创造性驱动优化算法

邹 儒 冯 翔
(华东理工大学信息科学与工程学院 上海 200237)

摘 要 模拟人类的创造性思维来求解问题一直是人工智能研究的热点和难点之一。基于当前的创造性思维研究理论,同时借鉴现有自然启发算法的建模过程,提出了一种新的智能优化算法——创造性驱动优化算法(Creativity Driven Optimization Algorithm,CDOA)。首先,构建出创造性驱动优化模型,并且为其5个子模型设计出具体的操作算子。而后,根据各子模型之间的联系,给出创造性驱动优化算法的执行步骤。为验证创造性驱动优化算法的有效性,使用8个CEC-2013实参数优化基准函数对CDOA进行了测试,并与当前最先进的3个同类算法进行对比。实验结果显示,CDOA在复杂函数上具有较好的寻优能力。最后,对CDOA进行的计算复杂度实验及分析表明,在相同实验条件下,与其他3个对比算法相比,CDOA具有更快的执行速度。

关键词 创造性驱动优化算法,创造性思维,自然启发算法,函数优化
中图法分类号 TP183 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.053

Creativity Driven Optimization Algorithm

ZOU Ru FENG Xiang
(School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)

Abstract Solving problem by mimicking human's creative thinking process has been one of the hotspots in artificial intelligence. According to the current researches about creative thinking, and also learning from the present nature-inspired algorithms, a novel intelligent algorithm named creativity driven optimization algorithm(CDOA) was proposed. First, the creativity driven optimization model was constructed and special operators were designed for its five sub-models. Then, the execution steps of CDOA were given out. In order to test CDOA's effectiveness, eight CEC-2013 real-parameter benchmark functions were used. The optimization results of CDOA were compared with three state-of-the-art algorithms. The result shows that CDOA has an appealing optimization performance, especially on the complex composition functions. At last, an experiment was carried out to analyze the computation complexity of CDOA. The results indicate that CDOA has lower computation complexity than the other three comparison algorithms.

Keywords Creativity driven optimization algorithm, Creative thinking, Nature-inspired algorithm, Function optimization

1 引言

自然计算是计算机科学和自然科学之间的桥梁。"道法自然"早已成为计算机科学研究者的常用手段[1]。

近年来,自然启发算法被广泛应用于求解大规模复杂问题。由于其具有一定的自适应性、自学习能力和内在并行性,能够有效解决传统优化算法难以解决的复杂问题,在过去的半个多世纪里,该类算法得到了广泛的关注。提出这些算法的灵感通常来自于不同的自然系统。例如,进化算法是受到进化理论的启示被提出的,群体智能算法是基于社会性生物的多种行为而被提出的,人工免疫系统是通过模仿生物免疫系统而被构建的。最近几年,还有一些新的自然启发算法不断被提出。例如菌群算法[2]、种群动力学优化算法[3]、蚊子群追踪算法[4]、根生长算法[5]等。这些自然启发算法都有各自的特征,并且在求解复杂问题上都取得了一定的效果。

然而,现有的这些自然启发算法仍存在着一些局限性。由于其研究对象并不具有高度的智能性,现有的大多数自然启发算法不能满足未来发展对于智能的需求。此外,在当前的作用机制下,现有的基于群体的自然启发算法中存在着过多的个体交互行为,这将影响到算法的并行性,尤其是在一些交互代价较高的现实应用中。该局限性的根源也是算法中个体的智能性较低。

人类作为自然界中最智能的生物,具有强大的问题求解能力。创造性思维在其中起着重要的作用。在一个创造性思维过程中,必定存在着众多有效机制值得我们学习。选择创造性思维作为研究对象,则上述两个问题都能得到解决。

国内外关于创造性思维的研究由来已久。各种从不同领域对创造性思维进行研究的成果纷纷被提出。这些研究大致

到稿日期:2014-05-15 返修日期:2014-07-29 本文受国家自然科学基金(60905043,61073107,61173048),上海市教育委员会科研创新项目,中央高校基本科研业务费资助。

邹 儒(1991-),男,硕士生,主要研究方向为分布并行计算、人工智能;冯 翔(1977-),女,博士,教授,博士生导师,CCF会员,主要研究方向为分布并行计算、人工智能、网络通信,E-mail:xfeng@ecust.edu.cn(通信作者)。

可以被划分为两类[6]。一类是从个体层面对创造性思维进行的研究。在这一类研究中，大多数都将发散思维和收敛思维视作创造性思维过程中的两种主要思维模式[7-9]。还有一些学者对灵感思维进行了研究[10]。此外，文献[11,12]中的研究表明"经验"对创造性想法的产生也有着重要的影响。另一类研究是从社会层面对创造性思维进行分析。这些研究主要指出，知识共享和社会交互在促进创造性想法的产生方面扮演着关键角色[13,14]。上述的这些研究为我们构建基于创造性思维的算法模型提供了理论基础。

