基于动态策略的差分进化柔性车间优化调度

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

摘 要 针对柔性作业车间调度问题,提出基于动态策略的差分进化优化方法。首先,基于差分进化算法框架,考虑个体之间的距离,设计种群拥挤度指标来衡量当前种群的分布情况,进而自适应判断算法所处阶段;然后,针对不同阶段的特点设计相应的变异策略池,实现变异策略的动态阶段选择,达到提高算法搜索效率的目的;最后,10 个标准测试函数的计算结果表明了所提方法的有效性,进一步,采用工序和机器双层编码的方式,以最大完工时间为目标,求解得到作业车间调度测试问题的最佳调度方案。

关键词 柔性作业车间调度,差分进化,变异策略,动态策略,双层编码
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.10.044

Dynamic Strategy-based Differential Evolution for Flexible Job Shop Scheduling Optimization

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

Abstract To solve the flexible job shop scheduling problem, a differential evolution optimization method based on dynamic strategy was proposed in this paper. Firstly, based on the framework of differential evolution algorithm, taking the distance between individuals into consideration, the indicator of population crowding degree was designed, which for measuring the distribution of the current population, and the stage of the algorithm can be further determined adaptively. Then, in view of characteristics of different stages, the corresponding mutation strategy pool was designed to realize the dynamic stage selection of mutation strategy, so as to improve the search efficiency of the algorithm. Finally, the test results on 10 benchmark functions show that the proposed algorithm is feasible and efficient. Based on the double layer coding method of working procedure and machine, the best scheduling scheme was obtained by minimizing the maximum completion time.

Keywords Flexible job shop scheduling, Differential evolution, Mutation strategy, Dynamic strategy, Double layer coding

制造业的发展水平是国民经济实力的重要体现,随着智能制造的不断推进,以多品种、小批量生产方式为主的柔性制造系统(Flexible Manufacturing System,FMS)逐步进入人们的视野,其中车间优化调度作为柔性制造生产计划的关键技术和核心内容[1],在提高生产效率和降低生产成本等方面起到了至关重要的作用[2]。

作业车间调度问题(Job Shop Scheduling Problem,JSP)一般是指如何在有限的生产资源和设备约束条件下,安排有效、合理的生产工序和加工设备,使得预设目标最优。该问题通常是多约束、多目标、随机不确定的,难以用经典优化方法求解,因此智能优化方法(如差分进化算法[3]、禁忌搜索算法[4]、粒子群算法[5]和蚁群算法[6]等)被广泛应用于该类问题的求解。

柔性作业车间调度问题(Flexible Job Shop Scheduling Problem,FJSP)是传统作业车间调度问题的一个扩展,它允许一个操作由一个给定集合中的任何一台机器处理。因为能完成每道工序加工任务的机器可能有多台,所以该问题比作业车间调度更复杂[7]。近几十年,国内外众多学者对柔性作业车间调度问题进行了广泛而深入的研究。Gu 等[8]提出了基于多种群改进协同进化思想和量子理论概念的两种群量子遗传算法,并将其应用于作业车间调度问题,表现出了良好的性能;蔡良伟等[9]提出了由多个普通种群和一个优良种群构成的多种群遗传算法来求解 FJSP,通过使用优良种群中的优良子个体来替代当前种群中的较劣个体,以改善种群的品质,提升算法性能;Wang 等[10-11]分别使用人工蜂群算法和分布估计算法求解 FJSP,在求解过程中平衡了算法的局部搜索和全局搜索性能;Yazdani 等[12]提出了并行变邻域结构搜索算法来缩短 FJSP 的完工时间;Chen 等[13]提出了一种对工序编码和机器编码方式区别对待的分布式编码方式的进化算法来求解单目标 FJSP,有效地描述了排产调度的整个过程,为后续对基于进化算法的 FJSP 的研究提供了较好的参考;杨晓梅等[14]提出了一种新的求解 FJSP 问题的遗传算法,利用编

到稿日期:2017-09-19 返修日期:2017-12-08 本文受国家自然科学基金(61773346,61573317),浙江省重点研发计划项目(2017C03060)资助。
张贵军(1974-),男,博士,教授,CCF 会员,主要研究方向为智能信息处理、全局优化理论及算法设计,E-mail:zgj@zjut.edu.cn(通信作者);**王 文**(1994-),男,硕士生,主要研究方向为智能信息处理;**周晓根**(1987-),男,博士生,CCF 学生会员,主要研究方向为智能信息处理和优化理论及算法设计;**王柳静**(1993-),女,硕士生,主要研究方向为智能信息处理。

