一种“自我”感知的高维混沌群体智能算法

陶 乾[1, 2] 黄哲学[2] 顾春琴[3]

[1](广东第二师范学院计算机科学系 广州 510310)
[2](中国科学院深圳先进技术研究院 深圳 518055)
[3](仲恺农业工程学院网络工程系 广州 510225)

摘 要 为避免早熟收敛和提升粒子在高维空间的搜索能力，文章提出了一种“自我”感知的高维混沌群体智能算法。首先，采用 *pBest* 和 *gBest* 混沌双扰动来增强粒子的搜索能力；其次，提出一种“自我”感知策略来帮助种群避免早熟收敛；最后，将三种不同微粒群优化(Particle Swarm Optimization，PSO)算法在旅行推销员问题(Traveling Salesman Problem，TSP)上进行了对比实验。实验结果显示“自我”感知的高维混沌群体智能算法简单、有效可行，值得推荐。

关键词 微粒群优化；混沌；扰动；自我感知；收敛
中图分类号 TP 391 **文献标志码** A

A Self-Perception High-Dimensional Chaotic Particle Swarm Algorithm

TAO Qian[1, 2] HUANG Zhexue[2] GU Chunqin[3]

[1](*Department of Computer Science*, *Guangdong University of Education*, *Guangzhou* 510310, *China*)
[2](*Shenzhen Institutes of Advanced Technology*, *Chinese Academy of Sciences*, *Shenzhen* 518055, *China*)
[3](*Department of Network Engineering*, *Zhongkai University of Agriculture and Engineering*, *Guangzhou* 510225, *China*)

Abstract To avoid the premature convergence and enhance the search capability of the high-dimensional space, a novel self-perception high-dimensional chaotic particle swarm algorithm was presented. Firstly, a double perturbation of *pBest* and *gBest* was used to enhance the searching capability of particles. Secondly, self-perception approach was proposed to help the particle swarm to avoid the premature convergence. Lastly, three discrete PSO variants were tested on the traveling salesman problem (TSP). Experimental results show that the self-perception high-dimensional chaotic particle swarm algorithm is simple, effective and promoting in a high-dimensional space.

Keywords particle swarm optimization; chaotic; perturbation; self-perception; convergence

收稿日期：2014-01-18
基金项目：中国博士后基金(2013M542219)；广东省重大科技攻关项目(2012A080104022)；广东省自然科学基金重点项目(S2012020011067)；广东第二师范学院博士专项(2012ARF05)。
作者简介：陶乾，博士，讲师，研究方向为人工智能、大数据和数据挖掘；黄哲学(通讯作者)，博士，研究员，研究方向为数据挖掘、大数据、云计算和人工智能，E-mail：zx.huang@siat.ac.cn；顾春琴，博士，讲师，研究方向为人工智能和数据挖掘。

1 引 言

群体智能是通过模拟自然界生物群体行为来实现人工智能的一种方法。对群体智能系统内个体行为建模所形成的算法即为群体智能算法。

作为一种高效的群体智能算法，微粒群优化(Particle Swarm Optimization，PSO)算法被广泛应用于各类科学问题求解，得到各领域学者的广泛关注[1-3]。但在处理当前一些复杂问题上性能仍欠佳，容易陷入局部极值，造成早熟收敛。且在高维复杂空间的迭代进化过程有一定的惰性，解的精度难以持续提高。为克服局部极值，避免在进化后期收敛速度慢、精度低，同时为防止粒子早熟收敛，柯西扰动[4,5]、正态扰动[6]、高斯扰动[5,7,8]和混沌 Logistic 随机时间序列扰动[3,9,10]等手段常作为辅助优化技术来优化粒子的搜索行为，改进 PSO 的性能。

混沌随机时间序列具有确定性、普遍性及随机性，有利于帮助粒子逃离局部极值并获取比高斯扰动更好的搜索质量[9]；同时，上述算法采取的扰动技术主要是对 *pBest* 的单极值扰动，且是随机维度的扰动，没有具体分析扰动阶段，扰动有很大的随意性和不准确性。因此我们选取混沌随机时间序列进行 *pBest*、*gBest* 双极值扰动并辅以“自我”感知策略来增强 PSO 算法的性能。

