文章编号: 1001-0920 (2014) 08-1483-06 **DOI:** 10.13195/j.kzyjc.2013.0598

改进的遗传算法求解旅行商问题

于莹莹, 陈 燕, 李桃迎

(大连海事大学 交通运输管理学院，辽宁 大连 116026)

摘 要: 提出一种解决旅行商问题的改进遗传算法. 在传统遗传算法的基础上, 引入贪婪算法进行种群初始化; 从遗传进化代数和个体适应函数值两个方面实现遗传参数自适应调节, 在加快寻优速度的同时防止寻优陷入局部最优; 采用基于贪婪方法的启发式交叉算子优化交叉结果; 对交叉前后的种群分别实施精英个体保留策略, 保证最优基因结构得以延续. 实验结果分析表明, 改进的遗传算法可以在种群规模较小的情况下具有更可靠的寻优能力.

关键词: 旅行商问题；遗传算法；贪婪算法；自适应调节

中图分类号: TP301.6；TP18 **文献标志码:** A

Improved genetic algorithm for solving TSP

YU Ying-ying, *CHEN Yan*, *LI Tao-ying*

(College of Transportation Management，Dalian Maritime University，Dalian 116026，China．Correspondent：YU Ying-ying，E-mail：uee870927@126.com)

Abstract: An improved genetic algorithm for solving traveling salesman problem(TSP) is proposed. Based on the traditional genetic algorithm, the proposed algorithm introduces the greedy method into species initialization. In order to improve the optimization speed and prevent the local minimum, the improved algorithm updates the crossover probability and mutation probability adaptively according to the evolution stages and the fitness value of individuals. The heuristic crossover operator based on greedy method is used to optimize the crossover results. The strategy of keeping the best individuals to propagate the optimal gene structure is introduced. The results of TSP example show that the improved algorithm can find the global optimal solution with high performance.

Key words: traveling salesman problem；genetic algorithm；greedy method；adaptive adjustment

0 引 言

旅行商问题 (TSP) 是组合优化领域中著名的 NP-hard 问题, 具有较为广泛的工程应用和现实生活背景, 如印刷电路钻孔、飞机航线的安排、公路网络的建设、网络通信节点的设置、物流货物配送、超市物品上架等, 所有这些实际应用问题均可以转变为 TSP 问题来解决. 因此, 如何快速、有效地解决 TSP 问题具有很高的实际应用价值. 当前, 随着人工智能的发展, 有许多智能优化算法应用于 TSP 问题, 高海昌等[1]对蚁群算法、遗传算法、模拟退火算法、禁忌搜索、神经网络、粒子群优化算法、免疫算法等进行了叙述. 随着研究的深入, 许多改进的优化算法不断涌现, 文献 [2] 提出了一种快速求解旅行商问题的蚁群算法. 文献 [3] 将蚁群算法与免疫算法相结合应用于 TSP 问题求解. Marinakis 等[4]对基于概率的旅行商问题进行研究, 提出了一种混合多粒子群优化算法. 文献 [5] 的混合遗传算法将局部寻优与广义分割交叉算子结合, 提高了算法的执行效率. 文献 [6] 对传统遗传算法的交叉操作进行研究, 设计了一种顺序构造交叉算子以提高解的质量. 为了提高遗传算法解决 TSP 问题的效率, 文献 [7] 将贪婪搜索算法引入遗传变异操作中, 设计出新的贪婪子路变异算子.

本文针对智能算法中的遗传算法进行分析, 对其在 TSP 问题求解应用上加以改进, 加强其寻优能力. 实验分析表明, 改进的遗传算法可以在种群规模较小的情况下具有更可靠的寻优能力.

收稿日期: 2013-05-09；**修回日期:** 2013-12-06.

基金项目: 国家自然科学基金项目(71271034)；辽宁省教育厅科学研究一般项目(L2012173)；中央高校基本科研业务费专项基金项目(3132013319, 2013YB06).

作者简介: 于莹莹(1987−), 女, 博士生, 从事数据挖掘、系统集成的研究；陈燕(1952−), 女, 教授, 博士生导师, 从事管理科学与决策、知识管理、数据仓库与数据挖掘等研究.

