文章编号:1671 - 6833(2018)06 - 0023 - 07

两阶段三存档集约束优化算法(TSDA)

李二超,李 进

(兰州理工大学 电气工程与信息工程学院,甘肃 兰州 730000)

摘 要:针对约束优化算法采用相同的进化策略处理位于 Pareto 边缘的解与函数值较差的解,使得寻优结果较差,提出一种两阶段三存档集约束优化算法.该算法分为两个阶段.第一阶段:根据$\varepsilon(t)$值将种群分为 3 个存档集,即非支配解存档集、支配解存档集以及非支配可行解存档集.非支配解存档集进行混合策略的双重寻优,既避免了算法陷入局部最优,又使得靠近前沿的解加速收敛;支配解存档集则注重于全局搜索,从而有利于算法搜索到更优可行解.非支配解存档集和支配解存档集使用不同的优化策略进行进化,提高了算法的寻优能力.第二阶段:在第一阶段达到设定的代数时,将各代保留到非支配可行解存档集中的个体进行快速非支配排序,选出的 N 个优秀个体则为最优解.最后,将提出的算法与其他约束多目标进化算法在 3 种经典约束测试函数上进行对比,仿真结果表明,所提出算法在不同类约束条件下的寻优能力均具有优势.

关键词:约束优化;三存档集;混合策略;两阶段;寻优能力

中图分类号:TP273 **文献标志码**:A **doi**:10.13705/j.issn.1671 - 6833.2018.06.002

0 引言

近些年,用多目标优化技术来解决约束问题是个热点[1-2].首先要找到全局内的可行解集,其次才能考虑解集的分布性和收敛性,其中设计有效的约束处理技术是关键所在.因此,将进化算法引入到约束多目标优化问题中是必须的.除了要解决多目标优化过程中面临的提高进化算法搜索能力、避免陷入局部最优、合理设置参数等多个问题外,还必须保证算法获得可靠的求解性能和全局的优化效果.因此,约束多目标优化算法的研究内容较多,不仅需要重点研究约束处理技术,还必须研究与之相适应的进化策略、多样性维护策略以及精英选择策略等多项关键技术.

孟红云等[3]提出一种用于约束多目标优化问题的双群体差分进化算法.该算法同时使用两个群体,一个群体用于保存搜索过程中找到的可行解;另一个用于记录在搜索过程中得到的部分具有某些优良特性的不可行解,避免了构造罚函数和直接删除不可行解.但是,该方法在更新可行解集时会存在丢失边界解和不利于保持种群的整体多样性的缺陷;而在更新不可行解集时会出现目标函数较差的个体,从而影响种群的收敛.Deb 提出了一种可行性法则[4]来比较个体,但 Deb 准则认为可行解优于不可行解,没有利用较优不可行解所携带的信息,不利于保持种群的多样性.最近,一些学者基于精英保留机制提出了双档案集方法,将进化过程产生的可行解和不可行解个体分别存入可行解档案和不可行档案.例如,文献[5]提出了基于遗传算法的双档案集机制;文献[6-9]提出了基于差分算法的双档案方法.然而这些方法仅粗略地以约束违反度值作为依据,优先选择靠近可行边界的不可行解,没有区分出最优不可行解,将直接影响算法的寻优能力.

基于此,笔者提出一种两阶段三存档集约束优化算法,主要创新点包括:①提出新的三存档集处理约束条件,避免了处于 Pareto 边缘的解与函数值较差的解采用同样的进化策略;②对非支配解存档集和支配解存档集采用侧重点不同的搜索方式.非支配解存档集采用混合策略进化,既进行局部搜索又进行全局搜索,保证在 Pareto 前沿附近的解快速收敛,又不会陷入局部最优;支配解存档集采用自适应变异,将变异率和个体信息联系起来,引导种群进化.

收稿日期:2018 - 03 - 01;**修订日期**:2018 - 07 - 19

基金项目:国家自然科学基金资助项目(61763026,61403175)

作者简介:李二超(1980—),男,河北定州人,兰州理工大学副教授,博士,博士生导师,主要从事多目标优化技术研究,E-mail:lecstarr@163.com.

