柔性车间生产排产调度优化方法

张贵军 丁 情 王柳静 周晓根
(浙江工业大学信息工程学院 杭州 310023)

摘 要 为满足柔性制造企业在车间生产中合理安排生产排产调度的需要,提出柔性车间生产排产调度优化方法。首先,通过分析车间生产排产问题的特点,制定满足车间应用需求和各种资源限制的生产排产总体流程,从而设计基于约束条件的生产对象关系模型;其次,提出一种动态策略差分进化算法,根据个体之间的拥挤度动态选择变异策略,设计基于工序位置的编解码方案,其能快速有效地进行求解,从而得到最佳调度方案,提高设备运行效率,实现资源利用的最大化;最后,通过6个标准测试函数、FT6-6测试问题及生产调度应用实例验证了算法的有效性。

关键词 生产排产调度,组合优化,差分进化,柔性制造

中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.02.046

Optimization Method of Production Scheduling in Flexible Job

ZHANG Gui-jun DING Qing WANG Liu-jing ZHOU Xiao-gen
(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract To meet the needs of the production scheduling of flexible manufacturing enterprises, an optimization method for production scheduling was proposed. Firstly, an overall flow of the production scheduling which meets the workshop application requirements and various resource constraints is designed by analyzing the characteristics of the production scheduling problem in the enterprise workshop, and a constraint condition based production objective relation model is presented. Secondly, a differential evolution with dynamic strategy is proposed. In the proposed algorithm, the mutation strategy is dynamically selected according to the crowding degree between each individual in the current population. Moreover, a decoding scheme is designed based on the position of the process. Therefore, the optimal scheduling scheme is obtained to improve the operational efficiency of equipment to maximize the utilization of resources. Finally, the effectiveness of the proposed method is verified by six benchmark functions, FT6-6 scheduling problem and practical example.

Keywords Production scheduling, Combinatorial optimization, Differential evolution, Flexible manufacturing

1 引言

随着工业互联网技术的不断发展和创新,柔性制造企业之间的市场竞争愈发激烈,其面临的挑战也越来越大。柔性制造企业要想赢取顾客、占领市场并在市场竞争中成为胜利者,就必须最大化地利用资源来提高生产效率,以最快的速度满足客户对产品的需求。

车间生产排产调度是离散型柔性制造生产计划的关键技术和核心内容[1],其可以定义为在企业车间有限的生产资源和设备工艺约束条件下,合理安排各个工件在相关设备上的加工顺序和加工时间,以保证所选定的目标的生产性能最优。车间生产排产调度问题是一个复杂的问题,具有离散性、复杂性、不确定性和多约束性等特点。若生产过程使用传统的生产排产方法,如根据长期经验进行生产排产,则很难使排产效果达到最优,导致生产效率较低,同时也浪费了生产资源,增加了企业生产成本。因此,有必要设计和应用先进的智能调度算法来实现柔性制造车间生产排产调度的自动化和智能化,从而合理利用有限的生产资源来提高生产效率,增强柔性制造企业的市场竞争力。

柔性车间生产排产调度问题(Flexible Job Shop Scheduling Problem, FJSP)是传统作业车间生产调度问题(Job Shop Scheduling Problem, JSP)的扩展。FJSP问题于1990年由Bruker和Schlic[2]首次提出,该问题除了要解决JSP中确定的工件加工顺序外,还要确定各工件工序分配到哪台机器上进行加工的问题,使生产排产调度问题更加复杂。目前主要采用智能算法或多个智能算法的混合算法[3-7]来求解该问题。Thomalla[8]对有多种可选的加工工序的工件的JSP进行建模,采用拉格朗日松弛法进行求解,对生产排产的处理时间和优化度的矛盾进行了很好的平衡。左益等[9]考虑最大完工时间、机器总负载和瓶颈机器负荷等问题,将混合多目标算法用

到稿日期:2016-11-28 返修日期:2017-01-08 本文受国家自然科学基金(61773346,61573317)资助。

张贵军(1974-),男,博士,CCF会员,主要研究方向为智能信息处理、智能交通系统、优化理论及算法设计,E-mail:zgj@zjut.edu.cn(通信作者);**丁 情**(1991-),女,硕士生,主要研究方向为智能信息处理;**王柳静**(1993-),女,博士生,主要研究方向为智能信息处理;**周晓根**(1987-),男,博士生,CCF会员,主要研究方向为智能信息处理、优化理论及算法设计。

于柔性作业车间调度问题。魏先民[10]将一种基于思维进化的蚁群算法应用于典型生产调度中。Shao 等[11]同时考虑加工工件的加工设备柔性和工艺路线柔性,采用改进的遗传算法对其进行求解。Xia 等[12]采用粒子群优化算法与模拟退火算法相结合的算法来求解多目标柔性车间调度问题,不仅提高了算法的效率,也在合理时间内获得了高质量的解。

