ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, Supplement, December 2007, pp.109−119

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于合作型协同和ε-占优的多目标微粒群算法*

郑向伟+, 刘 弘

(山东师范大学 信息科学与工程学院,山东 济南 250014)

A Cooperative Coevolutionary and ε-Dominance Based MOPSO

ZHENG Xiang-Wei+, LIU Hong

(School of Information Science and Engineering, Shandong Normal University, Ji'nan 250014, China)

+ Corresponding author: +86-531-86684378, Fax: +86-531-86180514, E-mail: sdnuzxw@126.com

Zheng XW, Liu H. A cooperative coevolutionary and ε-dominance based MOPSO. *Journal of Software*, 2007,18(Suppl.):109−119. http://www.jos.org.cn/1000-9825/18/s109.htm

Abstract: Particle Swarm Optimizers (PSOs) have been applied to solve Multi-Objective Optimization Problems (MOPs) for its successful applications in solving single objective optimization problems and are named as Multi-Objective PSOs (MOPSOs). However, MOPSOs are often trapped in local optima, cost more function evaluations and suffer from the curse of dimensionality. A cooperative coevolutionary and ε-dominance based MOPSO (CEPSO) is proposed to attack the above disadvantages. In CEPSO, the MOPs are decomposed according to their decision variables and are optimized by corresponding subswarms respectively. Uniform distribution mutation operator is adopted to avoid premature convergence. All subswarms share one archive based on ε-dominance, which is also used as leader set. Collaborators are selected randomly from archive and used to construct context vector in order to evaluate particles in subswarm. CEPSO is tested on several classical MOP benchmark functions and the simulation results show that CEPSO can escape from local optima, optimize high dimension problems and generate more Pareto solutions. Therefore, CEPSO is competitive in solving MOPs.

Key words: multi-objective optimization; evolutionary algorithm; cooperative coevolution; ε-dominance; MOPSO (multi-objective particle swarm optimizer)

摘 要: 在求解多目标优化问题时,微粒群优化算法有容易陷于局部极值、函数评价次数多和受到维数限制等不足之处.提出了一种基于合作型协同和ε-占优的多目标微粒群算法(cooperative coevolutionary and ε-dominance based multi-objective particle swarm optimizer,简称 CEPSO).依据决策变量分解问题,采用多个子群分别优化各个子问题,并在更新粒子位置时采用均匀分布变异算子防止微粒群早熟收敛;在保存非劣解时,使用ε-占优的档案更新策略,所有子群共享同一外部档案,并将其作为微粒的领导集合;随机从领导集合中选择协同粒子构造完整的解向量,然后基于它们来评价子群中的粒子.在几个经典多目标测试问题上进行了仿真实验,结果表明,CEPSO 算法能够摆脱局部极值,求解较高维优化问题,同时能够保持多样性和快速收敛,可以求得更多分布均匀的 Pareto 解.

关键词: 多目标优化;进化算法;合作型协同进化;ε-占优;多目标微粒群算法

* Supported by the National Natural Science Foundation of China under Grant No.60374054 (国家自然科学基金); the Natural Science Foundation of Shandong Province of China under Grant Nos.Y2003G01, Z2004G02 (山东省自然科学基金)

Received 2007-09-15; Accepted 2007-11-25

工程实践和科学研究中的优化问题大多是多目标优化问题(MOP),各目标之间通过决策变量相互制约,对其中一个目标优化必须以其他目标作为代价,而且各目标的单位又往往不一致,因此很难客观地评价多目标问题解的优劣性.与单目标优化问题的本质区别在于,多目标优化问题的解不是唯一的,而是存在一个最优解集合,集合中元素称为 Pareto 最优解或非劣解.自 1950 年以来,运筹学研究人员已经提出了许多方法解决 MOPs,如多目标加权法、分层序列法、约束法、目标规划法等.然而,传统数学规划方法存在一些缺陷,例如,有些方法对 Pareto 前沿比较敏感,当 Pareto 前沿是凹的或者不连续时,则无法使用这些方法;有些方法要求目标函数和约束条件可微;现有方法每次运行只产生一个解,求多个解时需要运行多次,效率较低[1].而自 1975 年 John Holland 提出遗传算法(GA)以来,基于生物模拟的进化算法得到了深入研究,由于其具有并行、无须求导或其他辅助知识、一次产生多个解和简单易于实现等优点,被视为求解 MOP 的有效方法,研究人员已经提出了多种用于求解多目标问题的进化算法(MOEA)[2].

微粒群优化算法(particle swarm optimizer,简称 PSO)是一种新的进化算法,它模拟了一群寻找食物的鸟群运动,通过群体中粒子间的协作与竞争产生的群体智能进行优化搜索[3].PSO 算法的相对简单性和基于种群的特点使其成为求解多目标优化的有效方法,1999 年,Moore 和 Chapman 首先在他们未公开发表的文稿中提出扩展 PSO 求解 MOP[4],称为多目标微粒群算法(multi-objective PSO,简称 MOPSO),此后,许多研究人员对这种方法产生了极大兴趣,并开始研究,但直到 2002 年才出现了第 2 种 MOPSO 算法.在目前的专业文献中,已经有 25 种 MOPSO 算法之多[5].但是,现有的多目标微粒群算法存在容易陷于局部极值、函数评价次数多和受到维数限制等问题.近年来,协同进化方法被视为求解复杂问题的有效方法,已被成功用于 GA,建立了多种求解单目标和多目标优化问题的协同 GA 算法,并取得良好的效果[6,7].目前已有基于协同进化建立协同 PSO 算法的工作,但从相关的文献来看,均为求解单目标优化问题[8−10].为此,本文采用协同进化机制,研究 PSO 求解多目标优化问题,提出了一种基于合作型协同和ε-占优的多目标微粒群算法(cooperative coevolutionary and ε-dominance based multi-objective particle swarm optimizer,简称 CEPSO).