迄今为止，尚没有基于人类的创造性思维过程而提出的自然启发算法。本文基于现有的关于创造性思维的研究，并参照已有自然启发算法的构建过程，提出一种新的自然启发算法——创造性驱动优化算法(Creativity Driven Optimization Algorithm,CDOA)。CDOA 与现有的其它自然启发算法有着很多共同特征，即都具有一定的自适应性、自学习能力和内在的并行性。这使得 CDOA 能够被用于求解许多传统优化算法难以求解的问题，即一些高维、非线性、大规模的复杂问题。此外，与其他同类算法相比，CDOA 中的个体更加智能和独立，个体之间的交互更少。这能够在一定程度上提升 CDOA 的并行性，尤其是当个体之间的交流代价较高时。同时，由于 CDOA 可以很容易地被引入人类的智能行为，它的智能扩展性也会更好。

2 创造性驱动优化模型

本文构造的创造性驱动优化模型包括发散思维模型、收敛思维模型、灵感思维模型、集体思维模型和经验模型 5 个子模型。各模型的描述和具体设计见 2.1－2.5 节。由于最小化问题和最大化问题可以相互转化，本文的具体操作算子都是针对最小化问题进行设计的。

2.1 发散思维模型

当前大多数的研究都表明，发散思维在创造性思维过程中起着重要作用。在这个阶段，人们常常会在当前想法的基础上，沿着不同的思维方向，产生多个不同的想法。人类在发散思维阶段总是尝试产生尽可能多的想法。发散思维强调对未知信息空间进行探索，以促进新信息的产生。

我们使用 $\boldsymbol{I}(x_1,x_2,\cdots x_n)$ 表示思考者当前的想法，n 是问题的维数。在发散思维过程中，思考者将以当前想法 $\boldsymbol{I}$ 为中心，向周围随机产生多个"想法更新矢量" $\Delta\boldsymbol{I}$，这些"想法更新矢量"与当前想法相结合形成多个新想法。本文将每个思考者在一次发散思维过程中产生的"想法更新矢量"的数目设置为一个定值，并使用 $DNum$ 来表示。因此，思考者可以根据式(1)得到多个新想法。

$$\boldsymbol{I}_k^{new}=\boldsymbol{I}+\Delta\boldsymbol{I}_k,k=1,2,\cdots,DNum \tag{1}$$

"想法更新矢量" $\Delta\boldsymbol{I}[v_1,v_2,\cdots,v_n]$ 为一个 n 维变量，其中，$v_1,v_2,\cdots,v_n$ 是均值为 0、方差为 σ^2 的高斯随机变量，我们称这个方差为"创造性方差"。"创造性方差"的值较大时，对应形成的新想法就与原想法差别较大，可以进行远端搜索，有利于跳出局部极值点；"创造性方差"的值较小时，对应的新想法与原想法的差别相对较小，可以在原想法附近进行搜索，有利于提高解的精度。为了使算法更具智能性，我们为每个思考者设定自适应的创造性方差，并使用 $\sigma_{i,k}^2$ 来表示第 i 个思考者的第 k 个创造性方差，其中 $k\in\{1,2,3\}$。$\sigma_{\max}^2$ 和 $\sigma_{\min}^2$ 分别表示创造性方差的上界和下界。初始时，$\sigma_{i,1}^2=\sigma_{\max}^2$，$\sigma_{i,2}^2=\sigma_{\min}^2$，$\sigma_{i,3}^2$ 一直取 $\sigma_{i,1}^2$ 和 $\sigma_{i,2}^2$ 的平均值。在算法执行过程中，我们使用一个预先设定的缩放因子 $Sfactor\in(0,1)$ 来控制"创造性方差"的改变。如果一个思考者在一次发散思维过后能产生一个更好的想法，则它的 3 个创造性方差都会增大($\sigma_k^2\leftarrow\sigma_k^2/Sfactor$)；否则，其"创造性方差"保持不变。如果一个思考者的创造性方差连续 $INum$ 次保持不变，则其所有的创造性方差都减小($\sigma_k^2\leftarrow\sigma_k^2 Sfactor$)。

2.2 收敛思维模型

在收敛思维阶段，需要根据优化目标对发散思维过程中产生的想法进行评估，选取最具创造性的想法作为收敛思维阶段得到的想法。在本文中，$f(\boldsymbol{I})$ 被用来表示想法 $\boldsymbol{I}$ 的适应度值。对于最小化问题来说，适应度值越小，表明对应的想法越具有创造性。收敛思维的数学模型则可以被描述为式(2)和式(3)。其中，$\boldsymbol{I}_i$ 为发散思维过程产生的想法，$\boldsymbol{I}_t$ 为思考者当前阶段的想法，$\boldsymbol{I}_{t+1}$ 为下一阶段的想法。

$$k=\underset{i\in\{1,2,\cdots,DNum\}}{\arg\min}(f(\boldsymbol{I}_i)) \tag{2}$$

$$\boldsymbol{I}_{t+1}=\begin{cases}\boldsymbol{I}_k, & \text{if } f(\boldsymbol{I}_k)\leqslant f(\boldsymbol{I}_t)\\ \boldsymbol{I}_t, & \text{otherwise}\end{cases} \tag{3}$$

