遗传算法的性能分析研究*

戴晓晖, 李敏强, 寇纪淞

(天津大学 系统工程研究所,天津 300072)

E-mail: mqli@tju.edu.cn

http://www.tju.edu.cn

摘要: 回顾了遗传算法的理论研究状况,介绍了 No Free Lunch 定理,描述了遗传算法的通用框架,构造了遗传算法的性能分析矩阵,并通过模拟实验分析了一系列遗传算法的性能.实验表明,这种评价算法性能的方法切实可行,可操作性好,具有一定的通用性.

关键词: 遗传算法;性能分析;概率密度函数

中图法分类号: TP18 **文献标识码**: A

遗传算法(genetic algorithms,简称 GA)的广泛应用已引起广大学者的注意,同时也引发了一系列的问题.例如,对于具体的问题,如何选择一个好的遗传算法?遗传算法的性能一定比别的算法好吗?关于这些问题的回答,关系到遗传算法的应用与发展.一方面,遗传算法的控制参数众多,基于实验的研究结论不具有普遍意义上的指导作用;另一方面,No Free Lunch 定理的出现使遗传算法作为 21 世纪关键智能计算技术的地位受到了冲击,因此,开展这方面的理论研究是一项很有实际意义的工作.

本文首先回顾了遗传算法的理论研究状况,介绍了 No Free Lunch 定理,描述了遗传算法的通用框架,构造了遗传算法的性能分析矩阵,并通过模拟实验分析了一系列遗传算法的性能.

1 No Free Lunch 定理

遗传算法是一种启发式随机搜索算法,其性能分析一直是该领域的研究重点.尽管遗传算法在实际应用中取得了巨大的成功,但相对于其鲜明的生物基础,其数学基础还不够完善[1,2],主要表现为:(1) 缺乏广泛而完整的遗传算法收敛性理论;(2) Holland 的模式定理尚不能清楚地解释遗传算法的早熟现象和欺骗问题;(3) 遗传算法的搜索效率及其时间复杂性[3-5].这些不足严重阻碍了遗传算法的应用与推广.

由此可见,遗传算法的基础理论研究至今还没有取得突破性的进展,理论与应用之间还存在着很大的差距.最近,美国 Stanford 大学的 Wolpert 和 Macready 教授提出了 No Free Lunch(简称 NFL)定理[6].它是优化领域中的一个重要理论研究成果,意义极为深远.我们将其结论概括如下:

定理 1(No Free Lunch). 假定有 A,B 两种任意(确定或随机)算法,对于所有的问题集,它们的平均性能是相同的(性能可采用多种方法度量,如最优解、收敛速率等).更精确地讲,即

* **收稿日期**: 1999-05-20; **修改日期**: 2000-02-22

基金项目: 国家自然科学基金资助项目(69574022;69974026)

作者简介: **戴晓晖**(1970—),男,广东兴宁人,博士生,讲师,主要研究领域为计算智能,并行计算;**李敏强**(1965—),男,河北无极人,教授,博士生导师,主要研究领域为决策支持系统,进化计算,人工智能;**寇纪淞**(1947—),男,上海人,博士,教授,博士生导师,主要研究领域为决策支持系统,先进管理模式.

$$\sum_f P(\vec{c}\,|f,N,A)=\sum_f P(\vec{c}\,|f,N,B).$$

其中 $\vec{c}$ 为个体适应度的概率曲线,f 为适应度函数,N 为群体大小.

关于 NFL 定理的证明,有兴趣的读者可参考文献[6]. 对于上述结论,Radcliffe 和 Surry 也有相同的结论[7].

根据 NFL 定理,算法性能的"好或坏"不仅与一定的问题有关,而且与个体适应度的概率曲线 $\vec{c}$ 有关. 显然,只要知道了 $\vec{c}$,我们就可以评价算法性能的好坏. 如何获得个体适应度的概率曲线 $\vec{c}$,则成为本文研究的重点.

2 性能分析矩阵

2.1 遗传算法的通用框架

根据遗传算法的运行过程可知,群体 $X(t)$ 在遗传算子的作用下变为群体 $X(t+1)$,因此,我们可将遗传算法看作如图 1 所示的系统.

