文章编号: 1001-0920 (2012) 06-0827-06

处理动态优化问题的捕食元胞遗传算法

陈 昊[1], 黎 明[2], 陈 曦[2]

(1. 南京航空航天大学 自动化学院，南京 210016；2. 南昌航空大学 无损检测技术教育部重点实验室，南昌 330063)

摘 要: 根据自然界中的捕食关系, 提出一种捕食策略来代替元胞遗传算法中的演化规则, 并构建了基于捕食策略的元胞遗传算法以处理动态环境下的优化问题. 在元胞空间中, 捕食者对其捕食范围内的被捕者进行猎取并捕获其中最弱的一个. 对捕食策略中种群规模的相互关系进行了研究, 通过引入正交交叉算子进一步提高了算法的搜索能力. 选择不同强度、复杂度的动态优化问题进行算法性能验证, 所得结果表明新算法具有良好的处理动态优化问题的能力.

关键词: 捕食策略；元胞遗传算法；动态环境；正交交叉算子

中图分类号: TP18 **文献标识码:** A

Predator-prey cellular genetic algorithm for solving dynamic optimization problems

CHEN Hao[1], *LI Ming*[2], *CHEN Xi*[2]

(1. College of Automation Engineering，Nanjing University of Aeronautics and Astronautics，Nanjing 210016，China; 2. Key Laboratory of Nondestructive Test of Ministry of Education，Nanchang Hangkong University，Nanchang 330063，China. Correspondent：LI Ming，E-mail：limingniat@hotmail.com)

Abstract: This paper proposes a predator-prey cellular genetic algorithm to solving dynamic optimization problems. A predator-prey model replaces the evolution rule in regular cellular genetic algorithm, which is proposed based on the predator-prey relationship in real world. In grid-world, each predator captures the weakest prey in its neighborhood. The population size of predator and prey scheme is researched. Orthogonal crossover operator is introduced to further improve the search ability of the algorithm. Three dynamic optimization problems with different complexity are selected to verify the algorithm performance. The computation results show that the proposed algorithm has the better performance in dealing with the dynamic optimization problems.

Key words: predator-prey scheme；cellular genetic algorithm；dynamic environment；orthogonal crossover

1 引 言

在工程实践中, 大多数优化问题都属于动态优化问题, 这类问题的设计变量、目标函数和约束条件等都随着时间的变化而变化, 已得到的最优解会随着上述因素的变化而失去其有效性. 为了更好地处理动态优化问题, 需要一类能够持续地适应环境变化的优化方法[1]. 这要求算法能够持续地保持一定的全局搜索能力, 以保证在环境变化时能够快速适应并找到新的最优解. 因此, 增加并保持种群的多样性是动态进化算法研究的根本出发点之一[2-3].

元胞遗传算法是遗传算法的重要分支, 是一种将遗传算法与元胞自动机相结合的优化方法. 在元胞结构下, 元胞遗传算法可以得到优于同类算法的种群多样性, 具有有效处理动态优化问题的潜力. 元胞遗传算法是 Manderick 和 Spiessens 提出的[4], 近 10 年来该领域的研究逐渐活跃. Ishibuchi 等人[5]提出了具有双邻域拓扑结构的元胞遗传算法, 两个邻域空间分别用于全局寻优和局部竞争. Giacobini 等人[6]对不同演化规则下元胞遗传算法的选择压力进行了对比分析. Nakashima 等人[7]对不同选择策略、拓扑结构和最优保留策略的元胞遗传算法的性能进行了研究. 张俞等人[8]通过元胞空间中个体的繁殖与存活能力对元胞

收稿日期: 2010-12-07；**修回日期:** 2011-03-20.

基金项目: 国家自然科学基金项目(60963002)；江西省自然科学基金项目(2009GZS0090).

作者简介: 陈昊(1982−), 男, 博士生, 从事进化计算、动态环境的研究；黎明(1964−), 男, 教授, 博士生导师, 从事人工智能、模式识别等研究.

遗传算法的演化规则进行了研究.

捕食关系是自然界中普遍存在的物种间相互作用的基本关系之一, 许多优化算法的构建来源于对生态捕食模型的模拟. Li 等人[9]通过捕食机制改进了多目标优化算法, 使被捕食种群规模能够随可行空间的变化而动态改变, 并通过混合交叉算子实现自适应搜索. Deb 和 Bhaskara[10]通过捕食策略来改进遗传算法的选择策略. 蒋忠中等人[11]提出了一种捕食策略, 通过控制搜索空间的大小实现算法的局域搜索和全局搜索, 将其与遗传算法相结合可用于处理时间窗车辆路径问题.

