文章编号: 1001-0920(2019)05-0908-09 DOI: 10.13195/j.kzyjc.2017.1492

求解区间柔性作业车间调度的多目标进化算法

王 春, 王 艳†, 纪志成

(江南大学 物联网工程学院，江苏 无锡 214122)

摘 要: 针对不确定多目标柔性作业车间调度问题,将工序加工时间采用区间数表示,以区间最大完工时间和区间机器总负荷为优化目标,构建多目标区间柔性作业车间调度模型,并设计一种多目标进化优化算法对该模型进行求解. 算法采用混合策略生成初始化种群,并采用贪婪插入法对染色体进行解码,通过基于可能度的占优关系评价个体性能,将区间目标归一化结合拥挤距离反映优化解的分布情况. 实验结果验证了所提出算法的有效性.

关键词: 多目标调度；柔性作业车间；区间加工时间；贪婪插入法；可能度

中图分类号: TP273 **文献标志码:** A

Multi-objective evolutionary algorithm to solve interval flexible job shop scheduling problem

WANG Chun, WANG Yan†, JI Zhi-cheng

(School of Internet of Things，Jiangnan University，Wuxi 214122，China)

Abstract: For the uncertain multi-objective flexible job shop scheduling problem, the processing time is presented by interval number. A multi-objective interval flexible job shop scheduling problem model is established, and an effective multi-objective evolutionary algorithm (MOEA) is proposed to minimize interval makespan and interval total workload. Firstly, a population is initialized by adopting the hybrid strategy, and a greedy insertion method is designed for chromosome decoding. Then, an interval dominance relationship based on the possibility degree is employed to evaluate two individuals. In addition, a crowding measure hybridized with interval normalization is further used to reflect the distribution of optimal solutions. Finally, the experimental results demonstrate the effectiveness of the proposed algorithm.

Keywords: multi-objective scheduling；flexible job shop；interval processing time；greedy insertion method；possibility degree

0 引 言

多目标柔性作业车间调度(Multi-objective flexible job shop scheduling problem, MOFJSSP)不仅是生产调度中的一类经典优化问题,而且是数学上的一类NP-hard组合优化问题[1]. 以往对MOFJSSP的研究大多集中于处理确定性加工信息,但在实际企业的车间生产中,受机器、工人认知水平、环境等实际不确定因素的影响,准确预知如加工时间和交货期等的相关信息会非常困难. 因此,现有的解决MOFJSSP的智能优化方法并不能适用于求解不确定MOFJSSP. 对于处理不确定加工时间的MOFJSSP,学者们通常将工序加工时间表示为随机数和模糊数,即解决多目标随机柔性作业车间调度问题(Multi-objective stochastic flexible job shop scheduling problem, MOSFJSSP)和多目标模糊柔性作业车间调度问题(Multi-objective fuzzy flexible job shop scheduling problem, MOFFJSSP).

针对多目标随机柔性作业车间调度问题,Zuo等[2]设计了一种多目标免疫算法求解加工时间满足均匀分布的调度问题,通过求取不同场景下目标函数值的期望值和标准差来实现鲁棒优化. Homg等[3]通过概率分布函数得到加工时间,并采用进化策略从搜索空间中的多个调度序列中得到最好解. Shen等[4]通过定义2种鲁棒性能指标来建立同时优化完

收稿日期: 2017-11-06；**修回日期:** 2018-01-16.

基金项目: 国家自然科学基金项目(61572238)；江苏省杰出青年基金项目(BK20160001).

责任编委: 王凌.

作者简介: 王春(1988−), 男, 博士生, 从事进化算法及其在生产调度中应用的研究；王艳(1978−), 女, 教授, 博士生导师, 从事智能制造系统能效优化等研究.

†通讯作者. E-mail: wangyan88@jiangnan.edu.cn.

工时间、最大机器负荷和鲁棒性的调度模型,设计了一类改进的基于分解的多目标进化算法. 朱传军等[5]也通过采用不确定参数描述随机工时的波动程度和约束条件允许违背程度,将MOSFJSSP转化为确定性优化问题,并给出一类有效求解转化后问题的混合进化算法. 对于MOFFJSSP, 王冰等[6]研究了梯形软交货期下的模糊车间作业满意调度问题; 刘爱军等[7]考虑工件交货期服从模糊时间窗分布等约束条件,提出了纵横协同的多种群遗传算法; Wang等[8-9]提出了两种多目标遗传算法,通过设计免疫熵维持种群多样性; Zheng等[10]提出了一种群邻域搜索算法MOSNS(multi-objective swarm-based neighborhood search for fuzzy flexible job shop scheduling),该算法采用基于工序字符串和三维矩阵的编码.

利用随机数表示加工时间需要事先知道该随机变量满足的分布,同样对于采用模糊数表示加工时间需要事先知道该模糊数的隶属度函数. 但是,对于实际企业的加工生产,由于缺乏有效的历史数据信息和加工经验,获取随机加工时间的概率分布函数和模糊加工时间的模糊隶属度函数通常是非常困难的,甚至不能获得. 相反,预知加工时间的上下界相对容易. 此外,随机加工时间和模糊加工时间可以分别通过置信水平和截集水平转化为区间加工时间. 因此,将加工时间表示为区间数是切实可行的. 针对单目标区间作业车间调度问题,Lei首先基于区间扩张原理证明了对于最大完工时间等调度正规性能指标,不确定性调度问题可以转化为区间调度问题,随后提出了遗传算法[11]和基于种群的邻域搜索算法[12]分别优化区间最大完工时间,针对带有机器维护的多目标区间作业车间调度问题,又设计了一种同时优化区间最大完工时间和区间最大拖期时间的多目标蜂群算法MOABC(multi-objective artificial bee colony algorithm)[13]. 杨宏安等[14]以提前/拖期惩罚的取值区间为优化目标构建了区间作业车间调度模型,并设计了求解该模型的遗传算法. Han等[15]利用区间中点和半径信息,将区间多目标阻塞批量流水线调度问题转化为传统的多目标优化问题,并设计了混合进化优化方法进行求解.