1 约束优化问题及相关定义

不失一般性,一个约束的多目标优化问题可定义如下[1]:

$$\begin{cases}\min\ F(x) = (f_1(x), f_2(x), \cdots, f_m(x));\\ \text{s.t. } g_j(x) \leqslant 0,\ j = 1,2,\cdots,q;\\ \quad h_j(x) = 0,\ j = q+1,\cdots,m;\\ \quad l_i \leqslant x_i \leqslant u_i,\ i = 1,2,\cdots,n.\end{cases} \tag{1}$$

式中:$x = [x_1, x_2, \cdots, x_n] \in S$ 是决策变量,S 表示 n 维搜索空间,它满足如下边界约束条件:$l_i \leqslant x_i \leqslant u_i$,$1 \leqslant i \leqslant n$;$F(x)$是目标函数;$q$ 表示不等式约束条件的个数;m 表示总约束条件的个数;$g_j(x)$为第 j 个不等式约束条件;$h_j(x)$为第 $j-q$ 个等式约束条件;u_i 和 l_i 分别为分量 x_i 的上界和下界.

满足所有约束条件的解是可行解,而所有的可行解组成可行域 $\Omega \in S$;不满足任意一个约束条件的解被称为不可行解,所有的不可行解组成不可行域. 如果 $g_j(x) = 0 (j = 1,2,\cdots,q)$在一个可行解处成立,则称该不等式约束条件在该可行解处是活跃的.

个体 x 在第 j 个约束条件上的约束违反度表示为:

$$G_j(x) = \begin{cases}\max\{g_j(x), 0\}, & 1 \leqslant j \leqslant q;\\ \max\{|\ h_j(x)\ | - \delta, 0\}, & q+1 \leqslant j \leqslant m,\end{cases} \tag{2}$$

式中:δ 为等式约束的容忍参数,该参数可以定义为一个特定的值(一般设为 0.000 1)或采用只适应变化的方式定义. 所以个体 x 总的约束违反度表示为:$v(x) = \sum_{j=1}^{m} G_j(x)$.

2 两阶段三存档集约束优化算法(TSDA)

通过对三存档集进行搜索机制不同的进化方式,来引导种群向真实 Pareto 前沿进行进化. 利用非支配解存档集双重寻优,一方面使得靠近Pareto前沿的解加速向前沿靠近;另一方面使存档集避免陷入局部收敛. 支配解存档集利用自适应的变异概率,将目标函数值与种群个体的信息联系起来,提高了算法的搜索效率. 最后通过非支配可行解存档集保留种群当中的精英个体,提高了算法约束优化性能.

2.1 最优不可行解

在非支配解存档集中让最优不可行解进入存档集参与进化,不仅能加快算法的收敛速度,同时也增强了可行域边界附近的寻优能力. 判定一个不可行解是否位于可行域附近,具体如下:若一个不可行解的约束违反度小于等于当代的 $\varepsilon(t)$值(可行阈值),则其为靠近可行域边界的不可行解.

定义 1 ε 值(可行阈值):

$$\varepsilon(t) = \begin{cases}\varepsilon(0) \cdot \left(1 - \dfrac{5}{3}\dfrac{t}{T}\right), & t \leqslant 0.6T;\\ 0, & t > 0.6T,\end{cases} \tag{3}$$

式中:t 是当前代数;T 为进化总代数;$\varepsilon(0)$是种群中个体根据约束违反度升序排序后第 $0.05 \cdot N$ 个个体,N 为种群的规模.

定义 2 最优不可行解:靠近可行域边界,且在可行域中找不到能支配该不可行解的可行解,则称该不可行解为最优不可行解,如图 1 所示.

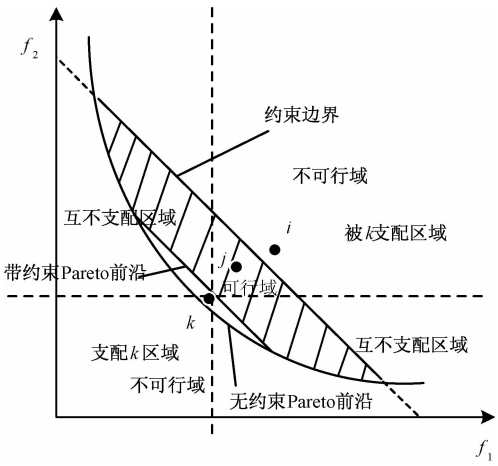

图 1 最优不可行解

Fig. 1 The optimal infeasible solution

图 1 中点 i 及 k 皆为靠近可行域边界(图中的阴影部分为可行域)的不可行解,不可行解 i 虽然靠近可行域边界,但其远离 Pareto 前沿,对种群进化没有引导作用,因此,在选择最优不可行解时,需要跳过这种不可行解. 而不可行解 k 既满足靠近可行域边界又满足在可行域中找不到能支配 k 的可行解,则 k 为最优不可行解,让其参与进化.

