doi:10.6041/j.issn.1000-1298.2015.01.051

多邻域链式结构的多目标粒子群优化算法*

王亚辉　唐明奇
(华北水利水电大学机械学院, 郑州 450011)

摘要: 为了提高多目标粒子群算法求解多目标问题的性能,改善算法的收敛性,提出一种多邻域链式结构的多目标粒子群优化算法。首先,以一种环形链式拓扑结构,将种群划分为多个邻域,每个邻域之间相互交叉重叠,并针对不同位置的粒子,进行不同的速度和位置更新策略。其次,对所有粒子采用速度钳制策略,并引入差分进化策略对粒子进行扰动,从而进一步提高算法的多样性。通过 14 个无约束和 3 个有约束函数仿真实验,表明该算法相对于 NSGA-II、SPEA2、MOEA/D-DE、SMPSO 和 OMOPSO 算法,获得 Pareto 解集分布更加均匀,算法的收敛性和多样性也更好。为了进一步验证算法的可行性和有效性,将其应用于 72 杆桁架结构尺寸设计,并与其他优化方法进行了比较,结果表明该算法获得的 Pareto 前端更均匀,收敛性更好。

关键词: 多目标优化　粒子群算法　多邻域链式结构　速度钳制策略　差分进化策略　桁架结构
中图分类号: TP301.6　　**文献标识码**: A　　**文章编号**: 1000-1298(2015)01-0365-08

Optimization of Multi-objective Particle Swarm Algorithm Based on Multi-neighborhood Cycle-chain Structure

Wang Yahui　Tang Mingqi
(*College of Mechanical Engineering*, *North China University of Water Resources and Electric*, *Zhengzhou* 450011, *China*)

Abstract: In order to enhance the performance and convergence of multi-objective particle swarm optimization (MOPSO) algorithm for multi-objective optimization, a multi-neighborhood cycle-chain structure of multi-objective particle swarm optimization (MNCS-MOPSO) was proposed. Firstly, the population was divided into many neighborhoods. The mutual overlaps were existed between the adjacent neighborhood, and updating strategy was used for different velocity and position aimed at particles of different positions. In addition, velocity control strategy was adopted for all particles and differential evolution strategy was introduced to make disturbance. Comparing with NSGA-II, SPEA2, MOEA/D-DE, SMPSO and OMOPSO by testing 14 unconstraint and 3 constrain benchmark functions, simulation experiments showed that the proposed algorithm could obtain a more uniform distribution of Pareto solution set, and better convergence as well as diversity than those state-of-the-art multi-objective metaheuristics. In order to verify the performance of MNCS-MOPSO algorithm, classical 72-bar truss sizing optimization problems were used to demonstrate the feasibility and effectiveness of this algorithm, and the results were compared with other optimization methods. The results indicate that the MNCS-MOPSO provides better performance in the diversity, the uniformity and the convergence of the obtained solution than other methods.

Key words: Multi-objective optimization　Particle swarm algorithm　Multi-neighborhood cycle-chain structure　Speed control strategy　Differential evolution　Truss structure

收稿日期: 2014-10-15　修回日期: 2014-11-10
* 国家自然科学基金资助项目(51301070)
作者简介: 王亚辉,副教授,主要从事先进制造技术和现代设计方法研究,E-mail: wangyahui@ncwu.edu.cn

引言

科学研究与工程领域中的优化问题大都是多目标优化问题(Multi-objective optimization problems, MOPs),如机械优化设计、方案优选、调度等。通常,这些 MOPs 中的搜索空间很大,要想获得 Pareto 解集不仅需要耗费大量的计算时间,而且对算法本身也存在很大的考验,而进化算法在解决此类问题方面具有很多优势,许多学者对其进行了大量研究。迄今为止,国际上出现许多经典的多目标进化算法,如 NSGA-II[1]、SPEA2[2]、MOEA/D[3]、MOPSO[4]和 NNIA[5]等。

粒子群算法(PSO)是 Kennedy 等[6]于 1995 年提出的一种进化计算方法,其概念简单,易于编码,而且很多情况下比遗传算法更有效率,因此许多学者将 PSO 算法应用于求解多目标优化问题[7-11]。

虽然许多研究者对 MOPSO 算法进行了改进,但是粒子群在解决多目标问题时,仍然存在以下问题:算法的多样性保持能力差,易于陷入局部收敛;算法的平衡全局搜索和局部寻优能力不佳。基于此,本文提出一种多邻域链式结构的多目标粒子群优化算法(Multi-neighborhood cycle-chain structure of multi-objective particle swarm optimization, MNCS-MOPSO)。

1 粒子群优化算法

粒子群优化算法中,每个粒子的飞行方向和位置由自身的历史最优位置和种群历史最优位置协同调整。由于其具有理解简单、实现容易、计算效率高等特点,在许多领域得到了广泛的应用。

假设在 D 维决策空间中,粒子种群 NP 大小为 m,对于每个粒子 i 具有如下属性:在第 t 代时粒子所处的位置 $x_i(t)=(x_{i1},x_{i2},\cdots,x_{id})$,速度为 $v_i(t)=(v_{i1},v_{i2},\cdots,v_{id})$,粒子自身的历史最佳位置 $p_l=(p_{l1},p_{l2},\cdots,p_{ld})$,种群到达的最优位置 $p_g=(p_{g1},p_{g2},\cdots,p_{gd})$。第 $t+1$ 代的粒子更新公式为

$$v_{i,j}(t+1)=wv_{i,j}(t)+c_1r_1(p_{l,j}-x_{i,j}(t))+c_2r_2(p_{g,j}-x_{i,j}(t)) \quad (1)$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1) \quad (2)$$

式中 w——权重因子

c_1、c_2——学习因子

r_1、r_2——[0,1]的随机数

式(1)等式右侧,第 1 部分为惯性部分,反映了粒子目前的状态;第 2 部分是认知部分,反映了粒子对自身历史经验的记忆;第 3 部分是社会部分,反映了粒子间协同合作与知识共享的种群历史经验。

2 MNCS-MOPSO 算法原理

2.1 邻域划分

为了更加有效地传递种群的全局信息,并较好地保持邻域内的独立搜索性能,本文设计一种简单的邻域划分方案。每个邻域之间相互独立,且有重叠个体进行连接,构成一种相互连接、首尾相连的环形链式结构,如图 1a 所示。具体的邻域划分如图 1b所示,将种群 NP 中 m 个粒子按照索引号进行划分,假设有 N 个邻域,则每个单独链中的粒子数为 $k=\frac{m}{N}+1$,对于第 $j-1$ 个单链的最后一个粒子 $(j-1)k$,会作为第 j 个单链的第 1 个粒子,最后一个链的最后一个粒子为索引号为 1 的个体。例如粒子数 $m=100$,分为 10 个单链,则每个单链的粒子数为 $k=\frac{100}{10}+1=11$,第 1 个单链的邻域粒子索引号为 $\{1,2,\cdots,10,11\}$,第 2 个为 $\{11,12,\cdots,20,21\}$,最后一个单链邻域粒子索引号为 $\{90,91,\cdots,100,1\}$。