码方式表示各工序的调度顺序及其加工的机器，由此产生可行的调度方案。

本文对柔性作业车间调度问题进行研究，对差分进化算法进行改进[15-17]，采用双层编码方式将离散调度问题转化为差分进化算法易处理的问题，进而提出一种动态策略的差分进化算法。该算法首先通过种群中个体的拥挤程度来判断算法处于全局探测阶段还是局部搜索阶段；其次根据最大完工时间选择车间排产调度问题的最优加工工序和相应的加工机器；最后对求解出的最优解进行解码。

1　柔性作业车间调度问题的数学模型

1.1　问题描述

柔性作业车间调度问题可描述为：有 n 种工件在 m 台机器上加工，每种工件包含 n_i 道工序，每道工序可以在多台性能不同的机器上加工，加工时间随机器性能的不同而变化。柔性作业车间调度要解决的问题是：确定最优加工顺序及每个工序加工的设备，在满足各种约束的条件下使得调度目标最优。

1.2　数学模型

在建立柔性作业车间调度问题的数学模型之前，做如下基本假设[18]：

1）每台机器在某一时间最多只能加工一道工序。

2）每个工件都有若干道工序，同一工件的不同工序有加工顺序的约束，不同工件间则没有。

3）工序一旦被机器加工就不能被打断。

4）可选机器集在零时刻都能被使用。

5）工序必须在其指定的机器集上加工。

变量描述：

i 表示工件序号，$i=1,2,3,\cdots,n$；j 表示工件的工序，$j=1,2,3,\cdots,n_i$；k 表示机器的序号，$k=1,2,3,\cdots,m$；P_{ijk} 表示工件 i 的第 j 道工序在机器 k 上的加工时间；C_{ijk} 表示工件 i 的第 j 道工序在机器 k 上的完成时间；

$$X_{ijk}=\begin{cases}1, & \text{若工件 } i \text{ 的第 } j \text{ 道工序在机器 } k \text{ 上加工}\\ 0, & \text{其他}\end{cases}$$

$$R_{ijegk}=\begin{cases}1, & \text{工件 } i \text{ 的第 } j \text{ 道工序和工件 } e \text{ 的第 } g \text{ 道}\\ & \text{工序在同一台机器 } k \text{ 上加工，且工序 } j\\ & \text{先于工序 } g\\ 0, & \text{其他}\end{cases}$$

目标函数：

$$f=\min(\max C_i),i=1,2,\cdots,n \tag{1}$$

约束条件：

1）工序约束，即前道工序完成后才可以开始后面的工序。

$$\begin{aligned}&C_{ijk}-C_{i(j-1)l}-P_{ijk}\geqslant 0\\&X_{ijk}=X_{i(j-1)l}=1\end{aligned} \tag{2}$$

2）机器约束，即同一台机器 k 上加工完一个任务后才能开始另一个加工任务。

$$\begin{aligned}&C_{abk}-C_{ijk}-P_{abk}\geqslant 0\\&X_{ijk}=X_{abk}=1,R_{ijabk}=1\end{aligned} \tag{3}$$

2　基于动态策略的差分进化算法

2.1　双层编码

本文采用基于工序和工序对应加工机器的双层编码方法，即对于一个可行的调度 $X=[x_1,x_2,\cdots,x_D]^{\mathrm{T}}$（$D=2d$，$d$ 表示当前工序的总和），令其前半部分 $X_1=[x_1,x_2,\cdots,x_d]^{\mathrm{T}}$ 表示工序调度的编码，即从第一个工件的第一道工序到最后一个工件的最后一道工序按照整数 1,2,3,…,d 的方式计数；后半部分 $X_2=[x_{d+1},x_{d+2},\cdots,x_D]^{\mathrm{T}}$ 表示工序所对应加工的机器，即 x_{d+l} 表示工序 l 在其可加工机器集上的第 x_{d+l} 个机器上加工，例如工序 l 可选的机器集为 $\{M_1,M_3,M_4\}$，当 $x_{d+l}=2$ 时，表示该工序选择在机器 M_3 上进行加工。

