计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
49(8): 1753-1761, 2012

聚类佳点集交叉的约束优化混合进化算法

龙 文[1,2] 梁昔明[2] 徐松金[3] 陈 富[2]
[1](贵州财经大学贵州省经济系统仿真重点实验室 贵阳 550004)
[2](中南大学信息科学与工程学院 长沙 410083)
[3](铜仁学院数学与计算机科学系 贵州铜仁 554300)
(dragon638@126.com)

A Hybrid Evolutionary Algorithm Based on Clustering Good-Point Set Crossover for Constrained Optimization

Long Wen[1,2], Liang Ximing[2], Xu Songjin[3], and Chen Fu[2]
[1](*Guizhou Key Laboratory of Economics System Simulation*, *Guizhou University of Finance and Economics*, *Guiyang* 550004)
[2](*School of Information Science and Engineering*, *Central South University*, *Changsha* 410083)
[3](*Department of Mathematics and Computer Science*, *Tongren University*, *Tongren*, *Guizhou* 554300)

Abstract A hybrid evolutionary algorithm based on multi-parent crossover of clustering good-point set and adaptive constraint-handling technique is proposed in this paper for solving constrained optimization problems. As for search mechanism, it utilizes good-point set to construct the initialization population that is scattered uniformly over the entire search space in order to maintain the diversity. The individuals of population are divided into several sub-populations according to the similarity of the two parents. The parents are selected randomly from the several sub-populations to arrange the crossover operation. The crossover operator can effectively make use of the information carried by the parents and generate representation offspring in order to maintain and increase the diversity of population. In addition, a local search scheme is introduced to enhance the local search ability and speed up the convergence of the proposed algorithm. As for constraint-handling technique, a new individual comparison criterion is proposed, which can adaptively select different individual comparison criterion according to the proportion of feasible solution in current population. The proposed algorithm is tested on 15 well-known benchmark functions, and the empirical evidence shows its effectivity.

Key words constrained optimization; evolutionary algorithm; clustering; adaptive; good point set

摘 要 提出一种基于聚类佳点集多父代交叉和自适应约束处理技术的混合进化算法用于求解约束优化问题.新算法的主要特点是:在搜索机制方面,利用佳点集方法构造初始化种群,使个体能够均匀地分布在整个搜索空间.然后根据父代个体的相似度将种群个体进行聚类分析,从聚类中随机选择个体进行佳点集多父代交叉操作,利用多个父代个体所携带的信息产生新的具有代表性的子代个体,能够维持和增加种群的多样性.另外,引入局部搜索策略以提高算法局部搜索能力和收敛速度.在约束处理技术上,新算法引入了一个自适应约束处理技术,即根据当前种群中可行解的比例自适应选择不同的个体比较准则.通过15个标准测试函数验证了新算法的有效性.

收稿日期:2010-10-21;**修回日期**:2012-03-07
基金项目:国家自然科学基金项目(60874070,61074069),湖南省研究生科研创新项目(CX2009B038);贵州财经大学引进人才科研启动项目

关键词 约束优化；进化算法；聚类；自适应；佳点集

中图法分类号 TP18；TP301

约束优化问题(constrained optimization problems，COPs)是科学研究、工程应用领域经常会遇到的一类数学规划问题.不失一般性，约束优化问题(最小化问题)可描述为

$$\begin{aligned} &\min \quad f(\boldsymbol{x}), \boldsymbol{x}=(x_1,x_2,\cdots,x_s)\in\mathbb{R}^s,\\ &\text{s.t.} \quad g_j(\boldsymbol{x})\leqslant 0, j=1,2,\cdots,l,\\ &\qquad g_j(\boldsymbol{x})=0, j=l+1,\cdots,m, \end{aligned} \tag{1}$$

其中，$\boldsymbol{x}\in S$ 为决策变量，S 为决策空间.通常 S 为 $\mathbb{R}^s$ 中的 s 维长方体：$l_i\leqslant x_i\leqslant u_i$，$l_i$，$u_i$ 为常数，$i=1,2,\cdots,s$，$f(\boldsymbol{x})$ 为目标函数，$g_j(\boldsymbol{x})\leqslant 0$ 表示第 j 个不等式约束条件，$g_j(\boldsymbol{x})=0$ 表示第 j 个等式约束条件，l 是不等式约束条件的个数，$(m-l)$ 为等式约束条件的个数.

遗传算法(genetic algorithm，GA)作为一种基于选择、交叉、变异机制的随机优化搜索算法，具有不依赖于问题的梯度信息、隐并行性、全局收敛的特点.结合合适的约束处理技术，遗传算法是求解约束优化问题的一种有效方法[1-3].虽然遗传算法在约束优化中取得了较好的效果，但是在其本身搜索机制上，传统的遗传算法也存在许多的缺陷，如早熟收敛、局部搜索能力差等.因此，研究者提出了大量的改进遗传算法.近年来，有些研究者尝试将优化实验设计方法，如佳点集、均匀实验设计、正交实验设计等嵌入到遗传算法中，取得了不错的搜索结果.张铃等人[4]利用数论中的佳点集理论和方法改进交叉算子，提出一种有效的佳点集遗传算法.Wang 等人[5]利用均匀表设计两个父代个体的交叉算子，提出了一种基于水平集进化和拉丁方的进化算法.江中央、蔡自兴等人[6]根据父代个体的相似度自适应调整正交表因素，提出一种自适应正交混合遗传算法.

