature. Krakow, Poland, 2010: 445-454

[27] Bongard J, Lipson H. Active Coevolutionary Learning of Deterministic Finite Automata. Journal of Machine Learning Research, 2005, 6(10): 1651-1678

[28] Potter M A, Couldrey C. A Cooperative Coevolutionary Approach to Partitional Clustering // Proc of the 11th International Conference on Parallel Problem Solving from Nature. Krakow, Poland, 2010: 374-383

[29] Tonda A, Lutton E, Squillero G. A Benchmark for Cooperative Coevolution. Memetic Computing, 2012, 4(4): 263-277

[30] Goh C K, Lim D, Ma L, *et al*. A Surrogate-Assisted Memetic Co-Evolutionary Algorithm for Expensive Constrained Optimization Problems // Proc of the IEEE Congress on Evolutionary Computation. New Orleans, USA, 2011: 744-749

[31] Dong Hongbin, Yang Baodi, Liu Jiayuan, *et al*. A Co-Evolutionary Algorithm for Clustering. Pattern Recognition and Artificial Intelligence, 2012, 25(4): 676-683 (in Chinese)
(董红斌,杨宝迪,刘佳媛,等. 协同演化算法在聚类中的应用. 模式识别与人工智能, 2012, 25(4): 676-683)

[32] Jansen T, Wiegand R P. The Cooperative Coevolutionary (1+1) EA. Evolutionary Computation, 2004, 12(4): 405-434

[33] Van den Bergh F, Engelbrecht A P. A Cooperative Approach to Particle Swarm Optimization. IEEE Trans on Evolutionary Computation, 2004, 8(3): 225-239

[34] Shi Yanjun, Teng Hongfei, Li Ziqiang. Cooperative Co-Evolutionary Differential Evolution for Function Optimization // Proc of the 1st International Conference on Advances in Natural Computation. Changsha, China, 2005: 1080-1088

[35] Yang Zhenyu, Tang Ke, Yao Xin. Large Scale Evolutionary Optimization Using Cooperative Coevolution. Information Sciences, 2008, 178(15): 2985-2999