基于上述编码方式，对种群进行初始化。初始种群个体规模为 N_p，对前半部分序列进行 N_p 次随机排序，将每次排序结果作为一个个体的前半部分序列；后半部分序列通过每个工序对应的机器集随机产生。

解码过程中，需要确定每台机器上所需加工的工序的顺序。首先，将实验个体前半部分序列转化为基于工件工序的加工序列；其次，通过后半部分序列确定每台机器上加工工序的顺序；最后，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。基于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完工时间，取其最大完工时间对应的解为最佳调度方案。

2.2　动态变异策略

变异策略利用种群中不同个体的差分向量对个体进行扰动，进而增加种群的多样性。目前，国内外学者提出了各种变异策略来提高 DE 算法的性能[19]，然而不同的变异策略具有不同的属性。例如，DE/rand/1 具有较强的全局探测能力，但是局部搜索能力较弱；而 DE/best/1 具有较强的局部搜索能力，但是容易陷入局部最优而出现早熟收敛现象[20]。因此，在算法的不同阶段选择不同的变异策略是提高算法性能的重要手段。

为了平衡算法的全局探测和局部搜索能力，在 2.1 节所述编码方式的基础上，本文提出了一种基于动态策略的差分进化算法。该算法通过种群中个体之间的相互距离来衡量种群的拥挤度，从而根据拥挤度来判断算法所处的阶段，进而选择适合不同状态的变异策略。假设当前种群的迭代次数为 G，当前种群的拥挤度即各种群个体之间的平均距离为：

$$dist=\frac{\sum_{i=1}^{N_P}\sum_{k=i+1}^{N_P}\sqrt{\sum_{j=1}^{N}(x_{ji,G}-x_{jk,G})^2}}{N_p(N_p-1)/2} \tag{4}$$

其中，N_P 为种群规模；$x_{ji,G}$，$x_{jk,G}$ 代表第 G 代种群中的第 i，k 个个体的第 j 维元素。

根据 $dist$ 的最大值和最小值对每一代 $dist$ 进行归一化处理，得到种群的拥挤度为：

$$\overline{dist}=\frac{dist-dist_{\min}}{dist_{\max}-dist_{\min}} \tag{5}$$

其中，当种群中所有个体收敛到一个全局最优解或局部最优解时，个体间的平均距离取得最小值 $dist_{\min}=0$；假设初始种群中各个体间的平均距离最大，即 $dist_{\max}=dist$，且在进化过程中，如果某一代的 $dist$ 大于 $dist_{\max}$，则取当前的 $dist$ 作为 $dist_{\max}$。

根据种群拥挤度 $\overline{dist}$ 的变化，对不同阶段选择相应的变异策略，即：

$$\phi=\begin{cases}S_1, & \text{if } rand(0,1)>\overline{dist}\\ S_2, & \text{otherwise}\end{cases} \tag{6}$$

其中，ϕ 代表算法所处的阶段，$rand(0,1)$ 代表 0 到 1 之间的

随机数，S_1 代表算法处于全局探测阶段，S_2 代表算法处于局部搜索阶段。

1)当算法所处的阶段为 S_1 时，从以下策略中随机选取一个策略进行变异：

①DE/rand/1

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

②DE/rand/2

$v_{ji,G}=x_{jr_1,G}+F\cdot(x_{jr_2,G}-x_{jr_3,G})+F\cdot(x_{jr_4,G}-x_{jr_5,G})$

③DE/current-to-rand/1

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

2)当算法所处的阶段为 S_2 时，从以下策略中随机选择一个策略进行变异。

①DE/best/1

$v_{ji,G}=x_{j,G}^{pbest}+F\cdot(x_{jr_2,G}-x_{jr_3,G})$

②DE/best/2

$v_{ji,G}=x_{jr_1,G}+F\cdot(x_{j,G}^{pbest}-x_{jr_1,G})+F\cdot(x_{jr_2,G}-x_{jr_3,G})$

③DE/current-to-best/1

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

其中，G 为进化代数；$r_1,r_2,r_3,r_4,r_5\in\{1,2,\cdots,N_p\}$，为第 G 代种群中第 i 个变异个体的第 j 维元素，且 $r_1\neq r_2\neq r_3\neq r_4\neq r_5\neq i$；$x_{jr_1,G},x_{jr_2,G},x_{jr_3,G},x_{jr_4,G},x_{jr_5,G}$ 分别为第 G 代种群中第 r_1,r_2,r_3,r_4,r_5 个个体的第 j 维元素；$K=0.5$；F 是缩放因子；$x_{j,G}^{pbest}$ 为从第 G 代种群中的 $0.5N_p\cdot rand(0,1)$ 个个体中随机选取的最优个体的第 j 维元素。

