一种自适应离散粒子群算法及其应用研究

张长胜,孙吉贵,欧阳丹彤
(符号计算与知识工程教育部重点实验室,吉林长春 130012)

摘 要: 本文提出了一种改进的离散粒子群算法.为了克服算法的早熟收敛问题,引入了一个排斥过程用于增加群体的多样性,提出了一种控制群体多样性的准则,实现了算法运行过程中吸引和排斥过程的动态自适应切换.为了提高算法的收敛速度,提出了一种惯性权重动态变化策略,在算法执行的不同阶段,使惯性权重随迭代次数动态自适应变化.试验中发现,引入局部搜索技术后,算法的性能会进一步提高.最后将此算法用于解决 TSP 问题及车间调度问题并与其他相关算法进行了比较,实验结果表明,收敛速度快,稳定性强.

关键词: 粒子群算法; 作业调度; 群体多样性
中图分类号: TP38 **文献标识码**: A **文章编号**: 0372-2112 (2009) 02-0299-06

A Self-Adaptive Discrete Particle Swarm Optimization Algorithm

ZHANG Chang-sheng, SUN Ji-gui, OUYANG Dan-tong
(*Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun, Jilin* 130012, *China*)

Abstract: A self-adaptive discrete particle swarm algorithm is proposed. In order to overcome the premature convergence of the algorithm, a repulsive process is introduced to increase the swarm diversity and a metric to measure the swarm diversity is also designed. The attractive and repulsive processes can adaptively change during running. To speed up convergence, a strategy used to control the inertia weight is advanced which changes dynamically with the iterations during different running phrase of the algorithm. Moreover, algorithm performance can be enhanced further if local search strategies are combined. Finally, the proposed algorithm is used to solve the TSP and FSSP problems and compared with other related algorithms. The experiment results showed its superiority.

Key words: DPSO; job scheduling; swarm diversity

1 引言

粒子群算法(PSO[1])主要被设计为在连续论域中搜索数值函数的最优值,并且实验证明它是一个非常有效的工具,具有模型和设计直观简单,容易操作,执行速度快,效率高等优点,而离散论域中 PSO 算法的研究却非常少.但是在实际应用中,很多问题被建模在离散空间中,典型的例子包括求离散元素的顺序,如调度、路径等问题.因为普通粒子群算法最后求得的解不能保证它一定是在离散空间的,因此失去了它原有的优势.为了解决粒子群算法求解离散问题的劣势,人们发展了几种离散粒子群算法(DPSO)[2,3].同连续粒子群算法(CPSO)一样,这些 DPSO 算法都存在早熟收敛问题,这主要是由于搜索过程中粒子间快速的信息流动使粒子聚集在一起,群体多样性快速下降,最终导致群体适应度停滞,陷入局部最优解[4].所以维持群体的高多样性能够有效地克服算法的早熟收敛问题.

人们已经提出了几种克服 CPSO 早熟收敛问题的方法[5],而目前针对 DPSO 的早熟收敛问题的研究还没有人提出有效的方法.本文提出了一种基于 AllDifferent 约束的离散粒子群算法[3]的自适应离散粒子群算法 SADPSO.为了克服算法的早熟收敛问题,提出了一种控制群体多样性的准则用来度量当前群体的多样性,并在原有算法的基础上引入了一个排斥过程用于增加群体的多样性.当群体多样性低于指定的阈值 d_{low} 时,执行排斥过程,使粒子偏离个体最差解及当前群体最优解,搜索未曾到达过的搜索区域,直到群体的多样性高于指定的阈值 d_{high},此时执行吸引过程,使个体向当前群体最优及个体最优解运动.为了进一步提高算法的性能,本文定义了一种自适应策略,在算法执行的不同阶段,使惯性权重随迭代次数动态自适应变化,并在算法中引入了几种局部搜索技术以提高算法的收敛速度

收稿日期:2007-10-15;修回日期:2008-08-15
基金项目:国家自然科学基金(No. 60473003, No. 60773097);吉林省青年科研基金(No. 20080107, 20080617)

及解的精度. 最后将此算法用于解决 TSP 问题及流水作业调度问题并与其他相关算法进行了比较,试验结果表明:对于 TSP 问题,SADPSO 算法在求解速度上与 DPSO[6]相当,较 ACO[7]算法快 10 倍左右,在求得的解的质量上也明显优于另外两种算法;对于调度问题,在不同规模的问题上对算法进行了测试,并于 GA[8]算法、DPSO[3]算法及最新提出的 SPSOA[9]算法进行了比较,结果表明在收敛速度、稳定性及最优解的质量方面 SADPSO 显著优于另几种算法.

