一种基于单纯形法的改进微粒群优化算法及其收敛性分析

张 勇 [1] 巩敦卫 [1] 张婉秋 [1]

摘 要 针对现有微粒群优化算法难以兼顾进化速度和求解质量这一难题, 提出一种基于单纯形法的改进微粒群优化算法 (Simplex method based improved particle swarm optimization, SM-IPSO). 该算法采用多个优化种群, 分别在奇数种群和偶数种群上并行运行微粒群算法和单纯形法, 并通过周期性迁移相邻种群间的最优信息, 达到微粒群算法和单纯形法的协同搜索: 单纯形借助微粒群算法跳出局部收敛点, 微粒群依靠单纯形提高局部开发能力. 为强化两种算法所起作用, 一种改进的微粒速度逃逸策略和 Nelder-Mead 单纯形法也被提出. 最后, 在 Linux 集群系统上运行所提算法, 通过优化五个典型测试函数验证了算法的有效性.

关键词 并行, 微粒群优化, 单纯形法, 多种群, 速度逃逸
中图分类号 TP301

A Simplex Method Based Improved Particle Swarm Optimization and Analysis on Its Global Convergence

ZHANG Yong[1] GONG Dun-Wei[1] ZHANG Wan-Qiu[1]

Abstract Considering that the existing particle swarm optimizations (PSO) do not give simultaneously attention to evolution speed and solution′s quality, a simplex method based improved particle swarm optimization (SM-IPSO) is proposed in this paper. In SM-IPSO, the conception of multipopulations is adopted, where PSO and SM run on odd populations and even populations, respectively. And a periodical migrating operation between adjacent populations is also introduced in SM-IPSO in order to achieve cooperative search of both PSO and SM for solution space: SM can get away from local converged points by virtue of PSO, and PSO can improve its local exploiting capability under the help of SM. Furthermore, an improved escape method of particle velocities and improved Nelder-Mead SM are proposed in order to enhance the functions of PSO and SM in this paper. Finally, the proposed algorithm is implemented on a Linux cluster system, and experimental results on optimizing five benchmark functions demonstrate its usefulness.

Key words Parallel, particle swarm optimization (PSO), simplex method (SM), multi-population，velocity escape

微粒群优化 (Particle swarm optimization, PSO)[1] 作为一种新的进化优化技术, 是由 Kennedy 等首先提出的. 由于该算法结构简单, 易于实现, 已被广泛用于函数优化、神经网络训练和模糊控制等众多科学领域[2]. 然而, 近年来随着科学技术的不断发展, 面对复杂程度越来越高的优化问题, PSO 在优化速度和求解质量上都显得 “力不从心”.

并行微粒群优化算法 (Parallel particle swarm optimization, PPSO) 作为 PSO 研究的新兴热点, 将计算机的高速并行计算能力和 PSO 的天然并行性相结合, 显著提高了 PSO 解决复杂工程优化问题的能力. 通常 PPSO 可分为全局式、扩散式和迁移式三类[3]. 全局式 PPSO[4−5], 因其主从处理器间通信频繁, 模型运算效率不高, 一般适用于函数评价工作量非常大的情况[6]; 扩散式 PPSO 尽管能最大限度地发挥算法的并行潜力, 但其实现对处理器数量要求很高; 迁移式 PPSO 将随机生成的初始微粒群依处理器个数分割成若干个子微粒群, 各个子微粒群在不同的处理器上并行进化. 相对前两种模型, 此类模型不仅通信代价较小, 而且非常适合在通信带宽较低的集群上运行[6], 是适应性强且应用范围广的并行模型.

但是同 PSO 一样, 迁移式 PPSO 依旧存在早熟现象. 克服算法早熟, 一种最为直接的方法就是把已有 PSO 改进算法嵌入 PPSO. 此时, 尽管 PSO 能最终求得理想解, 但在求解复杂问题中, 一种进化计算技术的实际利用会受到收敛速度慢所带来的高计算花费的严重制约[7]. 事实上, PSO 的收敛速度是明显慢于一些局部搜索算法的, 因为这些方法在确定一个最有希望的搜索方向时不需要利用太多的位置信息[8].

基于此, 许多学者尝试把单纯形法(Simplex method, SM)[9], 这一典型局部搜索算法嵌入到

收稿日期 2007-12-24 收修改稿日期 2008-10-13
Received December 24, 2007; in revised form October 13, 2008
国家自然科学基金 (60775044), 江苏省自然科学基金 (BK2008125), 江苏省普通高校研究生科研创新计划 (CX07B-115Z) 资助
Supported by National Natural Science Foundation of China (60775044), Natural Science Foundation of Jiangsu Province (BK2008125), and Graduate Student Research Innovation Program of Jiangsu Province College (CX07B-115Z)
1. 中国矿业大学信息与电气工程学院 徐州 221008
1. School of Information and Electronic Engineering, China University of Mining and Technology, Xuzhou 221008
DOI: 10.3724/SP.J.1004.2009.00289

PSO 中, 以提高其局部寻优能力, 具体方法包括两类: 一类是, 每隔固定代数对微粒群中最优点[10]、部分微粒或全部微粒[11−12] 进行若干代单纯形搜索. 此类方法可以显著提高 PSO 的局部搜索能力, 但存在算法全局寻优能力不理想或计算代价过大等缺点. 另一类是, 利用微粒周期性地构造单纯形的顶点, 完成对微粒群全局最优点的深度开发[8, 13]. 此类方法皆采用串联方式混合 PSO 和 SM, 这在一定程度上限制了它们对大规模复杂优化问题的处理能力.

为同时提高 PSO 的进化速度和质量, 本文提出一种基于 SM 的改进微粒群优化算法. 该算法通过在奇数种群和偶数种群上分别并行协同运行 PSO 和 SM, 以期达到算法全局和局部寻优能力的有效均衡.

1 相关知识

1.1 微粒群优化算法

PSO 源于对鸟类和鱼群捕食等行为的模拟. 在鸟类捕食的群体行为中, 每只鸟被看作一个微粒, 而每个微粒代表一个被优化问题的解. 在 D 维搜索空间中, 设微粒 $\boldsymbol{x}_i$ 本身所找到的最佳位置为 $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ (一般称其为微粒个体最优点), 整个微粒群迄今为止搜索到的最佳位置为 $\boldsymbol{p}_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$ (一般称为微粒群全局最优点), 微粒的当前速度为 $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$, 那么, 每个微粒将根据下式来调整自己下一步的位置[14].

$$v_{ij}(t+1) = wv_{ij}(t) + r_1c_1(p_{ij}(t) - x_{ij}(t)) + r_2c_2(p_{gj}(t) - x_{ij}(t)) \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

其中, t 为迭代次数, w 为惯性权值, c_1, c_2 为学习因子, r_1, r_2 为 $[0, 1]$ 之间的随机数. 此外, 为防止微粒远离搜索空间, 一个限制微粒飞行的最大速度 $\boldsymbol{V}_{\max}$ 也被定义, 其值通常取决策变量的上下界之差.

1.2 迁移式 PPSO 算法

对于迁移式 PPSO 算法, 已有研究主要集中在拓扑结构、迁移策略、迁移比率和迁移间隔等方面.