2.3 交叉操作

由于采用了编码设计方式，交叉操作后可能产生不可行解，而测试个体要参与后续选择操作，因此需对不符合编码规则的测试个体进行处理，具体操作为：若经交叉操作的测试个体不符合编码规则，则重新进行变异、交叉两项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，以实现种群的多样性。根据式(7)生成测试向量。

$$u_{ji,G+1}=\begin{cases}v_{ji,G+1}, & \text{if } rand(0,1)\leqslant C_R \text{ or } j=j_{rand}\\ x_{ji,G}, & \text{otherwise}\end{cases}\tag{7}$$

其中，$u_{ji,G+1}$ 表示第 $G+1$ 代种群中的第 i 个测试个体的第 j 维，$C_R\in[0,1]$ 为交叉概率，j_{rand} 为 0 到 D 之间的随机整数，D 为问题的维数。

2.4 选择操作

采用贪婪法选择测试个体是否会成为下一代种群中的个体，通过比较测试个体与当前种群中目标个体的适应值来选择最优个体。下一代种群中的所有个体都优于或等于当前种群中对应的个体。具体选择操作如式(8)所示：

$$x_{i,G+1}=\begin{cases}u_{i,G}, & \text{if } f(u_{i,G})\leqslant f(x_{i,G})\\ x_{i,G}, & \text{otherwise}\end{cases}\tag{8}$$

其中，$x_{i,G+1}$ 表示第 $G+1$ 代种群的第 i 个个体，$u_{i,G}$ 和 $x_{i,G}$ 分别表示第 G 代种群中第 i 个变异个体和第 G 代种群的第 i 个个体，$f(x)$ 是目标函数。

3 数值仿真

为了验证本文所提算法的性能，首先基于测试函数验证其有效性，然后对实例进行排产优化调度。

3.1 算法验证

为了验证所提算法的性能，选取 SaDE[21]，JADE[22] 和 CoDE[23] 算法进行比较分析。实验中，采用 10 个常用的标准测试函数进行测试[24]，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_5 为单模函数，f_6-f_{10} 为多模函数。

表 1 标准测试函数

Table 1 Benchmark functions

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
SumSquares	$f_2(x)=\sum_{i=1}^{N}ix_i^2$	$[-10,10]$	0
Schwefel2.22	$f_3(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]$	0
Table	$f_4(x)=10^6x_1^2+\sum_{i=2}^{n}x_i^2$	$[-100,100]$	0
Step	$f_5(x)=\sum_{i=1}^{n}\lfloor x_i+0.5\rfloor^2$	$[-100,100]$	0
Schaffer 2	$f_6(x)=\sum_{i=1}^{N-1}(x_i^2+x_{i+1}^2)^{0.25}(\sin^2(50(x_i^2+x_{i+1}^2)^{0.1})+1)$	$[-100,100]$	0
Levy and Montalvo 1	$f_7(x)=\pi(10\sin^2(\pi y_1)/n+\sum_{i=1}^{n-1}(y_i-1)^2(1+10\sin^2(\pi y_i+1)))+(y_n-1)^2,y_i=1+0.25(x_i+1)$	$[-10,10]$	0
Levy and Montalvo 2	$f_8(x)=0.1\{\sin^2(3\pi x_i)+\sum_{i=1}^{n-1}(x_i-1)^2(1+\sin^2(3\pi x_i+1))+(x_n-1)^2(1+\sin^2(2\pi x_n))\}$	$[-2,2]$	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
Penalized 1	$f_{10}(x)=\frac{\pi}{D}\{10\sin^2(\pi yi)+10\sin^2(\pi yi)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_D-1)^2\}+\sum_{i=1}^{D}u(x_i,10,100,4)$	$[-50,50]$	0

本文所提算法的参数设置如下：种群规模 $N_p=50$，增益常数 F 和交叉常数 CR 按照 SaDE[21] 的自适应方式进行设置，SaDE 算法、JADE 算法和 CoDE 算法的参数均按照原文设置。为了比较的公平性，每种算法对每个测试函数均独立运行 30 次。

