ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.11, November 2009, pp.2925−2938
doi: 10.3724/SP.J.1001.2009.03496

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

M-精英协同进化数值优化算法*

慕彩红[1,2+], 焦李成[1,2], 刘 逸[3,4]

[1](西安电子科技大学 智能感知与图像理解教育部重点实验室,陕西 西安 710071)

[2](西安电子科技大学 智能信息处理研究所,陕西 西安 710071)

[3](西安电子科技大学 综合业务网理论及关键技术国家重点实验室,陕西 西安 710071)

[4](西安电子科技大学 电子工程学院,陕西 西安 710071)

M-Elite Coevolutionary Algorithm for Numerical Optimization

MU Cai-Hong[1,2+], JIAO Li-Cheng[1,2], LIU Yi[3,4]

[1](Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education, Xidian University, Xi'an 710071, China)

[2](Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, China)

[3](State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China)

[4](School of Electronic Engineering, Xidian University, Xi'an 710071, China)

+ Corresponding author: E-mail: caihongm@mail.xidian.edu.cn, http://www.xidian.edu.cn

Mu CH, Jiao LC, Liu Y. *M*-Elite coevolutionary algorithm for numerical optimization. *Journal of Software*, 2009,20(11):2925−2938. http://www.jos.org.cn/1000-9825/3496.htm

Abstract: The *M*-elite coevolutionary algorithm (MECA) is proposed for high-dimensional unconstrained numerical optimization problems based on the concept of coevolutionary algorithm and elitist strategy. In the MECA, the individuals with high fitness, called elite population, is considered to play dominant roles in the evolutionary process. The whole population is divided into two subpopulations which are elite population composed of *M* elites and common population including other individuals, and team members are selected to form *M* teams by *M* elites acting as the cores of the *M* teams (named as core elites) respectively. If the team member selected is another elite individual, it will exchange information with the core elite with the cooperating operation defined in the paper; If the team member is chosen from the common population, it will be led by the core elite with the leading operation. The cooperating and leading operation above are defined by different combinations of several crossover operators or mutation operators. The algorithm is proved to converge to the global optimization solution with probability one. Tests on 15 benchmark problems show that the algorithm can find the global optimal solution or near-optimal solution for most problems tested. Compared with three existing algorithms, MECA achieves an improved accuracy with the same number of function evaluations. Meanwhile, the runtime of MECA is less, even

* Supported by the National Natural Science Foundation of China under Grant Nos.60703107, 60703108, 60703109, 60702062 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01Z107, 2007AA12Z136, 2007AA12Z223 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2006CB705700 (国家重点基础研究发展计划(973)); the Program for Cheung Kong Scholars and Innovative Research Team in University of China under Grant No.IRT0645 (长江学者和创新团队发展计划)

Received 2008-05-06; Revised 2008-08-07; Accepted 2008-10-28; Published online 2009-09-23

compared with the standard genetic algorithm with the same parameter setting. Moreover, the parameters of the MECA are analyzed in experiments and the results show that MECA is insensitive to parameters and easy to use.

Key words: unconstrained optimization problem (UOP); numerical optimization; elitist strategy; evolutionary algorithm; coevolutionary algorithm

摘 要: 为了解决高维无约束数值优化问题,借鉴协同进化和精英策略的思想,提出了 *M*-精英协同进化算法.该算法认为,适应度较高的个体群(称为精英种群)在整个种群进化中起着主导作用.算法将整个种群划分为由 *M* 个精英组成的精英种群和由其余个体组成的普通种群这样两个子种群,依次以 *M* 个精英为核心(称为核心精英)来选择成员以组建 *M* 个团队.若选中的团队成员是其他精英,则该成员与核心精英利用所定义的协作操作来交换信息;若团队成员选自普通种群,则由核心精英对其进行引导操作.其中,协作操作和引导操作由若干不同类型的交叉或变异算子的组合所定义.理论分析证明,算法以概率 1 收敛于全局最优解.对 15 个标准测试函数进行的测试显示,该算法能够找到其中几乎所有被测函数的最优解或好的次优解.与 3 个已有的算法相比,在评价次数相同时,该算法所求解的精度更高.同时,该算法的运行时间较短,甚至略短于同等设置下的标准遗传算法.此外,对参数的实验分析显示,该算法对参数不敏感,易于使用.

关键词: 无约束优化问题;数值优化;精英策略;进化算法;协同进化算法

中图法分类号: TP18 **文献标识码**: A

维数较高的数值优化问题往往比较复杂,进化算法由于基本上不需要问题的领域知识,并且对函数的类型和搜索空间的形状没有任何限制,因而能够比传统的数值优化方法更好地解决这些问题.对于无约束优化,进化算法面临的主要问题在于算法可能陷入局部极值点.为了解决这一问题,人们提出了许多新方法,如快速进化规划[1]、正交遗传算法[2]、佳点集遗传算法[3]等.这些方法均取得了良好的效果.

以上算法大多是采用达尔文进化论的适者生存、强调竞争的进化思想,然而在自然界,生物间不仅存在竞争也存在协作.协同进化算法是近十几年来在协同进化论基础上提出的一类新的进化算法,其与一般进化算法的区别在于,协同进化算法在进化算法的基础上,考虑了种群与环境之间、种群与种群之间在进化过程中的协调.

Potter较早提出了一种合作型协同进化遗传算法(CCGA)[4],其基本思想是将待求问题的解向量分裂成多个子元素,每个子元素通过一个物种种群进行进化,每个物种种群内某个体的适应度通过由该个体和来自其他物种种群的个体组合而成的解向量来评价.Potter通过实验证明,CCGA在求解高维函数优化的问题上性能要大大优于标准遗传算法.

van den Bergh将这一思想应用到标准粒子群(particle swarm optimization,简称PSO)算法,构造了一种新的合作型粒子群模型CPSO-S_K[5],其中下标K称为分裂因子,表示将一个解向量分裂为K个部分.但他也指出,对搜索空间进行分割引起的边缘效应,会使算法容易陷入伪极小值点(pseudominima).为避免这种情况的发生,van den Berg又将CPSO-S_K与标准PSO相结合,构造了CPSO-H_K算法,并在数值优化问题上取得了较好的效果[5].

目前,协同进化算法已经成为当前进化计算的一个热点问题[6−8].Liu等人提出了组织进化算法(OEA)[9].该算法使用分裂算子、吞并算子以及合作算子对种群在组织的基础上进行进化操作,在数值优化问题上取得了很好的效果.

精英策略是某些进化算法采用的一种进化机制,通常是指在进化算法中使种群中适应度最大的个体不经过遗传操作直接进入下一代,采用精英策略的进化算法可以较快地收敛到最优解[10].这说明适应度较高的个体(下文称为精英个体)对于种群的进化起着重要的推动作用.

基于这一思想并结合协同进化的思想,本文提出了一种新的求解高维无约束优化问题的算法,即 *M*-精英协同进化算法(*M*-elite coevolutionary algorithm,简称 MECA).该算法将整个种群划分为精英种群(含有 *M* 个精英个体)和普通种群两个子种群,并以 *M* 个精英为核心来组建 *M* 个团队,在组建团队的过程中,不同的精英之间采用

协作操作,精英对团队内来自普通种群的成员进行引导操作,并使用长方体交叉算子 I(cuboid crossover operator I)、离散交叉算子(discrete crossover operator)、翻转交叉算子(flip crossover operator)、长方体交叉算子 II(cuboid crossover operator II)以及变异算子(mutation operator)等操作算子来实现个体间信息的交换,促使种群进化.为表述方便,后文分别用 CCOI,DCO,FCO,CCOII 以及 MO 来指代上述各算子.

