第 27 卷第 1 期
2010 年 1 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 27 No. 1
Jan. 2010

文章编号: 1000−8152(2010)01−0032−07

改进自适应变空间差分进化算法

姚　峰, 杨卫东, 张　明, 李仲德

(北京科技大学 信息工程学院, 北京 100083; 北京科技大学 钢铁流程先进控制教育部重点实验室, 北京 100083)

摘要: 在基本差分进化算法的基础上融入自适应变空间思想, 提出自适应变空间差分进化算法, 在进化代数达到预设周期整数倍时, 按变空间算法自动扩展或收缩搜索空间, 实现了自动寻找合适搜索空间、提高收敛速度和精度的目的. 此外为了进一步的加快收敛速度, 对原变空间算法进行了改造, 对其上下限的变化规则进行了修改和添加, 提出了改进的变空间算法. 仿真结果表明改进方法在收敛精度、速度上优于基本差分进化算法和基于原变空间算法的差分进化算法. 最后将其应用到热连轧机精轧机组负荷分配优化计算中, 为其提供了一种有效的优化手段.

关键词: 改进自适应变空间算法; 差分进化算法; 热连轧机; 负荷分配

中图分类号: TP18　　**文献标识码**: A

Improved space-adaptive-based differential evolution algorithm

YAO Feng, YANG Wei-dong , ZHANG Ming, LI Zhong-de

(School of Information Engineering, University of Sciences and Technology Beijing, Beijing 100083, China;
Key Laboratory of Advanced Control of Iron and Steel Process (Ministry of Education),
University of Science and Technology Beijing, Beijing 100083, China)

Abstract: To improve the performance of differential evolution algorithm we present a differential evolution algorithm with space-adaptive idea, which expands or shrinks the search space by certain rules. It realizes the automatic search for the suitable space and improves the convergence rate and accuracy. For further improvement of the convergence rate, the original space-adaptive algorithm is modified by revising the existing rules and adding new rules. The simulation results show that the improved space-adaptive-based differential evolution algorithm is better than the original differential evolution algorithm in convergence rate and accuracy. This algorithm is applied to several hot strip mills for the optimal design of scheduling.

Key words: improved space adaptive algorithm; differential evolutional algorithm; hot strip mills; load distribution

1　引言(Introduction)

差分进化算法(differential evolution, DE)是一种新的进化计算技术, 具有良好的优化性能, 但是对于多峰值函数以及搜索空间较大时, 算法收敛速度较慢, 且易早熟.

其实质是实数编码的遗传算法, 但子代的生成方法不同, 其在子代生成过程中用到了父代多个个体的线性组合, 而不是遗传算法传统单一的父代染色体交叉技术. 同时, 差分进化算法子代的生成采用“贪婪”的选择方式, 这种方式能使算法具有快速的收敛特性, 但增加了算法限入局部最优或早熟收敛的概率. 产生早熟收敛的根本原因是随迭代次数的增加种群多样性的快速下降. 增加种群的规模在一定程度上能克服早熟收敛, 但大大增加了算法对适应度函数的计算量.

为提高DE的性能, 一些学者提出了改进的方法. 文献[1]提出一种广义的变异策略框架, 方便用户选择合适的变异操作类型, 同时也为开发新的变异操作算子提供了便利; 文献[2]在DE中引入三角法变异, 将个体看作一个超三角形的中心点, 沿着由3组加权差分向量所构成的超三角形的3条边, 分别以不同的步长移动来产生新的变异个体, 从而增加了算法跳出局部极小点的概率; 文献[3]在DE中引入加速和迁移操作, 其中加速操作利用梯度信息将最优个体引向更优的区域, 而为了防止算法早熟收敛, 当种群的分散度低于一定的阈值时, 利用迁移操作在最优个体附近区域重新产生新个体, 并替换旧个体, 从而维持了种群的多样性; 文献[4]提出了多种群DE, 并用于求解多极值的优化问题; 文献[5]通过设计自适应收敛因子构建自调整的权重质心变异策略, 能

收稿日期: 2008−10−23; 收修改稿日期: 2009−04−18.
基金项目: 国家高技术专项项目(2005–1).

有效提高算法后期的局部增强能力.

本文结合自适应变空间思想, 提出一种自适应变空间差分进化算法(space adaptive differential evolution, SADE), 用以克服群体搜索的盲目性以及早熟问题. 同时为了使搜索的速度更快, 对原有自适应变空间算法进行了改造, 发展了一种改进的自适应变空间差分进化算法(improved space adaptive differential evolution, ISADE).