1 基本问题描述

1.1 TSP 问题

TSP 问题最早的描述是 1759 年欧拉研究的骑士周游问题, 即对于国际象棋棋盘中的 64 个方格, 走访 64 个方格一次且仅一次, 并最终返回起点[8]. 其一般性描述为: 有一旅行商要访问 n 个城市, 每个城市必须访问且只能访问一次, 需要寻求到一条包含所有 n 个城市的最短访问路线[9]. 可以建立以下数学模型: 求满足下式的最优路径 $R=(c_0,c_1,\cdots,c_{n-1})$:

$$\min\ f(R). \tag{1}$$

其中

$$f(R)=\sum_{i=0}^{n-2} d(c_{\eta_i},c_{\eta_{i+1}})+d(c_{\eta_{n-1}},c_{\eta_0}), \tag{2}$$

η_i 为 $(0,1,\cdots,n-1)$ 的一个置换, c_{η_i} 为城市号, $d(c_{\eta_i},c_{\eta_{i+1}})$ 为城市 c_{η_i} 到城市 $c_{\eta_{i+1}}$ 的距离. 本文主要讨论对称的 TSP 问题, 因此有 $d(c_{\eta_i},c_{\eta_{i+1}})=d(c_{\eta_{i+1}},c_{\eta_i})$. 该问题是一个典型的优化组合难题, 已被证明属于 NP 完全问题, 即没有确定的算法能在多项式时间内得到问题的最优解.

1.2 遗传算法简介

遗传算法是一种基于自然选择和基因遗传学原理的随机并行搜索算法, 是一种寻求全局最优解而不需要任何初始化信息的高效优化方法[10]. 它将问题的解集看作一个种群, 通过不断地选择、交叉、变异等遗传操作, 使解的质量越来越好. 该算法具有全局寻优能力、适应性强、解决非线性问题具有较强的鲁棒性、对问题没有特定限制、计算过程简单、对搜索空间没有特殊要求、易于与其他算法结合等特点, 在函数优化、图像处理、系统辨识、自动控制、经济预测和工程优化等领域得到了广泛的应用[11], 在求解 NP 完全问题方面是一种较为有效的全局方法.

2 遗传算法解决 TSP 问题原理

遗传算法是以适应度为依据的逐代搜索过程, 主要由编码机制、控制参数、适应度函数和遗传算子 4 部分组成[12]. 简单遗传算法求解 TSP 问题的主要计算过程如下.

Step 1: 确定编码机制, 生成初始种群. 解决 TSP 问题通常采用城市序号对路径进行编码, 按照访问城市的顺序排列组成编码.

Step 2: 计算种群中每个个体的适应度值. TSP 求解是要寻找使目标函数最小的个体, 因此选择适应度函数 $\text{fitness}(i)=D/f(R_i)$. 设置常数 D, 防止路径值过大而导致适应度函数倒数接近于 0. 可以看出, 巡游路径越小, 适应度值越大.

Step 3: 选择算子. 通常采用精英个体保存策略和赌轮选择算子, 即适应度最高的个体一定被选择. 计算每个个体在整个种群适应度中的被选择概率和累计概率分别为

$$P_i=\text{fitness}(i)\Big/\sum_{i=1}^{\text{popsize}}\text{fitness}(i),\ Q_i=\sum_{j=1}^{i}P_j.$$

通过随机数 r 所在的区间范围选择遗传个体.

Step 4: 交叉算子. 由交叉概率 p_c 选择若干父体并进行配对, 按照交叉算法的规则生成新个体, 常用的规范方法有单点交叉、部分映射交叉、循环交叉等.

Step 5: 变异算子. 为了保持种群个体的多样性, 防止陷入局部最优, 需要按照某一变异概率 p_m 随机确定变异个体, 并实行相应变异操作, 通常采用逆序变异算子.

Step 6: 迭代终止条件. 若满足预定的终止条件 (达到最大迭代次数), 则停止迭代, 所得的路径认为是满意的路径; 否则, 转至 Step 2, 计算新一代种群中每个个体的适应度值.

简单遗传算法往往存在收敛速度慢、易陷入局部最优和优化精度低等明显不足. 如何在提高算法收敛速度的同时确保种群多样性, 使寻优结果接近最优解是遗传算法不断改进的目标.