2.3 灵感思维模型

灵感思维是一种非逻辑思维方式，它是创造性思维过程中很重要的一部分，但是又有着很强的不确定性。本文将灵感思维定义为这样一个过程：当发散思维连续 $INum$ 次未产生更优想法时，灵感思维被触发。思考者将以 $10\sigma_{\max}^2$ 的方差进行发散思维搜索，并将获得的最好的想法作为灵感思维的想法。之后，按照贪婪策略与当前想法进行比较和替换。

2.4 集体思维模型

在当前社会，科技发展迅速，社会分工越来越细，由于个人的知识和能力十分有限，单个人往往难以解决大规模复杂问题。这时，就需要多个思考者相互协作，在单独思考的基础上进行集体学习讨论，即集体思维。在集体思维中，由于每个思考者的知识范围不同，通过知识共享和相互学习往往会促进创造性想法的产生。本节将为集体思维设计具体的操作。

假设参与集体思维的思考者的数目为 NT。首先，为集体思维过程中的思考者定义一个影响力矩阵 $\boldsymbol{A}$，$\boldsymbol{A}$ 是一个 $NT\times NT$ 维的矩阵。a_{ij} 为矩阵 $\boldsymbol{A}$ 的第 i 行第 j 列的元素，表示第 i 个思考者当前的想法对第 j 个思考者产生的影响。可以这样认为，某个思考者的当前想法越好，他对其他思考者产生的影响就越大。我们使用式(4)和(5)来计算 a_{ij}。

$$M_{ij}=\begin{cases}0, & i=j\\ (f(\boldsymbol{I}_j)-f(\boldsymbol{I}_j))/|f(\boldsymbol{I}_j)|, & i\neq j\end{cases} \tag{4}$$

$$a_{ij}=M_{ij}/\sum_{i=1}^{NT}|M_{ij}| \tag{5}$$

M_{ij} 为一个中间变量，由上述公式可以看出，a_{ij} 可能为正值，也可能为负值。$a_{ij}>0$ 表示思考者 i 当前的想法优于思考者 j 当前的想法，对思考者 j 有着正面的影响；$a_{ij}<0$ 则表明思考者 i 当前的想法不如思考者 j，思考者 j 会倾向于参考思考者 i 的想法的反方向。

计算出影响力矩阵后，我们的集体思维将通过以下步骤来促进群体中的思考者相互交流学习，产生更好的想法。

(1)选择学习对象

在现实中,一个思考者可能会向很多个思考者学习,我们定义这里的学习既包括模仿优秀的想法,也包括反思较差的想法。我们假定,对于当前想法较好的思考者,其对于学习对象的选择比较随机。而对于当前想法较差的思考者,其更倾向于向当前想法较好的思考者学习,或者更倾向于反思当前较差的想法。因此,我们将给出两种选择方案。在此之前,需要对当前各思考者的想法做一个简单的评估,本文使用 CV_i 来衡量思考者 i 当前想法的好坏程度,其计算方法如式(6)所示。CV_i 越小,表明当前的想法越好。

$$CV_i = f(\boldsymbol{I}_i)/(\sum_{i=1}^{NT} f(\boldsymbol{I}_i) - \min_{k\in\{1,\cdots,NT\}}(f(\boldsymbol{I}_k))) \tag{6}$$

给出的两种选择方案:方案一,按照轮盘赌策略随机选择3个学习对象。其中,思考者 j 选取思考者 i 作为学习或反思对象的概率 $P(\boldsymbol{I}_{ij})$ 的计算公式如式(7)所示。方案二,随机选择两个学习对象。

$$P(\boldsymbol{I}_{ij}) = |a_{ij}| \tag{7}$$

对于思考者来说,如果其在参与集体思维时 CV_i 大于(0,1)范围内产生的一个均匀分布随机数,则其选择方案一;否则,选择方案二。

(2)产生新想法

对于第一种选择方案,假设思考者在选择学习对象阶段选取的3个想法分别为 I_a,I_b 和 I_c,其中 $a,b,c\in\{1,2,\cdots,ThinkerNum\}$,并且都不等于 i,则利用式(8)产生一个组合想法 $\boldsymbol{V}_i(v_{i1},v_{i2},\cdots,v_{i,D})$。

$$\boldsymbol{V}_i = \begin{cases} \boldsymbol{I}_a + \alpha_{a,i} * (\boldsymbol{I}_a - \boldsymbol{I}_i) + (\alpha_{b,i} - \alpha_{c,i}) * (\boldsymbol{I}_b - \boldsymbol{I}_c), & \text{if } \alpha_{a,i} > 0 \\ \boldsymbol{I}_i + \alpha_{a,i} * (\boldsymbol{I}_a - \boldsymbol{I}_i) + (\alpha_{b,i} - \alpha_{c,i}) * (\boldsymbol{I}_b - \boldsymbol{I}_c), & \text{otherwise} \end{cases} \tag{8}$$