热连轧精轧机组的负荷分配优化, 就是如何确定各机架的出口厚度, 使整个机组工作状态达到最优. 负荷分配的合理与否不仅直接决定轧制过程应力状态特性, 而且对板形和整个生产流程有着很大的影响. 因此, 在进行各机架设定计算时, 寻找一个最优的压下负荷分配方案, 对有效利用设备能力, 保证产品质量起着很重要的作用[6]. 传统的多目标函数的优化设定不适合在线的计算[7] .

最后将ISADE应用到热连轧精轧机组的负荷分配优化计算中, 解决了设定模型的在线计算问题,仿真结果显示了该方法的有效性.

2　基本DE算法(Differential evolution)

DE是一种基于群体进化的算法, 具有记忆个体最优解和种群内信息共享的特点, 即通过种群内个体间的合作与竞争来优化问题的解[8].

算法首先要取得一组随机初始化的种群:

$$X^0 = \left[x_1^0; x_2^0; \cdots; x_{NP}^0\right],$$

NP是种群规模. 经过一系列规定的操作, 第s代个体进化为

$$x_i^s = \left[x_{i,1}^s, x_{i,2}^s, \cdots, x_{i,D}^s\right],$$

式中D为所优化问题的维数.

父代两个不同随机个体相减得到的差分矢量加到随机选择的第3个个体上, 生成一个变异个体, 接着按照一定的概率, 父代个体与变异个体之间进行交叉操作, 生成一个试验个体, 然后在父代个体与试验个体之间根据适应度函数值的大小进行选择操作, 选择适应度更优的个体作为子代[9].

2.1　变异操作(Mutation operation)

变异可以防止进化陷于局部极值, 其方式有很多种, 在这里列出两种基本的变异方式, DE/rand/bin和DE/rand/best:

$$x_{\mathrm{m}} = x_{s3}^s + F * (x_{s1}^s - x_{s2}^s), \tag{1}$$

$$x_{\mathrm{m}} = x_{\mathrm{gbest}}^s + F * \left[(x_{s1}^s - x_{s2}^s) + (x_{s3}^s - x_{s4}^s)\right]. \tag{2}$$

式(1)和(2)中: $x_{s1}^s, x_{s2}^s, x_{s3}^s, x_{s4}^s$为互不相同的随机个体; x_{gbest}^s为当前种群中适应度最好的个体; $F \in [0, 2]$为缩放因子.

2.2　交叉操作(Crossover operation)

交叉策略为: 假设种群中的个体x_i^s与x_{m}进行交叉操作产生试验个体x_{T}, 为保证个体的进化, 首先通过随机选择使x_{T}至少有一位由x_{m}贡献. 其他位则利用交叉概率因子CR, 交叉操作方程为

$$x_{\mathrm{T}j} = \begin{cases} x_{\mathrm{m}j}, & \mathrm{rand} \leqslant CR, \\ x_{ij}^s, & \mathrm{rand} > CR, \end{cases} \quad j = 1, 2, \cdots, D. \tag{3}$$

2.3　选择操作(Selection operation)

选择操作采用“贪婪”的搜索策略, 谁的适应值高, 就选择谁作为子代:

$$x_i^{s+1} = \begin{cases} x_{\mathrm{T}}, f(x_{\mathrm{T}}) < f(x_i^s), \\ x_i^s, f(x_{\mathrm{T}}) \geqslant f(x_i^s). \end{cases} \tag{4}$$

反复进行上述操作, 直到产生满足适应值条件的子代, 结束[10~12].

3　改进自适应变空间差分进化算法(ISADE algorithm)

由于基本DE算法的搜索空间是固定的, 不可避免的存在群体搜索的盲目性, 如果搜索空间过大, 则算法的收敛速度变慢且容易陷入局部最优解; 如果搜索空间过小, 则可能搜索不到全局最优解. 所以当基本DE算法用于高维多峰值函数优化时易出现早熟, 且收敛精度和速度不是很理想. 这里采用文献[13]提出的一种自适应变空间策略, 根据当前解的分布情况按照一定的规则自动的调整解的搜索空间, 使算法能够通过调整逐渐运行在相对合适的搜索空间上. 同时为了加快自适应变空间算法的收敛速度, 对其上下界的变更规则进行了合理的修改和补充.

3.1　改进自适应变空间思想(The mind of ISA)

改进自适应变空间思想分为两个阶段:扩展阶段和收缩阶段.首先令搜索空间为

$$\left[l^t, u^t\right] = \left\{\left[l_k^t, u_k^t\right], k = 1, 2, \cdots, n\right\},$$

t为搜索空间的更新代数.

令$t = 0$, 设定的初始搜索空间为$[l^0, u^0] = \{[l_k^0, u_k^0], k = 1, 2, \cdots, n\}$, 则可得初始“零点” $x_0 = (l_k^0 + u_k^0)/2$.

