多目标细菌觅食优化算法*

李 珺, 党建武, 王 垚, 包 敏
(兰州交通大学 电子与信息工程学院, 兰州 730070)

摘 要: 传统的细菌觅食优化算法仅针对单目标优化问题寻优,为进一步发掘细菌群体智能在多目标优化问题中的寻优优势,提出了改进的多目标细菌觅食优化算法。在个体间互不支配时给出归一化的择优策略;引入差分思想完成复制操作,提高种群的多样性;采用栅格划分法进行迁徙操作,提高解集的分散性;同时使用外部集存放当前找到的非支配解,并不断对外部集进行优化。通过对多个标准函数进行测试并与其他几种算法的对比结果表明,所提出的多目标细菌觅食优化算法在解的收敛性和分散性指标上都有一定提升,能够有效解决多目标优化问题。

关键词: 多目标优化问题; 细菌觅食优化算法; 归一化; 差分进化; 外部集; 栅格
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2018)07-1996-05
doi:10.3969/j.issn.1001-3695.2018.07.017

Multi-objective bacteria foraging optimization algorithm

Li Jun, Dang Jianwu, Wang Yao, Bao Min
(*School of Electronic & Information Engineering, Lanzhou Jiaotong University, Lanzhou* 730070, *China*)

Abstract: Conventional bacterial foraging optimization algorithm simply optimizes the single target optimization problems. In order to exploit the further strengths of bacterial colony in multiple target optimization, this paper proposed the improved multiple target bacterial foraging algorithm. It put forward the optimization strategy via normalization method when individuals had no inter-dominance. It increased population diversity at maximum with the introduction of difference in the completion of replication. It enhanced the solution set dispersiblity with the assistance of grid portioning method in the targeted migration operation. Simultaneously, it put the found non-dominant solution at present in the external data set and continuously optimized the non-dominant solution set in the external data set applying the given update strategy. The outcome of comparison between several other algorithm and the test of numerous standard function manifest that, the proposed multiple target bacterial foraging algorithm raises both astringency and dispersiblity of solution, which can address multiple target optimization problem.

Key words: multi-objective optimization problems(MOP); bacterial foraging optimization algorithm(BFO); normalization; differential evolution(DE); external data set; grid

1 多目标优化问题

在大型工程和科学研究中,优化问题一直都是研究的热点。在优化问题的求解中,如果仅需要优化一个目标函数,此时为单目标优化问题;当需要同时优化多个目标函数时,这种最优化问题即为多目标优化问题(multi-objective optimization problems, MOP)[1]。

多目标优化问题一般由 n 个决策变量、m 个目标函数和多个约束条件组成,模型如下:

$$\begin{aligned}&\min\ \boldsymbol{y}=F(\boldsymbol{x})=[f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_m(\boldsymbol{x})]\\&\text{s.t.}\quad g_i(\boldsymbol{x})\leqslant 0\quad i=1,2,\cdots,q\\&\qquad h_j(\boldsymbol{x})=0\quad j=1,2,\cdots,p\end{aligned}\tag{1}$$

其中:$\boldsymbol{y}=F(\boldsymbol{x})$ 为优化目标,它包含 m 个目标函数;$\boldsymbol{y}=(y_1,y_2,\cdots,y_m)\in Y\subset \mathrm{R}^m$ 表示 m 维的目标向量;$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)\in X\subset \mathrm{R}^n$ 表示 n 维的决策向量,X 表示 n 维的决策空间;$g_i(\boldsymbol{x})\leqslant 0$ $(i=1,2,\cdots,q)$ 定义了 q 个不等式约束;$h_j(\boldsymbol{x})=0$ $(j=1,2,\cdots,p)$ 定义了 p 个等式约束。在此模型的基础上,给出以下几个定义[2]。

定义 1 可行解。对于某个 $\boldsymbol{x}\in X$,如果 $\boldsymbol{x}$ 满足式(1)中的约束条件 $g_i(\boldsymbol{x})\leqslant 0(i=1,2,\cdots,q)$ 和 $h_j(\boldsymbol{x})=0(j=1,2,\cdots,p)$ 则称 $\boldsymbol{x}$ 为可行解。

定义 2 可行解集合。由 X 中所有可行解组成的集合称为可行解集合,记为 X_f,且 $X_f\subseteq X$。

定义 3 Pareto 占优。假设 $\boldsymbol{x}_A,\boldsymbol{x}_B\in X_f$ 是式(1)所表示的多目标优化问题的两个可行解,则称与 $\boldsymbol{x}_B$ 相比,$\boldsymbol{x}_A$ 是 Pareto 占优的,当且仅当

$$\begin{aligned}&\forall i=1,2,\cdots,m\quad f_i(\boldsymbol{x}_A)\leqslant f_i(\boldsymbol{x}_B)\wedge\\&\exists j=1,2,\cdots,m\quad f_j(\boldsymbol{x}_A)<f_j(\boldsymbol{x}_B)\end{aligned}\tag{2}$$

记做 $\boldsymbol{x}_A\succ\boldsymbol{x}_B$,也称为 $\boldsymbol{x}_A$ 支配 $\boldsymbol{x}_B$。

