文章编号: 1673-3193(2007)01-0049-04

用遗传算法求解旅行商问题

*

李　飞, 白艳萍

(中北大学 理学院, 山西 太原 030051)

摘　要:　论述了用遗传算法求解旅行商问题(TSP)的算法步骤, 给出了在MATLAB 环境下用遗传算法解决旅行商问题的具体程序设计. 将此算法应用到 6 个旅行商问题中, 并将得到的运行结果与用弹性网络得到的结果进行了比较, 发现用遗传算法得到的结果与最优解较为接近

关键词:　遗传算法; 旅行商; 最优化
中图分类号:　O29　　**文献标识码:** A

A Genetic Algorithm for Traveling Salesman Problems

LI Fei, BAI Yan-ping

(School of Science, North University of China, Taiyuan 030051, China)

Abstract: The procedure of solving Traveling Salesman Problem (TSP) by genetic algorithm was expounded. The detailed program solving TSP in MATLAB environment was presented. The algorithm was applied to six Traveling Salesman Problems. The results from genetic algorithm are rather conform to the optimal result, compared the results of genetic algorithm with the elastic net.

Key words: genetic algorithm; traveling salesman problem; optimal

"旅行商问题"(Traveling Salesman Problem, TSP)可简单描述为: 一位销售商从n 个城市中的某一城市出发, 不重复地走完其余$n-1$ 个城市并回到原出发点, 在所有可能路径中求出路径长度最短的一条. 用数学语言描述如下: 设有限 n 个城市集合 $C=\{C_1, C_2, \cdots, C_n\}$; 设两个城市间的距离为 $d(C_i, C_j)\in \mathbf{Z}^+$, 其中 $C_i, C_j \in C\ (1\leq i, j\leq n)$; 求使 $\min\{\sum_{i=1}^{n-1} d(C_{I(i)}, C_{I(i+1)}) + d(C_{I(n)}, C_{I(1)})\}$ 的城市序列 $\{C_{I(1)}, C_{I(2)}, \cdots, C_{I(n)}\}$, 其中 $I(1)$, $I(2)$, $\cdots$, $I(n)$ 是 $1, 2, \cdots, n$ 的一个全排列.

该问题是一个相当古老的优化问题, 早在 1759 年 Euler 就研究过, 同时 TSP 问题已被证实是一个 NP 难解问题[1]. 在过去的 10 年中, 出现了一些逼近最优解的算法: 最近邻居、贪婪算法、最近插入、最远插入、双最小生成树、带解法、空间充填曲线及由 Karp, Litke, Christofides 等开发的算法[2]. 但因这些方法的局限性, 故需要寻找一些启发式算法. 20 世纪 70 年代初期由美国密执根大学的 Holland 教授发展起来的遗传算法, 是一种求解问题的高效并行全局搜索方法, 能够解决复杂的全局优化问题. 解决 TSP 问题也成为遗传算法界的一个目标.

1　用遗传算法求解 TSP

1.1　遗传算法的基本步骤

遗传算法是通过借鉴生物界自然选择和自然遗传机制而产生的一种计算方法, 与其他的优化算法一

*　收稿日期: 2006-06-16
基金项目: 山西省自然科学基金资助项目(20051006)
作者简介: 李飞(1979-), 男, 硕士生. 主要从事计算机科学中的数学问题的研究. 白艳萍(1962-), 女, 教授. 硕士生导师

样，遗传算法也是一种迭代算法．从选定的初始解出发，通过不断地迭代，逐步改进当前解，直到最后搜索到最优解或满意解．其迭代过程是从一组初始解(群体)出发，采用类似于自然选择和有性繁殖的方法，在继承原有优良基因的基础上生成具有更好性能的下一代解的群体．遗传算法的运算过程为：对给定问题，给出变量的编码方法，定义适应度函数．① 初始化．令 $t=0$，给出正整数 T(最大迭代次数)，交叉概率 p_c及变异概率 p_m，随机生成M 个个体作为初始群体 $p(0)$；② 个体评价．计算 $p(t)$ 中各个个体的适应度；③ 选择．对群体 $p(t)$ 进行选择操作，得到中间群体；④ 交叉．把交叉操作作用于中间群体；⑤ 变异．把变异操作作用于交叉之后所得到的群体，则得到第 $(t+1)$ 代群体 $p(t+1)$；⑥ 若 $t<T$，则令 $t=t+1$，转②；若 $t\leq T$，则以进化过程中所得到的具有最大适应度的个体作为最优解，运算停止．算法流程图如图1所示．

