GAlib 在遗传算法实现中的应用①

张泰忠，徐　成

(湖南大学 信息科学与工程学院，长沙 410082)

摘　要：遗传算法（Genetic Algorithm）是一种模拟达尔文生物进化论的随机搜索方法，被人们广泛应用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。然而，编程实现遗传算法的过程非常复杂且容易引入人为的错误，进而影响相关研究的准确性。针对这一问题，讨论了基于 GAlib（Genetic Algorithm Library）类库设计实现遗传算法的关键技术。最后，利用 GAlib 实现了遗传算法，并以解决旅行商人问题(TSP)对算法进行评估，实际效果表明，基于 GAlib 开发的遗传算法编程简单，可读性强，结果准确。

关键词：GAlib；遗传算法；染色体；种群；适应度

Application of GAlib to the Implementation of Genetic Algorithm

ZHANG Tai-Zhong, XU Cheng

(College of Information Science and Engineering, Hunan University, Changsha 410082, China)

Abstract: Genetic Algorithm (Genetic Algorithm) is a random investigation method simulating Darwin's biological evolution, and it is applied to the fields like combinatorial optimization, machine learning, signal processing, adaptive control and artificial life. However, the implementation process of genetic algorithm is very complicated and easily introduce error artificially, and influence the accuracy of research. for this problem, the key technology of implementing genetic algorithm based on GAlib（Genetic Algorithm Library）class library is discussed. Finally, genetic algorithm is implemented using GAlib and evaluate the algorithm by solving the traveling salesman problem(TSP), the results shows that the genetic algorithm implemented based on GAlib is easily programming, highly readable,and results are accurate.

Key words: GAlib; genetic algorithm; chromosome; population; fitness

1　引言

1975 年 J.Holland 教授提出了遗传算法[1]（Genetic Algorithm）思想，该算法是一种借鉴达尔文生物进化论“适者生存，优胜劣汰”遗传机制的随机搜索方法，具有内在并行性良好，全局寻优能力强，搜索方式智能等特点，广泛应用于函数优化、机器人学、图像处理、数据挖掘和人工生命等领域，已逐渐成为现代人工智能计算中的关键技术。然而，遗传算法在运用过程中需要经历初始化种群、染色体编码、选择、交叉、变异、适应度计算等一系列复杂操作，实现过程十分繁琐，造成人为错误的可能性较大。

GAlib[2]（Genetic Algorithm Library）是美国麻省理工学院的 Matthew Wall 用 C++开发的一套遗传算法类库，设计合理，功能强大且易于扩展。本文针对利用 GAlib 类库实现遗传算法需要解决的主要问题进行讨论，组织结构如下：第 2 节介绍 GAlib 的相关概念；第 3 节以解决旅行商人问题为实例，探讨基于 GAlib 实现遗传算法的关键技术；第 4 节统计旅行商人问题的实验结果并分析利用 GAlib 实现的遗传算法的实际效果，最后对本文的工作进行总结。

2　相关概念

一个完整的遗传算法包括染色体编码、初始化种群，遗传操作以及适应度计算，针对遗传算法的主要因素，GAlib 设计实现了遗传算法类、染色体类、种群类、选择方案类及各自的派生体系。

① 基金项目:国家自然科学基金(60973030)
收稿时间:2010-03-03;收到修改稿时间:2010-03-24

2.1 遗传算法类

GAlib 中的遗传算法类(GAGeneticAlgorthm)是一个抽象类，给出了遗传算法通用的算子和数据结构，包含了变异概率，交叉概率，进化代数以及群体相关参数的统计信息，同时定义了最大或最小化目标函数、控制显示频率及设置相关参数的操作。算法内置的进化停止准则是最大进化代数准则和最佳个体收敛准则。