由当前国内外研究成果可以看出: 1) 求解的区间调度问题主要针对区间作业车间调度和流水线调度,对于区间柔性作业车间调度的相关研究成果极少; 2) 对于单目标优化的研究较多,考虑优化多个目标的研究成果较少; 3) 对于一些将区间调度问题转化为确定性调度问题的研究,在转化过程中往往丢失有价值的信息. 鉴于此,本文以优化区间完工时间和区间机器总负荷为优化目标,建立一种多目标区间柔性作业车间调度MOIFJSSP(multi-objective interval flexible job shop scheduling problem)优化模型,并直接利用区间比较方法,设计一种有效直接求解该优化模型的多目标进化算法MOEA(multi-objective evolutionary algorithm). 最后通过仿真实验验证所提出算法的有效性.

1　问题描述

MOIFJSSP问题描述为: 工件集$\mathcal{J}$里的n个工件$\{J_1, J_2, \cdots, J_n\}$在机器集$\mathcal{M}$里的$m$台机器$\{M_1, M_2, \cdots, M_m\}$上进行加工,每个工件$J_i$包含一道或多道工序$O_{ij}(j \in \{1, 2, \cdots, n_i\})$,$n_i$为$J_i$包含的工序总数,所有工序按照给定的工艺路线进行加工,工序O_{ij}在相容机器集中的一台机器上进行加工. 工件J_i的第j道工序在机器M_k上的加工时间为区间数$p_{ijk} = (\underline{p_{ijk}}, \overline{p_{ijk}})$,其中$\underline{p_{ijk}}$和$\overline{p_{ijk}}$为区间的下限和上限. 故调度问题可以表述为: 为每道工序确定一台合适的机器,并对每台机器分配的所有工序进行排列以确定其区间开始加工时间,使所设定的优化目标达到最优. 表1为3个工件在3台机器上加工的区间柔性作业车间调度问题的加工时间表,“∞”表示相关工序不能在对应的机器上加工.

表1　区间柔性作业车间调度问题的加工时间表

工件	工序	可选机器和加工时间		
		M_1	M_2	M_3
J_1	O_{11}	[1,3]	[2,4]	[1,5]
	O_{12}	[1,2]	[2,3]	∞
	O_{13}	[2,8]	[10,18]	[2,4]
J_2	O_{21}	∞	[3,9]	[1,2]
	O_{22}	[2,3]	[3,4]	[4,9]
	O_{23}	[1,2]	∞	[4,6]
J_3	O_{31}	[7,11]	[2,5]	[5,8]
	O_{32}	[3,7]	[2,5]	[7,3]
	O_{33}	[3,6]	[1,4]	∞

工件工序在加工过程中需要满足如下约束:

1) 所有机器均相互独立且所有机器在零时刻均可用;

2) 同一台机器在某一时刻只能加工一道工序,工序在加工过程中不能中断;

3) 同一工件的同一道工序在同一时刻只能被一台机器加工;

4) 同一工件的工序之间有先后约束,不同工件的工序之间没有先后约束;

5) 所有工件的优先级相同.

本文求解MOIFJSSP考虑两个优化目标,即最小化区间最大完工时间和区间机器总负荷. 假设C_i为工件J_i的区间完工时间,则两个优化目标分别定义为

$$f_1: C_{\max}=\max\{C_i|i=1,2,\cdots,n\}; \tag{1}$$

$$f_2: W_T=\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n_i}p_{ijk}u_{ijk}. \tag{2}$$

其中u_{ijk}为二进制变量,如果工序O_{ij}在机器k上加工,则$u_{ijk}=1$,否则$u_{ijk}=0$.

2 相关知识介绍

2.1 区间数操作

为了求解合理的调度方案,需要定义区间数的一些操作以描述问题的加工条件和约束. 首先采用Sengupta等[16]定义的区间数求和操作和区间数取大操作. 区间数求和操作用来求解每一道工序的区间完工时间,区间数取大操作用于确定每一道工序的区间开始加工时间. 同时定义一种区间数比较操作,该操作主要用于比较各工件的区间完成时间从而确定整个调度的区间makespan,其次用来判断工序能否进行插入空隙操作. 对于两个区间数$u=[\underline{u},\overline{u}]$和$[\underline{v},\overline{v}]$,3种区间数操作描述如下:

1) 区间求和操作: $u+v=[\underline{u}+\underline{v},\overline{u}+\overline{v}]$.

2) 区间取大操作: $u\bigvee v=[\max(\underline{u},\underline{v}),\max(\overline{u},\overline{v})]$.

3) 假设$m(u),w(u),m(v),w(v)$分别代表区间u和v的中点和半径,根据如下操作比较两个区间数:

规则1 如果$m(u)=0.5(\underline{u}+\overline{u})<(>)0.5(\underline{v}+\overline{v})$,则$u<(>)v$.

规则2 如果$m(u)=m(v)$,则比较$w(u)=0.5(\overline{u}-\underline{u})$和$w(v)=0.5(\overline{v}-\underline{v})$,如果$w(u)<(>)w(v)$,则$u<(>)v$.

本文所解决的是实际工程最小化优化问题. 对于两个区间,当其对应的中点不相同时,区间中点小的区间其值相对较小. 但是,对于如图1所示的区间值相同的情况,由于决策者更偏好区间上下限差异较小的适应度值,即区间目标值的不确定性较小,决策者更倾向于接受a.

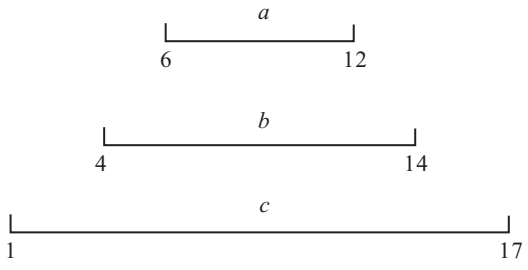

图1 中点相同且具有包含关系的2个区间的位置

2.2 基于可能度的占优关系

对于MOIFJSSP问题,优化的2个目标函数值都是区间数,传统的基于精确目标值的Pareto占优关系已不再适用. 因此,定义区间意义下的两个进化个体的占优关系具有十分重要的意义. 考虑进化个体$\boldsymbol{x}_i$和$\boldsymbol{x}_j$,求解第l个目标所对应的区间函数值分别为

$$f_l(\boldsymbol{x}_i)=[\underline{f_l(\boldsymbol{x}_i)},\overline{f_l(\boldsymbol{x}_i)}],$$

$$f_l(\boldsymbol{x}_j)=[\underline{f_l(\boldsymbol{x}_j)},\overline{f_l(\boldsymbol{x}_j)}].$$