在用遗传算法求解约束优化问题时，除了自身的搜索能力以外，如何处理好约束条件也是一个关键问题.目前常用的约束处理技术有惩罚函数法、可行解优于不可行解法等.惩罚函数法是处理约束最常用的方法之一，但很难确定合适的惩罚因子，算法的性能强烈地依赖于参数的选择[7].可行解优于不可行解法在搜索时没有充分利用不可行解的信息，尤其是求解一些可行域很小或全局最优解位于可行域边界上的问题时搜索效率不高[8].近年来，将约束优化问题转化为多目标优化问题来处理受到了极大的关注，并涌现出大量的研究成果[9-10].这类算法的主要特点是将问题的约束条件作为一个或多个目标来看待.

考虑到遗传算法自身的搜索能力和约束处理技术对求解约束优化问题的性能影响，在搜索策略上，本文提出了聚类佳点集多父代交叉操作，以维持和增加种群的多样性，避免算法出现早熟收敛.以一定概率对每个聚类进行非均匀变异操作.同时，利用单形交叉算子对每个聚类进行局部搜索，以提高算法的收敛速度.在约束处理技术上，相似于文献[11]，本文根据群体中可行解的比例将种群的进化过程分为 3 个阶段，在不同进化阶段选取不同的个体比较和选择策略，从而有效地引导群体不断向最优解逼近.对 15 个标准测试函数的数值实验结果表明了新算法的有效性.

1 基于聚类佳点集的约束优化混合进化算法

1.1 种群初始化

在求解优化问题前，我们对问题的全局最优解处在什么位置完全一无所知，如果随机初始化种群个体可能不具有代表性，不利于搜索到全局最优解，可能导致要增加迭代次数或种群大小来达到最优解或近似最优解，这势必会增加算法的计算量.而我们所期望的是，在初始化种群时个体应尽可能地均匀分布在整个搜索空间中，这样有利于算法在整个搜索空间内进行均匀搜索，才能以较优的方式快速逼近最优解.

在相同取点个数的条件下，佳点序列要比其他

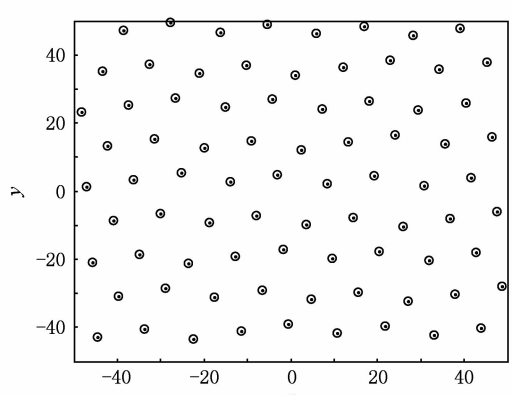


Fig. 1 80 initial individuals with good point set method.

图 1 佳点集产生的 80 个二维初始种群分布

方法选取的点序列更均匀,更逼近被积函数曲线.佳点集与决策变量同维,其规模只和所需的点数有关,这样用佳点集法求到的子集最能代表当前群体的性能.因此,本文采用佳点集方法生成初始种群,能克服正交设计方法的不足,从而保证了初始群体的多样性.图1是采用佳点集方法产生的规模为80的二维初始种群分布.

从图1可以很清楚地看出,与随机方法相比,佳点集方法产生的初始种群分布均匀,没有重叠点,具有较好的多样性.

1.2　聚类佳点集多父代交叉

交叉算子是遗传算法搜索整个决策空间的重要操作,它通过组合父代个体的信息形成新的个体.常见的交叉算子是在子代个体的样本空间中随机产生新的子代个体,搜索具有一定的盲目性,在一定程度上降低了算法的搜索效率.利用佳点集方法安排父代个体的交叉操作,在子代个体的样本空间中产生具有代表性的子代个体,以提高搜索效率.文献[4]将佳点集方法引入到遗传算法中,设计了一种基于二进制编码的佳点集交叉算子,但针对高维、高精度优化问题时会出现个体编码长度太长而导致计算量过大.文献[12]提出了一种基于佳点集的实数编码遗传算法,设计了新的佳点集交叉算子,结合约束处理技术用于求解约束优化问题.上述佳点集遗传算法虽然在求解各类优化问题时取得了不错的结果,但是在设计交叉算子时,无论父代个体所确定的空间区域大小都产生相同数量的子代个体.