对于第二种选择方案,假设思考者选择的学习对象为 $\boldsymbol{I}_a$ 和 $\boldsymbol{I}_b$,其中 $a,b,c\in\{1,2,\cdots,ThinkerNum\}$,并且都不等于 i,则使用式(9)为其产生一个组合想法,$rand$ 为(0,1)范围内满足均匀分布的随机变量。

$$\boldsymbol{V}_i = \begin{cases} \boldsymbol{I}_i + rand * (\boldsymbol{I}_a - \boldsymbol{I}_b), & \text{if } \alpha_{a,i} > 0 \\ \boldsymbol{I}_i + rand * (\boldsymbol{I}_b - \boldsymbol{I}_a), & \text{otherwise} \end{cases} \tag{9}$$

两种选择方案对应的后续操作完全相同。在得到组合想法后,将上述组合想法与思考者的当前想法 $\boldsymbol{I}_i$ 结合,产生一个新的想法 $\boldsymbol{U}_i(u_{i,1},u_{i,2},\cdots,u_{i,n})$。$D$ 是变量的维数,新想法的产生规则如式(10)所示,其中,$k=1,2,\cdots,D$。

$$u_{i,k} = \begin{cases} v_{i,k}, & \text{if } (crand(k)\leqslant r) \text{ or } k = rnbr(i) \\ x_{i,k}, & \text{if } (crand(k) > r) \text{ and } k \neq rnbr(i) \end{cases} \tag{10}$$

$crand$ 是(0,1)内满足均匀分布的随机变量,$r\in(0,1)$是一个预设定的参数,$rnbr(i)\in(1,2,\cdots,D)$也是随机产生的。

(3)确定想法

在本阶段,将把上一阶段产生的新想法与思考者的当前想法进行比较,通过式(11)来为思考者确定一个最终想法。

$$\boldsymbol{I}_i = \begin{cases} \boldsymbol{U}_i, & \text{if } f(\boldsymbol{U}_i)\leqslant f(\boldsymbol{I}_i) \\ \boldsymbol{I}_i, & \text{otherwise} \end{cases} \tag{11}$$

2.5 经验模型

我们使用一个存储表作为记忆单元,来记录过去一段时间内思考者产生的想法 $\boldsymbol{I}_h$,表的长度视具体问题而定。我们称这个表为"经验表"。

如果连续几次迭代,思考者产生的多个想法之间的相似度均过高,则说明当前思考者可能陷入思维困境,经验模型此时就可以指导思考者脱离困境。同时,对于一些适应度评估过程复杂的问题,经验模型还可以有效减少算法的计算时间。构造的经验模型如式(12)所示:

$$D(\boldsymbol{I}) = \omega_T \sum_{i=1}^{l} \exp\{-\frac{\|\boldsymbol{I}-\boldsymbol{I}_h\|}{2\sigma_T^2}\} \tag{12}$$

其中,ω_T 为权值系数,σ_T^2 为密度方差,$\boldsymbol{I}$ 表示当前的想法,$\boldsymbol{I}_h$ 表示历史想法,l 为经验表中的历史想法个数。ω_T 通常被设置为 $1/l$。每次只要当前的想法被更新,我们就会用其替代"经验表"中最旧的想法。$D(\boldsymbol{I})$的值表示当前想法和历史想法之间的相似度,反映了局部优化的程度。本文主要用经验模型来评估发散思维和收敛思维过程中产生的想法。如果某个想法的 $D(\boldsymbol{I})$大于一个均匀分布的随机变量 $\eta\in(0,1)$,那么这个想法就不会被接受。经验模型可以有效减少问题求解的时间,尤其是对于那些具有复杂评估过程的问题。

3 创造性驱动优化算法

综合考虑创造性驱动优化模型各子模型之间的联系,我们得到对应的创造性驱动优化算法的执行步骤,如算法1所示。

算法1 创造性驱动优化算法

输入:算法的相关参数
输出:最好的想法
1. 初始化,每个思考者随机产生一个想法
2. Repeat
3. For 每个思考者
4. 发散思维
5. 收敛思维
6. 更新经验模型
7. If 满足灵感思维条件
8. 灵感思维
9. End
10. End
11. If 满足集体思维条件
12. 集体思维
13. End
14. Until 满足迭代终止条件

4 仿真实验及分析

本节将通过函数优化问题来检测 CDOA 的有效性和计算复杂度。所有的实验都运行在联想深腾 1800 高性能计算机集群上。该集群有8个计算节点和一个控制节点。每个计算节点都是一个拥有 24GB 内存、2 个 2.4GHz 四核 CPU 的高性能服务器。服务器的操作系统为 Red Hat Enterprise Linux7。实验平台为单个计算节点上的 Matlab 软件,版本为64位 R2009b,具体实验及分析见 4.1 节和 4.2 节所述。