2.2 三存档集模型

笔者基于 NSGA-Ⅱ的基础上加入了两阶段三存档模型,该算法将进化过程划分为两个阶段. 首先,将种群利用快速非支配排序法进行分层. 根据 Pareto 等级、约束违反度及 $\varepsilon(t)$值划分为非支配解(包括可行解及不可行解)、支配解及非支配可行解,分别存入 3 个存档集:非支配解存档集、支配解存档集以及非支配可行解存档集. 然后,种群进行两阶段进化操作. 第一阶段:非支配解存档集及支配解存档集采取不同的策略分别进行进化;第二阶段:判断非支配可行解存档集中解的个数

是否超过种群数量,如果超过则将其进行快速非支配排序,计算出非支配可行解存档集中个体的 rank 值及拥挤度距离值,选出种群规模大小的优秀个体.

2.3　非支配解存档集进化

定义 3　非支配解存档集(NDA):根据 2.1 节所计算出当代的 $\varepsilon(t)$ 值区分出种群中约束违反度小于等于 $\varepsilon(t)$ 值的解,其中既包括可行解,也包括不可行解.若该解 rank 值等级为 1,则为非支配解,非支配解的集合称为非支配解存档集.

非支配解存档集(NDA)既进行局部搜索又进行全局搜索. NDA 存档集中的解虽然皆为 Pareto等级为 1 的解,但这些解有靠近真实 Pareto 前沿的、也有远离 Pareto 前沿的.若只进行局部搜索,很可能陷入局部最优当中,因此,笔者采用非支配解存档集局部搜索和全局搜索同时进行.因为非支配解存档集中的不可行解为靠近可行域且在可行域当中没有能支配它的可行解,所以非支配不可行解只进行局部搜索可以更好地引导种群向 Pareto 前沿进化.非支配解存档集当中的可行解分为两种情况:一种情况为该可行解位于 Pareto 前沿附近,采用局部搜索策略可以使其尽快收敛;另外一种情况为该可行解远离 Pareto 前沿,采用全局搜索策略可以扩大搜索的空间,避免陷入局部最优.鉴于以上两种情况,非支配解存档集当中的可行解为不确定因素,故笔者采用两种不同交叉变异概率对其进行进化.首先将可行解复制一份放入集合 CNDFA 中,然后非支配解存档集整体进行局部搜索,而复制的可行解集 CNDFA 进行全局搜索.非支配解存档集进化过程中,交叉算子取 0.5,变异算子取 0.05,在进化过程中生成一个随机数,若随机数 $u \leqslant 0.5$,u 为[0,1]上均匀分布的随机数,则进行 SBX 交叉操作;若随机数 $u \geqslant 0.95$,则进行多项式变异操作;$0.5 <$ 若随机数 $u < 0.95$,则按照约束违反度值从小到大,从非支配解存档集合中依次选择两个个体进入下一代.CNDFA 集合仍采用原 NSGA-Ⅱ的 SBX 交叉和多项式变异操作.

2.4　支配解存档集进化

定义 4　支配解存档集(DA):若该解不为非支配解,则其为支配解,支配解的集合称为支配解存档集.

支配解存档集进行全局搜索,具体操作如下.

(1)交叉操作:为了跳出局部最优以及加强空间搜索能力,笔者将正态分布引入到 SBX 算子中.由于正态分布的引入,算法可搜索到的空间更为广阔,因此更容易跳出局部最优,从而使 Pareto 前沿更具延展性,均匀地分布在 Pareto 域上,保证种群的多样性. NDX 算子的产生方式如下:

如果 $u \leqslant 0.5$,则

$$\begin{cases} c_{1,k} = 0.5[(p_{1,k} + p_{2,k}) + \\ \qquad 1.481 \mid N(0,1) \mid (p_{1,k} - p_{2,k})], \\ c_{2,k} = 0.5[(p_{1,k} + p_{2,k}) - \\ \qquad 1.481 \mid N(0,1) \mid (p_{1,k} - p_{2,k})]; \end{cases} \tag{4}$$