3 求解 TSP 问题的改进遗传算法

3.1 基于贪婪算法生成初始种群

简单遗传算法通过随机方法生成初始种群, 初始种群个体适应度较低, 在一定程度上会制约算法的收敛速度. 本文采用贪婪算法对初始个体进行优化, 利用贪婪算法局部寻优的优势产生新个体. 首先随机选择一个城市作为旅行商当前所在城市 c_{current}, 并加入个体中, 搜索所有未加入个体中的城市, 找到距离当前城市 c_{current} 最近的城市 c_{next}, 将其添加至个体中并作为当前城市, 继续搜索并添加下一最近城市, 直至所有城市都加入个体, 由此得到初步优化的个体. 由实验结果可知, 贪婪算法生成的初始种群不失随机性, 同时整体质量有所提高, 有助于加快寻优速度.

3.2 自适应参数调节

在简单遗传算法中, 参数交叉概率 p_c 和变异概率 p_m 在进化过程中固定不变, 但实际研究表明二者是影响遗传算法性能的关键. 许多学者对参数的自适应控制进行了研究[10,11,13-15], 本文在此基础上, 分别对交叉概率和变异概率设置调节机制.

3.2.1 自适应交叉概率调节机制

交叉算子对种群实现不断更新, p_c 的大小决定种群个体的更新速率, 其值过大, 会破坏优良的遗传模式, 取值过小会导致算法搜索速度缓慢, 种群难以得到进化. 在进化前期, 为了扩大整体搜索范围, 加快种

群更新速度, 应该增大 p_c 的值; 在进化后期, 种群整体解集趋于稳定, 为了使优良基因结构得以延续保存, 应适当降低 p_c. 另外, 交叉算子可以改变甚至破坏基因结构, 对适应度较差的个体而言, 更多地参与交叉操作可以促进其不断优化, 所以应给予较高的p_c. 相应地, 适应度越高的个体, 为了防止基因结构被破坏, 进行交叉操作的概率应当越小. 基于上述考虑, 设置如下调节机制:

$$p_{ci}=\begin{cases}p_{c\max}-(p_{c\max}-p_{c\min})\Big(\dfrac{g}{2G}+\dfrac{f_i-\bar{f}}{2(f_{\max}-\bar{f})}\Big),\\ \qquad\qquad f_i\geqslant\bar{f};\\ p_{c\max},\ f_i<\bar{f}.\end{cases}\tag{3}$$

$$p_{c\max}=\begin{cases}0.9,\ g\leqslant G/4;\\ 0.8,\ G/4<g\leqslant 3G/4;\\ 0.7,\ 3G/4<g\leqslant G.\end{cases}\tag{4}$$

其中: p_{ci} 为个体 i 发生交叉算子的概率, 在进化初期给予较高的交叉概率, 在后期降低交叉概率; G 为进化过程的最大迭代数; g 为当前迭代数; $p_{c\max}$ 的取值与进化迭代数相关, $p_{c\min}=0.6$; f_i 为个体 i 的适应度函数值, $f_{\max}$ 为当前所有个体的最大适应度值, 是当前种群的平均适应度值. 由式 (4) 可见, 交叉概率与当前迭代数和当前种群的进化状况密切相关.

3.2.2　自适应变异概率调节机制

p_m 影响种群的变异情况, 个体的适当变异可以保持种群多样性, 防止陷入局部最优. 但是, 如果 p_m 取值过大, 算法则近似于随机搜索, 失去了遗传进化特性. 从遗传进化代数和种群个体适应函数值两个方面对变异概率建立调节公式, 有

$$p_{mi}=\begin{cases}p_{m\min}+(p_{m\max}-p_{m\min})\Big(\dfrac{g}{2G}+\dfrac{f_i-\bar{f}}{2(f_{\max}-\bar{f})}\Big),\\ \qquad\qquad f_i\geqslant\bar{f};\\ p_{m\min},\ f_i<\bar{f}.\end{cases}\tag{5}$$

其中 p_{mi} 为个体 i 发生变异的概率. 由式 (5) 可见, 适应函数值越小的个体发生变异的可能性越小, 随着迭代次数的增加, 种群个体趋于相似的基因结构, 此时很有可能陷入局部最优. 为了避免这种情况, 应适当增大个体发生变异的概率, 鼓励新型个体的产生, 保持个体多样性. 设置 $p_{m\max}=0.005$, $p_{m\min}$ 随着进化次数有所调整, 有

$$p_{m\min}=\begin{cases}0.001,\ g\leqslant G/4;\\ 0.002,\ G/4<g\leqslant 3G/4;\\ 0.003,\ 3G/4<g\leqslant G.\end{cases}\tag{6}$$

在进化初期, 个体发生变异的可能性较小, 在进化末期, 提高种群个体变异操作的概率, 有利于扩大搜索范围, 跳出局部最优.

