文章编号:1006-3080(2009)03-0468-07

改进协同粒子群优化算法及其在 Flow Shop 调度中的应用

虞斌能[1], 焦 斌[1], 顾幸生[2]
(1. 上海电机学院,上海 200245; 2. 华东理工大学信息科学与工程学院,上海 200237)

摘要:针对协同粒子群优化算法存在的停滞现象,提出了一种改进的协同粒子群优化算法。采用优化法的子群协作方式,既保证了收敛速率,又可以防止陷入局部最优。同时引入综合学习策略,增加种群的多样性,防止种群出现停滞现象。在此基础上,又加入了扰动机制,进一步避免算法陷入局部最优。采用该算法对3个经典函数进行测试,并将其应用于Flow Shop调度问题,仿真实验结果表明:新算法有效克服了停滞现象,增强了全局搜索能力,比基本协同粒子群优化算法的优化性能更好。

关键词:粒子群优化算法; 协同; 优化; Flow Shop 调度
中图分类号:TP18 **文献标志码**:A

An Improved Cooperative Particle Swarm Optimization and Its Application to Flow Shop Scheduling Problem

YU Bin-neng[1], *JIAO Bin*[1], *GU Xing-sheng*[2]
(1. *Shanghai Dianji University*, *Shanghai* 200245, *China*; 2. *School of Information Science and Engineering*, *East China University of Science and Technology*, *Shanghai* 200237, *China*)

Abstract: Aiming at the stagnation problem of the cooperative particle swarm optimization, this paper presents an improved cooperative particle swarm optimization. This proposed method adopts the cooperation principle of optimization algorithm, so it not only ensures the convergence rate, but also avoids plunging into local optimum. Moreover, both comprehensive learning and disturbing mechanism are introduced to strengthen the diversity of population and avoid the stagnation and plunging into local optimum. The new algorithm is tested by three typical functions and the flow shop scheduling problems, respectively. The simulation results show that the proposed algorithm can avoid the stagnation, improve the global convergence ability, and attain better optimization performance.

Key words: particle swarm optimization; cooperative; optimization; Flow Shop scheduling

粒子群优化(Particle Swarm Optimization, PSO)[1-2]算法是一种基于群智能方法的演化计算技术。PSO的优势在于算法的简洁性,易于实现,没有很多参数需要调整,且不需要梯度信息。PSO是演化计算界的研究热点之一,受到越来越多的国内外学者的重视。目前,已有不少关于PSO的改进方法,使算法的性能有了很大的改善,并且已广泛应用于多个学科和工程技术领域[3-6]。

文献[7-8]提出了一种协同粒子群算法(Cooperative Particle Swarm Optimization,

收稿日期:2008-07-02
基金项目:国家自然科学基金(60774078);上海市教育委员会重点科研项目(05ZZ73);上海市自然科学基金(08ZR1408500)
作者简介:虞斌能(1983-),男,浙江舟山人,硕士研究生,研究方向:控制理论和控制工程。
通讯联系人:顾幸生,E-mail: xsgu@ecust.edu.cn

CPSO)，在一定程度上提高了算法性能，克服了PSO存在的“两步前进、一步后退”问题。但是这种协同PSO算法存在停滞现象，即在算法未找到局部最优值之前，无法找到更好的解，所有粒子陷入停滞状态。本文提出一种改进的协同粒子群优化算法(Improved Cooperative Particle Swarm Optimization，ICPSO)，该算法采用贪心法和随机法结合的子群协同方式，同时引入综合学习策略，增加粒子多样性；在此基础上，还采用了扰动机制，当一个子群的当前全局最优解未更新时间大于扰动因子时，重置粒子速度，迫使粒子群摆脱局部极小。用ICPSO和CPSO对几个经典函数进行测试，仿真结果比较显示，ICPSO的性能明显优于CPSO，且有效克服了CPSO算法存在的问题。