根据群体更新机制的不同，GAlib 提供的遗传算法可分为四种。一是标准遗传算法(GASimpleGA)，该算法仅保留父代最佳个体或不保留父代群体中任何个体，子代群体完全由交叉变异等遗传操作产生；二是稳态遗传算法（GASteadyStateGA），子代群体由进化操作产生的部分新个体和父代群体中部分个体按一定比例共同组成；其三是增量遗传算法（GAIncrementalGA），每代进化操作只产生一个和两个新个体，新个体替换其父代群体中的某些个体；四是种群遗传算法(GADemeGA)，该算法同时控制数个群体进行演化，每个群体的进化采用稳态遗传算法，群体间进行交换个体操作。

2.2 染色体类

遗传算法不能直接处理问题空间的参数，当使用遗传算法解决优化问题时，必须把相关参数转换成遗传空间的“染色体”，染色体在本质上是一种表示问题解的数据结构。染色体类(GAGenome)是一个抽象基类，定义了一些常量和函数原型、具体的染色体及其派生类的函数原型。GAlib 为其提供了四个派生类：链表型染色体（GAListGenome）、树型染色体(GATreeGenome)、数组型染色体(GAArrayGenome)和二进制串型染色体(GABinaryString)。

在实际应用中可根据问题特点选择染色体类型。遗传算法在运行过程中对染色体的操作主要有五种：初始化、变异、交叉、评估和比较操作，初始化操作随机确定群体中已存在个体的参数，变异和交叉操作随染色体种类的不同而不同。例如，链表型染色体的变异操作有次序交换、节点交换、节点删除和节点增加；交叉操作为单点交叉。

评估操作通过指向由用户定义的目标函数的指针来调用目标函数，用以评估基因组是否比其它同类更适宜生存，比较操作用于确定进化个体之间的差别。

2.3 进化种群类

进化种群类(GAPopulation)是染色体的容器，包含种群内染色体目标分数的平均值、最大值、最小值等统计数据，定义了初始化算子和评估算子。此外进化种群类实现了选择方法(GASelectionScheme)以在进化中决定使用群体中的哪个染色体参与产生子代的交叉操作，而适应度定标方法(GAScalingScheme)用来决定染色体的适应度。

2.4 适应度定标方法类

遗传算法进行选择操作时，通常并不直接利用目标函数值，而是使用目标函数值经过一定处理后得到的信息，该过程便是适应度定标。定标对象嵌入到群体对象中，追踪群体中各个染色体的适应度记分，其中，适应度定标方法类(GASelectionScheme)定义了遗传算法群体中基本选择算子的行为以挑选群体中配对的染色体。常用的选择方式包括归类选择、轮盘赌选择、竞争选择、剩余随机抽样、均匀随机抽样以及确定性采样。

3 GAlib在TSP问题中的应用

旅行商问题[4](Traveling Salesman Problems，简写为 TSP)是计算机科学中的一个经典问题。问题描述如下：旅行的商人必须在指定的活动区域内访问每个城市，且每个城市只能被访问一次，遍历所有城市后返回到起点。给定每对城市间的旅行路程，求解的任务是确定一个完整的旅行次序，使得商人行走的路程最短。形式化定义如下：对给定的完全图 G，有端点集合 V 和边集合 E，以及各边的权值 $d_{i,j}$，用 E 中的每条边连接整个图。权值 $d_{i,j}$ 是从节点 $i\in V$ 旅行到节点 $j\in V$ 的途中行程。根据上述信息，求解 TSP 就是返回其中行程最短的 Hamilton 回路。即回路中的每个节点均被访问且仅被访问一次[3]。

TSP 问题的数学模型为：对于城市 $V=\{v_1,v_2,\ldots,v_n\}$ 的访问顺序为 $T=(t_1,t_2,\ldots,t_n)$,其中 $t_i\in V(i=1\sim n)$，且 $t_{n+i}=t_i$，则问题为求 $\min\limits_{T\in\Omega}\sum\limits_{i=1}^{n}d_{ti,ti+1}$，其中 Ω 为这 n 个城市不重复排列的所有可能的回路。