本文将提出一种新的双种群捕食策略以代替元胞遗传算法中的演化规则, 通过捕食模型确定个体的生死状态, 并根据种群控制模型确定新一代种群的规模, 进而构建动态环境下基于捕食策略的元胞遗传算法. 通过对非周期、周期和带噪声周期动态环境下的动态测试问题进行性能验证, 所得结果表明了算法的有效性.

2 基于演化规则的元胞遗传算法

2.1 基本概念

元胞自动机的数学表达式为 $A=(L_d,S,N_d,f)$. 其中: A 为元胞自动机; L_d 为维数是 d 的元胞空间; S 为元胞的状态集合; N_d 表示邻居构成, 如: Von Neumann 型, Moore 型和扩展 Moore 型等; f 为演化规则, 可根据邻居内元胞的状态对中心元胞进行状态转换.

图 1 为 Moore 型邻居结构的示意图. 其中: 中间位置的元胞为中心元胞; 其他元胞的集合为中心元胞的 Moore 型邻居; 灰色与白色表示生与死两种元胞状态. 本文采用此种邻居结构.

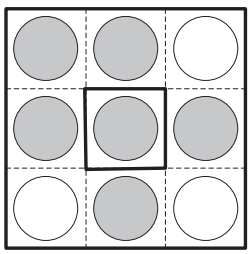


图 1 规则元胞空间中 Moore 型邻居结构

2.2 基于演化规则的元胞遗传算法

在基于演化规则的元胞遗传算法中, 种群中的个体被置于 2 维的元胞空间中. 算法开始时, 将初始化的种群依次置于元胞空间中, 并随机初始化元胞的生死状态. 每个元胞中的个体仅与其邻居的个体进行遗传操作. 遗传操作后, 对所产生的子代个体进行适应值估算, 若优于中心元胞内的个体, 则进行替换. 每一代元胞的生死状态由演化规则确定. 目前, 元胞遗传算法所采用的演化规则多由元胞自动机的研究中直接引入, 如生命游戏规则和改进的生命游戏规则 (分别表示为规则 I 和规则 II) 等.

规则 I

$$\text{if } S^t=1,\ \text{then } S^{t+1}=\begin{cases}1, & |N_S|=2,3;\\ 0, & |N_S|\neq 2,3.\end{cases}$$

$$\text{if } S^t=0,\ \text{then } S^{t+1}=\begin{cases}1, & |N_S|=3;\\ 0, & |N_S|\neq 3.\end{cases}\tag{1}$$

规则 II

$$\text{if } S^t=1,\ \text{then } S^{t+1}=\begin{cases}1, & |N_S|=2,4,6,8;\\ 0, & |N_S|\neq 2,4,6,8.\end{cases}$$

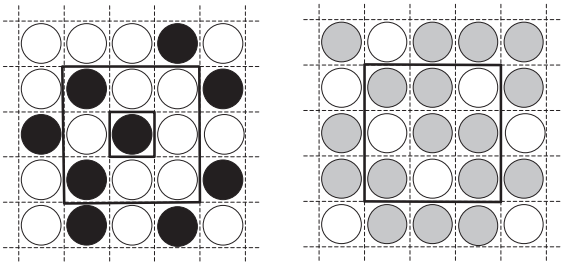
$$\text{if } S^t=0,\ \text{then } S^{t+1}=\begin{cases}1, & |N_S|=1,3,5,7;\\ 0, & |N_S|\neq 1,3,5,7.\end{cases}\tag{2}$$

在此类规则中, 中心元胞的新一代的状态仅与当前代的邻居内存活的个体数目有关, 而与个体适应值的优劣无关. 然而在自然界中, 生物个体生存能力的优劣以及与生存空间内其他个体的相互竞争关系等都对其生死演化有着直接影响. 此外, 在处理优化问题时通常需要根据实际情况选择演化规则, 若选择不当, 则容易造成元胞空间内个体的生命危机, 从而使得优化失败.

3 捕食元胞遗传算法

3.1 捕食模型

在捕食模型中, 捕食种群与被捕食种群被置于两个具有相同结构的元胞空间中, 如图 2 所示. 图 2(a) 与图 2(b) 分别为捕食与被捕食种群的示意图, 其中黑色与灰色分别表示存活状态的捕食与被捕食个体, 白色表示死亡状态的个体; 图 2(b) 中粗线内的区域为图 2(a) 中心元胞中捕食个体的捕食邻居.