与 Potter 的 CCGA 算法思想不同,本文算法没有对解空间进行分割,因而不会面临陷入伪极小值点的问题.CCGA 中的每个物种表示多维解向量中的一维变量,通过各个物种间的协作来完成对问题的优化.而本文算法只使用了一个物种种群来表示问题的解,只是将整个物种种群划分为两个子种群,协同思想主要体现在物种内部不同子种群之间的协作上.

理论分析证明,MECA 算法收敛于全局最优解.仿真实验和参数分析结果表明,MECA 寻优能力很强,能够很好地解决高维无约束优化问题,并且计算代价较低.

1 MECA 算法

无约束优化问题一般可以描述为下述的数学模型:

$$\text{Minimize } f(\boldsymbol{x}),\ \boldsymbol{x}=(x_1,\ldots,x_n)\in S \tag{1}$$

这里,$S\subseteq R^n$为搜索空间,其范围为

$$\underline{x_l}\le x_l\le \overline{x_l},\ l=1,2,...,n \tag{2}$$

本文中,个体用实值向量$\boldsymbol{x}=(x_1,x_2,\ldots,x_n)$来表示.只考虑最小化问题,则个体的适应度$Fitness(\boldsymbol{x})$定义为$-f(\boldsymbol{x})$.

设种群规模为N,种群$pop=(\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N)$,MECA算法在每代的初始阶段将整个种群$pop$划分为两个子种群:父代精英种群$popEp$和父代普通种群$popCp$,规模分别为$M$和$Nc=N-M$,其中,父代精英种群为当前种群中$M$个适应度最高的个体,其余个体构成普通种群,同时生成初始的子代精英种群$popEc$和子代普通种群$popCc$,且$popEc=popEp$,$popCc=popCp$.

MECA算法充分发挥精英种群在种群进化中的推动作用,在每代进化中以当前代M个精英作为进化操作的核心,依次利用M个精英$popEp_i(i=1,\ldots,M)$作为团队核心来组建M个团队.每个团队的成员数为$G=\lceil 0.8\times(N-M)/M\rceil$(其中,$\lceil x\rceil$表示对$x$向上取整),各个团队中的每个成员等概率地从精英种群和普通种群中选取.若成员来自精英种群,则与$popEp_i$进行协作操作;否则,利用$popEp_i$对其进行引导操作.通过各个团队的精英与其团队成员进行协作操作或引导操作来产生新个体,并不断更新子代精英种群$popEc$和子代普通种群$popCc$.当M个团队组建完毕时,子代精英种群$popEc$和子代普通种群$popCc$也同时更新完毕,将$popEc$和$popCc$合并,构成下一代的种群pop.

1.1 协作操作

设当前团队的核心为精英个体$popEp_i$:$\boldsymbol{x}=(x_1,x_2,\ldots,x_n)$,$i\in\{1,\ldots,M\}$,$popEp_i$所选择的当前成员个体为来自精英种群的精英个体$popEp_j$:$\boldsymbol{y}=(y_1,y_2,\ldots,y_n)$,$j\in\{1,\ldots,M\}$,且$j\ne i$.

本文为协作操作定义了如下 3 种算子:CCOI,DCO,FCO.如果$U(0,1)<Pcu$,新个体$\boldsymbol{zu}=(zu_1,zu_2,\ldots,zu_n)$与$\boldsymbol{zv}=(zv_1,zv_2,\ldots,zv_n)$由CCOI算子产生,否则由DCO算子产生.这里,$U(a,b)$表示区间$(a,b)$内的一个均匀分布的随机数产生器,$Pcu\in(0,1)$,称为长方体交叉概率,是预先设定好的参数,两种算子分别由公式(3)和公式(5)给出.

在 CCOI 算子中,新个体由公式(3)产生:

$$\begin{cases} zu_k=\lambda_k\times x_k+(1-\lambda_k)\times y_k\\ zv_k=(1-\lambda_k)\times x_k+\lambda_k\times y_k \end{cases},\ k=1,2,...,n \tag{3}$$

其中,$\lambda_k=U_k(0,2)$.

由公式(3)可以看出,利用 CCOI 算子所产生的新个体 $\boldsymbol{zu}$,$\boldsymbol{zv}$ 可能会超出解空间的约束范围,所以接下来需要进行如公式(4)所示的出界判别.

$$\begin{cases} u_k = \begin{cases} \underline{x_k}, & zu_k < \underline{x_k} \\ \overline{x_k}, & zu_k > \overline{x_k} \\ zu_k, & \text{otherwise} \end{cases} \\ v_k = \begin{cases} \underline{y_k}, & zv_k < \underline{y_k} \\ \overline{y_k}, & zv_k > \overline{y_k} \\ zv_k, & \text{otherwise} \end{cases} \end{cases}, \ k = 1,2,...,n \tag{4}$$

在 DCO 算子中,新个体由公式(5)产生.

$$\begin{cases} \boldsymbol{u} = (x_1, x_2, ..., x_{l_1-1}, y_{l_1}, y_{l_1+1}, ..., y_{l_2}, x_{l_2+1}, x_{l_2+2}, ..., x_n) \\ \boldsymbol{v} = (y_1, y_2, ..., y_{l_1-1}, x_{l_1}, x_{l_1+1}, ..., x_{l_2}, y_{l_2+1}, y_{l_2+2}, ..., y_n) \end{cases} \tag{5}$$

其中,$1<l_1<n$,$1<l_2<n$,且$l_1<l_2$.

当 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 比较接近时,二者进行离散交叉将很难产生新个体,为此设计了如公式(6)所示的翻转交叉算子 FCO.当 $\sqrt{\sum_{l=1}^{n}(x_l - y_l)^2} < 0.5 \times \left[\frac{1}{n}\sum_{l=1}^{n}(\overline{x_l} - \underline{x_l})\right]$ 且 $U(0,1)<0.5$ 时,利用公式(6)对 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 进行翻转交叉产生新个体 $\boldsymbol{zu}$ 和 $\boldsymbol{zv}$:

$$\begin{cases} \boldsymbol{zu} = (x_1, x_2, ..., x_{l_1-1}, y_{l_2}, y_{l_2-1}, ..., y_{l_1+1}, y_{l_1}, x_{l_2+1}, x_{l_2+2}, ..., x_n) \\ \boldsymbol{zv} = (y_1, y_2, ..., y_{l_1-1}, x_{l_2}, x_{l_2-1}, ..., x_{l_1+1}, x_{l_1}, y_{l_2+1}, y_{l_2+2}, ..., y_n) \end{cases} \tag{6}$$

翻转交叉产生的新个体 $\boldsymbol{zu}$ 和 $\boldsymbol{zv}$,需要利用公式(4)进行出界判别,产生 $\boldsymbol{u}$ 和 $\boldsymbol{v}$.

利用以上算子产生新个体 $\boldsymbol{u}$,$\boldsymbol{v}$ 后,利用公式(7)、公式(8)所给出的选拔准则 I 来更新子代种群 *popEc*:

$$popEc_i = \begin{cases} \boldsymbol{u}, & Fitness(\boldsymbol{x}) \le Fitness(\boldsymbol{u}) \\ \boldsymbol{x}, & \text{otherwise} \end{cases} \tag{7}$$

$$popEc_j = \begin{cases} \boldsymbol{v}, & Fitness(\boldsymbol{y}) \le Fitness(\boldsymbol{v}) \\ \boldsymbol{y}, & \text{otherwise} \end{cases} \tag{8}$$

需要说明的是,在实际操作中,公式(7)、公式(8)中的 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 取的是子代精英种群中位置对应于原父代个体的子代个体,以保证之前循环中所产生的好的个体不会被比它差的个体所覆盖.