2 SADPSO 算法

基于置换的 DPSO 算法[10]只是交换位置中元素的顺序,所以它的一个最突出的应用就是求解带有 AllDifferent 约束的离散问题. 因为置换不引入新的值,而只是改变位置向量中元素的顺序,所以如果在构造粒子初始位置的时候能够保证 AllDifferent 约束得到满足,那么在以后的算法运行中,此约束始终会被满足,这样能减少计算 AllDifferent 约束的费用,提高算法的运行效率. 与置换离散离子群算法不同,本文给出的 SADPSO 算法含有吸引子 attractor 和排斥子 depressor 两个过程并且每个粒子增加了一个保存个体历史最差解的记忆. 当群体的多样性小于预先给定的阈值 d_{low}时执行 depressor,直到群体的多样性指标大于 d_{high},此时执行 attractor. 为了描述 SADPSO 算法我们首先给出如下计算群体多样性及各种算子的定义.

定义 2.1 给定群体 S,多样性指标 div 根据下面公式定义:

$$div = D(S) = \frac{2}{1 + e^{-\frac{\sum_{i=1}^{N}\sum_{j=1}^{|S|}|\{p_{ij}\}|}{|S|}}} \tag{1}$$

其中,$|S|$ 表示群体规模,N 问题的维度,p_{ij}表示第个粒子当前位置的第 j 个成分. 可以看出此多样性准则不依赖于群体规模及问题的维度. 当前群体的多样性指标取值于$[0,1]$之间并随群体中的个体之间的差异量单调增加.

定义 2.2 给定两个粒子的位置 $X_i = \{x_{i1}, x_{i2}, \cdots, x_{im}\}$, $X_j = \{x_{j1}, x_{j2}, \cdots, x_{jm}\}$,算子 $SUB(X_i, X_j)$ 即 $X_i - X_j$ 定义如下:

$$SUB(X_i, X_j) = X_i - X_j = \bigcup_{k=1}^{m} P(x_{ik}, x_{jk}) \tag{2}$$

其中 $P(x_{ik}, x_{jk}) = \begin{cases} \{(x_{ik}, x_{j,k})\}, & \text{if} \quad x_{ik} \neq x_{jk} \\ \phi, & \text{if} \quad x_{ik} = x_{jk} \end{cases}$

定义 2.3 给定两个粒子的位置 $X_i = \{x_{i1}, x_{i2}, \cdots, x_{im}\}$, $X_j = \{x_{j1}, x_{j2}, \cdots, x_{jm}\}$,算子 $ADD(X_i, X_j)$ 即 $X_i + X_j$ 定义如下:

$$ADD(X_i, X_j) = X_i + X_j = \bigcup_{k=1}^{m} E(x_{ik}, x_{jk}) \tag{3}$$

其中 $E(x_{ik}, x_{jk}) = \begin{cases} \{x_{ik}\}, & \text{if} \quad x_{ik} = x_{jk} \\ \phi, & \text{if} \quad x_{ik} \neq x_{jk} \end{cases}$

有关位置 + 速度$(X_i + V)$;系数 × 速度$(\phi \times V)$;速度 + 速度$(V_i + V_j)$ 的定义可参见文献[3]. 此外本文将 SADPSO 算法中出现的符号定义为:$P_{k,best}$表示第 k 个粒子的个体历史最优解位置;$P_{k,worst}$表示第 k 个粒子的个体历史最差解位置;$P_{k,current}$表示第 k 个粒子的当前位置;G_{best}表示当前群体最优解位置;V_k^t 表示第 k 个粒子在第 t 次迭代中速度. 由此,SADPSO 算法中粒子的两种速度更新公式和位置更新公式定义如下:

速度更新公式:

Attractor:

$$V_k^{t+1} = w \times V_k^t + \phi_1(P_{k,best} - P_{k,current}) + \phi_2(G_{best} - P_{k,current}) \tag{4}$$

Depressor:

$$V_k^{t+1} = w \times V_k^t + \phi_1(P_{k,current} - P_{k,worst}) + \phi_2(P_{k,current} - G_{best}) \tag{5}$$

其中 w 表示惯性权重,用来控制粒子以前速度对当前速度的影响;ϕ_1, ϕ_2 表示学习系数,其值为 0 到 1 之间的一个随机数.