3.2.3　启发交叉算子

交叉算子的性能直接影响种群的进化速度和群体质量, 规范化的交叉算子按照某种交换模式, 并没有考虑如何使后代更加优化. 本文结合文献 [16] 引入一种启发式交叉算子, 该算子结合贪婪算法和淘汰机制, 以得到更好的进化后代.

假设交叉配对的父代染色体为 P_1 和 P_2, 生成的子代个体为 O_1 和 O_2. 随机产生起始出发城市 C_{start} 作为 O_1 和 O_2 的起始城市, 按照第 3.1 节的贪婪规则生成个体 O_3, 作为备选子代个体. 由贪婪算法产生的个体属于局部最优, 具有较好的基因结构.

按照正向添加的规则生成 O_1, 分别计算在父代 P_1 和 P_2 中出发城市 C_{start} 与下一邻近城市 C_{next1} 和 C_{next2} 的距离, 选择距离最近的城市 (C_{next1} 或 C_{next2}) 加入子个体 O_1 中, 并将该城市作为出发城市, 继续寻找下一最近邻城市, 直至个体基因添加完毕.

按照逆向添加的规则生成 O_2, 分别计算在父代 P_1 和 P_2 中出发城市 C_{start} 与上一邻近城市 $C_{\text{previous1}}$ 和 $C_{\text{previous2}}$ 的距离, 选择距离最近的城市 ($C_{\text{previous1}}$ 或 $C_{\text{previous2}}$) 加入子个体 O_2 中, 并将该城市作为出发城市, 继续寻找上一最近邻城市, 直至个体基因添加完毕.

计算并比较备选子代个体 O_3 与 O_1, O_2 的适应函数值大小, 用 O_3 替换适应函数值较小的个体, 从而得到优化后的子代个体.

假设 6 个城市的坐标分别为 $c_0(10,75)$, $c_1(36,9)$, $c_2(91,78)$, $c_3(54,53)$, $c_4(8,51)$, $c_5(78,51)$. 通过计算得到城市之间的距离如表 1 所示.

表 1　规模为 6 的城市间距离

city	c_0	c_1	c_2	c_3	c_4	c_5
c_0	0	70.936	81.055	49.193	24.083	72.111
c_1	70.936	0	88.238	47.539	50.477	59.396
c_2	81.055	88.238	0	44.654	87.281	29.966
c_3	49.193	47.539	44.654	0	46.043	24.083
c_4	24.083	50.477	87.281	46.043	0	70
c_5	72.111	59.396	29.966	24.083	70	0

对于个体 P_1 和 P_2, 随机产生的起始出发城市 C_{start} 的序号为 c_4, 按照上述规则, 生成子代个体, 如图 1 所示. 由图 1 可见, O_2 的路线距离最大, 被淘汰, 子代由 O_1 和 O_3 组成. 传统交叉算子随机选择交叉位置, 经过交叉操作后往往会破坏其他部分优良的基因结构. 改进的交叉算子可以充分利用父代的基因结构优化子代个体, 由图 1 可见, 子代个体较父代有进一

步的优化.

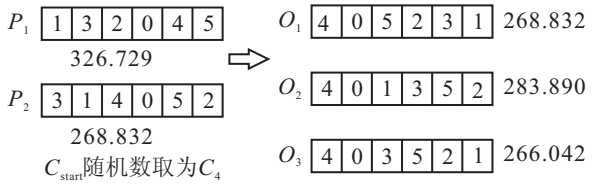

图 1 启发交叉算子示例

3.2.4 精英个体保留策略

在进行选择操作时, 采用精英个体保留策略, 将适应函数值最高的个体直接复制纳入交叉配对的父代群体中. 当所有父代个体执行完交叉算子后, 再次沿用精英个体保留机制, 用交叉前的精英个体替换交叉后群体中适应函数值最差的个体, 剔除质量低的个体, 使得精英个体得以延续.

3.2.5 算法结构

基于上述分析, 本文设计的改进算法如图 2 所示. 基本流程如下:

1) 基于贪婪算法生成初始种群;

2) 每一次迭代过程首先计算种群中个体的适应函数值, 然后依次执行选择、交叉、变异算子.

