求解作业车间调度问题的微粒群遗传退火算法

毛 帆,傅 鹂,蔡 斌
MAO Fan,FU Li,CAI Bin
重庆大学 软件工程学院,重庆 400044
School of Software Engineering,Chongqing University,Chongqing 400044,China

MAO Fan,FU Li,CAI Bin.Particle swarm genetic annealing algorithm for job-shop scheduling.Computer Engineering and Applications,2011,47(5):227-231.

Abstract: The Standard Particle Optimization algorithm(PSO) generally is used to solve continuous optimization problems, and is used rarely to solve discrete problems such as Job-shop Scheduling Problem(JSP).So,based on the problem of premature convergence and slow search speed of PSO,this paper proposes a hybrid particle swarm optimization algorithm to solve JSP.By combining PSO,Genetic Algorithm(GA) and Simulated Annealing(SA) algorithm,this algorithm not only enhances the global search ability,but also reduces the algorithm's dependence on the parameters,and at the same time,overtakes the premature convergence of GA and PSO algorithm.The experimental results indicate that this algorithm not only has great advantage of convergence property over PSO,but also can avoid the premature convergence problem effectively.

Key words: particle swarm optimization algorithm;genetic algorithm;job-shop scheduling;simulated annealing

摘 要: 标准微粒群算法(PSO)通常被用于求解连续优化的问题,很少被用于离散问题的优化求解,如作业车间调度问题(JSP)。因此,针对PSO算法易早熟、收敛慢等缺点提出一种求解作业车间调度问题(JSP)的混合微粒群算法。算法将微粒群算法、遗传算法(GA)、模拟退火(SA)算法相结合,既增强了算法的局部搜索能力,降低了算法对参数的依赖,同时改善了PSO算法和GA算法易早熟的缺点。对经典JSP问题的仿真实验表明:与标准微粒群算法相比,该算法不仅能有效避免算法中的早熟问题,并且算法的全局收敛性得到了显著提高。

关键词: 微粒群算法;遗传算法;作业车间调度;模拟退火
DOI:10.3778/j.issn.1002-8331.2011.05.068 **文章编号:** 1002-8331(2011)05-0227-05 **文献标识码:** A **中图分类号:** TP391

1 引言

作业车间调度问题[1](Job-shop Scheduling Problem,JSP)是一类及其复杂的调度问题,是许多实际生产调度问题的简化模型;作为生产制造系统中一个热门的研究领域,JSP一直备受调度领域的重视。其主要目的是在已知的资源和工艺约束条件下,确定工件加工的顺序和时间。作为一组复杂的组合优化问题,JSP典型的NP-hard问题[1],因此,JSP常用来测试智能算法的性能。这在研究和实际工程领域中具有重要的意义。

在JSP的求解上,目前主要的方法有:分支定界法、启发式算法、模拟退火算法(SA)、禁忌搜索算法(TS)、蚁群算法(AS)、免疫算法(IA)、神经网络算法(NN)、遗传算法(GA)等。分支定界法对小规模问题比较有效,但对大规模问题则难以接受;启发式算法能够快速建立问题的解,但通常质量较差;其他的各种邻域搜索算法和人工智能方法也都存在各自的优缺点。

微粒群优化(Particle Swarm Optimization,PSO)算法[1]作为一类针对连续函数优化问题的优化算法,在JSP上的研究相当少。目前,PSO的JSP的代表性研究有:Tu等提出的用于求解传统JSP的拓补PSO算法[2];Liu等将PSO算法和VNS相结合提出的求解柔性JSP的混合算法[3];Xia和Wu将SA算法嵌入PSO算法而提出的求解JSP和多目标柔性JSP的混合算法[4-5];Sha和Hsu用Giffler-Thompson启发式算法,并结合禁忌搜索而提出的针对最小化最大完工时间的混合PSO算法[6]等。