(a) 捕食种群 (b) 被捕食种群

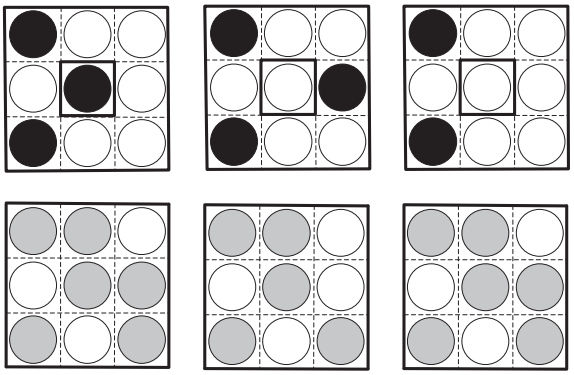
图 2 元胞空间中的捕食种群与被捕食种群

捕食个体的捕食能力和被捕食个体的逃脱能力均与个体的适应值有关. 对极大化优化问题适应值高的捕食和被捕食个体具有更强的捕食或逃脱能力, 从而获得更高的存活机会. 捕食能力与逃脱能力由其适应值的升岭形分布函数确定, 即

$$p(x)=\frac{1}{2}+\frac{1}{2}\sin\Big(\frac{\pi}{f_{\max}-f_{\min}}\Big(f(x)-\frac{f_{\max}+f_{\min}}{2}\Big)\Big). \quad (3)$$

其中: $p(x)$ 为个体 x 的捕食能力或逃脱能力, $f(x)$ 为 x 的适应度, $f_{\max}$ 和 $f_{\min}$ 分别为 x 所在种群中个体的最大适应值与最小适应值.

在捕食操作中, 捕食个体仅对其捕食邻居内的个体进行捕食, 图 3 为捕食操作的示意图. 对于一个存活状态的捕食个体 x (如图 3(a) 上图所示), 其潜在捕食对象位于其捕食邻居内 (如图 3(a) 下图所示). 如果在捕食邻居内存在逃脱能力小于 x 捕食能力的个体, 则捕食成功, x 对其中逃脱能力最弱的一个进行捕食, 并移动到死亡个体所在的元胞位置, 如图 3(b) 所示. 如果在捕食邻居内不存在逃脱能力小于 x 捕食能力的个体, 则捕食失败, x 所在元胞变为死亡状态, 如图 3(c) 所示.



(a) 捕食前　(b) 捕食成功　(c) 捕食失败

图 3　捕食操作示意图

3.2　种群控制

在自然界中, 任何种群的规模都与其天敌数量的多少、食物的丰富程度有关. 在捕食模型中, 对于捕食种群, 其种群规模的增长率与被捕食种群规模成正比; 对于被捕食种群, 其种群规模的增长率与捕食种群成反比, 与空白元胞数成正比. 采用下式对捕食与被捕食种群的规模进行控制:

$$(m'_t,n'_t)=\mathrm{PredatoryOperator}(m_t,n_t), \quad (4)$$

$$\begin{cases} m_{t+1}=m'_t\exp\Big(\dfrac{n'_t-m'_t}{N}\Big),\\ n_{t+1}=n'_t+n'_t\Big(1-\dfrac{n'_t}{N}\Big). \end{cases} \quad (5)$$

其中: m_t 和 m'_t 分别为 t 时刻捕食操作前后捕食种群的规模, n_t 和 n'_t 为被捕食种群规模. 由式 (5) 可知, 若 $m'_t>n'_t$, 则说明捕食操作后捕食种群处于食物匮乏状态, 种群规模减小, 即 $m_{t+1}<m'_t$; 若 $m'_t<n'_t$, 则说明捕食操作后捕食种群的食物丰盈, 种群规模增长, 即 $m_{t+1}>m'_t$; 捕食操作后被捕食者种群始终处于增长状态, 其增长率 $(1-n'_t/N)$ 与空白元胞数成正比.

考虑到 $n'_t+m'_t=m_t$, 可用 d_t 表示 t 时刻捕食成功的捕食个体数目, $0\leqslant d_t\leqslant\min(n_t,m_t)$, 则有

$$d_t=\mathrm{PredatoryOperator}(n_t,m_t), \quad (6)$$

$$\begin{cases} n_{t+1}=d_t\exp\Big(\dfrac{m_t-2d_t}{N}\Big),\\ m_{t+1}=(m_t-d_t)\Big(2-\dfrac{m_t-d_t}{N}\Big). \end{cases} \quad (7)$$

对捕食模型的平衡状态进行分析: 当捕食模型满足 $n_{t+1}=n_t$, $m_{t+1}=m_t$ 时, 系统达到平衡状态, 可解得 $d_t=(\sqrt{N}-\sqrt{N-m_t})\sqrt{N-m_t}$, 此时有

$$\begin{cases} n_t=d_t\exp\Big(\Big(\dfrac{N}{2}-d_t-\dfrac{\sqrt{N(N-4d_t)}}{2}\Big)\Big/N\Big),\\ m_t=\dfrac{N}{2}+d_t-\dfrac{\sqrt{N(N-4d_t)}}{2}. \end{cases} \quad (8)$$

3.3　正交交叉算子

与静态优化问题相同, 一种有效的动态进化算法同样需要在保证一定全局搜索能力的同时具有良好的局部搜索能力. 通过引入正交交叉算子可进一步提高算法的搜索能力.