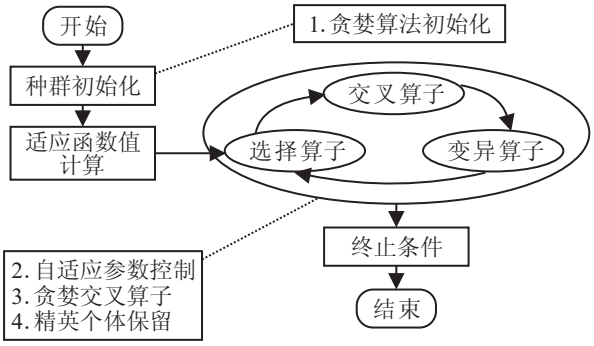

图 2 改进遗传算法流程

选择算子: 根据精英个体保存策略, 将适应值最高的个体纳入待交叉操作的个体中, 对于所有个体, 按照适应函数值比例执行赌轮选择算子, 得到待交叉的父代个体.

交叉算子: 结合当前进化情况动态调整交叉概率大小, 根据交叉概率对父代个体执行贪婪交叉算子, 得到交叉后子代群体. 再次沿用精英个体保留策略, 用交叉前的精英个体替换子代中适应函数值最差的个体.

变异算子: 结合当前进化情况计算当前变异概率大小, 对发生变异的个体执行逆序变异算子.

4 实验分析

基于上述分析, 采用 C# 语言对求解 TSP 问题的改进遗传算法进行设计实现, 并选用 TSPLIB测试库中的几组实例进行模拟分析. 实验环境为: Windows XP 系统, 2G 内存, Microsoft visual studio 平台.

算法伪代码设计如下:

Input: TSP 实例, 种群规模 popsize;

Output: optimal solution.

1: //Intialization Species

2: for each individual

3: randomly assign ccurrent into Species

4: add the next nearest cnext into Species

5: repeat

6: // Calculate Fitness

7: for each individual

8: calculate fitness(i)

9: //Select Operator

10: select the individual with the biggest fitness

11: for each individual i

12: calculate P_i,Q_i

13: for each individual i

14: if random $r > Q[i-1]$ and $r \leqslant Q[i]$

15: select i

16: //Crossover Operator

17: for each individual i

18: calculate p_{ci}

19: if random $r < p_{ci}$

20: add i into parent pairs

21: for each parent pair

22: generate O_3

23: generate child O_1

24: generate child O_2

25: compare O_1 and O_2 and choose the best one

26: replace the worst children with the best individual before crossover

27: //Mutationover Operator

28: for each individual i

29: calculate p_{mi}

30: if random $r < p_{mi}$

31: reverse the genes of individual from site j to k

32: until reach the max generation

表 2 改进策略对比结果

方法	执行 0.2 s 后的结果
传统	825.47
4.1	508.89
4.1+4.2.1	482.53
4.1+4.2.1+4.2.2	474.94
4.1+4.2.1+4.2.2 +4.3	423.91
4.1+4.2.1+4.2.2 +4.3+4.4	423.74

表 3　实验结果对比分析

实例	TSPLIB 提供的最优解	文献 [2] 最优解	传统遗传算法		改进遗传算法		
			最优解	时间/s	最优解	时间/s	优化率
berlin52	7 542	7 542	8 228	4.468	7 509	3.858	0.004 3
dantzig42	699	699	684	1.171	676	0.467	0.032 9
eil51	426	426	442	5.625	412	5.056	0.032 8
eil76	538	-	552	10.781	527	10.788	0.020 4
eil101	629	-	661	7.745	617	12.297	0.019 0
lin105	14 379	-	37 445	9.522	14 362	6.809	0.001 1
st70	675	675	700	4.467	658	2.721	0.025 1
pr76	108 159	-	119 315	4.381	107 873	10.577	0.002 6
pr107	44 303	44 302	47 954	14.599	44 278	49.381	0.000 5
rat99	1 211	-	1 291	3.410	1 201	6.031	0.008 2
rat195	2 323	-	2 482	20.578	2 279	19.862	0.018 9
tsp255	3 916	3 919	4 246	62.881	3 839	56.493	0.013 5
Oliver30	423.74	423.74	472.22	1.187	423.74	0.065	0

对 Oliver 30 问题进行实验, 在相同的运行时间 (0.2 s) 内, 分别得出传统方法和执行各项改进策略后的结果, 如表 2 所示. 由表 2 可见, 在相同的时间内, 改进的遗传算法可以在较短时间内更好地收敛到最优解.

