文章编号:1671 - 6833(2018)03 - 0015 - 07

进化算法在大规模优化问题中的应用综述

梁 静[1], 刘 睿[1], 瞿博阳[2], 岳彩通[1]

(1. 郑州大学 电气工程学院,河南 郑州 450001;2. 中原工学院 电子信息学院,河南 郑州 450007)

摘 要: 针对大规模问题的特点,对已有的大规模进化算法进行了简单的分析,主要介绍算法的初始化方法、不分组策略、静态分组策略、动态分组策略、自适应分组策略、大规模优化算法测试函数集以及算法结果的对比等方面;侧重描述优化算法的搜索策略、更新策略、突变策略和协同进化策略,并列出大规模优化算法测试函数集的特点及优化算法的评价方法;最后,给出了目前大规模优化问题的几个研究难点.

关键词: 大规模优化问题; 进化算法; 协同进化; 种群初始化; 基准测试函数

中图分类号: TP18 **文献标志码**: A doi:10.13705/j.issn.1671 - 6833.2017.06.016

0 引言

当今很多优化问题已经从简单问题发展成为复杂问题. 许多科学和工程应用问题都可以设计成大规模优化问题来进行求解,例如:大型电力系统、大量的资源调度问题、大规模交通网络的车辆路径规划等. 然而随着优化问题越来越复杂,一些经典的算法已经不能满足实际需要. 最近几年进化优化在许多实值和组合优化问题上取得了很大的成功,但是大多数的随机优化算法都会遭受“维数灾难”. 因此,近些年学者们利用进化算法进行了多种有价值的尝试,并且针对大规模优化问题组织了专题会议,如 Special Session on Evolutionary Computation for Large Scale Global Optimization,设计了新的测试函数、建立了相关的网站,并且在 IEEE Transactions on Evolutionary Computation、Information Sciences、Soft Computing、Applied Soft Computing 等优秀期刊也刊登了对于大规模问题的研究进展,显示出此研究领域的重要性.

1 数学表述

大规模优化问题可以用式(1)表述:

$$\min/\max F(x) = f(x_1, x_2, \cdots, x_n),\ x \in X. \quad (1)$$

式中,$X \subseteq \mathbb{R}^n$ 表示可行解集;n 表示搜索空间的维数(即决策变量的个数);$x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$ 表示决策变量;$f: X \to \mathbb{R}$ 则表示一个从 n 维空间映射到一维适应度值 $F(x)$ 的实值非连续性目标函数. 在大规模设置中决策变量的个数 n 一般大于 100[1],通常达到 1 000 维以上.

2 解决方法

在算法开始阶段,文献[2]给出了针对大规模问题的初始化方法,可以更加有效地寻找极点. 而对于算法的主体部分,一般来说主流的策略可以分为两大类:协同进化策略和不分组策略. 协同进化策略是由分治策略进化而来,主要思想是把大规模复杂问题分解成单变量或低维简单问题逐一解决;而不分组策略是运用一些特殊的策略或联合其他有效的算法来改进它们在解决大规模问题时的性能.

2.1 种群初始化方法

种群初始化方法主要包括以下 5 类:随机方法、定值设定法、两步式方法、混合方法和具体应用法. 随机生成是最常用的方法,然而,在面对大规模优化问题(决策变量超过 100)时,这种初始化方法效果不佳. 文献[2]主要列出了一些不同初始化方法的对比研究.

在随机方法中,比较常用的是使用随机数产生器随机生成. 而定值设定法则比较偏向于在搜索空间中产生均匀分布的点,在缺乏问题先验知识的情况下,一个比较均匀的种群可以促进算法在迭代早期的探索能力. 近些年,两步式初始化方

收稿日期:2017 - 05 - 04;**修订日期**:2017 - 08 - 22

基金项目:国家自然科学基金资助项目(61673404,61473266)

作者简介:梁静(1981—),女,河南郑州人,郑州大学教授,博士,博士生导师,主要从事进化计算理论与应用研究,E-mail:liangjing@zzu.edu.cn.

法在研究中较为常用,此方法分为前期产生初始点,后期根据条件改进这些点.混合方法一般来说是一些基础方法的组合.具体应用法则是指根据一些特殊的实值问题专门设计的初始化方法.文献[3]给出了现在常用的 8 种初始化方法(与 DE 算法相结合)的测试结果,使用的测试函数是 CEC'2008[4].