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

首先，选择函数评价次数(FES)和成功率(SR)两个指标来验证所提算法的收敛速度、计算代价和可靠性。FES 为算法在设定的最大目标函数评价次数内达到设定的精度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的比值。规定算法在最大函数评价次数内能达到设定的精度为成功。本次实验中，设置最大函数评价次数为 300000，精度为 0.00001。

表 2 列出了各函数的函数评价次数 FES 和成功率 SR。其中，加粗数据为最优结果。从表 2 中的数据可以看出，除了函数 f_6 以外，所提算法对于其他函数的收敛速度均优于其他算法，且能够以 100% 的成功率进行求解；其次，从表 2 中最后一行的总平均结果可以看出，所提算法所需的目标函数评价次数最少，为 20974，而 SaDE 算法、JADE 算法和 CoDE 算法所需的函数评价次数分别为 23710，48690 和 50740，即所提算法相对于 SaDE 算法、JADE 算法和 CoDE 算法分别节省了 11.5%，56.9% 和 58.7% 的函数评价次数。在成功率方面，所提算法、JADE 算法、CoDE 算法均为 1.00，但 SaDE 算法由于对 1 个函数没有成功求解，成功率为0.99。

表 2　函数评价次数和成功率

Table 2　Function evaluations and success rates

函数	维数	SaDE		JADE		CoDE		DSDE	
		FES	*SR*	*FES*	*SR*	*FES*	*SR*	*FES*	*SR*
f_1	30	20297	1.00	23557	1.00	40155	1.00	**16058**	1.00
f_2	30	18410	1.00	21600	1.00	36270	1.00	**15778**	1.00
f_3	30	24368	1.00	35787	1.00	56421	1.00	**18454**	1.00
f_4	30	21117	1.00	27007	1.00	41286	1.00	**19146**	1.00
f_5	30	10657	1.00	11790	1.00	20070	1.00	**8376**	1.00
f_6	30	**71833**	1.00	279660	1.00	172776	1.00	75531	1.00
f_7	30	12260	1.00	17340	1.00	26307	1.00	**10036**	1.00
f_8	30	12855	1.00	17373	1.00	26964	1.00	**10656**	1.00
f_9	30	30453	0.93	33080	1.00	56826	1.00	**22112**	1.00
f_{10}	30	14852	1.00	19710	1.00	30324	1.00	**13596**	1.00
平均值		23710	0.99	48690	1.00	50740	1.00	**20974**	1.00

为了验证算法所求得的解的质量，选用平均函数误差值(Mean error)和标准偏差值(Std dev)两个指标进行衡量。选用 Wilcoxon Signed Rank Test 对各算法 30 次优化得到的结果进行非参数假设检验，以验证所提算法相对于其他算法是否具有显著性优势，显著性水平设置为 5%。表 3 列出了各测试函数运行 30 次的平均函数误差和标准偏差，其中，“+”表示所提算法显著优于其他算法，“−”表示所提算法显著差于其他算法，“≈”表示两种算法间没有显著性差异。从表 3 可以看出，除了函数 f_2，f_5 和 f_6 外，所提算法的结果均优于其他算法。对于函数 f_5，SaDE 算法、JADE 算法、CoDE 算法以及所提算法均得到了全局最优解。从表 3 中最后一行的非参数检验结果可以看出，与 SaDE 算法、JADE 算法和 CoDE 算法相比，所提算法在 7 个函数上显著优于这 3 种算法，JADE 算法和 CoDE 算法没有在任何函数上显著优于所提算法，SaDE 算法在 2 个函数上显著优于所提算法。测试结果表明，所提算法收敛速度快、成功率高，且解的质量较高。

