一种改进的遗传退火进化算法

宁宁[1]，郭凤昌[2]
[1]电子科技大学电子工程学院，成都 (610054)
[2]电子科技大学机械电子工程学院，成都 (610054)
E-mail: NingNing@uestc.edu.cn

摘　要：本文对传统的遗传算法和模拟退火算法进行改进，同时把模拟退火算法引入了遗传算法，结合两种算法的优点，提出了一种新的遗传退火进化算法。它不但实现了遗传算法的全局搜索能力与模拟退火算法的局部搜索能力的结合，同时可使改进后的模拟退火算法能够充分利用遗传算法所得的全局信息。经验证，改算法能使遗传算法避免产生早熟收敛，增强了算法的全局收敛性，而且加快了算法的收敛速度。

关键词：遗传算法，模拟退火算法，优化

所有传统优化算法都是针对连续或可导的目标函数来说的，处理的问题也比较简单。而实际问题的参数优选常常表现出高维、多峰值、非线性、不连续、非凸性及带噪声等复杂特征。为了求解这些优化问题中的难解问题，现代优化算法孕育而生。现代优化算法主要包括禁忌搜索（Tabu Search）、遗传算法（Genetic Algorithms）、模拟退火（Simulated Annealing）和神经网络（Neural Networks）等。

遗传算法[1][2]具有良好的全局搜索能力[3][4]，可以快速地将解空间中的全体解搜索出，而不会陷入局部最优解的快速下降陷阱；并且利用它的内在并行性，可以方便地进行分布式计算，加快求解速度。但是遗传算法的局部搜索能力较差[5]，导致单纯的遗传算法比较费时，在进化后期搜索效率较低。在实际应用中，遗传算法容易产生早熟收敛的问题。采用何种选择方法既要使优良个体得以保留，又要维持群体的多样性，一直是遗传算法中较难解决的问题。

模拟退火算法[6][7]虽具有摆脱局部最优解的能力，能够以随机搜索技术从概率的意义上找出目标函数的全局最小点。但是，由于模拟退火算法对整个搜索空间的状况了解不多，不便于使搜索过程进入最有希望的搜索区域，使得模拟退火算法的运算效率不高。模拟退火算法对参数（如初始温度）的依赖性较强，且进化速度慢。

已有一些将模拟退火算法与遗传算法相结合的方法。如文献[8]中，通过模拟退火方法来控制变异概率的选取（$P_m = \exp(-\theta_m / T)$），文献**错误！未找到引用源。**中，通过模拟退火方法来控制交叉后子个体替代父个体的概率，但是这两种办法均不能很好的利用退火算法的局部搜索能力。文献[10][11]中，利用遗传算法来改变模拟退火的性能，首先通过 GA 来进化一群解，然后通过 SA 进一步调整优化解，然而对整个种群进行退火，没有很好的利用 GA 得到的全局信息。而国内的遗传退火进化算法[12][13][14]，多是使用文献[15]中的方法。先取一些初始点构成种群，对每个个体进行退火。由退火接受的新解构成新的种群，再进行选择，交叉，变异。而退火算法是局部搜索算法，所得个体有可能是局部极小值，再从中进行选择，很容易陷入局部极小值。

本文提出的遗传退火进化算法（Genetic Annealing Evolutionary Algorithm，GAEA）不同于上述方法。首先，它对 Matlab 中的遗传算法进行改进，使产生的种群更加合理。然后，将常规的模拟退火算法进行改进，根据遗传算法所得的全局信息，对其中的新解产生器和冷却进度表进行优化。最后，将模拟退火算法作为一个独立的算子，嵌入到遗传算法中，其过

程是随机产生一组比较分散的初始群体，通过选择、交叉、变异等常规遗传算子来产生一组新的个体，之后选择部分优秀个体，独立的对其进行模拟退火操作，其结果作为下一代群体，如此反复迭代进行，直到满足某个终止条件为止。

遗传退火进化算法不但实现了 GA 的全局搜索能力与 SA 的局部搜索能力的结合，同时可使改进后的 SA 能够充分利用 GA 所得的全局信息。经验证，该算法在进化初期，SA 温度较高，能使 GA 避免产生早熟收敛，增强了 GA 的全局收敛性；在进化后期，SA 温度较低，具有较强的爬山性能，加快了算法的收敛速度。