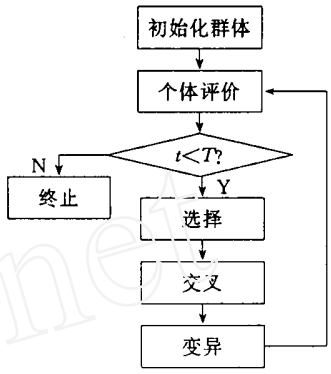

图1 遗传算法流程图

Fig. 1 Flow chart of genetic algorithm

1.2 在MATLAB下的程序设计

对于需要解决的TSP问题来说，其 n 个城市之间的距离实质上构成了一个 $n\times n$ 的矩阵，同时遗传算法的诸多算子(如选择、交叉、变异)，都是针对所谓染色体进行的，而染色体实质上是一个向量，可将其看成一个 $1\times n$ 的矩阵，因此这些算子实质上是一些矩阵运算，而MATLAB的基本数据单元就是一个维数不加限制的矩阵，故用它编写程序，比用其它高级语言更简单、灵活、快捷．

1) 初始化群体．在初始化群体之前，需要确定变量的编码方法及适应度函数．在遗传算法界有一个共识：旅行的二进制表达对TSP不是最适合的[3]．这里以城市的遍历次序作为算法编码，即 $(i_1, i_2, \cdots, i_n)$ 是 $\{1, 2, \cdots, n\}$ 的全排列．因为所求的是目标函数最小值的优化问题，因此以哈密顿圈的长度的倒数作为适应度函数，同时需要对种群大小、最大迭代次数、交叉概率、变异概率等变量赋值[4]，随机生成种群大小M．染色体长度(城市数目)为 num 的MATLAB程序为：

```
% 生成初始群体
popm = zeros(M, num);
for i = 1:M
    popm(i,:) = randperm(num);      % 随机产生一个由自然数 1 到 num 组成的全排列
end
```

2) 个体适应度计算．计算个体的适应度，即与种群中某一个个体 r 相对应的哈密顿圈长的倒数．

```
function lem = value(r, dmatrix, num)    % dmatrix 表示任意两个城市之间的距离构成的距离矩阵
minopt = 0;                              % 哈密顿圈长
for i = 1: num - 1                       % 按照遍历次序求哈密顿圈长
    minopt = minopt + dmatrix(r(i), r(i+1));
end
minopt = minopt + dmatrix(r(1), r(num));
lem = 1/minopt;
```

3) 比例选择操作．具体执行过程为：①计算种群中所有个体的适应度总和；②计算每个个体的相对适应度大小，即各个个体在选择操作中被选中的概率；③使用模拟赌盘操作，来确定各个个体被选中的次数，得到中间群体．

```
% 计算种群中所有个体的适应度总和
lengthall = 0;
for i = 1:M
    r = popm(i,:);
    lengthone = value(r, dmatrix, num); % 调用上面的value 函数
    lengthall = lengthone + lengthall;
end
```

```
% 确定各个个体的相对适应度大小
gailu= rands(1,M);                    % 生成1×n 随机矩阵,用于存储各个个体的相对适应度
for i= 1:M
    r= popm(i,:);
    gailu(1, i)= value(r, dmatrix, num)/lengthall;
end
% 确定被选择的群体
father= ones(M, num);                 % 生成一个矩阵,用于存储被选择的群体
i= 1;
while i<= M
    jc= rand;
    j= 1;
    for j= 1:M
        if jc<= gailu(1, j);          % 按照随机生成的数,来确定被选择的个体
        father(i,:)= popm(j,:);       % 生成中间群体
        i= i+ 1;
        j= M+ 1;
end
    end
end
```

4) **交叉操作**　对选择操作得到的中间群体进行以下操作: ① 以交叉概率 p_c 从中间群体中随机地选出需要进行交叉的个体,对这些个体随机地两两配对; ② 在 1～(num - 1)之间产生两个随机数 k, l, 这两个数表示交叉点的位置; ③ 对已经配对的两个个体,相互对应地交换第 k 到 l 位的数字.

```
son= father;                          % 用 son 矩阵表示交叉之后的群体,对其初始化为选择后群体
for i= 1:M * pc* 0.5
    a= floor(rand* M)+ 1;
    b= floor(rand* M)+ 1;
    father1= father(a,:);
    father2= father(b,:);
    k= floor(num * rand)+ 1;
    l= floor(num * rand)+ 1;
    minshu= min(k, l);
    maxshu= max(k, l);
    select1= father1(minshu:maxshu);
    select2= father2(minshu:maxshu);
    father12= delet (father1, select2);
    father21= delet(father2, select1);  % 其中 delet 是自定义函数
    son(a,:)= [select2 , father12];     % 将两矩阵拼接
    son(b,:)= [select1 , father21];
end
% 自定义函数delet(x, y)
function z= delet(x, y)
for n= 1: length(y)
    x(:, find(x== y(n)))= [];           % 功能为将 x 中与 y 相同的元素删除
end
z= x;
```