本文针对在求解JSP问题遇到的资源和工艺约束等问题,对PSO算法中种群位置和JSP之间的关系以及现有的JSP调度算法[7]进行分析,结合遗传算法[8](Genetic Algorithm,GA)中染色体交叉变异的特性,提出一种结合了模拟退火算法[9]、遗传算法、以及微粒群算法的微粒群遗传退火算法(Particle Swarm Genetic Annealing Algorithm,PSGAA)。

2 JSP问题描述

作业车间调度问题是研究n个工件在m台机器上加工的过程。已知条件有:工件集$J=\{j_i \mid i=(1,2,\cdots,n)\}$,$j_i=(1,2,\cdots,m)$表示工件$i$的加工顺序(工艺约束);机器集$M=\{m_i \mid i=(1,$

基金项目: 重庆市科委科技计划攻关项目(No.102074920080018)。

作者简介: 毛帆(1985—),男,硕士生,主要研究方向:人工智能,控制技术;傅鹂(1961—),男,教授,主要研究方向:控制技术,算法设计及优化计算;蔡斌(1979—),男,博士生,讲师,主要研究方向:生产管理及控制。E-mail:lovelycatty319@163.com

收稿日期: 2009-06-25;**修回日期:** 2009-09-09;CNKI出版:2011-1-27;http://www.cnki.net/kcms/detail/11.2127.TP.20110127.1042.201105.227_499.html

2,…,m)}；时间$T_{i,j}$为第i个工件p_i的第j道工序所用的时间；称一个工件在一台机器上的加工为一次操作，则操作集O={o_i | i=(1,2,…,n)}，o_i表示工件i在所有机器上的加工，o_i={o_{ij} |j=(1,2,…,m)}；工件每次加工所需时间T_{ij}与每一个o_{ij}相关联。求解的问题是：在已知资源和技术约束条件下确定工件在每台机器上的加工顺序或是完工时间$C_{\max}$。

求解JSP时，工件加工必须满足如下约束条件：

(1)既定工件加工工序顺序不能改变；

(2)在某一时刻一台机器上最多只能加工一个工件；

(3)当工件在机器上加工时，只有在当前操作完成后，工件才能离开机器；

(4)在某时刻工件只能在一台机器上加工。

3 PSGAA算法

3.1 PSGAA算法总体描述

PSO算法，是基于群体智能理论“进化”计算技术，具有全局寻优的特点，但容易陷入局部最优解。SA算法作为一种基于Mente Carlo迭代求解策略的随机寻优算法，具有概率突跳的能力，能有效避免搜索过程陷入局部最优解。GA算法以其编码和遗传原理简单易懂、适应能力强、搜索面广、算法收敛速度快等优点，在JSP以及其他优化调度中得到了广泛的应用。

本算法同时继承了这三个算法的优点，使得本算法既能在全局范围内寻找问题最优解，又能避免算法在搜索过程中陷入局部最优解。同时使用Giffler-Thompson启发式算法对微粒位置进行初始化，使其尽可能使微粒散落在可行域内，增加了算法的搜索效率。

为了保证整个搜索过程是在全局范围内进行、并且避免搜索过程陷入局部最优解、同时要确保搜索的效率和准确性。算法主要分为四个步骤。

第一步：种群寻优(swarm optimizing)

由于PSO算法通过将微粒的位置引入目标函数来评价当前微粒。因此，在本步骤中，算法首先使用目标函数对种群中所有微粒进行评价，找出每个微粒的最佳位置。然后从所有的最佳位置中找出使得目标函数值最优的微粒，记为种群最优位置。

接下来更新微粒的位置，按如上方法再次查找并更新微粒的最佳位置和群体最佳位置，直到满足算法终止条件。

第二步：SA邻域搜索(SA-neighborhood search)