定义 4 Pareto 最优解。一个解 $\boldsymbol{x}^*\in X_f$ 被称为 Pareto 最优解(或非支配解),当且仅当它满足如下条件:

$$\neg\,\exists\,\boldsymbol{x}\in X_f:\boldsymbol{x}\succ\boldsymbol{x}^*\tag{3}$$

定义 5 Pareto 最优解集。Pareto 最优解集是所有 Pareto 最优解的集合,其定义如下:

$$P^*\triangleq\{\boldsymbol{x}^*\mid\neg\,\exists\,x\in X_f:\boldsymbol{x}\succ\boldsymbol{x}^*\}\tag{4}$$

定义 6 Pareto 前沿面。Pareto 最优解集 P^* 中所有 Pareto 最优解所对应的目标矢量组成的曲面称为 Pareto 前沿面 PF^*:

$$PF^*\triangleq\{F(\boldsymbol{x}^*)=(f_1(\boldsymbol{x}^*),f_2(\boldsymbol{x}^*),\cdots,f_m(\boldsymbol{x}^*))^{\mathrm{T}}\mid\boldsymbol{x}^*\in P^*\}\tag{5}$$

早期的研究人员求解多目标优化问题是通过加权方式将其转换为单目标问题来完成的,每次只能得到一种权值情况下的最优解。在多目标优化问题的目标函数和约束函数是非线性、不可微或不连续的情况下,这种方法往往效率较低,当权重值或目标给定的次序不同时,所得结果相差很大。

随着各类进化算法的出现,越来越多的学者发现基于进化

收稿日期: 2017-02-14; **修回日期**: 2017-05-24 **基金项目**: 甘肃省科技计划资助项目(1506RJZA084);甘肃省教育厅科研项目(1204-13);甘肃省教育科学"十二五"规划课题(GS[2015]GHB0907);兰州市科技计划资助项目(2015-2-74)

作者简介:李珺(1974-),女,辽宁辽阳人,副教授,博士研究生,主要研究方向为智能计算与智能信息处理、图形图像处理(lijane@mail.lzjtu.cn);党建武(1962-),男,陕西富平人,教授,博导,博士,主要研究方向为人工智能、神经网络;王垚(1991-),男,甘肃天水人,硕士研究生,主要研究方向为智能计算方法.

的寻优方式对求解多目标问题有着得天独厚的优势。Deb 等人[3]提出了著名的多目标遗传算法 NSGA-Ⅱ;Zitzler 等人[4]提出了 the strength Pareto evolutionary algorithm(SPEA2);基于粒子群优化算法,Coello 等人[5]提出了 MOPSO(multi-objective particle swarm optimization)算法;基于免疫算法,Gong 等人[6]提出了 NNIA(nondominated neighbor immune algorithm);基于蚁群优化算法,肖菁等人[7]提出了多目标蚁群算法;将数学规划与进化算法结合,Zhang 等人[8]提出了 MOEA/D(multi-objective evolutionary algorithm based on decomposition)。不断有学者尝试将各类新的进化算法应用于多目标问题的求解中。

2　多目标细菌觅食优化算法

细菌觅食优化算法(bacteria foraging optimization algorithm,BFO)[9]是 Passino 于 2002 年提出的模拟人类大肠杆菌觅食行为的一类群体进化算法,它主要通过趋向性操作、复制操作、迁徙操作和群体感应机制的迭代计算,来模拟细菌个体在觅食过程中不断寻找富养区域的过程,从而寻找问题的最优解。算法一经提出,就引起了国内外学者的广泛关注,有研究显示,算法在寻优过程中收敛速度快、精度高,且能有效地跳出局部最优解[10],已成为群体优化算法中的一个研究热点。

随着多目标问题的研究越来越热,部分研究者也尝试使用细菌觅食优化算法来求解多目标优化问题。王帆[11]提出了基于区域搜索的细菌觅食算法,针对多目标优化问题,对细菌觅食算法进行改进。在趋向性操作中,当细菌个体所找到的新位置与旧位置互不支配时,使用多个目标新旧位置适应度值之差的累加和来判断新旧位置的优劣,这种判断方式并不恰当,多目标问题中的多个目标往往具有不同的量纲,如运输问题中需要运费最小、时间最短,此时将运输费用(元)与时间(h)相加,所得结果毫无意义。岑雪婷[12]使用细菌觅食优化算法解决多目标资源受限项目调度问题,仅在细菌觅食优化算法中引入 Parato 支配概念,通过简单的支配关系来确定解的优劣,并通过给出的编码/解码方案来解决特定的资源受限调度问题,没有给出算法在多目标优化问题上的完整改进方案,方法具有一定的局限性。还有一些细菌觅食算法在各类具体的多目标问题中的应用研究都仅对具体问题给出了算法的一些改进,对于细菌觅食算法,目前并没有一个针对多目标优化问题的全面、完善的改进策略。

标准的细菌觅食优化算法是针对单目标优化问题提出的,在求解多目标优化问题时,需要对其每一步的操作规则作出相应改变。本文针对多目标优化问题,结合差分进化思想,对标准细菌觅食算法进行改进,提出了多目标细菌觅食优化算法(multi-objective bacteria foraging optimization algorithm,MOBFO)。