本文对车间生产排产调度问题进行研究,根据连续差分进化算法[13]的连续特性,针对车间生产排产问题的离散特性对差分进化算法进行改进[14-16],提出一种动态多策略差分进化算法。该算法将生产排产调度问题中离散的工件工序编码问题转变为易处理的连续编码问题;然后通过种群中个体之间的距离变化来衡量种群的拥挤度,从而针对算法不同阶段动态选择不同的变异策略来指导算法搜索,提高搜索效率;最后对最优解进行解码来确定工件工序加工顺序,依据最大完工时间最小化来选择车间生产排产调度问题[17]的最优加工工序,并根据工件的加工参数来绘制甘特图,以方便工作人员进行车间生产排产。

2 车间生产排产调度问题的数学模型

2.1 问题描述

柔性制造车间生产排产调度问题一般是指在有限的机器设备上加工多个工件[18]。柔性车间生产的柔性体现在设备的使用和设备的安排两个方面。设备使用的柔性是指一台设备不同时刻可以用于多个工件的多道工序加工;设备安排的柔性是指工件的加工设备不是固定或预先确定的,具有可选择性,可以将多台设备组合作为一条或者多条生产线来加工一种工件,使得该工件的最大完工时间最小化。

车间生产排产调度问题一般描述为:有 n 个工件按照一定的顺序在 m 台设备上进行加工,每个工件有 k 道加工工序,每道工序可以在指定的若干台机器上进行加工。每一台机器在同一时间只能加工一个工件的一道工序,同一个工件的每道工序必须在上一道工序加工完成后才能开始加工,并且每道工序的加工时间已知。

2.2 数学模型

在建立车间生产排产问题的数学模型之前,做出如下基本假设[19]:

(1)任何一台机器在同一时刻只能加工某个工件的某一道工序。

(2)任一工件的每道工序必须在它前面的工序加工结束后才能进行下一道工序的加工。

(3)不同工件的加工工序之间没有任何约束。

(4)一旦工序开始加工,就认为其能正常加工完。

(5)机器与工件可能的开始时间都从 0 时刻开始。

(6)不同工件工序需在指定机器集上加工。

同时假定不考虑所讨论的工件在加工过程中某台设备出现故障导致设备停止工件加工的情况。为建模方便,将车间生产排产调度问题表述为:在 m 台机器设备集 $M=\{M_1,M_2,M_3,\cdots,M_m\}$ 上加工 n 个工件集 $F=\{F_1,F_2,F_3,\cdots,F_n\}$,工件 F_i 包含 n_i 道工序,则所有工件集的总工序为 $I=\sum_{i=1}^{n}n_i$。每个工件都有预定的工件加工顺序,每道工序需要在指定的机器上进行加工,并且已知每道工序的加工时间。在满足上述基本约束的条件下,本文采用的性能指标是最大完工时间最短,数学描述如下。

目标函数为:

$$S=\min\left(\max_{i=1}^{n}\left(\max_{k=1}^{m}C_{ik}\right)\right) \tag{1}$$

约束条件为:

$$C_{is}-P_{is}+X(1-x_{its})\geqslant C_{it} \tag{2}$$

$$C_{js}-C_{is}+X(1-Y_{ijs})\geqslant P_{js} \tag{3}$$

$$C_{is}\geqslant 0 \tag{4}$$

$$x_{its}=\begin{cases}1, & t\neq s\\ 0, & t=s\end{cases} \tag{5}$$

$$Y_{ijs}=\begin{cases}1, & i\neq j\\ 0, & i=j\end{cases} \tag{6}$$

上述各式中,$i,j=1,2,\cdots,n$;$s,t=1,2,\cdots,m$。其中 P_{is} 和 C_{is} 分别为工件 i 在机器 s 上的加工时间和完成时间;X 是一个无穷大的正数;x_{its} 是一个决策变量,若机器 t 在机器 s 之前对工件 i 进行加工,则 x_{its} 等于 1,否则等于 0;Y_{ijs} 是一个决策变量,若工件 i 在工件 j 之前在机器上进行加工,则 Y_{ijs} 等于 1,否则等于 0。

3 动态策略差分进化算法

针对车间生产排产调度问题,本文提出一种动态策略差分进化算法。首先对工件的工序进行编码,将排产调度问题中离散的工序转变为差分进化算法可处理的连续解;然后根据种群中个体之间的距离动态选择变异策略来指导算法搜索;最后对算法搜索得到的最优解进行解码,确定工件的工序加工顺序,从而依据最大完工时间最小化来选择车间生产排产调度问题的最优加工工序。

3.1 编码与种群初始化

(1)编码。本文提出基于工序的编码方法,将调度编码为所有工序的一个排序序列,赋予同一工件的工序相同的符号以区分不同工件的工序,根据工件号在 DE 个体向量中出现的顺序对同一工件的工序加以区分。对于 n 个工件、m 台机器的车间生产排产调度问题,每个 DE 个体是一个 $n\times m$ 维的向量,向量中的每一维是指具有前后依赖关系的工件的一道工序,对每个个体解码后可以得到一个可行的调度方案。