2.2 不分组策略

学者们一般根据算法在不同阶段的特性设置不同的策略来解决低维问题,所以,针对大规模优化问题改进这些策略是一种比较常用的方法.

2.2.1 子代产生策略

一般来说,每种进化算法都有固定的子代产生策略.但是,对于大规模问题,使个体广泛分布在高维空间中比较困难.在每次迭代过程中,算法不断的收敛,因此下一代的学习策略很重要,不仅要向好的方向进化还要在搜索空间中广泛探索.文献[5]提出了反向学习策略,这种产生策略具有空间的导向性,可以增加种群的多样性,因此将其融入 DE 算法来解决大规模优化问题.文献[6]提出了基于广义反向学习的 DE 算法,该算法用广义反向学习方法(generalized opposition-based learning,GOBL)产生子代个体,用 DE 算法对产生的个体进行优化. 图 1 表示 4 个不同的广义反向学习的模型,其中, x 是当前解,x^* 是反向解.

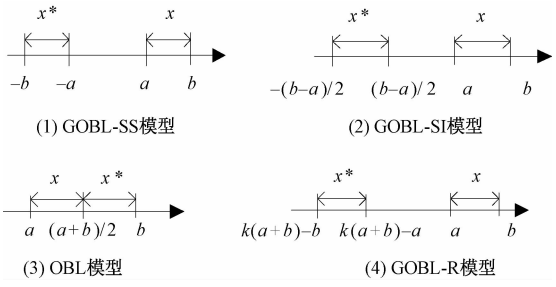

图 1 4 个不同的广义反向学习模型

Fig. 1 Four different GOBL models

图 1 分别表示了文献[6]中所使用的 4 个 GOBL 模型,其中 a 表示搜索空间的下界,b 表示搜索空间的上界,而 GOBL-R 模型是上述模型的一般表达式.当 $k=0$ 时,是 GOBL-SS 模型;当 $k=1/2$ 时,是 GOBL-SI 模型;当 $k=1$ 时,是 OBL 模型.这些模型用于产生新个体并与原始个体混合进行选择.

2.2.2 新的变异策略

JADE 是 Zhang 于 2009 年提出把变异策略和外部存档策略相结合的自适应方法[7].在无存档策略中,突变向量用式(2)产生:

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{i,g} + F_i \cdot (\boldsymbol{x}^p_{\text{best},g} - \boldsymbol{x}_{i,g}) + F_i \cdot (\boldsymbol{x}_{r1,g} - \boldsymbol{x}_{r2,g}), \tag{2}$$

式中,$\boldsymbol{x}_{i,g}$ 表示第 g 代第 i 个个体;$r1$、$r2$ 是从集合 $\{1,2,\cdots,i-1,i+1,\cdots,NP\}$ 中均匀抽取的两个不同的整数;把当前种群按适应度值从大到小排列,从前 $p\%$ 的个体中随机抽取一个作为 $\boldsymbol{x}^p_{\text{best},g}$,并且 $p \in (0,100]$;F_i 是由 $\boldsymbol{x}_i$ 决定的突变因子.

笔者在比较当前种群时,发现劣解可以为种群进化方向提供有用的信息.定义 $\boldsymbol{A}$ 为劣解归档集,$\boldsymbol{P}$ 为当前种群,有存档的突变策略"DE/current-to-pbest/1"的突变向量用式(2)产生时,$\boldsymbol{x}_{r2,g}$ 是从 $\boldsymbol{P} \cup \boldsymbol{A}$ 中随机选择的个体.

文献[8]则给出了针对大规模全局优化的连续差分进化邻域搜索算法(sequential differential evolution enhanced by neighborhood search,SDENS),此算法主要分为两部分:①针对每个个体通过局部和全局邻域搜索策略产生两个实验个体;②从当前个体和两个新产生的实验个体中选取合适的一个作为新的当前个体.局部和全局邻域搜索策略在文献[8]中是一种突变策略.