1.2 引导操作

设当前团队的核心为精英个体$popEp_i$:$\boldsymbol{x}=(x_1,x_2,\ldots,x_n)$,$i\in\{1,\ldots,M\}$,$popEp_i$所选择的当前成员个体为来自普通种群的普通个体$popCp_j$:$\boldsymbol{y}=(y_1,y_2,\ldots,y_n)$,$j\in\{1,\ldots,N-M\}$.

在引导操作中,定义了如下两种算子:CCOII和MO.如果$U(0,1)<Pcu$,新个体$\boldsymbol{zu}=(zu_1,zu_2,\ldots,zu_n)$由CCOII算子产生,否则由MO算子来产生.这里,$Pcu\in(0,1)$,与协作操作中的Pcu为同一参数,取值相同.两种算子分别由公式(9)和公式(10)给出.

在CCOII算子中,新个体$\boldsymbol{zu}=(zu_1,zu_2,\ldots,zu_n)$由公式(9)产生:

$$zu_k = x_k + \lambda_k \times (x_k - y_k), \ k = 1,2,...,n \tag{9}$$

其中,$\lambda_k=U_k(-1,1)$.

利用 CCOII 算子所产生的新个体 $\boldsymbol{zu}$ 同样需要利用公式(4)进行出界判别生成 $\boldsymbol{u}$.

在 MO 算子中,新个体由公式(10)产生:

$$u_k = \begin{cases} \underline{x_k} + \lambda \times (\overline{x_k} - \underline{x_k}), & U_k(0,1) < \dfrac{2}{n} \\ x_k, & \text{otherwise} \end{cases}, \ k = 1,2,...,n \tag{10}$$

其中,$\lambda=U(0,1)$且对每个x_k都不同.

$\boldsymbol{u}$ 产生后,利用公式(11)所给出的选拔准则 II 来更新子代种群 *popCc*:

$$\begin{cases} popCc_j = u,\ Fitness(u) \geq Fitness(y) \\ popCc_j = u,\ (Fitness(u) < Fitness(y)) \text{ and } [U_j(0,1) < \exp(Fitness(u) - Fitness(y))] \\ popCc_j = y,\ \text{otherwise} \end{cases} \tag{11}$$

当$popCc_j$取为$\boldsymbol{u}$时,从$popCp$中删除$popCp_j$,且令N_c=N_c−1.

1.3 算法流程

图 1 所示为 MECA 算法流程.

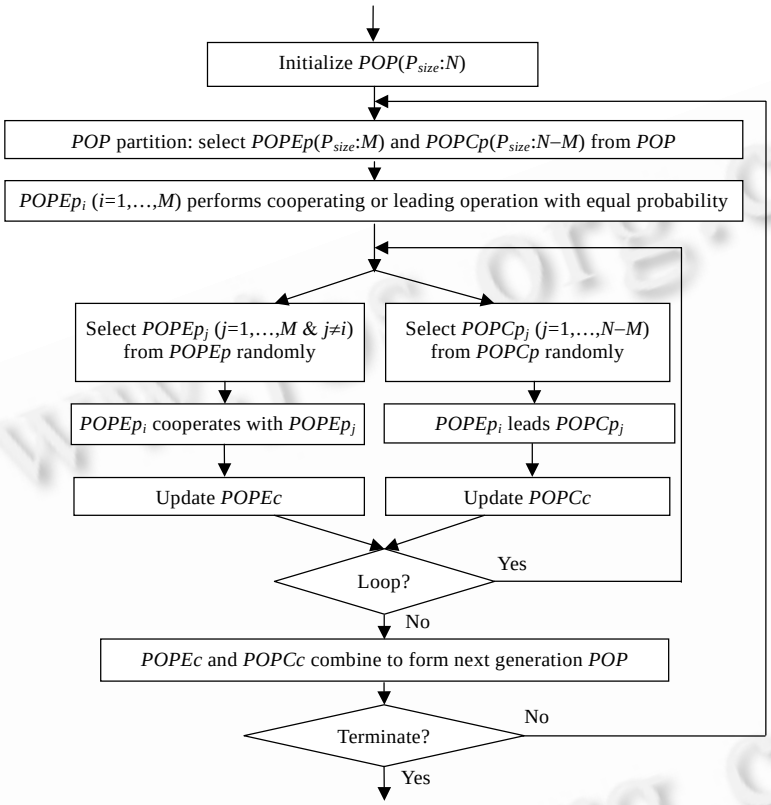

Fig.1 Flow diagram of MECA

图 1 MECA 算法流程图

2 算子的几何意义

下面说明算法中的长方体交叉算子 CCOI 和 CCOII 的几何意义.对协作操作中的 CCOI 算子,即公式(3)变形得:

$$\begin{cases} zu_k = y_k - \lambda_k \times (y_x - x_k) \\ zv_k = x_k + \lambda_k \times (y_x - x_k) \end{cases},\ k = 1,2,3,...,n \tag{12}$$

不失一般性,设y_k>x_k.如图 2 所示,当 0<λ_k<1 时,有x_k<zu_k<y_k且x_k<zv_k<y_k,即zu_k,zv_k位于x_k,y_k之间的实线上;当λ_k≥1 时,有zu_k≤x_k,zv_k≥y_k,即zu_k,zv_k位于x_k,y_k两点之外的虚线上;当 0<λ_k<2 时,zu_k,zv_k被限制在总长度为 3(y_k−x_k)的线段AB内.

对于n=3 的三维搜索空间来说,如图 3 所示,新个体$\boldsymbol{zu}$,$\boldsymbol{zv}$实际上就是通过在$\boldsymbol{x}$,$\boldsymbol{y}$的坐标所围成的长方体内部及其外部的有限空间内搜索而得到的,外部长方体(实线部分)的大小受到λ_k的限制.当n>3 时,新个体的搜索空间将是一个高维长方体,因其几何意义,我们将公式(3)称为长方体交叉算子I.由于新个体是在两个父代精英个

体的附近产生,因而更容易产生高质量的解;同时,该算子将搜索空间扩展到一个包含了两个父代精英个体的长方体空间内,从而增加了解的多样性,将更有利于种群的进化.类似地,引导操作中的CCOII算子所形成的三维搜索空间是如图 4 所示的长方体.该算子利用普通个体的信息,在包含精英个体的长方体内寻找新解,即利用精英个体对普通个体进行引导得到新个体,能够很好地保证解的质量及多样性.此外,通过在一定概率下使用离散交叉算子DCO、翻转交叉算子FCO以及变异算子MO,进一步增强了种群的多样性,推动种群不断进化.