$L(f_l(\boldsymbol{x}_i))=\overline{f_l(\boldsymbol{x}_i)}-\underline{f_l(\boldsymbol{x}_i)}$和$L(f_l(\boldsymbol{x}_j))=\overline{f_l(\boldsymbol{x}_j)}-\underline{f_l(\boldsymbol{x}_j)}$为两个区间对应的区间宽度. 达庆利等[17]给出了$f_l(\boldsymbol{x}_i)\leqslant f_l(\boldsymbol{x}_j)$的可能度$P_l(\boldsymbol{x}_i\leqslant\boldsymbol{x}_j)$,表示为

$$P_l(\boldsymbol{x}_i\leqslant\boldsymbol{x}_j)=\frac{\max\{0,(L(f_l(\boldsymbol{x}_i))+L(f_l(\boldsymbol{x}_j))-L(f_l(\boldsymbol{x}_i))+\max(\overline{f_l(\boldsymbol{x}_i)}-\underline{f_l(\boldsymbol{x}_j)},0))\}}{L(f_l(\boldsymbol{x}_j))}. \tag{3}$$

同理,记$f_l(\boldsymbol{x}_j)\leqslant f_l(\boldsymbol{x}_i)$的可能度为$P_l(x_j\leqslant\boldsymbol{x}_i)$,有

$$P_l(\boldsymbol{x}_j\leqslant\boldsymbol{x}_i)=\frac{\max\{0,(L(f_l(\boldsymbol{x}_i))+L(f_l(\boldsymbol{x}_j))-L(f_l(\boldsymbol{x}_i))+\max(\overline{f_l(\boldsymbol{x}_j)}-\underline{f_l(\boldsymbol{x}_i)},0))\}}{L(f_l(\boldsymbol{x}_j))}. \tag{4}$$

区间占优可能度具有如下性质:

1) $P_l(\boldsymbol{x}_i\leqslant\boldsymbol{x}_j)+P_l(\boldsymbol{x}_j\leqslant\boldsymbol{x}_i)=1$.

2) 如果$P_l(\boldsymbol{x}_i\leqslant\boldsymbol{x}_j)>0.5$,则$m(f_l(\boldsymbol{x}_i))<m(f_l(\boldsymbol{x}_j))$;如果$m(f_l(\boldsymbol{x}_i))=m(f_l(\boldsymbol{x}_j))$,则$P_l(\boldsymbol{x}_i\leqslant\boldsymbol{x}_j)=0.5$.

将单目标下区间数比较的可能度扩展到多个目标来定义一种个体区间意义下的Pareto占优关系. 对于$\forall l\in\{1,2,\cdots,M\}$,均有$P_l(\boldsymbol{x}_i\leqslant\boldsymbol{x}_j)\geqslant P_l(\boldsymbol{x}_j\leqslant\boldsymbol{x}_i)$,且至少$\exists k\in\{1,2,\cdots,M\}$使得$P_k(\boldsymbol{x}_i\leqslant\boldsymbol{x}_j)>P_k(\boldsymbol{x}_j\leqslant\boldsymbol{x}_i)$,则称$\boldsymbol{x}_i$占优$\boldsymbol{x}_j$,即$\boldsymbol{x}_i\prec\boldsymbol{x}_j$. 如果$\boldsymbol{x}_i$不占优$\boldsymbol{x}_j$,$\boldsymbol{x}_j$也不占优$\boldsymbol{x}_i$,则$\boldsymbol{x}_i$和$\boldsymbol{x}_j$互不占优,即$\boldsymbol{x}_i\parallel\boldsymbol{x}_j$.

3 MOEA算法描述

3.1 MOEA算法步骤

本文所提MOEA算法采用基于NSGA-II[18]算法的基本框架,具体步骤描述如下.

Step 1: 按混合机器分配规则和工序排序规则方法产生规模为N的初始化种群P_0,令进化代数$t=0$.

Step 2: 运用贪婪插入式法解码评价P_0中每一个个体的适应度值.

Step 3: 如果算法达到终止条件, 则转至Step 8; 否则转至Step 4.

Step 4: 对父代种群$P(t)$执行进化操作, 产生子代种群$Q(t)$.

Step 5: 合并种群$P(t), Q(t)$得到种群$R(t)$, 对$R(t)$中目标函数值相同的冗余个体进行变异操作, 得到种群$R'(t)$.

Step 6: 对$R'(t)$中的所有进化个体进行目标归一化, 得到种群$R^*(t)$.

Step 7: 对种群$R^*(t)$中的所有个体进行快速非支配排序, 产生下一代种群$P(t+1)$, 令$t = t+1$, 返回Step 3.

Step 8: 输出Pareto最优解集.

3.2　染色体编码和解码

考虑MOIFJSSP问题的特性, 所提出算法染色体编码由两部分组成, 即机器选择部分(MS)和工序排序部分(OS), 两部分的染色体长度均等于总工序数之和. MS中的每一位数字依次代表相应工序对应的加工机器. OS中的每一个数字代表加工工件号, 其在编码中出现的次数代表工件的第几道工序. 对于表1中的一个问题实例, 图2为一个染色体编码示意图. 可以看出, 工序O_{12}在机器M_1上加工, 工序O_{23}在机器M_3上加工, 工序的加工顺序为$O_{21} \rightarrow O_{31} \rightarrow O_{11} \rightarrow O_{12} \rightarrow O_{32} \rightarrow O_{33} \rightarrow O_{22} \rightarrow O_{23} \rightarrow O_{13}$.

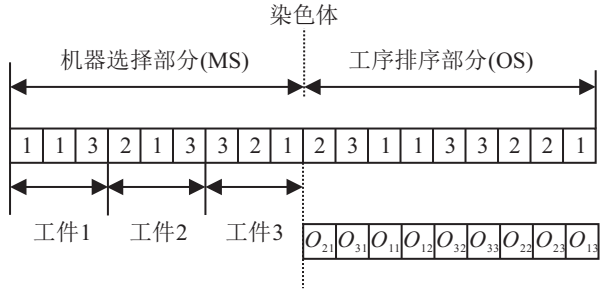

图 2　染色体编码

本文将区间数操作与主动解码方法[19]相结合, 设计一种贪婪插入空隙法对染色体进行解码, 解码过程具体描述如下.

Step 1: 依次从染色体的OS读取基因和对应的加工工序O_{ij}.