改进传统岛屿型拓扑结构, 黄芳等提出一类主从式岛屿模型[15], 以加快优质解在子微粒群中的传播速度. Matsumura 等利用大量实验发现: 单向环拓扑既保证了优良基因在群体间的扩散, 又较好地保护了群体间的多样性, 该拓扑结构虽然收敛速度慢, 但解的质量较高[16].

改进传统微粒群间信息的全局同步迁移策略, Li 等提出一种延时信息交换策略[17], 以节省微粒群间的等待时间, 但不利于优势信息的及时利用. El-Abal 等通过实验分析指出, 在仅替换微粒个体最优点的情况下, "选择最优微粒替代随机微粒" 法有较快的收敛速度, 而 "选择随机微粒替代随机微粒" 法能得到高质量的解[18].

对于迁移比率和迁移间隔, 文献 [19] 通过大量实验发现, 过多以及过频的迁移会破坏种群的多样性, 致使多个搜索进程集中到相同的区域, 不利于提高解的质量; 过少以及过低频率的迁移, 使各种群不能充分利用其他群体信息, 同样不利于提高解的质量.

由此可见, 仅仅针对微粒群间迁移算子进行改良, 所得成果大都具有两面性, 对提高 PPSO 算法的整体性能是有限的.

2 基于 SM 的改进微粒群优化算法: SM-IPSO

2.1 迁移式 PPSO 的早熟分析

分析导致迁移式 PPSO 早熟的原因, 首先重述文献 [20] 所给出的算法收敛定理.

定理 1. 若按期望值观察 PSO 的行为, 当 $0 < c_1 + c_2 < 4(1 + w)$ 且 $|w| < 1$ 时,

$$\mathrm{E}\boldsymbol{x}_i(t) = \begin{cases} [k_1(t-1) + k_2]\boldsymbol{\lambda}_{E1}^{t-1} + \boldsymbol{\mu}, & \text{若 } \boldsymbol{\lambda}_{E1} = \boldsymbol{\lambda}_{E2} \\ k_3\boldsymbol{\lambda}_{E1}^{t} + k_4\boldsymbol{\lambda}_{E2}^{t} + \boldsymbol{\mu}, & \text{否则} \end{cases}$$

其中 λ_{E1}, λ_{E2} 为 $\mathrm{E}Y_i(t+1) = [1 + w - 0.5(c_1 + c_2)]\mathrm{E}Y_i(t) - w\mathrm{E}Y_i(t-1)$ 的特征根, $\boldsymbol{\mu} = [c_1\boldsymbol{p}_i + c_2\boldsymbol{p}_g]/[c_1 + c_2]$, $k_1, k_2, \cdots, k_4$ 为常数.

定理 1 的重要性不仅在于给出了 PSO 收敛的充分条件, 而且刻画出了算法的收敛速度: 微粒群中每个微粒皆以指数级速度收敛到 $\boldsymbol{p}_i$ 和 $\boldsymbol{p}_g$ 间的某一均衡点. 然而, 对 PPSO 而言, 此机制也加速了微粒群自身有效信息的丢失. 对于多峰问题, 很多情况下问题的最优解可能位于某一微粒个体最优点附近. 此时, 对该微粒进行局部搜索会增加算法找到全局最优点的概率. 事实上, PSO 的快速寻优能力, 将使上述个体最优点很快被新的微粒位置所替代, 尤其是迁入微粒优于当前微粒群中最优微粒时, 导致微粒群丧失继续开发最优解潜在区域的机会.

图 1 (见下页) 展示了一个简单的函数最大化问题. 受迁入微粒信息的影响, 微粒 $\boldsymbol{x}_1$ 将以较大概率趋近并落入阴影区域, 进而丧失对最高峰的开发; 相反, 此时如果采用某种局部搜索算法, 如单纯形法对 $\boldsymbol{p}_1$ 继续进行深度开发, 将显著提高 PSO 找到全局最优点的概率.

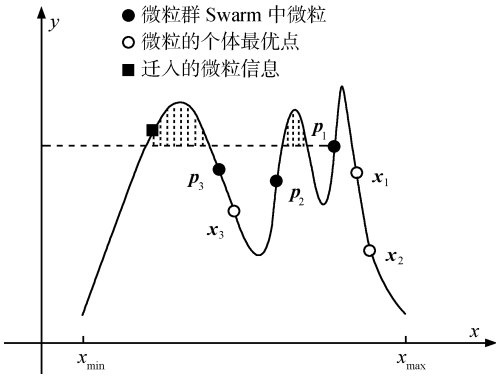


图 1　迁入微粒与算法性能间关系

Fig. 1　Relationship between migrating particles and algorithm′s performance

另一方面, 如 Li 所说 "传统 PSO 不是一种局部优化者, 它不能保证所得解为问题的局部最优解"[21]. 例如, 当 $\boldsymbol{x}_i(t) = \boldsymbol{p}_i(t) = \boldsymbol{p}_g(t)$ 时, 如果 $\boldsymbol{v}_i(t)$ 不为 $\mathbf{0}$, $\boldsymbol{x}_i(t)$ 将仅依靠 $w\boldsymbol{x}_i(t)$ 更新其位置; 如果微粒速度非常接近 $\mathbf{0}$, 此时一旦微粒们到达微粒群全局最优点 (该点不一定为问题的局部最优解), 它们将全部停止运动, 从而导致算法的早熟收敛[22].

2.2　SM-IPSO 算法的思想

为克服迁移式 PPSO 的上述缺点, 本文提出一种基于 SM 的改进微粒群优化算法, 其思想之一是: 对 PSO 所得解定期利用 SM 进行深度开发, 这样不仅可以提高解的精度, 而且可以增加算法对最优解潜在区域的开发机会; 其思想之二是: 通过 PSO 和 SM 的并行协同搜索, SM 利用 PSO 跳出局部最优, PSO 依靠 SM 迅速找到问题最优点, 可以做到算法探索与开发的有效均衡. 此外, 同其他并行算法一样, 多个种群同时搜索也提高了算法跳出局部收敛点的可能性.

图 2 展示了基于单纯形法的改进微粒群优化算法 (Simplex method based improved particle swarm optimization, SM-IPSO) 的算法模型. 在 SM-IPSO 中种群被分为奇数和偶数种群两类, 其中奇数种群上执行改进的 PSO, 偶数种群上执行改进的 NM-SM (Nelder-Mead simplex method). 当奇数种群中 PSO 满足迁移条件时, 向邻近的偶数种群迁移最优微粒信息, 以便让 NM-SM 对其继续进行深度开发; 同时, 偶数种群将经 NM-SM 开发的最优顶点传给邻近的奇数种群, 由 PSO 对其进行广度探索, 以帮助它跳出局部极值点; 如此继续, 直到算法收敛.

图 3 (见下页) 展示了 SM-IPSO 的具体流程, 可以看出, SM-IPSO 主要由改进 PSO、改进 NM-SM 和个体迁移操作三部分组成. 不同于传统 PSO 和 NM-SM, 本文所提改进方法分别增加了一个微粒速度的概率逃逸算子和单纯形顶点的强制扩张算子, 其目的在于完善两种算法在 SM-IPSO 中所起的作用, 即分别增强 PSO 的全局寻优能力和 NM-SM 对迁入微粒的快速响应能力, 具体方法参见第 2.3 节和 2.4 节.

