文章编号: 1001-0920 (2015) 11-2041-07 DOI: 10.13195/j.kzyjc.2014.1432

多比特概率幅编码的量子衍生粒子群优化算法

李盼池, 李滨旭

(东北石油大学 计算机与信息技术学院，黑龙江 大庆 163318)

摘 要: 为了提高粒子群算法的优化能力, 提出一种新的量子衍生粒子群优化算法. 该方法采用多比特量子系统的基态概率幅对粒子编码, 基于自身最优粒子和全局最优粒子确定旋转角度, 采用基于张量积构造的多比特量子旋转门实施粒子的更新. 在每步迭代中, 只需更新粒子的一个量子比特相位, 即可更新该粒子上的所有概率幅. 标准函数极值优化的实验结果表明, 所提出算法的单步迭代时间较长, 但优化能力较同类算法有大幅度提高.

关键词: 量子计算；粒子群优化；多比特概率幅编码；算法设计

中图分类号: TP18 **文献标志码:** A

Quantum-inspired particle swarm optimization algorithm encoded by probability amplitudes of multi-qubits

LI Pan-chi, *LI Bin-xu*

(School of Computer and Information Technology，Northeast Petroleum University，Daqing 163318，China. Correspondent：LI Pan-chi，E-mail：lipanchi@vip.sina.com)

Abstract: To enhance the optimization ability of the particle swarm algorithm, a novel quantum-inspired particle swarm optimization algorithm is proposed. In this method, the particles are encoded by the probability amplitudes of the basic states of the multi-qubits system. The rotation angles of multi-qubits are determined based on the local optimum particle and the global optimal particle, and the multi-qubits rotation gates are employed to update the particles. At each of iteration, updating any a qubit can lead to update all probability amplitudes of the corresponding particle. The experimental results of some benchmark functions optimization shows that, although its single step iteration consumes a long time, the optimization ability of the proposed method is significantly higher than other similar algorithms.

Keywords: quantum computing；particle swarm optimization；multi-qubits probability amplitudes encoding；algorithm design

0 引 言

1995 年, Kennedy 等[1]提出了粒子群优化算法 (PSO), 作为一种新兴的优化工具, 现已广泛应用于组合优化[2]和数值优化[3]. 在 PSO 性能改进方面, 目前主要采取以下几种策略: 针对控制参数的选择[4]、针对粒子速度和位置的更新规则[5]、与其他算法相融合[6]、采用量子计算机制设计更新策略的量子 PSO (QPSO)[7]. 这些改进使 PSO 性能均有不同程度的提高. 量子计算是信息科学与量子力学相结合的新兴交叉学科, 其与智能优化算法的融合始于 20 世纪 90 年代, 目前比较成熟的算法有量子行为粒子群优化算法[8]、量子衍生进化算法[9]、量子衍生和声搜索算法[10]、量子衍生免疫算法[11]、量子衍生遗传算法[12]、量子衍生差分进化算法[13]等. 这些算法的编码方式除文献 [8] 采用实数编码外, 均采用单比特概率幅编码. 具体而言, 在目前的量子衍生粒子群优化算法中, 每个粒子用 D 个概率幅描述 (其中 D 为维数), 即每一维用单个概率幅编码. 这种单比特概率幅编码的缺点是, 调整一个量子比特仅能改变个体上的一个基因位, 因此优化效率不够理想. 对于本文提出的多比特概率幅编码, “多比特”的含义并不仅仅在于一个粒子采用多个量子比特概率幅编码, 否则将与现有方法没有区别, 重点在于对粒子的每一维采用多个量子比特概率幅编码. 这种编码方法的核心优势是, 利

收稿日期: 2014-09-16；**修回日期:** 2015-03-09.

基金项目: 国家自然科学基金项目(61170132)；黑龙江省教育厅科学技术研究项目(12541059)；黑龙江省自然科学基金项目(F2015021).

作者简介: 李盼池(1969−), 男, 教授, 博士生导师, 从事量子智能优化、量子神经网络等研究；李滨旭(1992−), 女, 硕士生, 从事群智能优化算法的研究.

用量子态的相干性, 每调整一个量子比特, 均可改变多比特量子叠加态中所有基态的概率幅, 从而改变该粒子上的所有基因位, 并实现对该粒子的更新; 若每步迭代调整 n 个量子比特, 则每步迭代可对每个粒子实施 n 次更新, 从而可以显著提高优化效率.

鉴于此, 本文提出一种全新的采用多比特概率幅编码的量子衍生粒子群优化算法, 标准函数极值优化的实验结果验证了所提出方法的优越性.

1 基本 PSO 模型

设在 n 维空间中有 M 个粒子组成一个种群, 其中第 i 个粒子位置 $\boldsymbol{X}_i$、速度 $\boldsymbol{V}_i$、自身最优位置 $\boldsymbol{P}_i^L$、整个种群最优位置 $\boldsymbol{P}_g$ 分别记为

$$\begin{aligned}\boldsymbol{X}_i &= (x_{i1}, x_{i2}, \cdots, x_{in}),\\ \boldsymbol{V}_i &= (v_{i1}, v_{i2}, \cdots, v_{in}),\\ \boldsymbol{P}_i^L &= (p_{i1}, p_{i2}, \cdots, p_{in}),\\ \boldsymbol{P}_g &= (p_{g1}, p_{g2}, \cdots, p_{gn}).\end{aligned}$$

将 $\boldsymbol{X}_i$ 代入目标函数可计算其目标值, 粒子状态更新策略为

$$\begin{aligned}\boldsymbol{V}_i(t+1) = {}& w\boldsymbol{V}_i(t) + c_1 r_1(\boldsymbol{P}_i^L - \boldsymbol{X}_i(t)) + \\ & c_2 r_2(\boldsymbol{P}_g - \boldsymbol{X}_i(t)),\end{aligned} \tag{1}$$

$$\boldsymbol{X}_i(t+1) = \boldsymbol{X}_i(t) + \boldsymbol{V}_i(t). \tag{2}$$

其中: $i = 1, 2, \cdots, M$; w 为惯性因子; c_1 为自身因子; c_2 为全局因子; r_1、r_2 为 $(0, 1)$ 之间的随机数.