2.1　多目标趋向性操作

趋向性操作通过细菌个体在整个寻优空间中的随机搜索来寻找问题的最优解,是整个算法中最重要的一个环节。在随机搜索过程中,细菌个体每走出一步,都要比较个体所在新位置和旧位置哪个更优,如果新位置更好,则细菌按照此方向继续向前走,直到新位置不再优于旧位置或者达到了最大尝试次数。

在多目标优化问题中,由于此时有多个目标函数,新旧个体的优劣无法再像单目标优化问题中那样,通过某一函数的适应度值进行比较,因此,这里基于 Pareto 支配关系,给出了新的择优策略。

在新旧个体 $\boldsymbol{x}_{\text{old}}$ 和 $\boldsymbol{x}_{\text{new}}$ 之间,有以下情况:

a)两者互相支配。此时,支配个体优于被支配个体。

b)两者互不支配。由于多目标问题中,不同的目标具有不同的量纲,无法直接对具有不同量纲的目标值作出变化大小的评价,所以,首先对不同目标的适应度值进行归一化处理,根据归一化结果进行取舍。归一化方法如下:

```
begin
for(i = 1;i < = M;i ++)
{
  f_i(total) = f_i(x_new) + f_i(x_old);
  //将新个体与旧个体当中相同目标的函数值相加
  g_i(x_new) = f_i(x_new)/f_i(total), g_i(x_old) = f_i(x_old)/f_i(total);
  //分别计算新旧个体中相同目标的函数值所占的比例 g_i
}
sum = Σ_{i=1}^{M} F_i * (g_i(x_new) - g_i(x_old));
//统计新旧个体在各目标上的比例差值之和
end
```

其中:M 为目标函数个数;F_i 为各目标权值系数,取值为[0,1],且 $\sum_{i=1}^{M} F_i = 1$。用户可以根据所求解问题的特征或需求来调整目标函数的权值系数,在寻优过程中控制不同目标函数值的变化对优化结果的影响,以达到最佳寻优效果。若 sum < 0,则认为新个体更优,否则旧个体更优。通过归一化方法,根据当前目标函数值的收敛比例确定新旧个体哪个收敛性更好,按比例收敛越多的个体越优来保留更优个体。

趋向性操作中,细菌个体每走出一步,按照上述归一化择优策略进行判断,若新位置优于旧位置,则前进到新位置并按照此方向继续前进,直到新位置不再优于旧位置或者达到最大尝试次数。

2.2　多目标复制操作

针对多目标优化,细菌觅食优化算法中的复制操作面临两个问题:a)如何对菌群个体进行排序;b)如何复制产生新个体。

2.2.1　排序操作

标准 BFO 算法中,复制操作首先要对菌群中的个体按适应度值优劣排序,针对多目标优化问题的多个目标函数,每个细菌个体都有多个适应度值,如何排序是要解决的首要问题。

考虑在多目标优化中,根据 Pareto 支配关系,要求的是一组非支配解(非劣解),个体的被支配次数越少,个体越优。因此在排序中,根据 Pareto 支配关系,统计每个个体在当前群体中的被支配次数,依据个体的被支配次数,从小到大完成排序工作。

2.2.2　复制操作

标准 BFO 算法的复制操作简单地使用好的一半个体取代差的一半,能够提高算法的收敛速度,但使种群的多样性折半,算法容易陷入局部最优。在多目标优化问题中,要找的是一组非劣解,这些解不仅要具有良好的收敛性,而且要有较好的分散性,为用户提供更多的选择,因此种群的多样性更加重要。

为了使种群具有更丰富的多样性并充分利用较差一半个体的信息,引入差分进化思想完成复制操作,如式(6)所示。

$$\theta^{\text{new}} = \begin{cases} \theta^i + F \times (\theta^k - \theta^i) + F \times (\theta^l - \theta^i) & \theta^{\text{new}} \text{优于} \theta^i \\ \theta^k & \text{otherwise} \end{cases} \tag{6}$$

其中:F 为缩放因子;θ^i 表示需要进行差分运算的个体,来自种群中较差的一半,θ^k 和 θ^l 是从较好的一半个体中随机选出的两个个体。以式(6)中的差分方式产生新个体,若新个体优于旧个体,则替换旧个体;否则,使用 θ^k 来取代旧个体。

复制操作中,将菌群排序后,对排在后面的被支配个体,依次以差分方式按照式(6)完成复制操作。这种结合差分的复制方式,在提高收敛性的同时充分利用现有个体的信息,最大限度地保持种群的多样性,满足多目标优化的要求。

2.3　多目标迁徙操作

标准细菌觅食优化算法中,迁徙操作的目的是通过随机的迁徙,减小个体陷入局部极值的概率。在多目标优化问题中,不仅要考虑解的收敛性,而且要考虑分散性。这里将迁徙操作与解的分散性结合,采用栅格划分迁徙法提高解的分散性,尽可能找到分布均匀的解。

a)估算某一目标函数适应度值的大致范围,将该范围划分成若干长度相等的栅格;根据菌群的规模,考虑将菌群个体平均分布时,每个栅格中应有的细菌个体数目,从而得到一个

阈值。

b)根据每个细菌个体当前的位置,计算其在此目标函数上的适应度值,统计每一栅格中目前实际存在细菌个体的数目,若此数目超出阈值,则先对该栅格中的所有细菌个体按照归一化择优策略进行排序,选择其中最差的个体,将其迁往稀疏区域。