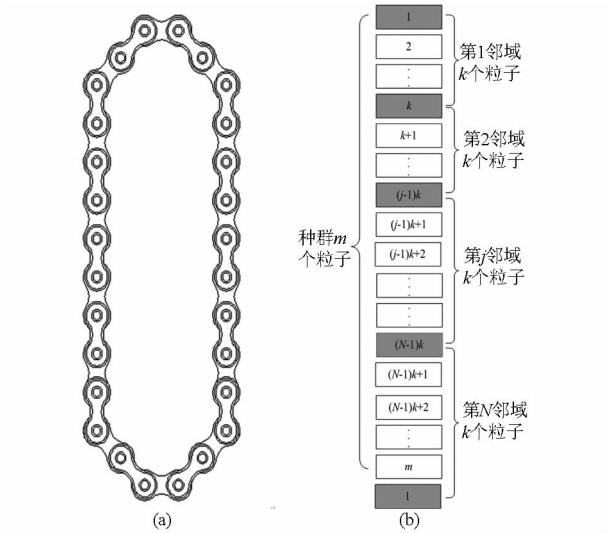

图 1 多邻域链式结构示意图

Fig. 1 Multi-neighborhood cycle-chain structure

(a) 环形链式结构 (b) 邻域划分方案

邻域信息传递方案:对于每个单链重叠的节点粒子,粒子自身历史最优位置 p_l 变为邻居历史最优位置 $p_{\text{neighborhoods}}$,链中间的粒子保持不变。对于邻居历史最优位置 $p_{\text{neighborhoods}}$ 的选择方式为

$$p_n=R_{\text{anking}}(f(p_{(j-1)k}),f(p_{(j-1)k+1}),\cdots,f(p_{(j-1)k+k})) \quad (3)$$

$$p_{\text{neighborhoods}}=S_{\text{orting}}(P_n) \quad (4)$$

式中 $p_{(j-1)k}$——粒子自身最优位置

$f(p_{(j-1)k})$——适应度函数

$R_{\text{anking}}(f(p_k))$——函数

p_n——邻居中粒子支配个数序号

$S_{\text{orting}}(P_n)$ ——函数，为选择邻居最优个体 $R_{\text{anking}}(f(p_k))$ 对不同粒子函数 $f(p_{(j-1)k})$ 之间的支配关系进行排序，其数值为该粒子被支配的其他粒子个数，如被一个个体支配，则 $R_{\text{anking}}(f(p_k))=1$，依次累加类推。

对于种群的历史最优位置 p_g，则采用随机的方式在外部文档中选取一个个体作为 p_g。

对于不同位置的粒子，其飞行速度和位置更新策略都不同。

单链中节点位置的粒子，采用 Clerc[12] 提出的收敛因子更新方式，则重叠粒子的更新公式为

$$v_{i,j}(t+1)=\gamma[v_{i,j}(t)+c_1r_1(p_{\text{neighborhoods},j}-x_{i,j}(t))+c_2r_2(p_{g,j}-x_{i,j}(t))] \tag{5}$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1) \tag{6}$$

$$\gamma=\frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|} \tag{7}$$

式中，γ 为收敛因子；如果 $c_1+c_2>4$，则 $\varphi=c_1+c_2$；如果 $c_1+c_2<4$，则 $\varphi=0$。

单链中间位置的粒子，速度和位置的更新公式为

$$v_{i,j}(t+1)=w_{\text{gen}}v_{i,j}(t)+c_1r_1(p_{l,j}-x_{i,j}(t))+c_2r_2(p_{g,j}-x_{i,j}(t)) \tag{8}$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1) \tag{9}$$

$$w_{\text{gen}}=(w_{\max}-w_{\min})\frac{g}{M}+w_{\min} \tag{10}$$

式中　w_{gen}——线性递减的权重因子

$w_{\max}$——权重因子最大值

$w_{\min}$——权重因子最小值

g——当前迭代数

M——最大迭代数

2.2　速度钳制策略

在 PSO 算法中，算法的全局搜索和局部寻优的平衡，主要在于粒子的速度更新处理方式。另外，文献[13]得出如果速度钳制操作配合收敛因子模型一起使用，算法的收敛速度会更快。

速度钳制一般处理办法为，如果粒子速度超过指定的速度最大值，就会被强制变为速度最大值。令 $V_{\max,j}$ 代表在第 j 维上允许的最大速度，$V_{\max,j}$ 一般选用决策变量的边界值，粒子的速度更新公式为

$$v_{i,j}(t+1)=\begin{cases}v'_{i,j}(t+1) & (v'_{i,j}(t+1)<V_{\max,j})\\ V_{\max,j} & (v'_{i,j}(t+1)\geqslant V_{\max,j})\end{cases} \tag{11}$$

本文引用一种改进的速度钳制速度策略[14]，能更好地控制粒子的飞行速度，有利于平衡算法的全局搜索和局部寻优。

$$v_{i,j}(t+1)=\begin{cases}0.5\Delta & (v_{i,j}(t+1)>\Delta)\\ -0.5\Delta & (v_{i,j}(t+1)<-\Delta)\\ v_{i,j}(t+1) & (\text{其他})\end{cases} \tag{12}$$

$$\Delta=u_j-l_j \tag{13}$$

式中　u_j、l_j——决策变量的上、下约束值

2.3　差分进化策略

粒子群算法易陷于局部收敛，主要原因是粒子飞行速度过快和进化过程种群多样性不佳导致。对于维护种群多样性，文献[15]发现差分进化算法在维护种群多样性及搜索能力方面功能较强。基于此，将一种差分进化策略加入粒子群算法中，加强对粒子的扰动，进一步提高种群的多样和搜索能力。

差分进化策略有多种进化类型，文献[16]分析了常见的差分进化策略特点，其中 DE/rand/1/bin 进化策略具有很好的全局探索能力，容易跳出局部最优解，有利于提高种群多样性。本文选用以 DE/rand/1/bin 进化策略为基础的差分交叉算子[17]。