Fig.2 Geometric drawing of one dimensional CCOI

图 2 一维 CCOI 算子几何图

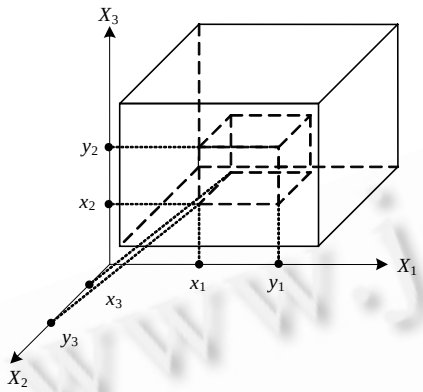

Fig.3 Geometric drawing of three dimensional CCOI

图 3 三维 CCOI 算子几何图

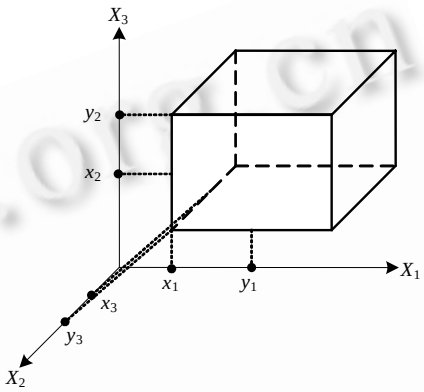

Fig.4 Geometric drawing of three dimensional CCOII

图 4 三维 CCOII 算子几何图

3 与 OEA 算法的比较

本文在部分算子的构造上受到了OEA算法[9]的启发,但在算法的设计思想上,MECA算法与OEA算法是不同的.如图 5 所示,OEA算法是在种群中产生组织,在组织中产生领导,进而由领导带领组织进行进化;如图 6 所示,MECA算法是直接在种群中产生M个精英,然后由精英来组建团队,在组建团队的过程中来推动种群进化.由于算法设计思想不同,与OEA算法相比,MECA算法具有如下几个优点:

(1) OEA 算法中,在每一代的初始阶段,个体是以组织的形式存在的,要确定每个个体属于哪个组织势必会增加所需内存或算法的运算量;MECA 算法中,在每一代的初始阶段,每个个体是相互独立的,个体进行交叉、变异等操作时是通过依次为每个精英个体选择成员来完成的,操作简便.

(2) OEA 算法中,每一代进化结束后,所有个体形成若干个组织,所以通常要先进行多次分裂操作才能进入之后的进化阶段,而分裂操作并不产生任何新个体,徒然增加了算法的复杂度;MECA 算法中,每一代进化结束后,直接将 *popEc* 和 *popCc* 两个子种群合并,使所有个体重新成为相互独立的个体,即可进入下一代,无需任何额外操作.

(3) OEA 算法中好的解即“领导”,是在局部的组织中产生的,自下而上地逐步产生全局最优解;MECA 算法中好的解即“精英个体”,是直接在全局中产生的,同时又在每个精英个体周围的局部范围内进一步搜索更好的解,兼顾了搜索机制的局部性和全局性,加快了收敛速度,从而能够在相同的评价次数的条件下得到更好的解.

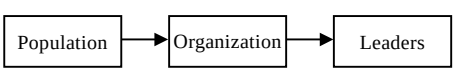

Fig.5 Simple block diagram of OEA

图 5 OEA 算法的简单框图

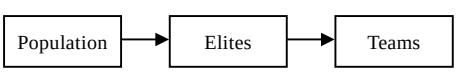

Fig.6 Simple block diagram of MECA

图 6 MECA 算法的简单框图

4 收敛性分析

定理 1. MECA算法的种群序列$\{X_k,k\geq 0\}$是有限齐次马尔可夫链,其中,k表示进化代数.

证明:本文采用的是实数编码,设所需精度为ε,则搜索空间 S 可看成是离散空间,其大小为

$$|S|=\prod_{i=1}^{n}\left\lceil(\overline{x_i}-\underline{x_i})/\varepsilon\right\rceil,$$

所以种群是有限的.而算法中采用的CCOI,DCO,FCO,CCOII,MO算子以及选拔准则均与进化代数k无关,因此, X_{k+1}仅与X_k有关,即$\{X_k,k\geq 0\}$是有限齐次马尔可夫链.证毕. □

设f是S上的适应度函数,设f^*为全局最佳值,令

$$s^*=\{\boldsymbol{x}\mid f(\boldsymbol{x})=f^*,\boldsymbol{x}\in S\} \tag{13}$$

称s^*为最优解集,则有如下定义:

定义 1. 设 $f_k=\max\limits_{\boldsymbol{x}_i\in X_k}\{f(\boldsymbol{x}_i):i=1,2,...,N\}$ 是一个随机变量序列,序列中的变量代表在时间步 k 状态中的最佳适应度.如果

$$\lim_{k\to\infty}P\{f_k=f^*\}=1 \tag{14}$$

则称算法收敛,即表明当算法迭代到足够多的代数后,群体中包含全局最优解的概率接近 1.

定理 2. 对于MECA算法的马尔可夫链序列的种群满意值序列是单调不减的,即对于任意的$k\geq 0$,有$f(X_{k+1})\geq f(X_k)$.

证明:MECA算法的协作操作中,父代的精英个体直接复制到子代精英种群中.由选拔准则I可知,只有新个体的适应度超过父代个体时,原来的精英个体才会被取代.而在引导操作的选拔准则II中,新个体只对子代普通种群进行更新,子代精英种群不受影响.以上操作及准则保证了每代的最佳个体可以保留到下一代,因而对于任意的$k\geq 0$,有$f(X_{k+1})\geq f(X_k)$,证毕. □

定理 3. MECA 算法是以概率 1 收敛的.

证明:由上所述,本算法的状态转移由马尔可夫链来描述.我们将规模为 N 的群体认为是状态空间 S 中的某个点,用$s_i\in S$表示s_i是S中的第i个状态,相应于本算法$s_i=\{x_1,x_2,\ldots,x_N\}$,显然, X_k^i 表示在第k代时种群X_k处于状态s_i,其中,随机过程$\{X_k\}$的转移概率为$p_{ij}(k)$,则

$$p_{ij}(k)=P\{X_{k+1}^j/X_k^i\} \tag{15}$$

下面给出$p_{ij}(k)$的两种特殊情况,设$I=\{i|s_i\cap s^*\neq\varnothing\}$.

I. 当 $i\in I,j\notin I$ 时,由于公式(15)和定理 2,使得

$$p_{ij}(k)=0 \tag{16}$$

即当父代中出现最优解时,无论经历多少代的进化,最优解都不会退化.

II. 当 $i\notin I,j\in I$ 时,由定理 2 可知, $f(X_{k+1}^j)>f(X_k^i)$,所以

$$p_{ij}(k)\geq 0 \tag{17}$$

在讨论了转移概率的两种特殊情况后,我们来证明公式(14).设$p_i(k)$为种群X_k处在状态s_i的概率, $p_k=\sum\limits_{i\notin I}p_i(k)$,则由马尔科夫链的性质可知:

$$p_{k+1}=\sum_{s_i\in S}\sum_{j\notin I}p_i(k)p_{ij}(k)=\sum_{i\in I}\sum_{j\notin I}p_i(k)p_{ij}(k)+\sum_{i\notin I}\sum_{j\notin I}p_i(k)p_{ij}(k) \tag{18}$$

由于

$$\sum_{i\notin I}\sum_{j\in I}p_i(k)p_{ij}(k)+\sum_{i\notin I}\sum_{j\notin I}p_i(k)p_{ij}(k)=\sum_{i\notin I}p_i(k)=p_k \tag{19}$$

因此,

$$\sum_{i\notin I}\sum_{j\notin I}p_i(k)p_{ij}(k)=p_k-\sum_{i\notin I}\sum_{j\in I}p_i(k)p_{ij}(k) \tag{20}$$