本文对 Matlab 中的遗传算法进行改进，得到的遗传退火进化算法可用于一般的最优化问题，求解无约束的或带有线性约束的连续函数的全局最小值。

1 传统算法概述

1.1 遗传算法概述

遗传算法是一种有效的解决最优化问题的方法，于 1975 年由美国 Michigan 大学 John Holland 教授根据生物进化论和遗传学的思想提出，是一种全局的启发式优化算法。从那以后，它逐渐发展成为一种通过模拟自然进化过程解决最优化问题的计算模型。起初进化的个体都是用二进制表示的，后来也用十进制表示，用十进制表示个体的遗传算法称作 CGA，以与前面的二进制 GA 相区别。它利用遗传算子（选择、交叉和变异），促进解集合类似生物种群在自然界中自然选择、优胜劣汰、不断进化，最终收敛于最优状态。

遗传算法的一般步骤为：

Stepl 对求解空间进行编码、初始化；

Step2 初始化种群 $pop(i)=(S_1,S_2,\cdots,S_N)$；

Step3 对当前种群 $pop(i)$中每个染色体 S_i 计算其适应度 $f(S_i)$，适应度表示了该个体适应性能；

Step4 应用遗传算子进行选择、交叉和变异，产生新一代群体 $pop(i+1)$；

Step5 判断终止条件。不满足返回 Step3。

GA 通过选择复制和遗传因子的作用．使优化群体不断进化，最终收敛于最优状态。选择复制使适应度函数值大的个体有较大的复制概率，它能加快算法的收敛速度。交叉算子通过对两个父代进行基因交换而搜索出更优的个体。变异操作能够给进化群体带来新的遗传基因，避免算法陷入局部极值点。

1.2 模拟退火算法概述

模拟退火（Simulated Annealing，简称 SA）算法的基本思想由 Metropolis 等于 1953 年提出，Kirkpatrick 等[6]认为固体退火过程与数学最优化过程之间存在着很好的相似性，在 1983 年成功的应用于组合最优化问题。模拟退火算法是局部搜索算法的扩展，该算法最为显著的特征是以一定的概率接受使目标函数值增大的移动，所以能够从局部最优解的“陷阱”中爬出来而不会简单地终止于一局部最优解上，即具有全局收敛性。模拟退火算法既能使系统在温度较高时逃离局部最优，又能保证系统在温度下降到一定程度而接近全局最优点的时接受次优解的可能性大大减小。

模拟退火算法的一般步骤为：

Stepl 随机给定初始状态 $T_0>0$ 和初始点，令 $k=0$，计算该点的函数值 $f(x_0)$。

Step2 若在该温度下达到内循环停止条件，则转 Step3，否则，从邻域 N(x_i)中随机选择一 x_j，计算 $\Delta f_{ij} = f(x_j) - f(x_i)$；若 $\Delta f_{ij} \le 0$，则令 $x=x_j$，否则 $\exp(\Delta f_{ij} / T_k) >$rand，（rand 为[0，1]之间均匀分布的为随机数），令 $x=x_j$；重复 Step2。

Step3 令 $T_{k+1}=d(T_k)$，$k=k+1$，若满足结束准则，停止迭代；否则回到 Step2（其中 $d(T_k)$ 为降温方式）。

整个过程包含一个内循环和一个外循环。内循环是 Step 2，它表示在同一个温度 T_k 时，在一些状态随机搜索。对于 Step 2 的处理方法主要分两类，第一类为时齐算法，对每一个固定的 T，计算对应的马尔科夫链，直至达到一个稳定状态，然后让温度下降。第二类为非时齐算法，要求在两个相邻的转移中，温度 T 是下降的。外循环是 Step 3 和停止条件。