生产调度问题属于一类典型的组合优化问题，调度所需要解决的是任务或者作业的次序问题以及资源的分配问题。调度问题是属于离散空间的非数值优化问题，随着调度问题规模的增加，如任务和资源数量的增加，调度问题的求解也愈加困难，难以使用一般的优化方法(如枚举法和整数规划方法等)求解，而基于启发式规则的优化方法也难以得到较好的调度解。目前，在有关粒子群算法的相关文献中，研究调度问题的相关资料较少，因此，展开调度问题中的粒子群优化方法的研究是粒子群算法研究领域的一项重要研究内容。本文将改进的协同粒子群优化算法应用于Flow Shop调度问题，一方面验证了算法良好的优化性能，另一方面，也为调度算法的研究拓展了一种新的智能优化方法。

1　协同粒子群优化算法

协同粒子群优化算法的原理是将优化的每一个向量分割为多个群体，每一个群体在其他群体的帮助下最优化向量不相关联的部分。与基本的PSO算法相比，CPSO算法增加了一个调整参数，即：使用的群体数，也可以解释为每个粒子被分割成的部分数，因此该参数也被称为划分因子，用 k 表示。假设每个群体有 M 个粒子，划分因子 k 将 n 维向量划分为 k 个群体(用 S_i 表示，$i=1,\cdots,k$)，则前 $n \bmod k$ 个粒子群是 $\lceil n/k \rceil$ 维的，后 $k-(n \bmod k)$ 个粒子群是 $\lfloor n/k \rfloor$ 维(这种算法称为CPSO-S_k算法)。在CPSO算法中，记 x_i 为第 i 个粒子的当前位置；v_i 为第 i 个粒子的“飞行”速度，$1 \leqslant i \leqslant s$，$s$ 为粒子个数。记第 i 个粒子迄今为止搜索到的最优位置为 y_i，$1 \leqslant i \leqslant s$；整个粒子群迄今为止搜索到的最优位置为 $\hat{y}$，则每个粒子的速度和位置按式(1)、(2)进行更新。

$$v_i=\omega v_i+c_1 r_1(y_i-x_i)+c_1 r_2(\hat{y}-x_i) \tag{1}$$

$$x_i=x_i+v_i \tag{2}$$

其中：$\omega \geqslant 0$，称为惯性权重；非负常数 c_1 和 c_2 称为学习因子；r_1 和 r_2 是均匀分布于[0,1]的2个随机数。

各子群粒子个体最优位置更新公式如下：

$$b(m,S_m y_i)=\begin{cases} b(m,S_m y_i), f(b(m,S_m x_i)) \geqslant f(b(m,S_m y_i)) \\ b(m,S_m x_i), f(b(m,S_m x_i)) < f(b(m,S_m y_i)) \end{cases} \tag{3}$$

其中：$1 \leqslant m \leqslant k$，$b(\cdot)$函数是评价适应度时，各子群最优位置向量组成的一个完整向量函数，其定义为：

$$b(m,Z)=(S_1\hat{y},\cdots,S_{m-1}\hat{y},Z,S_{m+1}\hat{y},\cdots,S_k\hat{y}) \tag{4}$$

$Z=S_m x_i$ 表示第 m 个子群第 i 个粒子的当前位置，$Z=S_m y_i$ 表示第 m 个子群第 i 个粒子的个体最优位置，$Z=S_m\hat{y}$ 表示第 m 个子群最优位置。式(3)表示在其他子群最优位置保持不变的情况下，第 m 个子群的粒子进行个体最优位置更新，即如果粒子当前位置的适应度比个体最优位置的适应度小，则粒子的个体最优位置等于当前位置，否则保持不变。

各子群最优位置更新公式：

$$b(m,S_m\hat{y})=\arg\min_{b(m,S_m y_i)} f(b(m,S_m y_i)),\quad 1 \leqslant i \leqslant s, 1 \leqslant m \leqslant k \tag{5}$$

式(5)表示第 m 个子群的最优位置取粒子群中适应度最小的粒子个体最优位置。

2　改进的协同粒子群优化算法

2.1　子种群个体的合作

所需优化问题的解空间分解成多个子种群后，如何设计子种群之间的合作也是协同进化算法一个十分重要的环节。子种群的某个个体通过与其他子种群的个体进行合作后，将获得一个适应度的数值，以此来评价这个个体与其他个体合作情况的好坏。适应度的定义如下：

个体适应度＝此个体与其他子种群代表个体合作的目标函数评估。

通过适应度的比较可以发现那些与其他子种群合作较好的个体。通过进化操作算子将个体中优良的基因保存下来，以便产生更好的基因，直至找到全局最优解。一般来说，子种群之间的协作方法有贪心法(即选取子种群中最好的个体作为代表个体)、