Step 2: 从MS读取工序O_{ij}对应的加工机器k, 令$p_{ijk} = [\underline{p_{ijk}}, \overline{p_{ijk}}]$为区间加工时间, 在机器$k$上依次查找空闲时间段$[t_k^S, t_k^E]$, 其中, $t_k^S = [\underline{t_k^S}, \overline{t_k^S}]$和$t_k^E = [\underline{t_k^E}, \overline{t_k^E}]$分别为空闲时间段的区间开始加工时间和区间结束加工时间. 令C_{ij-1}表示工序O_{ij-1}的区间完工时间, 则在满足工件加工约束的前提下, 工序O_{ij}的区间开始加工时间为

$$S_{ij} = \begin{cases} \max\{t_k^S, C_{ij-1}\}, \ j \geqslant 2; \\ t_k^S, \ j = 1. \end{cases} \tag{5}$$

Step 3: 按照下式判断空闲时间段是否满足插入条件:

$$\begin{cases} \max\{t_k^S, C_{ij-1}\} + p_{ijk} \leqslant t_k^E, \ j \geqslant 2; \\ t_k^S + p_{ijk} \leqslant t_k^E, \ j = 1. \end{cases} \tag{6}$$

如满足则插入空闲时间段, 否则将O_{ij}插入机器k已加工的最后一道工序之后, 其区间开始加工时间为$\max\{C_{ij-1}, \mathrm{FM}_k\}$. FM_k为机器k已加工的最后一道工序的区间完工时间.

本文提出的解码方法将工艺路线允许的最早开工时间与加工机器允许的最早开始时间进行取大操作作为工序的实际开工区间, 符合柔性作业车间的工艺和机器约束环境. 同时, 对于可能出现的几个工件区间完工时间中点相同的情况, 也可以通过第2.1节的区间数比较操作进行区分.

3.3　种群初始化

为了获取一定质量和分散度的初始化种群, 采用混合策略生成初始化种群, 即将区间数的相关操作融入不同的机器选择规则和工序排序规则, 生成每一个初始个体的MS和OS.

对于MS, 采用3种不同的机器选择规则: 全局选择[20]、局部选择[20]和随机选择. 全局选择和局部选择是为了机器负荷的均匀性, 随机选择的目的是为了维持种群初始解的多样性. 3种机器规则的使用概率为0.6、0.3和0.1.

在MS完成的基础上, 采用4种工序排序规则对OS进行初始化, 分别为剩余负荷最大规则[21]、剩余工序最多规则[21]、随机排序和短加工时间规则. 剩余负荷最大和剩余工序最多规则是将工件分别按照剩余区间加工时间和剩余工序数进行降序, 选择剩余区间加工时间和剩余工序数越多的工件优先加工. 短加工时间规则是优先考虑分配在机器上的加工时间较短的工序, 随机规则是将各个工序的加工顺序随机排序. 4种工序排序规则的选择概率为0.3、0.2、0.3和0.2.

3.4　进化算子

MOEA算法的进化算子包括选择、交叉和变异算子. 对于选择操作采用规模为2的随机联赛选择, 2个进化个体如果序值不同, 则选择序值较小的个体; 如果序值相同, 则选择拥挤距离值较大的个体. 考虑MOIFJSSP问题的特性, 针对染色体的MS和OS部分, 分别设计有效的交叉和变异操作以实现个体进

化. 对于MS和OS的交叉操作,分别采用文献[19]提出的均匀交叉和POX交叉. 对于MS部分的变异操作,随机选择两道工序,分别从这两道工序的可加工机器集合中随机选择一台不同的机器替换当前的加工机器;对于OS部分的变异操作,首先随机选择两道工序,然后将此两道工序进行交换.

3.5 拥挤测度

在求解多目标优化问题中,算法优化得到的Pareto解集不仅希望具有好的收敛性而且希望能够均匀分布于Pareto前端. NSGA-II算法采用拥挤距离来维持Pareto前沿解的分布性. 对于MOIFJSSP问题,由于每个目标函数值为区间,Pareto前沿中的每一个解不是一个点而是一个超体,传统的拥挤距离已不再适用. 这里,采用区间意义下的拥挤距离评价具有相同序值下的进化个体[22]. 考虑到不同目标函数值量纲上的差异,在快速非支配排序前对种群中的所有个体进行目标归一化,目标$f_l(\boldsymbol{x})(l\in\{1,2,\cdots,M\})$被替代为$f_l^*(\boldsymbol{x})=[\underline{f_l^*(\boldsymbol{x})},\overline{f_l^*(\boldsymbol{x})}]$,$\underline{f_l^*(\boldsymbol{x})}$和$\overline{f_l^*(\boldsymbol{x})}$满足

$$\underline{f_l^*(\boldsymbol{x})}=\frac{\underline{f_l(\boldsymbol{x})}-\underline{f_l^{\min}(\boldsymbol{x})}}{\overline{f_l^{\max}(\boldsymbol{x})}-\underline{f_l^{\min}(\boldsymbol{x})}},\tag{7}$$

$$\overline{f_l^*(\boldsymbol{x})}=\frac{\overline{f_l(\boldsymbol{x})}-\underline{f_l^{\min}(\boldsymbol{x})}}{\overline{f_l^{\max}(\boldsymbol{x})}-\underline{f_l^{\min}(\boldsymbol{x})}}.\tag{8}$$

其中

$$\underline{f_l^{\min}(\boldsymbol{x})}=\min_{1\leqslant i\leqslant|P|}\underline{f_l(\boldsymbol{x}_i)},$$

$$\overline{f_l^{\max}(\boldsymbol{x})}=\max_{1\leqslant i\leqslant|P|}\overline{f_l(\boldsymbol{x}_i)}.$$

记$f_l^*(\boldsymbol{x}_i)$和$f_l^*(\boldsymbol{x}_j)$的中点为$\mathrm{mid}(f_l^*(\boldsymbol{x}_i))$和$\mathrm{mid}(f_l^*(\boldsymbol{x}_j))$,$\boldsymbol{x}_i$和$\boldsymbol{x}_j$的目标函数超体积为$V(\boldsymbol{x}_i)$和$V(\boldsymbol{x}_j)$,那么$\boldsymbol{x}_i$和$\boldsymbol{x}_j$的距离$D(\boldsymbol{x}_i,\boldsymbol{x}_j)$可以表示为

$$D(\boldsymbol{x}_i,\boldsymbol{x}_j)=\frac{\sum\limits_{l=1}^{M}\left|\mathrm{mid}(f_l^*(\boldsymbol{x}_i))-\mathrm{mid}(f_l^*(\boldsymbol{x}_j))\right|}{\gamma(\boldsymbol{x}_i,\boldsymbol{x}_j)+V(\boldsymbol{x}_i)+V(\boldsymbol{x}_j)+1},\tag{9}$$