如果 $u > 0.5$,则

$$\begin{cases} c_{1,k} = 0.5[(p_{1,k} + p_{2,k}) - \\ \qquad 1.481 \mid N(0,1) \mid (p_{1,k} - p_{2,k})], \\ c_{2,k} = 0.5[(p_{1,k} + p_{2,k}) + \\ \qquad 1.481 \mid N(0,1) \mid (p_{1,k} - p_{2,k})]. \end{cases} \tag{5}$$

式中:$c_{1/2,k}$ 为子代染色体上对应的第 k 个变量;$p_{1,k}$、$p_{2,k}$ 分别为父代两个染色体上对应的第 k 个变量;$\mid N(0,1) \mid$ 为正态分布随机变量.

(2)变异操作:经典 NSGA-Ⅱ算法中多项式变异算子与种群个体的信息并没有联系,无论种群进化到何种程度,变异概率都为固定值,种群没有特定的方向进行进化.因此,将种群个体当前信息与变异算子结合起来时,才能使种群向着真实的 Pareto 前沿进化.自适应变异算子[10]产生方式如下:

$$p_{\mathrm{m}} = \begin{cases} p_{\mathrm{mmax}} - \dfrac{(p_{\mathrm{mmax}} - p_{\mathrm{mmin}}) \cdot (f_{\max} - f_{\mathrm{avg}})}{f_{\max} - f_{\min}}, & f \geqslant f_{\mathrm{avg}}; \\ p_{\mathrm{mmin}}, & f < f_{\mathrm{avg}}, \end{cases} \tag{6}$$

式中:$f_{\max}$、$f_{\min}$ 分别为当前种群中目标函数的最大值、最小值;f_{avg} 为当前种群所有个体平均目标函数值.

在进化的初期,个体的最大目标函数值 $f_{\max}$ 与最小目标函数值 $f_{\min}$ 差异较大,目标函数值较大和较小的个体数目大致相同,此时 f_{avg} 近似等于 $f_{\max}$ 与 $f_{\min}$ 的平均值.由式(6)知,$(f_{\max} - f_{\mathrm{avg}})/(f_{\max} - f_{\min})$ 的值大致为 0.5,此时求得的变异概率较大,有助于算法初期进行全局搜索,寻找最优解集.算法进化到后期,大部分种群个体求得的目标函数值大致相同,此时 f_{avg} 是一个略大于 $f_{\min}$ 的值.当所有个体都进化到最优解时,达到一种极限条件,即 $f_{\min} = f_{\mathrm{avg}}$,这时公式中 $(f_{\max} - f_{\mathrm{avg}})/(f_{\max} - f_{\min})$ 的值接近 1,交叉变异概率相对调整为较小的数,增进局部寻优的能力,与种群个体寻优的方向相一致,有利于最优解集的搜索.

2.5 非支配可行解存档集

笔者使用非支配可行解存档集来保存进化过程中产生的所有可行非劣解. 首先,在种群每次迭代过程中,将非支配存档集中约束违反度为 0 的解提取出来,加入到非支配可行解集当中;然后,通过快速非支配排序,选出的 N 个优秀个体进行输出,此时输出的解全部为最优的可行解.

2.6 算法流程

两阶段三存档集约束优化算法流程:

Step 1 开始 $gen = 1$.

Step 2 初始化种群. 从决策空间 S 随机产生 N 个个体的初代种群 $P(t) = \{x_{1,t}, \cdots, x_{N,t}\}$,并根据目标函数计算每个个体的目标函数值.

Step 3 非支配排序. 对 Step 2 产生的种群进行非支配排序,按照非支配等级由小到大排序,并计算每个个体的约束违反度.

Step 4 存档集划分. 将种群中每个个体的约束违反度值与可行阈值 $\varepsilon(t)$ 值相比,再根据 rank 值大小,分类到非支配解存档集(NDA)、支配解存档集(DA)及非支配可行解存档集(NDFA)当中.

Step 5 基因操作. 对非支配解存档集和支配解存档集采取不同的进化策略.