表 3　优化结果性能

Table 3　Performance of optimization results

函数	维数	SaDE	JADE	CoDE	DSDE
		Mean error(Std dev)	Mean error(Std dev)	Mean error(Std dev)	Mean error(Std dev)
f_1	30	1.29E−23(3.07E−23)+	3.01E−20(8.74E−20)+	2.16E−10(1.73E−10)+	**3.93E−29(1.68E−28)**
f_2	30	**6.59E−25(7.06E−04)−**	1.16E−21(1.59E−21)+	2.83E−11(2.61E−11)+	7.93E−24(1.81E−23)
f_3	30	8.70E−16(5.17E−16)+	3.11E−10(3.36E−10)+	3.86E−06(1.32E−06)+	**2.15E−19(7.47E−19)**
f_4	30	8.85E−23(4.35E−22)+	1.83E−18(5.03E−04)+	4.04E−10(3.04E−10)+	**1.98E−23(5.18E−23)**
f_5	30	**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_6	30	**7.51E−04(7.06E−04)−**	3.24E+00(1.23E+00)+	1.97E+00(4.24E−01)+	3.06E−03(4.28E−03)
f_7	30	2.47E−27(4.09E−27)+	3.69E−21(1.55E−20)+	1.80E−13(3.36E−13)+	**1.95E−30(3.90E−30)**
f_8	30	1.46E−03(3.80E−03)+	5.07E−22(2.22E−21)+	1.31E−13(1.51E−13)+	**6.73E−30(1.79E−29)**
f_9	30	2.40E−01(4.45E−01)+	4.19E−11(4.49E−11)+	3.75E−06(1.88E−06)+	**5.84E−15(1.77E−15)**
f_{10}	30	3.46E−03(1.89E−03)+	5.65E−21(1.54E−20)+	1.44E−12(1.26E−12)+	**2.44E−28(5.46E−28)**
+/≈/−		7/2/1	9/1/0	9/1/0	

3.2　案例分析

为了测试将本文所提算法应用于 FJSP 算法时的优化结果，采用文献[25]中的 4×6 测试案例及文献[26]中的 8×8 测试案例，以最大完工时间为性能指标进行仿真对比，测试案例的最优结果分别为 17 和 14。

表 4、表 5 列出了所用测试案例的信息。从表中可以看出，4×6 测试案例总的加工工序为 12 道，8×8 测试案例总的加工工序为 27 道，每道工序有其对应的机器加工集。按照上述工序及机器双层编码方式对所提算法进行实例仿真。为了验证所提算法的优势，选取 2 种先进的改进差分进化算法 SaDE[21] 和 CoDE[22] 进行比较。DSDE 算法的参数设置与第 4 节相同，且各算法对于上述问题独立仿真 10 次，终止条件均

为 3000 次函数评价。最优调度方案的仿真结果的甘特图如图 1、图 2 所示。以函数评价次数为横轴，最大完成时间为纵轴，绘制各算法的 10 次仿真的平均收敛曲线，如图 3、图 4 所示。所有算法均可达到最优，但本文提出的 DSDE 算法的收敛速度明显快于其他算法。

表 4 4×6 FJSP 测试案例

Table 4 4×6 FJSP benchmark function

工件及工序		加工设备及加工时间					
		1	2	3	4	5	6
J_1	O_{11}	2	3	4			
	O_{12}		3		2	4	
	O_{13}	1	4	5			
J_2	O_{21}	3		5		2	
	O_{22}	4	3			6	
	O_{23}			4		7	11
J_3	O_{31}	5	6				
	O_{32}		4		3	5	
	O_{33}			13		9	12
J_4	O_{41}	9		7	9		
	O_{42}		6		4		5
	O_{43}	1		3			3

表 5 8×8 FJSP 测试案例

Table 5 8×8 FJSP benchmark function

工件及工序		加工设备及加工时间							
		1	2	3	4	5	6	7	8
J_1	O_{11}	5	3	5	3	3		10	9
	O_{12}	10		5	8	3	9	9	6
	O_{13}		10		5	6	2	4	5
J_2	O_{21}	5	7	3	9	8		9	
	O_{22}		8	5	2	6	7	10	9
	O_{23}		10		5	6	4	1	7
	O_{24}	10	8	9	6	4	7		
J_3	O_{31}	10			7	6	5	2	4
	O_{32}		10	6	4	8	9	10	
	O_{33}	1	4	5	6		10		7
J_4	O_{41}	3	1	6	5	9	7	8	4
	O_{42}	12	11	7	8	10	5	6	9
	O_{43}	4	6	2	10	3	9	5	7
J_5	O_{51}	3	6	7	8	9		10	
	O_{52}	10		7	4	9	8	6	
	O_{53}		9	8	7	4	2	7	
	O_{54}	11	9		6	7	5	3	6
J_6	O_{61}	6	7	1	4	6	9		10
	O_{62}	11		9	9	9	7	6	4
	O_{63}	10	5	9	10	11		10	
J_7	O_{71}	5	4	2	6	7		10	
	O_{72}		9		9	11	9	10	5
	O_{73}		8	9	3	8	6		10
J_8	O_{81}	2	8	5	9		4		10
	O_{82}	7	4	7	8	9		10	
	O_{83}	9	9		8	5	6	7	1
	O_{84}	9		3	7	1	5	8	