其中$\gamma(\boldsymbol{x}_i,\boldsymbol{x}_j)$为$\boldsymbol{x}_i$和$\boldsymbol{x}_j$对应超体的重叠度. 对于区间$f_l^*(\boldsymbol{x}_i)$和$f_l^*(\boldsymbol{x}_j)$,其交区间表示为$f_l^*(\boldsymbol{x}_i)\bigcap f_l^*(\boldsymbol{x}_j)$,该区间宽度为$w\big(f_l^*(\boldsymbol{x}_i)\bigcap f_l^*(\boldsymbol{x}_j)\big)$,那么$\gamma(\boldsymbol{x}_i,\boldsymbol{x}_j)$的表达式为

$$\gamma(\boldsymbol{x}_i,\boldsymbol{x}_j)=\prod_{l=1}^{M}w\big(f_l^*(\boldsymbol{x}_i)\bigcap f_l^*(\boldsymbol{x}_j)\big).\tag{10}$$

对于个体$\boldsymbol{x}_i$,按照式(9)计算出与其超体距离最近的个体$\boldsymbol{x}_j$和$\boldsymbol{x}_k$,个体$\boldsymbol{x}_i$的拥挤距离$\mathrm{cd}(\boldsymbol{x}_i)$可以表示为

$$\mathrm{cd}(\boldsymbol{x}_i)=\frac{D(\boldsymbol{x}_i,\boldsymbol{x}_j)+D(\boldsymbol{x}_i,\boldsymbol{x}_k)}{2}.\tag{11}$$

cd值较大的个体将优先考虑进入下一代. 这里,考虑到Pareto前沿希望能具有好的延展性,对于处于边界的个体将其cd值设为无穷大,从而使其能够进入下一代进化种群.

本文所提出的MOEA算法在NSGA-II算法的基础上作了以下改进以使其适用于求解MOIFJSSP问题,总结如下:

1) 将机器分配与工序排序规则结合区间数操作来产生初始化种群;

2) 采用区间意义下的Pareto占优关系评价个体的好坏,并用于快速非支配排序中的分层操作;

3) 设计有效的进化算子产生下一代进化种群个体;

4) 对种群中重复个体进行变异操作维持种群多样性;

5) 将区间目标归一化技术结合超体距离设计拥挤算子,用来进一步区分非支配排序中具有相同序值的个体以维持Pareto前沿中超体的分布性.

4 仿真分析

为了验证MOEA在求解MOIFJSSP上的有效性,进行大量数值实验,实验平台为: Windows 7, Intel(R) Core(TM) i5-6500CPU 3.20 GHz处理器8 GB内存,编程语言为Matlab2011b.

4.1 实验设置

由于当前并没有关于MOIFJSSP的测试算例,本文构造3组不同的测试集考察MOEA算法的性能. 前两组测试集分别为LeiData测试函数(LeiData1 ~ LeiData5)[23-24]和WangData测试函数(WangData1 ~ WangData4)[9]. 这2组测试集原先用来测试MOFFJSSP,将每一个测例中工序模糊加工时间中的最小加工时间和最大加工时间分别作为区间加工时间的下限和上限. 第3组测试问题为BRData测试集(MK01 ~ MK10)[25],将加工时间p_{ijk}转化为$p_{ijk}^*=[\mathrm{round}(\delta_{ijk}),\mathrm{round}(p_{ijk}+\zeta_{ijk})]$,round(·)为四舍五入取整函数. $\delta_{ijk}\in[0.85p_{ijk},0.94p_{ijk}]$,$\zeta_{ijk}\in[0.1p_{ijk},0.19p_{ijk}]$,如果$\zeta_{ijk}<1$,则$\zeta_{ijk}\in(1,2)$.

MOEA算法相关参数设置为:种群规模100,交叉概率1.0,变异概率0.1,最大函数评价次数15 000. 为了克服随机误差,将所有涉及的对比算法均独立运行15次,取平均结果. 采用S测度[26]、I测度[26]和C测度[27]定量分析不同算法的性能好坏. 其中,S测度和I测度是专门用来评价区间多目标进化算法的性能指标,S测度表示在目标解空间中Pareto前沿超体覆盖的支配解的区域大小,I测度表示Pareto前沿超体的体积. 由于S测度是一个区间,为方便比较,采用区

间中点值代替区间值. 对于3个测度, S 测度综合评价了算法所得Pareto前沿的收敛性和分布性, C 测度主要反映算法收敛性, I 测度体现算法所得Pareto前沿超体的不确定性. S 测度和 C 测度均是越大越好, I 测度越小越好. 评价算法性能好坏时, 优先考虑 S 测度, 其次考虑 C 测度, 最后考虑 I 测度. 为了方便计算, 将所有对比算法取得的解进行目标归一化处理, 因此在求取 S 测度时, 参考点取 $(1.1, 1.1)^{\mathrm{T}}$.

4.2 种群初始化性能分析

为了验证混合初始化方法的有效性, 将其与(Multi-objective evolutionary algorithm with no rules, MOEA-NR) 进行对比, MOEA-NR 与 MOEA 的区别是种群初始化不采用混合机器分配和工序排序规则的方法, 而是随机初始化产生. 两种算法所得 S、C 和 I 测度的均值如表2所示, 较优的均值用粗体表示. 表2同时给出了Wilcoxon符号秩和检验结果, 显著水平为0.05, 其中“†”代表MOEA显著优于对比算法, “‡”代表MOEA显著劣于对比算法.

由表2可见: 对于 S 测度和 C 测度, MOEA 分别在13个和17个测试函数上显著优于MOEA-NR, 而MOEA-NR只在WangData3上得到了较好的 S 测度; 对于 I 测度, 虽然MOEA有5个测试函数优于MOEA-NR, 但MOEA-NR也在4个测试函数上得到了更好的结果. 上述结果表明, 结合规则的初始化方法能够使算法所得Pareto前沿具有更好的收敛性和分布性, 但只能使算法在部分测试问题上更好地控制不确定度.