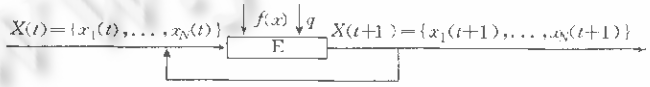

Fig. 1 Genetic algorithms

图1 遗传算法

其中 $f(x)$ 为适应度函数;q 为遗传算法的控制参数集合,如交叉概率、变异概率等;$N=|X|$ 为群体大小.

由此可见,遗传算法是一种高维非线性的随机离散动态系统. 各种遗传算法的性能差别主要是群体经系统 E 作用后个体基因的变化. 当我们比较两种算法的性能时,其本质是比较两种算法所依赖的系统状态(个体基因)矩阵,这种状态矩阵我们称为性能分析矩阵.

2.2 性能分析矩阵

评价两种算法的好坏,我们常常这样加以描述:如果算法 A 比算法 B 有"较大机会"发现"好的解",那么算法 A 比算法 B 优越. 对于这个直觉上的描述,我们必须使用数学语言来刻画. 因此,我们采用水平集来刻画"好的解",使用概率测度来评价算法发现"好的解"的可能性.

为清楚起见,本文仅讨论实数空间中的适应度函数,即 $f(x)$,$x\in R^n$. 对于其他情况,本文的分析思路仍然适用.

令 X_i^* 为当前群体中最好的解,f^* 为所求问题的全局最优解.

定义 1. 水平集 $L_f(l)$ 为适应度函数 $f(x)$ 大于阈值 l 的点的集合,即

$$L_f(l)=\{x\,|\,f(x)>l\}. \tag{1}$$

其中 $f(x)(x\in R^n)$ 为一个极大化适应度函数,且 $f(x)>0$.

在大多数情况下,我们感兴趣的是发现接近最优值 f^* 的点,即寻找处于水平集 $L_f(\tilde{f}^*)$ 中的点,其中 $f^*-\tilde{f}^*$ 很小.

由遗传算法的运行过程可知,群体中至少有一个个体处于水平集 $L_f(\tilde{f}^*)$ 中的概率随着遗传算法的运行逐步提高. 因此,我们使用概率测度来描述算法的性能分析矩阵.

定义 2. 算法性能分析矩阵为

$P(E,l,t)=$ 在 t 代群体中至少有一个个体处于水平集 $L_f(\tilde{f}^*)$ 中的概率.

注意:性能分析矩阵不仅与遗传算法的具体实施有关,而且与适应度函数有关.

如果个体 $x_i(t)$的概率密度函数 $f_{x_i(t)}$已知,那么第 i 个个体处于任意水平集 $L_f(l)$中的概率为

$$P_i(t)=P\{x_i(t)\in L_f(l)\}=\int_{L_f(l)}f_{x_i(t)}(y)\mathrm{d}y. \tag{2}$$

因此,在 t 代群体中至少有一个个体处于水平集 $L_f(l)$中的概率为

$$P(t)=1-\prod_{i=1}^{N}(1-P_i(t)). \tag{3}$$

显然,我们必须知道概率密度函数 $f_{x_i(t)}$在算法运行中的变化情况.

3 算法及其概率密度函数

3.1 算法描述

遗传算法的控制参数众多,这使得我们很难通过参数的组合来比较算法的性能,更何况这种实验研究的结论并不具有普遍意义上的指导作用.因此,我们首先从最简单的随机搜索算法入手,逐渐复杂化,这使得我们容易考察新算子对算法性能的影响.下面我们描述本文所要分析的4种算法.

3.1.1 全局搜索算法(global search,简称GS)

采用随机抽样来实施全局搜索.算法具体描述如下:

Step 1. 令 $t=0$,随机生成所对应的个体 $x_i(0)$,其概率密度函数为 $f_{x_i(0)}$;

Step 2. 随机生成所对应的个体 $r_i(t)$,其概率密度函数为 $f_{r_i(t)}$;

Step 3. 按以下方式选择下一代个体:

$$x_i(t+1)=\begin{cases}x_i(t) & \text{若 } f(x_i(t))\geqslant f(r_i(t))\\ r_i(t) & \text{若 } f(x_i(t))<f(r_i(t))\end{cases},\quad i=1,2,\ldots,N. \tag{4}$$

Step 4. 检验停止准则.若满足,则输出结果;否则,$t=t+1$,重复Step 2和Step 3.