2 高维混沌扰动方法

混沌运动是非线性系统中的一种普遍存在的现象，能在一定范围内不重复地遍历所有状态。Logistic 随机时间序列是经典的混沌运动模型，其公式如式(1)所示。

$$x(t+1)=r\cdot x(t)\cdot(1-x(t)),r\in N,x(0)\in[0,1] \quad (1)$$

其中，当 $r=4$ 时，Logistic 序列是[0, 1]区间上的满映射，混沌效果最佳。

为避免局部最优并加快收敛，*pBest* 和 *gBest* 由混沌 Logistic 随机时间序列进行扰动。*pBest* 和 *gBest* 的双扰动不仅增强了精英粒子的搜索能力，而且保留了 PSO 算法的多样性。采用双扰动后，粒子跟随 *pBest* 和 *gBest* 跳离局部最优解，重新向新的解收敛。

当混沌 Logistic 随机时间序列的幅值与 r_1、r_2、r_4 和 r_5 大整数相乘时会产生较大扰动，导致转轮进行不必要的重复旋转，我们将这一现象称为扰动冗余。为消除扰动冗余，提高扰动的有效性，我们采用求模取余来进行计算。*pBest* 和 *gBest* 的双扰动公式分别如式(2)和(3)所示。

$$pBest(t+1)=\begin{cases}\lceil pBest(t)+r_1x(t)\rceil\%scale\_dim(i),\varepsilon<p_m-r_3\\ \lfloor pBest(t)-r_2x(t)\rfloor\%scale\_dim(i),r_3>p_m\end{cases} \quad (2)$$

$$gBest(t+1)=\begin{cases}\lceil gBest(t)+r_4x(t)\rceil\%scale\_dim(i),\varepsilon<p_m-r_6\\ \lfloor gBest(t)-r_5x(t)\rfloor\%scale\_dim(i),r_6>p_m\end{cases} \quad (3)$$

其中，r_1、r_2、r_4 和 r_5 是大整数；r_3 和 r_6 为 U(0, 1)分布的随机数，$p=0.45$，*scale_dim*(*i*)为维度 *i* 的规模(该维候选解的数量)。

3 “自我”感知策略

为增强算法的整体性能并提升解的精度，*pBest* 和 *gBest* 双扰动的精确时间、精确维数和精确的粒子成为一个待解决的关键问题。因此，我们提出了一种微粒群的“自我”感知策略，该策略主要包含三方面内容：

3.1 扰动时间(迭代代数)的“自我”感知

扰动开始时间(即早熟收敛时间)：使用 *pBest* 与 *gBest* 的平均距离来判断算法是否陷入早熟收敛状态，该平均距离的计算公式如式(4)所示。

$$premature\_convergence(pBest,gBest)=\frac{1}{PNum}\sum_{i=1}^{PNum}\sqrt{\sum_{j=1}^{d}(pBest_{ij}-gBest_j)^2} \tag{4}$$

其中，$PNum$、d 分别为种群规模及总维数。

当 $premature\_convergence(pBest,gbest)\leqslant\varepsilon_d$（$\varepsilon_d$ 是早熟收敛的阈值）时，采取适当的扰动策略。如果过早扰动则导致粒子一直在低位适应值徘徊，难以搜到高精度的解；如果太晚扰动则粒子已经陷入了局部极值，扰动的效果和效率就不明显。

扰动结束时间：目前，扰动结束时间多是采用固定代数，但固定代数容易导致扰动缺乏灵活性，从而使得粒子的搜索性能受到影响。我们定义了一个以 $PNum$、j 和 $L(j)$ 为参数的正比例递增函数，其中 $PNum$ 为种群规模，j 为维度，$L(j)$ 为 j 维候选解的规模。根据各个维度、种群规模和解的复杂程度来确定扰动结束的代数，形式化定义为：$t\geqslant T=\left\lceil f_{\uparrow}\left(PNum,j,L(j)\right)\right\rceil$，$T$ 为 $PNum$、j 和 $L(j)$ 的正比例递增函数。

3.2　扰动粒子的“自我”感知

为减少在飞行过程中种群稳定性的破坏，同时也减轻 CPU 的计算负载，我们从总体 $PNum$ 个粒子中随机抽取$\left\lceil\frac{1}{K}PNum\right\rceil$个粒子作为扰动对象。由于经过迭代飞行后种群之间差异程度较小、种群数目一般只有几十个粒子，且种群中的每个粒子基本独立，彼此间无明显的排斥性，使得每一个粒子被抽中的概率相同。故通过简单随机抽样方法即可实现扰动粒子的“自我”感知，无须对每个粒子都进行 $pBest$ 扰动，同时也可以保证获得的数据对总体的数量特征得出具可靠性的估计判断，从而达到对种群整体的认识。