把公式(20)代入公式(18),再利用公式(16)和公式(17),则

$$0 \le p_{k+1} \le \sum_{i \in I} \sum_{j \notin I} p_i(k) p_{ij}(k) + p_k = p_k \tag{21}$$

因此,

$$\lim_{k \to \infty} p_k = 0 \tag{22}$$

又因为 $\lim_{k \to \infty} P\{f_k = f^*\} = 1 - \lim_{k \to \infty} \sum_{i \notin I} p_i(k) = 1 - \lim_{k \to \infty} p_k$ 和公式(22),可知:

$$\lim_{k \to \infty} P\{f_k = f^*\} = 1 \tag{23}$$

即所有包含在非全局最优状态中的概率收敛于 0.因此,包含在全局最优状态中的概率收敛为 1.证毕. □

5 算法仿真

本节用 15 个标准测试函数(F01~F15)来测试 MECA 算法求解无约束优化问题的性能.为便于比较,本文中的函数与文献[9]中的测试函数(F01~F15)完全一致.其中,F01~F05 是单峰函数;F06 是一个阶梯函数;F07 是一个有噪声的四次函数;F08~F15 是多峰函数,其局部最优值的个数随着问题维数的增长而增长.实验中,F01~F13 的维数设为 30,F14~F15 的维数设为 100.这样,函数具有非常多的局部极值,能够很好地测试算法的性能.具体函数参见文献[9]中的相关说明.

在第 5.1 节、第 5.2 节中,将把本文的MECA算法与OEA算法[9]进行比较,对比数据来源于文献[9].为进行公平的比较,MECA算法的终止条件也与文献[9]一致,即当评价次数超过 300 000 次时算法终止.MECA的其他参数设置如下:种群规模N=100,精英个数M=20,长方体交叉概率Pcu=0.3.

5.1 MECA的实验结果

表 1 给出了MECA算法对 15 个无约束测试函数 50 次独立运行的实验结果,包含了所求函数值的最优值、平均值、标准方差及最差值,也给出了平均评价次数,此外还列出了每个函数的全局最优值.从表中可以看出,所有函数的最优值、平均值、最差值都等于或非常接近于全局最优值,而且标准方差都相当小,其中对于F06,F09,F10,算法运行 50 次,每次都找到了全局最优值.此外需要注意的是,对于F12,将其 30 维的全局最优个体(−1,−1,…,−1)代入函数表达式,可求得对应的函数值为 1.571×10^{-32};而对于F13,将其 30 维的全局最优个体(1,1,…,1)代入函数表达式,可求得对应的函数值为 1.350×10^{-32}.这说明由于计算机截断误差的影响,对于这两个函数即使算法找到了理论的最优个体,所求得的函数值也只能达到有限精度.而本文的算法对这两个函数 50 次运行的解都分别等于上述两个值,这说明算法对于F12 和F13 也找到了全局最优解.

Table 1 Experimental results of MECA on 15 unconstrained benchmark functions over 50 trials

表 1 MECA 算法对 15 个无约束测试函数进行 50 次实验的实验结果

f	f_{min}	Best function value	Mean function value	Standard deviation	Worst function value	Mean number of function evaluations
F01	0	1.959×10^{-189}	4.228×10^{-183}	0	1.191×10^{-181}	300 059
F02	0	6.866×10^{-114}	1.845×10^{-110}	3.113×10^{-110}	1.607×10^{-109}	300 063
F03	0	3.253×10^{-105}	3.274×10^{-95}	2.313×10^{-94}	1.635×10^{-93}	300 057
F04	0	8.167×10^{-10}	5.124×10^{-2}	9.732×10^{-2}	5.037×10^{-1}	300 063
F05	0	0	7.973×10^{-2}	5.638×10^{-1}	3.987	300 055
F06	0	0	0	0	0	300 055
F07	0	3.760×10^{-6}	4.083×10^{-4}	3.800×10^{-4}	1.693×10^{-3}	300 060
F08	−12 569.5	−12 569.486 6	−12 569.486 6	7.350×10^{-12}	−12 569.486 6	300 055
F09	0	0	0	0	0	300 058
F10	0	0	0	0	0	300 052
F11	0	0	3.844×10^{-3}	7.130×10^{-3}	2.464×10^{-2}	300 066
F12	0	1.571×10^{-32}	1.571×10^{-32}	5.529×10^{-48}	1.571×10^{-32}	300 068
F13	0	1.350×10^{-32}	1.350×10^{-32}	1.106×10^{-47}	1.350×10^{-32}	300 062
F14	−99.60	−99.004 382 2	−98.709 489 1	1.450×10^{-1}	−98.243 522 6	300 059
F15	−78.332 36	−78.332 331 4	−78.332 331 4	1.005×10^{-13}	−78.332 331 4	300 061

5.2 MECA算法与OEA算法的比较

从表 2 的结果可以看出,对于 F14,MECA 算法略差于 OEA 算法;对于 F06,两种方法都找到了全局最优值;对于 F08 和 F15,两种方法的结果非常接近.除此之外,对于其他 11 个函数,MECA 算法的结果都优于 OEA 算法.其中,对于 F01~F03,MECA 算法的精度比 OEA 算法提高了几十甚至上百个数量级;对于 F09 和 F10,MECA 算法每次都找到了全局最优解,而 OEA 算法则没有;对于 F12 和 F13,如第 5.1 节所述,MECA 算法事实上也已经每次都找到了全局最优解,但 OEA 算法则没有.

Table 2 Comparison between MECA and OEA over 50 trials

表 2 MECA 算法与 OEA 算法 50 次实验结果的比较

f	$f_{\min}$	Mean function value		Standard deviation		Mean number of function evaluations	
		MECA	OEA	MECA	OEA	MECA	OEA
F01	0	4.228×10^{-183}	2.481×10^{-30}	0	1.128×10^{-29}	300 059	300 017
F02	0	1.845×10^{-110}	2.068×10^{-13}	3.113×10^{-110}	1.440×10^{-12}	300 063	300 014
F03	0	3.274×10^{-95}	1.883×10^{-9}	2.313×10^{-94}	3.726×10^{-9}	300 057	300 016
F04	0	5.124×10^{-2}	8.821×10^{-2}	9.732×10^{-2}	2.356×10^{-2}	300 063	300 018
F05	0	7.973×10^{-2}	0.227	5.638×10^{-1}	0.941	300 055	300 016
F06	0	0	0	0	0	300 055	300 017
F07	0	4.083×10^{-4}	3.297×10^{-3}	3.800×10^{-4}	1.096×10^{-3}	300 060	300 016
F08	−12 569.5	−12 569.486 6	−12 569.486 6	7.350×10^{-12}	5.555×10^{-12}	300 055	300 019
F09	0	0	5.430×10^{-17}	0	1.683×10^{-16}	300 058	300 019
F10	0	0	5.336×10^{-14}	0	2.945×10^{-13}	300 052	300 018
F11	0	3.844×10^{-3}	1.317×10^{-2}	7.130×10^{-3}	1.561×10^{-2}	300 066	300 020
F12	0	1.571×10^{-32}	9.207×10^{-30}	5.529×10^{-48}	6.436×10^{-31}	300 068	300 019
F13	0	1.350×10^{-32}	4.323×10^{-19}	1.106×10^{-47}	2.219×10^{-18}	300 062	300 015
F14	−99.60	−98.709 489 1	−99.502 404 2	1.450×10^{-1}	2.526×10^{-2}	300 059	300 017
F15	−78.332 36	−78.332 331 4	−78.332 331 4	1.005×10^{-13}	2.804×10^{-11}	300 061	300 018