上述算法实施了 N 次独立的随机抽样.在实际操作中,可令个体 $x_i(0)$和 $r_i(t)$服从均匀分布.

3.1.2 局部搜索算法(local search,简称LS)

采用爬山法来实施局部搜索,算法具体描述如下:

Step 1. 令 $t=0$.随机生成所对应的个体 $x_i(0)$,其概率密度函数为 $f_{x_i(0)}$.

Step 2. 按以下方式形成个体 $x'_i(t)$.

$$x'_i(t)=x_i(t)+w_i(t),\quad i=1,2,\ldots,N, \tag{5}$$

其中 $w_i(t)$为具有概率密度函数 $f_{w_i(t)}$的相互独立的随机变量.

Step 3. 按以下方式选择下一代个体:

$$x_i(t+1)=\begin{cases}x_i(t) & \text{若 } f(x_i(t))\geqslant f(x'_i(t))\\ x'_i(t) & \text{若 } f(x_i(t))<f(x'_i(t))\end{cases},\quad i=1,2,\ldots,N. \tag{6}$$

Step 4. 检验停止准则.若满足,则输出结果;否则,$t=t+1$,重复Step 2和Step 3.

上述算法实施了多点爬山算法.在实际操作中,可令个体 $x_i(0)$服从均匀分布,个体 $w_i(t)$服从对称于0的概率分布,如正态分布 $N(0,\sigma)$.

3.1.3 混合搜索算法(hybrid search,简称HS)

将上述两种算法结合起来,构成混合搜索算法.算法具体描述如下:

Step 1. 令 $t=0$,随机生成所对应的个体 $x_i(0)$,其概率密度函数为 $f_{x_i(0)}$.

Step 2. 按以下方式形成个体 $x'_i(t)$:

$$x'_i(t)=\begin{cases}x_i(t)+w_i(t) & \text{以概率 } p\\ r_i(t) & \text{以概率 } 1-p\end{cases},\quad i=1,2,\ldots,N, \tag{7}$$

其中 $w_i(t)$为具有概率密度函数 $f_{w_i(t)}$的相互独立的随机变量.

Step 3. 按以下方式选择下一代个体:

$$x_i(t+1)=\begin{cases}x_i(t) & \text{若 } f(x_i(t))\geqslant f(x'_i(t))\\ x'_i(t) & \text{若 } f(x_i(t))<f(x'_i(t))\end{cases},\quad i=1,2,\ldots,N. \tag{8}$$

Step 4. 检验停止准则. 若满足,则输出结果;否则,$t=t+1$,重复 Step 2 和 Step 3.

当 $p=0$ 时,算法变为全局搜索算法;当 $p=1$ 时,算法变为局部搜索算法;当 $0<p<1$ 时,算法具有两种算法的特点.

3.1.4 基于比例选择的混合搜索算法(hybrid search based on selector,简称 HS/S)

上述 3 种算法包含了 N 个独立的搜索过程,它们之间没有任何信息交换,这使得已有的资源(适应度评价)没有充分利用. 因此,我们引入选择算子,算法具体描述如下:

Step 1. 令 $t=0$,随机生成所对应的个体 $x_i(0)$,其概率密度函数为 $f_{x_i(0)}$.

Step 2. 按以下方式形成个体 $x'_i(t)$:

$$x'_i(t)=\begin{cases}x_i(t)+w_i(t) & \text{以概率 } p\\ r_i(t) & \text{以概率 } 1-p\end{cases},\quad i=1,2,\ldots,N, \tag{9}$$

其中 $w_i(t)$为具有概率密度函数 $f_{w_i(t)}$的相互独立的随机变量.

Step 3. 按比例选择机制选择下一代个体,即

$$P\{x_i(t+1)=x'_j(t)\mid x'_1(t),x'_2(t),\ldots,x'_N(t)\}=\frac{f(x'_j(t))}{\sum_{i=1}^{N}f(x'_i(t))},\quad i,j=1,2,\ldots,N. \tag{10}$$

Step 4. 检验停止准则. 若满足,则输出结果;否则,$t=t+1$,重复 Step 2 和 Step 3.