Step 5.1 非支配解存档集采用混合搜索策略. 根据 2.3 节中的内容分别对 NDA 和 CNDFA 两个集合进行局部搜索和全局搜索,合并进化后的两个集合,记作 $U(t)$.

Step 5.2 支配解存档集采用全局搜索. 利用正态分布算子和自适应变异算子对 DA 进行交叉变异操作,从而生成子种群 $Q(t)$.

Step 6 合并子父代种群. 将 $U(t)$,$Q(t)$及父代种群 $P(t)$进行合并,记作 $Y(t)$.

Step 7 选择操作. 将 $Y(t)$进行快速非支配排序,根据 rank 值及拥挤度距离值大小从 $Y(t)$中选择出 N 个个体作为下一代种群 $P(t+1)$.

Step 8 判断当前代数是否大于设定代数,如满足则跳转至 Step 9,进行第二阶段的操作;否则,$gen = gen + 1$,并跳转至 Step 3.

Step 9 对非支配可行解存档集(NDFA)进行快速非支配排序,根据 rank 值及拥挤度距离选择出 N 个个体,输出最终的 Pareto 最优解,算法终止.

2.7 计算复杂度分析

假设非支配解存档集规模为 N_1,支配解存档集规模为 N_2,非支配可行解存档集规模为 N_3. 则计算非支配解存档集、支配解存档集和非支配可行解存档集目标函数和约束违反度的时间复杂度为 $O(m(N_1 + N_2 + N_3))$. 其中,m 为目标函数个数;N 为种群规模;非支配解存档集变异和交叉操作的时间复杂度为 $O(mN_1)$;支配解存档集变异和交叉操作的时间复杂度为 $O(mN_2)$;选择操作的时间复杂度为 $O(m(N_1 + N_2 + N_3)^2)$. 综上,TSDA 算法迭代一次的最坏时间复杂度为 $O(m(N_1 + N_2 + N_3)) + O(mN_1) + O(mN_2) + O(m(N_1 + N_2 + N_3)^2)$,所以本文算法时间复杂度为 $O(mN^2)$.

3 TSDA 算法实验结果与分析

3.1 测试函数与参数设置

所有实验在硬件配置为 Intel Pentium、G2030 CPU、4G 内存、主频 3.0 GHz、Win10 系统的计算机上进行,程序采用 Matlab R 2010 编写.

为了验证 TSDA 算法的性能,选择了 SRN、TNK、OSY 作为测试函数,其函数表达式见表 1. 并与运用可行性规则的 NSGA-Ⅱ算法(FRNSGA-Ⅱ)进行比较.

TSDA 算法参数设置:随机生成的种群规模为 $N = 200$,进化代数为 2 000 代,交叉算子为 NDX(正态分布)交叉算子,采用自适应变异算子,交叉变异概率详见 2.4 节.

FRNSGA-Ⅱ算法参数设置:随机生成的种群规模为 $N = 200$,进化代数为 2 000 代,交叉算子为 SBX 算子,交叉概率为 0.95,变异算子为多项式变异,变异概率为 0.05.

测试函数特点:SRN 函数的 Pareto 前沿连续;TNK 函数为非线性约束函数,且其 Pareto 前沿不连续;OSY 函数共有 6 个不等式约束条件,既有线性约束又有非线性约束;Pareto 前沿由三段折线组成. 约束测试函数的真实 Pareto 前沿源自参考文献[11].

3.2 评价指标

3.2.1 GD 指标

世代距离(generational distance,GD)[12],用于评价所求得的近似 Pareto 前沿 P_{know} 相对于真实 Pareto 前沿 P_{true} 的逼近程度,该指标的定义如下:

$$GD = \frac{1}{N_{PF}} \sqrt{\sum_{i=1}^{n} d_i^2}, \tag{7}$$

式中:N_{PF} 为 P_{know} 中个体的数量;d_i 为 P_{know} 中第 i 个个体的目标向量到 P_{true} 中最近个体的欧式距离;GD 越小,表明 P_{know} 越逼近 P_{true},收敛性越好.