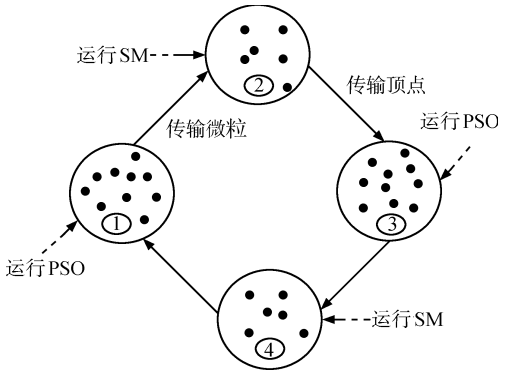


图 2　SM-IPSO 算法模型

Fig. 2　Model of SM-IPSO algorithm

个体迁移操作起着协调 SM-IPSO 全局和局部开发能力的重要作用. 为保证算法有较快的收敛速度, 本文采取 "微粒群中最优位置替代单纯形中随机顶点, 或单纯形中最优顶点替代微粒群中随机个体最优点" 这一迁移策略. 而算法的个体迁移间隔 Δ 由 PSO 决定, 当奇数种群满足迁移条件, 即 $\lceil t/\Delta \rceil = t/\Delta$ 时, PSO 和临近单纯形即刻进入迁移状态. 文章第 4.3 节定量分析了 Δ 与算法性能间关系, 指出 Δ 取 10 时最理想.

2.3　改进的 PSO 算法

为提高 PSO 的全局寻优能力, 赫然等结合生物界中发现生存密度过大时物种会自动迁移这一习性, 提出了自适应逃逸微粒群算法 (Self-adaptive escape particle swarm optimization, AEPSO)[23]. 该方法显著提高了算法跳出局部极值的能力, 但设置合理的逃逸阈值是较难的.

受此启发, 本节给出一种改进算法. 该算法赋予微粒 $\boldsymbol{x}_i(t)$ 一个运动阈值 Z, 当微粒速度 $\|\boldsymbol{v}_i(t)\| \le Z$ 时, 给其一个变异操作 — 概率逃逸运动:

$$v_{ij}(t) = N(0,1) \times v_{\max,j}, \quad \text{若 } r_3 < p_c$$

其中, $j = 1, 2, \cdots, D$, p_c 为变异概率, $N(0, 1)$ 为标准正态分布函数, r_3 为 [0, 1] 间随机数. 逃逸后的微粒将记住微粒群全局最优点, 而忘记先前个体最优点. 对于决定速度逃逸的阈值 Z, 因为在 SM-IPSO 中, 改进 PSO 侧重于问题的全局搜索, 所以, 无需对其进行缩减, 直接赋予常值即可. 而对 PSO 所得解的深度开发将由 NM-SM 完成, 进而达到两种算法的有效分工.

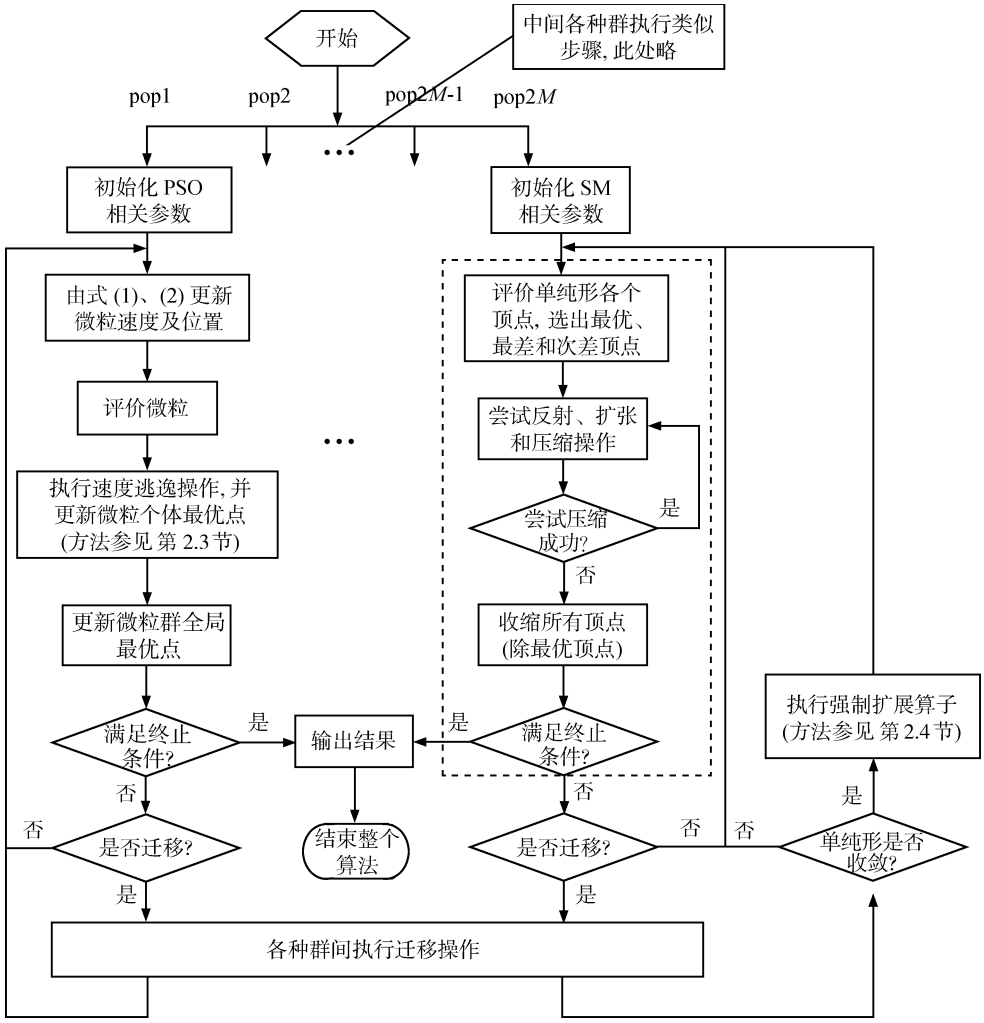


图 3 SM-IPSO 算法流程图

Fig. 3 Flow diagram of SM-IPSO algorithm

2.4 改进的 NM-SM

在 SM-IPSO 中, 尤其是算法末期, 运用 NM-SM 对 PSO 所提供微粒信息进行深度开发时, 经常遇到执行个体迁移操作前, 单纯形已在第 d 维变量空间上收敛的情况. 此时, 如果迁入微粒恰好位于此收敛点附近, 则迁移后单纯形 (即并入迁入微粒信息后的单纯形) 将无法对第 d 维变量空间进行有效开发; 相反, 如果迁入微粒远离单纯形的收敛点, 则单纯形将很难在短时间内对迁入微粒进行深度开发. 上述情况的存在, 将直接降低 NM-SM 对 PSO 所提供微粒信息的利用效率, 进而影响算法的整体性能.