将微粒的最佳位置做为SA算法初始值，利用SA算法生成当前最佳位置的邻域解，并计算用此邻域解代替当前最佳位置的邻域替代概率。

连续进行L次邻域搜索。然后使用轮盘赌策略从L个邻域值中选择一个值来代替当前微粒的最佳位置。重复此过程，直到对所有微粒都进行了SA邻域搜索。

第三步：遗传更新

用遗传算法代替微粒群微粒速度和位置更新，先通过选择、交叉操作来更新产生一个新的微粒个体，然后根据个体适应度与种群平均适应度动态调整变异概率P_m，并以概率P_m对微粒变异，然后更新微粒位置。

第四步：最优选择(optimization choice)

对第二步中所有微粒的最佳位置使用轮盘赌策略，选出一个作为整个种群的最优值gbest。

算法的终止条件为：种群达到最大迭代次数。或群体最佳位置k次保持不变。

3.2 算法相关定义描述

定义1 违反约束的投影。若对微粒位置在当前搜索空间上某个维度上的投影进行解码后，此投影不满足作业车间调度的约束条件，则称此投影为当前搜索空间上违反约束的投影。

定义2 不可行解约束违反量函数。用来反应某个微粒违反约束的投影的个数。

定义3 不可行解。若某个微粒的位置经解码后不满足当前问题的约束条件，则称此微粒的位置为不可行解。

定义4 温度衰减系数。在SA算法中，温度的控制通过温度衰变函数$T_{k+1}=\mu*T_k$进行，其中的μ称为温度衰减系数。μ通常为一个常数。

定义5 微粒最佳位置(pbest)。若当前微粒在当前时刻t的目标函数值优于该微粒在$t1$($t1<t$)时刻的目标值，则t时刻的微粒位置为微粒的最佳位置。

定义6 群体最佳位置(gbest)。所有微粒的pbest中，使适应度函数最优的pbest为群体最佳位置。

定义7 邻域解。在进行SA邻域搜索时在某个值的邻域范围内产生新解。计算方法如下：

$$X=p_i+\eta\times(x_{\max}-x_{\min})\times N(0,1)$$

其中，p_i为当前微粒的pbest，η为搜索步长，$N(0,1)$为服从均值为0方差为1的高斯分布的随机数，$x_{\max}$和$x_{\min}$分别为当前微粒在搜索空间上投影的最大和最小值。

定义8 邻域替代概率。用于描述用邻域解代替当前微粒最佳位置的概率。概率计算方法如下：

$$p_c=\frac{e^{-\frac{f(p_i)-f(p_g)}{t}}}{\sum_{j=1}^{N}e^{-\frac{f(p_j)-f(p_g)}{t}}}$$

其中N表示整个最优选择集中微粒的个数，p_i表示微粒i的pbest，p_g表示种群的gbest。

定义9 假微粒。当微粒在迭代中产生了不可行解时，用于代替此微粒的微粒。假微粒生成方式如下：

$$X=pbest+r\times(pbest-x_i+v_i)\times N(0,1)$$

其中r表示在0和1之间的随机分布数，$N(0,1)$为服从均值为0方差为1的高斯分布的随机数，x_i表示当前微粒的位置，v_i表示微粒当前的速度。

定义10 不可行解替换概率。用于表示产生不可行解的微粒被相应的假微粒替换的概率。$p=\min\{1,e^{-(f(pbest)-f(x))/t}\}$。

3.3 算法详细描述

3.3.1 种群寻优

算法参数引入：PSO涉及的参数，种群大小$size$，算法迭代次数N，微粒加速常数c_1和c_2；SA涉及的参数，初始温度衰减系数β，最大温度衰减系数β_1，最小温度衰变系数β_2，SA邻域搜索次数L；GA涉及的参数，交叉概率P_c，变异概率P_m。

步骤1 根据G&T启发式算法初始化微粒的位置和速度；

步骤2 计算种群中每个微粒的适应值；

步骤3 通过适应值更新微粒的pbest和gbest；

步骤4 判断是否满足算法终止条件，若是则转步骤5，否则继续执行如下步骤：

步骤4.1　对微粒的pbest进行SA邻域搜索;