扩展阶段: 考虑搜索空间的上界, 如果当前种群中最优个体x^*的变量x_k^*满足$x_k^* \in (u_k^t/2, u_k^t)$, 则说明该变量当前搜索空间的上界$u_k^t$以及前一搜索空间的上界$u_k^{t-1}$都偏小, 为了使最优解包含在搜索空间中, 应对其进行扩展. 用类似的方法处理下界即可. 处理方法如下:

1) 若$x_k^* \in (u_k^t/2, u_k^t)$, 则$t = t + 1, u_k^t = 2x_k^*$;

2) 若$x_k^* \in (l_k^t, l_k^t/2)$, 则$t = t+1, l_k^t = 2x_k^*$. 若所有参数的搜索空间都未扩展, 则转入收缩阶段.

收缩阶段: 当扩展阶段完成之后, 这时的最优解已经包含在搜索空间中了, 但是给出的搜索空间比较粗糙, 因此在收缩阶段采用精细的扩展操作和收缩算子来细化搜索空间.

同样考虑搜索空间的上界, 由于所考察区间在初始“零点”x_0的正方向, 所以在对上界的调整中统一增加对下界的收缩操作, 以当前搜索空间的上界u_k^t以及前一搜索空间的上界u_k^{t-1}作为度量标准, 如图1所示.

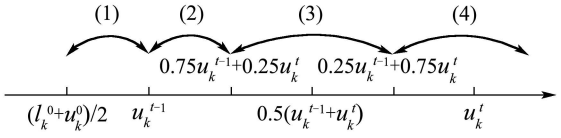


图 1 空间划分

Fig. 1 Space division

情况1中, x_k^*小于u_k^{t-1}, 表示当前以及前一搜索空间的上界u_k^t和u_k^{t-1}均过大, 应将其进行缩小, 采用如下处理方式:

情况 1 若$x_k^* \in ((l_k^0+u_k^0)/2, u_k^{t-1})$, 则

$$t = t+1, u_k^t = u_k^{t-2},$$
$$u_k^{t-1} = 0.5\left[u_k^{t-2} + (l_k^0+u_k^0)/2\right], l_k^t = l_k^{t-2},$$
$$l_k^{t-1} = 0.5\left[l_k^{t-2} + (l_k^0+u_k^0)/2\right];$$

情况2中, x_k^*略大于u_k^{t-1}, 而远小于u_k^t, 说明当前搜索空间上界u_k^t过大, 为获得x_k^*的较好的逼近, 采用如下处理方式:

情况 2 若$x_k^* \in [u_k^{t-1}, 0.75u_k^{t-1}+0.25u_k^t)$, 则

$$t = t+1, u_k^t = 0.5(u_k^{t-2}+u_k^{t-1}),$$
$$u_k^{t-1} = 1.2u_k^{t-2}, l_k^t = l_k^{t-2},$$
$$l_k^{t-1} = 0.5\left[l_k^{t-2} + (l_k^0+u_k^0)/2\right];$$

情况3中, x_k^*远大于u_k^{t-1}, 说明前一搜索空间的上界u_k^{t-1}过小, 为获得足够的搜索空间, 采用如下处理方式:

情况 3 若$x_k^* \in (0.25u_k^{t-1}+0.75u_k^t, \infty)$, 则

$$t = t+1, u_k^{t-1} = 0.5(u_k^{t-2}+u_k^{t-1}),$$
$$u_k^t = 1.2x_k^*, l_k^t = l_k^{t-2},$$
$$l_k^{t-1} = 0.5\left[l_k^{t-2} + (l_k^0+u_k^0)/2\right];$$

情况3的搜索空间合适不做调整. 经过这样的调整后, 真正的最优个体在$0.5(u_k^{t-1}+u_k^t)$附近, 可以认为这个界是合适的. 用类似的方法处理下界即可, 其相应的处理方式为:

情况 4 若$x_k^* \in (l_k^{t-1}, (l_k^0+u_k^0)/2)$, 则

$$t = t+1, l_k^t = l_k^{t-2},$$
$$l_k^{t-1} = 0.5\left[l_k^{t-2} + (l_k^0+u_k^0)/2\right], u_k^t = u_k^{t-2},$$
$$u_k^{t-1} = 0.5\left[u_k^{t-2} + (l_k^0+u_k^0)/2\right];$$

情况 5 若$x_k^* \in (0.75l_k^{t-1}+0.25l_k^t, l_k^{t-1}]$, 则

$$t = t+1, l_k^t = 0.5(l_k^{t-2}+l_k^{t-1}),$$
$$l_k^{t-1} = 1.2l_k^{t-2}, u_k^t = u_k^{t-1},$$
$$u_k^{t-1} = 0.5\left[u_k^{t-2} + (l_k^0+u_k^0)/2\right];$$