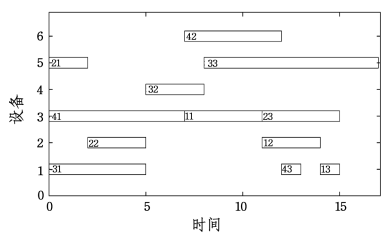

图 1 4×6 问题最优调度方案甘特图

Fig. 1 Optimal scheduling scheme Gantt diagram of 4×6 benchmark function

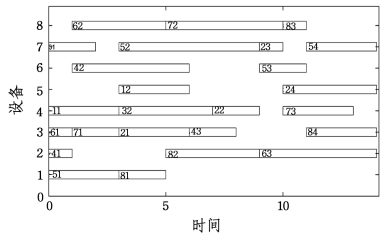

图 2 8×8 问题最优调度方案甘特图

Fig. 2 Optimal scheduling scheme Gantt diagram of 8×8 benchmark function

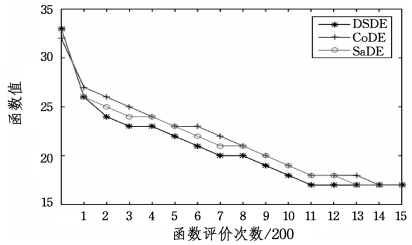

图 3 4×6 问题仿真收敛图

Fig. 3 Simulation convergence diagram of 4×6 benchmark function

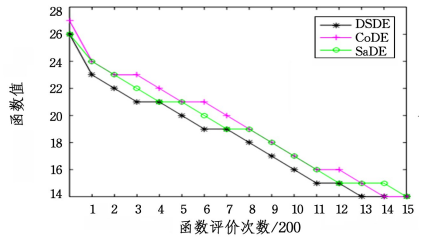

图 4 8×8 问题仿真收敛图

Fig. 4 Simulation convergence diagram of 8×8 benchmark function

结束语 本文对柔性作业车间调度问题进行研究，在满足工艺要求的前提下建立了数学模型。针对车间调度问题的特点采用基于工序和机器的双层编码方式，将离散的问题转化为差分进化能够处理的连续问题，并提出了一种基于动态策略的差分进化算法，根据种群的拥挤度变化选择合适的变异策略来指导算法搜索，进而对调度模型进行快速求解。仿真实验和系统排产调度甘特图表明了算法的可行性与有效性。

参 考 文 献

[1] WANG L, DENG J, WANG S Y. Survey on optimization algorithms for distributed shop scheduling[J]. Control and Decision, 2016, 31(1): 1-11. (in Chinese)
王凌，邓瑾，王圣尧. 分布式车间调度优化算法研究综述[J]. 控制与决策，2016，31(1)：1-11.

[2] LI C B, SHEN H, LI L L, et al. A Batch Splitting Flexible Job Shop Scheduling Model for Energy Saving under Alternative Process Plans[J]. Journal of Mechanical Engineering, 2017, 53(5): 12-23. (in Chinese)
李聪波，沈欢，李玲玲，等. 面向能耗的多工艺路线柔性作业车间分批优化调度模型[J]. 机械工程学报，2017，53(5)：12-23.

[3] STORN R, PRICE K. Differential evolution: a simple and ef-

ficient heuristic for global optimization over continuous spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.

[4] LI J Q, PAN Q K, LIANG Y C. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems[J]. Computers & Industrial Engineering, 2010, 59(4): 647-662.

[5] ZHANG J, WANG W L, XU X L, et al. Improved particle swarm algorithm for batch splitting flexible job shop scheduling [J]. Control and Decision, 2012, 27(4): 513-518. (in Chinese)
张静，王万良，徐新黎，等. 基于改进粒子群算法求解柔性作业车间批量调度问题[J]. 控制与决策，2012，27(4)：513-518.

[6] T'KINDT V, MONMARCHÉ N, TERCINET F, et al. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem[J]. European Journal of Operational Research, 2002, 142(2): 250-257.