具体操作过程为：对于个体 $x_{i,D}$，$i\in[0,m-1]$，D 为个体的决策变量个数。对于处于第 i 索引位置的父本向量 $\boldsymbol{x}_{i,D}=(x_{i,1},\cdots,x_{i,j},\cdots,x_{i,D})$ 中的每一分量 $x_{i,j}$，产生一个随机数 $r_j\in[0,1]$，比较 r_j 与交叉因子 R 或 j 与 K 的大小关系，以判断是否用 $v_{i,j}$ 来替换 $x_{i,j}$，以得到新个体 $\boldsymbol{u}_{i,D}=(u_{i,1},\cdots,u_{i,j},\cdots,u_{i,D})$，$K\in[0,D-1]$ 之间的整数，操作定义为

$$u_{i,j}=\begin{cases}x_{r1,j}+F(x_{r2,j}-x_{r3,j}) & (r_j<R\ ,\ j=K)\\ x_{i,j} & (\text{其他})\end{cases} \tag{14}$$

其中，$x_{r1,j}$、$x_{r2,j}$、$x_{r3,j}$ 为选择出来的 3 个父本，r_1、r_2、r_3 为 3 个父本的索引位置，F 为缩放因子。

2.4　约束处理

采用 Deb 等[1] 的多目标法处理违反约束的个体。具体如下：

（1）两个个体中，一个个体为可行解，另外一个个体为不可行解时，选择可行解。

（2）当两个个体均为可行解时，选择非支配的个体。

（3）当两个个体均为不可行解时，选择违反约束条件程度小的个体。

2.5　算法的时间复杂度分析

MNCS-MOPSO 算法的时间复杂度主要由 4 部分组成：粒子群操作的时间复杂度；差分进化的复杂度；利用子代个体对父代个体进行替换操作的时间复杂度；将优秀子代个体添加到外部文档的时间复杂度。

粒子的规模为 Z,目标的个数为 L,外部文档的大小为 H,则各个部分的复杂度为:粒子群操作的时间复杂度为 $O(Z)$;差分进化的时间复杂度为 $O(2Z)$;替换策略的时间复杂度为 $O(ZH)$;将粒子添加到外部文档的时间复杂度为 $O(ZLH)$。

因此,在每次迭代中,总的时间复杂度为

$$T = O(Z) + O(2Z) + O(ZL) + O(ZLH) \quad (15)$$

由于粒子群规模和外部文档的规模是相同的,即 $H = Z$,则时间复杂度是

$$T = O(Z) + O(2Z) + O(ZL) + O(LZ^2) \quad (16)$$

2.6 算法流程

在上述分析的基础上,给出 MNCS-MOPSO 算法的伪代码描述:

Pseudo code of MNCS-MOPSO

proc steps Up( )

particles←Create particles ( );

Evaluate each particles;

Identify the best position of particle p_{lbest};

Identify the best position of particle neighborhoods $p_{\text{neighborhoods}}$;

Identify the best position of population p_{gbest};

While not Termination Condition ( ) do

fori←to particles. popSize = m do

if(m mod i = = 0) //为单链节点粒子时

$v_i(t+1) = \gamma[wv_i(t) + c_1r_1(p_l - x_i(t)) + c_2r_2(p_g - x_i(t))]$;

$x_i(t+1) = x_i(t) + v_i(t+1)$;

else //为单链中间粒子时

$v_i(t+1) = w_{\text{gen}}v_i(t) + c_1r_1(p_l - x_i(t)) + c_2r_2(p_g - x_i(t))]$;

$x_i(t+1) = x_i(t) + v_i(t+1)$;

parent1←DE selection(particles);

parent2←DE selection(archive);

//差分进化交叉算子

DE Crossover (parents, individual);

evalutionFitness(offspring);

insert (position (individual), offspring, population);//替换策略

addToArchive(individual);

end for

Update p_{lbest}; //更新自身最优值

Update $p_{\text{neighborhoods}}$;//更新邻居最优个体

3 数值实验与分析

3.1 测试函数及算法的参数设置

为了验证 MNCS-MOPSO 算法在多目标优化问题上的求解性能,选取 ZDT 系列和 WFG 系列基准函数进行测试,为了求证在解决约束问题上的能力,选取了 Golinski、Tanaka、Srinivas 测试函数。同 MNCS-MOPSO 进行比较的算法为: NSGA-II[1]、SPEA2[2]、MOEA/D-DE[18]、OMOPSO[19] 和 SMPSO[14]算法。

所有算法在每个测试函数上的初始粒子种群 NP 大小均设置为 100,最大函数评价次数均设置为 250 次,外部文档规模为 100,其中 MNCS-MOPSO 算法的单链个数为 10 个,每个单链的邻域大小为 11 个个体,权重因子 $w_{\max} = 0.8$, $w_{\min} = 0.4$,学习因子 $C_1 \in [1.5, 2.5]$, $C_2 \in [1.5, 2.5]$,随机因子 $r_1 \in [0, 1)]$, $r_2 \in [0, 1]$,其他参数设置与各算法提出时所用参数设置一致。

3.2 实验结果与分析

仿真实验在硬件配置为 AMD CPU 3.00 GHz、内存 2.00 GB 的计算机上进行,采用 JAVA 语言编写,对比算法代码来源于 Nebro 和 Durillo 的 jMetal4.4 包[20]。将这 6 种算法分别对测试函数进行 30 次独立运行计算,统计分布性指标 HV[21](表 1)、多样性指标 Spread[1](表 2)和收敛性指标 Epsilon[22](表 3)的标准差平均值及方差进行对比分析。

从表 1 中 HV 性能指标看出,共 17 个测试函数,其中 MNCS-MOPSO 共求得 11 个最优值;NSGA-II 获得 1 个最优值;SPEA2 获得 2 个最优值;MOEA/D-DE 获得 3 个最优值;OMOPSO 和 SMPSO 算法未取得最优值。实验数据表明,MNCS-MOPSO 在求解 ZDT 系列函数时占有绝对优势,分布性最佳,说明多邻域链式结构、速度钳制策略和 DE 进化策略等有助于解集分布性的提高。对于 WFG 系列函数,分布性也较好,除了在 WFG1 函数分布性差于 NSGA-II,在 WFG4 和 WFG9 问题上分布性稍弱于 SPEA2,在 WFG8 问题求解上分布性稍弱于 SMPSO。对于约束函数,Srinivas 问题上 MNCS-MOPSO 表现最优,Golinski 和 Tanaka 上 MOEA/D-DE 表现最优。

从表 2 可知, MNCS-MOPSO 获得了 11 个最优值;SPEA2 获得 3 个最优值;OMOPSO 算法获得 2 个最优值,MOEA/D-DE 获得 1 个最优值,NSGA-II 和 SMPSO 算法未获得最优值。数据分析表明,MNCS-MOPSO 在求解 ZDT 系列函数时多样性能最佳,在求解 ZDT6 问题时 MOEA/D-DE 的多样性最好。对于 WFG 系列问题,MNCS-MOPSO 在 WFG(3,6,7,9)问题上多样性相对于另外 5 种算法表现最好;SPEA2 算法在 WFG(1,2,4)问题上多样性最好;