5.3 MECA算法与CCGA算法和CPSO算法的比较

CCGA算法和CPSO算法的数据结果来自文献[5],为方便比较,MECA算法的终止条件与文献[5]一致,即当评价次数超过 200 000 次时算法终止,MECA算法的其他参数设置不变.所给出的结果是对各函数 50 次独立实验得到的最优值的平均值.文献[5]中实验所用的 5 个函数f_0~ f_4分别对应本文中的F05,F03,F10,F09,F11.文献[5]在仿真中给出了CPSO算法的 4 种方法的结果,即CPSO-S,CPSO-S_6,CPSO-H,CPSO-H_6,其中CPSO-S_6和CPSO-H_6表示分裂因子K取 6,而CPSO-S和CPSO-H表示分裂因子K取n的特殊情况,n代表解向量的维数.具体的方法及其他参数设定请参看原文.对每个函数,文中给出了每种方法当每个粒子群的粒子数分别为 10,15,20 时的共 12 个结果,表 3 列出了最好结果及最差结果,并在相邻的“Algorithm”一列里注明了得到该结果的方法,小括弧里标注的是此时每个粒子群的粒子数.

Table 3 Comparison between MECA, CCGA and CPSO over 50 trials

表 3 MECA 算法与 CCGA 算法和 CPSO 算法 50 次实验结果的比较

f	MECA	CCGA	CPSO			
			Best results	Algorithm (number of particles)	Worst result	Algorithm (number of particles)
F05 (f_0)	6.43×10^{-1}	3.80×10^{0}	1.94×10^{-1}	CPSO-H_6 (10)	2.47×10^{0}	CPSO-S_6 (15)
F03 (f_1)	1.23×10^{-64}	1.38×10^{2}	2.55×10^{-128}	CPSO-S (10)	1.20×10^{-4}	CPSO-S_6 (20)
F10 (f_2)	0	9.51×10^{-2}	2.78×10^{-14}	CPSO-H (10)	5.42×10^{-5}	CPSO-S_6 (20)
F09 (f_3)	0	1.22×10^{0}	0	CPSO-S, CPSO-H	1.47×10^{0}	CPSO-H_6 (10)
F11 (f_4)	3.94×10^{-3}	2.20×10^{-1}	1.86×10^{-2}	CPSO-H (20)	8.95×10^{-2}	CPSO-S_6 (20)

从表 3 中可以看出,本文的 MECA 算法对于所有 5 个函数都远远优于 CCGA 算法;对于 F10,F11,MECA 算法优于 CPSO 算法的最好结果;对于 F05,F09,与最好结果相当;唯有 F03,MECA 算法不如其最好结果,但远远优于其最差结果.值得注意的是,CPSO 算法的结果是在不同的方法以及使用不同粒子数的情况下得到的.虽然文中最后给出了指导性的结论,说明了哪个方法更适合哪种类型的函数,然而在实际应用中,目标函数是多变的,其类型也是难以预知的,这无疑会给该算法的实际使用增加一些困难.相比之下,本文算法在统一的参数设置下

对不同函数都得出了较好的结果,因而更易于使用.

5.4 MECA算法收敛速度的仿真

在第 5.1 节、第 5.2 节的实验中,为了方便与文献[9]进行比较,将算法终止的条件设为评价次数超过 300 000 次时算法终止,但实际上对于 F06,F08~F10,F12,F13,F15 来说,用更少的评价次数即可收敛到所给精度.这里对这些函数收敛时实际所需的评价次数进行了统计,表 4 给出了 10 次独立运行的结果.此外,给出了 F9 和 F13 的收敛曲线.本节中,算法的终止条件改为当最优值达到指定的全局最优值时算法终止,其他参数不变.

由表 4 可以看出,对于以上 7 个函数,算法经过较少的评价次数即可收敛,从而达到或接近全局最优值.图 7、图 8 分别给出了 F09,F13 进行 10 次独立运行的收敛曲线,其中,虚线表示 10 次独立运行的结果,实线表示 10 次结果的平均值.注意,对于 F09,当算法收敛到 0 时,在对数坐标中无法显示对应的值.从图 7、图 8 可以看出,算法能够以很快的速度收敛到全局最优值.

Table 4 Mean number of function evaluations when MECA becomes convergent

表 4 MECA 算法收敛所需的平均函数评价次数

f	F06	F08	F09	F10	F12	F13	F15
Function value	0	−12 569.486 6	0	0	1.571×10^{-32}	1.350×10^{-32}	−78.332 331 4
Mean number of function evaluations	6 852	20 249	32 171	77 026	56 547	59 233	46 937

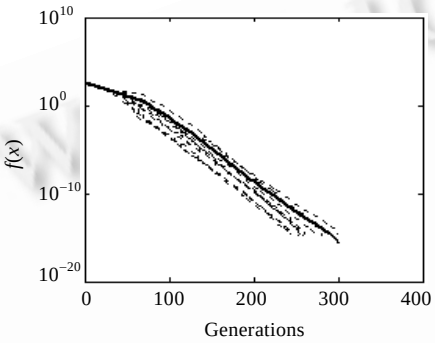

Fig.7 Convergence curve of F09

图 7 F09 的收敛曲线

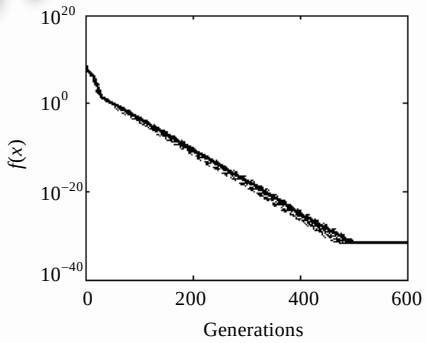

Fig.8 Convergence curve of F13

图 8 F13 的收敛曲线

5.5 MECA算法运行时间的仿真

不同作者的算法,其算法机制以及仿真环境都不尽相同,一般来说,使用相同的评价次数来比较不同算法的性能是相对比较公平和简便的方法.为了更直观地显示本文算法的时间复杂度,我们在相同的仿真环境下分别实现了本文的MECA算法、OEA算法以及标准遗传算法SGA,来比较 3 种算法达到相同评价次数时的运行时间.具体的仿真环境如下:CPU:Intel Pentium 4,3.20GHz;内存:1GB;操作系统:Windows XP Professional;开发工具:Microsoft Visual C++ 2005.各算法的终止条件是当评价次数超过 300 000 次时算法终止,MECA算法的参数设置与第 5.1 节、第 5.2 节相同.OEA算法的参数设置与文献[9]一致.为了进行公平的比较,SGA算法也采用实数编码,种群数与MECA算法相同,也取为 100,交叉概率p_c为 0.8,变异概率p_m为 0.01,交叉策略使用常规的算术交叉,变异策略类似于二进制编码中的变异策略,即依据变异概率对个体的某一维变量进行变异,采取轮盘赌选择及最优保留策略.各算法均独立运行 50 次.表 5 给出了 4 个 30 维函数F02,F06,F09,F11 以及两个 100 维函数 F14,F15 的运行结果.