步骤4.2　更新各微粒的pbest;

步骤4.3　微粒的最优选择操作,并更新种群的gbest;

步骤4.4　根据微粒的gbest和pbest更新微粒的速度和位置;

步骤4.5　微粒最优选择,转步骤4;

步骤5　输出种群最优解。

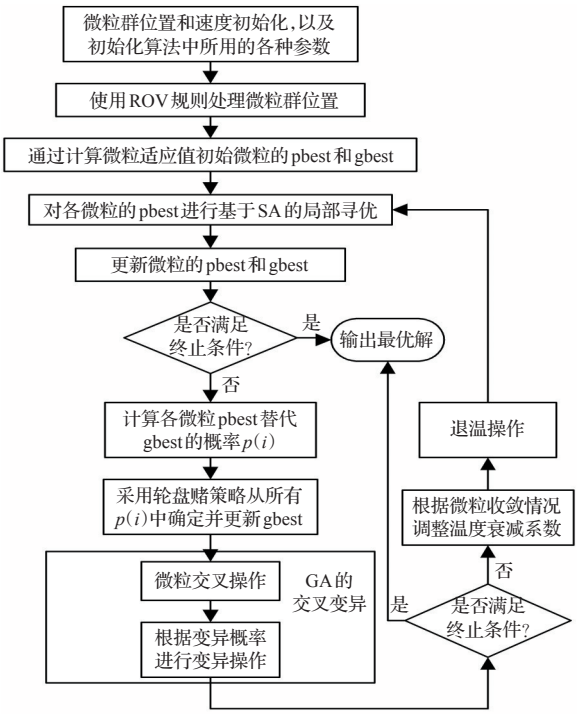

图1　PSGAA**算法流程图**

3.3.2　SA邻域搜索

进行SA邻域搜索时,算法将微粒的pbest作为SA算法初始解,在pbest的基础上进行邻域搜索。搜索步骤如下:

步骤1　接收微粒的pbest作为SA的初始解。

步骤2　按如下公式初始化算法初始温度$t_0=-f_{\text{gbest}}/\ln(0.2)$。

步骤3　对下列步骤执行L次。

步骤3.1　在pbest邻域内生成邻域解。

步骤3.2　由$T_{k+1}=\beta*T_k$计算当前温度T_{k+1}。

步骤3.3　计算用邻域解代替pbest的概率。

步骤4　根据步骤3.3得到的概率,使用轮盘赌策略在微粒的L个邻域解中选择一个值用于代替当前微粒的pbest。

3.3.3　遗传更新

传统的变异过程中,每一代被选中的个体只执行一次基因位交换的变异,因此种群的多样性被降低,为了提高种群多样性,本算法用SA算法对变异算子进行改进,即对每个被选中的个体均进行变异并进行SA局部搜索。

在选择交叉算子时,选择种群最佳位置gbest、当前微粒最佳位置pbest和微粒当前位置这三个部分。交叉时,首先对gbest和pbest进行交叉,然后再将交叉结果与微粒当前位置进行交叉,这样的交叉方式既保存了微粒当前状态、拥有"记忆"能力、也拥有信息共享的能力。

为了增强GA和SA算法的相互联系,增加了反映算法收敛性的操作,自适应地调整退火算法的温度衰减系数。根据当前微粒的收敛情况加强或减少对微粒搜索的深度,减少搜索时间。