选取正交表 $L_k(2^l)$, L 为编码长度, 将 $\{1,\cdots,L\}$ 随机划分为 l 个不相交的子集 F_i, $i\in[1,l]$. 将父代个体 P_1 和 P_2 的交叉操作视为 l 因素 2 水平的实验. l 因素对应 l 个不相交的非空子集 F_i. 2 水平对应子代个体在子集 F_i 相应基因位上的 2 种取值: 取 P_1, 则为 0-水平; 取 P_2, 则为 1- 水平. 被选取的父代个体按正交表 $L_k(2^l)$ 产生 k 个中间子代并进行内部竞争, 其中最优的 2 个子代将作为下一代个体.

3.4　捕食元胞遗传算法

将捕食操作算子与种群控制算子引入元胞遗传算法中, 提出捕食元胞遗传算法 (PCGA), 并将进一步引入正交交叉算子的捕食元胞遗传算法称为改进的捕食元胞遗传算法 (IPCGA). 算法的具体流程如下.

begin

初始化捕食种群 $P_1(0)$ 和被捕食种群 $P_2(0)$;

将 $P_1(0)$ 和 $P_2(0)$ 的个体置于元胞空间内, 并初始化生死状态.

repeat

适应值计算

$f_1=\mathrm{evaluate}(P_1(t)), f_2=\mathrm{evaluate}(P_2(t))$;

捕食能力计算 $\mathrm{Fit}_1=\mathrm{normlize}(f_1)$;

逃脱能力计算 $\mathrm{Fit}_2=\mathrm{normlize}(f_2)$.

// 捕食-被捕食模型

捕食操作

$(P'_1(t),P'_2(t))=\mathrm{PredatoryOperator}(P_1(t),P_2(t))$;

种群控制

$(|P_1(t+1)|,|P_2(t+1)|)=\text{PopControl}(|P_1'(t)|,|P_2'(t)|)$,
$(P_1(t+1)=[P_1'(t),|P_1'(t+1)|-P_1'(t)|$ 随机个体$]$.
if $|P_2(t+1)|>|P_2'(t)|$, then
$(P_2(t+1)=[P_2'(t),|P_2(t+1)|-|P_2'(t)|$ 随机个体$]$,
else $(P_2(t+1)=[P_2'(t)$ 内最优的 $|P_2'(t)|$ 个个体.
// 遗传操作
for 每一个状态为生的个体 I_i,
选择 I_i 的邻居 N_i 内的最优个体 I_i',
$\text{offspring}=\text{crossover}([I_i,I_i'],p_c)$. // 算法 PCGA
$\text{offspring}=0-\text{crossover}([I_i,I_i'],p_c)$.
// 算法 IPCGA
$\text{offspring}=\text{mutate}(\text{offspring},p_m)$,
$f_{\text{offspring}}=\text{evaluate}(\text{offspring})$,
if $\min(f_{\text{offspring}}<f(I_i))$, then
用 offspring 内的最优个体替代 I_i.
end for
untill terminated = true
end

4 实验与分析

4.1 测试问题

用 Yang[1]提出的二进制编码下动态测试问题构造方法生成动态环境. 选取最优解均为 100 的 One-Max 问题, NK(25, 4) 函数和 Deceptive 函数为静态极大化优化问题, 通过上述方法构造动态测试函数. 其中: 变化强度 s 取 0.1, 0.3, 0.5, 0.9; 变化频率 Δ 取 25, 模板 T 分别取非周期变化、周期变化和带噪声的周期变化 3 种复杂度[12].

4.2 性能指标

1) 收敛性指标 F_{acc}: 以每一环境下最终获得的最优解的均值测度算法收敛的准确性. 计算公式为

$$F_{\text{acc}}=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{N}\sum_{j=1}^{N}F_{ij}. \tag{9}$$

其中: N 为环境变化的次数, F_{ij} 为第 i 次运行第 j 个环境下最优个体适应值.

2) 多样性指标 D_E: 以种群适应值熵测度算法的多样性, 熵值越高, 表示种群分布越均匀. 计算公式为

$$D_E=\frac{1}{L}\sum_{l=1}^{L}\Big(-\sum_{k=1}^{K}p_{lk}\log(p_{lk})\Big). \tag{10}$$

其中: L 为编码长度, K 为等位基因阶数, p_{lk} 为第 k 个等位基因在第 l 个基因座上出现的概率.