5) **变异操作**　此操作作用于交叉操作之后的群体,其操作方法为: 首先在 1～num 之间产生两个随机数 g, h, 这两个数表示变异点的位置; 最后再以变异概率随机地从群体中选出的个体上, 将 g, h 两位

置上的数互换

```
% g, h 是两个变异点的位置
for i= 1:M * pm
    c= floor(rand * M) + 1;
    g= floor(rand * num) + 1;
    h= floor(rand * num) + 1;
    zhjian= son(c, g);
    son(c, g)= son(c, h);
    son(c, h)= zhjian;
end
```

2 结果评价

为检验该算法的实际应用效果, 文中以 16, 22, 24, 48, 70, 101 个城市为例(这 6 个 TSP 实例, 弹性网络算法的计算结果及最优解见参考文献[5]), 将该算法得到的结果与用弹性网络算法得到的结果进行了比较, 如表 1 所示

为了更好地说明遗传算法在求解 TSP 问题中良好的全局搜索能力, 文中以求解 101 个城市为例进行运算. 通过运算可以发现, 随着迭代次数的增加, 最佳个体的适应度函数值逐步增加, 最后收敛于一个相对稳定值. 用遗传算法求解得到的 101 个城市的遍历图, 如图 2 所示

从表 1, 图 2 中可以看出, 遗传算法有较好的全局最优解的求解能力.

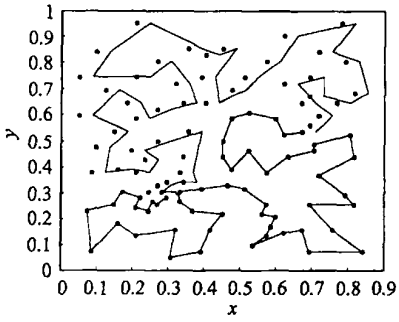

图 2 101 个城市的遍历图

Fig 2 Traversal figure of 101 cities

表 1 遗传算法与弹性网络算法计算结果比较

Table 1 Result of between genetic algorithm and elastic net

城市数	16	22	24	48	70	101
最优解	2.126	2.171	4.399	3.892	6.169	7.811
遗传算法	2.127	2.178	4.400	4.044	6.253	7.887
弹性网络	2.154	2.222	5.047	4.135	6.557	8.253

3 结 论

在算法实施过程中, M, T, p_c, p_m 等参数值的设定对于问题能否找到满意解起着非常重要的作用. 如果 M, T 值太小, 则不容易找到最优解; 反之, 则计算时间长. 对于交叉概率 p_c 来说, 如果取值过大, 则会破坏以前遗传的结果, 因而 p_c 不能取得太大. 对于突变概率 p_m, 其目的是为了保证算法对解空间的覆盖, 但 p_m 太大, 会破坏遗传和交叉选出的染色体而变成随机搜索; 反之, 染色体种群又会过于单调, 从而陷入局部极值. 因此, 在实际计算时, 要根据具体问题, 来选择合适的参数

参考文献:

[1] Garey M, Johnson D. Computers and Intractability[M]. Freeman W H, San Franciso, 1979.

[2] Johnson D S. Local optimization dan the traveling saleman problem. in M. S. Paterson (Editor) [C]. Proceedings of the 17th Collouium on Automata, Languages, and Programming Springer-Verlag, Lecture Notes in Compter Scinence, 1990, 443: 446-461.

[3] 米凯利维茨 Z. 演化程序——遗传算法和数据编码的结合[M]. 北京: 科学出版社, 2000.

[4] 解可新, 韩健, 林友联. 最优化方法[M]. 天津: 天津大学出版社, 1997.

[5] Zhang Wendong, Bai Yanping. A hybrid elastic net method for solving the traveling salesman problem [J]. International of Software Engineering and Knowledge Engineering, 2005, 15(2): 447-453.

[6] 白艳萍, 胡红萍. 一个改进的弹性网络算法求解 TSP 问题[J]. 华北工学院学报(自然科学版), 2004(4): 235-238.
Bai Yanping, Hu Hongping. A modified elastic net algorithm for traveling salesman problem [J]. Journal of North China Institute of Technology, 2004(4): 235-238. (in Chinese)