保守法(即选取子种群中最差的个体作为代表个体)、随机法(即随机选取子种群中的个体作为代表个体)等[9]。从式(4)和式(5)可以得知,文献[7-8]提出的CPSO算法的子群协作方式为贪心法。贪心法收敛快速,但是也容易使算法陷入局部最优。为了克服此缺点,本文采用优化法的合作方式,既将贪心法和随机法相结合。首先用贪心法计算一个个体的适应度,再用随机法计算该个体的适应度,选取两者中较优的值作为该个体的适应度。这样,即保证了算法的收敛速率,又能防止陷入局部最优,有效地提高了算法的性能。当然,付出的代价是耗费更多的计算时间。

2.2 综合学习策略

虽然协同进化算法是为了克服"维数灾难"问题而提出的,但是该算法存在着陷入局部次优的问题。当算法还未达到局部最优值时,所有的粒子都无法找到更好的解,这就是所谓的停滞现象,是由协同进化算法一次只能更新一个子群的局限性造成的,即一次只能搜索一个子空间。为了克服这个问题,本文提出使用综合学习策略来确定协同粒子群中各粒子的速率和位置。停滞现象归结于早熟收敛,采用综合学习策略的各子群粒子都有一定的概率改变粒子的速率。速率改变也就意味着由速率决定的粒子位置也会发生改变,这样就可以保证种群中粒子的多样性,从而防止出现早熟收敛现象[10]。由式(1)可知,粒子的速率主要由粒子迄今为止搜索到的最优位置 y_i(通常表示为 pbest)和种群的最优位置 $\hat{y}$(通常表示为 gbest)决定。因为种群的最优位置是唯一的,所以一个种群不同的粒子速率主要由相应粒子的最优位置决定。要想粒子具有新的速率,就要改变粒子当前的 pbest 值。

选择粒子使其学习新的速率主要遵循以下步骤:

Step 1 产生一个[0, 1]之间的随机数,如果这个数大于 P_c(选择概率),则该粒子的速率由其本身的 pbest 决定。否则,该粒子的速率由其他粒子的 pbest 决定。

Step 2 在当前粒子所在种群中随机选取除他本身以外的 2 个粒子。

Step 3 比较选取的 2 个粒子各自的 pbest 值,选择较好的 1 个。

Step 4 用选出的 pbest 值计算当前粒子的速率。

所以式(1)的速率更新式可以改写为:

$$v_i = \omega v_i + c_1 r_1 (y_{fi} - x_i) + c_2 r_2 (\hat{y} - x_i) \quad (6)$$

这里 fi 表示第 i 个粒子应选择哪个粒子的 pbest 值,具体由上述步骤决定。

综合学习策略只是使子群当中的小部分粒子改变原有的速率,目的是保持种群的多样性,防止出现停滞现象。但是当种群陷入局部极小时,粒子速率接近 0,单纯的综合学习策略不足以使种群跳出局部最优。因此,本文在此基础上再引入扰动因子机制:如果子群搜索到的全局最优适应值连续 n 步迭代没有更新,则重置该群中所有粒子的速率。扰动机制表示为:

$$\text{If } t - t_n > n \quad \text{Then reset } v \quad (7)$$

其中 t_n 表示最近一次更新搜索到的全局最优适应值的迭代步;n 是自然数。

扰动机制的思想是当粒子群陷入局部极小点时,重置粒子的速率,强迫粒子跳出局部极小点,从而引发新一轮的搜索过程。扰动机制很好地弥补了综合学习策略的不足,进一步改善了算法的整体性能。

2.3 ICPSO 算法步骤

ICPSO 算法的基本步骤如下:

(1) 将整个粒子群分成 k 个子群,初始化设置各参数。

(2) 初始化每个子群内的粒子位置及速率,将粒子的个体最优位置设置为粒子的当前位置,随机选择一粒子作为种群最优位置。

(3) 对每个子群由随机法构造完整的位置向量函数,计算粒子的适应值,求出每个子群全局最优适应值。

(4) 由贪心法构造完整的位置向量函数计算粒子适应值,再由随机法构造完整的位置向量函数计算粒子适应值。比较两个结果,选择较好的一个作为该粒子的适应值,更新每个子群的全局最优适应值。