本文通过引入一个强制扩张算子, 来提高 NM-SM 对迁入微粒的利用效率, 具体方法如下: 当偶数种群检测到有微粒迁入时, 即刻判断单纯形在各维变量空间上是否收敛, 如果收敛, 则扩张单纯形.

假设 $S = \{s_1, s_2, \cdots, s_{D+1}\}$ 为迁移前单纯形顶点集, q 为迁入微粒信息, S' 为迁移后单纯形顶点集, e 为判断单纯形是否收敛的阈值, 则强制扩张算子可描述如下:

步骤 1. 初始化 $d = 1$;

步骤 2. 判断单纯形 S 在第 d 维变量空间上是否收敛. 如果 $\max_{s_i \in S} s_{id} - \min_{s_j \in S} s_{jd} \leq e$, 则称单纯形 S 在 d 维变量空间上收敛; 否则, 终止算子;

步骤 3. 由

$$s'_{ld} = \begin{cases} q_d + N(0,1)Z, & \min\limits_{s_j \in S} s_{jd} \leq q_d \leq \max\limits_{s_i \in S} s_{id} \\ s'_{ld} + 2r_4(q_d - s'_{ld}), & \text{otherwise} \end{cases} \tag{3}$$

重置 S' 中所有顶点, 其中, $s'_l \in S', l = 1, 2, \cdots, D+1$, r_4 为 $[0, 1]$ 间随机数;

步骤 4. $d = d + 1$, 如果 $d \leq D$, 转入步骤 2.

2.5 算法收敛性分析

Solis 等[24] 提出了随机优化算法以概率 1 收敛于全局最优解的充分条件, 为分析方便, 下面重述其主要结论.

假设 1. F s.t. $\{f(x^t)\}_{t=0}^{\infty}$ 是非增的, 即如果 $f(F(x,\xi)) \le f(x)$ 且 $\xi \in U$, 则 $f(F(x,\xi)) \le \min\{f(x),\ f(\xi)\}$. 其中, F 为算法用于产生新个体的迭代函数, U 为变量可行域.

假设 2. 对于 U 的任意 Borel 子集 A, 若其测度 $\theta(A) > 0$, 则有 $\prod_{t=0}^{\infty}(1-\mu_t(A)) = 0$ 成立, $\mu_t(A)$ 为由采样策略得到的 A 中部分元素的概率.

引理 1[24]**.** 设目标函数 f 是可测函数, U 为可测子集, $\{x^t\}_{t=0}^{\infty}$ 为算法所生成的解序列, 并且假设 1 和假设 2 成立, 则 $\lim_{t\to\infty} p\,(x^t \in R_\varepsilon) = 1$ 成立. 其中, $p(x^t \in R_\varepsilon)$ 为第 t 次迭代算法生成解 $x^t \in R_\varepsilon$ 的概率, R_ε 为全局最优点集合.

定理 2. 假设 PSO 求解的目标函数 f 是可测函数, 其解空间 U 为可测集, 那么, 改进的 PSO 算法以概率 1 收敛于全局最优解.

证明. 由引理知, 只需证明改进的 PSO 满足假设 1 和假设 2 即可.

1) 改进 PSO 的迭代函数 F 可归结为

$$F(\boldsymbol{p}_g(t),\boldsymbol{p}_i(t)) = \begin{cases} \boldsymbol{p}_g(t), & f(\boldsymbol{p}_g(t)) \le f(\boldsymbol{x}_i(t)) \\ \boldsymbol{x}_i(t), & f(\boldsymbol{p}_g(t)) > f(\boldsymbol{x}_i(t)) \end{cases} \tag{4}$$

其中, t 为进化代数, 算法的解序列为 $\{\boldsymbol{p}_g(t)\}_{t=0}^{\infty}$. 容易证明式 (4) 满足假设 1.

2) 令 $\boldsymbol{x}_i^k$ 为微粒 $\boldsymbol{x}_i$ 第 k 次逃逸所得结果, 由逃逸算子 $x_{id} = x_{id} + N(0,1)v_{\max,d}$ 可得

$$\boldsymbol{x}_i^k \sim N(\boldsymbol{x}_i, \boldsymbol{V}_{\max}^2) \tag{5}$$

对于算法的产生矩阵, 如果一直执行逃逸算子, 则有

$$\boldsymbol{x}_i^{k+1} \sim N(\boldsymbol{x}_i^k, \boldsymbol{V}_{\max}^2) \tag{6}$$

此时, 由文献 [24] 中结论可知, n 个微粒样本空间的并集必然包含 U, 即 $U \subseteq \bigcup_{i=1}^{n} M_{i,k}$ 其中, $M_{i,k}$ 为第 k 次逃逸微粒 $\boldsymbol{x}_i$ 的样本空间的支撑集.

又因为在改进 PSO 中微粒速度更新公式是收敛的, 所以, 在微粒逃逸阈值的作用下, 微粒 $\boldsymbol{x}_i$ 两次逃逸的间隔代数是有限的; 并且两次逃逸间隔内微粒位置的更新对其逃逸结果没有任何影响. 因此, 对于 $\forall A \subset U$ 只要其测度 $\theta(A) > 0$, 改进 PSO 中必有微粒到达 A, 即满足假设 2. □

定理 3. SM-IPSO 以概率 1 收敛于全局最优解.

证明. 鉴于迁移操作有传递最优信息的作用, 只需证明改进 PSO 仍以概率 1 收敛于全局最优解即可. 此时算法收敛性证明与定理 2 的唯一区别是: 在迁入最优顶点的影响下, 微粒 $\boldsymbol{x}_i$ 两次逃逸的间隔代数是否有限. 因此, 只需证明式 (1) 满足 $\boldsymbol{v}_i(t) = 0$ $(t \to \infty)$ 即可. 不失一般性, 首先给出如下假设:

1) SM-IPSO 算法只包含两个种群;

2) $\boldsymbol{s}_{low}((k_5+1)\Delta) \approx \boldsymbol{p}_g(k_5\Delta)$, 其中 $\boldsymbol{p}_g(k_5\Delta)$ 为第 $k_5\Delta$ 代微粒群最优点, $\boldsymbol{s}_{low}((k_5+1)\Delta)$ 为第 k_5+1 次迁移前单纯形最优顶点.