一般来说,DE/target-to-best/1 突变策略(如式(3)所示)主要着重于开发(exploitation),即所有的个体都会向同样的最优点 $\boldsymbol{X}_{\text{best}}$ 移动,这样就造成算法收敛过快[9].Das 等在文献[9]中改进了 DE/target-to-best/1 突变策略,提出了两个突变策略:局部邻域和全局邻域.

$$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{i,G} + F \cdot (\boldsymbol{X}_{\text{best},G} - \boldsymbol{X}_{i,G}) + F \cdot (\boldsymbol{X}_{r1,G} - \boldsymbol{X}_{r2,G}). \tag{3}$$

式中:$\boldsymbol{X}_{\text{best},G}$ 表示在当前代 G 种群的最优位置;$r1$、$r2 \in \{1,2,\cdots,N_p\}$,且 $i \neq r1 \neq r2$.

局部邻域突变中最优位置是小邻域中的最优位置,不是全部种群的最优.改进后的模型表示如式(4):

$$\boldsymbol{L}_{i,G} = \boldsymbol{X}_{i,G} + \alpha \cdot (\boldsymbol{X}_{n-\text{best}_{i,G}} - \boldsymbol{X}_{i,G}) + \beta \cdot (\boldsymbol{X}_{p,G} - \boldsymbol{X}_{q,G}), \tag{4}$$

式中:下标 n-best$_{i,G}$ 表示 $\boldsymbol{X}_{i,G}$ 邻域的最优个体;邻域大小是 k;p、$q \in [i-k,i+k]$ 且 $p \neq q \neq i$.个体会向其相应邻域的最优点靠近,特殊点的吸引力减弱,这样就避免陷入局部最优.

全局邻域突变在原始 DE/target-to-best/1 突变策略中加上了 α 和 β 这两个比例因子,如式(5)所示:

$$\boldsymbol{G}_{i,G} = \boldsymbol{X}_{i,G} + \alpha \cdot (\boldsymbol{X}_{\text{best},G} - \boldsymbol{X}_{i,G}) + \beta \cdot (\boldsymbol{X}_{r1,G} - \boldsymbol{X}_{r2,G}). \tag{5}$$

针对这两个突变策略,文献[9]采用一个权

重 $w \in (0,1)$ 合并成一个新的突变策略,如式(6)所示:

$$\boldsymbol{V}_{i,G} = w \cdot \boldsymbol{G}_{i,G} + (1 - w) \cdot \boldsymbol{L}_{i,G}. \quad (6)$$

2.2.3　自适应策略

自适应策略可以适应多种类型的测试函数集,对于全局优化的问题比较有效,但是此策略一般都局限在低维问题中. 所以 Yang 在文献[10]中针对大规模优化问题对自适应策略进行了扩展,提供了更加广泛的参数自适应方案,提出了广义自适应差分进化算法(generalized adaptive DE, GaDE). 一般的自适应策略可以被分为两类:基于启发式规则的和基于概率分布规则的. 基于启发式规则的策略一般会引进一些新的参数,而且这些参数在某些情况下设置比较困难,JDE 和 DE-GL 算法就是用的此类策略. 而 SaDE、SaNSDE 和 JADE 则属于基于概率分布的自适应方法. 在这类算法中,不同的参数值是根据某一概率分布随机产生的,在进化操作和选择过程之后,好的参数值将作为下一次进化的分布规律被记录下来,用于产生更好的解. 文献[10]中提出的自适应方式用的是第二类基于概率分布的策略. 对于一个给定的进化算法,假设它有一个基于个体且非常敏感的参数 $\boldsymbol{A} \in [\boldsymbol{A}_{\min}, \boldsymbol{A}_{\max}]$,同时此参数需要在进化过程中调整. 对于大规模优化问题,自适应策略不仅用在算法的参数调节上,在 2.3 节的分组策略中也有应用.

2.2.4　局部搜索策略

文献[11]把基于局部搜索的动态多种群粒子群优化算法(dynamic multi-swarm particle swarm optimizer with local search, DMS-L-PSO)扩展到大规模问题上,并取得了良好的效果. DMS-PSO 是根据邻域结构把大种群分成很多小种群,这些小种群利用不同的重组策略被频繁重组,在频繁重组的过程中,种群不断交换它们之间的信息. 而局部搜索策略是解决大规模优化问题的一种有效方法,主要加强算法的局部搜索能力.

把局部搜索策略加入 DMS-PSO 算法中:①每 L 代,根据小种群适应度值进行排序,利用准牛顿法根据局部最优解抽取小种群的前 25%;②在搜索算法的结尾,利用准牛顿法更改当前的最优解.

