多智能体遗传算法用于超高维函数优化*

钟伟才 薛明志 刘 静 焦李成
西安电子科技大学雷达信号处理国家重点实验室，西安 710071

摘要 基于智能体对环境的感知与反作用的能力提出了一种新的函数优化方法——多智能体遗传算法. 该方法将智能体固定在网格上，而每个智能体为了增加自身能量将与其邻域展开竞争或合作，同样，智能体也可利用自身的知识进行自学习来增加能量. 理论分析证明算法具有全局收敛性. 实验结果表明，多智能体遗传算法对维数高达甚至10000的函数，都能以较少的计算量获得高质量的解，充分说明算法具有很快的收敛速度.

关键词 智能体 遗传算法 函数优化 进化计算

遗传算法是通过模拟生物在自然界中的进化过程而形成的一种全局优化算法，目前已得到广泛的应用[1]. 对于函数优化问题，遗传算法是一种很有潜力的方法，主要在于它对函数的类型、搜索空间的形状等都没有任何限制. 但当维数增高时，搜索空间变大，变量间耦合度增强，从而导致遗传算法陷入局部极值的概率增大. 为提高算法性能，提出了一些新方法，如非单调适应值标度变换方法[2]，基于函数分解的可伸缩宏进化算法[3]，自适应演化算法[4](AEA)等，均获得了良好的效果.

文献[5]利用智能体解决了高达7000个皇后的问题，显示出智能体的优越性能. 受其启发，本文针对函数优化问题，将智能体对环境的感知和反作用的能力与遗传算法的搜索方式相结合，提出了多智能体遗传算法(multi-agent genetic algorithm, MAGA)，设计了4个进化算子来实现智能体间的竞争、合作、自学习等行为，其收敛速度远远高于传统算法. 而收敛速度慢，尤其当搜索空间较大时，一直是影响遗传算法实用性的关键因素之一，但本文提出的方法对上千、上万维的函数，都能快速找到高质量的解，这充分显示了方法的实用价值，同时也显示出智能体与遗传算法解函数优化问题的巨大潜力.

2003-01-17收稿，2003-04-28收修改稿
* 国家自然科学基金重点资助项目(批准号：60133010)
E-mail：neouma@163.com

1 多智能体遗传算法

根据文献[6]的定义，任何一个能够感知环境并反作用于环境的实体都可看作为智能体，而所有由多个自治成分构成的系统都可表达为一个智能体系统，其特性主要表现为：(1) 每个智能体只有局部感知能力；(2) 没有系统的全局控制；(3) 数据是分散的；(4) 计算是异步的. 由此可见，对于不同问题，智能体的具体含义不同，我们要根据问题灵活设计智能体的含义与行为.

1.1 用于函数优化的智能体

函数优化问题的数学模型为

$$\min f(x) \quad x = (x_1, \cdots, x_n) \in S, \quad (1)$$

其中 $f(x)$ 为目标函数，$S \subseteq R^n$ 表示边界为 $\underline{x}_i \leqslant x_i \leqslant \bar{x}_i$，$i = 1, \cdots, n$ 的 n 维搜索空间.

在本文中，一个智能体就表示目标函数的一个解，即一个实值向量$(x_1, \cdots, x_n)$. 所有智能体都被放在一个规模为 $L_{\text{size}} \times L_{\text{size}}$ 的网格 L 上，每个智能体占一个格点位置且不能移动. 由于智能体只有局部感知能力，因此它只能与周围的智能体发生相互作用，这就构成了如图1所示的智能体网格. 根据图1，智能体邻域的概念定义如下：

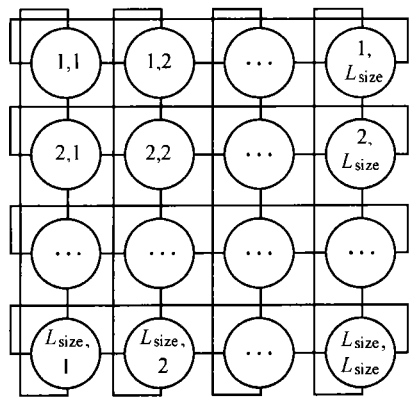

图1 智能体网格

圆圈代表智能体，圆中的数字代表该智能体的位置，有连线的两个智能体才能发生相互作用

定义1 将位置(i, j)的智能体表示成$L_{i,j}$，$i, j=1, 2, \cdots, L_{size}$，$L_{i,j}$的邻域为$Nbs_{i,j}$，$Nbs_{i,j}=\{L_{i_1,j}, L_{i,j_1}, L_{i_2,j}, L_{i,j_2}\}$，