4.3 结果及分析

针对 3 种不同复杂度的动态优化问题, 分别采用捕食元胞遗传算法 (PCGA), 改进的捕食元胞遗传算法 (IPCGA), 基于演化规则 I 和 II 的元胞遗传算法 (CGA-RI, CGA-RII) 以及改进的简单遗传算法 (ISGA) 进行优化. 采用二进制编码, 编码长度 $L=100$, 种群规模 $N=100$, 最大进化代数 $G=500$, 离散变异 $P_m=0.01$, 均匀交叉 $P_c=0.7$, 每种算法均计算 20 次. IPCGA 选用的正交表为 $L_8(2^7)$.

表 1～表 3 分别给出了 5 种算法对于 3 种动态测试问题优化的精确性能 F_{acc} 的数据对比, 采用平均值 ± 标准差的形式, 加粗数据表示相同复杂度下的最优值. 图 4 为 5 种算法在不同强度非周期动态环境下的 D_E 曲线对比, 受篇幅所限, 仅给出了非周期动态环境下的对比图.

1) 不同复杂度动态环境下的收敛性分析

由表 1 可知, 对于动态 One-Max 问题, IPCGA 的收敛性能最优, 在不同变化强度下的 F_{acc} 均为 100, 这说明 IPCGA 能够有效地处理动态 One-Max 问题, 且不易受噪声的干扰; PCGA 的 F_{acc} 劣于 IPCGA, 但优于其他算法; 两种基于演化规则的元胞遗传算法中, CGA-RII 的收敛性能优于 CGA-RI, ISGA 的收敛性能最差.

在非周期变化的动态环境下, PCGA 和 CGA-RII 的 F_{acc} 随强度的增加呈先下降后上升的趋势, 当 $s=$

表 1 不同复杂度动态 One-Max 问题的 F_{acc}

	s	PCGA	IPCGA	CGA-RI	CGA-RII	ISGA
非周期	0.1	99.5 ± 1.3	$\mathbf{100.0\pm0.0}$	97.8 ± 2.2	99.0 ± 1.3	83.8 ± 1.4
	0.3	94.2 ± 0.4	$\mathbf{100.0\pm0.0}$	87.3 ± 0.3	93.4 ± 0.2	68.7 ± 3.1
	0.5	93.2 ± 0.3	$\mathbf{100.0\pm0.0}$	82.0 ± 1.5	88.6 ± 1.3	63.6 ± 3.8
	0.9	94.3 ± 0.4	$\mathbf{100.0\pm0.0}$	79.8 ± 1.9	90.1 ± 1.6	59.9 ± 5.6
周期	0.1	99.7 ± 0.1	$\mathbf{100.0\pm0.0}$	97.7 ± 0.6	98.1 ± 1.2	93.3 ± 4.1
	0.3	97.1 ± 1.3	$\mathbf{100.0\pm0.0}$	90.2 ± 1.0	93.9 ± 0.4	83.8 ± 5.3
	0.5	95.0 ± 1.8	$\mathbf{100.0\pm0.0}$	87.4 ± 1.2	93.6 ± 0.4	80.4 ± 6.6
	0.9	99.7 ± 0.2	$\mathbf{100.0\pm0.0}$	97.8 ± 1.2	98.4 ± 1.2	93.5 ± 4.0
带噪声	0.1	99.7 ± 0.2	$\mathbf{100.0\pm0.0}$	97.2 ± 0.6	97.9 ± 1.1	92.6 ± 4.3
	0.3	97.0 ± 1.4	$\mathbf{100.0\pm0.0}$	89.0 ± 0.9	93.0 ± 0.3	81.0 ± 4.9
	0.5	94.7 ± 2.1	$\mathbf{100.0\pm0.0}$	86.4 ± 1.4	92.4 ± 0.3	79.9 ± 5.7
	0.9	99.6 ± 0.3	$\mathbf{100.0\pm0.0}$	97.4 ± 0.9	97.9 ± 1.0	92.6 ± 4.1