4.1 CDOA 有效性测试及分析

4.1.1 基准函数

为验证算法的有效性,本文从文献[15]给出的 28 个 CEC-2013 基准函数中选出 8 个函数来对 CDOA 进行测试。CEC-2013 实参数优化基准函数可被划分为 3 类,其中 f_1-f_5 为单模态函数,f_6-f_{20} 为多模态函数,$f_{21}-f_{28}$ 为复合函数。本文使用的测试函数包括两个单模态函数:f_1、f_5,3 个

多模态函数：f_9、f_{14}、f_{15}，3 个复合函数：f_{21}、f_{23}、f_{24}。由于这些测试函数比较复杂，其具体描述可参见文献[15]。各测试函数的理论最优值都为 0。

4.1.2 实验设置

根据 CEC-2013 函数优化实验设置标准，分别对 10 维、30 维和 50 维的基准函数进行测试。算法在每个函数上各独立执行 51 次，记录下实验结果的均值和标准差。最大函数评估次数被设定为 $10000\times D$ 次，D 为函数的维数。对于每次独立实验，当达到最大函数评估次数或者误差值（$f_i(\boldsymbol{X})-f_i(\boldsymbol{X}^*)$）小于 10^{-8} 时，实验将终止。小于 10^{-8} 的误差值和标准差都将被视作 0。实验结果将与 3 个最先进的同类算法（SMADE[16]，SHADE[17]，ABC-SPSO[18]）进行比较。

本文中，CDOA 的实验参数设置如表 1 所列，它是我们通过参数调试所得到的一个整体效果比较好的参数组合。

表 1 CDOA 参数设置表

参数	值
思考者个数 NT	12
一次发散思维产生的想法数 DNum	6
创造性方差下界 σ^2_{min}	1e−10
创造性方差上界 σ^2_{max}	1e+4
创造性方差缩放因子 Sfactor	0.95
灵感思维触发阈值 INum	3
集体思维触发阈值 ItervalNum	4
集体思维参数 r	0.5
经验表容量 L	50
经验模型参数 σ^2_1	1e−4

4.1.3 实验结果及分析

表 2—表 4 分别列出了各算法在函数维度为 10 维、30 维、50 维时的实验结果。最好的实验结果被加粗列出。根据实验结果，我们对各算法进行排名。函数维度为 10 维、30 维、50 维时，各算法的排名分别如图 1—图 3 所示。

表 2 各算法在 10 维函数上的实验结果

函数			算法							
本文编号	对应函数（CEC2013）	维数	CDOA		SMADE		SHADE		ABC-SPSO	
			mean	std	mean	std	mean	std	mean	std
F_1	f_1	10	**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	f_5	10	1.01e−04	5.33e−05	**0.00e+00**	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**	0.00e+00
F_3	f_9	10	**1.33e+00**	8.58e−02	2.29e+00	7.19e−01	3.38e+00	7.35e−01	3.79e+00	9.62e−01
F_4	f_{14}	10	3.59e+01	2.37e+01	3.64e+00	4.39e+00	**4.90e−03**	1.70e−02	4.35e−01	8.17e−01
F_5	f_{15}	10	**4.04e+02**	2.76e+02	7.36e+02	2.60e+02	4.21e+02	1.14e+02	4.96e+02	1.37e+02
F_6	f_{21}	10	**2.90e+02**	1.09e+02	3.83e+02	5.50e+01	4.00e+02	0.00e+00	3.96e+02	2.80e+01
F_7	f_{23}	10	**4.31e+02**	2.76e+02	5.78e+02	3.16e+02	4.61e+02	1.78e+02	5.97e+02	1.81e+02
F_8	f_{24}	10	**1.25e+02**	3.35e+01	2.02e+02	1.76e+01	1.93e+02	2.46e+01	2.01e+02	2.33e+01

表 3 各算法在 30 维函数上的实验结果

函数			算法							
本文编号	对应函数（CEC2013）	维数	CDOA		SMADE		SHADE		ABC-SPSO	
			mean	std	mean	std	mean	std	mean	std
F_1	f_1	30	**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	f_5	30	5.43e−04	8.24e−05	**0.00e+00**	0.00e+00	**0.00e+00**	0.00e+00	**0.00e+00**	0.00e+00
F_3	f_9	30	**1.62e+01**	1.96e+00	2.23e+01	3.57e+00	2.75e+01	1.77e+00	2.95e+01	2.62e+00
F_4	f_{14}	30	2.02e+03	6.14e+02	1.33e+02	1.27e+02	**3.18e−02**	2.33e−02	1.55e+01	6.13e+00
F_5	f_{15}	30	**3.17e+03**	3.99e+02	4.10e+03	8.47e+02	3.22e+03	2.64e+02	3.55e+03	3.04e+02
F_6	f_{21}	30	**2.92e+02**	5.30e+01	3.27e+02	8.65e+01	3.09e+02	5.65e+01	3.18e+02	7.53e+01
F_7	f_{23}	30	**2.24e+03**	4.38e+02	4.22e+03	8.74e+02	3.51e+03	4.11e+02	4.18e+03	5.62e+02
F_8	f_{24}	30	**2.02e+02**	6.21e+00	2.32e+02	2.57e+01	2.05e+02	5.29e+00	2.51e+02	1.43e+02