表 1　测试函数表达式

Tab. 1　Description of test functions

测试函数	目标函数 $\min F(x) = (f_1(x), f_2(x))$	变量约束	决策变量范围
SRN	$\begin{cases}\min f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2 \\ \min f_2(x) = 9x_1 - (x_2 - 1)^2\end{cases}$	$\begin{cases}x_1^2 + x_2^2 \leqslant 225 \\ x_1 - 3x_2 + 10 \leqslant 0\end{cases}$	$-20 \leqslant x_i \leqslant 20$ $i = 1,2$
TNK	$\begin{cases}\min f_1(x) = x_2 \\ \min f_2(x) = x_1\end{cases}$	$\begin{cases}-x_1^2 - x_2^2 + 1 + \\ 0.1\cos(16\arctan(x_1/x_2)) \leqslant 0 \\ (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leqslant 0.5\end{cases}$	$0 \leqslant x_i \leqslant \pi$ $i = 1,2$
OSY	$\begin{cases}\min f_1(x) = 4x_1^2 + 4x_2^2 \\ \min f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2\end{cases}$	$\begin{cases}x_1 + x_2 - 2 \geqslant 0 \\ 6 - x_1 - x_2 \geqslant 0 \\ 2 + x_1 - x_2 \geqslant 0 \\ 2 - x_1 + 3x_2 \geqslant 0 \\ 4 - (x_3 - 3)^2 - x_4 \geqslant 0 \\ (x_5 - 3)^2 + x_6 - 4 \geqslant 0\end{cases}$	$0 \leqslant x_i \leqslant 10$ $i = 1,2,6$ $1 \leqslant x_3, x_5 \leqslant 5$ $0 \leqslant x_4 \leqslant 6$

3.2.2　SP 指标

Schott 在 1995 年提出了 Spacing 指标[13],用来计算所求得的解集在目标空间上的分布均匀性,其计算公式如下所示:

$$S = \sqrt{\frac{1}{N_{PF} - 1}\sum_{i=1}^{N_{PF}} (d' - d_i)^2}. \tag{8}$$

式中:$d_i = \min_{j \in 1,\cdots,n \wedge j \neq i} \sum_{k=1}^{M} | f_k^i(x) - f_k^j(x) |$ 表示离第 i 个个体最小的目标距离和;d' 为所有 d_i 的均值;N_{PF} 为所求得解集的个体数量;S 越小表示所求得的解集分布的越均匀,当 S 为 0 时,表示所求解集中的所有解都是等距离分布的.

表 2 为 2 种算法对上述 3 种测试函数的 SP、GD 的统计结果. 可以看出,本文算法 TSDA 在测试函数 TNK、OSY 上的 GD 值明显小于对比算法——FRNSGA-Ⅱ算法,说明笔者所提算法采用的策略避免了种群陷入局部最优,对种群收敛到 Pareto 前沿起着重要的作用. 在 SP 值方面,算法也有着明显的优势,这说明所提算法的分布性更好,保证了种群的多样性.

3.3　仿真结果

图 2、图 4、图 6 为本文算法 TSDA 所求得的 Pareto 前沿,图 3、图 5、图 7 为基于可行性规则的 NSGA-Ⅱ算法所求得的 Pareto 前沿. 通过以上的对比可以看出:对于 SRN 函数,两种算法虽然都找到了真实的 Pareto 前沿,但 FRNSGA-Ⅱ算法的分布性和收敛性较差;对于 TNK 测试函数,本文算法 TSDA 有更好的收敛性,并保持良好的分布性;对于 OSY 测试函数,FRNSGA-Ⅱ算法所得到的解集明显偏离真实前沿. 由此可以得出,笔者提出的 TSDA 算法在 3 个约束测试问题上,其最优 Pareto 前沿在逼近性和分布性上都明显优于 FRNSGA-Ⅱ.

4　结论

针对约束优化算法处理位于 Pareto 边缘的解与函数值较差的解采用相同的进化策略使得寻优结果较差的问题,提出两阶段三存档集约束优化算法. 利用两个存档集不同的寻优方式,很好地兼顾了探索开发和收敛的平衡. 并且通过与其他算法在 3 种测试函数上的对比表明,本文算法在处理约束多目标优化问题时,分布性及收敛性均

表 2　2 种进化算法求解标准测试函数所得的 GD、SP 值

Tab. 2　GD and SP values obtained by two evolutionary algorithms for solving standard test functions