位置更新公式:

$$P_k^{t+1} = P_k^t + V_k^{t+1} \tag{6}$$

可以看出,Attractor 与基本的 SADPSO 类似,粒子之间共享当前较优解的信息,粒子之间相互吸引;而在 Depressor 中,粒子之间共享个体最差及当前群体最优解信息,粒子向远离这些位置的方向运动,搜索未曾到达的潜在最优解区域,增加群体多样性. SADPSO 算法具体可描述如下:

Algorithm *SADPSO*

```
begin
t = 0;
init();
do{
  if(dir > 0 && div < d_low)    dir = -1;
  if(dir < 0 && div > d_high)   dir = 1;
  //compute the initial weight
  w = calculateW();
  for (each particle k) {
    //update speed
    if (div > 0)
      V_k^{t+1} = w × V_k^t + φ1(P_{k,best} - Pk,current);
                + φ2(G_best - P_{k,current})
    else
      V_k^{t+1} = w × V_k^t + φ1(P_{k,current} - Pk,worst);
                + φ2(P_{k,current} - G_best)
    end if;
    //update position
```

$P_k^{t+1} = P_k^t + V_k^{t+1}$;
compute the particle fitness $F(P_k^{t+1})$;

update the particle personal best $P_{k,best}$);
update the swarm global best G_{best};
} endfor;
// compute the current swarm diversity
$div = calculateDiversity()$;
$t = t + 1$;
}while($t < t_{max}$);
output G_{best};
End.

对于 PSO 算法的参数中,惯性权重对粒子的全局和局部搜索能力有很大影响[10],它决定了搜索步伐的大小,较大的 w 有利于全局搜索,较小的 w 有利于局部搜索. 通常在开始时搜索步伐大些,随着迭代次数的增加,减小 w 值以防止错过最优解. 选择一个合适的 w 可以平衡全局和局部搜索能力,这样可以以最少的迭代次数找到比较满意的解. 这里我们提出一种如下的非线性自适应策略,使 w 在算法的运行过程中动态变化.

$$w = f(t) = \begin{cases} \left\{ \dfrac{(t_{\max} - t)^n}{(t_{\max} - t_{sw})^n} \right\} (w_{sw} - w_{final}) + w_{final}, & attract = true \\ C, & attract = false \end{cases} \quad (7)$$

其中 $t_{\max}$ 表示最大迭代次数,t 表示当前迭代次数,$w_{initial}$ 表示初始权重,w_{final} 运行结束时的最终权重,C 为常量其值略小于 $w_{initial}$,算法开始运行时 $w_{sw} = w_{initial}$,当 depressor 运行一次之后 $w_{sw} = C$. t_{sw} 的值等于算法中 depressor 停止运行,attractor 开始运行时刻的当前迭代次数,其初始值为 0. 通过调节参 n 数来动态平衡全局搜索及局部搜索能力. 由图 1 可以看出,当 $n = 1$ 时,w 呈线性变化;当 $n > 1$ 时算法倾向于全局搜索;$n < 1$ 时算法倾向于局部精化.

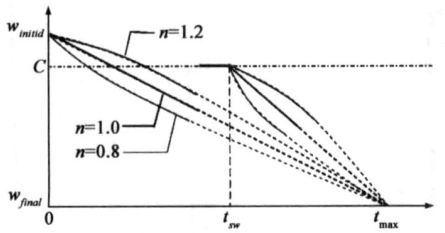

图1　w变化曲线

人们已经发现,给惯性权重较大初始值,在搜索空间中进行粗略的全局搜索,然后随着算法的运行逐渐减小,使算法在迭代后期具有出色的局部精化能力,这样能够提高算法快速的向全局最优解收敛. 在 SADPSO 算法中,当最大迭代次数与当前迭代次数的差小于预先给定的值 M 时,执行 attractor 并令参数 n 的取值小于 1.0,其他情况 n 的取值大于 1.0. 本文试验中 n 的取值分别为 0.6 和 1.6,w 的计算过程如下

procedure *calculate W*

begin
if($t_{max} - t < M$)
{
// $L1 < 1.0$
$n = L1$;
$div = 1$;
}
else
// $L2 > 1.0$
$n = L2$;
if($div = 1$)
$w = \{((t_{max} - t)^n)/((t_{max} - t_{sw})^n)\}$;
$(w_{sw} - w_{final}) + w_{final}$
else
$w = C$;
end.