当 $p=1$ 时,算法变为具有比例选择算子和变异算子的遗传算法. 如果引入交叉算子,那么上述算法就相当于标准遗传算法.

3.2 概率密度函数

为分析方便,我们定义如下算子:

定义 3. $[x]$算子为

$$[x]=\begin{cases}1 & \text{若条件 } x \text{ 为真}\\ 0 & \text{若条件 } x \text{ 为假}\end{cases}. \tag{11}$$

下面我们分析上述 4 种算法个体的概率密度函数.

3.2.1 全局搜索算法(GS)

$x_i(t+1)$的概率分布可以通过相互独立的个体概率密度函数 $f_{x_i(t)}$和 $f_{r_i(t)}$计算得出,即

$$F_{x_i(t+1)}(z)=\iint_{D_z}f_{x_i(t)}(x_i(t))f_{r_i(t)}(r_i(t))\mathrm{d}x_i(t)\mathrm{d}r_i(t), \tag{12}$$

其中 $D_z=\{(x_i(t),r_i(t))\mid x_i(t+1)<z\}$.

令 $D_z=D1_z\cup D2_z$,其中

$$D1_z=\{(x_i(t),r_i(t))\mid [x_i(t)\leqslant z]\wedge[f(x_i(t))\geqslant f(r_i(t))]\}, \tag{13}$$

$$D2_z=\{(x_i(t),r_i(t))\mid [r_i(t)<z]\wedge[f(x_i(t))<f(r_i(t))]\}, \tag{14}$$

那么,式(12)可变为

$$F_{x_i(t+1)}(z)=\iint_{D1_z} f_{x_i(t)}(x_i(t))f_{r_i(t)}(r_i(t))\mathrm{d}x_i(t)\mathrm{d}r_i(t)+\iint_{D2_z} f_{x_i(t)}(x_i(t))f_{r_i(t)}(r_i(t))\mathrm{d}x_i(t)\mathrm{d}r_i(t)$$
$$=I_1(z)+I_2(z). \tag{15}$$

$$I_1(z)=\iint[x_i(t)\leqslant z]\wedge[f(x_i(t))\geqslant f(r_i(t))]f_{x_i(t)}(x_i(t))f_{r_i(t)}(r_i(t))\mathrm{d}x_i(t)\mathrm{d}r_i(t), \tag{16}$$

$$I_2(z)=\iint[r_i(t)<z]\wedge[f(x_i(t))<f(r_i(t))]f_{x_i(t)}(x_i(t))f_{r_i(t)}(r_i(t))\mathrm{d}x_i(t)\mathrm{d}r_i(t). \tag{17}$$

对式(15)求导可得

$$f_{x_i(t+1)}(z)=\frac{\mathrm{d}}{\mathrm{d}z}F_{x_i(t+1)}=\frac{\mathrm{d}}{\mathrm{d}z}I_1(z)+\frac{\mathrm{d}}{\mathrm{d}z}I_2(z). \tag{18}$$

将上式简化可得

$$f_{x_i(t+1)}(z)=f_{x_i(t)}(z)\int_{R^N}[f(z)\geqslant f(y)]f_{r_i(t)}(y)\mathrm{d}y+f_{r_i(t)}(z)\int_{R^N}[f(z)>f(y)]f_{x_i(t)}(y)\mathrm{d}y. \tag{19}$$

3.2.2 局部搜索算法(LS)

与上面的推导相类似,

$$F_{x_i(t+1)}(z)=\iint_{D_z} f_{x_i(t)}(x_i(t))f_{w_i(t)}(w_i(t))\mathrm{d}x_i(t)\mathrm{d}w_i(t), \tag{20}$$

其中 $D_z=\{(x_i(t),w_i(t))\mid x_i(t+1)<z\}$.