2 改进的遗传退火进化算法

2.1 对 Matlab 遗传算法中初始种群的改进

Matlab 中遗传算法的工作流程如下：

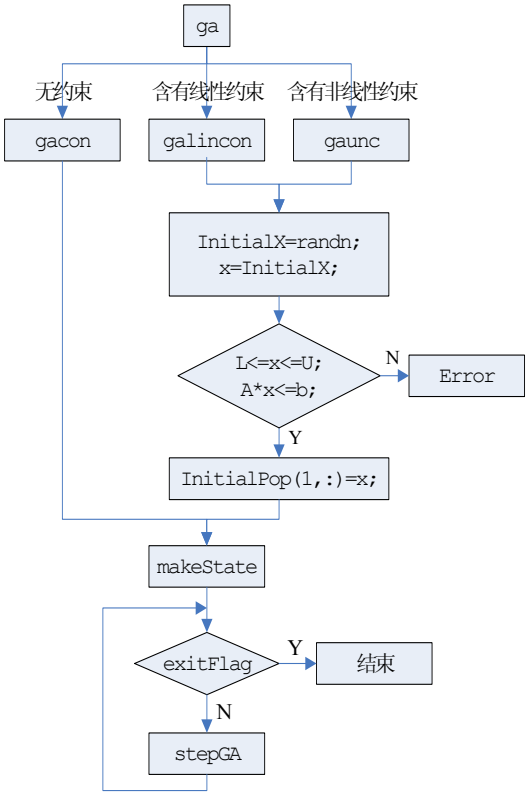


图 1 遗传算法的工作流程

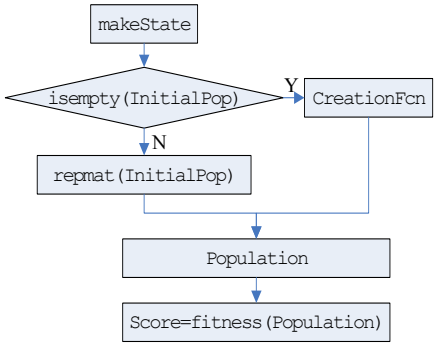


图 2 使用 makeState 函数产生初始种群

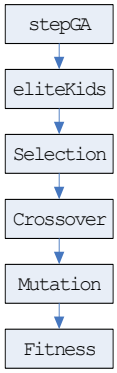


图 3 使用 stepGA 函数进行种群进化

由图 1 可以看出，产生 x 时使用 randn 函数，没有考虑可行域信息，产生的初始个体很容易落在可行域范围之外，或者对应的目标函数值非常大。

由图 2 可以看出，当有约束条件时，使用 makeState 函数产生的初始种群是前面产出的 InitialPop 个体的简单复制。这样会造成种群的单一性，使得遗传算法很容易出现早熟，且严重影响种群向最优值收敛。

为了使遗传算法初始种群中的个体分布的尽量分散，我们改写了遗传算法。将前面产生

一个正态随机数 x 作为 InitialPop 的过程去掉，然后改变 makeState 函数，直接进入 CreationFcn 函数。由于 Matlab 原带的 CreationFcn 函数无法保证产生个体满足约束条件，所以还要对 CreationFcn 函数进行改进。

在无约束的情况下，可使用 Matlab 原带的方法，代码省略。这里只考虑带有线性约束的情况。设自变量 x 须满足下列约束：

$$LB \le x \le UB; \quad A \cdot x \le b \tag{1}$$

根据上下界条件 $LB \le x \le UB$ 产生个体 x，放在 Pop0 中。

$$x = LB + (UB - LB) \cdot rand; \tag{2}$$

其中，rand 为[0,1]之间均匀分布的随机数。

如果有不等式约束时，还需要判断 $A \cdot x \le b$。步骤如下：

Stepl 将满足 $A \cdot x \le b$ 的有效个体放入种群 Pop2 中，并计算 Pop2 中个体的数目 *nPop2* 及有效个体占整个种群的比率 p。

Step2 使用上面的方法产生 n 个个体，其中，n 根据下面的式(3)进行计算。对新产生的个体判断不等式约束，将满足约束的有效个体继续放入 Pop2 中。

$$n = round[(\frac{1}{p} + 1)(popSize - nPop2)] \tag{3}$$

其中，round 为四舍五入取整。

Step3 计算 Pop2 中个体的数目 *nPop2*，若 *nPop2>popSize*，停止迭代；否则回到 Step2。

改进后的 CreationFcn 可使初始种群尽量分散，且个体产生过程使用了可行域信息，产生的初始种群更加合理。

2.2 对模拟退火算法的改进

现有的模拟退火算法大多是用于组合优化问题[6][7][12][13]，且没有考虑约束问题。这里将模拟退火算法引入到带线性约束的连续函数的优化问题。

2.2.1 新解产生器

邻域结构和新解产生器，接受准则和随机数产生器，冷却进度表被称为模拟退火算法的三大支柱，将模拟退火算法引入连续函数的优化问题关键是邻域结构和新解产生器的构造。因此问题的难点在于解 X 的邻域中新解 X' 的产生。我们使用下面的方法：

$$\begin{aligned} X' &= X + dx \\ dx &= D \cdot r(1, \text{GenomeLength}) \end{aligned} \tag{4}$$

其中，D 为跳变距离，GenomeLength 为自变量的个数， r(1,GenomeLength)表示行数为 1，列数为 GenomeLength 的随机数。随机数可使用 randn（均值为 0，方差为 1 的正态分布随机数）,或者 2*rand-1（rand 为[0,1]之间的均匀分布随机数）。