设从当前种群中选取两个个体 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 作为父代个体,设 $\boldsymbol{x}_i=(x_{i1},x_{i2},\cdots,x_{is})$,$\boldsymbol{x}_j=(x_{j1},x_{j2},\cdots,x_{js})$,$\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 共同确定了一个有界闭区间 $D=[\boldsymbol{a},\boldsymbol{b}]\subseteq\mathbb{R}^s$:

$$\begin{cases} a_k = \min(x_{ik},x_{jk}) \\ b_k = \max(x_{ik},x_{jk}) \end{cases},\ k=1,2,\cdots,s, \tag{2}$$

D 是 $\mathbb{R}^s$ 上的超长方体,即

$$U([\boldsymbol{a},\boldsymbol{b}]) = \prod_{i=1}^{s}(b_i-a_i). \tag{3}$$

对给定的两个父代个体,如果一个交叉算子能组合出的新的个体数目越多,说明它的组合能力越强,很明显交叉算子的组合能力直接反映了算子所能搜索决策空间的大小.父代个体通过佳点集交叉产生的子代个体只能在其所确定的超长方体内,子代个体之间的距离小于父代个体对之间的距离.它是通过切割父代个体确定的超长方体获得新的基因片段,然后重组这些基因片段来生成新的子代个体.如图2所示:

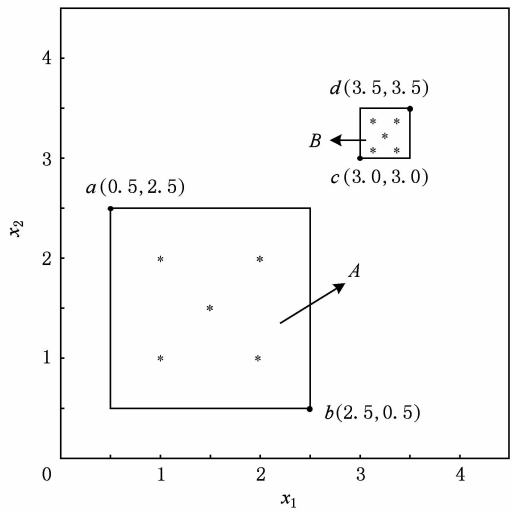


Fig. 2　Distribution of the offspring generated by using good point set crossover operation.

图2　佳点集交叉操作产生的子代个体

从图2可以看出,个体 c 和 d 的距离很近,它们所确定的空间区域相对较小,如果采用佳点集方法安排它们进行交叉,则会产生很多相似度高的子代个体,种群的多样性会变差.这种现象在群体进化后期更为明显.因为在群体进化后期,所有的个体不断向适应度高的方向聚集,个体之间的空间距离越来越近.

为了减少相似度高的父代个体进行佳点集交叉操作和增加种群的多样性,本文提出了一种聚类佳点集多父代交叉操作,即根据群体中个体间的空间距离不同进行聚类分析,将种群分为多个子种群,然后从每个子种群中随机选择个体来进行佳点集交叉操作.

设 $\boldsymbol{x}_1=(x_{1,1},x_{1,2},\cdots,x_{1,s})$,$\boldsymbol{x}_2=(x_{2,1},x_{2,2},\cdots,x_{2,s})$ 为参与交叉操作的一对父代个体.为了描述方便,我们作如下定义:

定义1. 个体 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 之间的空间距离 d 为

$$d=\sqrt{\sum_{i=1}^{s}(x_{1i}-x_{2i})^2}. \tag{4}$$

根据个体的空间距离先对群体进行聚类,将种群分割成互不相交的局部领域,使每个子种群仅覆盖搜索空间一个较小的邻域.在确定好种群聚类后,针对每一个子种群,随机选择个体进行佳点集交叉操作,从而更加增加了种群的多样性.

1.3　聚类局部搜索

为了提高算法的局部搜索能力和收敛速度,本文引入局部搜索策略.先对群体中个体根据其空间距离进行聚类,将种群分为若干个子种群.对每个

子种群，即局部邻域，采用单形交叉（SPX）算子[11]对其进行局部搜索，以加强算法的局部搜索能力和提高算法的收敛速度. 单形交叉算子使算法在进化初期有较强的全局搜索能力，而在进化后期有较强的局部搜索能力. 聚类单形交叉局部搜索策略将种群分割成互不相交的局部邻域，以提高单形搜索的局部收敛速度，同时对不相交的局部邻域进行并行搜索，能够充分地利用搜索空间的局部信息.

1.4 约束处理技术

现有的约束处理技术存在的主要问题是如何设计合理的个体比较准则和选择准则. 因此，本文提出一种自适应约束处理技术，即根据当前种群中可行解的比例而选择不同的个体比较与选择准则. 类似于文献[11]，本文将种群的进化过程分为 3 个阶段：1)种群中只有不可行解；2)种群中既有可行解又有不可行解；3)种群中只有可行解.

1) 当种群中可行解比例 $f_p=0$ 时，也就是当前种群中只存在不可行解时，说明当前种群中个体离可行域还有一定的距离. 由于对全局最优解没有任何先验知识，因此，在这个阶段中，个体的可行性比目标函数值最小化更重要. 其主要目标就是如何引导群体从不同方向进入可行域或向可行域边界靠近.

蔡自兴等人[3]和 Venkatraman 等人[13]建议在这个阶段完全不考虑个体的目标函数值，只按照个体的约束违反程度来选择. 但这样做缺乏找到或接近可行解的机会，这对于可行域高度稀疏和搜索空间离散的问题尤为关键. Wang 等人[11]提出一种分层选择非劣个体进入下一代种群的方案，取得了较好的效果. 本文提出一种带偏好策略的非劣个体比较和选择准则.