以 3 台设备和 3 个工件(每个工件有 3 道加工工序)为例,即求解 3×3 问题。假设 DE 个体向量为[1 2 2 3 3 2 1 1 3],数字代表工件号,且每个数字均只出现 m 次。设 O_{ij} 为第 i 个工件的第 j 道工序,则每个个体代表的有序工序为[O_{11} O_{21} O_{22} O_{31} O_{32} O_{23} O_{12} O_{13} O_{33}]。

由于车间生产排产问题具有离散性,因此在采用 DE 算法进行求解时需要进行相应的修改。具体做法:在基于工序的编码基础上,记录 DE 个体向量中各工序的位置,赋予同一工件的工序(即上例中个体向量中的相同数字)升序排列的 m 个数字,这 m 个数字的起始数字由工件号的升序排列决定。例如,上例基于工序的编码为[1 2 2 3 3 2 1 1 3],记录位置后的编码为[1 4 5 7 8 6 2 3 9],其中数字 1-3 代表第一个工件的工序,数字 4-6 代表第二个工件的工序,数字 7-9 代表第三个工件的工序,各工件对应的数字升序排列代表其工序的先后顺序,故每个数字均只出现一次。至此,将 DE 个体向量由基于工序的离散编码转化为表示工序位置的连续编码,进

而可以采用 DE 算法进行后续的操作。

(2)种群初始化。对上述编码序列中的元素进行 N_p 次随机排序，将每次排序作为一个个体，从而组成一个规模为 N_p 的初始种群。由于在编码设计时已经保证了随机产生的 DE 个体满足约束条件，因此初始种群可由简单的随机函数产生，无需进一步验证 DE 个体的可行性和合法性。

3.2　动态变异策略

变异操作是为了增加种群的多样性，DE 缩放种群内的差分向量会对种群内其他的相异个体形成扰动。根据不同的变异向量生成方法得到了多种变异策略，然而不同的变异策略具有不同的属性，例如，DE/rand/1 的全局探测能力较强，但是局部搜索能力较弱，而 DE/best/1 的局部搜索能力较强，但是容易陷入局部最优而出现早熟收敛现象[20]。因此，如何选择适当的变异策略来平衡算法的全局探测能力和局部搜索能力至关重要。

为了针对算法的不同阶段选择不同的变异策略，本文提出了一种动态变异策略。通过种群中的个体之间的相互距离来衡量种群的拥挤度，然后根据拥挤度来判断算法所达到的状态，进而选择适当的变异策略。假设当前种群迭代次数为 G，则当前种群的拥挤度(即各种群个体之间的平均距离)为：

$$d=\frac{\sum_{i=1}^{N_p}\sum_{k=i+1}^{NP}\sqrt{\sum_{j=1}^{N}(x_{ji,G}-x_{jk,G})^2}}{N_p(N_p-1)/2} \tag{7}$$

其中，N_p 为种群规模；当初始种群 $G=0$ 时，由于个体相对比较分散，因此认为初始种群的拥挤度值最大，用 $d_{\max}$ 表示；根据种群拥挤度的变化，可以对不同阶段选择合适的变异策略，即：

如果 $d\geqslant\frac{2}{3}d_{\max}$，那么

$$v_{ji,G}=x_{jr_1,G}+F\cdot(x_{jr_2,G}-x_{jr_3,G}) \tag{8}$$

如果 $\frac{1}{3}d_{\max}\leqslant d<\frac{2}{3}d_{\max}$，那么

$$v_{ji,G}=x_{ji,G}+F\cdot(x_{j,G}^{best}-x_{ji,G}+x_{jr_1,G}-x_{jr_2,G}) \tag{9}$$

如果 $d<\frac{1}{3}d_{\max}$，那么

$$v_{ji,G}=x_{j,G}^{best}+F\cdot(x_{jr_1,G}-x_{jr_2,G}) \tag{10}$$

其中，$j=1,2,\cdots,n\times m$；G 为进化代数；$r_1,r_2,r_3\in\{1,2,\cdots,N_p\}$，$r_1\neq r_2\neq r_3\neq i$；$v_{ji,G}$ 为第 G 代种群中第 i 个变异个体的第 j 维元素；$x_{jr_1,G}$，$x_{jr_2,G}$ 和 $x_{jr_3,G}$ 分别为第 G 代种群中第 r_1，r_2，r_3 个个体的第 j 维元素；F 是缩放因子；$x_{j,G}^{best}$ 为当前种群中的最优个体。

经变异操作后可能产生不可行解，但考虑到变异操作是为了增加种群的多样性，故对不可行解不做处理。

3.3　交叉操作

根据编码设计规则，交叉操作后可能产生不可行解，而实验个体要参与后续选择操作，因此必须设置约束条件，对不符合编码规则的实验个体进行处理。具体操作：若经交叉操作的实验个体不符合编码规则，则继续循环变异交叉操作，直至产生可行解。另外，将多个父代个体按照一定的规则进行交叉组合，实现种群的多样性，生成实验向量 $u_{ji,G+1}$：

$$u_{ji,G+1}=\begin{cases}v_{ji,G+1}, & \text{if } rand(0,1)\leqslant C_R \text{ or } j=j_{rand}\\ x_{ji,G}, & \text{otherwise}\end{cases} \tag{11}$$