重复调整,直至菌群中的个体在所有栅格中平均分布为止。

以测试函数 ZDT1(函数原型见表 1)为例,其 Pareto 前沿面示意图如图 1 所示。a)根据测试函数 ZDT1 的目标函数 $f_1(x)$的范围($f_1(x)\in[0,1]$),将其在 $f_1(x)$目标值上划分为 10 等份,假设种群规模为 100,则在解均匀分布时,每个等份中存在的个体数目不应该大于 10,因此这里确定阈值为 10;b)根据细菌个体当前在 $f_1(x)$上的适应度值,统计每一等份中包含细菌个体的数目,当某一等份中细菌个体数目超过上述阈值时,选择其中最差个体迁徙。此时在最稀疏的区域中随机生成此个体 $f_1(x)$的值,其余维度上的值可在搜索空间中随机生成。这样迁徙操作不再是随机选择迁徙个体,而是根据所求得个体的分散程度更有目的地进行迁徙,将迁徙操作与解的分散性相结合,最大限度提高解的质量。

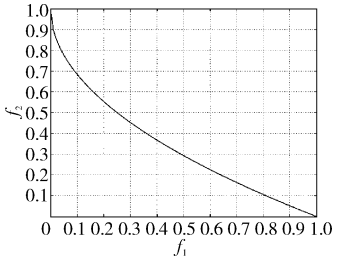

图 1 函数 ZDT1 Pareto 前沿面示意

2.4 外部集存放策略

为了更好地维护种群的多样性,有效保存寻优过程中找到的非劣解,在 MOBFO 中引入外部集,用来存储在寻优过程中找到的非劣解。经过一个周期的趋向性、复制、迁徙操作后,如果产生了非劣解,每个非劣解能否放入外部集需要进行以下判断:

a)将待存入非劣解与外部集中的个体依次比较,看外部集中是否已存在相同个体,若已有相同个体,则不需将此非劣解存入,判断下一个非劣解。若外部集中无相同非劣解存在,则转入步骤 b)。

b)比较此非劣解与外部集中已有非劣解的支配关系,若此非劣解支配外部集中的解,则用此非劣解替换外部集中所有被其支配的解,转入步骤 a),继续判断下一个待存入的非劣解;若此解被外部集中的某一非劣解支配,则抛弃此解,转入步骤 a);若两者互不支配,则转入步骤 c)。

c)判断外部集是否已满,如果外部集未满,直接将此非劣解加入外部集,转入步骤 a);若外部集已满,将此解预存入外部集中,按 2.3 节所述栅格法统计外部集中的解在每区域中分布的数目,选择密度最大的区域,对解进行归一化排序,将最差解从外部集中淘汰。转入步骤 a)继续判断待存入解,直到所有解判断完毕。

当寻优结束时,外部集中则保存了所需要的最优解集。

2.5 MOBFO 算法流程

MOBFO 算法整体流程如下:

a)初始化算法参数。S(种群规模)、N_c(趋向性操作次数)、N_s(同一个方向上泳动的最大步数)、N_{re}(复制操作次数)、N_{ed}(迁徙操作次数)、F(缩放因子)、F_i($i=1,2,\cdots,M$,F_i为各目标函数的权值系数)、out[S](外部集数组,数组规模与群体大小相同,初始化为全 0,数组中每个成员设有标记位,初始标记均为 0,表示此时没有有效数据,每存入一个有效数据,则该成员标记为 1)、generation(进化代数)=0,随机生成 S 个细菌个体的初始位置。

b)寻优过程。

```
for(l=0;l<N_ed;l++)   //迁徙循环开始
{
  for(k=0;k<N_re;k++)   //复制循环开始
  {
    for(j=0; j<N_c; j++)   //趋向性循环开始
    {
      for(i=0;i<S;i++)
        个体依据 2.1 节择优方法更新位置;
        采用文献[13]所述自适应分维步长机制;
    } //趋向性循环结束
    依据 2.2.1 节所述方法完成菌群排序;
    依据 2.2.2 节所述方法完成个体复制操作;
    依据 2.4 节所述方法完成外部集更新;
  } //复制循环结束
  if(l==N_ed-2) break;
  //最后一次循环不再迁徙
  依据 2.3 节所述完成迁徙操作;
} //迁徙循环结束
```

c)输出外部集中保存的最优解,算法结束。

3 算法性能测试

3.1 算法性能评价指标

不同于单目标优化问题的性能评价方法,多目标问题性能评价方法本身就是一个复杂的问题,通常需要考虑非支配解集与 Pareto 前沿面的距离、非支配解集的分布情况以及非支配解集对 Pareto 前沿的覆盖程度。这里选取两个最常用的评价指标对算法进行评价。

1)收敛性指标

采用世代距离(generation distance,GD)来衡量算法的收敛性。计算公式如下:

$$GD=\frac{1}{n_{pf}}\left(\sum_{i=1}^{n_{pf}}d_i^2\right)^{1/2}$$

$$d_i=\min\sqrt{\sum_{k=1}^{M}(f_k^i-f_k^j)^2}\quad j=1,2,\cdots,n \tag{7}$$

其中:n_{pf}为算法搜索到的非支配解个数;n 为已知 Pareto 前沿的非支配解个数;M 为目标函数个数;f_k^i为第 k 个目标的第 i 个个体适应度值;f_k^j为真实前沿面上第 j 个点的第 k 个目标函数值。GD 越小,说明算法的收敛性越好。