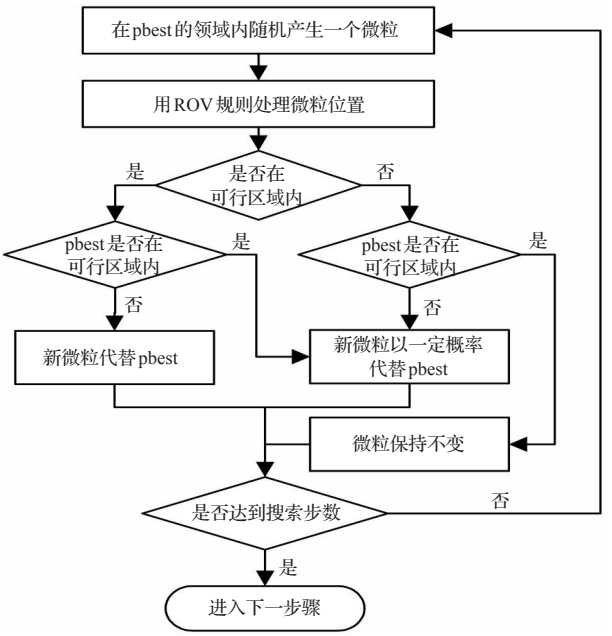

图2　SA**邻域搜索流程图**

3.3.3.1　选择交叉微粒

为了维持微粒群记忆、信息共享和保存当前状态的能力。在为遗传算法选择交叉微粒时,算法交叉算子为群体最优微粒位置、当前微粒最佳位置和当前微粒位置三部分。

3.3.3.2　交叉

由于JSP问题的特殊性,其编码的交叉操作受到工艺加工规则的约束,本文采用基于Giffler-Thompson启发式算法的交叉方法[4]进行算子的交叉操作。由于该算法是一种树搜索算法,因此,所生成的调度中至少包含一个可行解。

具体步骤如下:

步骤1　设S为所有工件当前可加工工序的集合;

步骤2　确定$c(o_k)=\min\{c(o_{ij})|o_{ij}\in S\}$和进行该相应操作$(o_k)$的机器$m_k$,其中$o_{ij}$表示工件$i$的第$j$个操作;

步骤3　设Gm_k为所有要求在机器m_k上加工的操作o_{ij}的集合,其中$o_{ij}\in S$;

步骤4　从集合Gm_k中按如下步骤选择一个操作:

步骤4.1　产生一个在(0,1)间的随机数r,若$r<p_m$,则在Gm_k中任选一个操作,记为$o_{ij}(k)$其中p_m为变异概率;

步骤4.2　否则,以相同概率选择两个父代中的一个,记为Ps;在Ps所有操作中找到在Gm_k中最早被调度的操作,记为$o_{ij}(k)$;

步骤4.3　在后代个体中根据$c(o_{ij})$调度$o_{ij}(k)$;

步骤5　更新S,首先从S中删除操作$o_{ij}(k)$,并在S中加入其下道工序;

步骤6　若生成完整的调度方案则结束;否则返回步骤2。

3.3.3.3　变异与变异概率

变异操作的主要目的是通过改变染色体上的某些基因片段来引入新个体,以增加种群多样性。

常用的遗传算法中,问题的编码串的变异概率在整个算法过程中是固定不变的,从而使得算法求解过程过长,且容易陷入局部最优值。

针对这种情况,本算法对适应值劣于平均适应值的微粒提高其变异概率,以便使其通过变异产生更好的解,并提高种群的多样性;对适应值优于平均适应值的微粒抑制其变异概

率,以提高算法计算速度。

变异概率计算如下:

$$p_{mi}(k)=\begin{cases}P_m, f\geq f_{avg}\\ P_m\left(1+\exp(\mu\frac{f_{avg}-f_i}{f_{avg}})\exp(-k)\right), f\leq f_{avg}\end{cases}$$

$p_{mi}(k)$为第k次迭代式第i个微粒X_i的变异概率;

f_{avg}表示种群的平均适应值;

f_i表示第i个微粒的适应值;

k为迭代次数;

p_m为初始化的变异概率。

3.3.4 最优选择

接收种群的pbest及gbest作为候选集。从中选出一个值用于更新gbest,并作为算法最优解。

步骤1 对候选集中的每个微粒计算微粒最优选择概率。

步骤2 根据步骤1中计算的各选择概率,执行S次轮盘赌策略,执行方式如下:

步骤2.1 前$S-1$次轮盘赌策略从候选集中选出$S-1$个pbest作为最优候选集;

步骤2.2 第S轮盘赌策略从最优候选集中选出一个pbest作为种群的gbest。

步骤3 输出gbest。

3.3.5 适应度函数

在PSO算法中,pbest和gbest的选取主要通过比较适应度函数来进行优胜劣汰的选择,并且,它们对微粒速度和位置的更新起着至关重要的作用。虽然用遗传算法的交叉和变异操作代替了PSO算法中微粒速度和位置的更新,但仍然要通过比较适应度函数来进行优胜劣汰的选择。适应度越好的微粒被选择的概率也就越大。由于研究的是作业车间调度的完工时间C_{max},因此,适应度函数$f(i)=C_{max}$。

$$C_{max}=\max C(i,j)\ (1\leq i\leq m, 1\leq j\leq n)$$

$$C(1,1)=C(O_{(P_{11}-1)},1)+T(p_{11},1)$$

$$C(1,j)=\max\{C(1,j-1),C(O_{(p_{1j}-1)j},j)\}+T(p_{1j},j)$$

$$C(i,1)=C(O_{(p_{i1}-1)},1)+T(p_{i1},1)$$

$$C(i,j)=\max\{C(O_{p_{ij}},j-1),C(O_{(p_{ij}-1)j},j)\}+T(p_{ij},j)$$

3.3.6 不可行解优化

在避免算法所产生的不可行解上,本算法虽然在初始化微粒时使用Giffler-Thompson启发式算法,但这只能使初始微粒尽可能有效,并不能保证微粒在整个“进化”过程中都有效。

针对产生不可行解的微粒进行如下优化:

步骤1 针对该微粒生成相应的假微粒。

步骤2 按如下公式计算不可行解替换概率p。

步骤3 用假微粒以概率p代替原微粒。

3.3.7 编码设计

考虑到编码的Lamarkian特性[10]、编码的空间特性、解码的复杂性以及存储空间等问题。本文的PSGAA算法采用基于操作的编码方式[1]。

该编码方式用$m*n$个代表操作的位置值表示微粒的位置矢量,即所有操作的一个序列,其中每个工件号均出现m次。解码过程是:先将微粒位置转化为一个有序的操作表,然后基于操作表和工艺约束将操作表中的元素按照从左到右的顺序解释为相应工件的操作顺序,进而产生调度方案。

基于操作的编码方式的基本思想是将所有工件的操作进行编码,不同工件用不同的编码表示,同一工件则用相同的编码表示。编码步骤如下:

步骤1 随机初始化微粒位置:$X_k=[x_{1,1},x_{1,2},\cdots,x_{1,n},x_{2,1},x_{2,2},\cdots,x_{2,n},\cdots,x_{m,1},x_{m,2},\cdots,x_{m,n}]$;其中$n$表示工件数,$m$表示机器数量。

步骤2 分别对X_k中的子部分$[x_{i,1},x_{i,2},\cdots,x_{i,n}]$,$i=(1,2,\cdots,m)$使用ROV规则[1]。ROV规则为:比较数组中元素值的大小,按从小到大依次将其标注为$1,2,\cdots,n$。

步骤3 将上述微粒位置经过ROV规则转换后,将微粒位置矢量映射为相应的整数,而整数则可用来代表相应的工件序号。操作则可用其对应的工件号来表示,操作可用符号O_{ijk}表示,表示第i个工件的第j道工序在第k台机器上加工。

如,n=3,m=2时,设微粒初始化位置为:

X_k=[1.32 1.16 2.10 2.13 1.83 1.56]

ROV变换之后,则X_k=[2 1 3 3 2 1];工艺约束和加工时间如表1所示,则该微粒位置对应的工件加工顺序为$[O_{212}O_{111}O_{312}O_{321}O_{221}O_{122}]$。