1 多目标优化问题的数学描述

多目标优化问题是指具有两个或者两个以上目标需要同时优化的问题,且多个目标相互制约,有时还存在约束.MOP 的解通常不止一个,而是多个,是一个解的集合,求解 MOP 常求解问题的 Pareto 最优解集.多目标优化问题的数学描述如下:

$$\begin{cases}\text{Min } f(x)=[f_1(x),f_2(x),...,f_k(x)]\\ \text{s.t. } g_j(x)\ge 0, i=1,2,...,h, x\in R^D\end{cases} \tag{1}$$

其中,$f_i(1\le i\le k)$为目标函数,g_j是约束,问题具有 D 个决策变量,k 个目标函数和 h 种约束.当只有单个目标时,最优解就是在给定约束条件下使目标函数值最小的解,当多个目标要同时最优时,最优解就是 Pareto 最优集.下面给出 MOP 中用到的几个基本概念.

定义 1 (Pareto 占优(Pareto dominance)). 解 x_0 占优 x_1 ($x_0 \prec x_1$), 当且仅当

$$\begin{cases} f_i(x_0)\le f_i(x_1), & \forall i=1,2,...,k\\ f_i(x_0)< f_i(x_1), & \exists i=1,2,...,k\end{cases} \tag{2}$$

定义 2 (Pareto 最优解或 Pareto 非劣解(Pareto optimal)). 如果解 x_0 是 Pareto 最优解当且仅当 $\neg\exists x_1 : x_1 \prec x_0$.

定义 3 (Pareto 最优解集(Pareto optimal set)). 所有 Pareto 最优解的集合:

$$P_s=\{x_0 \mid \neg\exists x_1 \prec x_0\} \tag{3}$$

定义 4 (Pareto 前沿或均衡面(Pareto front)). 所有 Pareto 最优解对应的目标函数值所形成的区域,表示为

$$Pf=\{f(x)=(f_1(x),f_2(x),...,f_k(x)) \mid x\in P_s\} \tag{4}$$

一般说来,求解 MOP 有以下 3 个要求[5]:(1) 找到最大 Pareto 最优解集;(2) 使算法所找到的 Pareto 前沿与已知的全局前沿距离最小(当已知全局前沿时才可以比较);(3) 所求解在 Pareto 前沿中分布均匀.

2 MOPSO 及研究现状

2.1 MOPSO

微粒群优化算法是一种新兴的进化计算技术,它是由 Kennedy 和 Eberhart 受鸟群觅食行为的启发于 1995 年提出的[3].尽管最初的设想是仿真简单的社会系统,研究并解释复杂的社会行为,但后来发现 PSO 是解决复杂优化问题的有效技术.PSO 是基于群体智能理论的优化算法,通过群体中粒子间的协作与竞争产生的群体智能指导优化搜索.PSO 与人工生命,特别是遗传算法有着极为特殊的联系.但相比遗传算法,PSO 保留了基于种群的全局搜索策略,采用简单的速度位移模型,避免了复杂的遗传操作,同时它特有的记忆能力使其可以动态跟踪当前的搜索情况以调整其搜索策略,具有较强的全局收敛能力和鲁棒性,且不需要借助问题的特征信息.因此,PSO 作为一种更高效的并行搜索算法,非常适于对复杂环境中优化问题的求解.目前,PSO 已经应用到电力、化工、机器人、机械设计、通信、经济学、图像处理、生物信息、医学、运筹学等领域[11].

在 PSO 中,粒子在超微搜索空间中飞行,假设一个微粒群由 m 个微粒组成,每个微粒代表 D 维搜索空间中的一个解,其中第 i 个微粒的空间位置为 $x_i=(x_{i1},x_{i2},x_{i3},...,x_{iD})$,$i=1,2,...,m$;第 i 个微粒所经历的最优位置称为其个体历史最优位置,记为 $p_i=(p_{i1},p_{i2},p_{i3},...,p_{iD})$;同时,每个微粒还具有各自的飞行速度 $v_i=(v_{i1},v_{i2},v_{i3},...,v_{iD})$.所有微粒经历过的最优位置称为全局最优位置,记为 $p_g=(p_{g1},p_{g2},p_{g3},...,p_{gD})$,则微粒的位置更新公式和速度更新公式分别为

$$\begin{cases} x_i(t)=x_i(t-1)+v_i(t) \\ v_i(t+1)=v_i(t)+c_1r_1(p_i-x_i(t))+c_2r_2(p_g-x_i(t)) \end{cases} \tag{5}$$

其中,c_1 和 c_2 分别是学习认知因子和社会认知因子,一般地,$c_1=c_2=2$,$r_1,r_2\in[0,1]$,是服从均匀分布的随机变量.

1999 年,Moore 和 Chapman 首先在他们未公开发表的文稿中提出扩展 PSO 求解 MOP[4],称为 MOPSO 方法.在目前的专业文献中,已经有 25 种 MOPSO 算法之多.一般说来,扩展 PSO 应用于 MOP 时,需要考虑的主要问题有以下 3 个[5]:(1) 为了让非劣解占据优势,如何选择作为领导的粒子?(2) 为了输出非劣解,如何保持搜索过程中发现的非劣解?(3) 为避免收敛到单一解,如何维持种群的多样性?目前的 MOPSO 算法多数从以上几个方面对 PSO 加以扩展和改进,从而形成了各种不同的 MOPSO 算法.