对种群中每个粒子应用式 (1) 和 (2) 循环迭代, 使整个种群逐步逼近全局最优解. 为便于叙述, 将式 (1) 重写为如下形式:

$$\boldsymbol{V}_i(t+1) = w\boldsymbol{V}_i(t) + [\Phi](\boldsymbol{P}_i - \boldsymbol{X}_i(t)). \tag{3}$$

其中

$$\begin{aligned}\boldsymbol{P}_i = {}& \operatorname{diag}\Big(\frac{c_1 r_1^1}{c_1 r_1^1 + c_2 r_1^2}, \cdots, \frac{c_1 r_n^1}{c_1 r_n^1 + c_2 r_n^2}\Big)\boldsymbol{P}_i^L + \\ & \operatorname{diag}\Big(\frac{c_2 r_1^2}{c_1 r_1^1 + c_2 r_1^2}, \cdots, \frac{c_2 r_n^2}{c_1 r_n^1 + c_2 r_n^2}\Big)\boldsymbol{P}_g,\end{aligned} \tag{4}$$

$$[\Phi] = \operatorname{diag}(c_1 r_1^1 + c_2 r_1^2, \cdots, c_1 r_n^1 + c_2 r_n^2). \tag{5}$$

为使 PSO 收敛, 所有粒子必须逼近式 (4) 定义的 $\boldsymbol{P}_i$.

2 多比特量子系统和多比特量子旋转门

2.1 量子比特和单比特量子旋转门

在量子计算中, 量子比特有两个可能的基态 $|0\rangle$ 和 $|1\rangle$, 与经典比特的区别在于, 量子比特的状态可以落在基态 $|0\rangle$ 和 $|1\rangle$ 之外, 即可以是基态的线性叠加态, 其中叠加系数称为基态的概率幅. 量子比特的状态也可以借助三角函数表示为

$$|\phi\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix}^{\mathrm{T}}. \tag{6}$$

其中: θ 为 $|\phi\rangle$ 的相位, $\cos\theta$ 和 $\sin\theta$ 为 $|\phi\rangle$ 的概率幅.

量子门是物理实现量子计算的基础, 它包含了量子计算的特点. 单比特量子旋转门的定义为

$$\boldsymbol{R}(\Delta\theta) = \begin{bmatrix}\cos\Delta\theta & -\sin\Delta\theta\\ \sin\Delta\theta & \cos\Delta\theta\end{bmatrix}. \tag{7}$$

令 $|\phi\rangle = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix}$, 由 $\boldsymbol{R}(\Delta\theta)|\phi\rangle = \begin{bmatrix}\cos(\theta+\Delta\theta)\\ \sin(\theta+\Delta\theta)\end{bmatrix}$ 可知, $\boldsymbol{R}(\Delta\theta)$ 实现了对 $|\phi\rangle$ 的相位旋转.

2.2 矩阵张量积

设 $\boldsymbol{A}$ 是 $m\times n$ 矩阵, $\boldsymbol{B}$ 是 $p\times q$ 矩阵, 则 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的张量积定义为

$$\boldsymbol{A}\otimes\boldsymbol{B} = \begin{bmatrix}A_{11}\boldsymbol{B} & A_{12}\boldsymbol{B} & \cdots & A_{1n}\boldsymbol{B}\\ A_{21}\boldsymbol{B} & A_{22}\boldsymbol{B} & \cdots & A_{2n}\boldsymbol{B}\\ \vdots & \vdots & \ddots & \vdots\\ A_{m1}\boldsymbol{B} & A_{m2}\boldsymbol{B} & \cdots & A_{mn}\boldsymbol{B}\end{bmatrix}. \tag{8}$$

其中: A_{ij} 为 $\boldsymbol{A}$ 的元素, $A_{ij}\boldsymbol{B}$ 为 $p\times q$ 子矩阵.

2.3 多比特量子系统和多比特量子旋转门

一般地, 对于 n 比特量子系统, 有 2^n 个形如 $|x_1x_2\cdots x_n\rangle$ 的基态, 类似于单比特量子系统, n 比特量子系统也可以处于 2^n 个基态的线性叠加态, 即

$$\begin{aligned}&|\phi_1\phi_2\cdots\phi_n\rangle = \\ &\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_n=0}^{1} a_{x_1x_2\cdots x_n}|x_1x_2\cdots x_n\rangle = \\ &[a_{00\cdots0}\ \ a_{00\cdots1}\ \ \cdots\ \ a_{11\cdots1}]^{\mathrm{T}},\end{aligned} \tag{9}$$

其中 $a_{x_1x_2\cdots x_n}$ 称为基态 $|x_1x_2\cdots x_n\rangle$ 的概率幅, 且满足

$$\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_n=0}^{1}|a_{x_1x_2\cdots x_n}|^2 = 1. \tag{10}$$

记 $|\phi_i\rangle = \cos\theta_i|0\rangle + \sin\theta_i|1\rangle$, 根据量子计算原理, $|\phi_1\phi_2\cdots\phi_n\rangle$ 可用张量积表示为

$$\begin{aligned}&|\phi_1\phi_2\cdots\phi_n\rangle = |\phi_1\rangle\otimes|\phi_2\rangle\otimes\cdots|\phi_n\rangle = \\ &[\cos\theta_1\ \ \sin\theta_1]^{\mathrm{T}}\otimes\cdots\otimes[\cos\theta_n\ \ \sin\theta_n]^{\mathrm{T}} = \\ &\begin{bmatrix}\cos\theta_1 & \cos\theta_2 & \cdots & \cos\theta_n\\ \cos\theta_1 & \cos\theta_2 & \cdots & \sin\theta_n\\ \vdots & \vdots & \ddots & \vdots\\ \sin\theta_1 & \sin\theta_2 & \cdots & \sin\theta_n\end{bmatrix}.\end{aligned} \tag{11}$$