由上述假设, 改进 PSO 中式 (1) 可写为

$$\begin{aligned} v_{ij}(t+1) =& wv_{ij}(t) + r_1c_1(p'_{ij}(t) - x_{ij}(t)) + \\ & r_2c_2(p_{gj}(t) - x_{ij}(t)) \end{aligned}$$

$$p'_{ij}(t) = \begin{cases} p_{ij}(t), & \left\lceil \dfrac{t}{\Delta} \right\rceil \ne \dfrac{t}{\Delta} \\ p_{gj}(t-\Delta), & \left\lceil \dfrac{t}{\Delta} \right\rceil = \dfrac{t}{\Delta} \end{cases} \tag{7}$$

又因为函数 f 有界可测且 $f(\boldsymbol{p}_g(t)) \le f(\boldsymbol{p}_g(t+1))$, 所以

$$\lim_{t\to\infty}[\boldsymbol{p}_g(t) - \boldsymbol{p}_g(t-\Delta)] = 0 \tag{8}$$

进而,

$$p'_{ij}(t) = \begin{cases} p_{ij}(t), & \left\lceil \dfrac{t}{\Delta} \right\rceil \ne \dfrac{t}{\Delta} \\ p_{gj}(t) + \varepsilon(t), & \left\lceil \dfrac{t}{\Delta} \right\rceil = \dfrac{t}{\Delta} \end{cases} \tag{9}$$

其中, $\lim_{t\to\infty}\varepsilon(t) = 0$, 将式 (9) 代入式 (7) 得

$$v_{ij}(t{+}1) = \begin{cases} wv_{ij}(t) + (r_1c_1 + r_2c_2)\times \\ (p_{gj}(t) - x_{ij}(t)) + \varepsilon(t), & \left\lceil \dfrac{t}{\Delta} \right\rceil = \dfrac{t}{\Delta} \\ wv_{ij}(t) + r_1c_1(p_{ij}(t) - x_{ij}(t)) + \\ r_2c_2(p_{gj}(t) - x_{ij}(t)), & \left\lceil \dfrac{t}{\Delta} \right\rceil \ne \dfrac{t}{\Delta} \end{cases} \tag{10}$$

显然, 此时, 改进 PSO 算法可近似为基本 PSO 和仅含社会 (Social) 项 PSO 的混合, 而上述两种 PSO 模型中微粒速度是收敛的. 因此, $\boldsymbol{v}_i(t) = 0\,(t \to \infty)$. □

3 算法并行设计及加速度比分析

3.1 算法并行设计

采用基于消息传递的并行编程环境 MPI (Message passing interface)[25] 实现 SM-IPSO 算法, 具体方法如下:

步骤 1. 设定算法参数, 包括种群数量 $2M$、微粒群或单纯形规模; PSO 操作算子和 NM-SM 系数 {反射系数 α, 收缩系数 β, 扩张系数 γ, 压缩系数 δ}; 个体迁移间隔 Δ 和算法中止策略等.

步骤 2. 随机初始化微粒群中微粒位置及速度, 单纯形的顶点, 微粒群进化代数 $t = 1$.

步骤 3. 初始化并行环境:

MPI_Init (&argc, &argv), 初始 MPI 运行环境;

MPI_Comm_rank (MPI_COMM_WORLD, &my_rank), 取得进程序号;

MPI_Comm_size (MPI_COMM_WORLD, &size), 取得进程个数.

步骤 4. 循环如下步骤, 直到满足算法终止条件:

步骤 4.1. 各进程同时运行改进的 PSO 和 NM-SM;

步骤 4.2. 如果 $t/\Delta \neq \lceil t/\Delta \rceil$, 令 $t \leftarrow t+1$, 并返回步骤 4.1;

步骤 4.3. 设置 MPI_Barrier() 函数, 确保所有进程同步;

步骤 4.4. 从各种群中选出微粒群全局最优点或单纯形最优顶点;

步骤 4.5. 各进程之间以单向环方式迁移所选个体, 不妨以第 i 个进程为例:

MPI_Send (&individual, 1, MPI_DOUBLE, $i+1$, tag, MPI_COMM_WORLD), 从第 i 个进程发送最优微粒或顶点到进程 $i+1$;

MPI_Recv (&individual, 1, MPI_DOUBLE, $i-1$, tag, MPI_COMM_WORLD, &status) 进程 i 接受从进程 $i-1$ 中迁移过来的最优微粒或顶点;

用迁入微粒或顶点替代所选随机顶点或微粒的个体最优点;

步骤 4.6. 判断单纯形是否收敛, 如果收敛, 对单纯形执行强制扩张算子;

步骤 4.7. 令 $t \leftarrow t+1$, 返回步骤 4.1.

3.2 加速比分析

设集群的计算节点数为 $2M$, 从以上算法步骤可知, 改进 PSO 每迭代 Δ 次, SM-IPSO 进行一次节点通信, 不妨设这个通信代价为 T_c. SM-IPSO 串行执行时每迭代 Δ 次, 单个种群运行改进 PSO 的代价设为 T_{pso}, 运行改进 NM-SM 的代价设为 T_{sm}, 则可认为算法串行计算所需代价为 $T_s = M(T_{pso}+T_{sm}) \times t_{\max}/\Delta$, 算法并行计算代价为 $T_p = t_{\max}/\Delta \times [T_c + \max\{T_{pso}, T_{sm}\}]$, 其中 $t_{\max}$ 为 SM-IPSO 终止时改进 PSO 的迭代次数. 由此, 算法的加速比为

$$\lambda = \frac{T_s}{T_p} = \frac{M(T_{pso}+T_{sm}) \times \dfrac{t_{\max}}{\Delta}}{\dfrac{t_{\max}}{\Delta} \times [T_c + \max\{T_{pso}, T_{sm}\}]} \geq \frac{M(T_{pso}+T_{sm})}{T_c + T_{pso} + T_{sm}} = \frac{\left\lceil \dfrac{T_{pso}+T_{sm}}{T_c} \right\rceil}{\left\lceil \dfrac{1}{M} + \dfrac{T_{pso}+T_{sm}}{M \times T_c} \right\rceil} \quad (11)$$

由上式可知, 用于算法计算的节点越多, 算法加速比越大; 相对改进 PSO 和 NM-SM 的自身计算代价, 算法用于通信的代价越小, 即 $(T_{pso}+T_{sm})/T_c$ 比值越大, 算法加速比越大. 特别地, 当问题的评价代价较大时, $T_c \ll T_{pso}+T_{sm}$ 算法可获得最佳加速比.

4 实验分析

测试 SM-IPSO 的性能, 设计 2 组实验: 1) 定量分析种群数量和微粒逃逸阈值等参数对算法性能的影响; 2) 选取全息 PSO (Fully informed PSO, FIPSO)[26]、AEPSO[23]、NM-PSO (Hybrid algorithm based on Nelder-Mead simplex method and particle swarm optimization)[8] 和迁移式 PPSO[3] (Migration parallel particle swarm optimization, MPPSO) 作为对比算法, 验证 SM-IPSO 的优越性.

4.1 测试环境设置

选取 Quadric、Rosenbrock、Rastrigrin、Griewank 和 Ackley 等常用 Benchmark 函数[2, 8, 10−11, 14−15]. 为避免算法性能受初始种群位置的影响, 前三个函数选取不对称形搜索空间, 后两个函数选取对称形搜索空间. 函数维数、搜索范围和理论最优解见表 1.

