基于动态比例变换的高效遗传算法*1)

刘立明　廖新维　陈钦雷

(石油大学试井中心　北京　102200)

HIGH PERFORMANCE ACHIEVED IN GA USING DYNAMIC LINEAR TRANSFORM

Liu Liming　Liao Xinwei　Chen Qinlei

(*Welltesting Center of Petroleum University, Beijing,* 102200)

Abstract

The dynamic linear transform (DLT) of fitness function is presented to cope with problems in genetic algorithm (GA), which always converges quickly in the early stage and very slowly in the later stage. How to choose parameters for DLT and how it works are analyzed. A hybrid algorithm called Optimal Point Searching by Interpolation is presented through analyzing the neighboring area. The later method can pick out the optimal points using the information available. Examples show that these methods work well.

Key words: Genetic Algorithm, Linear Transform, Interpolation

§0. 引　　言

遗传算法中，分别以在、离线性能表征算法的进行性能和收敛性能 [1]，两者是一对矛盾体. 在算法进行的早期，人们希望算法有较好的在线性能，以便能快速地搜索到最优点的附近. 否则便会出现算法过早收敛的情况，谓之“早熟”. 在群体演化的后期，很可能会出现下列情况：个体适应度之间的差值比群体最小适应度相差若干个数量级，此时群体演化的速度会非常慢，甚至很多代也不会达到最优点.

遗传算法的作用原理用模式理论能得到很好的解释. 根据模式理论，群体中第 j 个个体通常以概率 $p_j = f_j/\Sigma f_i$ 的概率被选择复制. 若包含于群体中的某模式 H 在当前代中有 $M(H,T)$ 个代表个体，则在下一代中此模式的代表个体的期望值将为

$$M(H,T+1) = M(H,T)f(H)/f_{av}, \tag{1}$$

即一个特定的模式按照其平均适应值与群体的平均适应值之比而生长. 因此，平均值以上的模式将逐渐增加，平均值以下的模式将逐渐消亡. 若某一特定的模式从第 0 代开始一直保持

* 2000 年 11 月 1 日收到.

1) 基金项目：石油天然气集团公司“九九”滚动项目“试井基础研究”(990507-04-03).

在平均适应值以上一个值为 cf_{av}，到 T 代此模式的代表串有 $m(H,T)=m(H,0)(1+c)^T$ 个．这表明，在平均适应值以上（以下）的模式将会以指数增长（衰退）的方式被复制．这就是高适应值个体为何迅速控制整个群体的原因，在初始个体选择不太恰当的情况时，若个体性能相差悬殊尤会出现这个问题．另一方面，一旦若干代之后，群体被几个模式所控制，c 值将会趋近 0，相对较优的模式在下一代中具有更多个体的可能性并不会比其它个体多．这时，算法离线性能便会大打折扣．

§1．动态比例函数的提出

近年来，针对算法的在、离线性能改进的方法比较多，主要有以下几个方法：组合基因算法 [1]、大变异 [2]、自适应技术 [7]、采用非标准基因操作子 [6]、小生境技术等 [3,4]．人工筛选初始群体的办法也能达到较好的效果，但这显然依赖于运用算法的人对所需求解问题的了解深度及对遗传算法的理解程度，因而从客观上限制了算法的通用性．适应度函数的性态对具体问题来说影响也是很大的．在实际运用中有时也需要对原问题函数进行一定处理，如求解最小值问题．工程中对函数进行处理一般采用下面三类函数：线形函数，指数函数，幂函数．除了线形函数外，其它两类函数都不能保持原函数的性态，且相应的系数取值不易界定，特别是当对原问题并不太了解时．但常系数的线形函数加于原函数上的强制性选择是一成不变的，不能适应问题的具体变化．本文提出动态比例函数的概念，试图解决这个问题．动态比例函数的一般形态如下：

$$y-y_b=\frac{y_u-y_b}{f_{\max}-f_{\min}}*(f-f_{\min})\ \text{或}\ y=af+b \tag{2}$$

其中 $a=\dfrac{y_u-y_b}{f_{\max}-f_{\min}}$，$b=y_b-af_{\min}$．上式中 f，f_{min}，f_{max} 分别表示第 T 代个体的原适应度值及原适应度值的最小和最大值．　y，y_u，y_b 为变换后的个体的适应值及其上、下限，其中 y_u，y_b 要求在算法开始之前事先给定．由于 y_u 与 y_b 均为常数，因此 (2) 式的变换是一个线形变换，它不影响函数的凸凹形态，但却具有将每一代形成的个体的适应值的值域进行动态放大或缩小的功能．式 (2) 对遗传算法进程的作用应从以下两个方面来考虑．