其中，$u_{ji,G+1}$ 表示第 $G+1$ 代种群中的第 i 个实验个体的第 j 维；$CR\in[0,1]$ 为交叉概率，每一代均会自动更新。另外，$rand(0,1)$ 表示 0 和 1 之间的随机小数，j_{rand} 为 0 到 $n\times m$ 之间的随机整数。

3.4　适应值与解码

(1)适应值

作为选择操作的依据，本文直接将目标函数值作为适应值来评估个体或解的优劣。

(2)解码

在解码过程中，需要确定多个机器上工件的工序加工顺序，而且这些加工顺序是相互耦合的。首先，将实验个体转化为基于工件工序的加工序列；其次，确定所有工件的每道工序在加工机器上的开始时间和完工时间；然后，基于该加工序列和工艺约束对各工序以最早允许加工时间逐一进行加工，计算每台机器最后的工作完成时间，取其中的最大值作为适应值。至此便完成了解码过程，产生了相应的调度方案。

3.5　选择操作

为判断实验向量是否会成为下一代种群中的个体，通过采用贪婪法则比较实验向量与当前种群中目标向量的适应值来选择最优个体。下一代中的所有个体都应该优于或等于当前种群中对应的个体。具体选择操作如下：

$$x_{i,G+1}=\begin{cases}u_{i,G}, & \text{if } f(u_{i,G})\leqslant f(x_{i,G})\\ x_{i,G}, & \text{otherwise}\end{cases} \tag{12}$$

其中，$x_{i,G+1}$ 表示第 $G+1$ 代种群的第 i 个个体，$u_{i,G}$ 和 $x_{i,G}$ 分别是第 G 代种群中的第 i 个变异个体和第 G 代种群的第 i 个个体，$f(x)$ 是适应度函数。

4　数值仿真

为了验证所提算法 DMDE 的性能，选取标准 DE[13]，DELLU[14]，LPDE[15] 和 LLUDE[16] 算法进行了比较分析。实验中，采用 6 个典型的标准测试函数进行测试。表 1 列出了各个测试函数的名称、数学表达式、搜索范围及全局最优值，其中，函数 f_1-f_3 为单模函数，f_4-f_6 为多模函数。

表 1　标准测试函数

Table 1　Benchmark function

函数名	数学表达式	搜索范围	全局最优值
Sphere	$f_1(x)=\sum_{i=1}^{N}x_i^2$	$[-100,100]$	0
Zakharov	$f_3(x)=\sum_{i=1}^{N}x_i^2+(\sum_{i=1}^{N}0.5ix_i)^2+(\sum_{i=1}^{N}0.5ix_i)^4$	$[-5,10]$	0
Rosenbrock	$f_4(x)=\sum_{i=1}^{N-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	$[-2,2]$	0
Griewank	$f_5(x)=1+\frac{1}{4000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}})$	$[-600,600]$	0
Ackley	$f_9(x)=-20\exp(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2})-\exp(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i))+20+e$	$[-30,30]$	0
Rastrigin	$f_{10}(x)=10N+\sum_{i=1}^{N}(x_i^2-10\cos(2\pi x_i))$	$[-5,5]$	0

DMDE 算法中各参数的设置如下：种群规模 $NP=50$，交叉概率 CR 和缩放因子 F 均设置为 0.5。DELLU，LPDE 和

LLUDE 算法的参数设置同文献[14-16]。标准 DE 算法选用“DE/rand/1/bin”策略，种群规模 $NP=50$，交叉概率 CR 和缩放因子 F 均设置为 0.5。为了实现公平比较，所有算法均设置相同的运行终止条件，且每种算法对每个测试函数均独立运行 30 次。

本文实验环境为：Intel(R) Core i5-2410M CPU @ 2.30GHz，8GB RAM，Windows 7；算法代码采用 Visual Studio C++ 2012 实现。

采用函数评价次数 *FES* 和成功率 *SR* 两个指标来验证算法的收敛速度和可靠性。*FES* 表示算法在设定的目标函数评价次数(300000)内达到设定精度(0.00001)时所需的目标函数评价次数；*SR* 为算法的成功求解次数和总运行次数之比。规定算法在数评价次数 300000 内达到设定的精度时是成功的。