3.3　扰动的精确维数的“自我”感知

在扰动过程中，粒子的最优信息应该加以保护以防破坏。如果每维都加以扰动，最优位置将不可避免地被破坏，并导致最优解退化。此外，如果随机选取若干维进行扰动，扰动策略则不能有效地改变影响解质量的关键维。显然，随机扰动并不是很好的选择，其正如一次“爆炸”，会破坏某些维度的最优位置。为解决此问题并增强算法的性能，我们采用了精确维度扰动来改变 $pBest$ 和 $gBest$。在多维空间及多粒子的种群中，发现粒子位置的变化并搜索合适的维加以扰动极其困难。因此，我们采用维度惰性来判断每一维是否发生了能感觉得到的变化。如果某转轮（某维）在长时间的迭代中不动或小幅摆动，则该维被定义为惰性维，在下一个进化阶段需要赋予较高的扰动优先级。在具体的研究实验中我们发现由于惰性维的存在导致粒子对应的适应值和算法的求解精度难以提升。若每个粒子有多个维数，则具有较高惰性的维优先进行扰动。$pBest$ 和 $gBest$ 第 j 维的惰性用均方差公式给出，分别如式（5）和（6）所示。

$$Inert\_detection\ (pBest_j)=\sqrt{\frac{1}{PNum}\sum_{i=1}^{PNum}(pBest_{ij}-average(pBest_{1j},pBest_{2j},\cdots,pBest_{PNum,j}))^2} \tag{5}$$

$$Inert\_detection\ (gBest_j)=\sqrt{\frac{1}{m-1}\sum_{t=1}^{m-1}\left(gBest_j(m)-gBest_j(t)\right)^2} \tag{6}$$

其中，$PNum$、m 分别是种群规模、当前迭代次数。

维度惰性监测：

如果 $Inert_detection(pBest_j)\leqslant\varepsilon_{\sigma(pBest)}$或 $Inert\_detection(gBest_j)\leqslant\varepsilon_{\sigma(gBest)}$，则第 j 维为惰性维，需要进行扰动。$\varepsilon_{\sigma(pBest)}$和 $\varepsilon_{\sigma(gBest)}$分别为 $pBest$ 和 $gBest$ 的惰性阈值。如果 $Inert_detection(pBest_j)$和 $Inert_detection(gBest_j)$的值较大，则说明该维在迭代过程中能有效进化，不需要进行扰动。

通过监测扰动的精确时间和精确维数，可实现双扰动的精准化，同时消除惰性维对粒子搜索

性能的影响，提升算法求解精度和收敛速度。

4 Self-Perception_PSO 算法

设 *Pnum* 为种群规模，*t*、*T* 分别为迭代的当前代数和最大代数，$x_{ij}(t)$ 、$v_{ij}(t)$分别为粒子在第 *t* 次迭代时第 *j* 维的位置和速度，$L(j)$ 为任务 *j* 的候选解的规模，$v_{max}(j)$为每个粒子第 *j* 维的最大速度，fitness(*i*)是粒子 *i* 的适应值，$pBest_{ij}(t)$ 是第 *t* 次迭代时第 *i* 个粒子第 *j* 维的最优位置，$gBest_{ij}(t)$是整个种群第 *j* 维的最优位置，则 Self-perception_PSO 算法具体描述如下：

(1)(初始化)：初始化 *PNum*、*T*、$L(j)$，$v_{max}(j)$、$pBest_{ij}(1)$和 $gBest_j(1)$。

(2)重复直到 $t \geqslant \lceil T = f_{\uparrow}(PNum, j, L(j)) \rceil$，*T* 为 *PNum*、*j* 和 $L(j)$ 的正比例递增函数。

①根据式(4)计算 *pBest*(*t*)_*convergence*(*pBest*, *gBest*)；

②如果 *pBest*(*t*)_*convergence*(*pBest*, *gBest*)$\leqslant \varepsilon_d$，则采用随机抽样抽取$\left\lceil \frac{1}{K} PNum \right\rceil$个粒子；

a. 根据式(5)计算 *Inert_detection*($pBest_j$)；如果 *Inert_detection*($pBest_j$)$\leqslant \varepsilon_{\sigma(pBest)}$，则根据式(2)精确扰动 $pBest_{ij}(t)$；

b. 根据式(6)计算 *Inert_detection*($gBest_j$)；如果 *Inert_detection*($gBest_j$)$\leqslant \varepsilon_{\sigma(gBest)}$，则根据式(3)精确扰动 $gBest_{ij}(t)$；