为了使函数能适应母本选择的要求，一般令 $y_u>y_b>0$．通过变换后，原适应值为 f_{max} 的个体的适应值为 y_u，原适应值为 f_{min} 的个体的适应值为 y_b．$y_b=0$ 时意味着每一代中适应值最差的个体总是被强制性地淘汰掉，这种情况是我们不愿意看到的．由于 y_u 与 y_b 均出现在 (2) 式的系数项中，　y_u 与 y_b 两个数的取值和大小对比对于整个算法的进程也会有很大的影响．不难分析出，在选择变换函数的上下限时，关注上下限之间的比值更具有实际价值．为此，再定义一个名为域差距的参数 $d=(y_u-y_b)/y_b$ 来表征变换函数值域的跨度．域差距的大小决定了遗传算法复制算子强制程度，域差距越大，强制程度越大，反之则越小．

动态比例函数的作用机理可分为两个方面．一．　$f_{max}-f_{min}>>y_u-y_b$，比例函数使群体各个体之间的适应度差距变小，增大小适应度个体在下一代中出现的几率，以保持群体的多样性．二．　$f_{max}-f_{min}<<y_u-y_b$，比例函数使群体各个体之间的适应度差距变大，使较优个体被复制的几率增大，从而使群体的平均适应度能得到更快的提高．现证明如下：

若设群体总数为 N, 第 i 个个体的适应值为 f_i, 群体的平均适应值为 f_{av}, 有

$$y_i = \frac{dy_b}{f_{\max} - f_{\min}} * (f_i - f_{\min}) + y_b, \quad i = 1 \sim N, d > 1. \tag{3}$$

个体根据 f_i, y_i 被选择进入下一代的概率分别为

$$p|_{fi} = \frac{f_i}{\sum f_j} = \frac{f_i}{N f_{av}}, \tag{4}$$

$$p|_{yi} = \frac{y_i}{\sum y_j} = \frac{f_i + \beta - f_{\min}}{\sum f_j + N(\beta - f_{\min})} = \frac{f_i + \beta - f_{\min}}{N(f_{av} + \beta - f_{\min})}, \tag{5}$$

其中 $\beta = \dfrac{f_{\max} - f_{\min}}{d}$. 变换前后个体 i 被选择进入下一代的概率变化为

$$p|_i = p|_{yi} - p|_{fi} = \frac{f_i + \beta - f_{\min}}{N(f_{av} + \beta - f_{\min})} - \frac{f_i}{N f_{av}} = \frac{[f_{av} - f_i](\beta - f_{\min})}{N f_{av}(f_{av} + \beta - f_{\min})}, \tag{6}$$

$f_{av} \geq f_{min}$. (6) 式的正负取决于分子, 分两种情况:

(1) $f_{\max} - f_{\min} > d f_{\min}$ 或 $f_{\max} > (1+d) f_{\min}$, 表明个体之间的适应度相差很大. 由于个体适应度之间的悬殊差距, 必然会导致适应度大的个体迅速控制整个群体, 减少其它个体的生存机会. 此时需要抑制大适应值个体的繁殖能力, 又要提高小适应值个体的繁殖能力. 可以看出, 对应小适应值的个体, 由于其适应值小于平均适应值, $f_{av} > f_i$, 所以 $p_i = p(y_i) - p(f_i) > 0$, 适应值上升. 对于大适应的个体, 其适应值大于平均适应值 $f_{av} < f_i$, 所以 $p_i = p(y_i) - p(f_i) < 0$, 适应值得到抑制.

(2) $f_{\max} - f_{\min} < d f_{\min}$ 或 $f_{\max} < (1+d) f_{\min}$, 此时群体的各个个体之间适应度差距较小, 这种情况通常出现在群体演化的中后期. 由于个体的适应度相差不多, 较优的个体难以 "脱颖而出", 这时需要提高那些较优个体的繁殖能力, 而抑制那些相对较弱个体的生存机会. 同样, 对于适应性能较差的个体, 由于 $f_{av} > f_i$ 所以 $p_i = p(y_i) - p(f_i) < 0$, 适应能力得到抑制. 对于较优的个体, 由于 $f_{av} < f_i$ 所以 $p_i = p(y_i) - p(f_i) > 0$, 适应能力提高. 此时通体比例函数使原群体的个体之间的适应度差距变大.