在多目标优化过程中，群体中的主要信息均包含在非劣个体中，因此，我们感兴趣的仅仅是非劣个体，从而避免了对群体其他个体的操作所带来的计算花销. 但是，对于非劣个体，用 Pareto 优超关系无法判断孰优孰劣. 文献[11]认为，没有引入偏好信息的多目标优化方法可能并不像人们所想象的那样有效. 因此，本文提出一种带有偏好策略的非劣个体选择和比较准则：关于约束优化问题，如果一个个体远离可行域边界，它对寻优过程是没有任何意义的. 因而我们只关心具有低约束违反度的非劣个体，这对寻优过程而言其实就是一种搜索偏好. 把当前种群中的所有非劣个体按约束违反度升序排序，选取前 N（N 为种群规模）个非劣个体直接进入下一代种群. 如果当前种群中的非劣个体数目为 N'，小于种群规模 N，先直接将 N' 个非劣个体进入下一代群体，然后再从剩下的劣于个体中随机选择（$N-N'$）个进入下一代种群. 由 Pareto 优超定义可知，与文献[3]和文献[13]中只根据个体的约束违反度选择而完全抛弃目标函数值的方法不同的是，本文方法能保证具有低约束违反度且具有低目标函数值的个体存在于下一代群体中，因为非劣个体可能同时具有两个目标都是较小的情况，在一定程度上也保留了目标函数值小的个体. 这说明本文方法比文献[3]和文献[13]中的方法更具有优越性.

2) 当种群中可行解比例 $0<f_p<1$ 时，也就是当前种群中既有可行解又有不可行解. 在这个阶段，一般认为具有目标函数值小和低约束违反度的不可行解应该存活在下一代群体中，理由是在某些情况这些个体可能比对应的可行解更重要，特别是当可行域非常小或全局最优解位于可行域边界上[13]. 另一方面，由于约束优化进化算法的最终目标是找到可行最优解，所以我们不能完全抛弃可行解. 因此，在这个阶段，我们要倾向于保留一些重要的可行和不可行个体在种群中，维持群体中可行解与不可行解之间的平衡，避免算法陷入局部最优. 其中，好的不可行个体可以协助群体快速地靠近可行域，好的可行个体有利于顺利找到全局最优解，这也正体现了约束优化进化算法两个明确目标的思路.

Mezura-Montes 和 Coello[14]认为，确保约束违反度最小的不可行解以一定的概率进入下一代种群，以此来增加群体的多样性，同时还可以引导其他不可行个体快速地靠近可行域边界，实验结果表明这种机制对提高算法的性能是有效的. 为了维持下一代种群中可行个体与不可行个体合理的比例，在此阶段我们采用一种自适应个体选择策略：若当前种群中的可行解比例 $0<f_p\leqslant 0.7$ 时，则根据文献[1]中提出的个体选择法来选择，具体个体比较准则为：若被选择的两个个体均为可行解，则比较它们的目标函数值，目标函数值小的个体进入下一代种群，同时将该个体从群体中删除；否则，如果 $rand<p_f$，则比较两个个体的目标函数值，目标函数值小的个体进入下一代；如果 $rand\geqslant p_f$，则比较两个个体的约束违反程度，约束违反度低的个体进入下一代，将该个体从当前种群中删除. 其中，参数 p_f 表示在不可行域中仅使用目标函数比较个体的概率，且满足 $0\leqslant p_f\leqslant 1$，$rand$ 是(0,1)中均匀分布的随机数. 实验表明，$p_f=0.45$ 能够使算法保持较好的性能[1]. 重复上述过程直至达到下代群体的规模. 若当前种群中的可行解比例 $0.7<f_p<1$ 时，则与 $f_p=0$ 时的选择个体方法相同. 本文方法能维持群体中可行个体

和不可行个体一定的比例,同时增加约束违反度最小的不可行个体进入下一代种群的概率,增强了群体的多样性,有利于群体向最优解逼近.

3) 当种群中的可行解比例 $f_p=1$ 时,即种群中的个体全部都是可行解.此阶段的主要目标是使群体尽快地收敛到全局最优解.其中,目标函数值的大小反映了个体与全局最优解的距离.因此,我们根据个体目标函数值的大小来进行选择操作:首先将当前种群中所有个体按目标函数值升序排序,选择前 N(N 表示种群规模)个个体直接进入下一代群体.

1.5　算法步骤

设种群规模为 N,s 为个体变量的维数,t 为进化代数,并设置算法的其他一些参数.本文算法的具体步骤如下:

Step1. 产生初始种群.利用佳点集方法产生初始种群 $N_0=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$,$\boldsymbol{x}_i\in S$,$i=1,2,\cdots,N$.令 $t=0$.

Step2. 交叉操作.根据相似个体聚类的规则,将种群 N_t 分解成若干个子种群,然后以一定概率 p_c 从各聚类中选择个体,按照 1.2 节中的方法进行佳点集多父代交叉操作,生成新的种群 N_t^1.

Step3. 变异操作.对于每个聚类子种群,以一定概率 p_m 选择个体进行非均匀变异操作,产生新的种群 N_t^2.

Step4. 聚类局部搜索.对每一个聚类子种群,利用单形交叉算子进行局部搜索,产生新的种群 N_t^3.

Step5. 选择操作.从种群 N_t^1,N_t^2 和 N_t^3 中,根据 1.4 节中的个体优劣比较和选择准则,选出 N 个最好的个体进入到下一代种群 N_{t+1} 中.