式 (11) 表明, 在 n 比特量子系统中, 任何一个基态的概率幅均为 n 个量子比特相位 $(\theta_1\theta_2\cdots\theta_n)$ 的函数, 换言之, 在 n 个量子比特相位中, 更新任何一个 θ_i, 均可更新所有 2^n 个基态的概率幅. 基态概率幅的更新可采用 n 比特量子旋转门实现, 其中 n 比特量子旋转门为 n 个单比特量子旋转门 $\boldsymbol{R}(\Delta\theta_i)$ 的张量积, 即

$$\begin{aligned}&\boldsymbol{R}_n(\Delta\theta_1\Delta\theta_2\cdots\Delta\theta_n) = \\ &\boldsymbol{R}(\Delta\theta_1)\otimes\boldsymbol{R}(\Delta\theta_2)\otimes\cdots\otimes\boldsymbol{R}(\Delta\theta_n).\end{aligned} \tag{12}$$

其中

$$\boldsymbol{R}(\Delta\theta_i)=\begin{bmatrix}\cos\Delta\theta_i & -\sin\Delta\theta_i\\ \sin\Delta\theta_i & \cos\Delta\theta_i\end{bmatrix},$$

$$i=1,2,\cdots,n.$$

以 $n=2$ 为例, 有

$$\boldsymbol{R}(\Delta\theta_1\Delta\theta_2)=$$

$$\begin{bmatrix}\cos\Delta\theta_1\cos\Delta\theta_2 & -\cos\Delta\theta_1\sin\Delta\theta_2\\ \cos\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2\\ \sin\Delta\theta_1\cos\Delta\theta_2 & -\sin\Delta\theta_1\sin\Delta\theta_2\\ \sin\Delta\theta_1\sin\Delta\theta_2 & \sin\Delta\theta_1\cos\Delta\theta_2\end{bmatrix}\rightarrow$$

$$\leftarrow\begin{bmatrix}-\sin\Delta\theta_1\cos\Delta\theta_2 & \sin\Delta\theta_1\sin\Delta\theta_2\\ -\sin\Delta\theta_1\sin\Delta\theta_2 & -\sin\Delta\theta_1\cos\Delta\theta_2\\ \cos\Delta\theta_1\cos\Delta\theta_2 & -\cos\Delta\theta_1\sin\Delta\theta_2\\ \cos\Delta\theta_1\sin\Delta\theta_2 & \cos\Delta\theta_1\cos\Delta\theta_2\end{bmatrix}. \quad (13)$$

容易证明

$$\boldsymbol{R}_n(\Delta\theta_1\Delta\theta_2\cdots\Delta\theta_n)|\phi_1\phi_2\cdots\phi_n\rangle=$$
$$|\widehat{\phi_1}\rangle\otimes|\widehat{\phi_2}\rangle\otimes\cdots\otimes|\widehat{\phi_n}\rangle, \quad (14)$$

其中 $|\widehat{\phi_i}\rangle=\cos(\theta_i+\Delta\theta_i)|0\rangle+\sin(\theta_i+\Delta\theta_i)|1\rangle$.

3　基于多比特概率幅的粒子编码方法

3.1　量子比特数的确定

n 比特量子系统有 2^n 个概率幅, 若优化空间为 D 维, 则需使 $D\leqslant 2^n$. 又因 2^n 个概率幅之间有平方和为 1 的约束关系, 所以必须使 $D<2^n$. 对于 D 维优化问题, 编码所需的量子比特数可按下式计算:

$$n=\lceil\log(D)\rceil+1. \quad (15)$$

3.2　多比特概率幅粒子群编码方法

首先随机生成 N 个 n 维相位向量 $\boldsymbol{\theta}_i\,(i=1,2,\cdots,N)$, 有

$$\boldsymbol{\theta}_i=[\theta_{i1},\theta_{i2},\cdots,\theta_{\mathrm{in}}], \quad (16)$$

其中 $\theta_{ij}=2\pi\times\mathrm{rand}$, rand 为 (0,1) 内均匀分布的随机数, $j=1,2,\cdots,n$.

记 $|\phi_{ij}\rangle=\cos\theta_{ij}|0\rangle+\sin\theta_{ij}|1\rangle$, 利用式 (11) 得到 N 个量子比特系统 $|\phi_{11}\phi_{12}\cdots\phi_{1n}\rangle$, $|\phi_{21}\phi_{22}\cdots\phi_{2n}\rangle$, $\cdots$, $|\phi_{N1}\phi_{N2}\cdots\phi_{Nn}\rangle$, 每个量子系统均有 2^n 个基态, 取每个量子系统的前 D 个基态的概率幅, 即可得到 N 个 D 维的粒子编码.

4　基于多比特概率幅的粒子更新方法

本文采用多比特量子旋转门实现粒子的更新. 记全局最优粒子的相位向量为 $\boldsymbol{\theta}_g=[\theta_{g1},\theta_{g2},\cdots,\theta_{gn}]$, 第 i 个粒子的当前位置和自身最优位置的相位向量分别为 $\boldsymbol{\theta}_i=[\theta_{i1},\theta_{i2},\cdots,\theta_{in}]$ 和 $\boldsymbol{\theta}_{bi}=[\theta^i_{b1},\theta^i_{b2},\cdots,\theta^i_{bn}]$.

由式 (11) 可知, 每更新 $\boldsymbol{\theta}_i$ 的一个相位, 便能更新 $\boldsymbol{\theta}_i$ 对应的所有概率幅, 从而实现该粒子所有 D 维解变量的更新. 为了提高搜索能力, 本文采用循环更新 $\boldsymbol{\theta}_i$ 中所有相位的方法, 从而实现对该粒子的 n 次更新. 设 $\Delta\theta_0$ 为相位更新步长, 具体更新方法如下.

Step 1: 置 $j=1$, $\boldsymbol{P}_i(\boldsymbol{\theta})=[\cos\theta_{i1}\cos\theta_{i2}\cdots\cos\theta_{in},\cdots,\sin\theta_{i1}\sin\theta_{i2}\cdots\sin\theta_{in}]^{\mathrm{T}}$.

Step 2: 置 $\Delta\theta_{i1}=\Delta\theta_{i2}=\cdots=\Delta\theta_{in}=0$.