其中

$$i_1=\begin{cases} i-1 & i\neq 1 \\ L_{size} & i=1 \end{cases},\quad j_1=\begin{cases} j-1 & j\neq 1 \\ L_{size} & j=1 \end{cases},$$

$$i_2=\begin{cases} i+1 & i\neq L_{size} \\ 1 & i=L_{size} \end{cases},\quad j_2=\begin{cases} j+1 & j\neq L_{size} \\ 1 & j=L_{size} \end{cases}.$$

每个智能体都具有一定的能量，其值等于目标函数值的相反数. 智能体进化的目标就是尽可能的增大自身的能量，这就导致了智能体间激烈的竞争，但这种竞争只能存在于智能体与其邻域间. 最终，能量低的智能体将死亡，而它的邻域将占领它的位置. 当然，合作也可能发生在智能体与其邻域间. 另一方面，由于智能体具有智能，它也可利用自己的知识来进行自学习.

1.2 智能体进化算子

邻域竞争算子和邻域正交交叉算子利用邻域的信息来分别实现智能体间的竞争与合作，变异算子和智能体自学习算子利用智能体自身的知识来增加能量. 设进化算子作用在智能体$L_{i,j}=(l_1, l_2, \cdots, l_n)$上，$\mathrm{Max}_{i,j}=(m_1, m_2, \cdots, m_n)$为$L_{i,j}$邻域内能量最大的智能体：

邻域竞争算子：若$L_{i,j}$比$\mathrm{Max}_{i,j}$的能量大，则可继续存活在网格上；否则必须死亡，空出的格点将被$\mathrm{Max}_{i,j}$占据——如果$U(0, 1)<P_o$，$\mathrm{Max}_{i,j}$则按(2)式产生新智能体$\mathrm{New}_{i,j}=(e_1, e_2, \cdots, e_n)$放在空出的格点上，否则先按(3)式将$\mathrm{Max}_{i,j}$映射到区间$[0, 1]$上，然后根据(4)式确定$\mathrm{New}'_{i,j}=(e'_1, e'_2, \cdots, e'_n)$，最后根据(5)式将$\mathrm{New}'_{i,j}$映射回区间$[\underline{x}_k, \bar{x}_k]$上以得到$\mathrm{New}_{i,j}$. 上述$U(\cdot, \cdot)$为均匀分布的随机数，$P_o$为占据概率.

$$e_k=\begin{cases} \underline{x}_k & (m_k+U(-1,1)\times(m_k-l_k))<\underline{x}_k \\ \bar{x}_k & (m_k+U(-1,1)\times(m_k-l_k))>\bar{x}_k \\ m_k+U(-1,1)\times(m_k-l_k) & \text{其他} \end{cases},\quad k=1,\cdots,n, \tag{2}$$

$$m'_k=(m_k-\underline{x}_k)/(\bar{x}_k-\underline{x}_k),\quad k=1,\cdots,n, \tag{3}$$

$$\mathrm{New}'_{i,j}=(m'_1,\cdots,m'_{i_1-1}, m'_{i_2}, m'_{i_2-1}, \cdots, m'_{i_1+1}, m'_{i_1}, m'_{i_2+1},\cdots,m'_n),\ 1<i_1<n,\ 1\leqslant i_2<n,\ i_1<i_2, \tag{4}$$

$$e_k=\underline{x}_k+e'_k\cdot(\bar{x}_k-\underline{x}_k),\quad k=1,\cdots,n. \tag{5}$$

邻域正交交叉算子：正交交叉算子是文献[7]提出的一种新交叉算子，篇幅所限，这里不再详述. 而邻域正交交叉算子将正交交叉算子[7]作用在$L_{i,j}$和$\mathrm{Max}_{i,j}$上以产生一个新的智能体来代替$L_{i,j}$.

变异算子：根据(6)式产生新智能体$\mathrm{New}_{i,j}=(e_1, e_2, \cdots, e_n)$：

$$e_k=\begin{cases} l_k & U(0,1)<\dfrac{1}{n} \\ l_k+G(0,1/t) & \text{其他} \end{cases},\quad k=1,\cdots,n, \tag{6}$$

其中$G(0, 1/t)$为Gauss分布的随机数，t为进化代数. 然后用$\mathrm{New}_{i,j}$代替$L_{i,j}$.