为了测试惯性权重及参数 n 对算法性能的影响,在用例 Ta021 上对算法进行了测试,令 w 取常量值和随迭代次数自适应变化时,算法分别运行 10 次求得的最优解的平均值如表 1 所示:

表 1　惯性权重对算法的影响

$w = 0.8$	$w = 0.1$	$w = 0.8 \sim 0.1$		
2380	2341	$n = 1.6$	$n = 1.0$	$n = 0.6$
		2308	2333	2332

可以看出令惯性权重自适应变化会提高算法的性能,在本文的试验中我们取 $w_{initila} = 0.8$, $w_{final} = 0.1$, $C = 0.68$, $n = 1.6$ 并且令群体规模为 20.

3　在 TSP 问题中的应用

TSP 问题(Traveling Salesman Problem)是经典的组合优化问题. 针对 TSP 问题的特点,首先对 SADPSO 算法速度公式中的算子作了重新定义. 试验分别采用了 Burmal14[11], Oliver30[7], eil51[11] 等不同规模的测试用例,并与 DPSO 算法[6]及 ACO 算法[7]进行了比较.

定义 3.1　对于含有 m 个元素的序列 $X = \{x_1, x_2, \cdots, x_m\}$,作用其上的滑动算子 $SL(X, k)$ 定义如下:

$$SL(X, k) = \{x_{(1+k)\%m}, x_{(2+k)\%m}, \cdots, x_{(m+k)\%m}\} \quad (8)$$

定义 3.2　对于含有 m 个元素的序列 $X = \{x_1, x_2, \cdots, x_m\}$,作用其上的翻转算子 $RE(X)$ 定义如下:

$$RE(X) = \{x_m, x_{m-1}, \cdots, x_1\} \quad (9)$$

可以看出对于 TSP 问题,以上两种算子并不会改变粒子所代表的解路径.

定义 3.3　给定两个粒子位置 $X_i = \{x_{i1}, x_{i2}, \cdots,$

$x_{im}\}$及 $X_j=\{x_{j1},x_{j2},\cdots,x_{jm}\}$,作用于其上的算子 $\odot$定义如下:

$$X_i\odot X_j=\mathrm{SUB}(X_i,SL(RE(X_j),l))\ \text{or}\ \mathrm{SUB}(X_i,RE(SL(X_j,l))) \tag{10}$$

其中 $l\in[0,m-1]$,最小化 $|X_i\odot X_j|$.

定义 3.4 对于给定的集合 $S=\{e_1,e_2,\cdots,e_s\}$,定义操作 $G(S)$按如下方式随机产生一个集合 R:

$$G(S)=R=\{r_1,r_2,\cdots,r_{\lfloor s/2\rfloor}\} \tag{11}$$

其中 $r_i=\{(r_{il},r_{ik})\mid r_{ik}\in S\wedge r_{il}\in S\wedge r_{ik}\neq r_{il}\}$并且不存在 $r_j=\{(r_{ik},r_{il})\}$.

定义 3.5 给定两个粒子位置 $X_i=\{x_{i1},x_{i2},\cdots,x_{im}\}$及 $X_j=\{x_{j1},x_{j2},\cdots,x_{jm}\}$,作用于其上的算子 $\oplus$定义如下:

$$X_i\oplus X_j=G(ADD(X_i,SL(RE(X_j),l)))\ \text{or}\ =G(ADD(X_i,RE(SL(X_j,l)))) \tag{12}$$

此时,SADPSO 算法中速度更新公式可表示为:

$$V_k^{t+1}=w\times V_k'+\phi_1(P_{k,best}\odot P_{k,current})+\phi_2(G_{best}\odot P_{k,current}) \tag{13}$$

$$V_k^{t+1}=w\times V_k'+\phi_1(P_{k,worst}\oplus P_{k,current})+\phi_2(G_{best}\odot P_{k,current}) \tag{14}$$