2)间距性指标

采用 SP(spacing metric)来衡量算法所找到的非支配解分布的均匀性,公式如下:

$$SP=\left(\frac{1}{n_{pf}-1}\sum_{i=1}^{n_{pf}}(\bar{d}-d_i)^2\right)^{1/2}$$

$$d_i=\min_j\left(\sum_{k=1}^{M}|f_k^i(x)-f_k^j(x)|\right)$$

$$i,j=1,2,\cdots,n_{pf};i\neq j \tag{8}$$

其中:n_{pf}为算法搜索到的非支配解个数;M 为目标函数个数;$\bar{d}$为所有 d_i 的平均值。SP 的值越小,表明所求得解集的均匀性越好。

3.2 实验分析

表 1 给出了实验中用到的 12 个测试函数,ZDT 问题由 Zitzler 等人[14]在 2000 年提出,KUR[15]和 FON(Fonseca and Fleming 1995a)分别由 Kursawe 和 Fonseca 提出,均为两目标优化问题;DTLZ 系列是典型的高维多目标测试函数[16],这里选择三个优化目标。

为了验证 MOBFO 算法的有效性,将其与文献[17]中的数据进行比较。文献[17]提出了基于多发现者和交叉算子的多目标群搜索算法(MCGSO),该算法是在已有的基于单发现者的群搜索算法中引入 Metropolis 准则改进发现者更新策略,并引入遗传算法中的交叉算子改进游荡者更新策略,同时参考 NSGA-Ⅱ中的精英保留策略避免丢失最优解,算法在收敛性和分散性上都取得了一定的效果;同时引用该文献中 NSGA-Ⅱ和 MGSO 的数据,首先在典型的多目标测试函数 ZDT 系列和另外两个有代表性的测试函数 FON、KUR 上进行测试。实验在

Windows 10 操作系统下使用 Visual Studio 2012 平台,利用 VC ++语言编程完成。MOBFO 中群体规模 $S=100$,$N_c=100$,$N_{re}=60$,$N_{ed}=5$,差分思想完成复制操作时,缩放因子 $F=0.8$,每个函数独立运行 30 次,所得 GD 及 SP 结果的均值如表 2 所示。

表 1　测试函数说明

problems	dimension n	variable domain	objective functions (minimized)
ZDT1	30	$[0,1]$	$f_1(x)=x_1$; $f_2(x)=g(x)[1-\sqrt{x_1/g(x)}]$; $g(x)=1+9(\sum_{i=2}^{D}x_i)/(D-1)$
ZDT2	30	$[0,1]$	$f_1(x)=x_1$; $f_2(x)=g(x)[1-(x_1/g(x))^2]$; $g(x)=1+9(\sum_{i=2}^{D}x_i)/(D-1)$
ZDT3	30	$[0,1]$	$f_1(x)=x_1$; $f_2(x)=g(x)[1-\sqrt{x_1/g(x)}-x_1/g(x)\sin(10\pi x_1)]$; $g(x)=1+9(\sum_{i=2}^{D}x_i)/(D-1)$
ZDT6	10	$x_i\in[0,1]$ $i=2,\cdots,D$	$f_1(x)=1-\exp(-4x_1)$; $f_2(x)=g(x)[1-(f_1/g(x))^2]$; $g(x)=1+9\sum_{i=2}^{D}[x_i/(D-1)^{1/4}]$
KUR	3	$[-5,5]$	$f_1(x)=\sum_{i=1}^{D-1}(-10\exp(-0.2\sqrt{x_{i2}+x_{i+1}^2}))$; $f_2(x)=\sum_{i=1}^{D}(\lvert x_i\rvert^{0.8}+5\sin(x_i)^3)$
FON	3	$[-4,4]$	$f_1(x)=1-\exp(-\sum_{i=1}^{D}(x_i-\frac{1}{\sqrt{n}})^2)$; $f_2(x)=1-\exp(-\sum_{i=1}^{D}(x_i+\frac{1}{\sqrt{n}})^2)$
DTLZ1	7	$[0,1]$	$f_1(x)=x_1x_2(1+g(x))/2$; $f_2(x)=x_1(1-x_2)\times(1+g(x))/2$; $f_3(x)=(1-x_1)(1+g(x))/2$; $g(x)=100(\lvert x\rvert+\sum_{x_i\in x}(x_i-0.5)^2-\cos(20\pi\times(x_i-0.5)))$
DTLZ2	12	$[0,1]$	$f_1(x)=\cos(x_1\pi/2)\cos(x_2\pi/2)(1+g(x))$; $f_2(x)=\cos(x_1\pi/2)\sin(x_2\pi/2)(1+g(x))$; $f_3(x)=\sin(x_1\pi/2)(1+g(x))$; $g(x)=\sum_{x_i\in x}(x_i-0.5)^2$
DTLZ3	12	$[0,1]$	$f_1(x)=(1+g(x))\cos(x_1\pi/2)\cos(x_2\pi/2)$; $f_2(x)=(1+g(x))\cos(x_1\pi/2)\sin(x_2\pi/2)$; $f_3(x)=(1+g(x))\sin(x_1\pi/2)$; $g(x)=100[\lvert x\rvert+\sum_{x_i\in x}((x_i-0.5)^2-\cos(20\pi\times(x_i-0.5)))]$
DTLZ4	12	$[0,1]$	$f_1(x)=(1+g(x))\cos(x_1^{\alpha}\pi/2)\cos(x_2^{\alpha}\pi/2)$; $f_2(x)=(1+g(x))\cos(x_1^{\alpha}\pi/2)\sin(x_2^{\alpha}\pi/2)$; $f_3(x)+(1+g(x))\sin(x_1^{\alpha}\pi/2)$; $g(x)=\sum_{x_i\in x}(x_i-0.5)^2$; $\alpha=100$
DTLZ5	12	$[0,1]$	$f_1(x)=(1+g(x))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)$; $f_2(x)=(1+g(x))\cos(\theta_1\pi/2)\sin(\theta_2\pi/2)$; $f_3(x)=(1+g(x))\sin(\theta_1\pi/2)$; $\theta_i=\pi(1+2g(x)\times x_2)/4(1+g(x))$; $g(x)=\sum_{x_i\in x}(x_i-0.5)^2$
DTLZ6	12	$[0,1]$	$f_1(x)=x_1$; $f_2(x)=x_2$; $f_3(x)=(1+g(x))\times h(f_1,f_2,g)$; $g(x)=1+(9/\lvert x\rvert)\sum_{x_i\in x}x_i$; $h(f_1,f_2,g)=3-\sum_{i=1}^{3}(f_i/(1+g)(1+\sin(3\pi f_i)))$