从表 5 中可以看出,当达到同样的评价次数时,MECA 算法最快,SGA 算法次之,OEA 算法最慢.MECA 算法比 OEA 算法快,原因已经在第 3 节进行了详细的分析.SGA 对于高维函数的优化显得无能为力,并且在轮盘赌选择中,种群的所有个体都要计算出选择概率;此外,判断选择哪些个体的次数也与种群规模相等,这些操作都增加了 SGA 算法的运行时间.本文的 MECA 算法的操作算子虽然看起来比较多,但各算子处于算法的不同分支

中,每次产生新个体时,只会使用其中的 1~2 个算子;算子类型多,增加了种群多样性,但并没有因此而增加算法的时间复杂度.此外,在为各个精英个体选取成员以产生新个体时采取了比较简单的选择机制,也节省了算法的运行时间.

Table 5 Comparison of runtime between MECA, OEA and SGA over 50 trials

表 5 MECA 算法与 OEA 算法和 SGA 算法进行 50 次实验的运行时间的比较

f	Mean function value			Mean number of function evaluations			Mean runtime (s)		
	MECA	OEA	SGA	MECA	OEA	SGA	MECA	OEA	SGA
F02	6.8×10^{-114}	9.59×10^{-37}	2.05×10^{0}	300 049	300 017	300 041	1.09	5.75	2.44
F06	0	0	5.11×10^{1}	300 044	300 017	300 044	1.40	5.91	3.60
F09	0	3.62×10^{-15}	2.42×10^{1}	300 045	300 019	300 049	1.48	6.08	2.91
F11	3.20×10^{-3}	2.62×10^{-2}	1.41×10^{0}	300 044	300 016	300 039	1.72	6.36	3.17
F14	−98.881 43	−99.300 66	−70.255 63	300 048	300 017	300 045	9.23	16.08	12.09
F15	−78.332 33	−78.332 33	−67.702 33	300 048	300 020	300 045	6.29	13.07	9.25

5.6 MECA算法的参数及算子分析

下面先给出平均成功率的定义,然后在平均成功率的准则下分析参数 *Pcu* 和 *M* 对算法性能的影响.

定义 2. 如果一次运行在规定条件下的结果满足公式(24),则此次运行称为成功的,否则称为失败的.

$$\begin{cases} |F^*-F_{best}|<\varepsilon\cdot|F^*|, & F^*\neq 0\\ |F_{best}|<\varepsilon, & F^*=0 \end{cases} \tag{24}$$

其中,F_{best}表示此次运行所得到的最优值,F^*为全局最优值.设M次运行中有M_s次成功,则平均成功率为$R_{as}=M_s/M$.显然,平均成功率是对算法在规定条件下搜索全局最优解的能力的概率估计,其值越大,表明算法的性能越稳定,寻优能力越强.这里取$\varepsilon=10^{-5}$.

这里选择了两个单峰函数 F2,F3,以及两个双峰函数 F8 和 F11 进行研究.

5.6.1 参数 *Pcu* 对 MECA 算法性能的影响

本节中,长方体交叉概率*Pcu*在区间[0.1,1]内以步长 0.1 采样,得到 10 个*Pcu*参数,在每个*Pcu*参数下进行 50 次实验(其他参数及终止条件与第 5.1 节、第 5.2 节相同),然后计算该参数对应的平均成功率,如此便得到R_{as}与*Pcu*的关系,如图 9(a)所示.从图 9(a)可以看出,对于F2,F3,F8 来说,当*Pcu*在 0.1~0.9 之间时,成功率都为 1;而对于F11,当*Pcu*在 0.1~0.4 之间时,成功率较高.此外,图 9(b)给出了每个*Pcu*参数对应的最优函数值,以便更直接地观察该参数对算法性能的影响.从图 9(b)可以看出,F8 对*Pcu*不敏感,F11 的变化幅度也较小,而F2,F3 的变化幅度较大,当*Pcu*在 0.1~0.4 之间时,二者的精度会非常高.所以从总体上看,当*Pcu*在 0.1~0.9 之间这样大的范围内时,算法对多数函数都可以达到较高的精度,尤其是当*Pcu*取值在 0.1~0.4 之间时,算法的搜索精度最高.因而在前面的实验中取*Pcu*为 0.3.

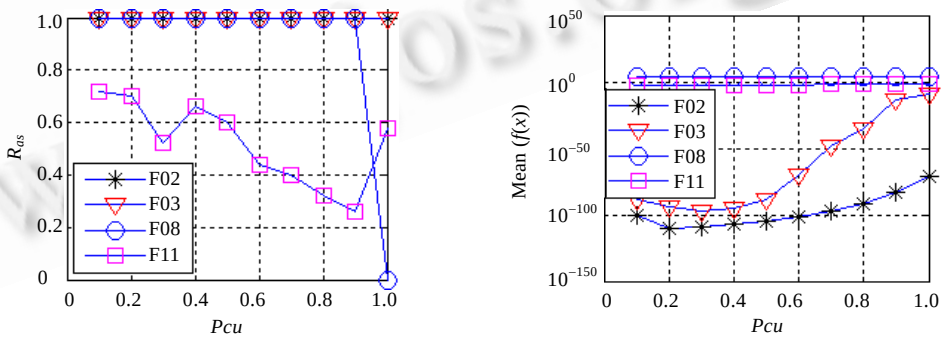

Fig.9 Influence of *Pcu* on MECA 算法

图 9 *Pcu* 对 MECA 的影响

5.6.2 参数 M 对 MECA 算法性能的影响

本节中,精英的个数M在区间(0,100)内进行非均匀采样,具体取值为[3,5,10,15,20,30,40,50,70,90],得到 10 个M参数,在每个M参数下进行 50 次实验(其他参数及终止条件与第 5.1 节、第 5.2 节相同),然后计算该参数对应的平均成功率,如此便得到R_{as}与M的关系,如图 10(a)所示.从图 10(a)可以看出:对于F2,F3,F8 来说,当M在 3~90 之间时,成功率都为 1;而对于F11,成功率随M的增加而上升,当M在 50~90 之间时,成功率较高,尤其是当M取 90 时,成功率可达到 1.图 10(b)给出了每个M参数对应的最优函数值.从图 10(b)可以看出,F8 对M不敏感,F11 的变化幅度也较小.但需要注意的是,当M取 90 时,F11 每次均可达到全局最优值 0.F2,F3 的变化幅度较大,当M在 10~20 之间时,二者的精度会非常高.所以从总体上看,当M在 3~90 之间这样大的范围内时,算法对多数函数都可以达到较高的精度.具体来看,对于单峰函数,M取得小一些效果更好;而对于多峰函数,M取得大一些效果更好.为了兼顾所有的函数,本文在前面的实验中取M为 20.

M 是每代种群中前 M 个最优个体的数目,当 M 取值较小时,算法会在较少的几个精英个体附近进行搜索,因而对于单峰函数,能够较快地找到最优解;但对于多峰函数,由于搜索空间有限,则可能会陷入局部最优.M 取值变大时,算法则会在更多的精英个体附近进行搜索,这样,算法收敛的速度会降低.因而对于单峰函数,在同样的评价次数下找到的最优解会变差;但对于多峰函数,由于搜索范围加大,陷入局部最优的可能性降低,因而搜索到的最优解会变好.

为便于使用对数坐标观察,图 9(b)和图 10(b)中对函数的均值取了绝对值.