表 1　HV 标准差平均值及方差

Tab. 1　Mean and standard deviation of HV

测试	MNCS-MOPSO		NSGA-II		SPEA2		MOEA/D-DE		OMOPSO		SMPSO	
函数	均值	方差	均值	方差	均值	方差	均值	方差	均值	方差	均值	方差
ZDT1	0.662*	3.0×10^{-5}	0.659	2.5×10^{-4}	0.660	2.3×10^{-4}	0.641	6.2×10^{-3}	0.661	3.3×10^{-4}	0.662	1.1×10^{-4}
ZDT2	0.329*	4.3×10^{-5}	0.326	3.2×10^{-4}	0.325	3.4×10^{-3}	0.308	7.3×10^{-3}	0.328	2.8×10^{-4}	0.329	8.5×10^{-5}
ZDT3	0.516*	3.4×10^{-5}	0.515	1.6×10^{-4}	0.514	6.0×10^{-4}	0.445	2.3×10^{-2}	0.514	2.8×10^{-4}	0.515	4.3×10^{-3}
ZDT4	0.662*	3.8×10^{-5}	0.656	2.4×10^{-3}	0.650	1.1×10^{-2}	0.269	1.8×10^{-1}			0.662	1.7×10^{-4}
ZDT6	0.401*	6.1×10^{-5}	0.388	2.0×10^{-3}	0.379	3.0×10^{-3}	0.401	3.4×10^{-3}	0.401	7.8×10^{-5}	0.401	7.1×10^{-4}
WFG1	0.194	5.8×10^{-2}	0.529*	7.5×10^{-2}	0.410	9.1×10^{-2}	0.361	1.0×10^{-1}	0.129	2.6×10^{-2}	0.116	3.8×10^{-3}
WFG2	0.564*	1.7×10^{-4}	0.562	1.4×10^{-3}	0.562	1.6×10^{-3}	0.562	3.5×10^{-4}	0.563	2.7×10^{-4}	0.561	5.7×10^{-4}
WFG3	0.442*	4.0×10^{-5}	0.441	2.6×10^{-4}	0.441	3.0×10^{-4}	0.441	5.0×10^{-5}	0.442	8.5×10^{-5}	0.441	1.8×10^{-4}
WFG4	0.209	1.2×10^{-3}	0.217	3.8×10^{-4}	0.218*	2.5×10^{-4}	0.212	1.9×10^{-3}	0.206	1.2×10^{-3}	0.203	1.9×10^{-3}
WFG5	0.197*	1.3×10^{-3}	0.195	1.3×10^{-3}	0.196	1.4×10^{-3}	0.195	3.1×10^{-5}	0.196	5.1×10^{-5}	0.196	1.0×10^{-4}
WFG6	0.210*	6.4×10^{-5}	0.198	1.0×10^{-2}	0.201	1.1×10^{-2}	0.209	1.2×10^{-4}	0.210	1.4×10^{-4}	0.209	3.0×10^{-4}
WFG7	0.210*	5.7×10^{-5}	0.209	3.5×10^{-4}	0.210	1.5×10^{-4}	0.209	9.7×10^{-5}	0.210	1.4×10^{-4}	0.209	2.4×10^{-4}
WFG8	0.149	1.4×10^{-3}	0.149	8.5×10^{-3}	0.148	7.2×10^{-3}	0.153*	1.3×10^{-2}	0.146	9.0×10^{-4}	0.148	1.1×10^{-3}
WFG9	0.238	6.0×10^{-4}	0.237	1.5×10^{-3}	0.238*	2.1×10^{-3}	0.236	5.7×10^{-4}	0.237	4.3×10^{-4}	0.235	4.2×10^{-4}
Golinski	0.968	3.5×10^{-4}	0.969	1.2×10^{-4}	0.967	5.9×10^{-4}	0.997*	5.9×10^{-4}	0.968	3.0×10^{-4}	0.967	8.0×10^{-4}
Srinivas	0.541*	5.1×10^{-5}	0.538	4.7×10^{-4}	0.540	1.0×10^{-4}	0.532	4.8×10^{-5}	0.541	7.1×10^{-5}	0.541	7.5×10^{-5}
Tanaka	0.308	3.0×10^{-4}	0.308	2.8×10^{-4}	0.309	3.4×10^{-4}	1.000*	0	0.306	5.4×10^{-4}	0.304	6.6×10^{-4}