2.2.5　新的进化策略

在解决大规模优化问题上,除了在原有的算法中加入新的策略,Cheng 等也提出了一些新的群集智能算法用于解决此类问题. 文献[12]提出了社会学习的粒子群优化算法(social learning particle swarm optimization algorithm, SL-PSO),不同于经典的粒子群优化算法(particle swarm optimization, PSO)利用历史信息(包括整个种群的最优位置 global best 和每个粒子的历史最优位置 personal best)更新粒子,新算法则是粒子向当前种群中比它优秀的个体学习,具体的学习方式如图 2 所示.

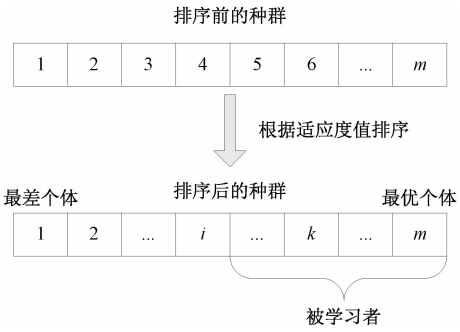

图 2　SL-PSO 的种群排序及学习行为

Fig. 2　Swarm sorting and behavior learning in SL-PSO

首先,对当前种群按适应度值排序,如果要更新第 i 个粒子,就向比第 i 个粒子好的个体(即图 2 中的被学习者)和平均位置学习. 而文献[13]则提出了竞争学习算法(competitive swarm optimizer, CSO),即随机从当前种群中抽取两个个体比较适应度值,失败者向胜利者学习,胜利者并不学习直接进入下一次循环,如图 3 所示.

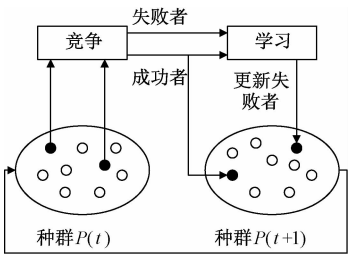

图 3　CSO 的一般构架

Fig. 3　The general idea of CSO

2.2.6　小种群搜索策略

文献[14]介绍了小种群搜索策略(minimum population search, MPS),这个策略主要是针对多模态问题. 为了提高 MPS 在解决多模态问题时的性能,文献[15]使用了阈值收敛(threshold convergence, TC)方法来完成有序无偏的探索(exploration). MPS 使用了相对较小的种群来提高可扩展性,种群越小,循环的代数越多,评价次数的利用率越高. 如果种群大小 n 小于问题维数 d,将其种群定义为 $n-1$ 维超平面. 新解要严格按照所定义的超平面产生. 在 MPS 中,每个种群成员用

式(7)的方式初始化:

$$\boldsymbol{S}_k = (rs_1 \cdot bound/2, rs_2 \cdot bound/2, \cdots, rs_i \cdot bound/2, \cdots, rs_n \cdot bound/2), \quad (7)$$

式中:$\boldsymbol{S}_k$ 是第 k 个粒子;rs_i 是介于 -1 到 1 之间的随机数;$bound$ 是维数上界.

阈值按式(8)更新:

$$min\_step_i = \alpha \cdot diagonal \cdot ([FEs - k]/FEs)^{\gamma}, \quad (8)$$

式中:α 是主要空间对角线的分数;FEs 是总评价次数;k 是使用过的评价次数;γ 是控制阈值衰减率的参数(注:$max\_step_i = 2 \cdot min\_step_i$).

根据式(9)用父代产生子代:

$$\boldsymbol{trial}_i = \boldsymbol{x}_i + F_i \cdot (\boldsymbol{x}_i - \boldsymbol{x}_c) + \boldsymbol{O}_{step\_i} \cdot orth, \quad (9)$$

式中:x_i 和 x_c 分别是父代中的个体及父代的中心点;F_i 是范围为 $[-max\_step, max\_step]$ 均匀分布的随机数. 为了确保产生的新试探解在阈值范围之内,增加了正交算子.