对于其中的跳变距离 D，我们使用以下几种方法产生：

方法 1：使用种群的全局信息

$$L = \min(Pop); U = \max(Pop); \tag{5}$$

$$D = \min(x - L, U - x); \tag{6}$$

其中，*Pop* 为种群个体。L，U 里为种群个体的最小值和最大值。

这种方法的优点是使用了进化过程中种群的信息。缺点是当种群出现单一化时，跳变距

离为 0。但由于前面对初始种群的优化，种群个体较分散，全部落入局部极小值的概率较小。

方法 2：使用可行域的上下界信息

$$D = \beta^k \cdot \min(x - LB, UB - x) \tag{7}$$

其中，*LB*，*UB* 里为可行域的上下界，β 为缩水系数。

这种方法的优点是不易落入局部极小值。缺点是没有使用进化过程中的种群信息，且 shrink 因子选择比较困难。

方法 3：综合种群全局信息和上下界信息

$$L_3 = \frac{L + \beta^k \cdot LB}{1 + \beta^k}; U_3 = \frac{U + \beta^k \cdot UB}{1 + \beta^k} \tag{8}$$

$$D = \min(x - L_3, U_3 - x) \tag{9}$$

这种方法为方法 1 和方法 2 的综合，从而避免了上述两种方法的缺点。在 GAEA 算法中，我们使用第 3 种方法。

产生新解 X'后，需判断不等式约束 $A \cdot X' \le b$。若不满足约束，则产生新的 X',若连续的不满足约束的 X'超过一定数量，使方差减半；若满足约束，则以 Metropolis 接受准则接受新解。Step2 中我们选择时齐算法，使得每次进化后能有多步局部搜索。当接受新解的个数=R 或迭代次数=L 时，跳出循环。

2.2.2 冷却进度表

一个冷却进度表应当规定下述参数：

1. 控制参数 T_k 的初温 T_0；
2. 控制参数 T_k 的下降函数 $T_{k+1}=d(T_k)$；
3. 马尔科夫链的长度 L_k。

下面我们对其分别讨论。

1). 关于控制参数 T_k 的初温 T_0

从理论上来说，初温 T_0 应保证平稳分布中每一状态的概率相等，也就是说：

$$p_{ij} = \exp(-\frac{\Delta f_{ij}}{T_0}) \approx 1 \tag{10}$$

其中 $\Delta f_{ij} = f(j) - f(i)$，很容易估计一个值为：

$$T_0 = K \cdot \Delta_0 \tag{11}$$

其中 *K* 为充分大的数，$\Delta_0 = \max\{f(i) \mid i \in D\} - \min\{f(i) \mid i \in D\}$。

这里的 *D* 为最优解可能存在的区域，即可行域。这时的 Δ_0 很难得到。考虑到由遗传算法得到的全局信息，将最优解所在的范围缩小到遗传算法产生的新种群所包围的范围之内，从而可得：

$$\Delta_k = \max(score) - \min(score) \tag{12}$$

其中，score 为第 *k* 代种群个体的目标值。

这里，我们取 $p_{ij} = 1 - 1/nPopAnn$，其中 *nPopAnn* 为退火产生新解的数量。即退火产生的新解中至少有一个与原解满足：

$$f(X') - f(X) \le \Delta_0 \tag{13}$$

这时，

$$T_{0k} = -\frac{\Delta_k}{\ln(p_{ij})} = \frac{\Delta_k}{\ln(nPopAnn) - \ln(nPopAnn - 1)} \tag{14}$$

由于每次退火的个体均不相同，不能使用退火算法中针对一个粒子的 T_0，所以每代个体退火的初温由式(14)计算。

2). 关于控制参数 T_k 的下降函数 $T_{k+1}=d(T_k)$

对于下降函数 $T_{k+1}=d(T_k)$，一般使用比较简单的 $T_{k+1} = \alpha \cdot T_k$（$\alpha$ 一般取 0.9，）或 $T_{k+1} = T_0 \cdot (K-k)/K$。这里我们使用第一种方法。因此可得每代个体的退火温度：

$$T_k = \alpha^{k-1} \cdot T_{0k} \tag{15}$$

这时，接受概率变为：

$$p_{ij} = \exp(-\frac{f(x+dx) - f(x)}{\alpha^{k-1} \cdot T_{0k}}) \tag{16}$$

其中，$dx = D \cdot r = \beta^{k-1} \cdot A \cdot r$，$\beta$ 为缩水系数，A 为 min(x-LB,UB-x)，r 为随机数。

考虑当进化到后期的时候，k 非常大，$dx = \beta^{k-1} Ar$ 越来越小，这时可认为 $f(x+dx) - f(x) = f'(x) \cdot dx$，这时的接受概率变为：

$$\begin{aligned} p_{ij} &= \exp[-\frac{f'(x) \cdot \beta^{k-1} \cdot A \cdot r}{\alpha^{k-1} \cdot T_{0k}}] \\ &= \exp[\ln(1 - \frac{1}{nPopAnn})(\frac{\beta}{\alpha})^{k-1} \cdot A \cdot r \frac{f'(x)}{(f_{\max} - f_{\min})}] \end{aligned} \tag{17}$$