表 4 各算法在 50 维函数上的实验结果

函数			算法							
本文编号	对应函数（CEC2013）	维数	CDOA		SMADE		SHADE		ABC-SPSO	
			mean	std	mean	std	mean	std	mean	std
F_1	f_1	50	**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	f_5	50	8.62e−04	2.25e−04	**0.00e+00**	0.00e+00	**0.00+00**	0.00e+00	**0.00e+00**	0.00e+00
F_3	f_9	50	**3.34e+01**	2.26e+00	4.36e+01	4.06e+00	5.54e+01	1.98e+00	5.84e+01	3.92e+00
F_4	f_{14}	50	4.89e+03	5.74e+02	3.41e+02	2.05e+02	**3.45e−02**	1.93e−02	2.64e+01	8.03e+00
F_5	f_{15}	50	**5.31e+03**	4.66e+02	8.54e+03	9.77e+02	6.82e+03	4.41e+02	7.42e+03	5.46e+02
F_6	f_{21}	50	**3.09e+02**	2.98e+02	8.46e+02	3.43e+02	8.45e+02	3.63e+02	8.96e+02	2.90e+02
F_7	f_{23}	50	**4.27e+03**	6.83e+02	9.89e+03	1.90e+03	7.63e+03	6.58e+02	9.04e+03	7.81e+02
F_8	f_{24}	50	**2.02e+02**	8.21e+00	3.02e+02	1.20e+01	2.34e+02	1.01e+01	9.04e+03	7.81e+02

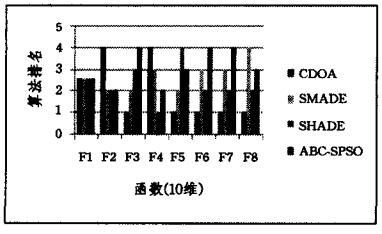

图 1 各算法在 10 维函数上的排名

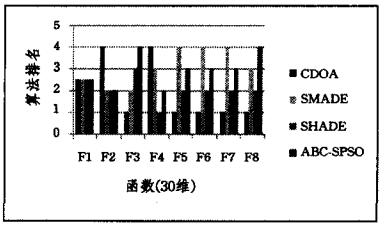

图 2 各算法在 30 维函数上的排名

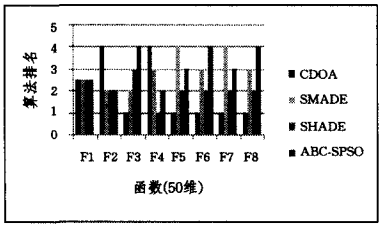

图 3　各算法在 50 维函数上的排名

从表中给出的数据以及图中的对比可以清楚地看出，不论函数维度为 10 维、30 维，还是 50 维，CDOA 在单模态函数 F_2 和多模态函数 F_4 上的效果都不如其他 3 种算法。在单模态函数 F_1 上，CDOA 与其它 3 种算法的效果相当。而在剩余的 5 个函数上，不论函数维度为 10 维、30 维，还是 50 维，CDOA 的效果都是最好的。这在一定程度上表明 CDOA 在求解复杂函数时具有相对优秀的优化效果。此外，通过分析 CDOA 的优化效果随函数维数改变而产生的变化，可以看出，其优化效果会随着函数维度的增大而有所减弱，但仍优于其它 3 种算法在相同情景下的优化效果。

4.2　CDOA 计算复杂度

根据文献[15]中设定的算法计算复杂度实验方案，我们使用与 4.1.2 节相同的实验环境测出 T_0，T_1，T_2 和 $\hat{T}_2$。其中 T_0 是在当前环境下执行下列程序所消耗的时间。

```
for:i=1:1000000
    x=0.55+(double)i;
    x=x+x;x=x/2;
    x=x*x;x=sqrt(x);
    x=log(x);x=exp(x);
    y=x/x;
end
```

T_1 是对函数 F_4 执行 200000 次函数评估所消耗的时间。T_2 是当函数评估次数为 200000 次时，使用当前算法求解 F_4 所需要的全部时间。$\hat{T}_2$ 为 5 次执行获得的 5 个 T_2 的均值。计算复杂度被定义为 $(\hat{T}_2-T_1)/T_0$。

表 5 列出了 CDOA 和 3 个对比算法(SMADE、SHADE、ABC-SPSO)的计算复杂度，对比结果分别来自于 3 个对比算法所在的文献(文献[16-18])，最好的实验结果被加粗显示。