表 2 列出了各函数的函数评价次数 *FES* 和成功率 *SR*。其中，加粗数值表示最优结果，“*NA*”表示算法在函数评价次数内的求解没有达到设定的精度。从表中数据可以看出，除了函数 f_6 以外，所提 DMDE 算法对于其他函数的收敛速度均优于对比算法，且能够以 100% 的成功率进行求解，DE 算法对于函数 f_3 和 f_6 无法成功求解。其次，从表 2 最后一行的平均结果可以看出，DMDE 所需的目标函数评价次数最少(为 32746)，而 DE，DELLU，LPDE 和 LLUDE 算法所需函数评价次数分别为 159945，53054，43860 和 39071，即 DMDE 算法相对于 DE，DELLU，LPDE 和 LLUDE 算法分别节省了 79.5%，38.3%，25.3% 和 16.2% 的函数评价次数。在成功率方面，由于 DE 算法对函数 f_3 和 f_6 未能成功求解，导致 DE 算法的成功率仅为 0.66，其他算法(如 DMDE，LLUDE，LPDE 和 DELLU)的成功率均为 1.00。

表 2 函数评价次数和成功率

Table 2 Function evaluations and success rates

函数	维数	DE		DELLU		LPDE		LLUDE		DMDE	
		FES	*SR*	*FES*	*SR*	*FES*	*SR*	*FES*	*SR*	*FES*	*SR*
f_1	30	30100	1.00	12027	1.00	8432	1.00	12841	1.00	**8059**	1.00
f_2	30	256389	1.00	80930	1.00	104312	1.00	84941	1.00	**76325**	1.00
f_3	30	*NA*	0.00	159574	1.00	98342	1.00	61651	1.00	**58032**	1.00
f_4	30	32900	0.97	20020	1.00	11702	1.00	15949	1.00	**10321**	1.00
f_5	30	40282	1.00	28393	1.00	14937	1.00	30656	1.00	**12903**	1.00
f_6	30	*NA*	0.00	**17383**	1.00	25436	1.00	28389	1.00	30837	1.00
平均值		159945	0.66	53054	1.00	43860	1.00	39071	1.00	**32746**	1.00

为了评价算法的求解精度，将各种算法的平均函数误差值(Mean error)和标准偏差值(Std dev)进行比较。其次，选用 Wilcoxon Signed Rank Test 对各算法进行 30 次优化后得到的结果进行非参数假设检验，从而验证所提算法相对于其他对比算法是否有显著性优势。将显著性水平设置为 5%，并且通过“+”表示所提算法显著优于对比算法，“−”表示所提算法显著差于对比算法，“≈”表示两种算法没有显著性差异。

表 3 列出了各测试函数运行 30 次后的平均函数误差和标准偏差。可以看出，除了函数 f_4 和 f_6 以外，DMDE 算法的结果均优于其他对比算法。对于函数 f_4，DELLU，LPDE，LLUDE 和 DMDE 算法均得到了全局最优解。总体而言，DMDE 在 6 个测试函数中的 5 个函数上的结果均优于其他 4 种算法。另外，从表中最后一行的非参数检验结果可以看出，DMDE 算法的结果分别显著优于 DE，DELLU，LPDE 和 LLUDE 算法 6，3，3 和 3 个测试函数，仅显著差于 LPDE 算法 1 个测试函数，从而说明通过 DMDE 得到的解的质量较高。

表 3 优化结果性能表

Table 3 Performance of optimization results

函数	维数	DE	DELLU	LPDE	LLUDE	DMDE
		Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)	Mean Error(Std Dev)
f_1	30	4.06E−13(1.66E−13)+	1.65E−37(2.25E−37)+	2.78E−40(1.93E−40)+	1.58E−37(2.03E−37)+	**3.90E−41(4.50E−41)**
f_2	30	2.86E+01(1.37E+01)+	1.53E−04(2.33E−04)+	8.35E−03(5.69E−03)+	1.48E−04(2.36E−04)+	**7.95E−05(3.57E−05)**
f_3	30	2.44E+01(7.40E−01)+	1.48E−05(1.62E−05)+	3.41E−04(5.03E−04)+	1.31E−05(1.56E−05)+	**2.93E−06(2.67E−06)**
f_4	30	8.06E−12(1.42E−11)+	**0.00E+00(0.00E+00)**≈	**0.00E+00(0.00E+00)**≈	**0.00E+00(0.00E+00)**≈	**0.00E+00(0.00E+00)**
f_5	30	1.38E−07(3.23E−08)+	7.55E−15(1.47E−15)≈	7.55E−15(1.47E−15)≈	7.55E−15(1.49E−15)≈	**4.47E−15(1.28E−15)**
f_6	30	1.26E+02(6.23E+00)+	6.37E−07(5.39E−07)≈	**5.91E−07(7.23E−07)**−	6.27E−07(5.35E−07)≈	7.12E−07(4.90E−07)
+/≈/−		6/0/0	3/3/0	3/2/1	3/3/0	

从上述实验结果和分析可以看出，本文所提 DMDE 算法不仅收敛速度快、成功率高，而且得到的解的质量较高。

5 案例分析

首先，基于 Visual Studio C++ 2012 和 MATLAB 组合编程对 FT6-6 车间生产排产调度问题进行仿真。其规模为 6 台机器和 6 种工件，每种工件包含 6 道工序，则其运算维数为 36，最优值为 55，工件的工序工时为矩阵 T，工件工序的加工设备为矩阵 M。