[7] AN Y W, YAN H S. Solution Strategy of Integrated Optimization of Production Planning and Scheduling in a Flexible Job-shop[J]. Acta Automatica Sinic, 2013, 39(9): 1476-1491. (in Chinese)
安玉伟，严洪森. 柔性作业车间生产计划与调度集成优化求解策略[J]. 自动化学报，2013，39(9)：1476-1491.

[8] GU J W, GU M Z, CAO C W, et al. A novel competitive co-evolutionary quantum genetic algorithm for stochastic Job Shop scheduling problem [J]. Computers and Operations Research, 2010, 37(5): 927-937.

[9] CAI L W, ZHANG J H, LI X. A Multi-Population Genetic Algorithm for Job Shop Scheduling Problem[J]. Acta Electronica Sinica, 2005, 33(6): 991-994. (in Chinese)
蔡良伟，张基宏，李霞. 作业车间调度问题的多种群遗传算法[J]. 电子学报，2005，33(6)：991-994.

[10] WANG L, ZHOU G, XU Y, et al. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem[J]. International Journal of Advanced Manufacturing Technology, 2012, 60(1-4): 303-315.

[11] WANG L, WANG S, XU Y, et al. A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem[J]. Computers & Industrial Engineering, 2012, 62(4): 917-926.

[12] YAZDANI M, AMIRI M, ZANDIEH M. Flexible job-shop scheduling with parallel variable neighborhood search algorithm [J]. Expert Systems with Applications, 2010, 37(1): 678-687.

[13] CHEN H, IHLOW J, LEHMANN C. A genetic algorithm for flexible job-shop scheduling[C]// 1999 IEEE International Conference on Robotics and Automation. IEEE, 1999: 1120-1125.

[14] YANG X M, ZENG J C. Solving flexible job shop scheduling problem using genetic algorithm[J]. Control and Decision, 2004, 19(10): 1197-1200. (in Chinese)
杨晓梅，曾建潮. 遗传算法求解柔性 job shop 调度问题[J]. 控制与决策，2004，19(10)：1197-1200.

[15] LI Z W, ZHOU X G, ZHANG G J. Dynamic Adaptive Differential Evolution Algorithm[J]. Computer Science, 2015, 42(S1): 52-56, 74. (in Chinese)
李章维，周晓根，张贵军. 一种动态自适应差分进化算法[J]. 计算机科学，2015，42(S1)：52-56，74.

[16] ZHOU X G, ZHANG G J, HAO X H, et al. Differentail Evolution Algorithm Based on Local Lipschitz Underestimate Supporting Hyperplanes[J]. Acta Automatic Sinica, 2015, 41(7): 1315-1327. (in Chinese)
周晓根，张贵军，郝小虎，等. 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.

[17] 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.

[18] WANG F, TANG Q H, RAO Y Q, et al. Effcient Estimation of Distribution for Flexible Hybrid Flow Shop Scheduling[J]. Acta Automatica Sinica, 2017, 43(2): 280-293. (in Chinese)
王芳，唐秋华，饶运清，等. 求解柔性流水车间调度问题的高效分布估算算法[J]. 自动化学报，2017，43(2)：280-293.

[19] ZHOU X G, ZHANG G J. Abstract convex underestimation assisted multistage differential devolution[J]. IEEE Transactions on Cybernetics, 2017, 47(9): 2730-2741.

[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] ZHANG J, SANDERSON A C. JADE: adaptive differential evolution with optional external archive[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 945-957.

[23] 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.

[24] ZHOU X G, ZHANG G J, HAO X H, et al. Differentail Evolution Algorithm Based on Local Lipschitz Underestimate Supporting Hyperplanes[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.

[25] WANG W L, ZHAO C, XIONG J, et al. Method to Resolve Flexible Job-shop Scheduling Problem Based on Improved Ant Colony Algorithm [J]. Journal of System Simulation, 2008, 20(16): 4326-4329. (in Chinese)
王万良，赵澄，熊婧，等. 基于改进蚁群算法的柔性作业车间调度问题的求解方法[J]. 系统仿真学报，2008，20(16)：4326-4329.

[26] KACEM I, HAMMADI S, BORNE P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems[J]. IEEE Transactions on Systems Man and Cybernetics Part C－Applications and Reviews, 2002, 32(1): 1-13.