通过上述的分析可以看出, 比例函数的上、下限之间的差距极大地影响着种群的个体之间相互竞争的能力. 适应性能低下的个体, 竞争力受到的影响比高适应性个体的大. 群体遗传的初期, 为了保持群体的多样性, 要选择较小的域差距; 反之应选择大的域差距.

§2. 插值搜寻最优点

群体演化的必然结果是群体的平均适应值将逐渐升高. 欲使群体平均适应值增加, 必使个体由分散而变为集中, 因为只有在适应度函数的极值附近才能达到较高的适应值. 个体在极值点附近的集中便形成了聚居区 (图 1).

可见, 虽然遗传算法的搜索不象启发式搜索那样按部就班, 但在遗传算子看似随机作用下, 经过若干代后, 还是能找到最优个体附近的, 这是遗传算法的精髓所在. 当演化进行到一定阶段以后, 算法收敛速度变慢. 原因在于, 算法没有直接利用当前信息推理出最优信息

的能力. 针对这个问题，提出了混合遗传算法 [1,7]，以结合随机搜索与启发式搜索的优点来提高计算效率.

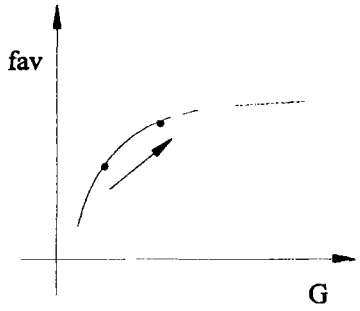

图 1(a) 平均适应值随演化而增加

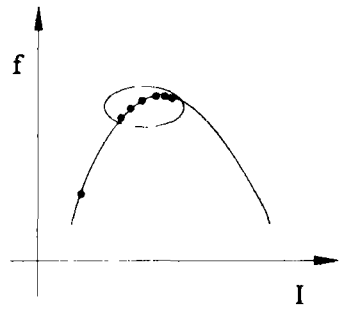

图 1(b) 个体的适应值在后期基本趋同

本文在此提出根据聚居区个体的信息，用简单插值寻求最优点的方法. 根据前面的介绍，在进行插值之前，要找到群体内个体的聚居区. 聚居区内的个体要满足以下两个条件：(1) 个体适应度值要在平均适应度值以上某个水平，也可以规定为个体适应度值达到当前最大适应值的某个水平，在已知全局最大适应值的情况下，也可以将这个下限设为此最大值的某一小数倍. (2) 个体之间的欧氏距离应在某一给定指标之内. 以上两点源于高等数学关于邻域结构的分析，不再赘述. (3) 在聚居区内个体的数目至少应为群体规模的 1/4，否则便不能成其为聚居区. 对聚居区中个体数目的限制是为了防止在算法未真正地收敛以前就开始插值搜索，从而影响演化过程的全局收敛性.

考察到聚居区形成以后，即可运用插值方法来生成聚居区的极值点了. 聚居区内的个体用插值方法寻求最优点的方法如下：从所有的个体中选择适应值最大的四个个体，令此四个个体的编号分别为 i_1, i_2, i_3, i_4，且个体对应的自变量的关系为 $x_{i1} < x_{i2} < x_{i3} < x_{i4}$，则个体之间可能具有如下三种关系：

1. 单调递增 (图 2a)，$f_{i1} < f_{i2} < f_{i3} < f_{i4}$，则插值点可选为 $x_0 = 2x_{i4} - x_{i3}$.
2. 单调递减 (图 2b)，$f_{i1} > f_{i2} > f_{i3} > f_{i4}$，则插值点可选为 $x_0 = 2x_{i1} - x_{i2}$.
3. 其它情况下 (图 2c)，则插值点可取为两直线的交点.