智能体自学习算子：该算子类似于一个小规模的多智能体遗传算法，其详细步骤在算法1中给出.

算法1 智能体自学习算子

$s\mathrm{Gen}$表示最大代数，$sR\in[0, 1]$表示相对搜索半径，$sL_{size}<L_{size}$，sP_m为预先设定的参数，$\mathrm{Eng}(\cdot)$表示智能体的能量. $s\mathrm{Best}^l$和$s\mathrm{CBest}^l$分别表示到第l代为止产生的最优智能体和第l代中的最优

智能体.

步骤1：根据(7)，(8)式产生 $sL_{\text{size}} \times sL_{\text{size}}$ 的智能体网格 $sL^0_{i',j'}$，i'，$j' = 1, 2, \cdots, sL_{\text{size}}$，更新 $s\text{Best}^0$，$l \leftarrow 0$；

$$sL_{i',j'} = \begin{cases} L_{i,j} & i' = 1 \text{ and } j' = 1 \\ \text{New}_{i',j'} & \text{其他} \end{cases}, \quad (7)$$

其中 $\text{New}_{i',j'} = (e_{i',j',1}, e_{i',j',2}, \cdots, e_{i',j',n})$ 由(8)式确定：

$$e_{i',j',k} = \begin{cases} \underline{x}_k & l_k \cdot U(1-sR, 1+sR) < \underline{x}_k \\ \bar{x}_k & l_k \cdot U(1-sR, 1+sR) > \bar{x}_k \\ l_k \cdot U(1-sR, 1+sR) & \text{其他} \end{cases}, \quad k = 1, \cdots, n. \quad (8)$$

步骤2：对 sL^l 上的每个智能体执行邻域竞争算子，最终得到 $sL^{l+1/2}$；

步骤3：对 $sL^{l+1/2}$ 上的每个智能体，如果 $U(0, 1) < sP_m$，则执行变异算子，最终得到 sL^{l+1}；

步骤4：从 sL^{l+1} 中找出 $s\text{CBest}^{l+1}$，如果 $\text{Eng}(s\text{CBest}^{l+1}) > \text{Eng}(s\text{Best}^l)$，则令 $s\text{Best}^{l+1} \leftarrow s\text{CBest}^{l+1}$，否则令 $s\text{Best}^{l+1} \leftarrow s\text{Best}^l$，$s\text{CBest}^{l+1} \leftarrow s\text{Best}^l$；

步骤5：如果 $l < s\text{Gen}$，则令 $l \leftarrow l+1$，并转步骤2；否则转步骤6；

步骤6：$L_{i,j} \leftarrow s\text{Best}^l$.

1.3 多智能体遗传算法

多智能体遗传算法将邻域竞争算子作用在每个智能体上，因此能量较低的智能体将被从网格上清除出去，以使得有较大的发展空间留给能量较高的智能体. 为了降低计算代价，智能体自学习算子只作用在当代中最优的智能体上. 每个智能体的进化只利用其邻域或自身的信息，因此整个算法没有全局性控制，从而是完全异步的. 算法2详细描述了多智能体遗传算法的整体流程.

算法2 多智能体遗传算法

Best^t 和 $C\text{Best}^t$ 分别表示到第 t 代为止产生的最优智能体和第 t 代中的最优智能体.

步骤1：初始化智能体网格 L^0，更新 Best^0，$t \leftarrow 0$；

步骤2：对 L^t 上的每个智能体执行邻域竞争算子，最终得到 $L^{t+1/3}$；

步骤3：对 $L^{t+1/3}$ 中的每个智能体，如果 $U(0, 1) < P_c$，则执行邻域正交交叉算子，最终得到 $L^{t+2/3}$；

步骤4：对 $L^{t+2/3}$ 中的每个智能体，如果 $U(0, 1) < P_m$，则执行变异算子，最终得到 L^{t+1}；

步骤5：从 L^{t+1} 中找出 $C\text{Best}^{t+1}$，将智能体自学习算子作用在 $C\text{Best}^{t+1}$ 上；

步骤6：如果 $\text{Eng}(C\text{Best}^{t+1}) > \text{Eng}(\text{Best}^t)$，则令 $\text{Best}^{t+1} \leftarrow C\text{Best}^{t+1}$，否则令 $\text{Best}^{t+1} \leftarrow \text{Best}^t$，$C\text{Best}^{t+1} \leftarrow \text{Best}^t$；

步骤7：如果停止准则满足，则输出 Best^t，停止；否则令 $t \leftarrow t+1$，并转步骤2.