Step 3: 确定旋转角度的值, 其中 sgn 为符号函数:

1) 若 $|\theta^i_{bj}-\theta_{ij}|\leqslant\pi$, 则

$$\Delta\theta^b_{ij}=\mathrm{sgn}(\theta^i_{bj}-\theta_{ij})\Delta\theta_0;$$

2) 若 $|\theta^i_{bj}-\theta_{ij}|>\pi$, 则

$$\Delta\theta^b_{ij}=-\mathrm{sgn}(\theta^i_{bj}-\theta_{ij})\Delta\theta_0;$$

3) 若 $|\theta_{gj}-\theta_{ij}|\leqslant\pi$, 则

$$\Delta\theta^g_{ij}=\mathrm{sgn}(\theta_{gj}-\theta_{ij})\Delta\theta_0;$$

4) 若 $|\theta_{gj}-\theta_{ij}|>\pi$, 则

$$\Delta\theta^g_{ij}=-\mathrm{sgn}(\theta_{gj}-\theta_{ij})\Delta\theta_0.$$

Step 4: 按下式实现粒子的更新, 其中 r_1 和 r_2 为区间 (0,1) 内均匀分布的随机数:

$$\Delta\theta_{ij}=r_1\Delta\theta^b_{ij}+r_2\Delta\theta^g_{ij},$$
$$\overline{\boldsymbol{P}}_i(\boldsymbol{\theta})=\boldsymbol{R}_n(\Delta\theta_{i1},\Delta\theta_{i2},\cdots,\Delta\theta_{in})\boldsymbol{P}_i(\boldsymbol{\theta}).$$

Step 5: 令 $\Delta\theta_{ij}=\mathrm{rnds}\Delta\theta_0$, rnds 为 $(-1,1)$ 内随机数, $\widehat{\boldsymbol{P}}_i(\boldsymbol{\theta})=\boldsymbol{R}_n(\Delta\theta_{i1},\Delta\theta_{i2},\cdots,\Delta\theta_{in})\overline{\boldsymbol{P}}_i(\boldsymbol{\theta})$. 若 $\widehat{\boldsymbol{P}}_i(\boldsymbol{\theta})$ 优于 $\overline{\boldsymbol{P}}_i(\boldsymbol{\theta})$, 则 $\boldsymbol{P}_i(\boldsymbol{\theta})=\widehat{\boldsymbol{P}}_i(\boldsymbol{\theta})$, 否则, $\boldsymbol{P}_i(\boldsymbol{\theta})=\overline{\boldsymbol{P}}_i(\boldsymbol{\theta})$.

Step 6: 若 $j<n$, 则令 $j=j+1$, 返回 Step2, 否则粒子 $\boldsymbol{P}_i(\boldsymbol{\theta})$ 更新结束.

5　多比特概率幅编码量子衍生粒子群算法

设粒子总数为 N, 优化空间为 D 维. 以极小值优化为例, 本文提出的多比特概率幅编码量子衍生粒子群算法 (MQPAPSO) 流程可描述如下.

Step 1: 粒子群初始化.

对于每个粒子, 按式 (15) 确定量子比特数 n, 按式 (16) 初始化 n 个量子比特相位, 按式 (11) 计算 2^n 个概率幅, 其中前 D 个概率幅即为该粒子的编码. 记第 i 个粒子的第 j 个概率幅为 x_{ij}, 编码结果为

$$\begin{cases}\boldsymbol{P}_1=[x_{11},x_{12},\cdots,x_{1D}]^{\mathrm{T}},\\ \boldsymbol{P}_2=[x_{21},x_{22},\cdots,x_{2D}]^{\mathrm{T}},\\ \quad\vdots\\ \boldsymbol{P}_N=[x_{N1},x_{N2},\cdots,x_{ND}]^{\mathrm{T}}.\end{cases} \quad (17)$$

初始化相位步长 $\Delta\theta_0$、限定步数 G, 置当前步数 $t=1$.

Step 2: 计算目标函数值.

记第 j 维变量的取值区间为 $[\min X_j,\max X_j]$, 由于概率幅 $x_{ij}\in[-1,1]$, 按下式进行解空间变换:

$$X_{ij}=\frac{1}{2}[\max X_j(1+x_{ij})+\min X_j(1-x_{ij})]. \quad (18)$$

其中: $i=1,2,\cdots,N$, $j=1,2,\cdots,D$.

利用变换后的 X_{ij} 计算所有粒子的目标函数值. 记第 i 个粒子的相位 $\boldsymbol{\theta}_i=[\theta_{i1},\theta_{i2},\cdots,\theta_{in}]$, 目标函数值为 f_i, 全局最优粒子的相位 $\widehat{\boldsymbol{\theta}}_g=[\widehat{\theta}_{g1},\widehat{\theta}_{g2},\cdots,\widehat{\theta}_{gn}]$, 全局最优目标函数值为 $\widehat{f}_g$. 记第 i 个粒子的自身最优相位 $\widehat{\boldsymbol{\theta}}_i=\boldsymbol{\theta}_i$, 自身最优目标函数值 $\widehat{f}_i=f_i$.

Step 3: 粒子位置更新.

对于粒子群中每个粒子 $\boldsymbol{P}_i$, 按第 4 节 Step 1 ~ Step 6, 循环更新 n 次. 利用式 (11) 计算概率幅, 利用式 (18) 实施解空间变换, 计算目标函数值. 记更新后的相位 $\boldsymbol{\theta}_i=[\theta_{i1},\theta_{i2},\cdots,\theta_{in}]$, 目标函数值为 f_i. 若 $f_i<\widehat{f}_i$, 则 $\widehat{f}_i=f_i$, $\widehat{\boldsymbol{\theta}}_i=\boldsymbol{\theta}_i$.

Step 4: 全局最优解更新.

记当代最优粒子相位 $\boldsymbol{\theta}_g=[\theta_{g1},\theta_{g2},\cdots,\theta_{gn}]$, 最优目标函数值为 f_g. 若 $f_g<\widehat{f}_g$, 则 $\widehat{f}_g=f_g$, $\widehat{\boldsymbol{\theta}}_g=\boldsymbol{\theta}_g$, 否则 $f_g=\widehat{f}_g$, $\boldsymbol{\theta}_g=\widehat{\boldsymbol{\theta}}_g$.

Step 5: 判断终止条件.

若 $t<G$, 则令 $t=t+1$, 转至 Step 3, 否则保存 $\widehat{\boldsymbol{\theta}}_g$ 和 $\widehat{f}_g$, 流程结束.