③更新 $v_{ij}(t)$、$x_{ij}(t)$、*fitness*(*i*)、$pBest_{ij}(t)$和 $gBest_{ij}(t)$；

(3)输出最优适应值和对应的最优解。

5 实验结果和分析

我们以 TSP 问题为例验证算法的有效性。TSP 问题是一个典型的组合优化问题，已被证明具有 NPC 计算复杂性，可以形式化定义如下：$G=(V, E)$，其中 *V* 表示城市节点集合；*E* 表示城市之间路径的集合。*n* 是城市总数，弧$(i, j) \in E$，城市 *i* 到城市 *j* 的距离为 $w_{i,j}$，在对称 TSP 中，$w_{i,j}=w_{j,i}$。TSP 问题的目标是求最短哈密顿回路。

$$|V|=n, V=\{v_i\}, 1 \leqslant i \leqslant n$$

$$E=E_1 \cup E_2 \cup \cdots \cup E_n$$

$$E_i=(v_i,v_1) \cup (v_i,v_2) \cup \cdots (v_i,v_{i-1}) \cup (v_i,v_{i+1}) \cup \cdots \cup (v_i,v_n)$$

现求出一组弧集(*ArcSet*)，

$$ArcSet=arc_1 \cup arc_2 \cup \cdots \cup arc_n,$$
$$arc_i=(v_i,v_j) \in E_i, i \neq j, 1 \leqslant j \leqslant n$$

使得弧集总长度 $len(ArcSet)=len(arc_1)+len(arc_2)+\cdots+len(arc_n)$最小，因此，TSP 问题的求解目标可描述为

$$\text{minimize } len(arc_1, arc_2, \cdots, arc_n)=\sum_{i=1}^{n} len(arc_i) \quad (7)$$

其中，*n* 是 TSP 问题的城市总数。

TSP 问题中，定义粒子维数等于城市总数。$E_i \in E$，*E* 表示候选弧集，E_i 表示弧集中从城市 *i* 出发的路径，每个路径对应一个索引号，如路径(1, 2)对应的索引号为 1，路径(1, 3)对应 2，以此类推。测试用例是 TSP 问题，城市数据从 TSPLIB 中抽取(http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/)，TSPLIB 包含了各种城市 TSP 数据。算法运行环境为：Pentium D 3.00 GHz CPU，1024MB RAM，操作系统为 MS Windows XP，编译器为 VC++ 6.0。

5.1 算法有效性验证

为验证算法的有效性，我们对 burma14 问题进行求解，结果如图 1、2 所示。

三个 PSO 算法的缩写描述如下：V1_CD_PSO(双扰动 PSO 算法)，V2_CD_PSO（取消初速度的双扰动 PSO 算法）和 Self-perception_