令 $D_z=D1_z\cup D2_z$,其中

$$D1_z=\{(x_i(t),w_i(t))\mid[x_i(t)\leqslant z]\wedge[f(x_i(t))\geqslant f(x_i(t)+w_i(t))]\}, \tag{21}$$

$$D2_z=\{(x_i(t),w_i(t))\mid[x_i(t)+w_i(t)<z]\wedge[f(x_i(t))<f(x_i(t)+w_i(t))]\}, \tag{22}$$

那么,式(20)可变为

$$F_{x_i(t+1)}(z)=\iint_{D1_z} f_{x_i(t)}(x_i(t))f_{w_i(t)}(w_i(t))\mathrm{d}x_i(t)\mathrm{d}w_i(t)+$$
$$\iint_{D2_z} f_{x_i(t)}(x_i(t))f_{w_i(t)}(w_i(t))\mathrm{d}x_i(t)\mathrm{d}w_i(t)=I_1(z)+I_2(z). \tag{23}$$

$$I_1(z)=\iint[x_i(t)\leqslant z]\wedge[f(x_i(t))\geqslant f(x_i(t)+w_i(t))]\cdot$$
$$f_{x_i(t)}(x_i(t))f_{w_i(t)}(w_i(t))\mathrm{d}x_i(t)\mathrm{d}w_i(t), \tag{24}$$

$$I_2(z)=\iint[x_i(t)+w_i(t)<z]\wedge[f(x_i(t))<f(x_i(t)+w_i(t))]\cdot$$
$$f_{x_i(t)}(x_i(t))f_{w_i(t)}(w_i(t))\mathrm{d}x_i(t)\mathrm{d}w_i(t). \tag{25}$$

对式(23)求导可得

$$f_{x_i(t+1)}(z)=\frac{\mathrm{d}}{\mathrm{d}z}F_{x_i(t+1)}=\frac{\mathrm{d}}{\mathrm{d}z}I_1(z)+\frac{\mathrm{d}}{\mathrm{d}z}I_2(z). \tag{26}$$

将上式简化可得

$$f_{x_i(t+1)}(z)=f_{x_i(t)}(z)\int_{R^N}[f(z)\geqslant f(z+y)]f_{w_i(t)}(y)\mathrm{d}y+$$
$$\int_{R^N}[f(z)>f(y)]f_{x_i(t)}(y)f_{w_i(t)}(z-y)\mathrm{d}y. \tag{27}$$

3.2.3 混合搜索算法(HS)

由算法的实施过程可知,

$$f_{x_i(t+1)}(z)=pf1_{y_i(t+1)}(z)+(1-p)f2_{z_i(t+1)}(z). \tag{28}$$

其中

$$f1_{y_i(t+1)}(z)=f_{y_i(t)}(z)\int_{R^N}[f(z)\geqslant f(z+y)]f_{w_i(t)}(y)\mathrm{d}y+$$
$$\int_{R^N}[f(y)<f(z)]f_{x_i(t)}(y)f_{w_i(t)}(z-y)\mathrm{d}y, \tag{29}$$

$$f2_{z_i(t-1)}(z)=f_{x_i(t)}(z)\int_{R^N}[f(z)\geqslant f(y)]f_{y_i(t)}(y)\mathrm{d}y+$$
$$f_{y_i(t)}(z)\int_{R^N}[f(z)>f(y)]f_{x_i(t)}(y)\mathrm{d}y. \tag{30}$$

3.2.4 基于比例选择的混合搜索算法(HS/S)

实施选择算子以后,个体的概率密度函数与群体中的其他个体有关,因此

$$f_{x_i(t+1)|x'_1(t),x'_2(t),\ldots,x'_N(t)}(z\mid y_1,y_2,\ldots,y_N)=\frac{\sum_{j=1}^{N}\delta(z-y_j)f(y_j)}{\sum_{k=1}^{N}f(y_k)}, \tag{31}$$

其中 $\delta(\cdot)$ 为 m 维的狄拉克 δ 函数.

根据总概率定理,可得

$$f_{x_i(t+1)}(z)=\int f_{x_i(t+1)|x'_1(t),x'_2(t),\ldots,x'_N(t)}(z\mid y_1,\ldots,y_N)f_{x'_1(t),\ldots,x'_N(t)}(y_1,\ldots,y_N)\mathrm{d}y_1\ldots\mathrm{d}y_N. \tag{32}$$

注意:为了计算 $f_{x_i(t+1)}(z)$,我们必须知道联合概率密度函数 $f_{x'_1(t),\ldots,x'_N(t)}(y_1,\ldots,y_N)$. 但是实施选择算子以后,个体之间不再相互独立,因此也就不可能像前面 3 种算法一样,由个体概率密度函数 $f_{x'_1(t)},f_{x'_2(t)},\ldots,f_{x'_N(t)}$ 推导出 $f_{x_i(t+1)}(z)$ 的解析表达式.