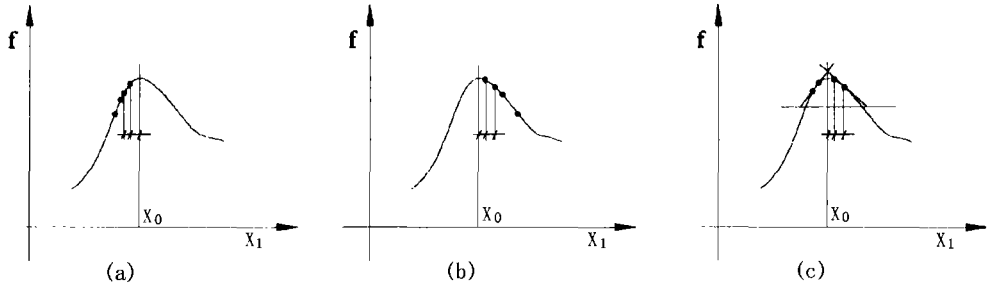

图 2 个体所对应自变量与其适应值之间的关系

找到 x_0 后，令以 x_0 为自变量的个体直接进入下一代. 这种方法实行简单，对原遗传算法的改动并不太大，而且完全不用考虑原函数的性态，很好地保持了遗传算法的独立性，因

此能够具有较广泛的适用性. 对于高维的问题, 考虑到聚居区定义的邻域结构已经很小, 对于每一维均可分别采用以上的插值方法.

§3. 算法实施步骤及算例

算法具体的实施步骤如下:

1. 随机产生当前代个体, 用动态比例函数调整各个体的适应度函数 f_1, 新的适应度函数为 f_2.

2. 根据 f_2 产生新的群体. 分两种情况:

a) 如为考察到聚居区的出现, 则所有个体均用遗传算子产生.

b) 如考察到 m 个聚居区出现, 且群体数目为 N, 则 $N-m$ 个新个体用遗传算法规则产生, m 个用插值方法产生.

3. 重复步骤 1, 2 直至达算法终止条件.

以数值实验的方法来验证本文提出的两种改进方法的有效性. 采用 Dejong 提出的两个用于检测遗传算法性能的目标函数, 并参照文 [2] 的工作, 对表 1 中的函数进行了实验.

表 1 实验采用的目标函数

函数号	表达式	定义域	优化阀值
F1	$f(x)=\sum_{i=1}^{3}x_i^2$	$[-5.12, 5.12]$	75.0
F2	$f(x)=100(x_1^2-x_2)^2+(1-x_1)^2$	$[-1.048, 2.048]$	3850

注. 当适应度函数取得优化阀值后, 我们就认为过程达到优化目标.

在实验中, 群体规模为 50, 每一分量的编码长度为 15 位. 以优化 50 次作为一次实验, 将优化到目标函数的优化阀值所需要的平均数作为衡量算法收敛速度的标准. 如算法进行到 300 代还未优化到目标函数的阀值, 则该次优化失败, 将这两种算法在 50 次中失败的总数作为衡量算法稳定性的标准. 实验结果如表 2 所示. 从实验中看到, 本文方法作了改进之后, 不但算法的收敛速度有所提高, 稳定性也有显著提高.

表 2 实验结果对比

函数号	标准遗传算法		大变异遗传算法		动态比例函数改进	
	平均演化代	未收敛次数	平均演化代	未收敛次数	平均演化代	未收敛次数
F1	30.83	4	18.48	1	13.24	0
F2	32.46	4	16.00	3	13.51	0

注. 结果参照文献 [2]

§4. 结 论

结论实验表明, 本文提出的方法确实是有助于遗传算法性能的改进, 但是在第二部分提出的简单插值方法不一定十分适用于多个自变量的函数. 对于 N 维空间问题要考虑超平面

的交点可能更为恰当，但插值函数的性态却要复杂得多．如何处理这一对矛盾是进一步研究的方向．

参 考 文 献

[1] 刘勇，康立山等，非数值并行算法 [M]，科学出版社，北京，1998.
[2] 马均水等，改进遗传算法搜索性能的大变异操作 [J]，控制理论与应用，**15**：3 (1998)，404–407.
[3] 郝翔，李人厚，适用于复杂函数优化的多群体遗传算法 [J]，控制与决策，**13**：3 (1998)，263–266.
[4] 徐金梧等，基于小生境技术的遗传算法 [J]，模式识别与人工智能，**12**：1 (1999)，104–107.
[5] 赵赫，杜端甫，遗传算法求解旅行推销员问题时算子的设计与选择 [J]，系统工程理论与实践，2 (1998)，63–65.
[6] 贺前华等，基因算法研究进展 [J]，电子学报，**26**：10 (1998)，25–31.
[7] 尹洪军等，基于自适应遗传算法的试井分析最优化方法 [J]，石油学报，**20**：2 (1999)，51–56.