(5) 判断 $t-t_n$ 是否大于扰动因子 n。是,则重置粒子速率;否则继续。

(6) 对每个子群中的各粒子执行规则 step (1)~(4),确定各粒子应选择的 pbest 值,根据式(6)、式(2)更新粒子的速率和位置。

(7) 更新迭代次数,若还在迭代范围内,则转到步骤(4);否则,停止更新。

(8) 输出整个粒子群的全局最优适应度,算法运行结束。

3 ICPSO 在函数优化上的应用

为了测试算法的性能,选取 3 个典型的测试函

数，用 ICPSO 算法和 CPSO 算法求函数的最小值，并进行比较。3 个测试函数为 Rosenbrock 函数、Ackley 函数、Griewank 函数，其函数的最小值都是 0。

(1) Generalized Rosenbrock 函数(单峰值函数)：

$$f_1(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2],\quad x_i\in[-32,32]$$

(2) Ackley 函数(多峰值函数)：

$$f_2(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+\mathrm{e},\quad x_i\in[-30,30]$$

(3) Generalized Griewank 函数(多峰值函数)：

$$f_3(x)=\frac{1}{4\,000}\sum_{i=1}^{n}(x_i-100)^2-\prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)+1,\quad x_i\in[-600,600]$$

为了使比较结果更加全面，在使用 CPSO 算法进行计算时，分别采用贪心法和随机法 2 种子群协作方式，2 种方式下的结果都与 ICPSO 算法进行比较。实验中具体的参数设置为：函数的维数 dim＝30，划分为 5 个子群，种群的规模 PS＝100，进化代数 GE＝800，惯性权重 $\omega=0.4$，学习因子 $c_1=c_2=1.49$。扰动因子 n 的大小会给计算带来不同的结果：如果 n 过大，扰动机制还未起作用，算法就已经达到最大迭代次数，失去了扰动机制的意义；如果 n 过小，种群还没有进化到一定程度就增加扰动，那么搜索过于随机，收敛较慢。所以要根据问题本身，适当地选取扰动因子。对 3 个不同的函数，经过多次测试性的实验表明，当 n 的取值为[100，300]时，实验结果较好。本文分别取 $n=100,150,200,250$。表 1 中的 $t1$ 表示 CPSO 算法运行 1 次的平均时间，$t2$ 表示 ICPSO 运行 1 次的平均时间。

对于选择概率 P_c 的设定，文献[10]指出，在同一种群中的不同粒子最好各自分配选择概率 P_c，这样可以使粒子具有不同层次的探测开发能力。计算粒子的 P_{c_i} 经验公式：

$$P_{c_i}=0.05+0.45\frac{\exp\frac{10(i-1)}{s-1}-1}{\exp 10-1}\qquad(8)$$

这里 P_{c_i} 在 0.05～0.50 变化。在测试实验中，分别将 P_c 作为一个常数和按式(8)变化的变量进行数次测试，比较数次结果中的最优值。比较结果发现 P_c 作为一个常数时的最优值较好。对 P_c 取[0.05，0.50]之间不同的常数进行测试，发现当 P_c 取 0.30 时，测试结果较优，因此在以下的实验中，将选择概率 P_c 均设定为 0.30。

实验中将每个例子在每个参数设定下各运行 10 次，比较 10 次运行结果各自的最优值(全局极小值下界 Min)和平均值 Average。最优的计算结果用黑体表出，具体结果见表 1。