情况 6 若$x_k^* \in (-\infty, 0.25l_k^{t-1}+0.75l_k^t)$, 则

$$t = t+1, l_k^{t-1} = 0.5(l_k^{t-2}+l_k^{t-1}),$$
$$l_k^t = 1.2x_k^*, u_k^t = u_k^{t-2},$$
$$u_k^{t-1} = 0.5\left[u_k^{t-2} + (l_k^0+u_k^0)/2\right].$$

当某个变量的搜索空间发生变化时, 则在变化的搜索空间中随机产生NP个个体并计算其适应度, 选择适应度较好的个体替换原种群适应度较差的个体, 选取替换的比例为5%∼15%, 该比例选取过大易出现早熟.

与原有变空间算法不同的是, 在收缩阶段当搜索空间上界(下界)按照原有算法规则改变时, 相应的为其增加了下界(上界)的变化, 并且给出了一种简单有效的修改规则, 加快了自适应搜索的速度, 原变空间算法参见文献[13].

3.2 改进自适应变空间算法与差分进化算法结合(ISA algorithm combined with DE algorithm)

结合后的算法步骤:

Step 1 设定种群大小NP, 维数D, 缩放因子F, 交叉概率因子CR, 变空间周期NC以及初始搜索空间$[l^0, u^0]$, 令初始进化代数$s = 1$.

Step 2 在$[l^0, u^0]$中随机产生大小为NP的种群P_1,并计算其适应度.

Step 3 如果满足终止条件, 则停止并输出结果.

Step 4 按照式(1)或式(2)进行变异操作.

Step 5 分别按照式(3) (4)进行交叉和选择操作.

Step 6 如果s是NC的整数倍, 则根据上述改进自适应变空间思想更新$[l^{t-1}, u^{t-1}]$, $[l^t, u^t]$ 和P_1.

Step 7 令$s = s+1$, 转Step 3.

在上述算法中增加了一个参数: 变空间周期NC, 该参数选取过大则自适应变空间算法的优势发挥的不明显, 当选取过小时, 会增加算法的

计算复杂度, 这里选取$NC = 10$; 种群大小一般取$NP = (5 \sim 10)D$; 交叉概率因子$CR \in [0, 1]$, CR值大有利于局部搜索和加速收敛速率, CR值小有利于保持种群的多样性; 缩放比例因子$F \in [0, 2]$.

3.3 性能测试(Performance test)

将ISAD, SADE, DE, PSO和FEP算法进行对比, 所采用的测试函数为: $f1$为Dejong函数, $f2$为Griewank函数, $f3$为Rastrigin函数, $f4$为Schaffer函数, 函数$f1 \sim f3$的全局最优值均为0, 函数$f4$的全局最优值为-1.

为显示自适应变空间差分进化算法在大空间搜索上的优势, 取初时搜索空间为$x_i \in [-10^6, 10^6]$, 设置差分进化算法参数: 维数$D = 30$, 种群规模$NP = 200$, 迭代最大次数20000, 交叉概率因子$CR = 0.6$, 缩放因子$F = 1$. 为凸显ISADE性能的优势, 将ISADE、SADE与DE算法设置相同的参数, 并且都采用变异方式(1), 变空间周期$NC = 10$; 微粒群算法(PSO)参数: 维数$D = 30$, 种群规模$NP = 60$, 惯性权值$w = 0.8$, 加速系数$c1 = c2 = 2$; 快速进化规划算法(FEP)参数: 维数$D = 30$, 种群规模$NP = 60$, 步长初值$b0 = 10^4$, 步长下界$b_{\min} = 10^{-4}$, 锦标赛规模$q = 30$. 对$f1 \sim f4$进行30次仿真求平均值, 结果见表1所示.