*表示最优值。

表 2　Spread 标准差平均值及方差

Tab. 2　Mean and standard deviation of Spread

测试	MNCS-MOPSO		NSGA-II		SPEA2		MOEA/D-DE		OMOPSO		SMPSO	
函数	均值	方差	均值	方差	均值	方差	均值	方差	均值	方差	均值	方差
ZDT1	0.069*	1.5×10^{-2}	0.372	2.9×10^{-2}	0.151	1.8×10^{-2}	0.375	4.9×10^{-2}	0.082	9.0×10^{-3}	0.073	1.1×10^{-2}
ZDT2	0.068*	1.2×10^{-2}	0.385	3.1×10^{-2}	0.172	4.5×10^{-2}	0.368	1.2×10^{-1}	0.083	9.8×10^{-3}	0.075	1.5×10^{-2}
ZDT3	0.703*	1.9×10^{-3}	0.751	1.6×10^{-2}	0.712	5.9×10^{-3}	0.992	2.9×10^{-2}	0.713	7.9×10^{-3}	0.717	2.1×10^{-2}
ZDT4	0.079*	1.3×10^{-2}	0.396	3.1×10^{-2}	0.293	1.4×10^{-1}	1.000	1.5×10^{-1}	0.912	6.1×10^{-2}	0.088	1.2×10^{-2}
ZDT6	0.264	4.0×10^{-1}	0.346	3.2×10^{-2}	0.227	2.6×10^{-2}	0.156*	1.9×10^{-2}	0.653	5.4×10^{-1}	0.467	5.4×10^{-1}
WFG1	1.130	1.0×10^{-1}	0.715	4.8×10^{-2}	0.688*	6.6×10^{-2}	1.110	1.6×10^{-1}	1.090	6.8×10^{-2}	1.010	4.0×10^{-2}
WFG2	0.762	3.1×10^{-3}	0.784	9.1×10^{-3}	0.760*	6.9×10^{-3}	1.100	3.3×10^{-3}	0.774	9.2×10^{-3}	0.801	1.6×10^{-3}
WFG3	0.361*	5.3×10^{-3}	0.575	2.3×10^{-2}	0.433	7.8×10^{-3}	0.563	4.1×10^{-4}	0.371	5.0×10^{-3}	0.381	5.0×10^{-3}
WFG4	0.353	4.4×10^{-2}	0.393	2.3×10^{-2}	0.275*	1.9×10^{-2}	0.508	4.3×10^{-2}	0.401	4.0×10^{-2}	0.460	4.6×10^{-2}
WFG5	0.159	6.4×10^{-2}	0.403	3.0×10^{-2}	0.286	1.7×10^{-2}	0.463	4.0×10^{-3}	0.136*	1.5×10^{-2}	0.139	1.5×10^{-2}
WFG6	0.114*	1.2×10^{-2}	0.385	2.7×10^{-2}	0.250	1.6×10^{-2}	0.415	7.6×10^{-3}	0.128	9.2×10^{-2}	0.148	1.5×10^{-2}
WFG7	0.117*	5.1×10^{-2}	0.395	3.2×10^{-2}	0.247	1.5×10^{-2}	0.416	7.8×10^{-3}	0.131	1.1×10^{-2}	0.149	1.2×10^{-2}
WFG8	0.727	5.1×10^{-2}	0.646	4.0×10^{-2}	0.614	4.8×10^{-2}	0.635	3.7×10^{-2}	0.575*	6.5×10^{-2}	0.741	5.6×10^{-2}
WFG9	0.184*	1.1×10^{-2}	0.390	2.8×10^{-2}	0.299	1.6×10^{-2}	0.483	8.5×10^{-3}	0.201	1.4×10^{-2}	0.216	1.3×10^{-2}
Golinski	0.244*	3.4×10^{-2}	0.444	3.4×10^{-2}	0.695	3.1×10^{-2}	1.040	3.1×10^{-2}	0.320	3.6×10^{-2}	0.360	4.3×10^{-2}
Srinivas	0.068 9*	1.3×10^{-2}	0.395	3.4×10^{-2}	0.171	1.2×10^{-2}	0.637	8.4×10^{-4}	0.082	1.0×10^{-2}	0.088	1.3×10^{-2}
Tanaka	0.658*	2.2×10^{-2}	0.810	2.8×10^{-2}	0.757	3.5×10^{-2}	1.000	0	0.738	3.9×10^{-2}	0.805	3.0×10^{-2}

* 表示最优值。

OMOPSO 算法在 WFG(5,8) 问题上多样性最佳。对于求解约束问题上,MNCS-MOPSO 相对另外 5 种算法多样性方面都获得了最优值,说明该算法在求解有约束问题时有较好的多样性。MNCS-MOPSO 的多样性较好,表明使用差分进化策略有利于提高算法的多样性;在求解 ZDT 系列函数时,MNCS-MOPSO 和 SMPSO 算法表现分列第 1 和第 2,而两种算法都使用了速度钳制策略,证明了速度钳制策略有利于提高算法的多样性,增强粒子的局部寻优能力。

从表 3 中 Epsilon 性能指标可知,MNCS-MOPSO 共求得 12 个最优值;MOEA/D-DE、NSGA-II、SPEA2、OMOPSO 和 SMPSO 算法各获得 1 个最优值。通过实验数据表明,MNCS-MOPSO 在求解 ZDT 系列函数时收敛性方面占有绝对优势。对于 WFG 系列问题,MNCS-MOPSO 在 WFG(2,3,6,7,9) 问题上收敛性相对于另外 5 种算法有一定的优势;NSGA-II 算法依旧在 WFG1 问题上收敛性最好;SPEA2 在 WFG4 问题上收敛最好;SMPSO 算法在 WFG8 问题的求解中收敛性最佳;对于 WFG5 问题 OMOPSO 算

法收敛性最优,MNCS-MOPSO 次之。对于约束问题,Srinivas 问题上 MNCS-MOPSO 依旧表现最优,MOEA/D-DE 对求解 Golinski 收敛性最好,在 Golinski 问题求解上,MNCS-MOPSO 收敛性排第 2。收敛性的好坏,直接反映算法对问题的求解性能,MNCS-MOPSO 算法有较好的收敛性,使用链式结构对不同位置的粒子采用不同的速度和位置更新策略,对粒子的全局搜索性能有一定程度的提高,另外通过控制粒子的飞行速度,能够更好地平衡粒子的全局搜索和局部寻优,能够较好地避免粒子陷入局部收敛。

图 2 和图 3 为 6 种算法在求解 ZDT3 和 WFG3 时 30 次独立运行实验盒装图,其中算法 1 ~ 6 依次为:MNCS-MOPSO、NSGA-II、SPEA2、MOEA/D-DE、OMOPSO 和 SMPSO 算法。从定性的角度分析图 2 和图 3,MNCS-MOPSO 算法分布性、收敛性和多样性都优于其他算法,而且从盒图中解的分布范围可知,算法的稳定性较好。