6 对比实验

实验采用标准测试函数验证 MQPAPSO 的优化能力, 并与普通粒子群优化 (PSO)[14]、量子 Delta 势阱粒子群优化 (QDPSO)[15]、混合蛙跳算法 (SFLA)[16]进行对比. 所有函数均为极小值优化, D 为自变量个数, Ω 为解空间, $\boldsymbol{X}^*$ 为精确极小值点, $f(\boldsymbol{X}^*)$ 为极小值.

6.1 测试函数

测试函数如下:

$$f_1(\boldsymbol{X})=\sum_{i=1}^{D}x_i^2,\ \Omega=[-100,100]^D,$$

$$\boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_2(\boldsymbol{X})=\sum_{i=1}^{D}|x_i|+\prod_{i=1}^{D}|x_i|,\ \Omega=[-100,100]^D,$$

$$\boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_3(\boldsymbol{X})=\sum_{i=1}^{D}\Big(\sum_{j=1}^{i}x_j\Big)^2,\ \Omega=[-100,100]^D,$$

$$\boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_4(\boldsymbol{X})=\max_{1\leqslant i\leqslant D}(|x_i|),\ \Omega=[-100,100]^D,$$

$$\boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_5(\boldsymbol{X})=\sum_{i=1}^{D-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2),$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[1,1,\cdots,1],\ f(\boldsymbol{X}^*)=0.$$

$$f_6(\boldsymbol{X})=\sum_{i=1}^{D}ix_i^4(1+\mathrm{random}(0,1)),$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_7(\boldsymbol{X})=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10],$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_8(\boldsymbol{X})=-20\exp\Big(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\Big)-\exp\Big(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\Big)+20+e,$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_9(\boldsymbol{X})=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\Big(\frac{x_i}{\sqrt{i}}\Big)+1,$$

$$\Omega=[-500,500]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{10}(\boldsymbol{X})=\frac{1}{D}\sum_{i=1}^{D}(x_i^4-16x_i^2+5x_i)+78.332\,331\,4,$$

$$\Omega=[-100,100]^D,\ x_i^*=-2.903\,534,\ f(\boldsymbol{X}^*)=0.$$

$$f_{11}(\boldsymbol{X})=\frac{D(D+4)(D-1)}{6}+\sum_{i=1}^{D}(x_i-1)^2-\sum_{i=2}^{D}x_ix_{i-1},$$

$$\Omega=[-D^2,D^2]^D,\ x_i^*=i(D+1-i),\ f(\boldsymbol{X}^*)=0.$$

$$f_{12}(\boldsymbol{X})=\frac{\pi}{D}\Big[10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2(1+10\sin^2(\pi y_{i+1}))+(y_D-1)^2\Big]+\sum_{i=1}^{D}u(x_i,10,100,4);$$

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m,\ x_i>a;\\ 0,\ -a\leqslant x_i\leqslant a;\\ k(-x_i-a)^m,\ x_i<-a;\end{cases}$$

$$y_i=1+\frac{1}{4}(x_i+1),\ \Omega=[-100,100]^D;$$

$$\boldsymbol{X}^*=[-1,-1,\cdots,-1],\ f(\boldsymbol{X}^*)=0.$$

$$f_{13}(\boldsymbol{X})=\sum_{i=1}^{D-1}(x_i^2+2x_{i+1}^2-0.3\cos(3\pi x_i)\cos(4\pi x_{i+1})+0.3),$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{14}(\boldsymbol{X})=\sum_{i=1}^{D/4}[(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-2x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4],$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{15}(\boldsymbol{X})=\sum_{i=1}^{D-1}g(x_i,x_{i+1})+g(x_D,x_1);$$

$$g(x,y)=(x^2+y^2)^{0.25}[\sin^2(50(x^2+y^2)^{0.1})+1],$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{16}(\boldsymbol{X})=10D+\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)),$$

$$y_i=\begin{cases}x_i,\ |x_i|<1/2;\\ \mathrm{round}(2x_i)/2,\ |x_i|\geqslant 1/2;\end{cases}$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{17}(\boldsymbol{X})=\sum_{i=1}^{D}\Big\{\sum_{k=0}^{k\max}[a^k\cos(2\pi b^k(x_i+0.5))]\Big\}-D\sum_{k=0}^{k\max}(a^k\cos(\pi b^k)),\ a=0.5,\ b=0.3,$$

$$k\max=30,\ \Omega=[-100,100]^D,$$

$$\boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{18}(\boldsymbol{X})=\sum_{i=1}^{D}x_i^2+\Big(\sum_{i=1}^{D}0.5ix_i\Big)^2+\Big(\sum_{i=1}^{D}0.5ix_i\Big)^4;$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{19}(\boldsymbol{X})=\sum_{i=1}^{D-1}\Big(0.5+\frac{\sin^2(\sqrt{100x_i^2+x_{i+1}^2})-0.5}{1+0.001(x_i^2-2x_ix_{i+1}+x_{i+1}^2)^2}\Big),$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

$$f_{20}(\boldsymbol{X})=-\sum_{i=1}^{D-1}\Big[\exp\Big(\frac{-(x_i^2+x_{i+1}^2+0.5x_ix_{i+1})}{8}\Big)\times\cos(4\sqrt{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}})\Big]+D-1,$$

$$\Omega=[-100,100]^D,\ \boldsymbol{X}^*=[0,0,\cdots,0],\ f(\boldsymbol{X}^*)=0.$$

6.2　实验方案和参数设计

所有测试函数的维数均取 $D=50$ (f_{14} 为 $D=52$) 和 $D=100$ 两种情况. 4 种算法种群规模均取 $N=50$. PSO、QPSO、SFLA 的限定迭代步数均取 $G=100$ 和 $G=1\,000$ 两种情况. MQPAPSO 的限定迭代步数取 $G=100$.