下面选用国际上通用的 TSPLIB 测试库中的几组实例进行模拟分析. 设置种群数量为 20, 传统遗传算法的交叉概率为 0.95, 变异概率 0.005. 实验结果如表 3 所示.

由表 3 可见: 在种群规模较小的情况下, 改进的遗传算法获得最优解所需要的运行时间较短, 最优结果较文献 [2] 的结果有所提高; 传统遗传算法虽然在短时间内可以找到解决方案, 但容易陷入局部最优 (如 pr76). 通过程序的仿真模拟, 得到改进的遗传算法解决部分实例的最优路线如图 3 ～图 9 所示.

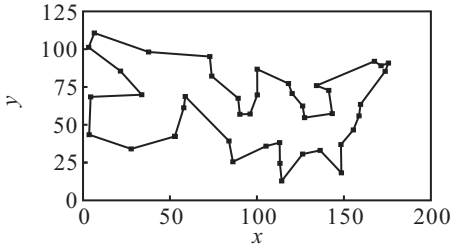

图 3　dantzig42 最优路线图

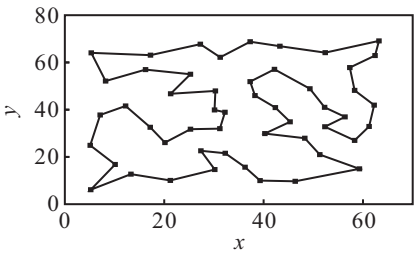

图 4　eil51 最优路线图

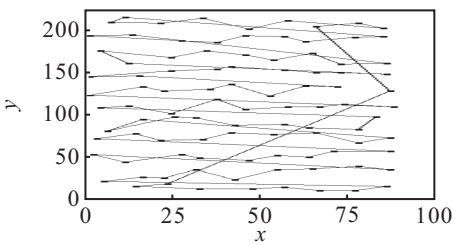

图 5　rat99 最优路线图

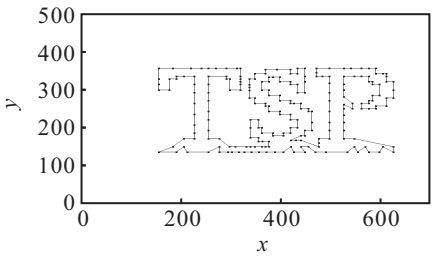

图 6　tsp255 最优路线和进化对比曲线

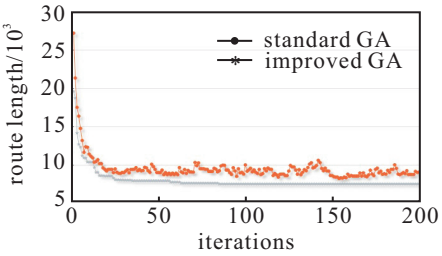

图 7　berlin52 传统和改进算法的进化曲线

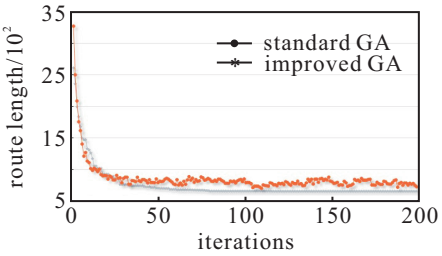

图 8　st70 传统和改进算法的进化曲线

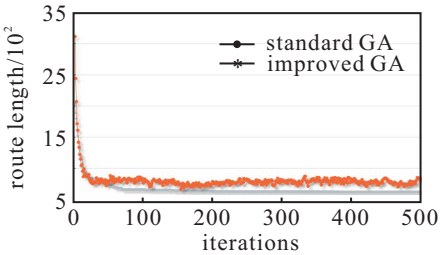

图 9　eil101 传统和改进算法的进化曲线

由图3～图9可见, 引入贪婪算法生成初始种群, 能加快进化初期寻优速度. 通过实验数据发现, 改进的遗传算法可以在更短更少的时间内找到最优解.

5 结 论

本文设计了一种改进的解决TSP问题的遗传算法, 引入贪婪算法初始化种群和实现启发式交叉算子, 利用其局部寻优能力改善种群质量, 加快寻优速度. 同时对遗传过程的两个重要参数(交叉概率和变异概率)进行自适应调节, 防止解集陷入局部最优. 实验对比结果表明, 改进算法在解决规模较小的TSP问题上具有较好的寻优效果, 当种群规模增大时, 运行时间较长, 在以后的研究中应加以改进.