从表 2 中可以看出,本文算法在求解以上四个函数时,仅对非连续函数 ZDT3 和 KUR 的评价指标不如其他三种算法,其余指标均优于其他对比算法。

同时,将以上 ZDT 系列测试函数的测试结果在 GD 和 SP 两个指标上做盒图统计,并与 PESA Ⅱ、NSGA-Ⅱ 以及 SPEA2 三种经典的多目标优化算法进行对比,对比数据来自文献[2],对比结果如图 2、3 所示。从以上统计盒图可以看出,在收敛性指标上,MOBFO 算法优于其余三种对比算法;在分散性指标上,仅对 ZDT3 函数的求解结果略差,对其余三个测试函数,本文 MOBFO 算法均优于对比算法。

表 2　优化算法对 ZDT 系列函数的优化性能指标

problems		MCGSO	MGSO	NAGA-Ⅱ	MOBFO
ZDT1	GD	0.000 9	0.013 1	0.041 6	0.000 313
	SP	0.005 7	0.006 6	0.006 1	0.002 073
ZDT2	GD	0.000 7	0.022 1	0.063 4	0.000 547
	SP	0.005 7	0.006 6	0.005 7	0.003 636
ZDT3	GD	0.002 4	0.006 2	0.025 7	0.000 645
	SP	0.005 9	0.007 6	0.004 8	0.012 545
ZDT6	GD	0.000 9	2.620 5	0.001 0	0.000 315
	SP	0.005 7	0.035 2	0.006 3	0.002 654
KUR	GD	0.001 5	0.004 0	0.003 5	0.013 8
	SP	0.047 8	0.082 8	0.060 0	0.148 0
FON	GD	0.009 8	0.060 0	0.010 5	0.001
	SP	0.005 0	0.006 1	0.004 5	0.004 5