对于 SFLA, 最大跳跃步长[16]取 $D_{\max}=5$. 由于蛙群分组情况与具体问题有关, 考虑多种分组情况, 并将最好结果作为对比指标. 具体而言, 蛙群分组取 $N=1\times 50=2\times 25=5\times 10=10\times 5=25\times 2=50\times 1$ 六种情况, 其中第 1 个数为组数, 第 2 个数为组内蛙数. 对于每种组合分别独立优化 30 次, 记录 30 次优化结果的平均值和单步迭代的平均时间. 在 6 种分组情况中, 将优化结果平均值最小者和相应单步迭代时间作为对比指标.

对于 PSO, $w=0.729\,8$, $c_1=c_2=1.496\,18$[14]. 对于 QDPSO, 控制参数取 $\lambda=1.2$[15]. 对于 MQPAPSO, 相位更新步长取 $\Delta\theta_0=0.05\pi$. 这 4 个算法对于每个函数均独立优化 30 次, 并取平均优化结果和单步迭代的平均时间作为对比指标.

6.3　实验结果对比

所有实验在 Windows XP 系统、主频 3.39 GHz、内存 3.47 GB 的 PC 机上, 采用 Matlab R2009a 编程实现. 以 $G=100$ 为例, 4 种算法单步迭代的平均时间对比如表 1 所示, $D=50$ 和 $D=100$ 时的平均优化结果对比如表 2 和表 3 所示.

表 1　4 种算法的单步迭代平均时间对比　　s

f_i	MQPAPSO		QDPSO		PSO		SFLA	
	$D=50$	$D=100$	$D=50$	$D=100$	$D=50$	$D=100$	$D=50$	$D=100$
f_1	0.018 6	0.029 0	0.001 1	0.001 9	0.000 9	0.001 6	0.001 4	0.002 0
f_2	0.018 7	0.029 2	0.001 2	0.002 0	0.001 2	0.001 6	0.001 7	0.002 5
f_3	0.024 8	0.042 8	0.006 4	0.012 7	0.009 9	0.022 7	0.006 8	0.016 8
f_4	0.018 8	0.029 6	0.001 1	0.001 9	0.000 9	0.001 6	0.001 4	0.002 4
f_5	0.023 0	0.039 7	0.004 9	0.009 5	0.001 6	0.002 5	0.002 2	0.004 3
f_6	0.023 5	0.038 7	0.001 8	0.003 1	0.002 8	0.004 8	0.001 9	0.003 2
f_7	0.018 7	0.029 1	0.001 3	0.002 1	0.001 6	0.002 2	0.001 4	0.002 4
f_8	0.019 1	0.029 5	0.001 5	0.002 3	0.001 9	0.002 8	0.002 4	0.003 6
f_9	0.023 4	0.038 2	0.001 6	0.002 4	0.001 9	0.002 8	0.001 7	0.002 7
f_{10}	0.019 3	0.030 1	0.001 9	0.003 1	0.002 8	0.004 8	0.001 7	0.002 8
f_{11}	0.023 4	0.038 3	0.001 6	0.002 4	0.001 6	0.002 5	0.001 7	0.002 7
f_{12}	0.026 2	0.044 1	0.009 6	0.017 3	0.005 4	0.008 9	0.003 3	0.007 0
f_{13}	0.019 3	0.029 8	0.002 0	0.003 0	0.002 5	0.004 1	0.001 7	0.003 0
f_{14}	0.023 3	0.037 2	0.004 8	0.008 9	0.002 8	0.004 4	0.002 4	0.003 3
f_{15}	0.025 6	0.044 9	0.003 1	0.005 7	0.005 1	0.009 6	0.003 8	0.006 0
f_{16}	0.024 8	0.041 8	0.006 5	0.012 4	0.002 8	0.004 8	0.002 7	0.004 3
f_{17}	0.037 8	0.070 6	0.024 6	0.048 6	0.111 6	0.219 2	0.029 2	0.065 1
f_{18}	0.023 4	0.038 2	0.001 7	0.002 4	0.001 9	0.002 5	0.002 2	0.002 8
f_{19}	0.020 6	0.031 4	0.002 8	0.003 7	0.003 8	0.005 4	0.003 3	0.004 1
f_{20}	0.021 2	0.032 0	0.002 8	0.003 9	0.004 1	0.005 7	0.004 3	0.005 2

表 2　4 种算法的平均优化结果对比 ($D=50$)

f_i	MQPAPSO	QDPSO		PSO		SFLA	
	$G=100$	$G=100$	$G=1\,000$	$G=100$	$G=1\,000$	$G=100$	$G=1\,000$
f_1	**1.9e-08**	1.5e+03	3.4e-05	3.4e+03	6.0e-05	8.5e+02	0.001 08
f_2	**1.3e-04**	9.4e+10	33.195 3	3.8e+15	1.3e+02	2.8e+02	2.6e+02
f_3	**3.7e-09**	3.9e+04	1.1e+04	6.7e+04	1.6e+04	6.3e+03	2.5e+03
f_4	**0.001 01**	36.936 4	10.202 9	61.967 5	54.662 5	12.125 8	9.714 06
f_5	**73.215 4**	1.2e+08	1.3e+02	2.7e+08	2.0e+02	1.1e+07	4.8e+02
f_6	**4.1e-11**	7.2e+07	2.1e+02	3.7e+08	1.5e+05	1.2e+04	2.3e-09
f_7	**7.9e-06**	1.9e+03	2.9e+02	3.3e+03	3.7e+02	1.4e+03	1.0e+03
f_8	**3.3e-05**	21.162 9	20.596 4	21.277 8	21.174 4	17.052 4	15.716 9
f_9	**4.9e-10**	11.185 7	0.003 52	23.675 7	0.032 75	2.005 64	0.012 09
f_{10}	**18.582 4**	2.7e+04	10.574 3	6.5e+04	23.426 0	1.8e+03	12.779 8
f_{11}	**2.8e+04**	1.6e+06	8.4e+04	5.1e+06	4.2e+05	9.7e+05	2.4e+04
f_{12}	**0.181 50**	2.9e+07	0.282 76	6.5e+07	1.658 72	1.1e+04	17.377 0
f_{13}	**7.6e-07**	4.2e+03	0.002 31	1.0e+04	4.008 30	3.2e+03	13.900 8
f_{14}	**3.9e-11**	2.9e+06	7.3e+04	3.9e+07	4.5e+07	8.1e+05	3.4e+04
f_{15}	**0.254 13**	2.3e+02	26.517 5	3.0e+02	1.7e+02	1.7e+02	1.5e+02
f_{16}	**2.2e-06**	1.9e+03	3.5e+02	3.5e+03	3.9e+02	1.3e+03	1.0e+03
f_{17}	**0.512 01**	67.331 0	47.580 5	78.813 1	75.889 2	48.027 4	33.539 3
f_{18}	**1.1e-05**	8.0e+04	4.1e+04	1.2e+05	9.8e+04	8.0e+03	5.4e+03
f_{19}	**1.5e-04**	0.499 97	0.499 59	0.499 99	0.499 98	0.494 69	0.491 68
f_{20}	**1.1e-06**	46.275 9	34.4948	47.201 4	45.757 8	44.043 3	43.417 5