表 2　不同复杂度动态 NK(25, 4) 问题的 $F_{\rm acc}$

	s	PCGA	IPCGA	CGA-RI	CGA-RII	ISGA
非周期	0.1	89.4 ± 3.9	$\mathbf{99.8 \pm 0.3}$	70.3 ± 3.3	76.9 ± 3.0	43.8 ± 4.0
	0.3	74.6 ± 0.9	$\mathbf{96.4 \pm 0.9}$	48.7 ± 4.0	56.2 ± 4.4	27.5 ± 6.3
	0.5	73.1 ± 0.8	$\mathbf{95.8 \pm 0.9}$	41.0 ± 5.2	52.1 ± 5.2	24.2 ± 6.2
	0.9	72.5 ± 1.1	$\mathbf{97.0 \pm 0.6}$	42.8 ± 4.5	52.0 ± 4.4	29.1 ± 6.8
周期	0.1	89.7 ± 4.4	$\mathbf{99.6 \pm 0.5}$	81.6 ± 2.7	86.2 ± 1.6	61.6 ± 4.9
	0.3	76.7 ± 1.6	$\mathbf{96.4 \pm 0.8}$	59.5 ± 3.2	66.1 ± 4.0	51.5 ± 11.5
	0.5	74.0 ± 1.3	$\mathbf{95.7 \pm 0.9}$	54.8 ± 4.3	57.9 ± 5.2	48.4 ± 14.5
	0.9	89.5 ± 4.6	$\mathbf{99.5 \pm 0.5}$	84.9 ± 2.2	89.0 ± 0.8	62.2 ± 5.0
带噪声	0.1	88.8 ± 4.8	$\mathbf{99.5 \pm 0.3}$	79.6 ± 2.3	83.0 ± 1.3	61.4 ± 4.2
	0.3	75.0 ± 1.3	$\mathbf{95.7 \pm 0.7}$	57.8 ± 3.8	64.8 ± 2.8	50.0 ± 10.2
	0.5	73.9 ± 1.4	$\mathbf{95.3 \pm 1.0}$	53.1 ± 3.7	55.5 ± 6.6	47.4 ± 13.1
	0.9	88.6 ± 4.6	$\mathbf{99.3 \pm 0.4}$	82.8 ± 2.2	88.3 ± 1.5	61.9 ± 5.4

表 3　不同复杂度动态 Deceptive 问题的 $F_{\rm acc}$

	s	PCGA	IPCGA	CGA-RI	CGA-RII	ISGA
非周期	0.1	83.3 ± 0.8	$\mathbf{83.7 \pm 0.6}$	79.3 ± 1.8	75.6 ± 2.2	65.0 ± 1.6
	0.3	81.4 ± 0.6	$\mathbf{83.3 \pm 0.4}$	72.6 ± 1.3	56.7 ± 4.6	55.1 ± 4.1
	0.5	81.2 ± 0.6	$\mathbf{84.3 \pm 0.3}$	71.5 ± 1.4	47.5 ± 6.2	53.8 ± 4.0
	0.9	85.5 ± 1.1	$\mathbf{87.9 \pm 1.4}$	82.6 ± 1.8	49.1 ± 6.8	64.5 ± 1.7
周期	0.1	$\mathbf{87.0 \pm 0.7}$	85.0 ± 0.4	78.8 ± 0.4	83.7 ± 1.7	74.6 ± 2.9
	0.3	82.6 ± 0.6	$\mathbf{85.3 \pm 0.3}$	74.4 ± 1.2	66.0 ± 2.6	69.1 ± 4.8
	0.5	81.4 ± 0.5	$\mathbf{86.0 \pm 0.4}$	74.7 ± 1.1	59.4 ± 6.8	68.5 ± 5.1
	0.9	$\mathbf{87.6 \pm 0.6}$	84.3 ± 0.5	81.5 ± 0.5	83.5 ± 1.6	76.0 ± 2.4
带噪声	0.1	$\mathbf{86.2 \pm 0.5}$	83.9 ± 0.5	77.9 ± 0.6	82.9 ± 1.4	73.9 ± 2.8
	0.3	82.5 ± 0.5	$\mathbf{84.5 \pm 0.6}$	72.6 ± 1.2	69.5 ± 1.7	67.7 ± 5.7
	0.5	81.2 ± 0.7	$\mathbf{85.1 \pm 0.8}$	73.3 ± 0.9	59.1 ± 5.8	68.8 ± 4.0
	0.9	$\mathbf{84.3 \pm 0.5}$	84.1 ± 0.5	80.2 ± 0.3	82.6 ± 1.5	74.4 ± 2.3