取 $p_0 = 1 - 1/nPopAnn$，$a = \beta/\alpha$，则：

$$p_{ij} = p_0^{a^{k-1} A \cdot r \frac{f'(x)}{(f_{\max} - f_{\min})}} \tag{18}$$

对式(18)，分三种情况讨论。

① 当 $a = \beta/\alpha > 1$ 时，即缩水系数大于温度下降系数时：

$$\lim_{k \to \infty} a^{k-1} = \infty \tag{19}$$

这时 $p_{ij} \to 0$，进化后期接受概率趋向于 0。

② 当 $a = \beta/\alpha < 1$ 时，即缩水系数小于温度下降系数时：

$$\lim_{k \to \infty} a^{k-1} = 0 \tag{20}$$

这时 $p_{ij} \to 1$，进化后期接受概率趋向于 1。

③ 当 $a = \beta/\alpha = 1$ 时，即缩水系数等于温度下降系数时：

$$p_{ij} = p_0^{A \cdot r \frac{f'(x)}{(f_{\max} - f_{\min})}} \tag{21}$$

即接受概率与 x 点的导数有关，当 $f'(x) \to 0$ 时，$p_{ij} \to 1$。

3). 关于马尔科夫链的长度 L_k

对于 Markov 链的长度 L_k，一般使用 $L_k = \gamma \cdot n$，其中 n 为自变量维数。这里的 γ 一般取 100。

传统退火算法中 T_0 应充分大，才能保证退火完成时，晶体达到平衡态，即全局最优解。而满足这样的要求，会导致相当长的运行时间。这里由于使用遗传算法全局搜索后得到的优秀个体进行部分退火，温度的选择不必很高，算法对温度的依赖性不大。

2.3 将退火算子加入到遗传算法中

这里我们使用先由遗传算法进行全局搜索，得到新的种群后，选择最优秀的几个个体进行退火，在它们各自的邻域分别进行局部搜索，而距全局最优解较远的个体不进行局部搜索。这样可以避免一些不必要的搜索，节省大量的计算时间。

改进后的 GAEA 工作流程如图 4 所示。

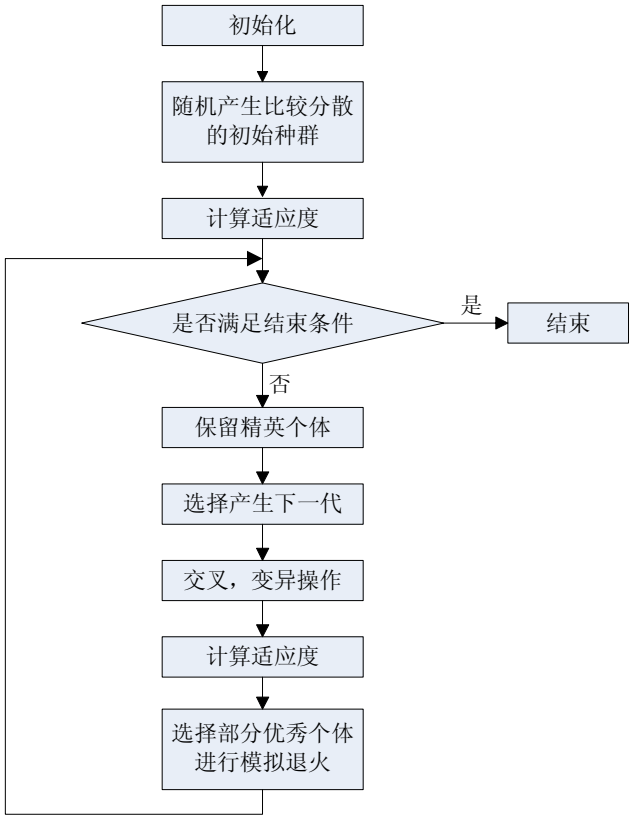


图 4 遗传退火进化算法工作流程

传统遗传算法容易产生早熟收敛的问题，即在进化群体中少数个体的适应函数值远优于其他个体，这样经过少数几次迭代后，这些个体就占据了整个种群，进化过程就提前收敛了。这里使用对优秀个体进行退火的方法，使优秀个体在各自的邻域产生一些相近的新解，可以避免极少数优秀个体占据种群，从而能够防止算法进入早熟收敛。