Step6. 判断算法是否满足终止条件.如果满足,则输出最优解;否则,$t=t+1$,返回 Step2.

1.6　时间复杂度

根据上述对算法的描述可以方便地计算本文算法的时间复杂度,其主要取决于将种群聚类分成若干子种群和个体比较与选择准则.设算法的种群规模为 N,假设种群经过交叉、变异和局部搜索后产生的个体为 N',在最坏的情况下,算法的时间复杂度计算为:算法进行聚类时将种群分为若干个子种群的时间复杂度为 $O(N^2)$.经过交叉、变异和局部搜索后种群最多有($N+N'$)个个体,在第 1 种个体选择准则中,利用快速排序法辨识非劣个体需要 $O(2(N+N')\log(N+N'))$ 次比较,对非劣个体进行排序选择的时间复杂度为 $O(\bar{N}^2)$,其中 $\bar{N}$ 为群体中的非劣个体数.第 2 种个体选择准则的复杂度为 $O(2(N+N')\log(N+N'))+O(\bar{N}^2)$ 或 $O(N)$.第 3 种个体选择准则的复杂度为 $O((N+N')^2)$.其他操作的时间复杂度可忽略不计.因此,本文算法总的时间复杂度为 $O(N^2)$或 $O((N+N')^2)$.

2　数值实验

为了评价本文算法(记为 COHEA)的性能,我们选取文献[15]中的 15 个标准测试函数进行实验研究.对于每个测试函数,算法参数设置如下:种群规模 $N=100$,佳点集交叉概率 $p_c=0.2$,变异概率 $p_m=0.1$,$m=5$,b 的取值为 2～5,交叉概率 $p_1=0.7$,$\mu=10$,$\lambda=5$,扩张比 $\varepsilon=10$,算法的适应度函数值评价次数为 300 000 次.新算法 COHEA 在上述参数设置下均进行了 30 次独立的实验,记录其最好结果(best)、中间结果(median)、平均结果(mean)、最差结果(worst)和解的标准方差(std. dev).表 1 给出了新算法 COHEA 对 15 个测试函数的实验结果.

Table 1　The Experiment Results of the COHEA on 15 Benchmark Functions (30 times)