测试函数	GD				SP			
	TSDA		FRNSGA-Ⅱ		TSDA		FRNSGA-Ⅱ	
	平均值	方差	平均值	方差	平均值	方差	平均值	方差
SRN	2.8346E-04	2.5521E-03	0.022 3	0.528 6	0.093 2	0.001 4	1.176 2	0.972 3
TNK	1.137E-04	8.1149E-03	7.499 5	7.042 7	0.134 8	7.865E-04	2.435 7	3.071 5
OSY	0.612 5	0.304 0	2.178 5	0.718 6	3.246 8	1.559 7	8.523 6	2.461 8

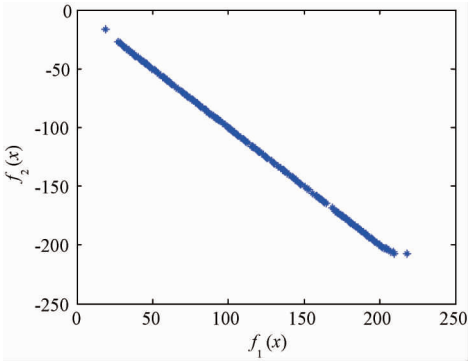

图2 TSDA算法SRN函数仿真结果

Fig. 2 Simulation result of TSDA algorithm on SRN function

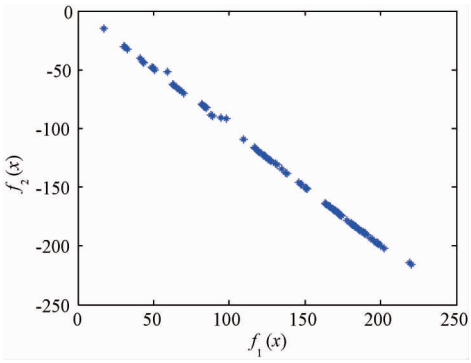

图3 FRNSGA-Ⅱ算法SRN函数仿真结果

Fig. 3 Simulation result of FRNSGA-Ⅱ on SRN function

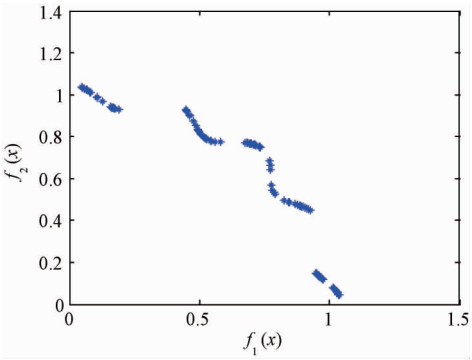

图4 TSDA算法TNK函数仿真结果

Fig. 4 Simulation result of TSDA algorithm on TNK function

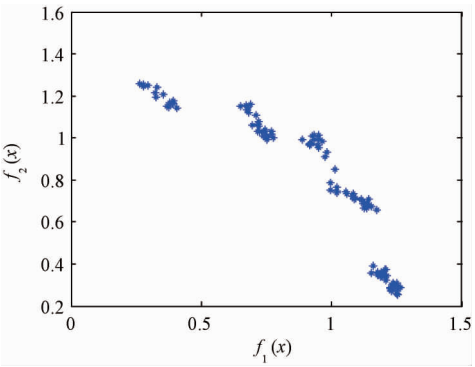

图5 FRNSGA-Ⅱ算法TNK函数仿真结果

Fig. 5 Simulation result of FRNSGA-Ⅱ on TNK function

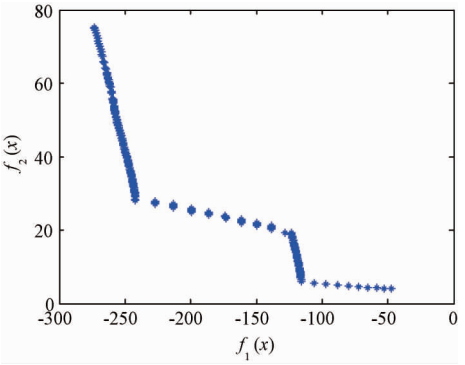

图6 TSDA算法OSY函数仿真结果

Fig. 6 Simulation result of TSDA algorithm on OSY function

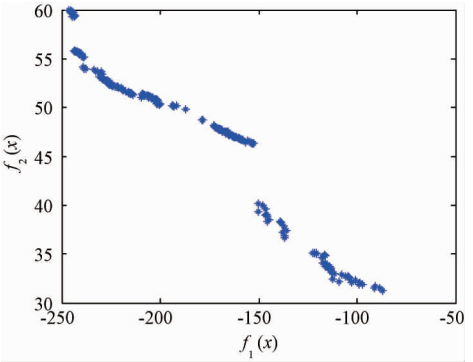

图7 FRNSGA-Ⅱ算法OSY函数仿真结果

Fig. 7 Simulation result of FRNSGA-Ⅱ on OSY function

具有一定优势.