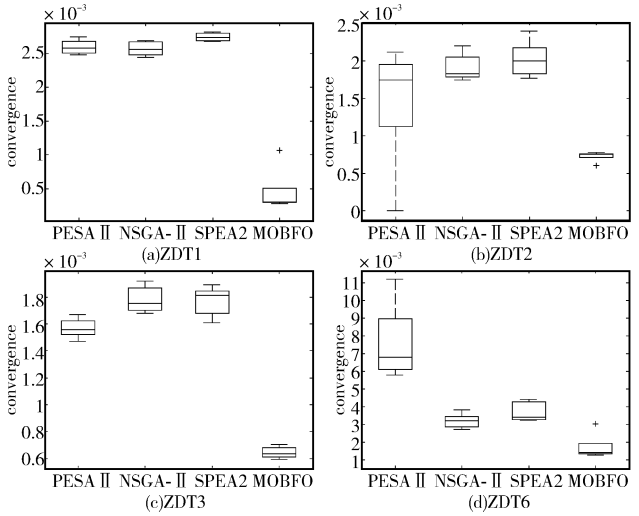

图 2　四种算法求解四个测试问题的收敛性指标统计盒

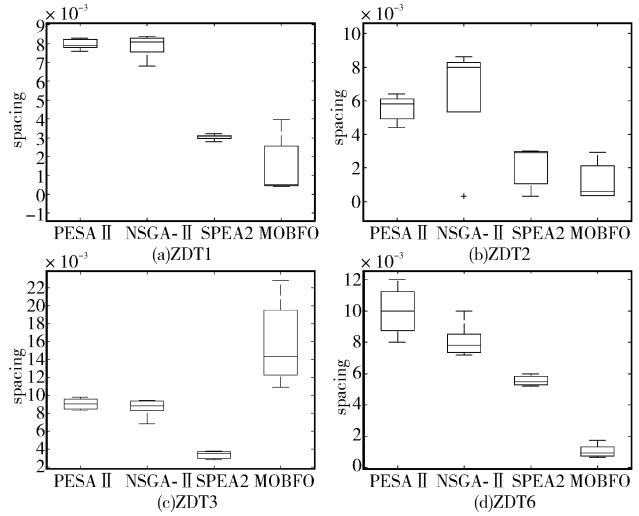

图 3　四种算法求解四个测试问题的分散性指标统计盒

在相同的实验环境下,继续对 DTLZ 系列函数进行测试,仍旧与文献[17]中的测试结果进行对比,结果如表 3、4 所示。

表 3　优化算法在 DTLZ 系列测试函数下的 GD 性能指标

algorithms	results	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6
CMOGSO	mean	0.024 1	0.011 3	0.873 6	0.021 0	0.008	0.004 5
	variance	0.072 3	0.002 1	1.160 6	0.002	0.001 5	0.000 4
MGSO	mean	–	0.338 5	0.343 4	0.334 3	0.093 1	0.372 5
	variance	–	0.000 3	0.110 6	0.079 5	0.047 8	0.704
NSGA-Ⅱ	mean	0.495 7	0.012 6	12.673 5	0.021 3	0.006	0.004 8
	variance	1.009 2	0.001 1	6.184 2	0.001 1	0.000 3	0.000 4
MOBFO	mean	0.016 2	0.010 8	0.866 5	0.020 3	0.006 9	0.001 6
	variance	0.063 3	0.002 1	0.992 6	0.001 6	0.010 6	0.002 4

表 4　优化算法在 DTLZ 系列测试函数下的 SP 性能指标

algorithms	results	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6
CMOGSO	mean	0.011 4	0.025 6	0.374	0.025 6	0.007 4	0.004 9
	variance	0.003 5	0.002 5	0.691 20	0.002 5	0.001	0.000 7
MGSO	mean	–	0.092	0.095 4	0.090 9	0.021 2	0.049 7
	variance	–	0.009 7	0.023 1	0.010 9	0.007	0.108 9
NSGA-Ⅱ	mean	0.621	0.058 5	8.041 6	0.056 5	0.009 4	0.012 6
	variance	3.029 5	0.004 9	10.773 2	0.004 1	0.001 0	0.000 7
MOBFO	mean	0.006 2	0.015 4	0.655 4	0.011 1	0.005 9	0.004 6
	variance	0.002 8	0.002 3	0.952 6	0.001 6	0.000 8	0.000 7

从表3、4的实验数据可以看出,对以上六个三目标测试函数,本文算法仅对DTLZ3的寻优效果略差于对比算法,对其余测试函数的寻优均值都优于对比算法。

综合以上图表的实验数据可以看出,在两目标测试函数中,本文算法仅对非连续函数ZDT3和KUR的SP指标略差,这与所求函数前沿面的连续性有关,在不连续的前沿面上,点与点之间的距离本就不均匀,因此会导致计算出的SP指标增大,对两目标测试函数的其余指标都优于对比算法;在所有的三目标优化测试函数中,本文算法仅对DTLZ3函数优化的指标GD均值弱于MGSO,而优于其余两种对比算法,SP值弱于CMOGSO和MGSO,而优于NSGA-Ⅱ,对其余函数的优化指标均优于对比算法。

考虑算法的复杂度,对比文献[2,17]中的算法,初始寻优种群规模均为100,存放最优解集的空间大小也为100,与本文中种群规模相同。对比文献[2]算法,函数评价次数为50 000,本文改进算法对函数的评价次数为30 000,时间复杂度上优于文献[2];文献[17]中,函数评价次数与本文相同。综合考虑算法的时间和空间复杂度,本文算法优于文献[2],与文献[17]相同。

综上所述,本文算法求解多目标优化问题可以得到较好的收敛性指标,对于分散性指标,除开连续的ZDT3和KUR函数外,比其他对比算法有明显优势。

4 结束语

对传统的单目标细菌觅食优化算法进行改进,设计了针对多目标优化问题的多目标细菌觅食优化算法MOBFO。针对多目标优化问题的特征,对传统单目标细菌算法的每一步操作进行调整。a)在寻优过程中得到两个解互不支配而无法判定优劣时,给出了归一化的择优策略,解决了多目标优化中解的优劣判定问题;b)在复制操作中引入差分思想,通过个体间的差分运算来完成复制,最大限度地提高种群的多样性;c)采用栅格划分迁徙法,将迁徙操作与解集的分散性结合,有目的将解迁徙到稀疏区域,取代了传统的随机迁徙过程,提高了解集的分散性;d)使用外部集来存储找到的非劣解,并设计了完整的外部集存放策略,使外部集中存储的非支配解集不断优化。实验证明,本文设计的MOBFO算法具有良好的性能,能够有效解决多目标优化问题。

参考文献:

[1] Deb K. Multi-objective optimization using evolutionary algorithms [M]. Hoboken: Wiley, 2001.