2 多智能体遗传算法的收敛性分析

如果 $(x_1, \cdots, x_n)$ 中的分量 x_i 用 M 位二进制串表示，则相当于将区间 $[\underline{x}_i, \bar{x}_i]$ 量化为 2^M 个离散值，其精度为 $\varepsilon = (\bar{x}_i - \underline{x}_i)/2^M$，因此本文直接通过实数编码来分析MAGA的收敛性. 设所需精度为 ε，则搜索空间 S 可看成是离散空间，其大小为 $|S| = \prod_{i=1}^n \lceil (\bar{x}_i - \underline{x}_i)/\varepsilon \rceil$，每个元素 $x (\in S)$ 就是一个智能体. 令 $\boldsymbol{E} = \{\text{Eng}(x) | x \in S\}$，显然 $|\boldsymbol{E}| \leqslant |S|$，于是 $\boldsymbol{E}$ 可以写成 $\boldsymbol{E} = \{E^1, E^2, \cdots, E^{|\boldsymbol{E}|}\}$，这里 $E^1 > E^2 > \cdots > E^{|\boldsymbol{E}|}$. 根据能量的不同又可把集合 S 划分为若干非空子集 $\{S^i\}$，这里

$$S^i = \{x \mid x \in S \text{ and } \text{Eng}(x) = E^i\}, \quad i = 1, 2, \cdots, |\boldsymbol{E}|, \quad (9)$$

$$\sum_{i=1}^{|\boldsymbol{E}|} |S^i| = |S|; \forall i \in \{1, 2, \cdots, |\boldsymbol{E}|\}, \quad S^i \neq \emptyset; \forall i \neq j, S^i \cap S^j = \emptyset; \bigcup_{i=1}^{|\boldsymbol{E}|} S^i = S, \quad (10)$$

显然，E^1 即为全局最优解，且集合 S^1 包含了所有能量等于 E^1 的智能体.

在MAGA的整个进化过程中，智能体的个数是保持不变的. 令 $\mathcal{L}$ 表示所有智能体网格的集合，由于一个智能体网格中的智能体是允许相同的，因此有(11)式：

$$|\mathscr{L}| = \begin{pmatrix} |S| + L_{size} \times L_{size} - 1 \\ L_{size} \times L_{size} \end{pmatrix}. \quad (11)$$

为了衡量智能体网格的优劣，定义智能体网格 L 的能量为

$$\mathrm{Eng}(L) = \max\{\mathrm{Eng}(L_{i,j}) \mid i,j = 1,\cdots,L_{size}\}, \quad (12)$$

这样对 $\forall L \in \mathscr{L}$，均有 $E^{|E|} \leqslant \mathrm{Eng}(L) \leqslant E^1$. 因此可以把集合 $\mathscr{L}$ 划分为非空子集 $\{\mathscr{L}^i\}$，

$$\mathscr{L}^i = \{L \mid L \in \mathscr{L} \text{ and } \mathrm{Eng}(L) = E^i\}, \quad i = 1,2,\cdots,|E|, \quad (13)$$

$$\sum_{i=1}^{|E|} |\mathscr{L}^i| = |\mathscr{L}|; \forall i \in \{1,2,\cdots,|\mathscr{L}|\},$$

$$\mathscr{L}^i \neq \emptyset; \forall i \neq j, \mathscr{L}^i \cap \mathscr{L}^j = \emptyset; \bigcup_{i=1}^{|E|} \mathscr{L}^i = \mathscr{L}, \quad (14)$$

集合 $\mathscr{L}^1$ 包含所有能量为 E^1 的智能体网格.