为了验证所提 DMDE 算法的优势，选取两种先进的改进差分进化算法 SaDE[21] 和 CoDE[22] 进行比较。为了公平比较，设置 SaDE 和 CoDE 算法的参数均与原文献相同，设置 DMDE 算法的参数与第 4 节相同，且各算法针对上述问题独立仿真 10 次，终止条件均为 20000 次函数评价。

$$T=\begin{bmatrix}1&3&6&7&3&6\\8&5&10&10&10&4\\5&4&8&9&1&7\\5&5&5&3&8&9\\9&3&5&4&3&1\\3&3&10&10&4&6\end{bmatrix},M=\begin{bmatrix}3&1&2&4&6&5\\2&3&5&6&1&4\\3&4&6&1&2&5\\2&1&3&4&5&6\\3&2&5&6&1&4\\2&4&6&1&5&3\end{bmatrix}$$

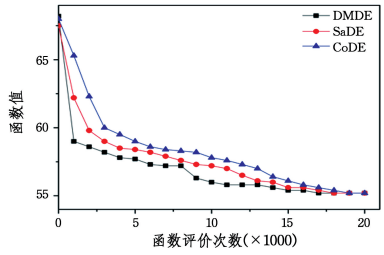

图 1　FT6-6 问题的仿真收敛图

Fig. 1　Simulation convergent diagram of FT6-6

以函数评价次数为横轴，以最大完成时间（目标函数值）为纵轴，绘制各算法进行 10 次仿真的平均收敛曲线图，如图 1 所示。

从图 1 中可以看出，DMDE，SaDE 和 CoDE 3 种算法均能达到最优值 55，所对应的最优解为：[2 6 3 1 2 5 1 3 4 1 5 3 4 2 6 4 6 2 3 6 4 5 2 3 1 5 5 1 4 3 6 4 6 1 2 5]，但是 DMDE 算法的收敛速度快于 SaDE 和 CoDE。

其次，基于 J2EE 规范，采用 AngularJS 前端开发框架、SSH 后端开发框架设计并开发了柔性制造系统，系统主界面如图 2 所示。系统的生产计划部分包含了生产排产调度模块，这是系统的主要模块。该模块可以通过选择不同的排产问题测试基准生成对应的最优调度甘特图，同时还生成相应的最优工序调度表，以便工作人员查看。将上述算法所求最优解通过编程在系统排产调度模块中实现，其对应的甘特图如图 3 所示。

图 2　系统主界面图

Fig. 2　Main interface of system

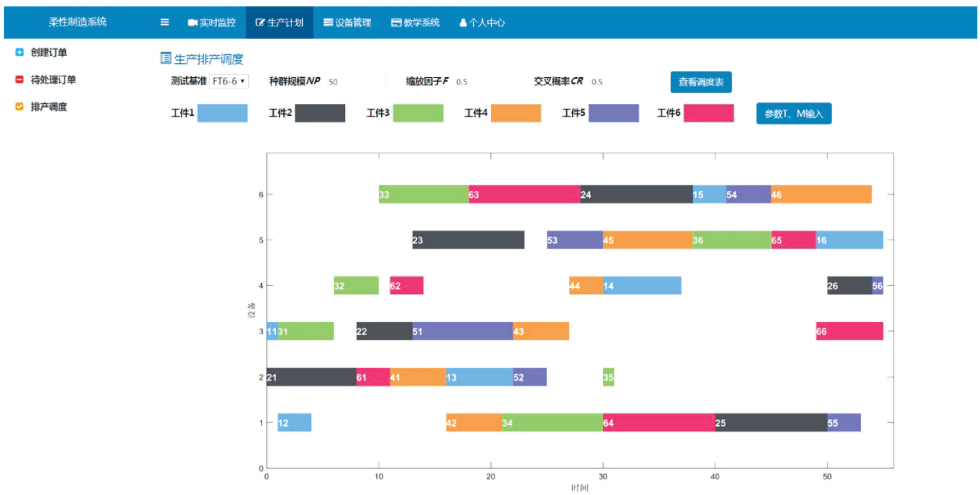

图 3　最优调度方案的甘特图

Fig. 3　Gantt graph of the optimal scheduling scheme

由图 3 可知，在柔性制造系统排产调度模块的最优调度甘特图上，不同的工件用不同的颜色的深浅程度标注，以便工作人员清晰直观地查看不同设备上各个工件的不同加工工序。通过“查看调度表”功能得到对应的最优调度表，如图 4 所示。通过图 4 中的最优调度表可知，算法可以根据生成解的代码在该系统中计算出工件工序所对应的加工机器编号、工序的开始加工时间和工序的结束时间。经调度表查询可知，机器号为 1 的设备工件工序的最大完成时间为 53 个单位时间；机器号为 2 的设备工件工序的最大完成时间为 31 个单位时间；机器号分别为 3，4 和 5 的设备的最大完成时间均为 55 个时间单位；机器号为 6 的设备的最大完成时间为 53 个单位时间。因此，通过分析可知，程序最终所得的最大完工时间为 55 个单位时间，即所有工件最终的完工时间为 55 个单位时间。