2.2 研究现状

在目前的 MOPSO 中,具有代表性的方法有以下几种:(1) 加权法.如 Parsopoulos 和 Vrahatis 方法[12],此方法采用了 3 种加权函数:传统的线性加权函数、动态加权函数和突变加权函数.(2) 层次序列法.如 Hu 和 Eberhart 方法[13],它使用类似字典序的模式,每次只优化一个目标,这种方法采用环形拓扑,没有使用附加的外部档案.在其下一个版本中,作者引入了一个外部档案(或称外部内存),并对动态邻居 PSO 方法加以改进.(3) 基于 Pareto 的方法,是研究最多的方法.如:Moore 和 Chapman 方法[4],这种方法出现在他们未出版的文稿中,基于 Pareto 支配.Fieldsend 和 Singh 方法[14]基于无约束的外部精英档案存储搜索过程中找到的非劣个体,其外部档案采用了一种被称为占优树的特定数据结构.Coello 等人的方法[15,16]引入外部档案保存非劣解,根据每个粒子的目标函数值定义一种基于地理位置的系统,然后再根据此系统更新外部档案,被搜索的空间按超立方体划分.Mostaghim 和 Teich 方法[17]赋予群和外部档案中每个粒子一个 Sigma 值,每个粒子选择外部档案中与其 Sigma 值最接近的粒子作为其领导,可以促进 MOPSO 方法的收敛性和多样性.Ho 等人的方法[18]基于对全连接拓扑飞行公式的 3 种修改,提出了一个更新粒子速度矢量和位置的公式.(4) 组合方法,如 Mahfouf 等人的方法[19],该方法是一种权重适应 PSO 算法(adaptive weighted PSO,简称 AWPSO),其速度矢量采用加速度方式修改,加速度随迭代数目的增加逐步增加.(5) 其他方法,如,Li 的最大最小 PSO 方法[20],该方法根据 Balling 提出的最大最小策略获取适应度值决定 Pareto 占优.

基于协同进化的 PSO 研究相对较少,可以分为两类:(1) 从问题的决策空间出发,采用多个子群分别搜索不同的区域.Iwamatsu 从决策空间出发,建立了多种群微粒群算法(MSPSO),用于求解多个全局最优解[9].Jang-Ho Seo 等人则建立了多小组微粒群算法(MGPSO)求解多模函数[10].Parsopoulos 等人研究了并行版本 VEPSO,是受

VEGA 的启示提出来的,用于求解 MOP.在 VEPSO 中,每个子群使用其中的一个目标函数评价微粒,子群拥有的信息在不同子群中交换[21].这类方法虽然采用了多个子群进行优化,但协同机制并不完善.(2) 从问题的决策变量出发将决策变量分解,从而形成不同的子问题,然后分别采用子群进行优化.Frans van den Bergh 提出了协同微粒群算法(CPSO)[8],可以将一个决策变量对应一个子群,也可以将多个决策变量对应一个子群,这是 PSO 方面的典型协同方法,但进一步的研究较少.在协同 GA 方面有不少的工作,如 Potter 等人的协同 GA[6],KC Tan 的 CCEA[7],都取得良好的效果.

3 一种基于合作型协同和ε-占优的多目标微粒群算法(CEPSO)

3.1 协同进化与适应度计算

协同进化是借鉴生物学模型近几年发展起来的一类新的进化算法,最早的研究始于 Potter 和 De Jong 的工作[6].Potter 等人扩展 GA 提出了一种协同结构的 GA,引入多个子群,每个子群分别优化问题的一个变量,子群数目与变量数目一致,协同 GA 在解决复杂问题方面表现出了一定的优势.其后,协同 GA 的研究逐步增多,既有求解单目标优化问题的,也有求解多目标优化问题的.协同进化可以分为竞争型协同进化和合作型协同进化两种.竞争型协同进化通过进化使得个体更具竞争力,是在两个种群的交互过程中互相促进,来源于生物学的捕食链模型.合作型协同进化则是将复杂的问题或系统分解为多个小的子问题或子系统,然后采用多个子群独立地进行协同进化,其方式是采纳那些优良个体构造更好的系统,因此对个体评价是从整个系统出发而不只是局限于单个子问题或子系统,合作型协同进化是分而治之的策略.Potter 的协同 GA 属于合作型协同进化.CEPSO 采用合作型协同方式,子种群的分解方式参照文献[8]的方式,设 D 为问题的维数,将变量分为 K 组,D_1 维有 K_1 组,D_2 维有 K_2 组,则 $K=K_1+K_2$,定义方式如下:

$$\begin{cases} K_1 = D \bmod K \\ K_2 = K - (D \bmod K) \\ D_1 = \left\lceil \dfrac{D}{K} \right\rceil \\ D_2 = \left\lfloor \dfrac{D}{K} \right\rfloor \end{cases} \tag{6}$$

如何选择协同者是协同进化的重要问题.Potter 等人提出了两种协同方式:(1) 从每个子群中选择一个最优粒子作为协同者;(2) 从每个子群中选择两个粒子:一个最优粒子和一个随机选择的粒子,取它们中的优者作为协同者.这两种方式成为许多协同研究的基础,多是基于他们的改进.但在多目标优化问题中,最优解往往是一个集合,各个解是平等的,因而无法确定最优.针对多目标优化问题,CEPSO 从外部档案中随机选择粒子,为每个子群选择一个粒子作为代表,然后用所选择出来到粒子构造完整的解向量,然后基于它们评价子群中所产生的粒子.

在 MOP 中,由于多个目标往往无法比较优劣关系和定义主次之分,Pareto 占优关系得到广泛的应用.本文中,粒子的适应度计算也按照 Pareto 占优关系来计算.在一个子群中,粒子适应度的计算过程如下:

select collaborartors from other subswarms;

construct context vector;

calculate the values of multi-objective functions;

calculate the Pareto ranks of the individuals;

基于以上分析,CEPSO 的协同过程和算法流程可描述为图 1.