表 1 典型测试函数

Table 1 Benchmark functions

Function	Dim$D$	$[x_{\min}, x_{\max}]$	Optimal
Quadric	30	$[-50, 100]^{30}$	$0, (0, \cdots, 0)$
Rosenbrock	30	$[-30, 60]^{30}$	$0, (1, \cdots, 1)$
Rastrigrin	30	$[-5.12, 10.24]^{30}$	$0, (0, \cdots, 0)$
Griewank	100	$[-600, 600]^{100}$	$0, (0, \cdots, 0)$
Ackley	100	$[-30, 30]^{100}$	$0, (0, \cdots, 0)$

设置算法参数, FIPSO 选择性能较好的 URing + wFIPS 组合, 其他参数同文献 [26]; 取 AEPSO2 算法, 其参数同文献 [23]; NM-PSO 采用文献 [8] 中参数; SM-IPSO 参数设置如下: $w = 0.79$, $c_1 = c_2 = 1.47$, $\alpha = 1.0$, $\beta = \delta = 0.5$, $\gamma = 2.0$, $p_c = 0.1$, $Z = 0.01$, Number of swarms = Number of simplexes = 3, swarm size = 10, 其中

微粒逃逸半径 Z、逃逸概率 p_c 和迁移间隔 Δ 等取值的合理性将在第 4.3 节加以验证; MPPSO 中相关参数设置与 SM-IPSO 相同. 此外, 五种算法设置相同的函数评价次数 300 000 或迭代次数 5 000 作为强制终止条件.

4.2　算法的性能测度

采用文献 [27] 所列测度评价 SM-IPSO 的有效性: 1) 所得结果满足规定阈值时, 算法所需平均函数评价次数. 此测度为度量算法进化速度的重要指标. 对最大函数评价次数内仍不能满足规定阈值的实验, 不参与此测度计算. 由于优化高维函数时, 上述五种算法的计算花费主要集中在微粒适应值的计算上, 因此, 此测度还可反映算法优化问题时的计算代价. 文中 Rosenbrock 和 Rastrigrin 所对应阈值取 1, 其他函数取 0.01. 2) 达标率. 通过统计算法到达规定阈值的实验次数占总实验次数的比例, 测试其可靠性.

4.3　参数敏感性分析

通过观察参数变化对 SM-IPSO 性能的影响, 以确定其最佳参数设置. 实验以第 4.1 节中参数值为基础, 每次仅在特定区间内改变 SM-IPSO 中某一参数.

利用 SM-IPSO 分别优化五个测试函数 20 次, 表 2 展示了对迁移间隔的敏感性分析. 考虑到迁移间隔过大会严重降低算法的全局搜索性能, 实验中设置 $\Delta \leq 100$. 从表 2 可以看出, 迁移间隔取 5 或 10 时算法具有好的达标率, 但考虑到迁移间隔较小会增加算法通信代价, Δ 取 10 是合理的.

表 2　迁移间隔的敏感性分析

Table 2　Sensitivity analysis of migrating interval

Δ	达标率 (%)				
	Quadric	Rosenbrock	Rastrigrin	Griewank	Ackley
1	100	90	55	95	50
5	100	100	70	100	85
10	100	85	90	100	100
25	100	25	40	100	80
50	85	0	25	100	55
100	40	0	10	90	30

逃逸阈值 Z 和概率 p_c 作为决定改进 PSO 全局搜索能力的重要参数, 其值设置过大或过小对提高算法性能都是不利的. 这是因为, Z 设置过大, 改进 PSO 将近似为随机搜索算法; 而 Z 设置过小, 改进 PSO 算法又近似于基本 PSO. 同样, 逃逸概率设置过小, 微粒不易跳出局部最优点; 而逃逸概率设置越大, 算法已得优秀模式被破坏概率越大. 实验证明, $Z = 0.01$, $p_c = 0.1$ 时, SM-IPSO 可取得较好的达标率.

图 4 展示了种群个数与算法结果之间变化曲线. 可以看出, SM-IPSO 的种群个数在 2 到 12 之间取值时, M 值越大, 算法所得平均结果越好; 当 $M \geq 3$ 时, SM-IPSO 优化五个测试函数皆能得到满足规定阈值的满意解. 因此, 第 4.4 节将在 M 取值为 3 的情况下, 比较 SM-IPSO 与其他算法性能的优劣.

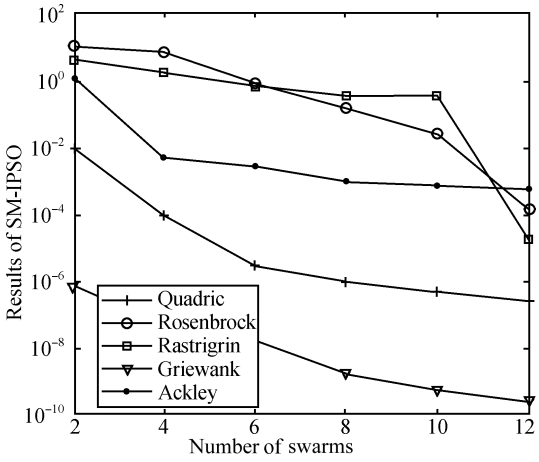


图 4　种群个数与算法平均结果之间的关系

Fig. 4　Relationship between number of swarms and average results of algorithm

4.4　算法性能对比

与 FIPSO、AEPSO、NM-PSO 和 MPPSO 等对比, 验证 SM-IPSO 的有效性. 表 3 (见下页) 展示了 5 种算法分别优化 5 个函数 20 次所得平均结果及标准差, 其中最佳结果已用粗体标出. 由表 3 可知: 1) 5 种算法中, 只有 SM-IPSO 所得平均结果均满足所设阈值; 2) 与其他算法相比, 除在优化 Rastrigrin 时, SM-IPSO 的结果劣于 AEPSO 外, 对于其他函数 SM-IPSO 均好于其他算法. 这一点, 从图 5 (见下页) 中算法结果与进化代数间的关系曲线也能看出.

进一步分析图 5 可知: 对于高维多模态函数 Griewank 和 Ackley, SM-IPSO, 所得结果明显优于包括 AEPSO 在内的其他算法; 而对于维数较低的 Rosenbrock 和 Rastrigrin 函数, 存在 AEPSO 优于或明显优于 SM-IPSO 的情况. 这是因为函数变量的维数越低, AEPSO 中单方向速度逃逸使微粒跳出局部极值点的概率越大. 特别地, 对于各维决策变量间相互独立的 Rastrigrin 函数而言, 在逃逸速度的作用下微粒位置在某一维上的进化 (即接近 0), 对提高微粒适应值的贡献是十分显著的. 因此, 在规定计算花费下对表 1 中全部测试函数而言, 我们有理由认为 SM-IPSO 所得结果好于其他 4 种比较算法. 然而, 当 SM-IPSO 所得结果非常接近问题全局