表 3 Epsilon 标准差平均值及方差

Tab. 3 Mean and standard deviation of Epsilon

测试	MNCS-MOPSO		NSGA-II		SPEA2		MOEA/D-DE		OMOPSO		SMPSO	
函数	均值	方差	均值	方差	均值	方差	均值	方差	均值	方差	均值	方差
ZDT1	0.005 24*	1.4×10^{-4}	0.013 1	2.4×10^{-3}	0.008 68	6.2×10^{-4}	0.241	7.1×10^{-3}	0.006 29	3.7×10^{-4}	0.005 65	2.2×10^{-4}
ZDT2	0.005 28*	1.7×10^{-4}	0.013 2	1.7×10^{-3}	0.025 4	5.4×10^{-2}	0.050 4	2.0×10^{-2}	0.006 02	1.8×10^{-4}	0.005 51	1.5×10^{-4}
ZDT3	0.005 01*	3.1×10^{-4}	0.008 85	2.2×10^{-3}	0.020 3	5.5×10^{-2}	0.133	3.2×10^{-2}	0.007 87	1.8×10^{-3}	0.009 99	2.1×10^{-2}
ZDT4	0.005 34*	1.7×10^{-4}	0.014 7	2.4×10^{-3}	0.050 4	4.9×10^{-2}	0.510	3.1×10^{-1}	7.03	3.7	0.006 16	3.0×10^{-4}
ZDT6	0.004 67*	3.3×10^{-4}	0.014 7	1.6×10^{-3}	0.024 0	3.8×10^{-3}	0.005 49	2.2×10^{-3}	0.005 21	4.9×10^{-4}	0.004 97	4.1×10^{-4}
WFG1	1.00	1.1×10^{-1}	0.045 0*	2.3×10^{-1}	0.084 6	2.7×10^{-1}	0.618	1.6×10^{-1}	1.19	6.7×10^{-2}	1.14	3.6×10^{-2}
WFG2	0.010 2*	6.2×10^{-4}	0.043 2	3.4×10^{-1}	0.034 1	3.5×10^{-1}	0.029 3	1.5×10^{-3}	0.012 0	1.4×10^{-3}	0.014 9	1.9×10^{-3}
WFG3	2.00*	1.3×10^{-4}	2.00	7.2×10^{-4}	2.00	2.3×10^{-3}	2.00	1.5×10^{-4}	2.00	2.5×10^{-4}	2.00	8.5×10^{-4}
WFG4	0.038 5	4.2×10^{-3}	0.0351	6.1×10^{-3}	0.025 4*	3.1×10^{-3}	0.050 9	8.7×10^{-2}	0.0437	3.1×10^{-3}	0.052 9	4.3×10^{-3}
WFG5	0.063 6	7.1×10^{-4}	0.085 0	8.7×10^{-3}	0.073 6	3.5×10^{-3}	0.072 9	2.1×10^{-4}	0.0635*	4.7×10^{-4}	0.063 9	1.4×10^{-3}
WFG6	0.013 9*	7.1×10^{-4}	0.048 6	1.7×10^{-2}	0.036 1	1.9×10^{-2}	0.024 5	8.0×10^{-4}	0.0153	6.3×10^{-4}	0.016 9	1.1×10^{-3}
WFG7	0.014 4*	3.2×10^{-4}	0.035 7	6.3×10^{-3}	0.025 5	3.2×10^{-2}	0.025 3	4.1×10^{-4}	0.0163	5.8×10^{-4}	0.018 4	1.2×10^{-3}
WFG8	0.415	6.0×10^{-2}	0.431	1.1×10^{-1}	0.043 9	1.0×10^{-1}	0.412	3.7×10^{-2}	0.499	3.0×10^{-2}	0.386*	4.9×10^{-2}
WFG9	0.024 6*	3.2×10^{-3}	0.037 9	6.3×10^{-3}	0.003 00	3.7×10^{-3}	0.034 1	1.2×10^{-3}	0.025 8	2.1×10^{-3}	0.027 9	2.5×10^{-3}
Golinski	8.92	1.1	9.12	1.9	15.8	3.5	2.31*	2.2×10^{-1}	1.21	1.6	14.4	2.5
Srinivas	1.36*	1.7×10^{-1}	3.44	8.9×10^{-1}	1.85	1.2×10^{-1}	3.26	4.8×10^{-2}	1.47	2.2×10^{-1}	1.48	1.5×10^{-1}
Tanaka	0.009 11*	1.8×10^{-3}	0.008 72	1.5×10^{-3}	0.008 27	1.7×10^{-3}	0.044	0	0.013 9	3.2×10^{-3}	0.016 5	3.3×10^{-3}

*表示最优值。

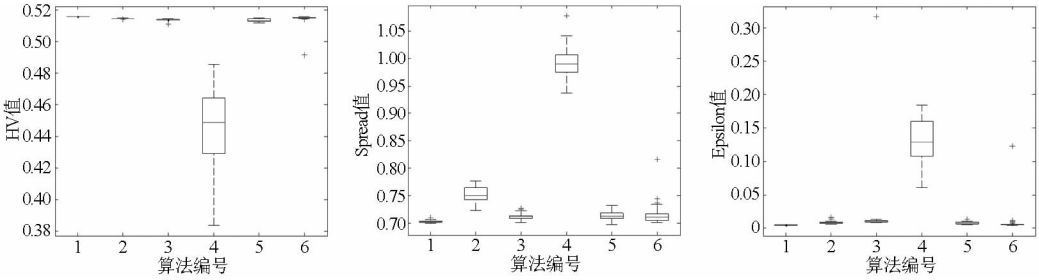

图 2 ZDT3 测试函数的性能指标统计盒图

Fig. 2 Performance index statistic box diagram of ZDT3 function

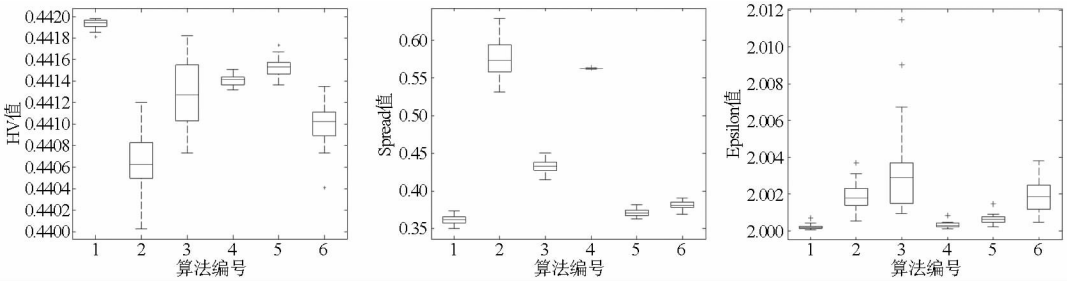

图 3 WFG3 测试函数的性能指标统计盒图

Fig. 3 Performance index statistic box diagram of WFG3 function

为了在统计意义上更加全面地比较分析多个算法的性能，采用 Friedman 检验对结果进行分析，表 4 给出了 6 种算法在不同指标上的排序，图 4 直观地给出了各算法不同指标的分布图，其中对于分布性指标 HV，数值越大说明分布性越好，Epsilon 和 Spread 指标，数值越小说明性能更好。在分布性方面，MNCS-MOPSO 算法的分布性指标最大，说明分布性较另外 5 种算法更好；在收敛性方面，从数值分析，MNCS-MOPSO 算法获得的 Epsilon 值约等于是其他算法的 1/2，说明该算法的收敛性能优异；在多样性方面，MNCS-MOPSO 算法值最小，且远优于另外 5 种算法。

表 4　6 种算法的平均排名

Tab. 4　Average ranking of six algorithms

算法	HV 值	Epsilon 值	Spread 值
MNCS-MOPSO	5. 055 555 555 555 555	1. 833 333 333 333 333 5	1. 944 444 444 444 444 9
NSGA-II	3. 277 777 777 777 778	4. 111 111 111 111 112	4. 333 333 333 333 333
SPEA2	3. 444 444 444 444 444	4. 277 777 777 777 778	3. 0
MOEA/D-DE	3. 055 555 555 555 556	4. 055 555 555 555 555	5. 388 888 888 888 888
OMOPSO	3. 388 888 888 888 888	3. 277 777 777 777 778	2. 833 333 333 333 333 5
SMPSO	2. 777 777 777 777 777 2	3. 444 444 444 444 443 8	3. 499 999 999 999 999 6