表 2 是算法在各测试集上分别运行 20 次得到的最优解,最优解的均值的比较结果. 试验中发现,在求解速度上 SADPSO 算法明显快于 ACO 算法. 对于 burma14,SADPSO 算法每次运行都能得到最优解,对于另外两个问题,每次求得的解也较接近最优解,明显优于另外两种方法,由最优解均值可以看出,对于 TSP 问题,SADPSO 算法有较强的稳定性. 由于 Oliver30 被广泛的用于各种算法的测试中,为了便于比较,图 2 给出了 SADPSO 算法求得的最优路径图,图 3 给出了 DPSO 及 SADPSO 算法的最优解随迭代次数变化的 20 次运行的平均值的收敛曲线. 可以看出无论是收敛速度还是求得的最优解的精度都明显优于 DPSO 算法.

图2 Oliver30的最优路径图

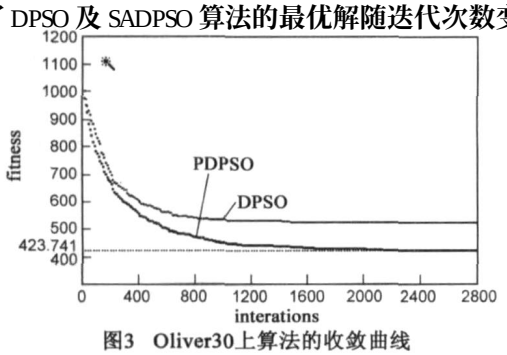

图3 Oliver30上算法的收敛曲线

表 2 TSP 问题的试验结果

问题	算法	得到的最优解	平均值	问题最优解
burma14.tsp	SADPSO	30.8785	30.8785	30.8785
	DPSO	30.8785	31.5551	
	ACO	30.8785	31.4075	
Oliver30.tsp	SADPSO	423.7410	424.8267	423.7406
	DPSO	453.4200	515.4413	
	ACO	434.2214	447.9351	
eil51.tsp	SADPSO	436.7730	440.7810	426.0000
	DPSO	476.9910	523.5421	
	ACO	449.6437	454.3333	

4 在流水作业调度中的应用

目前使用 DPSO 算法求解调度问题的研究非常少. 所谓流水车间作业调度问题(FSSP)是指有 n 个作业,每个作业 T_i 又分为 m 个子作业 $T_{i1},T_{i2},\cdots,T_{im}$,$1\le i\le n$,子作业 T_{ij}必须由处理机 P_j 来完成,$1\le j\le n$. 完成 T_{ij} 的时间是 t_{ij}. 一台处理机在同一时间只能处理一个作业,对任何作业 T_i,T_{ij-1}必定在 T_{ij}前完成,各处理机加工作业的次序相同. 调度目标是如何安排加工次序,使得能用最少的时间完成这批作业. 流水调度是一种典型的生产调度问题,已经被证明是强 NP 完全问题. 为了便于比较,算法中引入了文献[3]中的局部搜索技术,每次迭代执行一次局部搜索. 试验采用 Carlier、Reeves、Taillard 提供的不同规模的数据.

表 3 Carlier 上与构造启发式算法的比较

问题	最优解	问题规模		构造启发式算法					SADPSO
		n	m	Palmer	Gupta	CDS	RA	NEH	算法
Car1	7308	11	5	7472	7348	7202	7817	7308	7308
Car2	7166	13	4	7940	7534	7410	7509	7940	7166
Car3	7312	12	5	7725	7399	7399	7399	7503	7312
Car4	8003	14	4	8423	8423	8423	8357	8003	8003
Car5	7720	10	6	8520	8773	8627	8940	8190	7720
Car6	8505	8	9	9487	9441	9553	9514	9159	8505
Car7	6590	7	7	7639	7639	6819	6923	7668	6590
Car8	8366	8	8	9023	9224	8903	9062	9032	8366