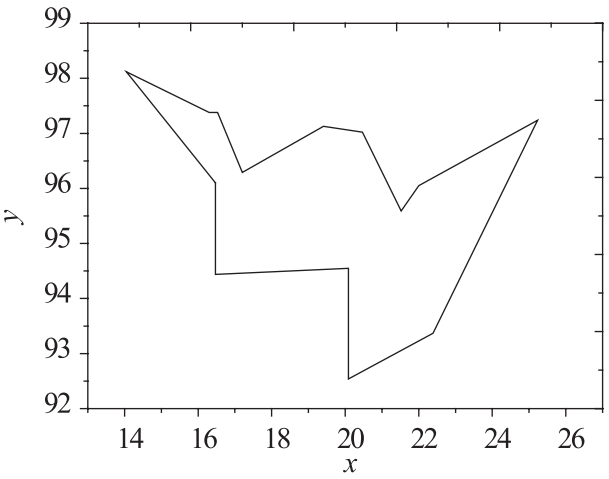

图 1　Self-perception_PSO 算法搜索到的最优路径

Fig. 1.　The optimal path of Self-perception_PSO

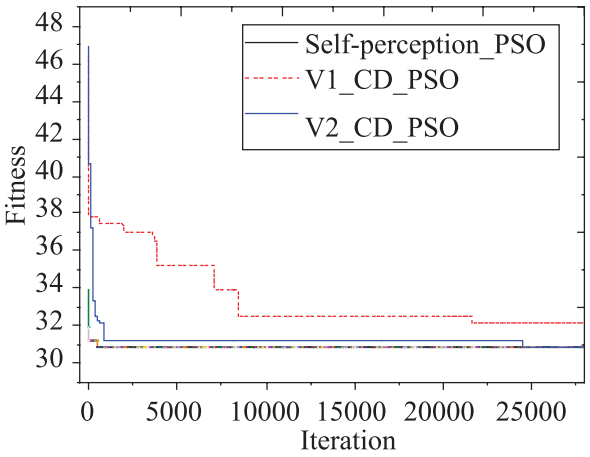

图 2　burma14 问题的三种算法比较

Fig. 2.　Comparing self-perception_PSO with the other PSO algorithms in burma14

PSO。

在求解的精度方面：

25000 代时，V2_CD_PSO 和 Self-perception_PSO 都能搜索到 TSP 的最优解，其解构成的城市路径轨迹如图 1 所示(著名最优解：30.8785)，V1_CD_PSO 不能搜索到著名最优解。

在求解的速度方面：

Self-perception_PSO 算法明显优于 V1_CD_PSO 和 V2_CD_PSO，其在 800 代左右即搜索到了最优值。

5.2　算法鲁棒性验证

为了进一步验证 Self-perception_PSO 算法的鲁棒性，对 6 个经典 TSP 问题进行了验证求解，对比求解的平均值(算法独立进行了 20 次实验)、最优解、最差解和运行时间(根据第一次达到最优解所花费的时间来计算)。实验结果如表 1 所示。通过实验我们发现，在前 5 个测试用例中都能获得最优解，但 KroA150 中 Self-perception_PSO 算法的实际最优解为 26781(著名最优解：

表 1　Self-perception_PSO 算法求解各类 TSP 问题

Table 1.　Self-perception_PSO in the different TSPs

TSP 测试用例（TSPLIB）	算法	平均值	运行时间(ms)	最差解	最优解	著名最优解
Burma14	Self-perception_PSO	31.09	108	32.64	30.87	30.87
Berlin52	Self-perception_PSO	7591.1	1399	8067	7542	7542
eil51	Self-perception_PSO	427.1	1447	440	426	426
eil76	Self-perception_PSO	552.1	2790	550	538	538
kroA100	Self-perception_PSO	21487.3	6089	22882	21282	21282
kroA150	Self-perception_PSO	27199.4	16877	27935	26781	26524

注：运行时间为 20 次 CPU 平均时间，解精度由 20 次独立实验的最优解、平均值和最差解来衡量。

26524)。

5.3 算法优越性验证

为了进一步验证 Self-perception_PSO 算法的优越性，我们对比了 Self-perception_PSO 算法与 RCPSO[3]、RHDPSO[10]、T-PSO[11]和 CLD-PSO[12]等 PSO 变体算法。其中，RCPSO、RHDPSO 为采用了扰动技术的 PSO 算法；T-PSO 算法是为解决多目标网格服务组合及最优选择问题(MO-MRSCOS)而提出的最小化时间和费用，最大化可靠性的算法；CLD-PSO 算法是 Comprehensive-Learning PSO(CLPSO)基于 RD-Rule 的离散版本。实验运行了 3 个测试用例，如表 2 所示，每个测试用例的平均值、最优解和平均 CPU 时间的最优解在表中用加粗字体显示。

5.4 实验结果小结

通过实验我们发现混沌 Logistic 时间序列扰动有助于粒子逃离局部极值，取得更好的性能；粒子历史速度阻碍了当前速度的改变，降低了粒子空间搜索能力。基于混沌双扰动的“自我”感知策略有利于粒子在高维离散空间搜索到最优解，同时也增强了 PSO 算法的性能。

6 结 论

本文提出了基于混沌双扰动的“自我”感知策略来增强 PSO 算法的整体性能并提升解的精度，该策略实现了 *pBest* 和 *gBest* 双扰动的精确时间、精确维数和精确的粒子的探测和“自我”

表 2 Self-perception_PSO 算法与其它算法的比较

Table 2. Comparing self-perception_PSO with the other existing PSO algorithms in the TSPs