表 3 4 种算法的平均优化结果对比 ($D = 100$)

f_i	MQPAPSO	QDPSO		PSO		SFLA	
	$G=100$	$G=100$	$G=1\,000$	$G=100$	$G=1\,000$	$G=100$	$G=1\,000$
f_1	**5.7e-08**	2.3e+04	1.2e+02	3.9e+04	2.7e+02	3.3e+03	5.480 43
f_2	**6.3e-04**	1.0e+20	6.0e+02	5.4e+25	1.2e+15	5.9e+02	5.7e+02
f_3	**2.2e-08**	1.9e+05	1.1e+05	3.0e+05	2.4e+05	2.4e+04	1.4e+04
f_4	**0.001 33**	67.712 3	44.933 4	85.504 9	85.418 6	15.218 5	13.047 1
f_5	**1.3e+02**	4.1e+09	5.3e+05	9.4e+09	6.5e+07	3.0e+07	1.0e+05
f_6	**1.6e-09**	4.0e+09	2.2e+08	1.5e+10	5.9e+08	2.4e+06	28.163 5
f_7	**2.5e-05**	2.1e+04	1.4e+03	4.3e+04	2.5e+03	4.4e+03	3.7e+03
f_8	**5.4e-05**	21.262 7	21.088 7	21.423 4	21.374 5	18.407 8	17.120 0
f_9	**1.5e-08**	1.4e+02	1.423 71	2.4e+02	2.741 91	18.760 4	0.228 63
f_{10}	**21.666 0**	4.7e+05	1.0e+02	9.4e+05	6.7e+03	2.6e+03	16.415 6
f_{11}	**2.4e+05**	2.0e+08	2.1e+07	4.9e+08	1.1e+08	3.0e+08	1.8e+06
f_{12}	**0.256 14**	1.8e+09	2.7e+03	4.2e+09	1.3e+07	7.5e+04	26.576 0
f_{13}	**1.6e-06**	6.9e+04	7.8e+02	1.1e+05	1.0e+03	1.0e+04	58.725 2
f_{14}	**9.6e-11**	9.3e+07	4.4e+06	1.0e+09	1.5e+09	3.8e+06	3.3e+05
f_{15}	**0.67362**	6.7e+02	3.5e+02	8.1e+02	5.5e+02	3.8e+02	3.4e+02
f_{16}	**1.0e-05**	2.2e+04	1.4e+03	4.3e+04	3.2e+03	3.9e+03	3.7e+03
f_{17}	**1.069 24**	1.4e+02	1.1e+02	1.7e+02	1.6e+02	1.2e+02	1.0e+02
f_{18}	**1.3e-05**	1.9e+05	1.4e+05	3.0e+05	2.4e+05	2.1e+04	1.9e+04
f_{19}	**0.001 27**	0.499 99	0.499 98	0.499 99	0.499 99	0.497 74	0.498 30
f_{20}	**0.002 22**	96.320 8	83.629 3	97.019 8	95.616 2	92.853 0	90.566 4

对于函数 f_i, 令 MQPAPSO, QDPSO, PSO, SFLA 单步迭代的平均时间分别为 T_i^M、T_i^Q、T_i^P、T_i^S, 平均优化结果分别为 O_i^M、O_i^Q、O_i^P、O_i^S. 为便于进一步对比, 以 MQPAPSO 和 QDPSO 为例, 按如下两式定义单步迭代平均时间比值和平均优化结果比值:

$$\frac{T^M}{T^Q} = \frac{\sum_{i=1}^{20} T_i^M/T_i^Q}{20}, \tag{19}$$

$$\frac{O^M}{O^Q} = \frac{\sum_{i=1}^{20} O_i^M/O_i^Q}{20}. \tag{20}$$

4 种算法关于函数 $f_1 \sim f_{20}$ 的单步迭代平均时间比值如表 4 所示, 平均优化结果比值如表 5 所示.

表 4 4 种算法的单步迭代平均时间比值

D	T^M/T^Q	T^M/T^P	T^M/T^S
50	9.863 512	9.800 094	9.620 418
100	9.785 713	10.039 69	9.752 004
平均	9.824 613	9.919 894	9.686 211

由表 1 和表 4 可见, 就 $f_1 \sim f_{20}$ 的单步迭代平均时间而言, 基于多比特概率幅的粒子编码和多比特量子旋转门的粒子更新机制使 MQPAPSO 的单步运行时间为 QDPSO、PSO、SFLA 的将近 10 倍. 为使对比公平, 不仅需要考察相同迭代步数下的对比, 而且必须进一步考察相同优化时间下的对比. 这便是将 QDPSO、PSO、SFLA 的迭代步数分别设置为 $G=100$ 和 $G = 1\,000$ 的根本原因. 就 $f_1 \sim f_{20}$ 的优化结果而言, 由表 2 和表 3 可见, 当 $G = 100$ 时, MQPAPSO 比其他 3 种算法小 3～4 个数量级; 当 $G = 1\,000$ 时, MQPAPSO 也比其他 3 种算法小 1～2 个数量级. 这表明采用多比特概率幅编码和进化机制能够提高寻优能力. 由表 5 可见, 在相同迭代步数下, MQPAPSO 的优化结果仅为 QDPSO 的千分之一; 在相同优化时间下, MQPAPSO 的优化结果仅为 QDPSO 的百分之九. 实验表明, 多比特概率幅编码方法能够显著地提高传统粒子群算法和其他同类算法的优化能力.