表 4 Carlier 上与随机优化算法的比较

问题	问题规模		随机优化算法					SADPSO
	n	m	GA	Hybrid GA	DPSO	DPSO+LS	SPSOA	算法
Car1	11	5	0.00	0.00	0.00	0.00	0.00	0.00
Car2	13	4	0.00	0.00	0.00	0.00	0.00	0.00
Car3	12	5	2.42	0.00	1.09	0.00	0.00	0.00
Car4	14	4	0.00	0.00	0.00	0.00	0.00	0.00
Car5	10	6	0.36	0.00	0.62	0.00	0.00	0.00
Car6	8	9	0.00	0.76	0.00	0.00	0.00	0.00
Car7	7	7	0.00	0.00	0.00	0.00	0.00	0.00
Car8	8	8	0.00	0.00	0.00	0.00	0.00	0.00

在 Carlier 和 Reeves 数据集上分别将 SADPSO 算法

与几种构造启发式算法和随机优化算法[8,12]进行了比较,试验结果如表3、表4、表5、表6所示. 其中表3、表5是各种算法分别运行20次得到的最优结果的比较,表4、表6是各种算法求得的最小完工时间的相对百分比的比较.

表5　Reeves上与构造启发式算法的比较

问题	最优解	问题规模		构造启发式算法					SADPSO
		n	m	Palmer	Gupta	CDS	RA	NEH	算法
Rec01	1247	20	5	1391	1434	1399	1399	1334	1247
Rec03	1109	20	5	1223	1380	1273	1159	1136	1109
Rec05	1242	20	5	1290	1429	1338	1434	1294	1245
Rec07	1566	20	10	1715	1678	1697	1722	1637	1566
Rec09	1537	10	10	1915	1792	1639	1714	1692	1537
Rec11	1431	20	10	1685	1765	1597	1636	1635	1431

表6　Reeves上与随机优化算法的比较

问题	问题规模		随机算法				SADPSO
	n	m	GA	Hybrid GA	DPSO	DPSO+LS	算法
Rec01	20	5	8.26	0.14	1.36	0.16	0.00
Rec03	20	5	7.21	0.09	0.54	0.18	0.00
Rec05	20	5	5.23	0.29	0.97	0.24	0.00
Rec07	20	10	8.56	0.69	3.70	1.15	0.00
Rec09	10	10	5.14	0.64	5.47	2.41	0.00
Rec11	20	10	8.32	1.11	1.11	1.05	0.00

表7　Tailliard数据集上算法的比较

问题	最优解	问题规模		其它算法		SADPSO
		n	m	GA	SPSOA	算法
Ta005b0	1235	20	5	1304	1244	1235
Ta010b0	1108	20	5	1188	1108	1108
Ta020b0	1591	20	10	1732	1600	1597
Ta030b0	2178	20	20	2368	2194	2184
Ta050b0	3065	50	10	3384	3173	3131
Ta060b0	◆	50	20	4338	3917	3852
Ta070b0	5322	100	5	5409	5342	5328
Ta080b0	5845	100	10	6302	5903	5903

在Tailliard提供的数据上,比较了本文的SADPSO算法、A. C. Nearchou提出的GA算法[7],它目前用GA求解FSSP问题中效果最好的[8]、最新提出的SPSOA[18]算法. 对于SPSOA算法使用的是所有交叉策略求得的最好解. 试验结果如表7所示,◆表示目前最优解未知的问题.

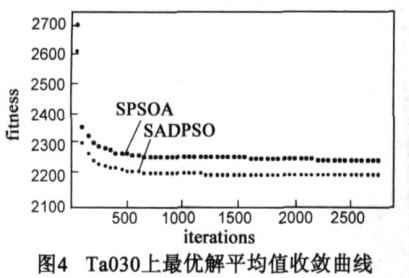

图4　Ta030上最优解平均值收敛曲线

文献[18]表明在求解调度问题时SPSOA算法明显优于GA算法,并且GA算法的收敛速度较慢,因此我们只需比较SADPSO算法与SPSOA算法的收敛性,图4,图5是两种算法在中等规模数据集ta030及大规模数据集ta060上分别运行20次的最优解平均值随迭代次数的进化曲线. 可以看出无论是解的质量还是收敛速度,SADPSO算法都明显优于SPSOA算法.