[2] 公茂果, 焦李成, 杨咚咚, 等. 进化多目标优化算法研究 [J]. 软件学报, 2009, 20(3): 272-289.

[3] Deb K, Pratap A, Agarwal S, *et al.* A fast and elitist multi-objective genetic algorithms: NSGA-Ⅱ [J]. IEEE Trans on Evolutionary Computation, 2002, 6(2): 182-197.

[4] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm, TIK-report 103 [R]. Zurich: Swiss Federal Institute of Technology, 2001: 126-140.

[5] Coello C A, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization [J]. IEEE Trans on Evolutionary Computations, 2004, 8(3): 256-279.

[6] Gong Maoguo, Jiao Licheng, Du Haifeng, *et al.* Multi objective immune algorithm with nondominated neighbor-based selection [J]. Evolutionary Computation, 2008, 16(2): 225-255.

[7] 肖菁, 陈凤莲, 汤建超. 基于蚁群算法的多目标优化技术研究 [J]. 华南师范大学学报: 自然科学版, 2014, 46(1): 1-6.

[8] Zhang Qingfu, Li Hui. MOEA/D: a multiobjective evolutionary algorithm based on decomposition [J]. IEEE Trans on Evolutionary Computation, 2007, 11(6): 712-731.

[9] Passino K M. Biomimicry of bacterial foraging for distributed optimization and control [J]. IEEE Control Systems, 2002, 22(3): 52-67.

[10] 周雅兰. 细菌觅食优化算法的研究与应用 [J]. 计算机工程与应用, 2010, 46(20): 16-21.

[11] 王帆. 面向高维及多目标优化的协同细菌觅食算法研究 [D]. 大连: 大连理工大学, 2013.

[12] 岑雪婷. 基于细菌觅食优化算法的多目标资源受限项目调度问题研究 [D]. 广州: 华南理工大学, 2013.

[13] 李珺, 党建武. 细菌觅食优化算法求解高维优化问题 [J]. 计算机应用研究, 2016, 33(4): 1024-1027, 1033.

[14] Zitzler E, Deb K, Thiele L. Comparison of multi-objective evolutionary algorithms: empirical results [J]. Evolutionary Computation, 2000, 8(2): 173-195.

[15] Kursawe F. A variant of evolution strategies for vector optimization [C]//Proc of International Conference on Parallel Problem Solving from Nature. Berlin: Springer-Verlag, 1991: 193-197.

[16] Deb K, Thiele L, Laumanns M, et al. Scalable multi-objective optimization test problems [C]//Proc of Congress on Evolutionary Computation. Washington DC:IEEE Computer Society,2002: 825-830.

[17] 李亚洲. 多目标群搜索算法研究及其应用[D]. 济南: 山东师范大学, 2016.

(上接第1991页)

[6] Wang Weina, Ying Lei. Data locality in MapReduce: a network perspective[J]. Performance Evaluation, 2016, 96(2): 1-11.

[7] Chen Quan, Zhang Daqiang, Guo Minyi, *et al.* SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment [C]//Proc of the 10th IEEE International Conference on Computer and Information Technology. Washing DC: IEEE Computer Society, 2010: 2736-2743.

[8] Wang Bo, Jiang Jinlei, Yang Guangwen. ActCap: accelerating MapReduce on heterogeneous clusters with capability-aware data placement[C]// Proc of IEEE Conference on Computer Communications. 2015: 1328-1336.

[9] Fujishima E, Yamaguchi S. Dynamic file placing control for improving the I/O performance in the reduce phase of Hadoop[C]//Proc of the 10th International Conference on Ubiquitous Information Management and Communication. New York: ACM Press, 2016: 1-7.

[10] Hammoud M, Sakr M F. Locality-aware reduce task scheduling for MapReduce[C]// Proc of 10 3rd IEEE International Conference on Cloud Computing Technology and Science. Washington DC: IEEE Computer Society, 2011: 570-576.

[11] Arslan E, Shekhar M, Kosar T. Locality and network-aware reduce task scheduling for data-intensive applications[C]//Proc of the 5th International Workshop on Data-Intensive Computing in the Clouds. Piscataway, NJ: IEEE Press, 2014: 17-24.

[12] 董西成. Hadoop技术内幕: 深入解析MapReduce架构设计与实现原理(Hadoop internals: in-depth study of mapreduce)[M]. 北京: 机械工业出版社, 2013.

[13] Kao Y C, Chen Yashu. Data-locality-aware mapreduce real-time scheduling framework [J]. Journal of Systems and Software, 2016, 112: 65-77.

[14] Yang S J, Chen Y R. Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds[J]. Journal of Network and Computer Applications, 2015, 57(11): 61-70.

[15] Zaharia M, Konwinski A, Joseph A D, *et al.* Improving MapReduce performance in heterogeneous environments[C]// Proc of USENIX Conference on Operating Systems Design and Implementation. [S. l.]: USENIX Association, 2008: 29-42.

[16] Wang Wenzhu, Wu Qingbo, Tan Yusong, *et al.* Optimizing the MapReduce framework for CPU-MIC heterogeneous cluster[M]//Proc of the 11th International Symposium on Advanced Parallel Processing Technologies. Berlin: Springer International Publishing, 2015: 33-44.