表 2　MOEA和MOEA-NR在所有测试问题实例上所得 S 测度和 I 测度的均值

实例	$n \times m$	S 测度		MOEA(A) vs MOEA-NR(B)		I 测度	
		MOEA	MOEA-NR	$C(A,B)$	$C(B,A)$	MOEA	MOEA-NR
LeiData1	10 × 10	**0.689 7**	0.683 6	**0.7081**	0.00 14†	0.388 2	**0.358 1‡**
LeiData2	10 × 10	**0.708 0**	0.689 4†	**0.908 1**	0.000 0†	0.302 6	**0.280 1‡**
LeiData3	10 × 10	**0.684 3**	0.659 2†	**0.9136**	0.000 0†	**0.41 57**	0.429 3
LeiData4	10 × 10	**0.690 3**	0.671 3†	**0.833 4**	0.000 0†	**0.360 8**	0.379 5†
LeiData5	15 × 10	**0.716 8**	0.639 8†	**1.000 0**	0.000 0†	**0.341 5**	0.359 5†
WangData1	4 × 6	0.647 5	0.647 5	0.000 0	0.000 0	0.564 2	0.564 2
WangData2	6 × 10	**0.636 1**	0.636 0	**0.295 2**	0.000 0†	**0.598 2**	0.598 3
WangData3	8 × 8	0.674 7	**0.686 0‡**	**0.094 4**	0.000 0	**0.366 9**	0.37 01
WangData4	10 × 10	**0.697 2**	0.684 8†	**0.827 1**	0.010 7†	0.395 2	**0.384 3**
MK01	10 × 6	**0.635 7**	0.619 9†	**0.359 6**	0.000 0†	**0.441 6**	0.444 3
MK02	10 × 6	**0.655 4**	0.648 5	**0.799 6**	0.000 0†	0.379 3	**0.369 5**
MK03	15 × 8	**0.823 4**	0.798 7†	**0.903 5**	0.000 0†	0.159 9	**0.146 5‡**
MK04	15 × 8	**0.728 6**	0.724 4	**0.421 9**	0.001 5†	0.215 5	**0.21 19**
MK05	15 × 4	**0.655 0**	0.647 2†	**0.279 6**	0.002 2†	**0.475 1**	0.483 5†
MK06	10 × 15	**0.750 2**	0.680 4†	**1.000 0**	0.000 0†	**0.279 4**	0.294 9†
MK07	20 × 5	**0.711 2**	0.697 7†	**0.707 3**	0.001 7†	0.284 3	**0.271 9‡**
MK08	20 × 10	**0.649 8**	0.649 1†	**0.163 2**	0.000 0†	0.559 5	**0.557 6**
MK09	20 × 10	**0.758 9**	0.6936 †	**0.964 9**	0.000 0†	0.225 7	**0.225 6**
MK10	20 × 15	**0.797 4**	0.684 8†	**1.000 0**	0.000 0†	**0.169 1**	0.178 7†

4.3 与其他多目标优化算法的性能对比

因为本文研究问题的特殊性, 缺少能够直接与所提出算法进行性能比较的相关文献, 这里将求解MOFJSSP问题的NSGA-II[28]、求解带机器维护区间作业车间调度问题的MOABC[13]以及用于求解MOFFJSSP问题的MOSNS[10]作部分修改, 来与所提出算法进行对比. 设计思路是: 将MOABC中提出的区间意义下个体占优关系和区间意义下拥挤距离[22]用于NSGA-II, 使其能够求解MOIFJSSP. 同理, 也将MOABC中基于区间数的操作代替MOSNS中的模糊数操作. 对于MOABC, 将其结合本文所提出MOEA中对于机器选择部分的操作.

为了公平比较不同方法的优化结果, 4种算法均设置相同的种群规模, 都采用本文提出的插入解码方法, 算法的终止条件都是达到第4.1节设置的最大函数评价次数. 3种对比算法自身的参数设置为: NSGA-II中的交叉和变异概率与MOEA相同; MOABC中替代策略发生代数间隔为4, 构造邻域时基因串交换的对数为1; MOSNS中基因串进行邻域搜索的概率为0.7, 三维阵进行邻域搜索的概率为0.6, 外部储备集的规模为100.

表3给出了MOEA算法与其他3种算法在S测度和I测度上的比较情况. 由表3可见, 对于S测度, MOEA比其他算法具有明显的优越性. 除了WangData3和MK08, 在其他测试问题上都显著占优. 对于I测度, 虽然MOEA在7个测试问题上优势明显, 但是在其余8个问题上MOEA表现并不理想, 尤其在MK04～MK08问题上, 表现性能较差. 对于表4中算法之间的C测度比较, 其结果与S测度相似, MOEA在几乎所有实例上都表现出了最好的性能. 表3和表4中符号意义与表2相同.

表3 MOEA和其他对比算法在所有测试问题实例上所得S测度和I测度的均值

实例	S测度				I测度			
	MOEA	NSGA-II	MOABC	MOSNS	MOEA	NSGA-II	MOABC	MOSNS
LeiData1	**0.6897**	0.6319†	0.5915†	0.6168†	0.3882	0.3704	**0.3693**	0.3721
LeiData2	**0.7080**	0.6438†	0.6176†	0.6284†	0.3026	0.2851	0.2942	**0.2845**
LeiData3	**0.6843**	0.6234†	0.5662†	0.5763†	**0.4157**	0.4293	0.4482†	0.4561†
LeiData4	**0.6903**	0.6288†	0.5776†	0.5813†	**0.3608**	0.3837†	0.4089†	0.4202†
LeiData5	**0.7168**	0.5991†	0.5127†	0.5279†	**0.3415**	0.3667†	0.4143†	0.4073†
WangData1	**0.6475**	0.6380†	0.6245†	0.6243†	0.5642	0.5480‡	**0.5315‡**	0.5565
WangData2	**0.6361**	0.6245†	0.5448†	0.6150†	0.5982	0.5821‡	**0.5394‡**	0.5728‡
WangData3	**0.6747**	0.6607	0.6420†	0.6623†	**0.3669**	0.3700	0.3821	0.3952†
WangData4	**0.6972**	0.6287†	0.6228†	0.6275†	**0.3952**	0.4125	0.4420†	0.4379†
MK01	**0.6357**	0.5980†	0.5513†	0.5787†	0.4416	**0.4385**	0.4597†	0.4613†
MK02	**0.6554**	0.6199†	0.5481†	0.5820†	0.3793	0.3809	**0.3747**	0.3830
MK03	**0.8234**	0.7659†	0.5610†	0.5446†	0.1599	**0.1480‡**	0.1487‡	0.1567
MK04	**0.7286**	0.6997†	0.6805†	0.6525†	0.2155	0.2035‡	0.1844‡	**0.1660‡**
MK05	**0.6550**	0.6299†	0.6338†	0.6053†	0.4751	0.4833†	0.4588‡	**0.4369‡**
MK06	**0.7502**	0.6817†	0.4818†	0.3950†	0.2794	0.3178†	0.2626‡	**0.2565‡**
MK07	**0.7112**	0.6594†	0.5556†	0.5716†	0.2843	0.2768	0.2546‡	**0.2501‡**
MK08	0.6498	**0.6506**	0.6295†	0.6023†	0.5595	0.5556‡	0.5486‡	**0.5460‡**
MK09	**0.7589**	0.6746†	0.5132†	0.4404†	**0.2257**	0.2350†	0.2530†	0.2638†
MK10	**0.7974**	0.7171†	0.4602†	0.4121†	**0.1691**	0.1774†	0.2043†	0.2089†