表 1　ICPSO 和 CPSO 函数优化结果比较

Table 1　Comparison results of function optimization between ICPSO and CPSO

Function	GE	PS	CPSO			ICPSO			
			Min	Average	$t1$/s	n	Min	Average	$t2$/s
f_1	800	100	26.181 3 21.748 7	120.210 4[1)] 39.753 1[2)]	17.31 13.61	100	0.039 3	15.164 0	28.79
						150	**0.030 4**	**14.321 1**	28.96
						200	0.103 0	21.634 8	28.91
						250	0.141 3	20.384 6	28.80
f_2	800	100	$1.509\,9\times10^{-14}$ $1.509\,9\times10^{-14}$	0.770 0[1)] $1.794\,1\times10^{-14}$[2)]	25.23 22.12	100	$\mathbf{4.440\,9\times10^{-15}}$	$5.862\,0\times10^{-15}$	46.09
						150	$4.440\,9\times10^{-15}$	$\mathbf{5.862\,0\times10^{-15}}$	46.48
						200	$4.440\,9\times10^{-15}$	$6.098\,8\times10^{-15}$	46.51
						250	$4.440\,9\times10^{-15}$	$6.217\,2\times10^{-15}$	46.20
f_3	800	100	0.022 2 0	0.050 4[1)] 0.033 5[2)]	25.36 22.63	100	0	0.020 3	48.01
						150	**0**	**0.007 9**	48.20
						200	0	0.026 7	48.12
						250	0	0.028 0	48.26

1) Random method; 2) Greedy method

从表1中可以看出,在参数设置和迭代次数相同的情况下,对于3个测试函数,ICPSO的优化效果要明显好于CPSO。无论CPSO采用贪心法还是随机法,10次结果中的最优值和平均值,ICPSO都要比CPSO小。特别是Griewank函数,本文算法都能获得全局最优解。这说明ICPSO算法在克服停滞问题和不成熟收敛问题上有一定的突破。仿真结果验证了改进的协同粒子群优化算法的搜索能力和寻优能力。但是从算法的原理中可以看出,该算法能力提高的同时也付出了耗费更多计算时间的代价。从$t1$和$t2$中可以看出,算法运行一次的时间不是很多,而且这个时间很大程度上受编程语言和运行环境的影响。在现阶段的计算机水平下,用这些时间换取算法性能的提高,是完全值得的。因此,本算法具有很好的生命力。

4 ICPSO在Flow Shop调度问题中的应用

4.1 FSSP的描述和数学模型

Flow Shop调度问题(Flow Shop Scheduling Problem, FSSP)是许多实际流水线生产调度问题的简化模型,也是一种典型的NP-hard问题,因此其研究具有重要的理论意义和实用价值。

FSSP一般可以描述为:n个工件在m台机器上加工;每个工件需要经过m道工序,每道工序要求不同的机器;n个工件在m台机器上的加工顺序相同,不妨设所有产品依次通过$1,2,\cdots,m$台设备才能完成加工任务,工件没有先后次序的限制。为了简化问题,该类调度的理论研究中通常使用以下假设:

(1) 每个工件在同一时刻只能在一个机器上进行加工;

(2) 工件的各个工序加工时间始终为非零定值,不因操作人员的不同而不同;

(3) 机器不发生故障,某一工件一旦开始加工就不能中断,直至完成;

(4) 每个机器在同一时刻只能加工一个工件;

(5) 每个工件在每台机器上只能加工一次;

(6) 各种导致生产的意外不作考虑,并且不考虑机器的准备时间(事实上可以把加工准备时间包含在加工时间中)。

FSSP是一类复杂且极有代表性的流水线生产调度问题的抽象。问题的关键就是求得耗费时间最少的作业顺序安排方案,即工件的加工顺序,使得完成最后一个工件所需的加工周期(Makespan)达到最小。

设工件j在机器i上的加工时间为t_{ij},其中$i=1,2,\cdots,m;j=1,2,\cdots,n$。$C(k,j)$表示工件$j$在机器$k$上的加工完成时间。$j=1,2,\cdots,n$表示工件的调度顺序。那么对于无限中间存储方式的$n$工件、$m$台机器的FSSP,其数学模型可描述为:

$$\begin{aligned}&C(1,1)=t(1,1)\\&C(1,j)=C(1,j-1)+t(1,j)\quad j=1,2,\cdots,n\\&C(i,1)=C(i-1,1)+t(i,1)\quad i=1,2,\cdots m\\&C(i,j)=\max\{C(i,j-1),C(i-1,j)\}+t(i,j)\end{aligned}\tag{9}$$