在大规模优化问题中,由于增加了维数,MPS 的种群也会增加,而这些增加的种群会使评价次数的有效利用率降低. 为了增加这一利用率,MPS 的种群大小将会动态减小如式(10)所示:

$$pop\_size_t = init\_pop \cdot ((FEs - k)/FEs). \quad (10)$$

2.3 分组策略

分组策略是指将原始的大规模问题分解成一系列小且简单的子问题,用分别优化独立子问题的方式解决. 这种被称为分治策略最早是由 Descartes 在文献[16]中提出的,后来 Potter 在文献[17]中介绍了分组策略在大规模问题的求解方法,设计了协同进化(cooperative coevolution, CC)算法来改进标准遗传算法的性能.

协同进化策略在早期是静态分组的,分组并不会改变,但是这种方法在解决不可分或部分可分问题时主要依赖于在初始化时的分组情况,性能很不稳定,所以在改进该策略时使用动态分组方法,包括随机动态分组和学习动态分组.

2.3.1 基于静态分组的协同进化算法

Potter 在文献[17]中提出了协同进化遗传算法(cooperative coevolutionary genetic algorithms, CCGAs),但是 CCGA-1 和 CCGA-2 只在最高为 30 维的问题中测试. 2004 年 Frans 等在文献[18]中将该策略和粒子群优化算法(PSO)相结合,提出了 CPSO-SK 和 CPSO-HK. CPSO-SK 将维数分为 K 组,每组的维数是 $[n/K]$,用 PSO 算法对每组的维数进行更新. 虽然 CPSO-SK 可以跳出次优解,但是在一些测试函数中收敛过快,为了使算法同时具有 PSO 的开发能力,CPSO-HK 把这两种算法结合起来,一部分使用 CPSO-SK 算法,将 CPSO-SK 算法的最优解随机赋给用 PSO 优化的种群,条件满足时停止.

2.3.2 基于动态分组的协同进化算法

对于不可分问题来说,静态分组效率十分低. 在不可分的测试函数中,决策变量存在着一些相互关系(正相关或负相关),如果把这些相互关联的决策变量一直分在一个组内,结果并不能收敛到最小.

2.3.2.1 随机动态分组

Yang 于 2008 年在文献[19]中提出了新的协同进化(cooperative coevolution, CC)框架,同时还加上了自适应权重策略. 其中,新的 CC 框架设计成动态改变群体结构,这种设计增加了相互关联决策变量分在一起优化的几率. 此框架的主要思想是将 n 维的目标向量分成 m 个 s 维的子部分(假设 $n = m \cdot s$),使用 EA 算法对子部分进行优化;自适应权重策略则是将每个子部分都设置一个权重,用某一算子对权重进行优化.

文献[20]对文献[19]中的算法进行了改进,提出了新的多层协同进化算法(multilevel CC algorithm, MLCC),通过使用一个分解池来使分组的大小和目标函数联系的更加紧密. MLCC 算法可以自适应选择合适的相互作用层次而不用考虑目标问题和优化阶段的特点,文献[7]介绍的 JADE 算法也使用了分组策略和权重优化的方法来寻找更加精确的解决方案,同时,这种解决方法可以加入其他的优化算法来改进它们在大规模优化问题中的性能. Li 在文献[21]中同样将随机分组的协同进化策略与自适应权重用在 PSO 中,提出了 CCPSO 算法,并且于次年提出了 CCPSO2 算法,完善了 CCPSO 算法,使其在 2 000 维的问题中也有很好的性能.

2.3.2.2 基于学习的动态分组

虽然动态随机分组策略对于静态分组来说在解决决策变量相关性问题上比较有优势,但是决策变量的相关性并不能完全地区分出来. 为了解决不可分问题并且减少决策变量之间的相互关联,Ray 在 2009 年提出 CCEA-AVP(自适应可变分区的协同进化算法, cooperative coevolutionary algorithm with adaptive variable partitioning). 该算法的过程类似于标准的 EA 算法,首个 M(一般设为 5)次迭代包含所有的变量. 在下一次迭代中,使用前 50% 的个体计算关联矩阵,并且将变量分

成多个子种群. 当变量之间的相关系数大于一个阈值时就被分到预先定义的一个子种群中,之后的每次迭代都把变量按其相关性分组.