表4 MOEA和其他对比算法在所有测试问题实例上所得C测度的均值

实例	MOEA(A) vs NSGA-II(D)		MOEA(A) vs MOABC(E)		MOEA(A) vs MOSNS(F)	
	$C(A,D)$	$C(A,D)$	$C(A,E)$	$C(E,A)$	$C(A,F)$	$C(F,A)$
LeiData1	**0.9905**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†
LeiData2	**0.9676**	0.0000†	**0.9639**	0.0000†	**0.9911**	0.0000†
LeiData3	**0.9478**	0.0000†	**1.0000**	0.0000†	**0.9758**	0.0000†
LeiData4	**0.9556**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†
LeiData5	**1.0000**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†
WangData1	0.0000	0.0000	**0.0667**	0.0000	**0.7683**	0.0000†
WangData2	**0.5167**	0.0000	**0.3028**	0.0000†	**0.7094**	0.0000†
WangData3	**0.6889**	0.0000†	**1.0000**	0.0000†	**0.9833**	0.0000†
WangData4	**1.0000**	0.0000†	**0.9667**	0.0000†	**0.9846**	0.0000†
MK01	**0.6263**	0.0000†	**0.9611**	0.0000†	**0.9233**	0.0000†
MK02	**0.9444**	0.0029†	**1.0000**	0.0000†	**0.9714**	0.0000†
MK03	**1.0000**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†
MK04	**0.8541**	0.0014†	**0.9732**	0.0000†	**0.9706**	0.0000†
MK05	**0.6994**	0.0000†	**0.9689**	0.0000†	**0.9805**	0.0000†
MK06	**0.9852**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†
MK07	**0.9317**	0.0000†	**0.9917**	0.0000†	**1.0000**	0.0000†
MK08	**0.6174**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†
MK09	**0.9627**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†
MK10	**1.0000**	0.0000†	**1.0000**	0.0000†	**1.0000**	0.0000†

为了量化算法的整体性能,引入表现分指标[29]对算法进行排名. 对于任意算法,表现分显示了在所考虑的实例上有多少算法显著优于该算法. 因此表现分越低,算法性能越好. 图3为MOEA和其他对比算法在所有问题上的平均表现分及排名(排名见图3中分数后面对应括号里的数值),可以看出MOEA的表现分最低,排名第1. 图4为MOEA和NSGA-II、MOABC分别求解MK05单次运行所获得的Pareto前沿超体中点,所选取的运行为S测度最接近平均值的一次. 可以看出MOEA比其他两种算法获得Pareto前沿解的个数多,分布更均匀.

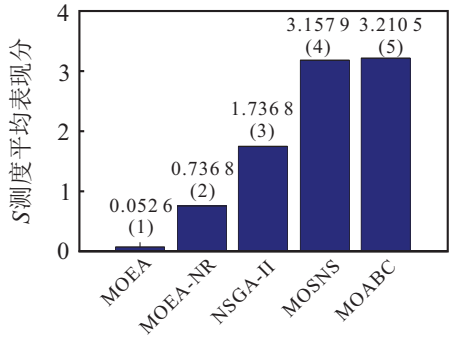

图3　在所有测试问题上5个算法所得平均表现分

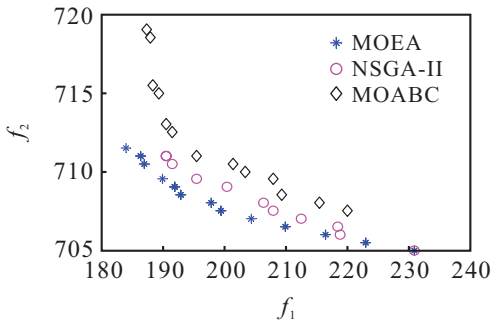

图4　3种算法求解MK05时所得PF中点

表5为MOEA与3种对比算法求解LeiData和WangData测试集时的CPU消耗时间(单位: s). 可以看出,除了WangData1,MOEA在其他所有问题上的运行时间都少于NSGA-II和MOSNS,略多于MOABC. 因此,MOEA能够在保证求解效率的同时得到质量更高的解,比其他3种算法更适合解决MOIFJSSP问题.

表5　MOEA算法与其他算法的CPU运行时间比较

算例	MOEA	NSGA-II	MOABC	MOSNS
LeiData1	157.4	230.9	102.1	210.3
LeiData2	149.5	228.8	102.9	210.9
LeiData3	177.2	259.9	133.7	270.1
LeiData4	164.3	259.1	132.6	273.8
LeiData5	290.6	366.7	262.9	484.5
WangData1	64.7	228.7	46.8	45.5
WangData2	120.3	253.2	101.8	167.3
WangData3	96.0	219.9	70.4	126.4
WangData4	110.6	217.2	71.8	151.2

5　结　论

本文将区间数学理论结合柔性作业车间调度,建立了以区间最大完工时间和区间机器总负荷为指标的区间多目标调度模型,并提出了一种有效求解该模型的多目标进化算法. 在区间数运算下,算法采用混合策略产生初始化种群,并设计贪婪插入法对染色体进行解码. 同时,算法将基于区间意义下的Pareto占优关系和结合目标归一化的超体拥挤距离算子作用于快速非支配排序. 通过MOEA对19个实例进行测试实验,验证了初始化策略的有效性和MOEA对求解MOIFJSSP有较强的寻优能力. 下一步将在本文研究的基础上,进一步研究不确定动态多目标柔性作业车间调度问题.