表 1　ISADE, SADE与DE各运行30次求平均值

Table 1　ISADE, SADE and DE run 30 times for average

f	算法	最优性能	平均性能	迭代次数	方差	t/s
	ISADE	5.96×10^{-9}	7.92×10^{-9}	1574	1.77×10^{-18}	83.0
	SADE	6.36×10^{-9}	8.16×10^{-9}	2645	2.32×10^{-18}	130.7
$f1$	DE	8.11×10^{-9}	9.19×10^{-9}	4384	1.69×10^{-18}	50.8
	PSO	6.18×10^{4}	7.42×10^{4}	20000	2.34×10^{7}	78.6
	FEP	4.37×10^{7}	4.64×10^{7}	20000	7.48×10^{13}	335.3
	ISADE	5.53×10^{-9}	7.81×10^{-9}	1459	4.25×10^{-18}	95.2
	SADE	6.51×10^{-9}	8.52×10^{-9}	2322	5.35×10^{-18}	142.3
$f2$	DE	7.48×10^{-9}	9.11×10^{-9}	4478	8.10×10^{-18}	70.3
	PSO	4.82×10^{1}	5.23×10^{1}	20000	8.38×10^{1}	113.0
	FEP	5.85×10^{3}	6.15×10^{3}	20000	1.17×10^{5}	236.6
	ISADE	6.47×10^{-9}	7.08×10^{-9}	1781	3.82×10^{-18}	149.9
	SADE	7.01×10^{-9}	8.15×10^{-9}	2760	1.35×10^{-18}	188.5
$f3$	DE	1.60×10^{2}	1.62×10^{2}	20000	4.22	224.0
	PSO	6.55×10^{4}	7.91×10^{4}	20000	2.57×10^{7}	114.6
	FEP	3.35×10^{7}	3.86×10^{7}	20000	1.03×10^{13}	239.5
	ISADE	-1	-1	273	9.32×10^{-18}	4.2
	SADE	-9.91×10^{-1}	-1	640	2.20×10^{-5}	10.1
$f4$	DE	-8.71×10^{-1}	-5.42×10^{-1}	20000	1.39×10^{-2}	22.1
	PSO	-9.92×10^{-1}	-9.87×10^{-1}	17332	1.48×10^{-3}	86.5
	FEP	-1	-9.23×10^{-1}	6246	3.82×10^{-3}	27.9

测试函数如下:

$$f1 = \sum_{i=1}^{n} x_i^2,$$

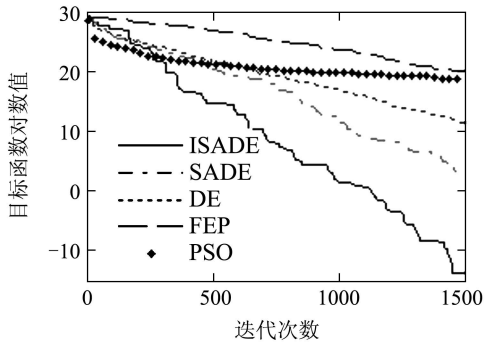
$$f2 = \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1\,,$$

$$f3 = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10),$$

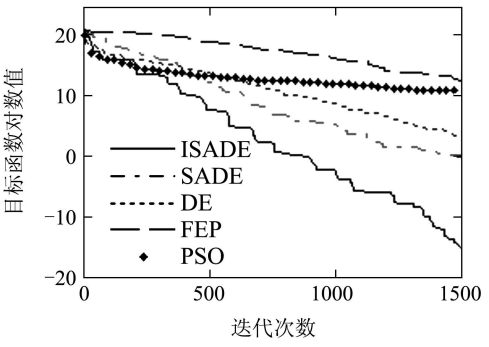
$$f4 = \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2} - 0.5.$$

在收敛速度上的排序为: ISADE>SADE>DE. 而在收敛精度上, ISADE略优于SADE, 两者精度明显好于DE, 尤其在$f3$和$f4$上的表现更加明显, 基本DE在这两个函数上均出现早熟现象, 相距最优值很远, 而ISADE和SADE算法均能找到全局最优值点. 在最优收敛值方差中, ISADE小于SADE, 两者都小于DE的. 在CPU时间上, 由于ISADE和DE相对于基本DE算法增加了很多判断步骤, 增加了算法的计算复杂度, 所以运行时间相对较长.

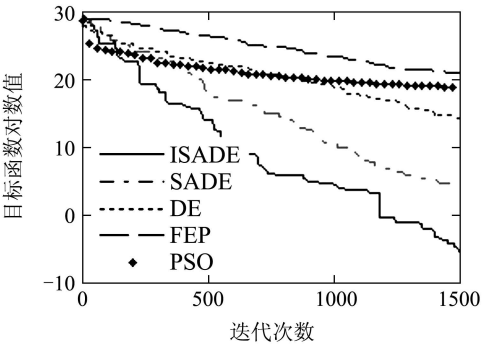
而另外两种算法FEP算法和PSO算法在高维多模态函数优化上的性能不好, 收敛速度很慢并且容易出现早熟. 图2(a)~(d)为几种算法在测试函数$f1 \sim f4$上各运行一次的目标函数值收敛曲线, 纵坐标采用对数坐标以便于比较, 由于测试函数$f4$的最优值为负数, 这里取其收敛值相反数的对数作为参考. 图2(a)~(d)的收敛曲线表明SADE的收敛速度优于基本DE, 而ISADE的收敛速度最快, 并且改进算法可在一定程度上避免算法的早熟, 提高收敛精度, 通过仿真结果得到了很好的验证.