文献[22]给出了辨识变量之间的相互关系的一个简单方法,"best"是目前的最优解;"new"是使用 CC 算法优化第 i 维后的最优个体;"rand"是随机从种群中选择的个体. 根据这 3 个向量产生新的个体,如式(11)所示:

$$x_j = \begin{cases} \text{new}_i, & \text{if } j = i; \\ \text{best}_j, & \text{otherwise}, \end{cases}$$
$$x'_j = \begin{cases} \text{new}_i, & \text{if } j = i; \\ \text{rand}_k, & \text{if } j = k; \\ \text{best}_j, & \text{otherwise}. \end{cases} \tag{11}$$

若 $f(x')$ 比 $f(x)$ 的适应度值好,则维数 i 和 j 相互关联的概率增加.

Omidvar 在文献[23]中引入了向量 $\Delta = \{\bar{\delta}_1, \bar{\delta}_2, \cdots, \bar{\delta}_n\}$(其中 n 表示维数),粗略估计相互关联的变量在同一区间的指标,然后基于相应 δ 的值对决策变量进行排序,根据预先定义的分组长度对决策变量分组,使用文献[19]中的方法对每组进行优化. 文献[24]介绍了另一种基于学习的维数分组方法,确定决策变量是否可分,理论定义如下所示:

定义:粒子 $x = (x_1, x_2, \cdots, x_n)$ 的优化函数是 $f(x)$,若变量 x_i 和 x_j 可分,其他变量为 $\forall c = (\cdots, x_{i-1}, x_{i+1}, \cdots, x_{j-1}, x_{j+1}, \cdots)$,且满足 $\forall x_i, x'_i \in S$, $\forall x_j \in S$. 如果 $f(\alpha) \leqslant f(\beta)$,则 $\forall x'_j \in S, f(\alpha') \leqslant f(\beta')$,其中 $\alpha = (\cdots, x_i, \cdots, x_j, \cdots)$, $\beta = (\cdots, x'_i, \cdots, x_j, \cdots)$, $\alpha' = (\cdots, x_i, \cdots, x'_j, \cdots)$, $\beta' = (\cdots, x'_i, \cdots, x'_j, \cdots)$, $x_i \neq x'_i, x_j \neq x'_j$;如果上述条件不满足,则 x_i 和 x_j 不可分. 上面定义判断两个变量之间是否可分. 但是,在实际问题中使用上面的定义,计算代价太大,所以文献[24]提出了快速独立搜索策略,若 $(f(\alpha) - f(\beta)) \cdot (f(\alpha') - f(\beta')) < 0$,则相应的变量不可分. Mahdavi 在文献[25]中提出了使用 k 聚类分组的方法,将决策变量分成了不同的低维水平.

3　测试函数

为了对比算法的性能,一般用统一的测试函数来测试. 文献[4]介绍了针对大规模优化问题的 CEC'08 专题会议函数测试集. CEC'10 专题会议上的大规模优化算法测试集[4]则包含 20 个测试函数. 为了更好地代表实际问题范围广泛的特性及对于基于分组的优化算法提供更加完备的测试,提出了 CEC'13 的测试函数集. 与 CEC'10 的区别在于:CEC'10 编写的不可分子部分的大小是均等的;而 CEC'13 则根据实际问题使不可分子部分的大小不均等,其相应变量的贡献度也会有所区分,并且不可分子部分也会有所覆盖,另外还具有病态、对称破裂及不规则等属性. CEC'13中包含了 15 个测试函数,这些测试函数均有平移和旋转特性.

4　算法对比

测试函数为 CEC'08 和 CEC'10 中的测试函数,主要测试的维数是 1 000 维,使用评价次数记录测试结果. 在对比算法时保证最大评价次数相同,对算法的优化结果进行对比.