遗传算法求解 TSP 问题的基本流程如下图所示：

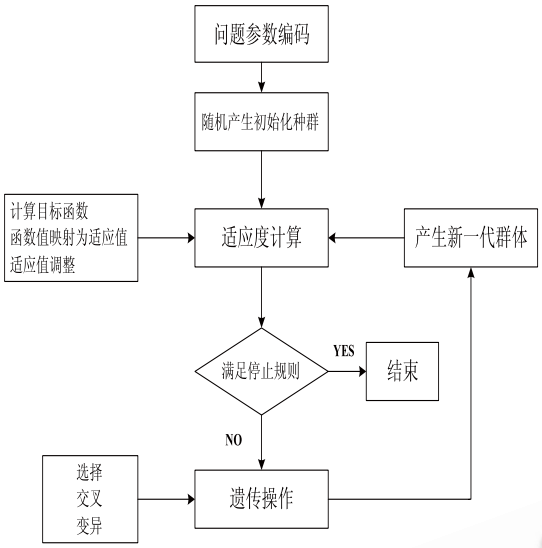

图 1　遗传算法流程图

本节以下部分介绍利用 GAlib 实现遗传算法来解决 TSP 问题的关键技术。

3.1 染色体编码

用遗传算法求解 TSP 问题，染色体编码的方式很重要。相比于二进制表示、近邻表示、次序表示、矩阵表示和边的表示，路径编码最为自然、直观。所谓路经编码是指直接采用城市在路径中的位置进行表示，例如回路：2-4-5-8-7-9-6-1-3-2 直接编码为(2458796132)。本文采用 GAlib 中的链表染色体(GAListGenome)，以路径表示的方式对染色体进行编码，核心代码如下所示：

```
town=GARandomInt(0,ntowns-1);
visit[town]=1;
child.insert(town,GAListBASE::HEAD);
for( i=1; i<ntowns; i++) {
  do {
    town=GARandomInt(0,ntowns-1);
  } while (visit[town]);
  visit[town]=1;
  child.insert(town);
}
```

其中 child 是链表型染色体，twon 和 ntowns 分别代表城市编号和城市总数，visit 数组记录已经访问过的城市，GARandomInt 函数生成指定范围内的随机整数。本段程序首先初始化出发城市，然后利用一个循环操作，随机确定城市遍历的顺序。

3.2 初始化种群

本文初始化种群的方式如下，采用 GAlib 中的稳态遗传算法 GASteadyStateGA，种群在进化过程中以选择最短旅行路径为目的，父代与子代间的覆盖率为 1%，种群规模为 100，迭代次数为 1000 代，其间，变异率为 0.1%，交叉率为 1%，主要代码如下所示：

```
GASteadyStateGA ga(genome);
ga.minimize();
ga.pReplacement(1.0);
ga.populationSize(100);
ga.nGenerations(1000);
ga.pMutation(0.1);
ga.pCrossover(1.0);
```

3.3 适应度函数

遗传算法的适应度函数是用来判断群体中个体的优劣程度的指标，根据所求问题的目标函数进行评估。

在 TSP 问题中，染色体优劣的明确指标便是商人旅行的路程总和，根据这一特点，本文以染色体对应路径的长度为评估个体适应程度的标准，主要过程如下所示：

```
float dist = 0;
if(genome.head()) {
  for(int i=0; i<ntowns; i++) {
    int xx = *genome.current();
    int yy = *genome.next();
    dist += DISTANCE[xx][yy];
  }
}
return dist;
```

其中，dist 代表路程的长度，genome 为考虑的染色体，DISTANCE 二维数组存储的是两城市间的距离。

3.4 遗传操作

3.4.1 选择算子

在遗传算法的群体中选择优胜的个体，淘汰劣质个体的操作叫做选择。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。本文采用轮盘赌选择法（roulette wheel selection），在该方法中，各个个体的选择概率与其适应度值成比例。