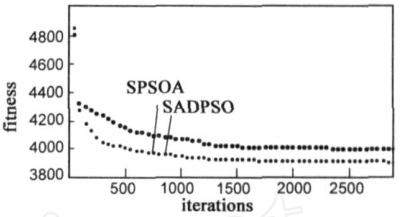

图5　ta060上最优解平均值收敛曲线

5　结束语

本文工作主要体现在以下几个方面:(1)提出了一种改进的离散粒子群算法SADPSO,引入了一个排斥过程防止群体早熟收敛,使用吸引和排斥过程使群体不断进化;(2)定义了一种DSPO算法解决离散优化问题时群体多样性指标,实时度量群体的多样性;(3)为了提高算法的性能,提出了一种惯性权重的动态自适应变化策略并引入了局部搜索技术;(4)将此算法用于解决不同规模的TSP问题及FSSP问题,并于其他相关算法进行了试验对比,验证了算法的有效性.

参考文献:

[1] J Kennedy, R C Eberhart. Particle swarm optimization[A]. in: Proceedings of the IEEE International Joint Conference on Neural Networks[C]. Piscataway, NJ: IEEE Service Center, IEEE Press, 1995. 1942 - 1948.

[2] Qingyun Yang, Jigui Sun, Juyang Zhang, Chunjie Wang. A hybrid discrete particle swarm algorithm for open-shop problems [A]. Proceedings of the 6th International Conference on Simulated Evolution And Learning (SEAL 2006)[C]. Hefei, China, LNCS 4247, 2006. 158 - 165.

[3] K Rameshkumar, R K Suresh, K M Mohanasundaram. Discrete particle swarm optimization (DPSO) algorithm for permutation flowshop scheduling to minimize makspan[A]. In: Proc. ICNC 2005[C]. Changsha, China, LNCS 3612, 2005. 572 - 581.

[4] Pant, M Radha, T Singh, V P. A simple diversity guided particle swarm optimization[A]. IEEE Congress on Evolutionary Computation[C]. Singapore, CEC2007. 2007. 3294 - 3299.

[5] Christopher K. Monson, Kevin D. Seppi, Adaptive Diversity in PSO[A]. Proceedings of the 8th annual conference on Genetic and evolutionary computation Seattle[C]. Washington, USA, 2006. 59 - 66.

[6] M Clerc: Discrete particle swarm optimization, illustrated by the

Traveling Salesman Problem[A]. In: New Optimization Techniques in Engineering[C]. Heidelberg, Germany, 2004. 219 - 239.

[7] 吴庆洪,张纪会,徐心和. 具有变异特征的蚁群算法[J]. 计算机研究与发展,1999,36(10):1240 - 1245.
Wu Qinghong, Zhang Jihui, Xu Xinhe. An ant colony algorithm with mutation features[J]. Journal of Computer Research and Development, 1999, 36(10): 1240 - 1245.

[8] A C. Nearchou, The effect of various operators on the genetic search for large scheduling problems[J]. Int. J. Product. Econom. 2004, 88(1): 191 - 203.

[9] Zhigang Lian, Xingsheng Gu and Bin Jiao, A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan[J]. Applied Mathematics and Computation, 2006, 175(1): 773 - 785.

[10] A Chatterjee, P Siarry. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization[J]. Computers & Operations Research, 2006, 33(3): 859 - 871.

[11] G. Reinelt, TSPLIB[OL] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/, 2007-09-10.

[12] Ponnambalam, S G, Aravindan P, Chandrasekaran S. Constructive and improvement flow shop scheduling heuristics: an extensive evaluation[J]. Production Planning & Control, 2001, 12(4): 335 - 344.

作者简介:

张长胜 男,1980 年 8 月出生于吉林长春. 吉林大学计算机学院博士研究生,研究方向为智能信息处理,图形图像.
E-mail: zcs820@yahoo.com.cn

孙吉贵 男,1962 年 11 出生,博士、教授、博士生导师. 1993 年博士毕业于吉林大学计算机科学系,1997 年 7 月晋升为教授;97 年至 98 年在英国 Ulster 大学留学;现任教育部"符号计算与知识工程"重点实验室(吉林大学)主任、吉林大学计算机学院副院长,兼任中国计算机学会理事、教育部高等学校计算机科学与技术专业教学指导分委员会委员,吉林省计算机学会副理事长,南京大学软件新技术国家重点实验室学术委员会委员. 研究方向为智能信息处理.

欧阳丹彤 女,1968 年生,教授,博士生导师,研究方向为基于模型的诊断. 本文通信作者.
E-mail: paper.820@yahoo.com.cn