表1 加工时间和工艺约束表

项目	工件	操作序列	
		1	2
操作时间	J_1	3	3
	J_2	1	5
	J_3	3	2
机器	J_1	M_1	M_2
	J_2	M_2	M_1
	J_3	M_2	M_1

4 实验结果分析

为了测试PSGAA算法求解JSP的性能,以Matlab7.0.1搭建仿真环境。参数选取如表2所示。

表2 PSGAA算法参数设置表

算法	参数	符号	数值
PSO	种群规模	N	50
	进化次数	G	200
	最大速度	V_{max}	1.5
	加速常数	c_1	2.1
		c_2	2.0
GA	变异概率	P_m	0.002
	交叉概率	P_c	0.4
局部SA搜索	最差解接受概率	p	0.2
	退温速率	η	0.98
	搜索步数	L	10
	搜索步长	λ	0.002

根据以上描述的算法,对表3中的问题分别进行20次数值仿真。仿真结果如表3中所示。

由表3可见,提出的PSGAA算法对表3中的各种问题均能获得最优值,且平均求解情况比其他算法要好,从而表明本文提出的PSGAA算法具有很好的搜索质量。

图3给出FT06问题的作业车间调度情况。

为了验证PSGAA算法的收敛性,将其收敛性和PSOGA算法的收敛性进行比较,比较结果如图4中所示。

由图4中的收敛性比较曲线可以看出,本文提出的PSGAA算法求解FT06问题的平均值优于PSOGA算法的平均值,且有效地避免了微粒群算法普遍存在的“早熟”现象,在提高收敛性方面也起到了明显的效果。从而保证了解的全局最优性。

表3　PSGAA、PSOGA、HPSO、GA算法比较表

问题	算法	最优解	平均解	最差解
	PSOGA	55	55.9	59
FT06	PSGAA	55	55.0	56
(6,6)	HPSO	55	55.0	55
	GA	55	56.0	58
	PSOGA	937	1 022.2	1 030
FT10	PSGAA	930	938.3	950
(10,10)	HPSO	930	947.1	959
	GA	946	950.0	958
	PSOGA	666	666.0	666
LA01	PSGAA	666	666.0	666
(10,5)	HPSO	666	666.0	666
	GA	666	668.0	673
	PSOGA	926	927.0	930
LA06	PSGAA	926	926.1	930
(15,5)	HPSO	926	927.2	935
	GA	926	927.0	934