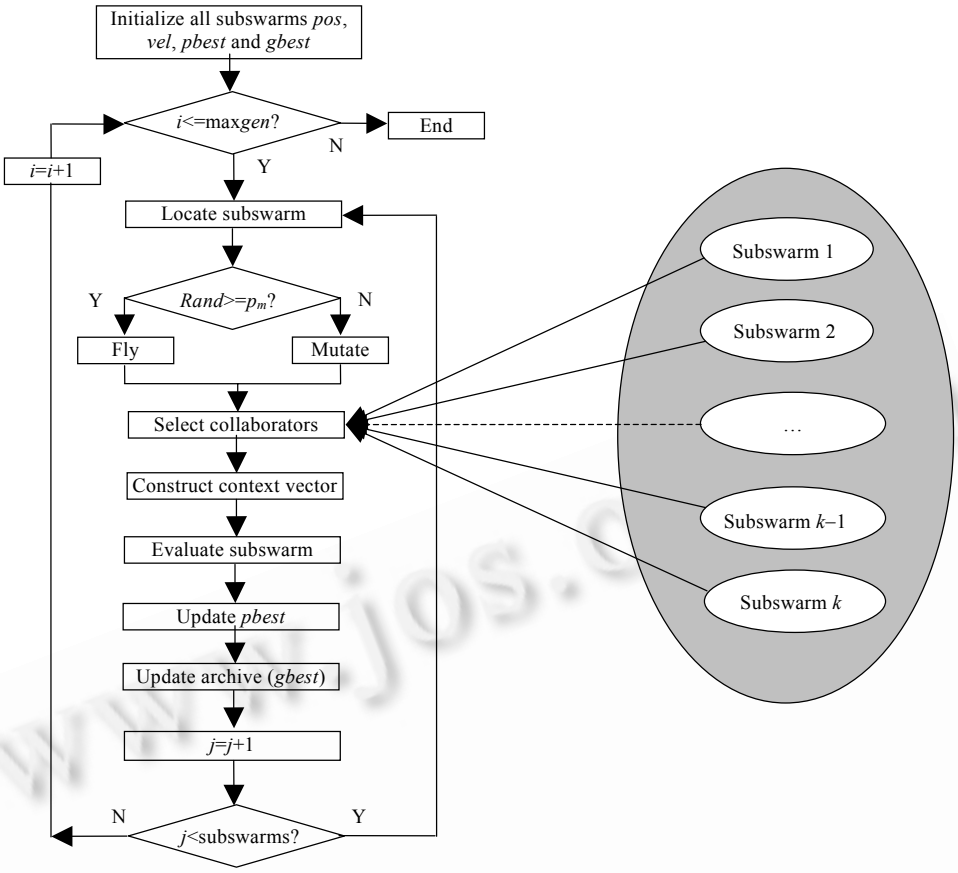

Fig.1 Cooperation and flow chart of CEPSO

图 1 CEPSO 的协同过程和算法流程

3.2 基于ε-占优(ε-dominance)的存档方法

在将 PSO 扩展为 MOPSO 时,一个很重要的方面就是要保存非劣解,为其建立外部档案.为此,在 CEPSO 中建立了基于ε-占优的外部档案,并且将档案集作为微粒的领导集合直接使用,各个子群不再单独建立领导集,而是使用一个统一的领导粒子集.非劣解的保存直接影响着 MOEA 的收敛性和多样性,许多现有的算法只能保证其中的一个方面,针对这种现象,Laumanns 等学者提出了基于ε-占优的存档策略[22],并且证明了该存档策略能够保证算法的收敛性,具有良好的分布特征,且能够根据ε值自动限制外部档案的大小.ε-占优的存档策略通过两个阶段完成,在粗粒度的方块划分阶段,将搜索空间离散化成一些方块,每个向量属于一个方块,在方块上进行占优关系的比较,并维持一组非劣方块,则可保证ε-近似的特性.方块的划分按照以下方法进行:

$$b_i = \left\lfloor \frac{\log f_i}{\log(1+\varepsilon)} \right\rfloor \tag{7}$$

从细粒度层次分析,每个方块至多保持一个元素,方块代表的向量仅能被占优的向量取代,因而保证了算法的收敛性.而整个档案的大小也限制在:

$$|A| \le \left(\frac{\log K}{\log(1+\varepsilon)} \right)^{(m-1)} \tag{8}$$

文献[23]对ε-占优的存档策略在 MOPSO 中的作用进行了详细分析,说明了该方法在保持收敛性和多样性方面优于传统的聚类方法等,并且减少了计算量.

3.3 速度更新公式和变异

在CEPSO中,由于采用了ε-占优的存档策略,将外部档案集和领导集进行了统一,因此直接随机地从外部档案集选择粒子作为领导粒子.Maurice Clerc通过对速度更新公式的研究指出改变随机变量可以提高PSO的性能[24],所提出的粒子更新公式为

$$\begin{cases} r = r_1 + r_2 \\ vel = K\left(vel + c_1 \dfrac{r_1}{r}(p_i - x_i) + c_2 \dfrac{r_2}{r}(g_i - x_i) \right) \end{cases} \tag{9}$$

这种更新公式使得当 c_1=c_2 时,$c_1 \times r_1 / r + c_2 \times r_2 / r = c_1$.本文中没有使用收缩因子,而是采用了线性下降的惯性矢量,即

$$vel = iwt(i) \times vel + c_1 \frac{r_1}{r}(p_i - x_i) + c_2 \frac{r_2}{r}(g_i - x_i) \tag{10}$$

这种更新公式有更广泛的应用范围,也能够提高PSO算法的性能.