我们注意到,实施选择算子后的个体的标号可以任意分配,因此选择后的个体仍然是相互独立的,这意味着选择后群体中每个个体的概率密度函数是相同的[8],即 $f_{x_1(t)}=f_{x_2(t)}=\ldots=f_{x_N(t)}=f_{x(t)}$. 这样,我们就可以通过数值实验来估计 $f_{x(t)}$.

为了获得这个估计,我们首先通过实验来获得随机变量 $x_i(k)$ 的实验样本. 如果这个样本数足够大,那么我们就可以通过事件 $x_i(k)=z(-1\leqslant z\leqslant 1)$ 出现的次数来估计概率密度函数 $f_{x_i(t)}(z)$.

3.3 数值实验

尽管我们已经得到了前 3 种算法概率密度函数的递归解析表达式,但实际上我们并不知道 $f_{x_i(t)}$ 的具体形式,因此,我们必须借助于数值方法. 具体实施步骤如下:

Step 1. 令 $t=0$,给定个体的初始概率密度函数 $f_{x_i(0)}$(如均匀分布).

Step 2. 对方程右边实施数值积分就可获得 $f_{x_i(t)}(z)$,重复多次可得多个 $f_{x_i(t)}(z)$ 值.

Step 3. 根据 $f_{x_i(t)}(z)$ 的值,通过遗传规划回归一个近似表达式作为 $f_{x_i(t)}(z)$.

Step 4. 检验停止准则. 若满足,则输出结果;否则,$t=t+1$,重复 Step 2 和 Step 3.

在计算机实现上述步骤时,应当注意以下几点:

(1) 方程右边的积分是一个 N 维多重积分;

(2) 积分应在区间 $z_{\min}\leqslant z\leqslant z_{\max}$ 中进行;当 $z<z_{\min}$ 和 $z>z_{\max}$ 时,$f_{x_i(t)}(z)=0$. 在区间 $z_{\min}\leqslant z\leqslant$

z_{max}中,选择合适的$f_{x_i(t)}(z)$样本数,以便在近似$f_{x_i(t)}(z)$时满足一定的精度要求.

(3) 因为$f_{x_i(t+1)}(z)$的计算精度依赖于$f_{x_i(t)}$的积分精度,这使得误差代代积累,因此积分间隔必须足够小,以便满足一定的精度要求.

4 模拟实验与讨论

4.1 测试函数

选择一类傅里叶级数是光滑多峰函数的函数(如图2所示)作为测试函数的主要原因是便于我们考察遗传算法的性能.考虑如下傅里叶三角级数:

$$f(x)=a_0+\sum_{n=1}^{N}\left[a_n\cos\left(\frac{2n\pi x}{T}\right)+b_n\sin\left(\frac{2n\pi x}{T}\right)\right].$$

其中$-1\leqslant x\leqslant 1$,T为周期,本文取$T=2$.

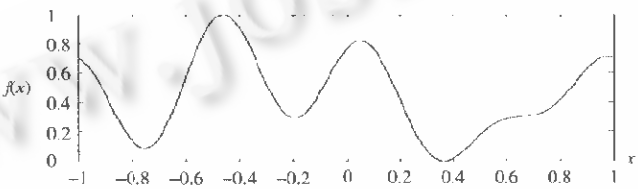

Fig.2 A smooth Fourier function

图2 一种光滑多峰傅里叶函数

随机生成整数$N>1$,然后随机生成傅里叶系数a_n和b_n,最后将函数$f(x)$定标,使其满足$0<f(x)<1$,这样就可以生成上述函数.

4.2 实验设计与结果

通过计算概率密度函数$f_{x_i(t)}(z)$就可以由式(2)和式(3)来计算算法性能分析矩阵,初始概率密度函数$f_{x_i(0)}(z)$为均值.