参考文献(References)

[1] 高海昌, 冯博琴, 朱利. 智能优化算法求解TSP问题[J]. 控制与决策, 2006, 21(3): 241-247.
(Gao H C, Feng B Q, Zhu L. Reviews of the meta-heuristic algorithm for TSP[J]. Control and Decision, 2006, 21(3): 241-247.)

[2] 冀俊忠, 黄振, 刘椿年. 一种快速求解旅行商问题的蚁群算法[J]. 计算机研究与发展, 2009, 46(6): 968-978.
(Ji J Z, Huang Z, Liu C N. A fast ant colony optimization algorithm for traveling salesman problems[J]. J of Computer Research and Development, 2009, 46(6): 968-978.)

[3] 刘朝华, 张英杰, 章兢, 等. 蚁群算法与免疫算法的融合及其在TSP中的应用[J]. 控制与决策, 2010, 25(5): 695-700.
(Liu Z H, Zhang Y J, Zhang J, et al. Using combination of ant algorithm and immune algorithm to solve TSP[J]. Control and Decision, 2010, 25(5): 695-700.)

[4] Yannis Marinakis, Magdalene Marinaki. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem[J]. Computers and Operations Research, 2010, 37(3): 432-442.

[5] Darrell Whitley, Doug Hains, Adele Howe. A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover[C]. Proc of the 11th Int Conf on Parallel Problem Solving from Nature. Berlin: Springer Heidelberg, 2010, 6283: 566-575.

[6] Zakir H Ahmed. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator[J]. Int J of Biometrics and Bioinformatics, 2010, 3(6): 96-105.

[7] Murat Albayrak, Novruz Allahverdi. Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms[J]. Expert Systems with Applications, 2011, 38(3): 1313-1320.

[8] Baidu Baike. Traveling saleman problem[EB/OL]. (2014-3-17). http://baike.baidu.com/view/1162183.htm.

[9] Lin W, Delgadofiras Y G, Gause D C, et al. Hybrid Newton-Raphson genetic algorithm for the travelling salesman problem[J]. Cybernetics and Systems, 1995, 26(4): 387-412.

[10] 任子武, 伞冶. 自适应遗传算法的改进及其在系统辨识中应用研究[J]. 系统仿真学报, 2006, 18(1): 41-43.
(Ren Z W, San Y. Improved adaptive genetic algorithm and its application research in parameter identification[J]. J of System Simulation, 2006, 18(1): 41-43.)

[11] 沐阿华, 周绍磊, 于晓丽. 一种快速自适应遗传算法及其仿真研究[J]. 系统仿真学报, 2004, 10(1): 122-125.
(Mu A H, Zhou S L, Yu X L. Research on fast self-adaptive genetic algorithm and its simulation[J]. J of System Simulation, 2004, 10(1): 122-125.)

[12] 邓先习. 遗传算法求解TSP问题的研究与改进[D]. 沈阳: 东北大学信息科学与工程学院, 2008.
(Deng X X. Research and improvement on genetic algorithm for solving TSP[D]. Shenyang: College of Information Science and Engineering, Northeastern University, 2008.)

[13] Srinivas M, Patnail K L M. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Trans on System, Man and Cybernetics, 1994, 24(4): 656-667.

[14] 任海艳, 陈飞翔. 自适应遗传算法的改进及在曲线化简中的应用[J]. 计算机工程与应用, 2012, 48(11): 152-155.
(Ren H Y, Chen F X. Improvement of adaptive genetic algorithms and application in line simplification[J]. Computer Engineering and Application, 2012, 48(11): 152-155.)

[15] 向佐勇, 刘正才, 申平安. 一种改进的基于进化阶段的自适应遗传算法[J]. 武汉大学学报: 工学版, 2008, 41(1): 133-136.
(Xiang Z Y, Liu Z C, Shen P A. An improved adaptive genetic algorithm based on evolutionary stages[J]. J of Wuhan University: Engineering, 2008, 41(1): 133-136.)

[16] 彭丹平, 林志毅, 王江晴. 求解TSP的一种改进遗传算法[J]. 计算机工程与应用, 2006, 42(13): 91-93.
(Peng D P, Lin Z Y, Wang J Q. An improved genetic algorithm for TSP problem[J]. Computer Engineering and Applications, 2006, 42(13): 91-93.)

(责任编辑: 郑晓蕾)