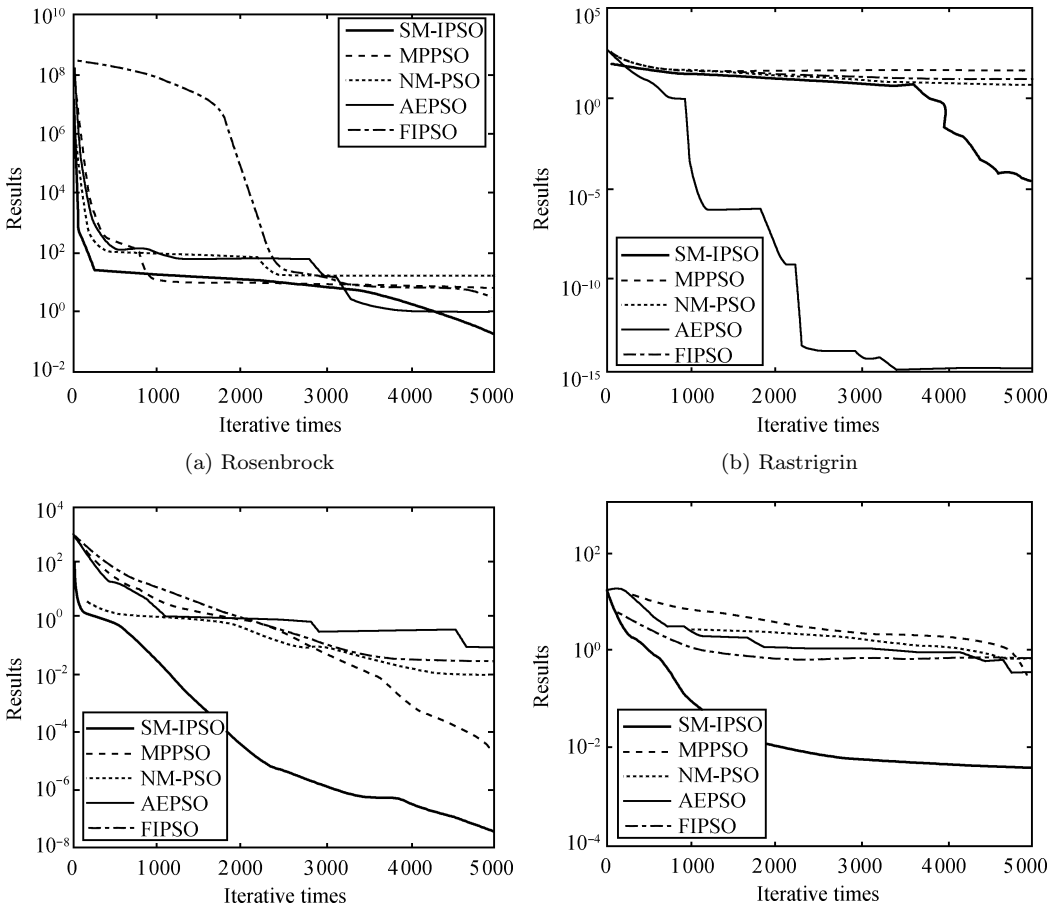
最优解时, 在强制扩张算子的作用下 SM-IPSO 存在收敛速度变慢这一不足, 图 5 显示 SM-IPSO 在第 5 000 次迭代时仍未完全收敛.

为定量分析算法的可靠性和进化速度, 表 4 (见下页) 展示了不同算法优化 5 个测试函数的达标率及所需函数平均评价次数, 其中最佳结果已用粗体标出. 比较表 4 中数据可知: 1) 对于 Quadric 和 Griewank, 虽然 AEPSO、NM-PSO 或 MPPSO 能以 100 % 的达标率得到满意结果, 但它们所用函数评价次数都明显多于同样以 100 % 达标的 SM-IPSO; 2) 对于 Rosenbrock 和 Ackley, 相比于其他 4 种算法, SM-IPSO 算法能以少的函数评价次数、高的达标率得到满意解; 3) 对于多模态 Rastrigrin 函数, 只有 SM-IPSO 和 AEPSO 分别以较高的达标率, 即 90 % 和 95 % 得到满意结果. 此时, 尽管 SM-IPSO 和 AEPSO 在优化 Rosenbrock 等函数时不能以 100 % 的达标率得到满意结果 (这是因为最大函数评价次数限制了算法的进一步搜索), 但是只要给予足够多的迭代次数, 两种算法在变异算子的作用下就有能力找到近似全局最优解.

表 3 最大函数评价次数内 5 种算法所得平均最优结果及其标准差

Table 3 Averages of optimal results with different algorithms after maximal iterations

	FIPSO	AEPSO	NM-PSO	MPPSO	SM-IPSO
Quadric	8.3E-5 (E-5)	2.5E-14 (E-4)	5.1E-6 (E-5)	0.011 (E-3)	**2.1E-6 (E-6)**
Rosenbrock	11.27 (7.13)	2.189 (7.45)	13.7 (10.2)	7.62 (10.54)	**0.79 (0.34)**
Rastrigrin	10.85 (6.55)	**0.658** (1.004)	7.62 (8.24)	45.8 (13.7)	0.84 (0.52)
Griewank	0.077 (0.122)	0.124 (0.481)	0.055 (0.098)	9.2E-5 (1.1E-4)	**1.5E-8 (E-8)**
Ackley	0.684 (0.746)	0.348 (0.554)	0.321 (0.258)	0.572 (1.245)	**2.6E-3 (E-3)**



(a) Rosenbrock (b) Rastrigrin

(c) Griewank (d) Ackley

图 5 五种算法所得结果与进化代数之间关系

Fig. 5 Relationship between results of five algorithms and iterative times

表 4　不同算法优化结果的达标率及其所用函数评价次数

Table 4　Averages of iterations and satisfactory ratios with different algorithms

	FIPSO	AEPSO	NM-PSO	MPPSO	SM-IPSO
Quadric	100 % (240350)	100 % (109520)	100 % (147200)	25 % (235440)	**100 % (90850)**
Rosenbrock	0 % (—)	70 % (264720)	0 % (—)	0 % (—)	**85 % (259580)**
Rastrigrin	10 % (273410)	**95 % (85200)**	0 % (—)	0 % (—)	90 % (285430)
Griewank	45 % (282420)	25 % (163340)	65 % (179290)	100 % (168200)	**100 % (42460)**
Ackley	15 % (285660)	65 % (256190)	35 % (2709540)	65 % (272150)	**100 % (202470)**

可见, 相对其他 4 种算法, SM-IPSO 对表 1 中除 Rastrigrin 外的其他测试函数, 均能以较快的进化速度 (或较少的计算花费) 和较高的可靠性求得满意解或最优解; 而对于 Rastrigrin, 尽管 SM-IPSO 所得结果劣于 AEPSO, 但明显优于其他三种算法.

5　结论

本文介绍并分析了已有 PPSO 改进算法和影响 PPSO 算法性能的主要原因, 结合 PSO 的全局快速寻优能力和 SM 的局部快速开发能力, 给出了一种基于单纯形法的改进微粒群优化算法, 即 SM-IPSO, 并从理论上证明了它具有全局收敛性. SM-IPSO 作为一种新的并行混合策略, 分别在奇数种群和偶数种群上并行运行基于概率逃逸运动的微粒群改进算法和基于强制扩张算子的单纯形改进法, 并通过周期性迁移种群间个体, 达到了算法探索与开发能力的有效均衡. 实验结果显示: 该算法除在优化 Rastrigrin 时劣于 AEPSO 外, 对其他典型测试函数, 均好于其他 4 种比较算法, 这是符合进化算法中 "没有免费午餐" 定理的. 然而, 通过实验发现, SM-IPSO 在算法后期存在种群收敛速度慢这一不足, 因此, 如何改善算法的收敛速度是需要进一步研究的课题.