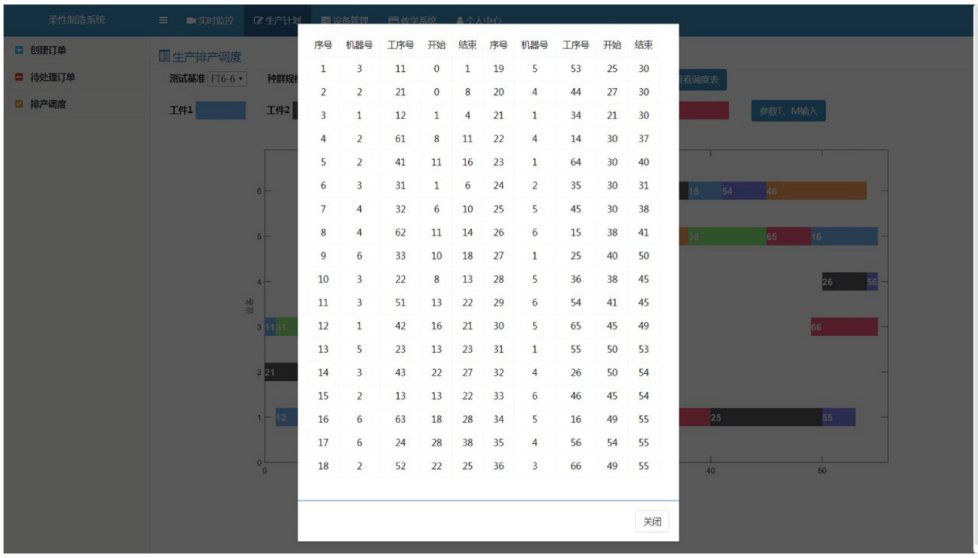

图 4 FT6-6 问题的最优调度方案

Fig. 4 The optimal scheduling scheme of FT6-6

结束语 本文对车间生产排产调度问题进行了研究，在满足约束条件的基础上建立了相应的调度模型。由于车间生产排产调度问题属于典型的组合优化问题，而组合优化问题属于离散问题，但差分进化算法主要解决连续域内的优化问题，因此需要对算法进行改造以适应相应问题。本文针对车间生产排产调度问题的特点，设计了相适应的编解码方案，将离散工序优化问题转化为能够处理的连续问题，并提出了动态策略差分进化算法，根据种群的拥挤度变化选择合适的策略来指导算法搜索，从而对调度模型进行快速求解。仿真实验和系统排产调度模块方案图表明了算法的可行性与有效性。

参 考 文 献

[1] ZHU X H，ZHU J F，JIANG T. Comparison of several models of job shop scheduling problem[J]. Statistics and Decision，2007(23)：174-176. (in Chinese)
朱星辉，朱金福，姜涛. 作业车间调度问题的几种模型之比较[J]. 统计与决策，2007(23)：174-176.

[2] BRUKER P，SCHLIC R. Job-shop Scheduling with Multi-purpose Machines[J]. Computing，1990，4(2)：369-375.

[3] 张智海. 调度：原理、算法和系统[M]. 北京：清华大学出版社，2007.

[4] XIONG R，WU C. Technical status and development trend of workshop production scheduling problem[J]. Journal of Tsinghua University，1998，38(10)：55-60. (in Chinese)
熊锐，吴澄. 车间生产调度问题的技术现状与发展趋势[J]. 清华大学学报，1998，38(10)：55-60.

[5] GRAVES S C. A review of production scheduling [J]. Operations Research，1981，29(4)：646-675.

[6] LIU M. A review of the research on production scheduling based on data[J]. Acta automatica Sinica，2009，35(6)：785-806. (in Chinese)
刘民. 基于数据的生产过程调度方法研究综述[J]. 自动化学报，2009，35(6)：785-806.

[7] CHEN I J，CHUNG C S. Sequential modeling of the Planning and Scheduling Problems of Flexible Manufacturing Systems [J]. Journal of the Operational Research Society，1996，47(10)：1216-1227.

[8] THOMALLA C S. Job shop scheduling with alternative process plans[J]. International Journal of Production Economics，2001，74(1-3)：125-134.

[9] ZUO Y，GONG M G，CENG J L，et al. Hybrid multi-objective algorithm for flexible job shop scheduling problem[J]. Computer Science，2015，42(9)：220-225. (in Chinese)
左益，公茂果，曾久琳，等. 混合多目标算法用于柔性作业车间调度问题[J]. 计算机科学，2015，42(9)：220-225.

[10] WEI X M. Application of ant colony algorithm based on mind evolution in typical production scheduling [J]. Computer Science，2013，40(7)：236-238，257. (in Chinese)
魏先民. 基于思维进化的蚁群算法在典型生产调度中的应用[J]. 计算机科学，2013，40(7)：236-238，257.