表 5 4 种算法的平均优化结果比值

D	$O^M_{G=100}/O^Q$		$O^M_{G=100}/O^P$		$O^M_{G=100}/O^S$	
	$G=10^2$	$G=10^3$	$G=10^2$	$G=10^3$	$G=10^2$	$G=10^3$
50	0.001 361	0.165 868	0.000 671	0.067 214	0.002 587	0.140 945
100	0.000 623	0.012 133	0.000 510	0.000 795	0.001 124	0.073 973
平均	9.92e-04	0.089 001	5.91e-04	0.034 004	0.001 856	0.107 459

6.4 实验结果分析

对于实验结果表现出的 MQPAPSO 运行时间长、优化能力高的特性, 给出如下分析:

1) 在基于多比特概率幅的个体编码方法中, 若编码的量子比特数为 n, 则每步迭代, 每个粒子均被更新 n 次, 因此加大计算量导致运行时间延长. 但在延长运行时间的同时, 也大幅度提高了粒子更新次数, 从而有利于优化能力的提高.

2) 多量子比特系统的所有基态概率幅均为量子比特相位的 n 元函数, 只需改变任何 1 个相位, 即可导致所有概率幅的更新. 这种通过逐个调整量子比特相位循环更新个体的方法, 使得个体更新的程度更为精细, 从而也增强了算法对解空间的遍历性.

3) 在基于多比特量子旋转门的个体更新策略中, 利用多比特量子旋转门, 一次操作即可实现粒子上所有概率幅的更新, 量子旋转门的酉性保证了量子位的“长度”不变, 有效避免了迭代序列的发散性, 从而提高了算法的收敛能力.

综上所述, 粒子的多比特概率幅编码和更新机制, 是以牺牲优化时间换取优化能力大幅度提升的, 这与无免费午餐定理是一致的.

7 结 论

本文提出了一种采用多比特概率幅编码的量子衍生粒子群算法. 该算法直接采用多比特量子系统中

各基态的概率幅对粒子编码, 基于张量积理论设计了多比特量子旋转门对粒子更新. 函数极值优化结果表明, 在相同优化时间下, 所提出算法的优化能力比传统算法有大幅度提升, 从而揭示出多比特概率幅编码是大幅度提高传统粒子群算法优化能力的有效途径.

参考文献(References)

[1] Kennedy J, Eberhart R C. Particle swarms optimization[C]. Proc of IEEE Int Conf on Neural Networks. New York: IEEE Press, 1995: 1942-1948.

[2] Guo W Z, Chen G L, Peng S J. Hybrid particle swarm optimization algorithm for vlsi circuit partitioning[J]. J of Software, 2011, 22(5): 833-842.

[3] 秦华, 万云芳, 张伟元, 等. 用粒子群算法进行单个非球面透镜的球差校正[J]. 计算物理, 2012, 29(3): 426-432.
(Qin H, Wan Y F, Zhang W Y, et al. Aberration correction of single aspheric lens with particle swarm algorithm[J]. Chinese J of Computational Physics, 2012, 29(3): 426-432.)

[4] Cai X J, Cui Z H, Zeng J C. Dispersed particle swarm optimization[J]. Information Processing Letters, 2008, 105(6): 231-235.

[5] Liu Y, Qin Z, Shi Z W. Center particle swarm optimization[J]. Neurocomputing, 2007, 70(4/5/6): 672-679.

[6] 张英杰, 邵岁锋. 一种基于云模型的云变异粒子群算法[J]. 模式识别与人工智能, 2011, 24(1): 90-96.
(Zhang Y J, Shao S F. Cloud mutation particle swarm optimization algorithm based on cloud model[J]. Pattern Recognition and Artificial Intelligence, 2011, 24(1): 90-96.)

[7] 方伟, 孙俊, 谢振平, 等. 量子粒子群优化算法的收敛性分析及控制参数研究[J]. 物理学报, 2010, 59(6): 3686-3694.
(Fang W, Sun J, Xie Z P, et al. Convergence analysis of quantum-behaved particle swarm optimization algorithm and study on its control parameter[J]. Acta Physica Sinica, 2010, 59(6): 3686-3694.)

[8] Sun J, Wu X J, Fang W, et al. Convergence analysis and improvements of quantum-behaved particle swarm optimization[J]. Information Sciences, 2012, 193: 81-103.

[9] Lu T C, Yu G R. An adaptive population multi-objective quantum inspired evolutionary algorithm for multi-objective 0/1 knapsack problems[J]. Information Sciences, 2013, 243: 39-56.

[10] Abdesslem L. A hybrid quantum inspired harmony search algorithm for 0-1 optimization problems[J]. J of Computational and Applied Mathematics, 2013, 253: 14-25.

[11] Gao J Q. A hybrid quantum inspired immune algorithm for multiobjective optimization[J]. Applied Mathematics and Computation, 2011, 217(9): 4754-4770.

[12] Han K H, Kim J H. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization[J]. IEEE Trans on Evolutionary Computation, 2002, 6(6): 580-593.

[13] 刘显德, 李盼池, 杨淑云. 量子衍生差分进化算法的设计与实现[J]. 信号处理, 2014, 30(6): 623-633.
(Liu X D, Li P C, Yang S Y. Design and implementation of quantum-inspired differential evolution algorithm[J]. J of Signal Processing, 2014, 30(6): 623-633.)

[14] Eberhart R C, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization[C]. Proc of IEEE Congress on Evolutionary Computation. New York: IEEE Press, 2000: 84-88.

[15] 李盼池, 王海英, 宋考平. 量子势阱粒子群优化算法的改进研究[J]. 物理学报, 2012, 61(6): 060302.
(Li P C, Wang H Y, Song K P. Research on improvement of quantum potential well-based particle swarm optimization algorithm[J]. Acta Physica Sinica, 2012, 61(6): 060302.)

[16] 赵玉新, 刘立强. 新兴元启发式优化算法[M]. 北京: 科学出版社, 2013: 240-251.
(Zhao Y X, Liu L Q. Emerging heuristic optimization algorithm[M]. Beijing: Science press, 2013: 240-251.)

(责任编辑：郑晓蕾)