参考文献(References)

[1] 雷德明. 基于新型教学优化算法的低碳柔性作业车间调度[J]. 控制与决策, 2017, 32(9): 1621-1627.
(Lei D M. Novel teaching-learing-based optimization algorithm for low carbon scheduling of flexible job shop[J]. Control and Decision, 2017, 32(9): 1621-1627.)

[2] Zuo X, Mo H, Wu J. A robust scheduling method based on a multi-objective immune algorithm[J]. Information Sciences, 2009, 179(19): 3359-3369.

[3] Homg S C, Lin S S, Yang F Y. Evolutionary algorithm for stochastic job shop scheduling with random processing time[J]. Expert Systems with Applications, 2012, 39(3): 3603-3610.

[4] Shen X N, Han Y, Fu J Z. Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems[J]. Soft Computing, 2017, 21(21): 6531-6554.

[5] 朱传军,邱文,朱孟周. 随机工时下的多目标柔性作业车间鲁棒调度问题[J]. 中国机械工程, 2017, 27(12): 1667-1671.
(Zhu C J, Qiu W, Zhu M Z. Multi-objective flexible job shops robust scheduling problem under stochastic processing times[J]. China Mechanical Engineering, 2017, 27(12): 1667-1671.)

[6] 王冰,李巧云,羊晓飞. 模糊车间作业调度的三点满意度模型[J]. 控制与决策, 2012, 27(7): 1082-1086.
(Wang B, Li Q Y, Yang X F. Three-point satisfaction-degree model for fuzzy job-shop scheduling problem[J]. Control and Decision, 2012, 27(7): 1082-1086.)

[7] 刘爱军,杨育,刑青松,等. 多目标模糊柔性作业车间调度中的多种群遗传算法[J]. 计算机集成制造系统, 2011, 17(9): 1954-1961.
(Liu A J, Yang Y, Xing Q S, et al. Multi-population genetic algorithm in multiobjective fuzzy and flexible job

shop scheduling[J]. Computer Integrated Manufacturing Systems, 2011,17(9): 1954-1961.)

[8] Wang X, Gao L, Zhang C, et al. A multi-objective genetic algorithm for fuzzy flexible job-shop scheduling problem[J]. Int J of Computer Applications in Technology, 2012, 45(45): 115-125.

[9] Wang X, Li W, Zhang Y. An improved multi-objective genetic algorithm for fuzzy flexible job-shop scheduling problem[J]. Int J of Computer Applications in Technology, 2013, 47(2/3): 280-288.

[10] Zheng Y L, Li Y X, Lei D M. Multi-objective swarm-based neighborhood search for fuzzy flexible job shop scheduling[J]. Int J of Advanced Manufacturing Technology, 2012, 60(9): 1063-1069.

[11] Lei D M. Population-based neighborhood search for job shop scheduling with interval processing time[J]. Computers & Industrial Engineering, 2011, 61(4): 1200-1208.

[12] Lei D M. Interval job shop scheduling problems[J]. Int J of Advanced Manufacturing Technology, 2012, 60(1-4): 291-301.

[13] Lei D M. Multi-objective artificial bee colony for interval job shop scheduling with flexible maintenance[J]. Int J of Advanced Manufacturing Technology, 2013, 66(9-12): 1835-1843.

[14] 杨宏安,王周锋,吕阳阳,等. 工序加工时间不确定条件下作业车间调度问题的区间数求解方法[J]. 计算机集成制造系统, 2014, 20(9): 2231-2240.
(Yang H A, Wang Z F, Lv Y Y, et al. Interval number solving method for job-shop scheduling problem with processing time variability[J]. Computer Integrated Manufacturing Systems, 2014, 20(9): 2231-2240.)

[15] Han Y Y, Gong D W, Jin Y C, et al. Evolutionary multi-objective block lot-streaming flow shop scheduling with interval processing time[J]. Applied Soft Computing, 2016, 42: 229-245.

[16] Sengupta A, Pal T K. On comparing interval numbers[J]. European J of Operational Research, 2000, 127(1): 28-43.

[17] 达庆利,徐泽水. 不确定多属性决策的单目标最优化模型[J]. 系统工程学报, 2002, 17(1): 50-55.
(Da Q L, Xu Z S. Single-objective optimization model in uncertain multi-attribute decision-making[J]. J of Systems Engineering, 2002, 17(1): 50-55.)

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

[19] Zhang G H, Gao L, Shi Y. An effective genetic algorithm for the flexible job-shop scheduling problem[J]. Expert Systems with Applications, 2011, 38(4): 3563-3573.

[20] 赵诗奎,方水良,顾新建. 基于极限调度完工时间最小化的机器选择及FJSP求解[J]. 计算机集成制造系统, 2014, 20(4): 854-865.
(Zhao S K, Fang S L, Gu X J. Machine selection and FJSP solution based on limit scheduling completion time minimization[J]. Computer Integrated Manufacturing Systems, 2014, 20(4): 854-865.)

[21] Pezzella F, Morganti G, Cia G. A genetic algorithm for the flexible job-shop scheduling problem[J]. Computers & Operations Research, 2008, 35(10): 3202-3212.

[22] Gong D W, Qin N N, Sun X Y. Evolutionary optimization algorithm for multi-objective optimization problems with interval parameters[C]. Proc of the 5th IEEE Conf on Bio-Inspired Computing: Theories and Applications. Changsha: IEEE Press, 2010: 411-420.

[23] Lei D M. A genetic algorithm for flexible job shop scheduling with fuzzy processing time[J]. Int J of Production Research, 2010, 48(10): 2995-3013.

[24] Lei D M. Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling[J]. Applied Soft Computing, 2012, 12(8): 2237-2245.

[25] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.

[26] Limbourg P, Aponte D E. An optimization algorithm for imprecise multi-objective problem function[C]. Proc of IEEE Congress on Evolutionary Computation. Edinburgh: IEEE Press, 2005: 459-466.

[27] Ziter E, Thiele L. Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach[J]. IEEE Trans on Evolutionary Computation, 1999, 3(4): 257-271.

[28] Rahmati S H A, Zandieh M, Yazdani M. Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem[J]. Int J of Advanced Manufacturing Technology, 2013, 64(5-8): 915-932.

[29] Bader J, Zitzler E. Hype: An algorithm for fast hypervolume-based many-objective optimization[J]. Evolutionary Computation, 2011, 19(1): 45-76.

(责任编辑：郑晓蕾)