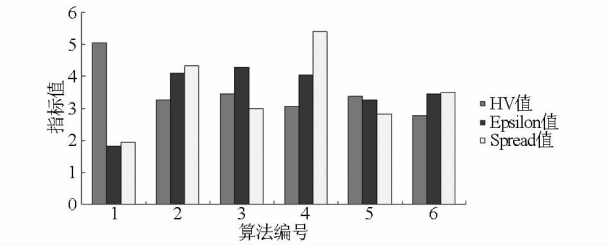

图 4　Friedman 测试对比图

Fig. 4　Friedman number of comparison histogram

综上可知，多邻域链式结构的不同位置粒子分配不同的速度和位置更新策略、速度钳制策略和差分进化策略，有助于提高算法的搜索能力，且能较好地平衡算法的全局搜索和局部寻优性能，通过实验对比分析，说明 MNCS-MOPSO 算法收敛性和多样性在都较好的同时，能够有效保留分布性能良好的非支配解，使其分布更加均匀。

4　桁架结构优化

为了进一步验证 MNCS-MOPSO 的性能，将其应用于含有约束条件的 72 杆桁架结构优化设计问题。以 72 杆桁架结构系统为研究对象，在规定的静力载荷条件下，基本参数（材料密度、弹性模量、最大许用应力、节点坐标等）已知，在满足结构给定的所有约束条件下，使得结构的质量最轻和变形最小。

72 杆桁架为空间桁架结构，平面示意图如图 5 所示，其中 $L = 1\ 524$ mm。72 杆桁架结构包括 2 个子目标，分别为桁架结构质量 W 最小和节点 5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20 的最大位移最小。72 杆桁架属于离散变量优化问题，杆件截面组数为 16，则其自变量的维数为 16，其中约束条件为每个杆件的应力约束，其基本参数有：弹性模量为 6.895×10^{10} N/m^2，杆件密度为 2 768.0 kg/m^3，许用应力为 ±172.37 N/m^2，设计变量为离散设计变量 $645.16 \times \{0.174, 0.220, 0.225, 0.270, 0.287, 0.350, 0.414, 0.431, 0.477, 0.508, 0.587, 0.600, 0.694, 0.667, 0.774, 0.882, 0.908, 1.017, 1.071, 1.277, 1.349, 1.400, 1.457, 1.566, 1.705, 1.783, 1.845, 1.907, 2.046, 2.186, 2.217\}$。载荷参数如表 5 所示，杆件分类如表 6 所示。

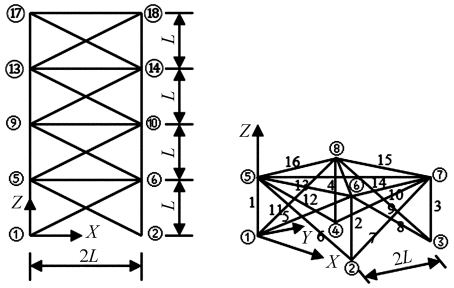

图 5　72 杆空间桁架示意图

Fig. 5　72-bar space truss

表 5　72 杆桁架节点载荷

Tab. 5　Loading conditions of the 72-bar space truss

节点号	F_x/kN	F_y/kN	F_z/kN
17	0	0	−22.24
18	0	0	−22.24
19	0	0	−22.24
20	0	0	−22.24

表 6　72 杆件分组表

Tab. 6　Group table of 72-bar space truss

组号	杆件号	组号	杆件号
X1	1,2,3,4	X9	37,38,39,40
X2	5,6,7,8,9,10,11,12	X10	41,42,43,44,45,46,47,48
X3	13,14,15,16	X11	49,50,51,52
X4	17,18	X12	53,54
X5	19,20,21,22	X13	55,56,57,58
X6	23,24,25,26,27,28,29,30	X14	59,60,61,62,63,64,65,66
X7	31,32,33,34	X15	67,68,69,70
X8	35,36	X16	71,72

图 6 为两种算法获得的 Pareto 前端比较图,可以明显看出 MNCS-MOPSO 获得极端值点更加宽广,分布更加均匀,能够为决策者提供更多的选择方案,说明了算法的工程有效性。

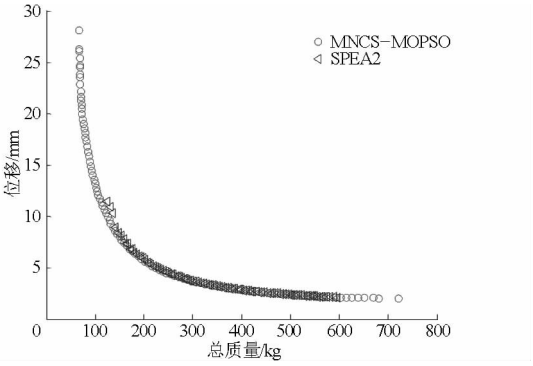

图 6 Pareto 前端比较对比图

Fig. 6 Comparison of Pareto front

5 结束语

针对多目标粒子群算法易陷入局部收敛、全局搜索与局部寻优难以平衡等问题,本文提出一种多邻域链式结构的多目标粒子群优化算法(MNCS-MOPSO)。将粒子种群划分称为多邻域链式拓扑结构,对单链中节点粒子使用 φ 收敛因子,并将粒子历史最优位置替换为邻居历史最优位置;对于单链中间粒子采用线性速度惯性权重因子。为了平衡粒子的全局搜索和局部寻优的能力,引入一种速度钳制策略,实验证明该策略有助于提高算法的收敛性和多样性。在粒子种群加入差分进化策略,增强对粒子的扰动,在一定程度上提高了算法的多样性能。通过 17 个测试函数仿真分析,表明 MNCS-MOPSO 算法较另外 5 种算法,在分布性、收敛性和多样性方面均有一定优势,是求解高维多目标问题的有效方法。为了验证算法在求解约束问题和解决实际工程问题的性能,将其用于 72 杆桁架多目标优化问题中,求解结果表明了本文算法的实用性及可行性。

参 考 文 献

1 Deb K, Pratap A, Agarwal S, et al. A fast and elitist multi-objective genetic algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2):182 – 197.

2 Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm [M] // Giannakoglou K, Tsahalis D T, Périaux J, et al. Evolutionary methods for design, optimization and control with applications to industrial problems. Berlin: Springer-Verlag, 2002:95 – 100.

3 Coello C A, Pulido G T, Lechuga M S. Handing multiple objectives with particle swarm optimization [J]. IEEE Transactions on Evolutionary Computations, 2004, 8(3):256 – 279.