[11] SHAO X Y，LI X Y，GAO L，et al. Integration of process planning and scheduling——A modified genetic algorithm-based ap-

proach[J]. Computers & Operations Research, 2009, 36(6): 2082-2096.

[12] XIA W J, WU Z M. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J]. Computers & Industrial Engineering, 2005, 48(2): 409-425.

[13] STORN R, PRICE K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[14] ZHOU X G, ZHANG G J, HAO X H, et al. A differential evolution algorithm based on local Lipschitz lower bound to estimate support surface [J]. Chinese Journal of Computers, 2016, 39(12): 2631-2651. (in Chinese)
周晓根，张贵军，郝小虎，等. 一种基于局部 Lipschitz 下界估计支撑面的差分进化算法[J]. 计算机学报，2016，39(12)：2631-2651.

[15] ZHOU X G, ZHANG G J, HAO X H. Differential evolution algorithm based on local abstract convex region[J]. Acta automatica Sinica, 2015, 41(7): 1315-1327. (in Chinese)
周晓根，张贵军，郝小虎. 局部抽象凸区域剖分差分进化算法[J]. 自动化学报，2015，41(7)：1315-1327.

[16] ZHOU X G, ZHANG G J, HAO X H, et al. Enhanced differential evolution using local Lipschitz underestimate strategy for computationally expensive optimization problems [J]. Applied Soft Computing, 2016, 48(11): 169-181.

[17] LIU X P, XU B Z, PENG J, et al. Job shop scheduling model and its solution for concurrent work processes[J]. Journal of Computer-aided Design & Computer Graphics, 2012, 24(1): 120-127. (in Chinese)
刘晓平，徐本柱，彭军，等. 工件工序可并行的作业车间调度模型与求解[J]. 计算机辅助设计与图形学学报，2012，24(1)：120-127.

[18] CUI J S, LI T K, ZHANG W X. Hybrid flow shop scheduling model and its genetic algorithm[J]. Journal of University of Science and Technology Beijing, 2005, 27(5): 623-626. (in Chinese)
崔建双，李铁克，张文新. 混合流水车间调度模型及其遗传算法[J]. 北京科技大学学报，2005，27(5)：623-626.

[19] ZHOU X. Design and implementation of workshop scheduling system for distributed CNC system[D]. Shenyang: University of Chinese Academy of Sciences, 2014. (in Chinese)
周鑫. 分布式数控系统车间排产系统的设计与实现[D]. 沈阳：中国科学院大学，2014.

[20] ZHOU X G, ZHANG G J, HAO X H, et al. A novel differential evolution algorithm using local abstract convex underestimate strategy for global optimization[J]. Computers & Operation Research, 2016, 75(11): 132-149.

[21] QIN A K, HUANG V L, SUGANTHAN P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 398-417.

[22] WANG Y, CAI Z, ZHANG Q. Differential evolution with composite trial vector generation strategies and control parameters [J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 55-66.

(上接第 248 页)

[28] GOEL A L, OKUMOTO K. Time-dependent error-detection rate model for software reliability and other performance measures[J]. IEEE Transactions on Reliability, 1979, 3: 206-211.

[29] ZHANG R H, JIANG N, GOU L, et al. Software reliability growth model considering defect correlation[J]. Computer Engineering, 2008, 34(8): 44-46. (in Chinese)
张荣辉，姜楠，勾朗，等. 一种考虑缺陷关联的软件可靠性增长模型[J]. 计算机工程，2008，34(8)：44-46.

[30] MUSA J D, IANNINO A, OKUMOTO K. Software reliability, measurement, prediction and application [M]. McGraw Hill, 1987.

[31] HUANG C Y, LIN C T, KUO S Y, et al. Software reliability growth models incorporating fault dependency with various debugging time lags[C]// Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004 (COMPSAC 2004). IEEE, 2004: 186-191.

[32] LIN C T. Enhancing the accuracy of software reliability prediction through quantifying the effect of test phase transitions[J]. Applied Mathematical Computation, 2012, 219(5): 2478-2492.

[33] PILLAI K, NAIR V S. A model for software development effort and cost estimation[J]. IEEE Transactions on Software Engineering, 1997, 23(8): 485-497.

[34] CHIU K C, HUANG Y S, LEE T Z. A study of software reliability growth from the perspective of learning effects[J]. Reliability Engineering & System Safety, 2008, 93(10): 1410-1421.

[35] WOOD A. Predicting software reliability[J]. Computer, 1996, 29(11): 69-77.

[36] HUANG C Y, KUO S Y. Analysis of incorporating logistic testing-effort function into software reliability[J]. IEEE Transactions on Reliability, 2002, 51(3): 261-270.

[37] XIE M, YANG B. A study of the effect of imperfect debugging on software development cost[J]. IEEE Transactions on Software Engineering, 2003, 29(5): 471-473.