TSP 测试用例（TSPLIB）		RCPSO	RHDPS	T-PSO	CLD-PS	Self-perception _PSO
Berlin52	平均值	7712.3	7814.2	7824.4	7728.9	**7591.1**
	Per*	101.59%	102.94%	103.07%	101.82%	
	最优解	**7542**	7589	7672	7609	**7542**
	Per*	**100%**	100.62%	101.72%	100.88%	
	运行时间(s)	1780	2536	2261	2754	**1399**
	Per*	127.23%	181.27%	161.62%	196.85%	
eil51	平均值	431.2	434.5	449.2	442.7	**427.1**
	Per*	100.96%	101.73%	105.17%	103.65%	
	最优解	**426**	**426**	**426**	**426**	**426**
	Per*	**100%**	**100%**	**100%**	**100%**	
	运行时间(s)	1587	3001	2785	2897	**1447**
	Per*	109.68%	207.39%	192.47%	200.21%	
kroA100	平均值	22048.3	22698	24087.9	23015.3	**21487.3**
	Per*	102.61%	105.63%	112.10%	107.11%	
	最优解	**21301**	21871	22011	22457	**21282**
	Per*	100.08%	102.77%	103.43%	105.52%	
	运行时间(s)	7065	14891	11561	12877	**6089**
	Per*	116.03%	244.56%	189.87%	211.48%	

注：算法性能由 20 次独立实验的最优解、平均值和运行时间来衡量；Per*为各算法求解值与 Self-perception_PSO 算法求解值的百分比。

感知。为了进一步验证该策略和相应的 PSO 算法性能，我们利用上述算法求解 TSP 问题并进行了算法性能比较。实验结果表明我们提出的策略简单且高效，基于混沌时间序列的“自我”感知策略，可用来防止早熟收敛，并增强 PSO 算法性能；Self-perception_PSO 算法性能优越。

参 考 文 献

[1] Selakov A, Cvijetinović D, Milović L, et al. Hybrid PSO–SVM method for short-term load forecasting during periods with significant temperature variations in city of Burbank [J]. Applied Soft Computing, 2014, 16: 80-88.

[2] Khan SA, Nadeem A. Automated test data generation for coupling based integration testing of object oriented programs using particle swarm optimization (PSO) [C] // Proceedings of the Seventh International Conference on Genetic and Evolutionary Computing, Advances in Intelligent Systems and Computing, 2014: 115-124.

[3] Tao Q, Chang H, Yi Y, et al. A rotary chaotic PSO algorithm for trustworthy scheduling of a grid workflow [J]. Computers & Operations Research, 2011, 38(5): 824-836.

[4] Wang H, Liu Y, Zeng SY. A hybrid particle swarm algorithm with Cauchy mutation [C] // IEEE Swarm Intelligence Symposium, 2007: 356-360.

[5] Krohling RA, Mendel E. Bare bones particle swarm optimization with Gaussian or Cauchy jumps [C] // IEEE Congress on Evolutionary Computation, 2009: 3285-3291.

[6] Tan Y, Xiao ZM. Clonal particle swarm optimization and its applications [C] // IEEE Congress on Evolutionary Computation, 2007: 2303-2309.

[7] Krohling RA. Gaussian particle swarm with jumps [C] // The 2005 IEEE Congress on Evolutionary Computation, 2005, 2: 1226-1231.

[8] Sun Y, Zhang WG, Zhang M, et al. Design of neural network gain scheduling flight control law using a modified PSO algorithm based on immune clone principle [C] // Second International Conference on Intelligent Computation Technology and Automation, 2009, 1: 259-263.

[9] Liu B, Wang L, Jin YH, et al. Improved particle swarm optimization combined with chaos [J]. Chaos, Solitons & Fractals, 2005, 25(5): 1261-1271.

[10] Tao Q, Chang HY, Yi Y, et al. QoS constrained grid workflow scheduling optimization based on a novel PSO algorithm [C] // Eighth International Conference on Grid and Cooperative Computing, 2009: 153-159.

[11] Tao F, Zhao DM, Hu YF, et al. Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system [J]. IEEE Transactions on Industrial Informatics, 2008, 4(4): 315-327.

[12] Liang JJ, Qin AK, Suganthan PN, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions [J]. IEEE Transactions on Evolutionary Computation, 2006, 10(3): 281-295.