早熟收敛是PSO的一个主要不足,往往使粒子群限于停滞状态,导致算法限于局部极值,然而一旦该情形被打破,则算法会很快收敛到全局最优解.变异算子常常起到打破停滞状态将粒子群引导到一个新状态的作用.在最初的PSO中并没有使用变异算子,但在后来出现的PSO和MOPSO中大多使用了变异算子.在这些算法中,变异的对象不外乎粒子速度矢量和粒子位置.就速度矢量的变异而言,一般是考虑到速度矢量为0时,粒子群中不会产生新解了,故采用变异来改变停滞状态;而粒子位置的变异则是考虑到了个体最优和全局最优不再改进的情形,通过变异算子,能够使PSO摆脱局部极值,加速收敛.变异的另一个重要作用是保持粒子群的多样性,其目的是避免出现早熟,可看作是一种预防措施.在CEPSO中,使用基于均匀分布的变异算子,设*rand*为[0,1]之间服从均匀分布的随机数,*varrange*(*d*)是变量的取值范围,其定义如下:

if $rand \le p_m$
  $x(d)=rand \times varrange(d)$
end

3.4 算法描述

Algorithm CEPSO
calculate the number *subswarmnum* and size of subswarms;
initialize all subswarms *pos*;
initialize all subswarms velocity *vel*;
pbest=*pos*;
construct archive *gbest* baesd on ε-dominance;
i=1;
while *i*<=max*gen*;
  for *j*=1: *subswarmnum*
    if $rand > p_m$
      for each particle in subswarm *j*
        select a leader from archive;
        calculate velocity *vel*; limit velocity *vel*;
        update position *pos*; limit position *pos*;
      end
    else
      $x(d)=rand \times varrange(d)$;
    end

```
        select collaborators from other subswarms;
        construct context vector;
        evaluate subswarm j;
        update pbest of subswarm j;
        update archive gbest of swarm based on ε-dominance;
    end
    i=i+1;
end
output archive gbest;
```

以上采用伪 Matlab 语言对 CEPSO 算法进行了描述,限于篇幅,不再详述,这里仅说明与其他算法的不同之处.CEPSO 与其他 MOPSO 的不同之处在于:(1) CEPSO 的飞行和变异这两种算子以一定的概率 p_m 交替执行,每一次迭代要么执行飞行算子,要么执行变异算子.而其他的 MOPSO 算法往往是先飞行再变异(速度矢量或位置),这样就有可能把通过飞行得到的优良解更新,反而得到一个更差的解.(2) CEPSO 将粒子的领导集与外部档案集统一,采用了ε-占优的档案更新策略,因而,在微粒选择全局领导粒子时,可直接从外部档案中选择.

4 仿真实验及分析

4.1 Benchmark函数

文献[25]中设计了实验函数产生器,根据不同的参数可以自动生成一组测试函数,其定义如下:

$$\begin{cases}\text{Minimize} & T(x)=(f_1(x_1),f_2(x))\\ \text{Subject to} & f_2(x)=g(x_2,...,x_m)h(f_1(x_1),g(x_2,...,x_m))\\ \text{where} & x=(x_1,...,x_m)\end{cases} \tag{11}$$

但由于参数的选择比较复杂,文中专门给出了一些 Benchmark 函数实例,这也是目前使用最为广泛的多目标 Benchmark 函数,其 Pareto 前沿具有不同的特点,具有 Benchmark 函数所需要的特征:凸函数、凹函数、含有大量局部极值的函数以及非均匀函数.限于篇幅,本文中只选择其中的 4 个函数,见表 1.

Table 1 ZDT benchmark functions for MOPs

表 1 ZDT 系列多目标 Benchmark 函数

Function	Definition	Decision variables
ZDT1	$\begin{cases}f_1(x_1)=x_1\\ g(x_2,...,x_m)=1+9\cdot\sum_{i=2}^{m}x_i/(m-1)\\ h(f_1,g)=1-\sqrt{f_1/g}\end{cases}$	$x=(x_1,x_2,\ldots,x_m)$ $x_i\in[0,1]$, $i=1,2,\ldots,D$
ZDT2	$\begin{cases}f_1(x_1)=x_1\\ g(x_2,...,x_m)=1+9\cdot\sum_{i=2}^{m}x_i/(m-1)\\ h(f_1,g)=1-(f_1/g)^2\end{cases}$	$x=(x_1,x_2,\ldots,x_m)$ $x_i\in[0,1]$, $i=1,2,\ldots,D$
ZDT4	$\begin{cases}f_1(x)=x_1\\ f_2(x)=g(x)\left[1-\sqrt{x_1/g(x)}\right]\\ g(x)=1+10(n-1)+\sum_{i=2}^{n}\left[x_i^2-10\cos(4\pi x_i)\right]\end{cases}$	$x_1\in[0,1]$, $x_i\in[-5,5]$, $i=2,\ldots,D$
ZDT6	$\begin{cases}f_1(x_1)=1-\exp(-4x_1)\sin^6(6\pi x_1)\\ g(x_2,...,x_m)=1+9\cdot\left(\sum_{i=2}^{m}x_i/(m-1)\right)^{0.25}\\ h(f_1,g)=1-(f_1/g)^2\end{cases}$	$x=(x_1,x_2,\ldots,x_m)$ $x_i\in[0,1]$, $i=1,2,\ldots,D$

4.2 性能评价指标

为了评价 CEPSO 算法的性能,采用以下评价指标对 CEPSO 算法加以评价.

(1) 近似解的个数(solution number in archive,简称 SNA)

在求 Pareto 解时,常常是得到非常近似的解,近似解的数目则同样可以反映算法求解的能力.若近似解分布均匀且非常接近 Pf_{true},那么也说明算法是有效的.本文将近似解的个数定义为外部档案中 Pareto 解的数目.