然而,必须承认,本文算法TSDA在计算量上要高于FRNSGA-Ⅱ算法.接下来的研究将致力于如何降低算法的时间复杂度及本文算法的实际应用.

参考文献:

[1] 李智勇,黄滔,陈少淼,等.约束优化进化算法综述[J].软件学报,2017,28(6):1529-1546.

[2] 梁静,刘睿,瞿博阳,等.进化算法在大规模优化问题中的应用综述[J].郑州大学学报(工学版),2018,39(3):15-21.

[3] 孟红云,张小华,刘三阳.用于约束多目标优化问题的双群体差分进化算法[J].计算机学报,2008,31(2):228-235.

[4] DEB K. An efficient constraint handling method for genetic algorithms[J]. Computer methods in applied mechanics & engineering, 2000, 186(2):311-338.

[5] HAMMACHE A, BENALI M, AUBÉ F. Multi-objective self-adaptive algorithm for highly constrained problems: novel method and applications[J]. Applied energy, 2010, 87(8):2467-2478.

[6] 俞国燕,李鹏,何真,等. 一种用于多目标约束优化的改进进化算法[J]. 计算机集成制造系统,2009,15(6):1172 - 1178.

[7] 毕晓君,张磊,肖婧. 基于双种群的约束多目标优化算法[J]. 计算机研究与发展,2015,52(12):2813 - 2823.

[8] 暴励,曾建潮. 一种双种群差分蜂群算法[J]. 控制理论与应用,2011,28(2):266 - 272.

[9] 毕晓君,张磊. 基于混合策略的双种群约束优化算法[J]. 控制与决策,2015,30(4):715 - 720.

[10] 肖易寒,李明逵,陈立伟. 基于改进 NSGA-Ⅱ算法的多光谱测温数据处理[J]. 应用科技, 2017, 44(1): 33 - 39.

[11] DEB K, PRATAP A, MEYARIVAN T. Constrained test problems for multi-objective evolutionary optimization[C]//Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization, EMO 2001. Zurich, Switzerland:EMO, 2001:284 - 298.

[12] 韩红艳. 基于 Pareto 支配的高维多目标进化算法研究[D]. 大连:大连理工大学计算机科学与技术研究, 2016.

[13] JIANG S, YANG S. A steady-state and generational evolutionary algorithm for dynamic multi-objective optimization [J]. IEEE transactions on evolutionary computation, 2017, 21(1): 65 - 82.

Constraint Optimization Algorithm with Two-Stage and Three-Archive

LI Erchao, LI Jin

(School of Electrical Engineering and Information Engineering, Lanzhou University of Technology, Lanzhou 730000, China)

Abstract: Constrained optimization algorithm adopted the similav evolutionary strategy to deal with solutions located on the Pareto edge and solutions with poor function values, which could lead to poor search results. Aiming to solve this problem, a constrained optimization algorithm with two-stage and three-archive was proposed. The algorithm was divided into two stages. In the first stage, the population was divided into three archives according to the $\varepsilon(t)$ value. These archives were non-dominated solution archives, dominant solution archives, and non-dominated feasible solution archives, respectively. The dual optimization of hybrid strategy is applied to the non-dominated solution archives. It could not only avoid being trapped in local optimum, but also accelerate the convergence of solutions near the frontier. The dominant solution archives focused on the global search, which was beneficial for the algorithm to search better feasible solution. The non-dominated solution archives and the dominant solution archives were evolved using different optimization strategies to improve the optimization capability of algorithm. In the second stage, non-dominated sorting was performed on individuals when the first stage reached the certain generation. These individuals were concentrated from each generation to the non-dominated feasible solution archives. The selected N individuals were the optimal solution. Finally, the proposed algorithm was compared with other constrained multi-objective evolutionary algorithms on the three classical constraint test functions. The simulation results showed that the proposed algorithm had advantages in different kind of constraints.

Key words: constraint optimization; three-archive; hybrid strategy; two-stage; search ability