采用 CEC'08 测试集的主要有 CODE、sep-CPM-ES、CSO、CCPSO、DECC-G、CDECC、MTS、CCPSO2、EPUS-PSO、DMS-PSO、MLCC. 文献[21]中将算法 CSO、CCPSO2、MLCC、sep-CMA-ES、EPUS-PSO、DMS-PSO 进行了对比,并且采用了 T 检验方法对结果进行了统计. DMS-PSO 可以准确地找出 f_1 和 f_5 的全局最优,其他 5 个函数的测试结果却没有 CSO 好. 在与 MLCC 比较时,对于函数 f_4 测试结果显示,MLCC 明显优于 CSO,而测试函数 f_4 是 shifted Rastrigin function,具有十分多的局部最优解,MLCC 中针对每组的优化使用的是微分进化变异的方法,而 CCPSO2 的收敛性是这几个算法中最快的. sep-CMA-ES 和 EPUS-PSO 在 CEC'08 的测试结果相比来说并不好. 文献[21]同样是采用 T 检验来显示测试结果,因为 CCPSO 中含有一些用户自定义的参数,所以文献中对每个参数进行了测试,总的来说,除了函数 f_1、f_2 和 f_3, CCPSO2 的性能要比 DECC-G 的好. 文献[24]直接列出了 CDECC 和 MTS 在 CEC'08 上的测试结果,并使用 Friedman 检验了算法之间是否具有显著性差异,其置信度为 0.05. 测试结果表明 CDECC 在函数 f_6和 f_7中结果比较好,在 f_3、f_4 和 f_5 没有 MTS 性能好,但是两者没有显著性差异.

使用 CEC'10 测试函数集主要有 DECC-DG、MOFBVE、DECC-DML、DECC-G、DECC-D 和 ML-CC 等算法. 文献[23]主要对比了 DECC-DML、DECC-G、DECC-D 和 MLCC 算法的测试结果,DECC-DML 算法在 20 个测试函数结果中有 14 个函数的表现比 DECC-G 和 DECC-D 好,在与 ML-CC 的比较中有 12 个函数的测试结果比较好,并

且在文献[23]中使用多次运行的成功率来显示算法的性能.文献[25]主要是DECC-DG和MOFBVE的对比,MOFBVE算法的7个测试函数结果比DECC-DG优秀,但是在函数f_{11}和f_{16}中DECC-DG结果比较好.

5 结论

笔者主要总结了几个大规模优化的常用方法,解决大规模问题主要面临以下难点:搜索空间随着变量增加以指数形式扩大;问题的特性随着维数的增加变得更难,例如单峰问题可能会转变为多峰问题;算法在解决问题时花费的代价十分大;对于协同进化策略来说,变量之间是否可分是主要的问题.

(1)分组优化问题.协同进化策略是解决大规模问题的主要方法,但是,决策变量之间是否可分限制了协同进化策略计算,虽然使用学习的方式来判断是否可分,但是该方法的代价随着变量的增多而迅速加大.

(2)全部可分和全部不可分问题.在大多基准测试函数集中,都有全部可分或不可分问题,这些问题使用一般协同进化方法效果并不明显.对于全部可分问题,显然对每维单独优化比较好;但是对于不可分问题,变量的分组方式仍然是一个难题.

(3)不平衡的测试函数.在实际问题中,一般都会遇到子部分分布的不平衡特性,CEC'13大规模优化问题测试集中就针对该特性设计了测试函数.

(4)更加高维的问题.在上述算法中,测试任务一般是1 000维的,大规模优化方法可扩展性是未来研究工作的一个至关重要的要求.

参考文献:

[1] MAHDAVI S, SHIRI M E, RAHNAMAYAN S. Metaheuristics in large-scale global continues optimization: a survey [J]. Information sciences, 2015, 295: 407 – 428.

[2] RAHNAMAYAN S,TIZHOOSHH R, SALAM M M A. A novel population initialization method for accelevating evolutionary algorithms[J]. Computers & mathenmatics with applications, 2007,53(10):1605 – 1614.

[3] KAZIMIPOUR B, LI X, QIN A K. Initialization methods for large scale global optimization[C]//IEEE Congress on Evolutionary Computation. Cancun: Springer, 2013: 2750 – 2757.

[4] TANG K, YAO X, SUGANTHAN P N, et al. Benchmark functions for the CEC'2010 special session and competition on large scale global optimization[R]. Hefei: Nature inspired computation and applications laboratory, USTC, China, 2009.

[5] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Quasi-oppositional differential evolution [C]// IEEE Congress on Evolutionary Computation. Tokyo: Springer, 2007:2229 – 2236.

[6] WANG H, WU Z, RAHNAMAYAN S. Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems[J]. Soft computing, 2011, 15(11): 2127 – 2140.

[7] ZHANG J, SANDERSON A C. JADE: adaptive differential evolution with optional external archive [J]. IEEE transactions on evolutionary computation, 2009, 13(5):945 – 958.