References

1 Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. Perth, Australia: IEEE, 1995. 1942−1948

2 Hu Wang, Li Zhi-Shu. A simple and more effective particle swarm optimization algorithm. *Journal of Software*, 2007, **18**(4): 861−868
(胡旺, 李志蜀. 一种更简化而高效的粒子群优化算法. 软件学报, 2007, **18**(4): 861−868)

3 Belal M, EI-Ghazawi T. Parallel models for particle swarm optimizers. *International Journal of Intelligent Computing and Information Sciences*, 2004, **4**(1): 100−111

4 Zhang Lei, Yang Bo. Research and implemention of parallel particle swarm optimization algorithm. *Journal on Communications*, 2005, **26**(z1): 289−292
(张蕾, 杨波. 并行粒子群优化算法的设计与实现. 通信学报, 2005, **26**(z1): 289−292)

5 Byung-II K, George A D, Haftka R T, Fregly B J. Parallel asynchronous particle swarm optimization. *International Journal for Numerical Methods in Engineering*, 2006, **67**(4): 578−595

6 Guo Tong-Cheng, Mu Chun-Di. The drifts of parallel genetic algorithms. *Systems Engineering – Theory and Practice*, 2002, **22**(2): 15−23
(郭彤城, 慕春棣. 并行遗传算法的新进展. 系统工程理论与实践, 2002, **22**(2): 15−23)

7 Smith S. The simplex method and evolutionary algorithms. In: Proceedings of IEEE International Conference on Evolutionary Computation and IEEE World Congress on Computational Intelligence. Anchorage, USA: IEEE, 1998. 799−804

8 Fan Shu-Kai S, Zahara E. A hybrid simplex search and particle swarm optimization for unconstrained optimization. *European Journal of Operational Research*, 2007, **181**(2): 527−548

9 Nelder J A, Mead R. A simplex method for function minimization. *The Computer Journal*, 1965, **7**(4): 308−313

10 Wang F, Qiu Y H, Feng N Q. Multi-model function optimization by a new hybrid nonlinear simplex search and particle swarm algorithm. In: Proceedings of the 1st International Conference on Advances in Natural Computation. Changsha, China: Springer, 2005. 562−565

11 Li Bing-Yu, Xiao Yun-Shi, Wang Lei. A hybrid particle swarm optimization algorithm for solving complex functions with high dimensions. *Information and Control*, 2004, **33**(1): 27−30
(李炳宇, 萧蕴诗, 汪镭. 一种求解高维复杂函数优化问题的混合粒子群优化算法. 信息与控制, 2004, **33**(1): 27−30)

12 Wang Fang, Qiu Yu-Hui. A novel particle swarm algorithm using the simplex method operator. *Information and Control*, 2005, **34**(5): 517−522
(王芳, 邱玉辉. 一种引入单纯形法算子的新颖粒子群算法. 信息与控制, 2005, **34**(5): 517−522)

13 Chen J F, Ren Z W, Fan X N. A hybrid optimized algorithm based on an improved simplex method and particle swarm optimization. In: Proceedings of the 25th Chinese Control Conference. Harbin, China: IEEE, 2006. 1448−1453

14 Shi Y H, Eberhart R. A modified particle swarm optimizer. In: Proceedings of IEEE International Conference on Evolutionary Computation and IEEE World Congress on Computational Intelligence. Washington D. C., USA: IEEE, 1998. 69−73

15 Huang Fang, Fan Xiao-Ping. Parallel particle swarm optimization algorithm with island population model. *Control and Decision*, 2006, **21**(2): 175−179
(黄芳, 樊晓平. 基于岛屿群体模型的并行粒子群优化算法. 控制与决策, 2006, **21**(2): 175−179)

16 Matsumura T, Nakamura M, Okech J, Miyazato D, Onaga K. A parallel and distributed genetic algorithm on loosely-coupled multiprocessor system. *IEICE Transactions on Fundamental of Electronics, Communications and Computer Science*, 1998, **81**(4): 540−546

17 Li B, Wada K. Parallelizing particle swarm optimization. In: Proceedings of the IEEE Pacific-Rim Conference on Communications, Computers and Signal Processing. Victoria, British Columbia: IEEE, 2005. 288−291

18 El-Abd M, Kamel M S. On the convergence of information exchange methods in multiple cooperating swarms. In: Proceedings of IEEE Congress on Evolutionary Computation. Vancouver, Canada: IEEE, 2006. 1052−1056

19 Lienig J. A parallel genetic algorithm for performance-driven VLSI routing. *IEEE Transactions on Evolutionary Computation*, 1997, **1**(1): 29−39

20 Jiang M, Luo Y P, Yang S Y. Stagnation analysis in particle swarm optimization. In: Proceedings of IEEE Swarm Intelligence Symposium. Honolulu, USA: IEEE, 2007. 92−99

21 Li X D. Particle swarm optimization. In: Proceedings of the 2007 Genetic and Evolutionary Computation Conference. London, UK: ACM, 2007. 3391−3414

22 Van den B F. An Analysis of Particle Swarm Optimizers [Ph. D. dissertation], University of Pretoria, South Africa, 2002

23 He Ran, Wang Yong-Ji, Wang Qing, Zhou Jin-Hui, Hu Chen-Yong. An improved particle swarm optimization based on self-adaptive escape velocity. *Journal of Software*, 2005, **16**(12): 2036−2044
(赫然, 王永吉, 王青, 周津慧, 胡陈勇. 一种改进的自适应逃逸微粒群算法及实验分析. 软件学报, 2005, **16**(12): 2036−2044)

24 Solis F J, Wets R J B. Minimization by random search techniques. *Mathematics of Operations Research*, 1981, **6**(1): 19−30

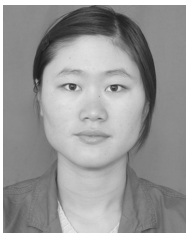
25 Du Zhi-Hui. *High Performance Calculation Parallel Programming Techniques – MPI Parallel Programming*. Beijing: Tsinghua University Press, 2001
(都志辉. 高性能计算之并行编程技术 — MPI 并行程序设计. 北京: 清华大学出版社, 2001)

26 Mendes R, Kennedy J, Neves J. The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 2004, **8**(3): 204−210

27 Matthew S, Terence S. Breeding swarms: a GA/PSO hybrid. In: Proceedings of Conference on Genetic and Evolutionary Computation. Washington D. C., USA: ACM, 2005. 161−168

张 勇 中国矿业大学自动化研究所博士研究生. 主要研究方向为微粒群优化. 本文通信作者.
E-mail: yongzh401@126.com
(**ZHANG Yong** Ph. D. candidate at the Institute of Automation, China University of Mining and Technology. His main research interest is particle swarm optimization. Corresponding author of this paper.)

巩敦卫 中国矿业大学教授. 主要研究方向为智能优化与控制.
E-mail: dwgong@vip.163.com
(**GONG Dun-Wei** Professor, Ph. D. at China University of Mining and Technology. His research interest covers intelligence optimization and control.)



张婉秋 中国矿业大学自动化研究所硕士研究生. 主要研究方向为软件测试和遗传算法.
E-mail: amyseven03@163.com
(**ZHANG Wan-Qiu** Master student at the Institute of Automation, China University of Mining and Technology. Her research interest covers software testing and genetic algorithm.)