表 5　算法计算复杂度对比

算法	计算复杂度				
	维数	T_0	T_1	$\hat{T}_2$	$(\hat{T}_2-T_1)/T_0$
CDOA	D=10	0.315	2.333	9.927	**24.107**
	D=30		4.830	14.010	**29.143**
	D=50		7.390	17.443	**31.915**
SMADE	D=10	0.021	0.824	1.847	48.7
	D=30		2.005	5.764	179
	D=50		3.417	13.281	469.7
SHADE	D=10	0.0009	0.5144	0.6089	99.7084
	D=30		1.5479	1.7198	181.4152
	D=50		2.6709	2.8802	220.7530
ABC-SPSO	D=10	0.27	0.95	25.62	91.37
	D=30		2.68	215.58	788.52
	D=50		4.41	896.10	3302.56

可以清楚看出：CDOA 算法在 10 维、30 维和 50 维函数优化实验上的计算复杂度都是最小的。而在 4.1 节的有效性实验中，各算法的函数评估次数设定是相同的，由此可以推出，在相同的实验条件下，CDOA 具有更快的速度。

结束语　人类的创造性思维具有强大的问题求解能力。受创造性思维过程的启发，本文提出了一种创造性驱动优化算法。与传统的群体智能算法相比，创造性驱动优化算法中的个体具有更高的独立性和更强的问题求解能力。这有效地提高了算法的智能性和并行性。为检验算法的有效性，8 个复杂的 CEC-2013 基准函数分别在 10 维、30 维和 50 维的情况下被用于测试。实验结果与 3 个最先进的算法进行了对比，对比结果表明 CDOA 具有相对较好的优化效果，尤其是在更为复杂的复合函数上。同时，对 CDOA 进行的计算复杂度实验及其对比分析显示，CDOA 在相同的实验条件下，相比 SMADE、SHADE、ABC-SPSO，更具有速度优势。

参 考 文 献

[1] Kephart J O. Learning from nature [J]. Science, 2011, 331 (6018): 682-683

[2] 陈建超，胡桂武，杜小勇. 广义菌群优化算法[J]. 计算机科学，2013，40(3)：251-254
Chen Jian-chao, Hu Gui-wu, Du Xiao-yong. Generalized Bacterial Foraging Optimization[J]. Computer Science, 2013, 40(3): 251-254

[3] 黄光球，李涛，陆秋琴. 种群动力学优化算法[J]. 计算机科学，2013，40(11)：280-286
Huang Guang-qiu, Li Tao, Lu Qiu-qin. Population Dynamics-based Optimization[J]. Computer Science, 2013, 40(11): 280-286

[4] Feng X, Lau F, Yu H. A novel bio-inspired approach based on the behavior of mosquitoes[J]. Information Sciences, 2013, 233: 87-108

[5] Zhang H, Zhu Y, Chen H. Root growth model: a novel approach to numerical function optimization and simulation of plant root system[J]. Soft Computing, 2014, 18(3): 521-537

[6] DeHaan R L. Teaching creative science thinking[J]. Science, 2011, 334(6062): 1499-1500

[7] Chermahini S A, Hommel B. The(b) link between creativity and dopamine: spontaneous eye blink rates predict and dissociate divergent and convergent thinking[J]. Cognition, 2010, 115(3): 458-465

[8] Chermahini S A, Hommel B. Creative mood swings: divergent and convergent thinking affect mood in opposite ways[J]. Psychological Research, 2012, 76(5): 634-640

[9] Mumford M D, Medeiros K E, Partlow P J. Creative thinking: Processes, strategies, and knowledge[J]. The Journal of Creative Behavior, 2012, 46(1): 30-47

[10] Kounios J, Beeman M. The cognitive neuroscience of insight[J]. Annual Review ofPpsychology, 2014, 65: 71-93

[11] Balter M. Did Working Memory Spark Creative Culture? [J]. Science, 2010, 328(5975): 160-163

[12] Lee C S, Therriault D J. The cognitive underpinnings of creative thought: A latent variable analysis exploring the roles of intelligence and working memory in three creative thinking processes [J]. Intelligence, 2013, 41(5): 306-320

[13] Ashton-James C E, Chartrand T L. Social cues for creativity: The impact of behavioral mimicry on convergent and divergent thinking[J]. Journal of Experimental Social Psychology, 2009,

45(4):1036-1040

[14] Sternberg R J. Innovation: Lighting the creative spark[J]. Nature, 2010, 468(7321): 170-171

[15] Liang J J, Qu B Y, Suganthan P N, et al. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization [R]. Zhengzhou: Computational Intelligence Laboratory, Zhengzhou University, Singapore: Nanyang Technological University, 2013

[16] Caraffini F, Neri F, Cheng J, et al. Super-fit Multicriteria Adaptive Differential Evolution[C]// 2013 IEEE Congress on Evolutionary Computation(CEC). IEEE, 2013: 1678-1685

[17] Tanabe R, Fukunaga A. Evaluating the performance of SHADE on CEC 2013 benchmark problems[C]// 2013 IEEE Congress on Evolutionary Computation(CEC). IEEE, 2013: 1952-1959