[8] WANG H, WU Z, RAHNAMAYAN S, et al. Sequential DE enhanced by neighborhood search for large scale global optimization[C]//IEEE Congress on Evolutionary Computation. Shanghai: Springer, 2010: 1 – 7.

[9] DAS S, ABRAHAM A, CHAKRABORTY U K, et al. Differential evolution using a neighborhood-based mutation operator[J]. IEEE transactions on evolutionary computation, 2009, 13(3):526 – 553.

[10] YANG Z, TANG K, YAO X. Scalability of generalized adaptive differential evolution for large-scale continuous optimization[J]. Soft computing, 2011, 15(11):2141 – 2155.

[11] ZHAO S Z, LIANG J J, SUGANTHAN P N, et al. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization[C]// IEEE Congress on Evolutionary Computation. Washington: Springer, 2008: 3845 – 3852.

[12] CHENG R, JIN Y. A social learning particle swarm optimization algorithm for scalable optimization[J]. Information sciences, 2015, 291(6):43 – 60.

[13] RAN C, JIN Y. A Competitive swarm optimizer for large scale optimization[J]. IEEE Transactions on Cybernetics, 2014, 45(2):191 – 204.

[14] BOLUFE R A, FIOL G S, CHEN S. A minimum population search hybrid for large scale global optimization [C]//IEEE Congress on Evolutionary Computation. Sendai: Springer, 2015: 1958 – 1965.

[15] MONTGOMERY J, CHEN S. A simple strategy for maintaining diversity and reducing crowding in differential evolution[C]//IEEE Congress on Evolutionary Computation. Vienna: Springer, 2012: 692 – 2699.

[16] DESCARTES R. Discourse on method[J]. Elizabeth haldane & g. r. t. ross the philosophical works of descartes, 1998, 10(1991):11 – 25.

[17] POTTER M A, JONG K A D. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Berlin, Heidelberg: Springer, 1994: 249 – 257.

[18] FRANS V D B, ENGELBRECHT A P. A cooperative approach to particle swarm optimization[J]. IEEE transactions on evolutionary computation, 2004, 8(3):225 – 239.

[19] YANG Z, KE T, XIN Y. Large scale evolutionary optimization using cooperative coevolution[J]. Information sciences, 2008, 178(15):2985 – 2999.

[20] YANG Z, TANG K, YAO X. Multilevel cooperative coevolution for large scale optimization[C]//2008 IEEE Congress on Evolutionary Computation. Washington: Springer, 2008: 1663 – 1670.

[21] LI X, YAO X. Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms[C]//2009 IEEE Congress on Evolutionary Computation. Vienna: Springer, 2009: 1546 – 1553.

[22] WEICKER K, WEICKER N. On the improvement of coevolutionary optimizers by learning variable interdependencies[C]//IEEE Congress on Evolutionary Computation. Washington: Springer, 1999(3):1627 – 1633.

[23] OMIDVAR M N, LI X, YAO X. Cooperative co-evolution with delta grouping for large scale non-separable function optimization[C]//IEEE Congress on Evolutionary Computation. Shanghai: Springer, 2010: 1 – 8.

[24] GE H, SUN L, YANG X, et al. Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation[J]. Applied soft computing, 2015, 36: 300 – 314.

[25] MAHDAVI S,RAHNAMAYAN S, SHIRIM E. Muctilevel framework for large-scale global optimization[J]. Soft computing, 2017,21(14):4111 – 4140.

A Survey of Evolutionary Algorithms for Large Scale Optimization Problem

LIANG Jing[1], LIU Rui[1], QU Boyang[2], YUE Caitong[1]

(1. School of Electrical Engineering, Zhengzhou University, Zhengzhou 450001, China; 2. School of Electric and Information Engineering, Zhongyuan University of Technology, Zhengzhou 450007, China)

Abstract: Based on the characteristics of large-scale problems, large-scale optimization algorithms were grossly analyzed. This paper introduced some methods for large-scale problems. The methods included the initialization method, decomposition strategy, updating strategy and so on. This paper mainly focued on the search strategy, update strategy, mutation strategy and cooperative coevolution. Meanwhile, the characteristics of large-scale optimization algorithm testing function set and evaluation method were listed. Finally, the future research directions were given.

Key words: large scale optimization problem; evolutionary algorithm; cooperative coevolution; population initialization; benchmark function