对于传统的遗传算法，竞争是在子代中进行的，而子代和父代之间没有竞争，这样父代中的优良个体有可能丢失。一些算法通过直接将群体中的最优解放入下一代群体中来保存最优解，但这有可能引起早熟收敛的问题。此外，遗传算法中新解的产生依赖于其中的交叉和变异，而交叉过程只是对父代信息的复制，只有变异过程可以引入新的信息。且交叉和变异过程都没有计算所得新解的适应度，无法保证子代比父代更优秀。而退火算法中引入的新解以一定概率优于原解，因而能够大大加速种群的进化。

退火算子的过程：

1.对遗传算法产生的新种群进行排序后，选择前 *nAnneal* 个个体进行退火。

$$nAnneal = \text{round}(popSize / 2R) \tag{22}$$

其中 round 表示四舍五入方法取整，R 为前面提到的内循环中接受新解的个数上限。

2.对每个个体进行分别退火。这样，退火得到的新解的总数为

$$nPopAnn = \sum_{i=1}^{nAnneal} r(i) \tag{23}$$

其中 $r(i)$ 为第 i 个个体在 L 次迭代之内接受的新解的个数。$r(i) \le R$，因此 $nPopAnn \le popSize/2$。这里取 *popSize*/2 是为了防止退火产生个体占领整个种群，从而陷入局部极小值。

3.将产生的新解放入原种群，排序后取前 *popSize* 个进入下一次进化。

3 对比结果

3.1 使用常用测试函数进行对比

在下面的实验中，我们使用两个经典的 Schaffer 函数来进行测试，将遗传算法（GA），传统的混合遗传退火算法（MGASA），遗传退火进化算法（GAEA）三种算法进行比较。两个 Schaffer 函数分别为：

$$\begin{cases} f(x_1,x_2) = (x_1^2 + x_2^2)^{0.25}[\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1] \\ -100 \le x_i \le 100 \quad (i = 1,2) \end{cases} \tag{24}$$

$$\begin{cases} f(x_1,x_2) = 0.5 + \dfrac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{[1.0 + 0.001(x_1^2 + x_2^2)]^2} \\ -100 \le x_i \le 100 \quad (i = 1,2) \end{cases} \tag{25}$$

两个函数在其定义域内都只有一个全局最小点 $f(0,0)=0$。

函数图形如下：

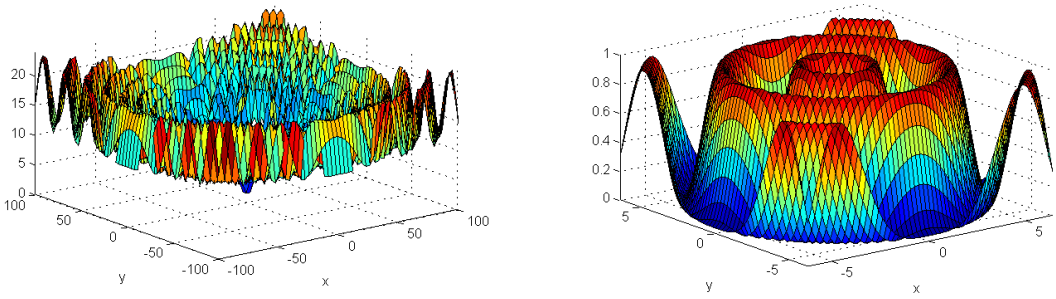


图 5 Schaffer 函数立体图

3.1.1 测试函数一

当精度设置为 1e-6 时，将每个算法独立地运行 50 次，求解结果见表 1。

表 1 三种算法运算结果比较

各种算法	结果比较			
	平均时间(s)	平均值	最差解	最好解
GAEA	0.653	7.9e-6	1.5e-5	2.8e-6
MGASA	0.989	0.042	0.087	0.0059
GA	0.687	0.246	0.549	0.0869

当精度设置为 1e-15 时，将每个算法独立地运行 50 次，求解结果见表 2。

表 2 三种算法运算结果比较

各种算法	结果比较				
	平均时间(s)	平均值	最差解	最好解	最好解个数
GAEA	4.38	2.5e-12	3.8e-11	0	8
MGASA	80.62	2.1e-3	5.4e-3	2.6e-5	1
GA	2.02	0.033	0.057	0.015	4

从对比结果可以看出，当精度要求较低时，GAEA 算法和 GA 算法的运算时间相当，结果比 GA 的精度高；当精度要求较高时，GAEA 算法运算时间略高于 GA 算法，而结果远远优于 GA 算法。

3.1.2 测试函数二

精度要求设置为 1e-20，将每个算法独立地运行 50 次，求解结果见表 3。

表 3 三种算法运算结果比较

各种算法	结果比较				
	平均时间(s)	平均值	最差解	最好解	最好解个数
GAEA	3.56	5e-16	5e-15	0	10
MGASA	5.98	0.0025	0.0097	1e-15	6
GA	3.09	0.0094	0.0372	1e-8	2