3.4.2 交叉算子

交叉算子设计应遵循两个基本原则：其一，交叉

生成的子代应尽量保留父代的优良子路径：其二，交叉生成的子代必须是有效染色体，即符合有序编码规则。因此，TSP 问题对交叉算子的设计要求是：对任意两条巡回路线进行交叉操作之后，都能够得到另外两条新的并且具有实际意义的巡回路线。遗传算法中传统的交叉方式有部分匹配交叉、顺序交叉、循环交叉、基于位置的交叉等，然而这些交叉算子要么没有充分考虑 TSP 的特点，忽视了父代中边的邻接状况；要么交叉后的子代个体不能很好地保留父代的优秀基因。鉴于此，本文采用边重组交叉算法，该算法能够较好地保留父代的边。以 10 个城市的 TSP 问题为例，假设已知：

P1 = (1 4 2 8 6 9 10 7 5 3)

P2 = (2 3 1 4 5 10 8 9 6 7)

与 i 点邻接的点用 Ai 来表示，则有

A_1 = {3,4}，A_2 = {4,8,3,7}，A_3 = {1,2,4}，

A_4 = {1,2,5}，A_5 = {3,4.7,10}，A_6 = {7,8,9}，

A_7 = {2,5,6,10}，A_8 = {2,6,9,10}，A_9 = {6,8,10}，

A_{10} = {5,7,8,9}

P’ = (1 X X X X X X X X X) 从 A_1 中随机选择 4

P’ = (1 4 X X X X X X X X) 从 A_4 中随机选择 5

P’ = (1 4 5 X X X X X X X) 从 A_5 中随机选择 3

P’ = (1 4 5 3 X X X X X X) 从 A_3 中随机选择 2

P’ = (1 4 5 3 2 X X X X X) 从 A_2 中随机选择 8

P’ = (1 4 5 3 2 8 X X X X) 从 A_8 中随机选择 6

P’ = (1 4 5 3 2 8 6 X X X) 从 A_6 中随机选择 7

P’ = (1 4 5 3 2 8 6 7 X X) 从 A_7 中随机选择 10

P’ = (1 4 5 3 2 8 6 7 10 X) 从 A_{10} 中随机选择 9

P’ = (1 4 5 3 2 8 6 7 10 9)

可见，边重组算法步骤如下：

1. 从 P1 ，P2 中构造 A_i，i = 1,2,…n。k = 1,$c_1 = a_1$。

2. 从 Ai 删除 ck，从 Ack 中随机生成 a，k=k+1, c_k = a。

3. 若 k<=n 则转至步骤 2，否则输出 P1’=($c_1c_2…c_n$)。

采用类似方法构造 P2’

3.4.3 变异算子

变异算子的基本内容是对群体中的个体串的某些基因组上的基因值作变动。遗传算法引入变异的目的有两个：一是使遗传算法具有局部的随机搜索能力。二是使遗传算法可维持群体多样性，以防止出现未成熟收敛现象。本文在利用 GAlib 解决 TSP 问题时，采用的变异操作是对换变异，该操作随机选择串中的两点，交换其值。例如对于旅行城市次序 A：l-2-3-4-5-6-7-8-9，若对换城市 4 和 7，则经对换后 A’：1-2-3-7-5-6-4-8-9。

3.5 终止

遗传算法终止方式有三种，一是当最优个体的适应度达到给定的阀值，二是最优个体的适应度和群体适应度不再上升，三是迭代次数达到预设的代数。本文利用 GAlib 中 GASteadyStateGA 类的成员函数 nGenerations()终止遗传算法的运行，预设的停止代数为 1000 代。

4 实验结果

对于解决 TSP 问题的算法来说，结果的精度和时间复杂度是最重要的两个性能评价标准。本文在 VC6.0 开发环境下基于 GAlib 类库(本文采用的是 galib247)实现了遗传算法，实验平台为 Pentium-IV/3.2-GHz/2.0-Gbyte RAM，实验所用测试数据是从 TSPLIB[5]中选择的 tburma14，bayg29，DANTZIG42 及 eil51。其中 TSPLIB 为国内外研究 TSP 问题通用的测试用例集。