令 L^{ij}，$i = 1, 2, \cdots, |E|$，$j = 1, 2, \cdots, |\mathscr{L}^i|$ 表示 $\mathscr{L}^i$ 中的第 j 个智能体网格. 在智能体进化算子的作用下，从 L^{ij} 到 L^{kl} 的状态转移可表示为 $L^{ij} \to L^{kl}$. 令 $p_{ij.kl}$ 表示从 L^{ij} 到 L^{kl} 的转移概率，$p_{ij.k}$ 表示从 L^{ij} 到 $\mathscr{L}^k$ 中任一网格的转移概率，而 $p_{i.k}$ 表示从 $\mathscr{L}^i$ 中任一网格到 $\mathscr{L}^k$ 中任一网格的转移概率. 显然有：

$$p_{ij.k} = \sum_{l=1}^{|\mathscr{L}^k|} p_{ij.kl}; \sum_{k=1}^{|E|} p_{ij.k} = 1; p_{i.k} \geqslant p_{ij.k}, \quad (15)$$

明确了上述概念后，在文献[8]的基础上，我们将给出 MAGA 全局收敛性的理论证明.

定理 1[9] 令 $\boldsymbol{P}'$ 是一个 n 阶可归约随机矩阵，即通过相同的行变换和列变换后可以得到 $\boldsymbol{P}' = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{T} \end{pmatrix}$，其中 $\boldsymbol{C}$ 是一个 m 阶本原随机矩阵且 $\boldsymbol{R}$，$\boldsymbol{T} \neq \boldsymbol{0}$. 则

$$\boldsymbol{P}'^{\infty} = \lim_{k\to\infty} \boldsymbol{P}'^{k} = \lim_{k\to\infty} \begin{pmatrix} \boldsymbol{C}^k & \boldsymbol{0} \\ \sum_{i=0}^{k-1} \boldsymbol{T}^i \boldsymbol{R} \boldsymbol{C}^{k-i} & \boldsymbol{T}^k \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}^{\infty} & \boldsymbol{0} \\ \boldsymbol{R}^{\infty} & \boldsymbol{0} \end{pmatrix} \quad (16)$$

是一个稳定的随机矩阵且 $\boldsymbol{P}'^{\infty} = 1'\boldsymbol{p}'^{\infty}$，这里 $\boldsymbol{p}'^{\infty} = \boldsymbol{p}'^{0}\boldsymbol{P}'^{\infty}$ 惟一确定且与初始分布无关，$\boldsymbol{p}'^{\infty}$ 满足 $\boldsymbol{p}'^{\infty}_i > 0$，$1 \leqslant i \leqslant m$ 且 $\boldsymbol{p}'^{\infty}_i = 0$，$m < i \leqslant n$.

定理 2 在多智能体遗传算法中，对 $\forall i$，$k \in \{1, 2, \cdots, |E|\}$，有 $p_{i.k} = \begin{cases} > 0 & k \leqslant i \\ = 0 & k > i \end{cases}$.

证明 对 $\forall L^{ij} \in \mathscr{L}^i$，$i = 1, 2, \cdots, |E|$，$j = 1, 2, \cdots, |\mathscr{L}^i|$，均 $\exists \boldsymbol{x}^* = (x_1, \cdots, x_n) \in L^{ij}$，$\mathrm{Eng}(\boldsymbol{x}^*) = E^i$. 设在智能体进化算子的作用下，$L^{ij}$ 转变为 L^{kl}；若 L^{ij} 为第 t 代，则 L^{kl} 为第 $(t+1)$ 代，为了方便分别记为 L^t 和 L^{t+1}.

首先，由算法 2 Step 6 可得(17)式.

$$\mathrm{Eng}(C\mathrm{Best}^{t+1}) = \mathrm{Eng}(\mathrm{Best}^{t+1}) \geqslant \mathrm{Eng}(\mathrm{Best}^t) = \mathrm{Eng}(\boldsymbol{x}^*), \quad (17)$$

因此有：$\mathrm{Eng}(L^{t+1}) \geqslant \mathrm{Eng}(L^t) \Rightarrow k \leqslant i \Rightarrow \forall k > i$，$p_{ij.kl} = 0 \Rightarrow \forall k > i$，$p_{ij.k} = \sum_{l=1}^{|\mathscr{L}^k|} p_{ij.kl} = 0 \Rightarrow \forall k > i$，$p_{i.k} = 0$. (18)

其次，在邻域竞争算子中 $\boldsymbol{x}^*$ 必然胜过其所有邻域，因此 $\boldsymbol{x}^* \in L^{t+1/3}$；对 $\boldsymbol{x}^*$ 执行邻域正交交叉算子的概率为 P_c，因此 $\boldsymbol{x}^*$ 属于 $L^{t+2/3}$ 的概率为 $(1 - P_c)$，而对 $\boldsymbol{x}^*$ 执行变异算子的概率 Pr_1 为 $Pr_1 = (1 - P_c) \cdot P_m > 0$.