运算结果中发现，GA 有 45 次陷入局部极值，得到结果 0.0097159；MGASA 有 13 次陷入局部极值，得到结果 0.0097159；而 GAEA 所得结果均在 5e-15 以下。

GA 算法和 SA 算法都是以一定概率接近全局最优解，而从对比结果可以看出，GAEA 算法接近全局最优解的概率较大，陷入局部极值的概率很小。

3.2 使用软件可靠性模型中的参数估计问题进行对比

我们将两种算法用于软件可靠性模型中的参数估计问题来进行对比。由于遗传算法工具箱用于求最小值，我们在似然函数前加一负号作为目标函数。

这里我们使用的软件可靠性模型为 Yamada Rayleigh 模型。它的均值函数为：

$$m(t) = a(1-\exp(-\alpha(1-\exp(-\beta t)))) \tag{26}$$

第 i 个测试间隔内发现 n_i 个失效的概率为：

$$f(n_i) = \frac{\left[m(s_i)-m(s_{i-1})\right]^{n_i}\exp\left\{-\left[m(s_i)-m(s_{i-1})\right]\right\}}{n_i!} \tag{27}$$

从而得到似然函数为：

$$\ln L(n_1,n_2,\cdots,n_k)=\sum_{i=1}^{k}\left\{n_i\ln\left[m(s_i)-m(s_{i-1})\right]-\ln\left[n_i!\right]\right\}-m(s_k) \tag{28}$$

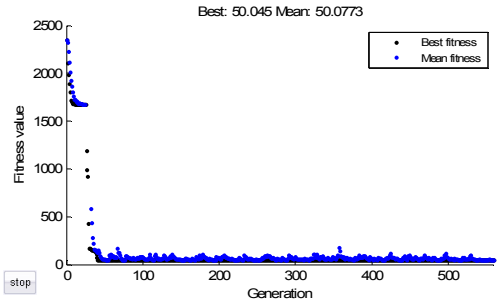
测试数据我们使用 Wood 提供的每周内发生的失效次数统计数据[16]。

由于均值函数中出现了 $\exp(\exp(-\beta\cdot t^2/2))$ 的形式，当 β 的选择距全局最值较远时，似然函数的值很容易超过 Matlab 的精度限制而出现 Inf 解。所以计算困难很大。

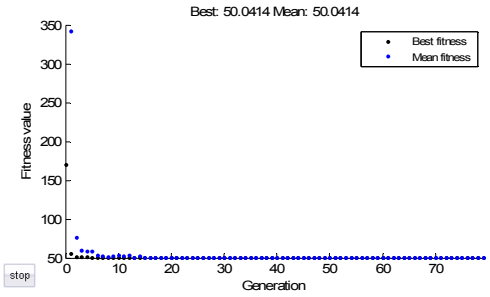
我们用作图的方法将最优解的取值范围缩小到[100,1e-10,1e-30]~[250,1,10]之间。设置上下界，种群设置为 100，精度设置为 1e-15，使用遗传算法和遗传退火进化算法分别求解。

使用遗传算法，计算时间 16.15s，进化到第 550 代得到最优解，最小目标值为 50.045。使用遗传退火进化算法，计算时间 2.46s，进化到第 80 代时得到最优解，最小目标值为 50.0414。

以下是两种算法分别得到的目标值进化过程。其中黑色点代表每一代种群中最小的目标值，蓝色点代表每一代种群目标值的平均值。可以看出遗传算法种群收敛较慢。

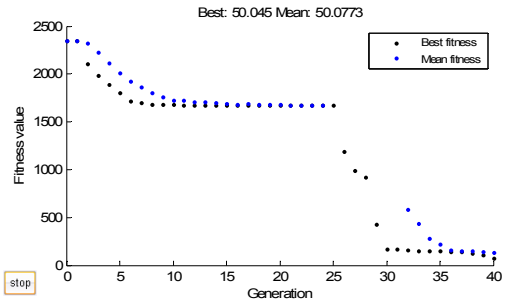


GA 进化过程的目标值图

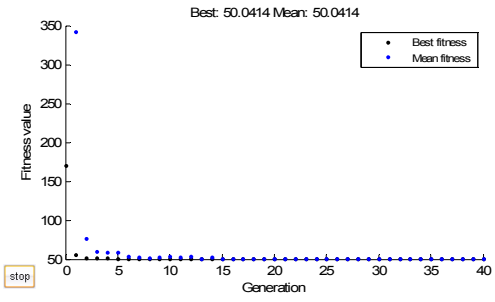


GAEA 进化过程的目标值图

取前 40 代进化结果进行对比。可以看出 Matlab 遗传算法工具箱产生的种群在进化初期出现单一化，进化速度缓慢。而改进后的遗传退火进化算法的初始种群比较分散，进化速度快。