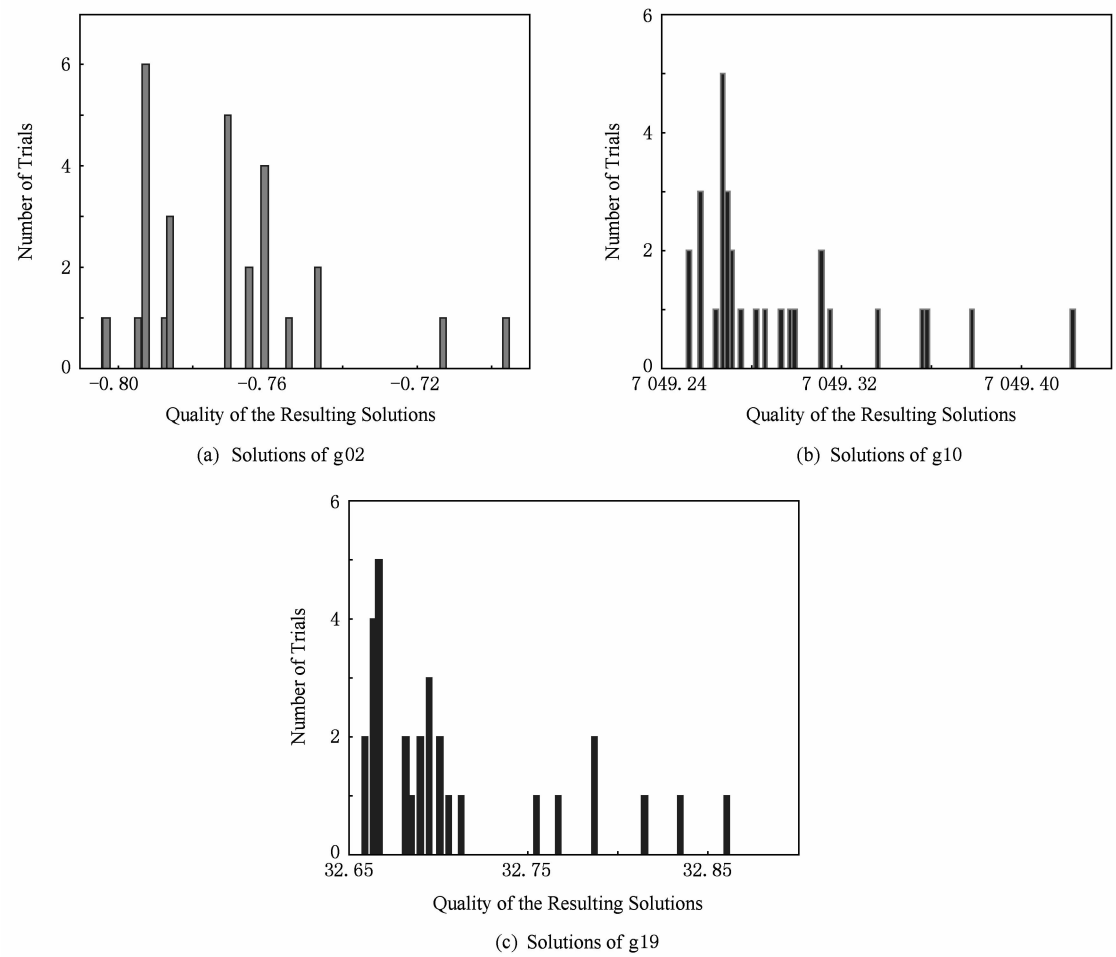
表 1　COHEA 算法对 15 个标准测试函数独立运行 30 次的实验结果

Function	Optimal	Best Result	Median Result	Mean Result	Worst Result	Std. Dev
g01	−15.000	−15.000 00	−15.000 00	−15.000 00	−15.000 00	0.000E+00
g02	−0.803 619	−0.803 523	−0.775 518	−0.772 942	−0.696 219	3.325E−02
g03	−1.000	−1.000 00	−1.000 00	−1.000 00	−1.000 00	6.286E−08
g06	−6 961.813 88	−6 961.813 88	−6 961.813 88	−6 961.813 88	−6 961.813 88	1.851E−12
g07	24.306 2	24.306 5	24.307 3	24.307 9	24.315 0	1.915E−03
g09	680.630 0	680.630 0	680.630 0	680.630 0	680.630 0	4.601E−10
g10	7 049.248	7 049.252	7 049.271	7 049.292	7 049.423	4.141E−02
g11	0.749 9	0.749 900	0.749 900	0.749 900	0.749 900	3.387E−08

Continue

Function	Optimal	Best Result	Median Result	Mean Result	Worst Result	Std. Dev
g12	−1.000 00	−1.000 000 0	−1.000 000 0	−1.000 000 0	−1.000 000 0	0.000E+00
g14	−47.763	−47.761	−47.761	−47.758	−47.710	1.083E−02
g15	961.715	961.715	961.715	961.715	961.715	4.625E−06
g16	−1.905 155	−1.905 155	−1.905 155	−1.905 155	−1.905 155	4.002E−13
g18	−0.866 025	−0.866 025	−0.866 025	−0.866 025	−0.866 025	3.488E−04
g19	32.656	32.659	32.695	32.708	32.861	2.141E−02
g24	−5.508 013	−5.508 013	−5.508 013	−5.508 013	−5.508 013	1.807E−15

从表 1 可知，除了测试函数 g02，g07，g10，g14 和 g19 以外，COHEA 对其他 10 个函数 30 次实验中一致地找到了全局最优解，对函数 g02，g07，g10，g14 和 g19 的实验结果也非常接近最优解，其中 g02，g10 和 g19 的实验结果分布图如图 3 所示. 较小的标准方差值也说明了 COHEA 具有较强的稳定性：



(a) Solutions of g02

(b) Solutions of g10

(c) Solutions of g19

Fig. 3 The number of trials versus the quality of the resulting solutions achieved for the test function g02, g10 and g19.

图 3 测试函数 g02，g10 和 g19 进行 30 次独立实验所得的结果分布

为了进一步比较算法的性能，我们将 COHEA 的运行结果与其他 6 种较好的约束优化进化算法进行了比较，它们分别是文献[1]中的基于随机排序的约束优化进化算法(记为 SR)，它是目前最为经典的约束优化进化算法之一；文献[7]中的自适应适应值法(记为 SAFF)；文献[9]中的多目标法(记为 CW)；文献[11]中的自适应均衡模型法(记为 ATM)；文献[14]中的简单多元进化策略(记为 SMES)和文献[16]中的加速自适应均衡模型(记为 AATM)，比较结果如表 2 所示：