$\exists \boldsymbol{x}' \in L^{t+1}(L^{kl})$，$k \leqslant i$，$\mathrm{Eng}(\boldsymbol{x}') = E^k$，不妨设 $\boldsymbol{x}'$ 有 n_1 个分量，$x'_1, \cdots, x'_{n_1}$，与 $\boldsymbol{x}^*$ 对应位置上的分量不同. 通过变异算子由 $\boldsymbol{x}^*$ 产生 $\boldsymbol{x}'$ 的概率 Pr_2 为 $Pr_2 = \left(1 - \frac{1}{n}\right)^{n-n_1} \cdot \prod_{i=1}^{n_1} \frac{\sqrt{t}}{\sqrt{2\pi}} e^{-\frac{t(x_i - x'_i)^2}{2}} > 0$.

由以上分析可知，在智能体进化算子作用后，从 L^{ij} 转到 $\mathscr{L}^k$ 中任一智能体网格的概率 $p_{ij.k}$ 为 $p_{ij.k} > Pr_1 \times Pr_2 > 0$. 因此，$\forall k \leqslant i$，$p_{i.k} \geqslant p_{ij.k} > 0$. 证毕.

该定理说明，能量低的网格可以转移到能量相同或更高的网格，但能量高的网格不能转移到能量低的网格，因此一旦 MAGA 进入到集合 $\mathscr{L}^1$ 中就不可能逃逸出来.

定理 3 多智能体遗传算法具有全局收敛性.

证明 每个 $\mathscr{L}^i$，$i = 1, 2, \cdots, |E|$ 可以看作为时齐有限 Markov 链上的一个状态，根据定理 2，该 Markov 链的转移矩阵 P' 可表示为(19)式：

$$\boldsymbol{P}' = \begin{pmatrix} p_{1.1} & 0 & \cdots & 0 \\ p_{2.1} & p_{2.2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_{|\boldsymbol{E}|.1} & p_{|\boldsymbol{E}|.2} & \cdots & p_{|\boldsymbol{E}|.|\boldsymbol{E}|} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{R} & \boldsymbol{T} \end{pmatrix}, \tag{19}$$

其中 $\boldsymbol{R} = (p_{2.1}, p_{3.1}, \cdots, p_{|\boldsymbol{E}|.1})^T > 0$, $\boldsymbol{T} \neq \boldsymbol{0}$, $\boldsymbol{C} = (p_{1.1}) = (1) \neq \boldsymbol{0}$.

根据定理1可得(20)式.

$$\boldsymbol{P}'^{\infty} = \lim_{k\to\infty} \boldsymbol{P}'^{k} = \lim_{k\to\infty} \begin{pmatrix} \boldsymbol{C}^k & \boldsymbol{0} \\ \sum_{i=0}^{k-1} \boldsymbol{T}^i \boldsymbol{R} \boldsymbol{C}^{k-i} & \boldsymbol{T}^k \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}^{\infty} & \boldsymbol{0} \\ \boldsymbol{R}^{\infty} & \boldsymbol{0} \end{pmatrix}, \tag{20}$$

其中 $\boldsymbol{C}^{\infty} = 1$, $\boldsymbol{R}^{\infty} = (1, 1, \cdots, 1)^T$, 这样 $\boldsymbol{P}'^{\infty}$ 就是一个稳定的随机矩阵, 且 $\boldsymbol{P}'^{\infty} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}$.

由此可得 $\lim_{t\to\infty} Pr\{\mathrm{Eng}(L^t) = E^1\} = 1$, 其中 Pr 为概率, 所以多智能体遗传算法具有全局收敛性. 证毕.