GA 前 40 代进化过程的目标值图



GAEA 前 40 代进化过程的目标值图

考虑到两种算法都是以概率方式搜索，为了客观地评价算法的性能，将每个算法独立地运行 50 次，精度设置为 1e-15，求解结果见表 4。其中平均值是将 Inf 解去掉后计算而得。

表 4 两种算法运算结果比较

各种算法	结果比较				
	平均时间(s)	平均值	Inf 解个数	最好解	最好解次数
GAEA	2.04	50.0414107	0	50.0414034015	44
GA	9.47	50.0476542	12	50.0419728865	1

可以看出，当函数非常复杂时，用 GAEA 计算速度较快，且不会陷入局部极小值，运算结果精度高。

4 结论

遗传退火进化算法本身具有很多优点，如隐含的计算并行性、数值计算的稳定性、全局搜索与局部快速收敛搜索。经改进后的遗传退火进化算法不但实现了 GA 的全局搜索能力与 SA 的局部搜索能力的结合，同时可使改进后的 SA 能够充分利用 GA 所得的全局信息。经验证，该算法在进化初期，SA 温度较高，能使 GA 避免产生早熟收敛，增强了 GA 的全局收敛性；在进化后期，SA 温度较低，具有较强的爬山性能，加快了算法的收敛速度。

参考文献

[1] J. H. Holland, Adaptation in Natural and Artificial Systems. Cambridge, MA: MIT Press, 1992.
[2] Goldberg D E, Genetic algorithms in search, optimization and machine learning. MA: Addison, Wesley, 1989
[3] Man. K.F., Tang. K. S. and Kwong. S.: Genetic Algorithms: Concepts and Applications. IEEE Trans. Industrial Electronics. Vol. 43. (1996) 519-533
[4] Vasconcelos. J. A.，Ramirez. J. A., Takahashi. R. H. C. and Saldanha. R. R.: Improvements in Genetic Algorithms. IEEE Trans. Magnetics. Vol. 37. (2001) 3414-3417
[5] Prügel-Bennett. A.: When a Genetic Algorithm Outperforms Hill-Climbing. Theoretical Computer Science. Vol. 320. (2004) 135–153
[6] Kirkpatrick St, Gelatt C D, Vecchi M P, Optimization by simulated annealing. Science, 1983, 220. 671～680
[7] Anton Dekkers, et al., Global optimization and simulated annealing, Math. Programming, 50(1991), pp.367-393.
[8] Sirag D, Weisser P. Toward a unified thermo dynamic genetic operator Proceedings of the 2nd international conference on genetic a1gorithms. 1987. 116～122
[9] Z. G. Wang, Y. S. Wong, M. Rahman. Development of a parallel optimization method based on genetic simulated annealing algorithm. Parallel Computing, 2005, 31(8-9): 839-857
[10] Lin F T, Kao C Y, HsuC C. Applying the genetic approach to simulated annealing in solving some NP hard, IEEE Trans on SMC. 1993, 23(6): l752～1767
[11] Wang Jin, Xu Li, Zheng Baoyu. A Genetic Annealing Hybrid Algorithm based Clustering Strategy in Mobile Ad hoc Network.
[12] 周开俊,李东波. 基于遗传模拟退火算法的产品装配序列规划方法 [J]. 计算机集成制造系统. 2006 年 07 期
[13] 阎庆, 鲍远律. 新型遗传模拟退火算法求解物流配送路径问题[J]. 计算机应用, 2004, 24(3), 261～263
[14] 陈学松,曹炬,方仍存.遗传模拟退火算法在矩形优化排样系统中的应用[J].锻压技术,2004,(1):27-29.
[15] 邢文训, 谢金星. 现代优化计算方法[M]. 北京:清华大学出版社, 1999. 181 一 182.
[16] A. Wood, Predicting software reliability. Computer. 1996, 29(11): 69-77

An Improved Genetic Annealing Evolutionary Algorithm

Ning Ning，Guo Suchang
University of Electronic Science and Technology of China，Chengdu （610054）

Abstract
This paper improves the traditional Genetic Algorithm (GA) and Simulation Annealing Algorithm (SA), and then introduces SA into GA, proposing a new Genetic Annealing Evolutionary Algorithm. It combines GA global search ability and SA local search ability. At the same time the improved SA can take full advantage of the global information obtained from GA. The result shows that, the new algorithm can not only avoid GA from premature convergence, enhance the global convergence, but also speed up the convergence rate.
Keywords: Genetic Algorithm，Simulation Annealing Algorithm，Optimization