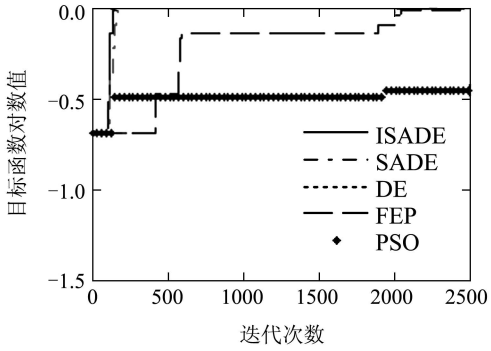
(a) $f1$目标函数值收敛曲线



(b) $f2$目标函数值收敛曲线



(c) $f3$目标函数值收敛曲线



(d) $f4$目标函数值收敛曲线

图 2 目标函数值收敛曲线

Fig. 2 Convergence curve of the objective function values

4 ISADE负荷分配算法(ISADE load distribution algorithm)

4.1 目标函数和约束条件(Objective function and constraints)

负荷分配的优化策略为: 前几个机架以负荷均衡为目标, 后几个机架则综合考虑板形板厚精度和性能要求. 采用目标函数的形式如下[1,2,8]:

$$\begin{aligned} J = \min\{&(P_1 - KP_2)^2 + (P_2 - P_3)^2 + \\ &\lambda_1(h_n - \widetilde{h}_n)^2 + \lambda_2(CR_n - \widetilde{CR}_n)^2 + \\ &\lambda_3 \sum_{i=4}^{7} (CR_i/h_i - CR_n/h_n \pm \Delta)^2\}. \end{aligned} \tag{5}$$

K可根据工艺条件选取; $\lambda_1 \sim \lambda_3$为加权系数; P_i为各机架的实测轧制力; h_n为成品带钢厚度; $\tilde{h}_n$为成品带钢目标厚度; CR_n为出口实测凸度; $\widetilde{CR}_n$为出口目标凸度; Δ为调节量.

由于带钢板坯的厚度较薄, 并且穿带的速度又不是很高, 在优化模型中不必考虑咬入条件的限制. 确定的约束条件如下[8]:

$$\begin{cases} 0 \leqslant P_i \leqslant P_{\max}, \\ 0 \leqslant I_i \leqslant I_{\max}, \\ h_{i+1} < h_i. \end{cases} \tag{6}$$

4.2 计算模型(Computing model)

经典的轧制力计算模型:

$$P_{\rm m} = 1.15 B_{\rm c} l'_{\rm c} Q_{\rm p} \sigma, \tag{7}$$

$l'_{\rm c}$计算公式:

$$l'_{\rm c} = \sqrt{R(1 + 2.2 \times 10^{-4} P_{\rm m}/B)}. \tag{8}$$

式(7)和(8)中: $B_{\rm c}$为轧件平均宽度; $l'_{\rm c}$为考虑轧辊压扁的变形区长度; $Q_{\rm p}$为应力状态系数; σ为变形阻力; R为轧辊半径.由于$P_{\rm m}$和$l'_{\rm c}$的计算相互关联,

可以先初设$l'_{\rm c}$, 代入式(7)计算$P_{\rm m}$, 而后将$P_{\rm m}$代入式(8)再计算$l'_{\rm c}$, 如此反复, 经过不到5次迭代便可求的准确的$P_{\rm m}$和$l'_{\rm c}$[14].

速度计算采用武钢按成品厚度规格通过查表设定末机架穿带速度的方法, 然后按照秒流量相等原则计算各机架的穿带速度. 速度计算公式为

$$\begin{cases} B_ih_iv_i = B_nh_nv_n, \\ v_i = v_{i0}(1+f_i), \\ f_i = \sqrt{0.028\varepsilon + 0.00064} - 0.0264. \end{cases} \tag{9}$$

式中: B为板宽; h为板厚; v为板带速度; f为前滑; ε为相对压下量[14].

热连轧生产线上最可靠的测温点是粗轧出口处, 因此各机架轧制温度均由粗轧出口温度计算得到

$$\begin{cases} T_{\rm FT0} = 6\varepsilon\sigma\tau/(\gamma CH_{\rm RC}) + 100(T_{\rm RC}/100)^{-3}, \\ \dfrac{(T_i - T_{\rm w})}{(T_{\rm FT0} - T_{\rm w})} = \exp(-K_{\rm a}\sum\limits_{j=1}^{i} L_j/(h_nv_n)). \end{cases} \tag{10}$$

式中: σ为玻尔兹曼常数; ε为黑度; γ为密度; τ为轧件从粗轧出口测温点到精轧入口测温点的时间; C为比热容; $H_{\rm RC}$为粗轧出口板厚; $T_{\rm RC}$为粗轧出口实测温度; $T_{\rm FT0}$为精轧入口估算温度; $T_{\rm w}$为机架喷水水温; $K_{\rm a}$为综合冷却系数; L_j为机架间距[14].