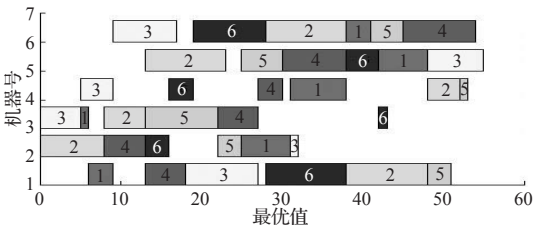

图3　FT06调度仿真实例1

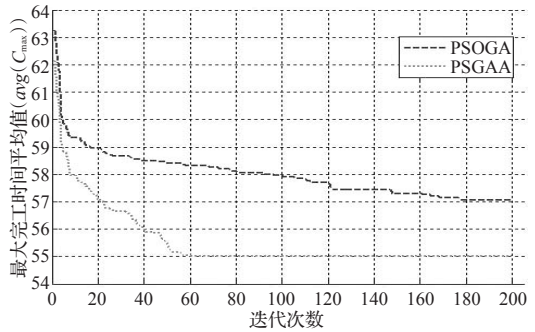

图4　PSGAA与PSOGA算法求解FT06问题的收敛性比较图

5　结束语

结合PSO、GA和SA算法，针对目前PSO算法在求解JSP问题时存在的早熟、收敛慢等缺点，提出了一种新的求解JSP问题的PSGAA算法。PSO进化为SA邻域搜索提供了一组具有较好质量和分散度的初始解，采用SA机制对这些解进行邻域搜索，不仅是对PSO搜索的补充、有利于对优良解的局部改良，同时也赋予算法概率突跳的能力。从而解决了微粒多样性差、易早熟、陷入局部最优解等缺点。同时该算法通过对不可行解的优化，使得算法能充分利用微粒资源，提高了搜索效率。用GA的交叉和变异代替PSO算法中微粒位置的更新不仅增加了种群的多样性，而且还在一定程度上避免微粒跳出可行域。从而表明：PSGAA对小规模的JSP问题是可行、有效的；并且在搜索质量和收敛性上也得到了较大提高。但是对大规模的JSP问题的有效性还没有得到证明，这也是今后研究的重点。且由于算法考虑到了搜索过程中诸如搜索的准确性、解决问题早熟、优化不可行解等问题，从而导致算法在搜索过程中耗费了许多时间，这也是将来的另一个研究重点。

参考文献：

[1] 王凌，刘波.微粒群优化与调度算法[M].北京：清华大学出版社，2008.

[2] Tu K，Hao Z F，Chen M.PSO with improved strategy and topology for job shop scheduling[C]//LNCS 4222，2006：146-155.

[3] Liu H B，Abraham A，Choi O，et al.Variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems[C]//LNCS 4247，2006：197-204.

[4] Xia W J，Wu Z M.A hybrid particle swarm optimization approach for the job-shop scheduling problem[J].Int J Adv Manuf Technol，2006，29：336-360.

[5] Xia W J，Wu Z M.An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J].Comput Ind Eng，2005，48：409-425.

[6] Sha D Y，Hau C Y.A hybrid particle swarm optimization for job shop scheduling problem[J].Ind Eng，2006，51：791-808.

[7] Jain A S，Meeran S.Deterministic job-shop scheduling：Past，present and future[J].Eur J Oper Res，1999，133：390-434.

[8] Holland J H.Outline for a logical theory of adaptive systems[J].Journal of the Association for Computing Machinery，1962，9(3)：297-314.

[9] Dekkers A，Aarts E.Global optimizations and stimulated annealing[J].Math Program，1991，50：367-393.

[10] Whitley D，Gordan V，Mathias K.Lamarckian evolution，the Baldwin effect and function optimization[C]//Parallel Solving from Nature-PPSN III.Berlin：Springer，1994：6-15.

[11] Kennedy J，Eberhart R.Particle swarm optimization[C]//IEEE International Conference on Neural Networks，Perth，Australia，1995：1942-1948.

（上接205页）

参考文献：

[1] 杨秋芬，桂卫华，胡豁生，等.驾驶员疲劳驾驶中的眼睛定位创新算法[J].计算机工程与应用，2008，44(6)：20-21.

[2] 曹菊英，赵跃龙.一种驾驶员面部识别中眼睛定位算法[J].湘南学院学报，2007，28(5)：76-77.

[3] 李贤帅，李赣华，周东翔.基于人眼定位的快速人脸检测及归一化算法[J].计算机工程与科学，2006，28(12)：63-65.

[4] Kumar T R，Raja S K，Ramakrishnan A G.Eye detection using color cues and projection functions[C]//Proc of International Conference on Image Processing，2002：337-340.

[5] Gonzalez R C，Woods R E.Digital image processing[M].阮秋琦，阮宇智，译.2版.北京：电子工业出版社，2005：152-154.

[6] 耿新，周志华，陈世福.基于混合投影函数的眼睛定位[J].软件学报，2003，14(8)：1394-1395.

[7] 陈雅茜，王玲.基于肤色和几何特性的人脸特征区域定位方法[J].计算机工程，2006，32(3)：212-213.

[8] 周长发.精通Visual C++图像处理编程[M].3版.北京：电子工业出版社，2006：280-286.