3 数值实验与分析

在使用 MAGA 前, 需要预先设定一些参数: $L_{size} \times L_{size}$ 实际对应于遗传算法中群体的大小, 因此 L_{size} 不易选得过大或过小, 一般选 5～10 即可. P_o 实际控制了算法在利用已有信息与探索新空间之间的平衡——当 $P_o < 0.5$ 时, 侧重于探索新空间, 而当 $P_o > 0.5$ 时, 侧重于利用已有信息. P_c 一般选小于 0.5 即可, 否则计算量增加太大. P_m 相应于一般的变异概率. 智能体自学习算子实际是一个小规模的 MAGA, 所以其 4 个参数也非常易于选取——从计算量方面考虑, sL_{size} 一般选小于 5, 而 sGen 一般选 5～10. sR 控制搜索半径, 起到局部搜索的作用, 不易选得过大, 一般取小于 0.3. sP_m 与 P_m 类似. 本实验中 MAGA 参数设置为: $L_{size} = 5$, $P_o = 0.2$, $P_c = 0.1$, $P_m = 0.1$, $sL_{size} = 3$, $sR = 0.2$, $sP_m = 0.05$, sGen $= 10$; 停止准则为: 如果 $f_{min} \neq 0$, 则考察式子 $|f_{best} - f_{min}| < \varepsilon \cdot |f_{min}|$ 是否成立, 否则考察式子 $|f_{best}| < \varepsilon$, 此处 f_{min} 和 f_{best} 分别表示全局最优解和到当前代为止找到的最优解. 实验所用的 4 个多峰标准函数如(21)～(24)式所示.

Schwefel 函数: $f_1(\boldsymbol{x}) = \sum_{i=1}^{n} (-x_i \sin\sqrt{|x_i|})$, $S = [-500, 500]^n$, $f_{min} = -418.983 \times n$, (21)

Rastrigin 函数: $f_2(\boldsymbol{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$, $S = [-5.12, 5.12]^n$, $f_{min} = 0$, (22)

Ackley 函数: $f_3(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$, $S = [-32, 32]^n$, $f_{min} = 0$, (23)

Griewank 函数: $f_4(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(x_i/\sqrt{i}\right) + 1$, $S = [-600, 600]^n$, $f_{min} = 0$. (24)

3.1 MAGA 与 AEA[4] 对 20～1000 维函数优化结果比较

在 AEA 的停止准则中, 对 f_1 取 $\varepsilon = 10^{-4}$, 对 f_2 取 $\varepsilon = 10^{-1}$, 对 f_3 和 f_4 均取 $\varepsilon = 10^{-3}$. 为了一致, 在 MAGA 中, 4 个函数均取 $\varepsilon = 10^{-4}$. 表 1 给出了平均函数评价次数的比较结果, 其中 MAGA 的结果为 50 次随机运行的平均. 由表 1 可见, 对于 f_1, 当 $n \leqslant 100$ 时, MAGA 所用评价次数略大于 AEA, 而当 $200 \leqslant n \leqslant 1000$ 时, 略小于 AEA. 对于 f_2, 当 $n \leqslant 200$ 时, MAGA 所用评价次数大于 AEA, 而当 $400 \leqslant n \leqslant 1000$ 时, 小于 AEA. 对于 f_3 和 f_4, MAGA 所用评价次数均远远小于 AEA, 而且 MAGA 仅用几千次评价就使解的精度达到了 10^{-4}. 总的来说, 虽然 MAGA 的评价次数随着维数的增长有所增长, 但增长速度明显慢于 AEA, 显示出了其在解高维问题时的优越性能.

表 1 MAGA 与 AEA 对 20～1000 维函数优化平均函数评价次数的比较

n	f_1		f_2		f_3		f_4	
	MAGA	AEA	MAGA	AEA	MAGA	AEA	MAGA	AEA
20	2,483	1,603	4,301	1,247	3,583	7,040	2,566	3,581
100	5,443	5,106	10,265	4,789	5,410	22,710	4,447	17,228
200	7,284	8,158	14,867	10,370	6,051	43,527	5,483	36,760
400	12,368	13,822	17,939	23,588	6,615	78,216	6,249	61,975
1000	22,827	23,867	20,083	46,024	7,288	160,940	7,358	97,660

3.2 MAGA对1000～10000维函数优化的性能分析

本实验对 n 在1000～10000中采样，间隔为500. 将MAGA在每个采样的维数上随机运行50次，图2～5给出了平均评价次数随维数的变化. 由图可知，对于 f_1，虽然评价次数随维数的增长较明显，但即使当维数增到10000时，评价次数也少于200000. 对于 f_2，维数对评价次数没有明显的影响，尤其当维数大于1000时，评价次数则分布在10000～20000间. 对于 f_3，随着维数的增长，评价次数增长速度降低，当维数增到10000时，也只用了约8000次评价. 对于 f_4，评价次数随维数的增长速度也比较缓慢，当维数大于2000时，评价次数则分布在10000～30000间.