(2) 代距离(generational distance,简称 GD)

代距离用于衡量已经求得的 Pareto 前沿 Pf_{known} 与真实 Pareto 前沿 Pf_{true} 之间的距离,d_i 为目标空间中 Pf_{known} 中粒子与 Pf_{true} 中最近粒子的距离[7],其定义如下(本文中采用 p=1):

$$GD=\left(\frac{1}{n}\sum_{i=1}^{n}(d_i^{\,p})\right)^{\frac{1}{p}} \tag{12}$$

(3) 间隔(spacing)

间隔指标用于衡量所求得解的分布情况[16].其定义如下:

$$\begin{cases} S=\sqrt{\dfrac{1}{n-1}\sum_{i=1}^{n}(\overline{d}-d_i)^2} \\ d_i=\min_j\left(\left|f_1^i(x)-f_1^j(x)\right|+\left|f_2^i(x)-f_2^j(x)\right|\right) \end{cases},\ i,j=1,\ldots,n;\overline{d}=\frac{1}{n}\sum_{k=1}^{n}d_i \tag{13}$$

4.3 实验结果分析

针对不同的 Benchmark 函数,算法的主要参数设置如下:子种群大小为 *subswarmsize*=10,每一决策变量对应一个子种群 *subwarmnumber*=*D*,ε=0.05,ZDT1,ZDT2 和 ZDT6 的迭代代数为 *gen*=30 代;由于 ZDT4 有大量的局部极值点,因而计算的迭代代数设为 *gen*=50 代.函数的评价次数按照如下公式计算:

$$fe=subswarmsize\times D\times gen \tag{14}$$

也就是说,函数评价次数与函数的维数 *D* 成线性比例.各个 Benchmark 函数的计算结果为分别计算 30 次的平均值,见表 2~表 5.

Table 2 Results for ZDT1

表 2 计算 ZDT1 的结果

D	SNA	GD	Spacing
30	38	7.85E−19	1.13E−02
60	37	1.21E−18	1.33E−02
90	37	7.85E−19	1.45E−02
120	36	9.43E−05	1.70E−02
150	35	2.30E−04	1.84E−02

Table 3 Results for ZDT2

表 3 计算 ZDT2 的结果

D	SNA	GD	Spacing
30	22	9.25E−19	3.16E−03
60	22	1.32E−18	4.42E−03
90	22	8.41E−19	3.31E−03
120	22	2.66E−18	3.99E−03
150	22	1.14E−18	3.84E−03

Table 4 Results for ZDT4

表 4 计算 ZDT4 的结果

D	SNA	GD	Spacing
10	39	6.14E−05	9.83E−03
20	38	1.18E−04	9.80E−03
30	38	4.07E−06	1.34E−02
40	39	2.20E−05	1.00E−02
50	38	3.96E−05	1.15E−02

Table 5 Results for ZDT6

表 5 计算 ZDT6 的结果

D	SNA	GD	Spacing
10	18	2.06E−19	1.38E−02
20	19	1.22E−18	1.59E−02
30	18	7.90E−19	1.68E−02
40	18	1.23E−18	1.66E−02
50	18	1.01E−18	1.92E−02

表 2 和图 2 是 CEPSO 对 ZDT1 函数的计算结果.ZDT1 函数是所选择的 Benchmark 函数中最简单的一个,其 Pareto 前沿分布均匀,为凸函数,且没有局部极值,现有的多目标微粒群算法都可以求出 ZDT1 函数的 Pareto 前沿.针对比较高维数的 ZDT1 函数,CEPSO 能够求得 ZDT1 函数的 Pareto 前沿,从档案中包含的解数目 SNA 和代距离 GD 的图形中就可以明确看出,在维数偏低时,30 次计算都可以求得准确的 Pareto 前沿,当维数偏高时,也只有 1 次偏离不能求得精确的 Pareto 前沿,但所求结果也非常近似,代距离为 10^{-3}.从 Spacing 的分布可以看出,各种维数的 Pareto 解分布都比较均匀,看不出差别.

表 3 和图 3 是 CEPSO 对 ZDT2 函数的计算结果.ZDT2 函数的 Pareto 前沿是凹函数,也没有局部极值,只要采用较大的变异比例,现有的 MOPSO 也都可以求出 ZDT2 函数的 Pareto 前沿.针对比较高维数的 ZDT2 函

数,CEPSO 能够求得 ZDT2 函数的 Pareto 前沿,从档案中包含的解数目 SNA 和代距离 GD 的图形中就可以看出,这两个值都没有大的起伏,30 次计算都可以求得非常近似的 Pareto 前沿.但从 Spacing 的分布来看,所求解的分布有一些变化,主要原因是 f_1 在[0,1]区间的解较难保持.