加工周期为:

$$C_{\max}=C(m,n)\tag{10}$$

上述模型是FSSP优化的一个目标,即找到一个可行调度方案,使得加工周期最小。

4.2 仿真实验

本文采用的编码是实数编码转换为自然数编码的策略。将一个随机的粒子按照权重分量从小到大排序得到对应的新粒子序列,而新粒子序列对应于原粒子的位置就可以组合成一个新的自然数序列。通过这一策略,即可将实数编码的粒子序列转换成对应的自然数编码的FSSP可行解粒子序列。因CPSO算法和ICPSO算法实际是在PSO算法的基础上提出的,因此在应用CPSO和ICPSO处理FSSP的约束条件时,方法与PSO算法有相似之处。本文参考了文献[11]的方法。

选取4个不同规模的经典FSSP问题(TA类),分别用ICPSO算法和CPSO算法进行优化计算。为了能充分合理比较各自的优化性能,CPSO分别采用贪心法和随机法两种子群协作方式。两种算法在相同的参数设定下对每个例子各运算10次,记录10次结果中的最优值(Min)和平均值(Average)。具体结果和部分参数设置见表2,其中Pr表示具体的问题,Size表示问题的规模,The best为对应问题的最优Makespan,n为扰动因子。其他参数的设置:划分成5个子群,惯性权重$\omega=0.4$,学习因子$c_1=c_2=2$,选择概率$P_c=0.3$。

用ICPSO算法和CPSO算法优化4个FSSP问题的收敛曲线如图1所示,其中CPSO1采用的是随机法的子群协作方式,CPSO2采用的是贪心法的子群协作方式。

从4个经典FSSP对比实验中可以发现:采用ICPSO算法时,4个问题的寻优值都比CPSO算法小,说明ICPSO在指定的进化代数下能够比CPSO

找到更好的调度结果。特别是对问题 Ta061，在扰动因子为 100 的情况下，ICPSO 算法在 10 次的计算中，都找到了最优 Makespan，且收敛速率也相对较快。仿真实验说明 ICPSO 算法的性能比 CPSO 算法有了一定的提高，在调度问题中也能发挥出很好的效果。

表 2　ICPSO 和 CPSO 求解 Flow shop 调度问题结果比较

Table 2　Comparison results of Flow Shop scheduling problem between ICPSO and CPSO

Pr	Size	GE	PS	The best	CPSO			ICPSO			
					Min	Average	*t*1/s	*n*	Min	Average	*t*2/s
Ta041	50×10	800	150	2 991	3 108 3 075	3 152.1[1)] 3 105.2[2)]	98.65 93.34	100 150 200	3 046 **3 034** 3 059	3 094.3 **3 085.3** 3 105.8	192.15 192.79 192.76
Ta051	50×20	800	150	3 771～ 3 856	4 020 3 970	4 089.8[1)] 3 993.7[2)]	145.14 140.11	100 150 200	3 942 **3 923** 3 942	3 971.7 **3 964.6** 397 3	285.96 286.40 286.29
Ta061	100×5	400	150	5 493	5 503 5 493	551 6.8[1)] 549 3.2[2)]	59.26 53.98	100 150 200	**5 493** 5 493 5 493	**5 493** 5 493.4 5 493.8	111.96 112.25 112.95
Ta071	100×10	100 0	150	5 770	5 910 5 836	5 972.4[1)] 5 855.2[2)]	203.93 189.18	100 150 200	5 800 5 787 **5 785**	5 829.7 5 844.2 **5 831.4**	395.06 391.03 388.57