Table 2　Comparison of 6 Algorithms on 9 Benchmark Functions

表 2　6 种算法对 9 个标准测试函数的实验结果比较

Function/Optimal	Status	Methods					
		SR[1]	SAFF[7]	CW[9]	ATM[11]	SMES[14]	COHEA
g01/−15.000	Best	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000 00
	Mean	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000 00
	Worst	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000 00
	St. Dev	0.0E+00	0	1.3E−14	1.6E−14	0	0.0E+00
g02/−0.803619	Best	−0.803 315	−0.802 97	−0.803 619	−0.803 388	−0.803 601	−0.803 523
	Mean	−0.781 975	−0.790 10	−0.803 220	−0.790 148	−0.785 288	−0.772 942
	Worst	−0.726 288	−0.760 43	−0.792 608	−0.756 986	−0.751 322	−0.696 219
	St. Dev	2.0E−02	1.2E−02	2.0E−03	1.3E−02	1.7E−02	3.325E−02
g03/−1.000	Best	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000 00
	Mean	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000 00
	Worst	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000 00
	St. Dev	1.9E−04	7.5E−05	1.9E−16	1.9E−04	2.1E−04	6.286E−08
g06/−6961.81388	Best	−6 961.814	−6 961.800	−6 961.814	−6 961.814	−6 961.814	−6 961.813 88
	Mean	−6 875.940	−6 961.800	−6 961.814	−6 961.814	−6 961.284	−6 961.813 88
	Worst	−6 350.262	−6 961.800	−6 961.814	−6 961.814	−6 952.482	−6 961.813 88
	St. Dev	1.6E+02	0	1.8E−12	4.6E−12	1.9E+00	1.851E−12
g07/24.306 2	Best	24.307	24.48	24.306	24.306	24.372	24.3065
	Mean	24.374	26.58	24.306	24.316	24.475	24.3079
	Worst	24.642	28.40	24.306	24.359	24.843	24.3150
	St. Dev	6.6E−02	1.1E+00	5.7E−12	1.1E−02	1.3E−01	1.915E−03
g09/680.6300	Best	680.630	680.64	680.630	680.630	680.632	680.6300
	Mean	680.656	680.72	680.630	680.639	680.643	680.6300
	Worst	680.763	680.87	680.630	680.673	680.719	680.6300
	St. Dev	3.4E−02	5.9E−02	4.7E−13	1.0E−02	1.6E−02	4.601E−10
g10/7049.248	Best	7 054.316	7 061.34	7 049.248	7 052.253	7 051.903	7 049.252
	Mean	7 559.192	7 627.89	7 049.248	7 250.437	7 253.047	7 049.271
	Worst	8 835.655	8 288.79	7 049.248	7 560.224	7 638.366	7 049.423
	St. Dev	5.3E+02	3.7E+02	4.0E−09	1.2E+02	1.4E+02	4.141E−02
g11/−0.7499	Best	0.750	0.750	0.750	0.75	0.75	0.749 900
	Mean	0.750	0.750	0.750	0.75	0.75	0.749 900
	Worst	0.750	0.750	0.750	0.75	0.75	0.749 900
	St. Dev	8.0E−05	0	0.0E+00	3.4E−04	1.5E−04	3.387E−08
g12/−1.000	Best	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000 00
	Mean	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000 00
	Worst	−1.000	−1.000	−1.000	−0.994	−1.000	−1.000 00
	St. Dev	0.0E+00	0	0.0E+00	1.0E−03	0	0.000E+00

从表 2 可以看出,与 SR 算法、SAFF 算法和 SMES 算法相比,COHEA 在函数 g02,g06,g07,g09,g10 和 g11 上的结果明显优于 3 种算法,而在函数 g01,g03 和 g12 上则取得了相似的结果. 这说明算法 COHEA 比 SR,SAFF,SMES 算法具有更好的处理约束优化问题的能力. CW 算法是当前已知最具竞争力的求解约束优化问题的进化算法之一,COHEA 算法除了函数 g02,g07 和 g10 的结果稍逊外,在其他 6 个函数上取得了与 CW 算法相似的结果. 与 ATM 算法相比,在函数 g02,g09 上略优于 ATM 算法,而在其他 7 个函数上取得了相似的结果.

对于函数 g14，g15，g16，g18，g19 和 g24，由于是最近才提出来的标准测试函数，所以文献中出现的求解结果还不多，本文将 COHEA 与文献[16]中的 AATM 算法进行了比较，比较结果如表 3 所示. 从表 3 中的结果可知，除了函数 g14 和 g19 较优外，COHEA 在其他 4 个函数上取得了与 AATM 相似的结果.

另外，就计算代价而言，对于同一个测试函数，本文算法 COHEA 的适应值函数评价次数为 300 000 次，而 SAFF 算法则为 1 400 000 次，SR 算法和 CW 算法则为 350 000 次，SMES 算法和 ATM 算法则为 240 000 次，AATM 算法为 120 000 次. 基于以上比较可知，COHEA 优于或相似于其他一些尖端算法.

Table 3 Comparing COHEA with AATM on 6 Benchmark Test Functions

表 3 COHEA 与 AATM 算法对 6 个测试函数的实验结果比较

Function	Optimal	Methods	Best	Median	Mean	Worst	St. Dev
g14	−47.763	COHEA	−47.761	−47.761	−47.758	−47.710	1.083E−02
		AATM	−47.762	−47.753	−47.750	−47.712	1.0E−02
g15	961.715	COHEA	961.715	961.715	961.715	961.715	4.625E−06
		AATM	961.715	961.715	961.715	961.716	3.0E−04
g16	−1.905 155	COHEA	−1.905 155	−1.905 155	−1.905 155	−1.905 155	4.002E−13
		AATM	−1.905 155	−1.905 155	−1.905 155	−1.905 155	2.4E−14
g18	−0.866 025	COHEA	−0.866 025	−0.866 025	−0.866 025	−0.866 025	3.488E−04
		AATM	−0.866 025	−0.866 013	−0.865 952	−0.864 863	2.1E−04
g19	32.656	COHEA	32.659	32.695	32.708	32.861	2.141E−02
		AATM	32.725	32.934	32.952	33.243	1.4E−01
g24	−5.508 013	COHEA	−5.508 013	−5.508 013	−5.508 013	−5.508 013	1.807E−15
		AATM	−5.508 013	−5.508 013	−5.508 013	−5.508 013	1.8E−15

3 结 论

算法本身的搜索能力和约束处理技术是利用进化算法求解约束优化问题时所要面临的两个关键问题. 针对这两个问题，本文提出一种新的求解约束优化问题的混合进化算法，其特点为：

1）利用佳点集理论产生初始种群，以保证初始种群个体能均匀地分布在整个搜索空间内；

2）根据相似个体聚类规则，将种群分为若干个子种群，然后针对每个子种群，利用佳点集理论设计交叉算子，使其产生具有代表性的个体，维持种群的多样性，以便更好地搜索整个解空间，另外，引入局部搜索策略以加快算法的收敛速度；

3）在进化过程中，根据群体处在不同的阶段，采用不同的个体比较和选择准则，选择优秀的个体进入下一代种群；

4）通过 15 个标准测试函数的实验结果表明了新算法的有效性. 由于本文算法的参数设置是固定的，因此，下一步研究工作就是探讨参数对算法性能的影响和将算法应用到实际中.