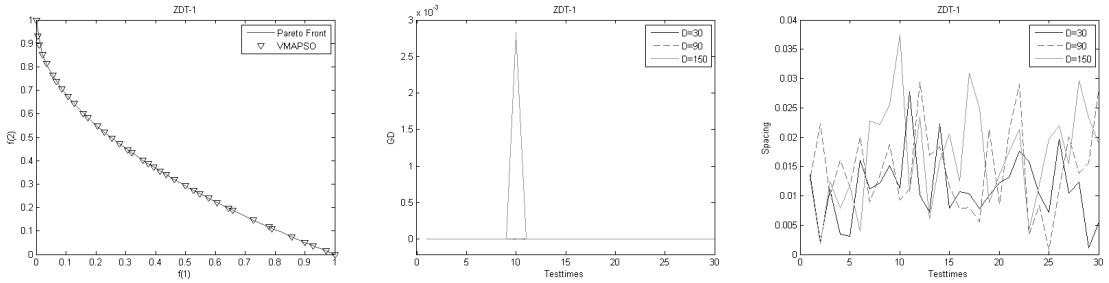

Fig.2 The Pareto front (D=90), Spacing and GD of ZDT1

图 2 ZDT1 的 Pareto 前沿(D=90)、间隔距离和代距离

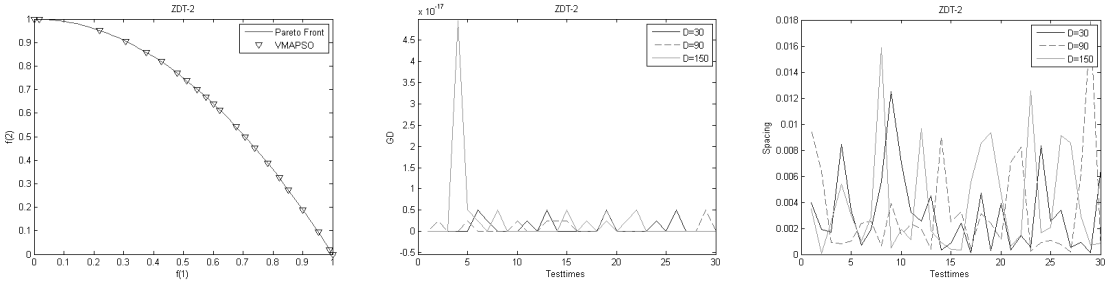

Fig.3 The Pareto front (D=90), Spacing and GD of ZDT2

图 3 ZDT2 的 Pareto 前沿(D=90)、间隔距离和代距离

表 4 和图 4 是 CEPSO 对 ZDT4 函数的计算结果.ZDT4 函数是所选择的 Benchmark 函数中最复杂的一个,其第 1 个变量的取值范围与其余变量的取值范围不同,包含 21^9 个局部极值,求解极其困难,使许多算法陷于局部极值无法摆脱,但 Pareto 前沿与 ZDT1 相同,Pareto 前沿分布均匀,为凸函数.针对比较高维数的 ZDT4 函数,CEPSO 能够求得 ZDT4 函数的 Pareto 前沿,从档案中包含的解数目 SNA 和代距离 GD 的图形中就可以明显看出,在维数偏低时,30 次计算都可以求得准确的 Pareto 前沿,当维数偏高时,有多次计算中不能求得精确的 Pareto 前沿,代距离在 10^{-4} 级别,因此所求结果也非常近似.从 Spacing 的分布来看,所求解的分布情况比前面几个函数要差一些,但间隔指标都保持在 10^{-2} 以内.

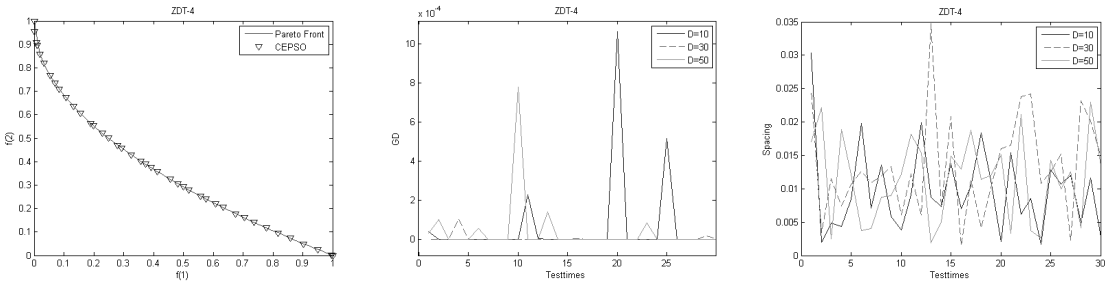

Fig.4 The Pareto front (D=30), Spacing and GD of ZDT4

图 4 ZDT4 的 Pareto 前沿(D=30)、间隔距离和代距离

表 5 和图 5 是 CEPSO 对 ZDT6 函数的计算结果.ZDT6 函数的 Pareto 前沿与 ZDT1 相同,但 Pareto 解的分布不均匀,因而难以求得准确的 Pareto 前沿.针对比较高维数的 ZDT6 函数,CEPSO 能够求得 ZDT6 函数的近似

Pareto 前沿,从档案中包含的解数目 SNA 和代距离 GD 的图形中可以看出,30 次计算都可以求得非常近似的 Pareto 前沿,代距离非常小,保持在 10^{-18} 水平.从 Spacing 的分布来看,所求解的分布比较均匀,间隔指标在 10^{-2} 左右.

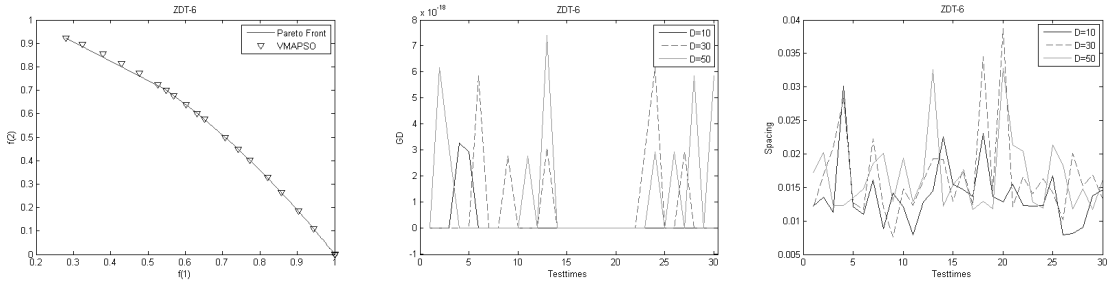

Fig.5 The Pareto front (D=30), Spacing and GD of ZDT6

图 5 ZDT6 的 Pareto 前沿(D=30)、间隔距离和代距离

从以上的计算结果和分析来看,在求解偏低维数的 MOPs 时,CEPSO 算法可以求得精确的 Pareto 前沿;在求解复杂偏高维数和多局部极值等 MOPs 时,Pareto 解的数目基本恒定,代距离较小,说明所求得的 Pareto 前沿非常接近真实的 Pareto 前沿;同时,Pareto 解的分布也比较均匀.因而,在计算代数不变的情况下,计算各种维数的 ZDT 函数的结果没有大的差异,但 CEPSO 算法所需要的函数评价次数仅以近似线性的比例随测试函数维数增加而增加,表明 CEPSO 在计算偏高维数和多局部极值点的 MOP 问题时更具有优势.

5 结 论

针对多目标微粒群算法存在容易陷于局部极值、函数评价次数多和受到维数限制等问题,提出了一种基于合作型协同和ε-占优的多目标微粒群算法(CEPSO),以期克服已有算法所存在的不足之处.CEPSO 结合了协同 GA 的研究成果和 PSO 算法的优势,基于协同进化的框架构造.在多个具有不同特征的经典 Benchmark 函数上进行仿真实验,结果表明,在求解比较高维数的多目标优化问题时,CEPSO 算法能够摆脱局部极值,保持多样性,所求的解分布更均匀、更接近 Pareto 前沿,并且函数的评价次数与维数成近似线性的比例关系.表明 CEPSO 是一种比较有研究前景的方法.