4.3 优化步骤(Optimization steps)

基于ISADE算法的热连轧精轧机组负荷分配优化步骤:

1) 读取设备、轧件、工艺初始参数及带钢成品参数;

2) 利用负荷分配经验公式确定各机架出口厚度的基础值, 采用二分法确定参数的搜索空间:

$$\begin{cases} X_{\min} = [\bar{h}_{i1} + \bar{h}_{i2}, \bar{h}_{i2} + \bar{h}_{i3}, \cdots, 2\bar{h}_{i7} + \delta]/2, \\ X_{\max} = [H_0 + \bar{h}_{i1}, \bar{h}_{i1} + \bar{h}_{i2}, \cdots, \bar{h}_{i6} + \bar{h}_{i7}]/2. \end{cases}$$

在该搜索空间中初始化种群;

3) 利用公式(7)~(10)计算各机架设定参数;

4) 调用ISADE算法程序, 并且判断个体是否满足约束条件(6), 若不满足则将其适应值置很大值, 表示将其在选优操作中淘汰掉, 寻找使J最小的厚度分配方案h_i;

5) 输出最优负荷分配值.

5 仿真研究(Simulation)

仿真采用的钢种为Q235, 板宽B = 1535 mm, 来料厚度H_0 = 36.7 mm, 成品厚度为h_n = 5.7 mm, 粗轧出口实测温度$T_{\rm RC}$ = 1340 K, 目标凸度CR_n= 0.01 mm.

设定种群大小$NP = 40$, 维数$D = 7$, 交叉概率因子$CR = 1.0$[12,15,16]. 3种不同方法的轧制力分配见图3, 其余计算值见表2~3.

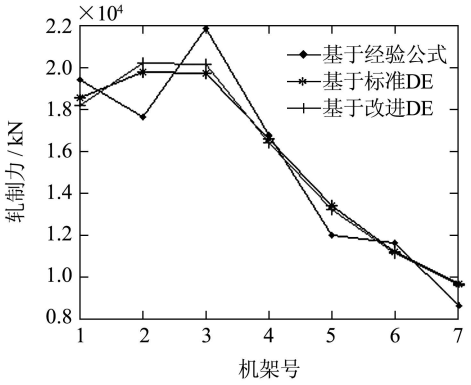


图 3　轧制力分配对比

Fig. 3　Rolling power distribution comparison

表 2　各机架相对凸度

Table 2　The rack relative crown

	经验法	DE	ISADE
CR_1/h_1	0.0008	0.0007	0.0007
CR_2/h_2	0.0010	0.0011	0.0011
CR_3/h_3	0.0017	0.0016	0.0016
CR_4/h_4	0.0018	0.0017	0.0017
CR_5/h_5	0.0015	0.0017	0.0017
CR_6/h_6	0.0018	0.0017	0.0017
CR_7/h_7	0.0015	0.0017	0.0017

表 3　各机架出口厚度分配

Table 3　Thickness distribution

mm	经验法	DE	ISADE
h_1	25.49	25.99	26.20
h_2	18.53	18.03	18.21
h_3	12.65	12.69	12.83
h_4	9.54	9.69	9.76
h_5	7.84	7.84	7.87
h_6	6.52	6.60	6.61
h_7	5.70	5.70	5.70

对比仿真结果发现: 基于ISADE算法的负荷分配优化的收敛精度要高于基本DE算法的, 并且收敛速度很快, 其仿真时间小于3 s, 迭代步数小于100, 明显要优于基于免疫遗传算法的负荷分配(<10 s)[2], 满足在线计算的要求.

6 结语(Conclusion)

本文对基本差分进化算法有两点改进: 一是将自适应变空间思想与差分进化算法结合, 提出了自适应变空间差分进化算法, 提高了收敛速度、收敛精度, 并可有效防止算法早熟; 二是对原有变空间思想进行了改进. 仿真结果表明改进的变空间算法具有更快的收敛速度和更高的收敛精度. 最后将其应用到热连轧精轧机组的优化计算中, 为其提供了一条更为有效的优化手段.

参考文献(References):

[1] FEOKTISTOV V, JANAQI S. Generalization of the strategies in differential evolution[C] //*Proceedings of the 18th International Parallel and Distributed Processing Symposium*. Santa Fe: [s.n.], 2004: 165 – 170.

[2] FAN H Y, LAMPINEN J. A trigonometric mutation operation to differential evolution[J]. *Journal of Global Optimization*, 2003, 27(1): 105 – 129.