[18] El-Abd M. Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks[C]// 2013 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2013: 2215-2220

(上接第 239 页)

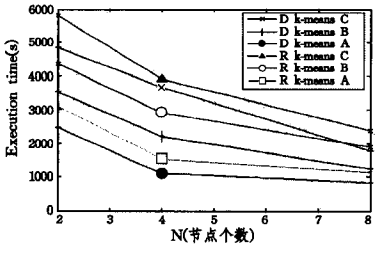

图 8 与随机选择初始聚类中心算法的性能比较

结束语 本文研究分布式环境下大规模数据集的 k-means 聚类问题，首先提出了当下流行的 MapReduce 计算框架下的 k-means 算法。MapReduce 框架下由于每个MapReduce 任务都需要读写分布式文件系统，因此 MapReduce 表达作业之间依赖关系时是低效的，这就导致现有的基于MapReduce 的算法的效率不高。本文在 MapReduce 框架的基础上提出了基于数据流的框架，并在该框架上提出了一种基于数据流计算框架的 k-means 算法 DKmeans，该算法将k-means 聚类过程归结为两个步骤，从而可以减少读写分布式文件系统的次数，有效降低查询过程中的 I/O 代价。针对步骤一中初始聚类中心选择的问题，本文提出了基于采样的初始聚类中心选择算法。随着摩尔定律渐渐失效以及待处理数据的爆炸式增长，基于数据流计算框架的 k-means 算法能高效地处理大规模数据的 k-means 聚类问题。实验表明，本文算法具有良好的可扩展性和执行效率，具有实用价值。

参 考 文 献

[1] Han Jia-wei, Kamber M. Data mining concepts and techniques, second edition[M]. Elsevier (Singapore) Pte Ltd, 2006: 251-263

[2] Kriegel H P, Kröger P, Zimek A. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2009, 3(1): 1

[3] Forgy E. Cluster analysis of multivariate data: Efficiency vs. Interpretability of classifications [J]. Biometrics, 1965, 21(3): 768

[4] Malewicz G, Austern M H, Bik A J C, et al. Pregel: a system for large-scale graph processing [C] // Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. ACM, 2010: 135-146

[5] Wang J, Su X. An improved K-Means clustering algorithm[C]// 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN). IEEE, 2011: 44-46

[6] Kumar N S, Rao K N, Govardhan A, et al. Undersampled K-means approach for handling imbalanced distributed data[J]. Progress in Artificial Intelligence, 2014, 3(1): 1-10

[7] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[8] Apache. Hadoop[EB/OL]. (2014-4-10)[2014-4-22]. http://hadoop.apache.org/

[9] Ordonez C, Omiecinski E. Efficient disk-based K-means clustering for relational databases[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(8): 909-921

[10] Pelleg D, Moore A W. X-means: Extending K-means with Efficient Estimation of the Number of Clusters[C]// ICML. 2000: 727-734

[11] Huang J Z, Ng M K, Rong H, et al. Automated variable weighting in k-means type clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(5): 657-668

[12] AlSabti K, Ranka S, Singh V. An efficient space-partitioning based algorithm for the K-means clustering[M]// Methodologies for Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 1999: 355-360

[13] 王守强，朱大铭，韩爱丽. 基于初始点选取的 k-means 聚类近似常数算法[J]. 计算机研究与发展，2007(z2)：69-74
Wang Shou-qiang, Zhu Da-ming, Han Ai-li. A Constant Approximate Alogritnm for K-Means Problem Based on Inital Point Selecting [J]. Journal of Computer Research and Development, 2007(z2): 69-74

[14] 李飞，薛彬，黄亚楼. 初始中心优化的 K-Means 聚类算法[J]. 计算机科学，2002，29(7)：94-96
Li Fei, Xue Bin, Huang Ya-lou. K-Means Clustering Algorithm with Refined Inital Center[J]. Computer Science, 2002, 29(7): 94-96

[15] Dhillon I S, Modha D S. A data-clustering algorithm on distributed memory multiprocessors [M] // Large-Scale Parallel Data Mining. Springer Berlin Heidelberg, 2000: 245-260

[16] Zhao W, Ma H, He Q. Parallel k-means clustering based on mapreduce[M] // Cloud Computing. Springer Berlin Heidelberg, 2009: 674-679

[17] 江小平，李成华，向文，等. K-means 聚类算法的 MapReduce 并行化实现 [J]. 华中科技大学学报(自然科学版)，2011，39(1)：120-124
Jiang Xiao-ping, Li Cheng-hua, Xiang Wen, et al. Parallel implementing k-means clustering algorithm using MapReduce programming mode [J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2011, 39(1): 120-124

[18] Li Q, Wang P, Wang W, et al. An Efficient K-means Clustering Algorithm on MapReduce [C]// Database Systems for Advanced Applications. Springer International Publishing, 2014: 357-371