对于全局搜索算法(GS)、局部搜索算法(LS)和混合搜索算法(HS),计算积分时均采用辛普森(Simpson)规则,每代的积分误差不得大于0.02%,否则,调整步长,重新计算.最后一代的总误差不得大于5%,否则结果无效.由于算法开始时每个个体服从均匀分布,且在算法运行过程中个体之间没有任何影响,因此,每一代个体的概率密度函数是相同的,所以只需计算群体中任一个体的概率密度函数.

对于基于比例选择的混合搜索算法(HS/S),我们通过2 000次实验得到每个个体的2 000个样本,通过这些样本来估计个体的概率密度函数$f_{x_i}(z)$.

对于局部搜索算法(LS),令个体$w_i(t)$服从正态分布$N(0,\sigma)$,$\sigma=s=0.01,0.05,0.1,0.5,1.0$,其中$s=0.01$表示允许算法有小的扰动,$s=1.0$表示允许算法有大的跳动.

对于混合搜索算法(HS),令$p=0.25,0.5,0.75$.对于每个p值,算法以相同的s值运行,即$s=0.01,0.05,0.1,0.5,1.0$.

对于基于比例选择的混合搜索算法(HS/S),算法以群体大小10,50和100运行,而p和s的值取混合搜索算法(HS)所观察到的最好值.其他3种算法运行时的群体大小为50.

对于每个算法,迭代次数$t=50$,水平集$L_f(l)$的水平值$l=0.98$(1是最优值).实验在一台PC586/166MHz/48M的微机上进行,每次模拟实验大约需8~12个小时.对于不满足精度要求的

结果,本文没有给出.实验结果($P_i(t)\sim t$ 的关系曲线)如图 3~8 所示.

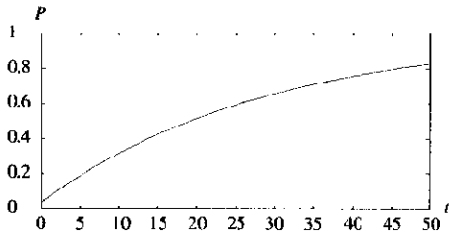

Fig. 3 Performance analysis of GS

图 3 GS 性能分析

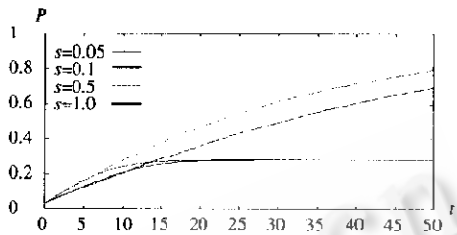

Fig. 4 Performance analysis of LS

图 4 LS 性能分析

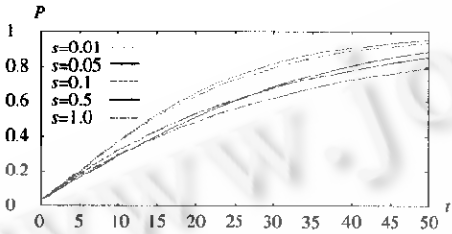

Fig. 5 Performance analysis of HS ($p=0.25$)

图 5 HS($p=0.25$)性能分析

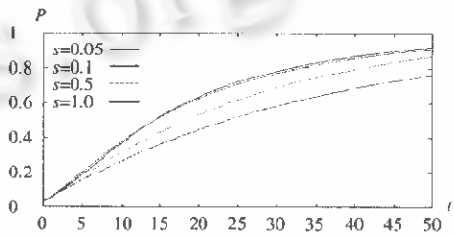

Fig. 6 Performance analysis of HS ($p=0.5$)

图 6 HS($p=0.5$)性能分析

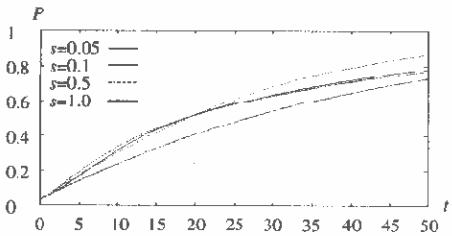

Fig. 7 Performance analysis of HS ($p=0.75$)

图 7 HS($p=0.75$)性能分析

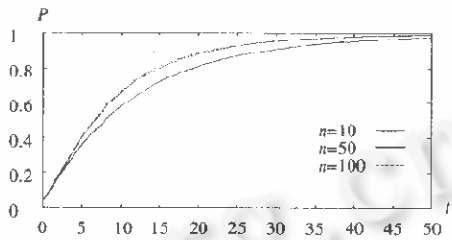

Fig. 8 Performance analysis of HS/S

图 8 HS/S 性能分析

4.3 结果分析与讨论

由实验结果可以看到,图 4~6 中的曲线相互交叉.显然,算法发现最优解的概率与代数有关,因此可以得到如下结论:算法的性能不仅与算法的控制参数有关,而且与算法的代数有关.