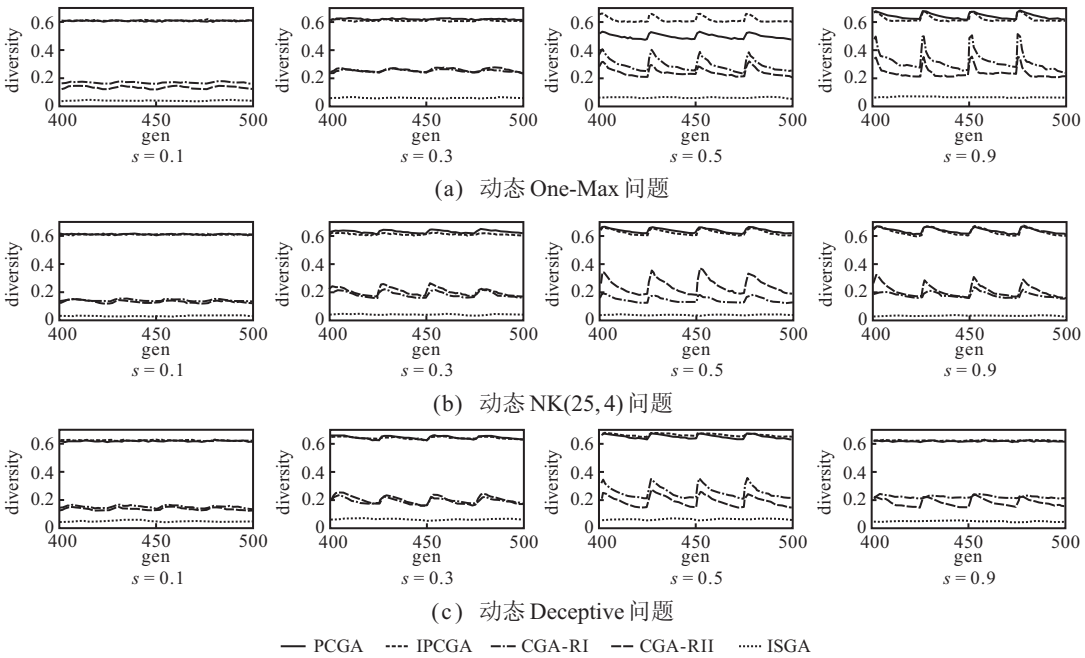


图 4　非周期动态环境下多样性曲线对比

0.5 时, $F_{\rm acc}$ 与全局最优解的相对误差最大, 分别为 6.80% 和 11.35%. CGA-RI 和 ISGA 的 $F_{\rm acc}$ 随强度的增加呈下降趋势, 当 $s = 0.9$ 时相对误差最大, 分别为 20.12% 和 40.03%. 在周期变化的动态环境下, 除 IPCGA 之外的 4 种算法的 $F_{\rm acc}$ 均随强度的增加呈先增加后减小的变化趋势, 当 $s = 0.5$ 时相对误差最大, 分别为 5.00%, 12.55%, 6.40%, 19.58%. 引入噪声后, IPCGA 算法仍能寻找到全局最优解; 其他 4 种算法均受到不同程度的影响, 当 $s = 0.5$ 时相对误差分别为 5.28%, 13.58%, 7.58%, 20.03%.

由表 2 可知, 对于动态 NK(25, 4) 问题, IPCGA 算法的收敛性能最优; PCGA, CGA-RI, CGA-RII 和

ISGA 依次劣之.

在非周期变化的动态环境下, PCGA 和 CGA-RII 的 $F_{\rm acc}$ 随强度的增加呈下降的趋势, 当 $s = 0.9$ 时, $F_{\rm acc}$ 与全局最优解的相对误差最大, 分别为 27.44% 和 47.92%. IPCGA, CGA-RI 和 ISGA 的 $F_{\rm acc}$ 随强度的增加呈先下降后上升的趋势, 当 $s = 0.5$ 时相对误差最大, 分别为 4.12%, 58.98%, 75.77%. 在周期变化的动态环境下, 各种算法的 $F_{\rm acc}$ 均随强度的增加呈先增加后减小的变化趋势, 当 $s = 0.5$ 时相对误差最大, 分别为 $25.92\%, 4.24\%, 45.14\%, 42.10\%, 51.57\%$. 引入噪声后, 各种算法均受到不同程度的影响, 当 $s = 0.5$ 时相对误差分别增大为 $26.07\%, 4.66\%, 46.98\%, 44.49\%, 52.52\%$, 其中 PCGA 和 IPCGA 受噪声的影响最小.

由表 3 可知, 对于非周期变化的动态 Deceptive 问题, IPCGA 优于 PCGA; 对于周期和带噪声周期变化的问题, 当环境微弱 ($s = 0.1$) 或剧烈 ($s = 0.9$) 变化时 PCGA 优于 IPCGA, $s = 0.3$ 或 0.5 时 IPCGA 优于 PCGA. CGA-RII 的收敛性能劣于 CGA-RI, 在某些环境下 (如非周期变化 $s = 0.5$时) 甚至劣于 ISGA.