[3] WANG F S, JING C H, TSAO G T. Fuzzy-decision-making problems of fuel ethanol production using a genetically engineered yeast[J]. *Industrial & Engineering Chemistry Research* , 1998, 37(8): 3434 – 3443.

[4] ZAHARIE D. A multipopulation differential evolution algorithm for multimodal optimization[C] //*The 10th International Conference on Soft Computing*. Mendel: [s.n.], 2004: 16 – 18.

[5] 张贵军, 王信波, 俞力, 等. 求解高维多模优化问题的自适应差分进化算法[J]. 控制理论与应用. 2008, 25(5): 862 – 866.
(ZHANG Guijun, WANG Xinbo, YU Li, et al. Adaptive differential evolution for high-dimension multimodal optimization problems[J]. *Control Theory & Applications*, 2008, 25(5): 862 – 866.)

[6] 王建辉, 徐林, 闫勇亮, 等. 改进粒子群算法及其对热连轧机负荷分配优化的研究[J]. 控制与决策. 2005, 21(12): 1379 – 1383.
(WANG Jianhui, XU Lin, YAN Yongliang, et al. Improved PSO and its application to load distribution optimization of hot strip mills[J]. *Control and Decision*, 2005, 21(12): 1379 – 1383.)

[7] 王焱, 刘景录, 孙一康. 免疫遗传算法对精轧机组负荷分配的优化[J]. 北京科技大学学报. 2002, 24(3): 339 – 341.
(WANG Yan, LIU Jinglu, SUN Yikang. Immune genetic algorithms (IGA) based scheduling optimization for finisher[J]. *Journal of University of Science and Technology Beijing*, 2002, 24(3): 339 – 341.)

[8] 刘波, 王凌, 金以慧. 差分进化算法研究进展[J]. 控制与决策. 2007, 22(7): 721 – 729.
(LIU Bo, WANG Lin, JIN Yihui. Advances in differential evolution[J]. *Control and Decision*, 2007, 22(7): 721 – 729.)

[9] 吴亮红, 王耀南, 周少武, 等. 双群体伪并行差分进化算法研究及应用[J]. 控制理论与应用. 2007, 24(3): 453 – 458.
(WU Lianghong, WANG Yaonan, ZHOU Shaowu, et al. Research and application of pseudo parallel differential evolution algorithm with dual subpopulations[J]. *Control Theory & Applications*, 2007, 24(3): 453 – 458.)

[10] KAELO P, ALI M M. A numerical study of some modified differential evolution algorithms[J]. *European Journal of Operational Research*, 2005, 169(3): 1176 – 1184.

[11] STORN R, PRICE K. Differential evolution–a simple and efficient adaptive scheme for global optimization over continuous spaces[R]. Berkeley : International Computer Science Institute 2006.

[12] ZHANG M, LUO W J, WANG X F. Differential evolution with dynamic stochastic selection for constrained optimization[J]. *Information Sciences*, 2008, 178(4): 3043 – 3074.

[13] 焦李成, 刘静, 钟伟才. 协同进化计算与多智能体系统[M]. 北京: 科学出版社, 2006.
(JIAO Licheng, LIU Jing, ZHONG Weicai. *Co-evolutionary Computation and Multi-Agent System*[M]. Beijing: Science Press, 2006.)

[14] 孙一康. 带钢热连轧的模型与控制[M]. 北京: 冶金工业出版社, 2002.
(SUN Yikang. *The Model and Control of Hot Strip Mill*[M]. Beijing: Metallurgical Industry Press, 2002.)

[15] MAYER D G, KINGHORN B P, ARCHER A A. Differential evolution–an easy and efficient evolutionary algorithm for model optimisation[J]. *Agricultural Systems*, 2004, 83(5): 315 – 328.

[16] 袁俊刚, 孙治国, 曲广吉. 差异演化算法的数值模拟研究[J]. 系统仿真学报. 2007, 19(20): 4646 – 4648.
(YUAN Jungang, SUN Zhiguo, QU Guangji. Simulation study of differential evolution[J]. *Journal of System Simulation*, 2007, 19(20): 4646 – 4648.)

作者简介:

姚 峰 (1984—), 男, 博士研究生, 目前研究方向为轧钢过程优化、建模, E-mail: yf2002043227@163.com;

杨卫东 (1952—), 男, 教授, 博士生导师, 目前研究方向为带钢连轧计算机控制;

张 明 (1981—), 男, 博士研究生, 目前研究方向为带钢热连轧精轧机组AGC控制;

李仲德 (1976—), 男, 博士研究生, 目前研究方向为预测控制在冷轧中的应用.