将进化算法用于上万维的函数优化问题在国内外还均未见报道，而本实验的结果充分证明了MAGA对超高维函数优化的能力，这主要得益于智能体邻域间的作用和智能体的自学习能力. 在进化算法中，评价次数是反映算法时间复杂度的主要因素，而MAGA与传统遗传算法相比，除了评价次数也没有更复杂的操作，由此可见MAGA具有很快的收敛速度，这对于实际应用具有很大价值.

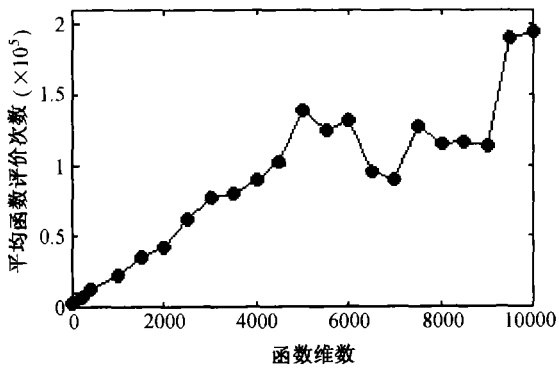

图2 f_1 平均评价次数随维数的变化

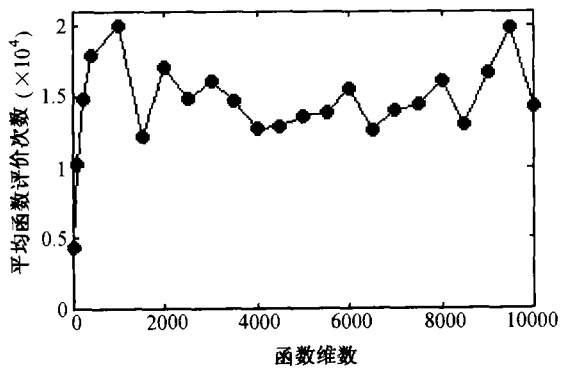

图3 f_2 平均评价次数随维数的变化

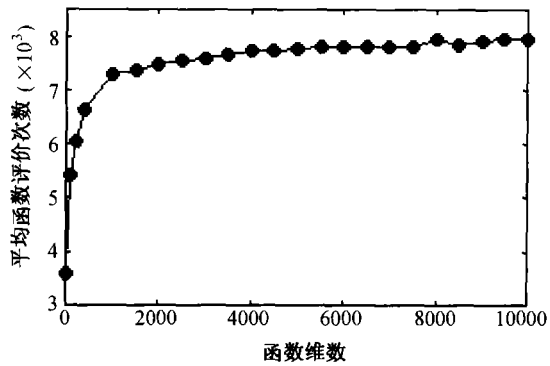

图4 f_3 平均评价次数随维数的变化

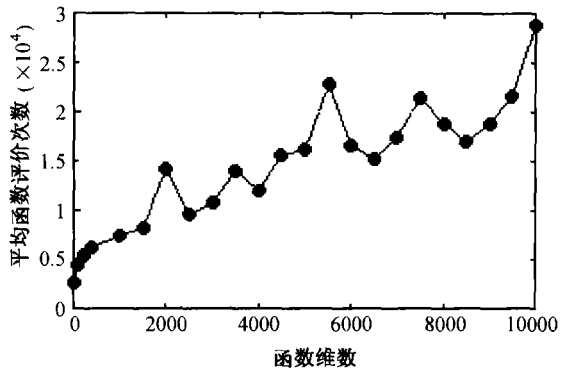

图5 f_4 平均评价次数随维数的变化

参 考 文 献

1 潘 进，等. 用遗传算法确定短序列正交多子波. 自然科学进展，2000，10(3)：266

2 李敏强，等. 遗传算法的一种非单调适应值标度变换方法. 自然科学进展，2001，11(5)：530

3 谢 涛，等. 基于函数分解的可伸缩宏进化算法. 自然科学进展，2001，11(6)：662

4 潘正君，等. 演化计算. 北京：清华大学出版社，1998. 72～76

5 韩 靖，等. AER模型中的智能涌现. 模式识别与人工智能，2002，15(2)：134

6 Russell S, et al. Artificial Intelligence: A modern Approach. New York: Prentice-Hall, 1995

7 Leung Y W, et al. An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Trans on Evolutionary Computation, 2001, 5(1): 41

8 Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Trans on Neural Networks, Special Issue on Evolutional Computing, 1994, 5(1): 96

9 Iosifescu M. Finite Markov Processes and Their Applications. Chichester: Wiley, 1980