在非周期变化的动态环境下, 5 种算法的 $F_{\rm acc}$ 均随强度的增加呈先下降后上升的趋势, 当 $s = 0.3$ 时 IPCGA 的 $F_{\rm acc}$ 与全局最优解的相对误差最大, 为 16.65%; 当 $s = 0.5$ 时 PCGA, CGA-RI, CGA-RII, ISGA 的相对误差最大, 为 $18.74\%, 28.44\%, 52.49\%, 46.20\%$. 在周期变化的动态环境下, IPCGA 的 $F_{\rm acc}$ 随强度的增加呈先上升后下降的趋势, 当 $s = 0.9$ 时相对误差最大, 为 15.61%. 其他 4 种算法的 $F_{\rm acc}$ 随强度的增加呈先下降后上升的趋势, 当 $s = 0.3$ 或 0.5 时相对误差最大, 分别为 $18.56\%, 25.54\%, 40.54\%, 31.41\%$. 引入噪声后, 相对误差的最大值分别增大为 $18.73\%, 14.86\%, 27.33\%, 40.88\%, 32.23\%$.

2) 不同复杂度动态环境下的多样性分析

由图 4 可知, 多样性曲线随着环境的变化呈周期性变化. PCGA 与 IPCGA 的多样性曲线相近, 多样性保持能力优于其他算法, 均位于 0.6 附近; ISGA 的多样性保持能力最差, 低于 0.1.

对于动态 One-Max 问题, PCGA 略优于 IPCGA; 在两次相邻的环境变化之间, 多样性曲线呈先上升后下降的变化趋势, $s = 0.1$ 时多样性曲线较为平滑, 曲线波动幅度随着 s 的增加而增加. 对于 CGA-RI 和 CGA-RII, $s = 0.1$ 时 CGA-RII 的多样性曲线低于 CGA-RI; $s = 0.3$ 时两种算法的多样性曲线相似; 当 $s = 0.5$ 或 0.9 时 CGA-RII 的多样性曲线高于 CGA-RI, 且波动幅度大于 CGA-RI.

对于动态 NK$(25, 4)$ 问题, PCGA 与 IPCGA 的多样性曲线相似, 曲线的波动幅度随 s 的增加而增加. 对于 CGA-RI 和 CGA-RII, $s = 0.1$ 时 CGA-RII 的多样性曲线要高于 CGA-RI; 曲线的波动幅度随 s 的增加呈先增加后减小的趋势, $s = 0.5$ 时曲线波动幅度最大.

对于动态 Deceptive 问题, 各种算法的多样性曲线的波动幅度相近, 随 s 的增加呈先增加后减小的趋势, 环境微弱 ($s = 0.1$) 或剧烈 ($s = 0.9$) 变化时曲线较为平滑, $s = 0.5$ 时曲线波动幅度最大.

5 结 论

根据物种间的捕食关系构建捕食策略以代替元胞遗传算法中的演化规则, 与已有演化规则相比, 捕食策略依据捕食个体捕食能力、被捕食个体逃逸能力的优劣推动种群的进化, 并通过种群控制策略维持捕食与被捕食种群间的空间布局. 针对元胞遗传算法搜索能力的不足, 引入正交交叉算子以进一步提高算法在动态环境下的寻优准确性. 通过实验分析, 与 CGA-RI 和 CGA-RII 等算法相比, PCGA 能够更有效地处理不同复杂度的动态测试问题, 且引入正交交叉算子的 IPCGA 可以在不降低种群多样性的前提下进一步提高算法动态寻优的能力.

参考文献(References)

[1] Yang Shengxiang, Yao Xin. Population-based incremental learning with associative memory for dynamic environments[C]. Proc of Congress on Evolutionary Computing. Hong Kong, 2008: 1-20.

[2] Jin Yaochu, Branke J. Evolutionary optimization in uncertain environments — A survey[J]. IEEE Trans on Evolutionary Computation, 2005, 7(3): 303-317.

[3] 王洪峰, 汪定伟, 杨圣祥. 动态环境中的进化计算[J]. 控制与决策, 2007, 22(2): 127-131.
(Wang H F, Wang D W, Yang S X. Evolutionary algorithms in dynamic environments[J]. Control and Decision, 2007, 22(2): 127-131.)

[4] Manderick B, Spiessens P. Fine-grained parallel genetic algorithms[C]. Proc of the 3rd Int Conf on Genetic Algorithms. San Mateo: Morgan Kaufmann, 1989: 428-433.

[5] Ishibuchi H, Tsukamoto N, Nojima Y. Use of local ranking in cellular genetic algorithms with two neighborhood structures[C]. Proc of 7th Int Conf on Simulated Evolution and Learning. Melbourne, 2008: 309-318.

[6] Giacobini M, Alba E, Tomassini M. Selection intensity in asynchronous cellular evolutionary algorithms[C]. Proc 2003 Int Conf on Genetic and Evolutionary Computation. Chicago, 2003: 955-966.

(下转第838页)