虽然仿真实验结果表明了 CEPSO 算法具有一定的优势,但结果仍然比较初步,仍有许多方面值得进一步研究,包括研究 CEPSO 求解更高维数的 MOP;研究更完善的协同进化机制以提高算法的性能;将 CEPSO 算法用于实际的多目标优化问题,验证算法解决实际问题的效果.

References:

[1] Coello CAC. An updated survey of evolutionary multi-objective optimization techniques: State of the art and future trends. In: Proc. of the Congress on Evolutionary Computation. Washington: IEEE Service Center, 1999. 3−13.

[2] Coello CAC. Evolutionary multi-objective optimization: A historical view of the field. IEEE Computational Intelligence Magazine, 2006,1(1):28−36.

[3] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proc. of the IEEE Conf. on Neural Networks, IV. Perth: IEEE Press, 1995. 1942−1948.

[4] Moore J, Chapman R. Application of particle swarm to multi-objective optimization. Department of Computer Science and Software Engineering, Auburn University, 1999.

[5] Reyes-Sierra M, Coello CAC. Multi-Objective particle swarm optimizers: A survey of the state-of-the-art. Int'l Journal of Computational Intelligence Research, 2006,2(3):287−308.

[6] Potter MA, de Jong KA. A cooperative coevolutionary approach to function optimization. In: Proc. of the 3rd Parallel Problem Solving from Nature. Berlin: Springer-Verlag, 1994. 249−257.

[7] Tan KC, Yang YJ, Goh CK. A distributed cooperative coevolutionary algorithm for multiobjective optimization. IEEE Trans. on

Evolutionary Computation, 2006,10(5):527−549.

[8] van den Bergh F, Engelbrecht AP. A cooperative approach to particle swarm optimization. IEEE Trans. on Evolutionary Computation, 2004,8(3):225−239.

[9] Iwamatsu M. Locating all global minima using multi-species particle swarm optimizer: The inertia weight and the constriction factor variants. In: Proc. of the 2006 IEEE Congress on Evolutionary Computation. Vancouver, 2006. 816−822.

[10] Seo JH, Im CH, Heo CG, Kim JK, Jung HK, Lee CG. Multimodal function optimization based on particle swarm optimization. IEEE Trans. on Magnetics, 2006,42(4):1095−1098.

[11] Liu B, Wang L, Jin YH, Huang DX. Advances in particle swarm optimization algorithm. Control and Instruments in Chemical Industry, 2005,32(3):1−6 (in Chinese with English abstract).

[12] Parsopoulos KE, Vrahatis MN. Particle swarm optimization method in multi-objective problems. In: Proc. of the 2002 ACM Symp. on Applied Computing (SAC 2002). Madrid: ACM Press, 2002. 603−607.

[13] Hu XH, Eberhart R. Multi-Objective optimization using dynamic neighborhood particle swarm optimization. In: Proc. of Congress on Evolutionary Computation (CEC 2002). Piscataway: IEEE Service Center, 2002. 1677−1681.

[14] Fieldsend JE, Singh S. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In: Proc. of the 2002 U.K. Workshop on Computational Intelligence. Birmingham, 2002. 37−44.

[15] Coello CAC, Lechuga MS. MOPSO: A proposal for multiple objective particle swarm optimization. In: Proc. of Congress on Evolutionary Computation (CEC 2002). Piscataway: IEEE Service Center, 2002. 1051−1056.

[16] Coello CAC, Pulido GT, Lechuga MS. Handling multiple objectives with particle swarm optimization. IEEE Trans. on Evolutionary Computation, 2004,8(3):256−279.

[17] Mostaghim S, Teich J. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: Proc. of the 2003 IEEE Swarm Intelligence Symp. Indianapolis: IEEE Service Center, 2003. 26−33.

[18] Ho SL, Yang SY, Ni GZ, Edward WC, Wong HC. A particle swarm optimization based method for multiobjective design optimizations. IEEE Trans. on Magnetics, 2005,41(5):1756−1759.

[19] Mahfouf M, Chen MY, Linkens DA. Adaptive weighted particle swarm optimization for multi-objective optimal design of alloy steels. In: Proc. of Parallel Problem Solving from Nature-PPSN VIII. LNCS 3242, Birmingham: Springer-Verlag, 2004. 762−771.

[20] Li XD. Better spread and convergence: Particle swarm multi-objective optimization using the maximin fitness function. In: Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2004). LNCS 3102, Seattle, Washington: Springer-Verlag, 2004. 117−128.

[21] Parsopoulos KE, Tasoulis DK, Vrahatis MN. Multi-Objective optimization using parallel vector evaluated particle swarm optimization. In: Proc. of the IASTED Int'l Conf. on Artificial Intelligence and Applications (AIA 2004). Innsbruck: ACTA Press, 2004. 823−828.

[22] Laumanns M, Thiele L, Deb K, Zitzler E. Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation, 2002,10(3):263−282.

[23] Mostaghim S, Teich J. The role of ε-dominance in multi objective particle swarm optimization methods. In: Proc. of the 2003 Congress on Evolutionary Computation (CEC 2003). Canberra: IEEE Press, 2003. 1764−1771.

[24] Clerc M, Kennedy J. The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space. IEEE Trans. on Evolutionary Computation, 2002,6(1):58−73.

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

附中文参考文献:

[11] 刘波,王凌,金以慧,黄德先.微粒群优化算法研究进展.化工自动化及仪表,2005,32(3):1−6.

郑向伟(1971－),男,山东泰安人,博士生,高级工程师,主要研究领域为计算智能,计算机辅助设计.

刘弘(1955－),女,博士,教授,博士生导师,主要研究领域为计算智能,CSCW,多Agent 系统.