综上所述,我们分析了 4 种算法求解一维傅里叶函数时的性能.在实际中,应当注意:

(1) 随着求解问题维数的增加,计算量呈指数增长;

(2) 随着遗传代数的增加,误差积累也急剧增加,使得结果不可行.

尽管这些因素限制了它在实际中的应用,但性能分析矩阵仍为我们评价算法的性能提供了一个有力的工具.

5 结 论

基于目前遗传算法基础理论的研究现状以及 No Free Lunch 定理,本文将遗传算法看作是一种高维非线性的随机离散动态系统,构造了算法的性能分析矩阵,通过计算个体概率密度函数的变化情况来实现算法性能的评价.模拟实验表明,这种评价算法性能的方法切实可行,可操作性好,具

有一定的通用性,为我们选择算法提供了理论指导,解决了目前遗传算法应用中存在的一些问题.

对算法实施中存在的一些问题,如误差积累、维数过大时计算量急剧增长、计算时间过长等问题,尚需做进一步的改进和研究.

References:

[1] Xu, Zong-ben, Li, Guo. Simulated biology algorithms for global optimization, Part (I) simulated evolutionary algorithms. Chinese Journal of Operations Research, 1995,(2):1~13 (in Chinese).

[2] Fogel, D. B. An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks, 1994,5(1):3~14.

[3] Asoh, H., Muhlenbein, H. On the mean convergence time of evolutionary algorithms without selection and mutation. In: Davidor, Y., Schwefel, H., Manner, R., eds. Parallel Problem Solving from Nature, PPSN III. Berlin: Springer-Verlag, 1994. 88~97.

[4] Goldberg, D. E., Segrest, P. Finite Markov chain analysis of genetic algorithms. In: Grefenstette, J. J., ed. Proceedings of the 2nd International Conference on Genetic Algorithms. Hillsdale, NJ: Lawrence Erlbaum Associates, 1987. 1~8.

[5] Ankenbrandt, C. A. An extension to the theory of convergence and a proof the time complexity of genetic algorithms. In: Rawlins, G. J. E., ed. Foundations of Genetic Algorithms. San Francisco: Morgan Kaufmann Publishers, Inc., 1994. 53~58.

[6] Wolpert, D. H., Macready, W. G. No free lunch theorems for search. IEEE Transactions on Evolutionary Computation, 1997,1(1):67~82.

[7] Radcliffe, N. J., Surry, P. D. Fundamental limitations on search algorithms: evolutionary computing in perspective. In: Van Leeuwen, J., ed. Computer Science Today: Recent Trends and Developments. Berlin: Springer-Verlag, 1995. 274~291.

[8] David, H. A. Order Statistics. 2 ed. New York: John Wiley and Sons, Inc., 1981. 73~78.

附中文参考文献:

[1] 徐宗本,李国.解全局优化问题的仿生类算法(I)——模拟演化算法.运筹学杂志,1995,(2):1~13.

Study on the Performance Analysis of Genetic Algorithms*

DAI Xiao-hui, LI Min-qiang, KOU Ji-song

(*Institute of Systems Engineering, Tianjin University, Tianjin* 300072, *China*)

E-mail: mqli@tju.edu.cn

http://www.tju.edu.cn

Abstract: In this paper, the state of the theory of genetic algorithms is examined, and the No Free Lunch theorem is introduced. The general framework of genetic algorithms is described, in which the performance analysis matrix is built and used to test some typical genetic algorithms. The experimental results indicate that it is a feasible method, which is easy and adaptable to the evaluation of the performance of genetic algorithms.

Key words: genetic algorithms; performance analysis; probability density function

* Received May 20, 1999; accepted February 22, 2000

Supported by the National Natural Science Foundation of China under Grant Nos. 69574022, 69974026