表 1 基于 GAlib 实现的遗传算法精确度比较

TSP	Opt-out	City-num	GA-result	Std%
tburma14	30.8785	14	30.8785	0.0
bayg29	9047	29	9076.98	0.3
DANTZIG42	699	42	717.766	2.8
eil51	426	51	454.105	6.6

表 1 显示的是算法运行 tburma14，bayg29，DANTZIG42 及 eil51 的结果，其中 Std%代表求得的解与最优解的差值百分比。不难看出，本文实现的遗传算法求得的解与最优解平均误差小于 2.5%，由此可见利用 GAlib 实现的遗传算法得出的结果比较精确。

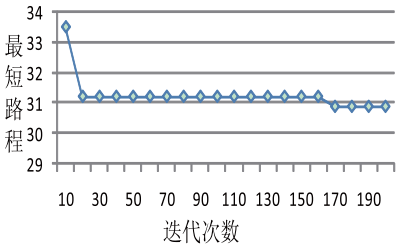

图 2 运行 tburma14 实例收敛情况

图 2 和图 3 分别显示的是 tburma14 和 bayg29 在运行过程求得的最优解与迭代次数的关系。

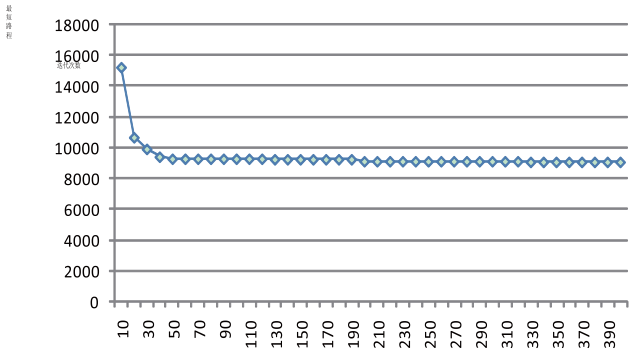

图 3　运行 bayg29 实例收敛情况

从图中曲线趋势可以看出，本文实现的算法迭代 100 次以内便接近问题的最优解，由此可见利用 GAlib 实现的遗传算法可以快速向最优解收敛。

5　结语

本文利用 GAlib 实现了遗传算法，并以解决 TSP 问题为例对相关技术进行说明，实际开发过程表明，基于 GAlib 实现的遗传算法代码精简，开发迅速，可读性良好，并且实验结果显示，算法收敛快速，运行结果准确。

参考文献

1 Holland JH. Adaptation in Natural and Artificial Systems. AnnArbor:University of Michigan press, 1975.

2 Wall M. Galib.[2007-03]. http://lancet.mit.edu/ga/dist/.

3 王小平,曹立明.遗传算法-理论、应用与软件实现.西安:西安交通大学出版社,2002.

4 Dantzig GB, Fulkerson DR, Johnson SM. Solution of a large scale traveling salesman problem. Operation Research, 1954, (2):393–410.

5 http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsp/.

（上接第 167 页）

4 Moonja J, Gi OK, Seong AK. Dynamics of Newton's method for solving some equations. Computers & Graphics, 2002, 26(2):271–279.

5 Gilbert WJ. Generalizations of Newton's method. Fractals, 2001,9(3):251–262.

6 Wang XY, Liu W. The Julia set of Newton's method for multiple roots. Applied Mathematics and Computation, 2006,171(1):101–110.

7 王兴元,刘威.三阶广义牛顿变换的 Julia 集.工程图学学报, 2005,26(2):119–127.

8 胡多能,王京,张瑞秋.分形图形与混沌图案的应用.计算机工程与设计,2007,28(4):893–895.