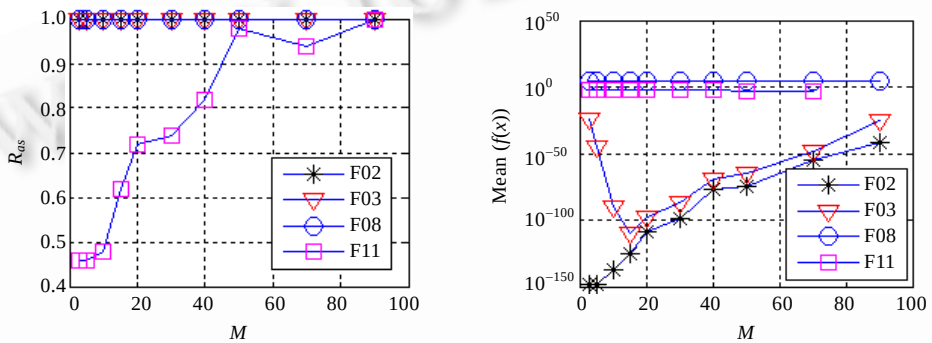

Fig.10 Influence of M on MECA

图 10 M 对 MECA 的影响

5.6.3 DCO,FCO,MO 算子对 MECA 算法性能的影响

在较好的算法机制的保证下,为进一步增加种群的多样性,本文还使用了 DCO,FCO,MO 这几个算子.其中,DCO 算子实际上属于离散交叉中的两点交叉,用于产生新的子代个体,而离散交叉所产生的子代个体位于由两个父代个体的元素所定义的超立方体的顶点上;FCO 算子是对 DCO 算子的改进,对于理论最优值在各维上都相同的函数,该算子使得某些维上得到的好的结果能够快速传播到其他维上,能够起到加快收敛的作用;MO 算子是一种变异算子,在精英个体的基础上产生新个体,即在精英个体的引导下进行变异,使算法能够跳出局部最优,同时又能在一定程度上避免过于随机的变异所导致的退化.为验证几个算子对算法所起到的推动作用,下面分别用 CON0,CON1,CON2 和 CON3 来表示以下 4 种情况:本文的 MECA 算法、MECA 算法中剔除 DCO 算子、MECA 算法中剔除 FCO 算子、MECA 算法中剔除 MO 算子.在这 4 种情况下分别对两个单峰函数 F1,F2,以及两个多峰函数 F10,F14 进行 10 次独立实验.算法的终止条件仍然是当评价次数超过 300 000 次时算法终止,参数设置与第 5.1 节、第 5.2 节相同.

从表 6 的 4 种情况依次可以看出,3 种算子中 DCO 算子对算法性能的影响最小,但 DCO 算子确实在一定程度上提高了算法的精度;FCO 算子对于理论最优值在各维上都相同的函数,如表 6 中前 3 个函数,有较强的加快

收敛作用,而对于理论最优值在各维上不同的函数,如 F14,由于算法中选拔机制的保证,该算子并不会对算法产生过强的抑制作用;MO 算子作为一种变异算子,对于算法跳出局部最优起着主要作用,去掉该算子,算法性能会大幅度下降.因而只有几种算子综合使用,才能使算法达到最理想的效果.

Table 6 Comparison of mean function value under four conditions over 10 trials

表 6 4 种情况下进行 10 次实验的函数值平均值的对比

f	f_{min}	Mean function value			
		CON0	CON1	CON2	CON3
F01	0	2.1566×10^{-184}	4.7444×10^{-175}	6.2883×10^{-61}	1.0718×10^{-65}
F02	0	4.2019×10^{-110}	3.3740×10^{-103}	8.9025×10^{-21}	7.5862×10^{-40}
F10	0	0	0	1.0658×10^{-14}	2.804
F14	−99.60	−98.720 658 8	−97.658 115 6	−98.737 508 8	−82.128 619 0

6 结 论

本文提出了一种求解数值优化问题的进化算法——*M*-精英协同进化算法,详细分析了主要算子的机理,并从理论上证明了其具有全局收敛性.实验中对无约束函数优化的测试以及与 OEA,CCGA,CPSO 这些方法的比较结果表明,MECA 算法具有较强的寻优能力,总体的求解质量要优于所对比的方法.由于受到精英策略的启发,本文算法在进化中更强调精英个体的作用,使得算法的收敛速度加快,但同时也使得算法对于某些函数容易陷入局部最优,如函数 F04,F05,F07,F11,但是其平均值仍然略优于所对比的其他几种方法;对于 F14,本文结果不如 OEA 方法,说明本文算法在脱离局部最优的性能方面仍有不足.对算法参数的实验分析表明,MECA 算法当参数在较大范围内发生变化时都能够取得很好的解,对参数不敏感.对参数 M 的分析表明,对于单峰函数,M 取较小的值效果更好;而对于多峰函数,M 取较大的值能够获得更好的解.因此,下一步我们将改进算法,使其具有自适应调节能力,并利用该算法解决其他优化问题,如组合优化、多目标优化等.

致谢 在此,我们向对本文工作给予支持和建议的同行表示感谢.

References:

[1] Yao X, Liu Y, Lin GM. Evolutionary programming made faster. IEEE Trans. on Evolutionary Computation, 1999,3(2):82−102.

[2] Leung YW, Wang YP. An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Trans. on Evolutionary Computation, 2001,5(1):41−53.

[3] Zhang L, Zhang B. Good point set based genetic algorithm. Chinese Journal of Computers, 2001,24(9):917−922 (in Chinese with English abstract).

[4] Potter MA, De Jong KA. A cooperative coevolutionary approach to function optimization. In: Davidor Y, Schwefel HP, Männer R, eds. Proc. of the Parallel Problem Solving from Nature—PPSN III, Int'l Conf. on Evolutionary Computation. LNCS 866, Berlin: Springer-Verlag, 1994. 249−257.

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

[6] Cao XB, Luo WJ, Wang XF. A co-evolution pattern based on ecological population competition model. Journal of Software, 2001, 12(4):556−562 (in Chinese with English abstract). http://www.jos.org.cn/ch/reader/view_abstract.aspx?file_no=20010410&flag=1

[7] Hu SC, Xu XF, Li XY. A virus coevolution genetic algorithm for project optimization scheduling. Journal of Software, 2004, 15(1):49−57 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/49.htm

[8] Tan KC, Yang YJ, Goh CK. A distributed cooperative coevolutionary algorithm for multiobjective optimization. IEEE Trans. on Evolutionary Computation, 2006,10(5):527−549.

[9] Liu J, Zhong WC, Jiao LC. An Organizational Evolutionary Algorithm for Numerical Optimization. IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics, 2007,37(4):1052−1064.

[10] Ahn CW, Ramakrishna RS. Elitism-Based compact genetic algorithms. IEEE Trans. on Evolutionary Computation, 2003,7(4): 367−385.

附中文参考文献:

[3] 张铃,张钹.佳点集遗传算法.计算机学报,2001,24(9):917−922.

[6] 曹先彬,罗文坚,王煦法.基于生态种群竞争模型的协同进化.软件学报,2001,12(4):556−562. http://www.jos.org.cn/ch/reader/view_abstract.aspx?file_no=20010410&flag=1

[7] 胡仕成,徐晓飞,李向阳.项目优化调度的病毒协同进化遗传算法.软件学报,2004,15(1):49−57. http://www.jos.org.cn/1000-9825/15/49.htm

慕彩红(1978－),女,河南武陟人,博士生,讲师,CCF 高级会员,主要研究领域为进化计算,通信信息处理.

焦李成(1959－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为智能信息处理,非线性理论.

刘逸(1976－),男,博士生,讲师,主要研究领域为进化计算,通信信息处理.