4 Gong M G, Jiao L C, Du H F, et al. Multi-objective immune algorithm with nondominated neighbor-based selection [J]. Evolutionary Computation, 2008, 16(2):225 – 255.

5 Li H, Zhang Q. Multi-objective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2009, 12(2): 284 – 302.

6 Kennedy J, Eberhart R. Particle swarm optimization [C] // Proceedings of the 1995 IEEE International Conference on Neural Networks, 1995,4:1942 – 1948.

7 任子晖, 王坚. 动态拓扑结构的多目标粒子群优化算法[J]. 同济大学学报:自然科学版,2011,39(8):1222 – 1226.
Ren Zihui, Wang Jian. Dynamical topology multi-objective particle swarm optimization algorithm [J]. Journal of Tongji University: Natural Science, 2011, 39(8):1222 – 1226. (in Chinese)

8 聂瑞,章卫国,李广文,等. 一种自适应混合多目标粒子群优化算法[J]. 西北工业大学学报,2011,29(5):695 – 701.
Nie Rui, Zhang Weiguo, Li Guangwen, et al. A more useful AHMOPSO (adaptive hybrid multi-objective particle swarm optimization) algorithm [J]. Journal of North Western Polytechnical University, 2011, 29(5):695 – 701. (in Chinese)

9 贾树晋,杜斌,岳恒. 基于局部搜索与混合多样性策略的多目标粒子群算法[J]. 控制与决策,2012,27(6):813 – 818.
Jia Shujin, Du Bin, Yue Heng. Local search and hybrid diversity strategy based multi-objective particle swarm optimization algorithm [J]. Control and Decision, 2012, 27(6):813 – 818. (in Chinese)

10 朱大林,詹腾,张屹,等. 元胞多目标粒子群优化算法与其应用[J]. 农业机械学报,2013,44(12):280 – 287.
Zhu Dalin, Zhan Teng, Zhang Yi, et al. Algorithm and application of cellular multi-objective particle swarm optimization [J]. Transactions of the Chinese Society for Agricultural Machinery, 2013, 44(12):280 – 287. (in Chinese)

11 冯琳,毛志忠,袁平. 一种改进的多目标粒子群优化算法及其应用[J]. 控制与决策,2012,27(9):1313 – 1319.
Feng Lin, Mao Zhizhong, Yuan Ping. An improved multi-objective particle swarm optimization algorithm and its application [J]. Control and Decision,2012,27(9):1313 – 1319. (in Chinese)

12 Clerc M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization [C] // Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, 1999.

(下转第 358 页)

Yu Faguo, Gao Feng, Shi Qiaoshuo. Type synthesis for forging manipulators based on GF set[J]. Chinese Journal of Mechanical Engineering, 2008, 44(2): 152 - 159. (in Chinese)

11 赵勇,林忠钦,王皓,等. 重型锻造操作机的操作性能分析[J]. 机械工程学报,2010,46(11):69 - 75.
Zhao Yong, Lin Zhongqin, Wang Hao, et al. Manipulation performance analysis of heavy manipulators[J]. Chinese Journal of Mechanical Engineering, 2010, 46(11): 69 - 75. (in Chinese)

12 陈燎原. 电液伺服系统的模糊控制研究[J]. 农业机械学报,2002,33(1):90 - 93.
Chen Liaoyuan. Fuzzy control of electro-hydraulic servo system[J]. Transactions of the Chinese Society for Agricultural Machinery, 2002, 33(1): 90 - 93. (in Chinese)

13 王幼民,刘有余. 电液位置伺服系统的自适应控制[J]. 农业机械学报,2006,37(12):160 - 163.
Wang Youmin, Liu Youyu. Adaptive control research for electro-hydraulic position servo system[J]. Transactions of the Chinese Society for Agricultural Machinery, 2006, 37(12): 160 - 163. (in Chinese)

14 郑江. 锻造操作机电液比例位置控制系统研究[D]. 武汉:华中科技大学,2003.
Zheng Jiang. Research on electro-hydraulic proportional control of forging manipulator[D]. Wuhan: Huazhong University of Science and Technology, 2003. (in Chinese)

15 石红雁,许纯新,付连宇. 基于 SIMULINK 的液压系统动态仿真[J]. 农业机械学报,2000,31(5):94 - 96.
Shi Hongyan, Xu Chunxin, Fu Lianyu. Study on dynamic simulation of hydraulic system based on SIMULINK[J]. Transactions of the Chinese Society for Agricultural Machinery, 2000, 31(5): 94 - 96. (in Chinese)

(上接第 372 页)

13 Eberhart R C, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization[C]// Proceedings of 2000 IEEE Congress on Evolutionary Computation, CEC2000, 2000, 1: 84 - 88.

14 Nebro A J, Durillo J J, García-Nieto J, et al. SMPSO: a new PSO-based metaheuristic for multi-objective optimization[C]// IEEE Symposium on Computational Intelligence in Multi-criteria Decision-making, 2009: 66 - 73.

15 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.
Meng Hongyun, Zhang Xiaohua, Liu Sanyang. A differential evolution based on double populations for constrained multi-objective optimization problem[J]. Chinese Journal of Computers, 2008, 31(2): 228 - 235. (in Chinese)

16 贺毅朝,王熙照,刘坤起,等. 差分演化的收敛性分析与算法改进[J]. 软件学报,2010,21(5):875 - 885.
He Yichao, Wang Xizhao, Liu Kunqi, et al. Convergent analysis and algorithmic improvement of differential evolution[J]. Chinese Journal of Computers, 2010, 21(5): 875 - 885. (in Chinese)

17 Kukkone S, Lampinen J. GDE3: the third evolution step of generalized differential evolution [C]// IEEE Congress on Evolutionary Computation, CEC 2005, 2005: 443 - 450.

18 Li H, Zhang Q. Multi-objective optimization problems with complicated pareto sets, MOEA/D and NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2009, 12(2): 284 - 302.

19 Coello Coello, Carlos A, Lechuga M S. MOPSO: a proposal for multiple objective particle swarm optimization[C]// Proceedings of the 2002 Congress on Evolutionary Computation, 2002: 1051 - 1056.

20 Durillo J J, Nebro A J, Luna F, et al. jMetal: a java framework for developing multi-objective optimization [R]. Technical Report ITI-2006-10, Dpto. De Lenguajes y Ciencias de la Computaci'on, University of M'alaga, 2006.

21 Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach [J]. Evolutionary Computation, 1999, 3(4): 257 - 271.

22 Durillo J J, Nebro A J. jMetal: a java framework for multi-objective optimization [J]. Advances in Engineering Software, 2011, 42(10): 760 - 771.