参 考 文 献

[1] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization [J]. IEEE Trans on Evolutionary Computation, 2000, 4(3): 284-294

[2] Deb K. An efficient constraint handling method for genetic algorithm [J]. Computation Methods in Applied Mechanics and Engineering, 2000, 186(2/3/4): 311-338

[3] Cai Zixing, Jiang Zhongyang, Wang Yong, et al. A novel constrained optimization evolutionary algorithm based on orthogonal experimental design [J]. Chinese Journal of Computers, 2010, 33(5): 855-864 (in Chinese)
(蔡自兴，江中央，王勇，等. 一种新的基于正交实验设计的约束优化进化算法[J]. 计算机学报，2010，33(5)：855-864)

[4] Zhang Ling, Zhang Bo. Good point set based genetic algorithm [J]. Chinese Journal of Computers, 2001, 24(9): 917-922 (in Chinese)
(张铃，张钹. 佳点集遗传算法[J]. 计算机学报，2001，24(9)：917-922)

[5] Wang Y P, Dang C Y. An evolutionary algorithm for global optimization based on level-set evolution and Latin squares [J]. IEEE Trans on Evolutionary Computation, 2007, 11(5): 579-595

[6] Jiang Zhongyang, Cai Zixing, Wang Yong. Hybrid self-adaptive orthogonal genetic algorithm for solving global optimization problems [J]. Journal of Software, 2010, 21(6): 1296-1307 (in Chinese)
(江中央,蔡自兴,王勇. 求解全局优化问题的混合自适应正交遗传算法 [J]. 软件学报, 2010, 21(6): 1296-1307)

[7] Farmani R, Wright J A. Self-adaptive fitness formulation for constrained optimization [J]. IEEE Trans on Evolutionary Computation, 2003, 7(5): 445-455

[8] Liang Ximing, Long Wen, Qin Haoyu, et al. Constrained optimization evolutionary algorithm based on individual feasibility of population [J]. Control and Decision, 2010, 25(8): 1129-1132 (in Chinese)
(梁昔明,龙文,秦浩宇,等. 基于种群个体可行性的约束优化进化算法[J]. 控制与决策, 2010, 25(8): 1129-1132)

[9] Cai Z X, Wang Y. A multiobjective optimization-based evolutionary algorithm for constrained optimization [J]. IEEE Trans on Evolu-tionary Computation, 2006, 10(6): 658-675

[10] Dong Hongbin, Huang Houkuan, He Jun, et al. An evolutionary program-ming to solve constrained optimization problems [J]. Journal of Computer Research and Development, 2006, 43(5): 841-850 (in Chinese)
(董红斌,黄厚宽,何军,等. 一种求解约束优化问题的演化规划算法[J]. 计算机研究与发展, 2006, 43(5): 841-850)

[11] Wang Y, Cai Z X, Zhou Y R, et al. An adaptive trade-off model for constrained evolutionary optimization [J]. IEEE Trans on Evolutionary Computation, 2008, 12(1): 80-92

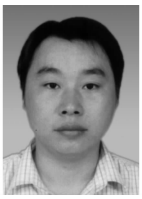
[12] Liu Hui, Cai Zixing, Wang Yong. Novel constrained optimization evolutionary algorithm based on good point set [J]. Journal of System Simulation, 2009, 21(6): 1620-1623 (in Chinese)
(刘慧,蔡自兴,王勇. 基于佳点集的约束优化进化算法[J]. 系统仿真学报, 2009, 21(6): 1620-1623)

[13] Venkatraman S, Yen G G. A generic framework for constrained optimization using genetic algorithms [J]. IEEE Trans on Evolutionary Computation, 2005, 9(4): 424-435

[14] Mezura-Montes E, Coello C C A. A simple multimembered evolution strategy to solve constrained optimization problems [J]. IEEE Trans on Evolutionary Computation, 2005, 9(1): 1-17

[15] Liang J J, Runarrson T P, Mezura-Montes E, et al. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization [OL]. [2010-10-21]. http://www3.ntu.edu.sg/home/EPNSugan/

[16] Wang Y, Cai Z X, Zhou Y R. Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization [J]. International Journal for Numerical Method in Engineering, 2009, 77(11): 1501-1534



Long Wen, born in 1977. PhD. His main research interests include intelligence constrained optimization and optimization algorithm.

Liang Ximing, born in 1967. PhD, professor and PhD supervisor of Central South University. His main interests include evolution computation and optimization method

Xu Songjin, born in 1972. PhD candidate of Central South University. His main interests include optimization method and fuzzy control.

Chen Fu, born in 1987. Master candidate of Central South University. His main interests include genetic algorithm and optimization method.