1) Random method ; 2) Greedy method

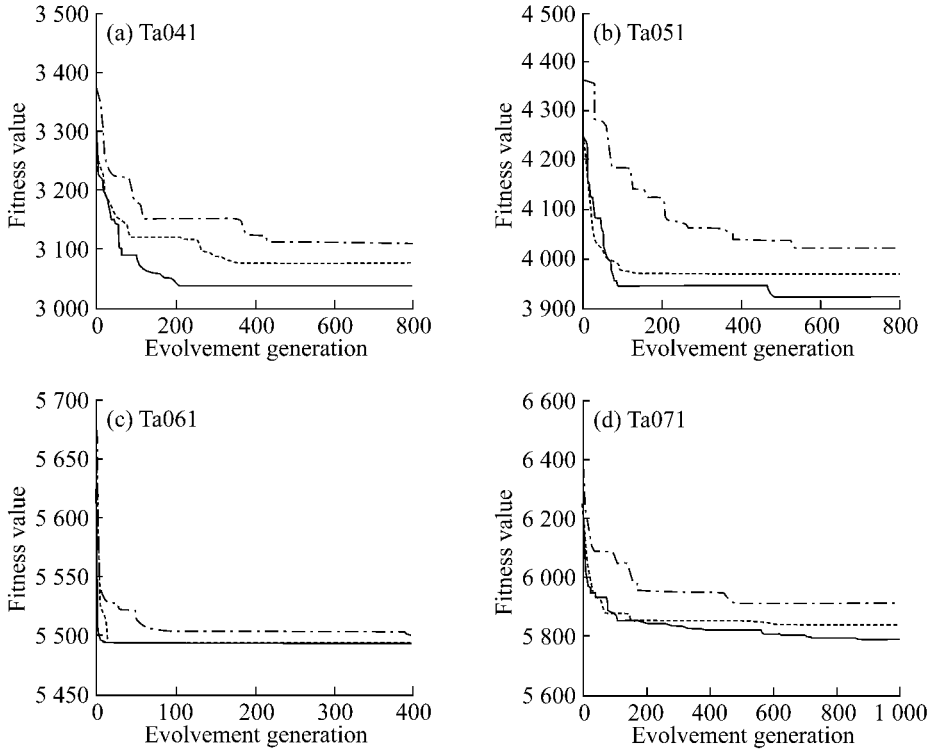

图 1　CPSO 和 ICPSO 的收敛曲线比较图

Fig. 1　Convergence rate contrast figure of CPSO and ICPSO

— · CPSO1；- - - CPSO2；——ICPSO

5　结　论

针对协同粒子群优化算法（CPSO）本身的缺陷，本文提出改进的协同粒子群优化算法（ICPSO）。在 CPSO 算法的框架上，采用优化法的子群协作方式，引入了综合学习策略和扰动机制，以克服算法的停滞现象和早熟收敛问题。通过对函数优化和

Flow Shop 调度问题的仿真实验,验证了本文的算法,有效克服了 CPSO 算法的缺陷,增强了 0 粒子搜索较好解的能力,且在收敛速率上也相对较快,是一种高效的协同优化搜索算法。

参考文献:

[1] Kennedy J, Eberhart R. Particle swarm optimization [C]// IEEE Int Conf on Nural Networks. Perth, Austrlia: IEEE Service Centre, 1995: 1942-1948.

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

[3] Zhang Hong, Li Xiaodong, Li Heng. Particle swarm optimization-based schemes for resource-constrained project scheduling [J]. Automation in Construction, 2005, **14**(3): 393-404.

[4] Lian Zhigang, Gu Xingsheng, Jiao Bin. A dual similar particle swarm optimization algorithm for job-shop scheduling with penalty [C]//Proceedings of the World Congress on Intelligent Control and Automation (WCICA). Dalian, China: IEEE Service Centre, 2006: 7312-7316.

[5] Zhu Jin, Gu Xingsheng, Jiao Bin. An improved particle swarm optimization algorithm for short-term scheduling of single-stage multiproduct batch plants with parallel lines [C]//Proceedings of the International Conference on Bio-inspired Computing——Theory and Applications. Wuhan, China: Watam Press, 2006: 18-22.

[6] Zwe-Lee G. A particle swarm optimization approach for optimum design of PID controller in AVR system [J]. IEEE Transactions on Energy Conversion, 2004, **19**(2): 384-391.

[7] Van den Bergh F, Engelbrecht A P. Training product unit networks using cooperative particle swarm optimizers [C]// Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO). San Francisco, USA: Morgan Kaufmann, 2001: 126-131.

[8] Van den Bergh F, Engelbrecht A P. Effects of swarm size on cooperative particle swarm optimizers [C]//Proceedings of the GECCO. San Francisco, USA: Morgan Kaufmann, 2001: 892-899.

[9] Potter M A. The design and analysis of a computational model of cooperative coevolution [D]. Washington, DC: George Mason University, 1997.

[10] Liang J J, Qin A K, Suganthan P N, *et al*. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions [J]. IEEE Transactions on Evolutionary Computation, 2006, **10**: 281-295.

[11] Lian Zhigang, Gu Xingsheng, Jiao Bin. A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan [J]. Chaos, Solitons and Fractals, 2006, **35**: 851-861.