第26卷 第3期 华侨大学学报(自然科学版) Vol.26 No.3
2005年7月 Journal of Huaqiao University (Natural Science) Jul.2005

文章编号 1000-5013(2005)03-0231-04

TSP问题的一种快速近似算法及应用

宋 海 洲

(华侨大学数学系,福建 泉州 362021)

摘要 给出求解度约束最小生成树(DCMST)问题的一种快速近似算法.在此基础上,又给出求解TSP问题的一种快速近似算法,并在微机上实现且其数值试验的效果良好.最后,将求解TSP问题的近似快速算法作一些改进,应用于遗传算法的初始种群生成并进行数值实验.结果表明,用文中算法生成的初始种群,比起一般方法产生的初始种群性能有很大改进.该算法可以加速遗传算法的寻优速度.

关键词 TSP,近似算法,遗传算法,初始种群

中图分类号 TP 301.6;O 141.4 **文献标识码** A

TSP问题[1]可以简述有 n 个城市 $1,2,3,\cdots,n$,一旅行商从某个城市出发,环游所有城市回到原出发地.每个城市只能经过一次,要求找出一条最短的路线.这个问题可以说是一个典型的组合优化问题.TSP问题求解,一直以来倍受人们的关注.求解TSP问题的方法,主要有贪婪法、分支定界法、动态规划法、最近邻试探法、模拟退火法和遗传算法[2].贪婪法、分支定界法和动态规划法这3种方法,它们的优点是在 n 较小时一定可以求得最优解,缺点是在 n 较大时因计算量太大而无法实现.最近邻试探法的优点是求解速度最快,但缺点是求解精度最差.模拟退火法和遗传算法的优点是求解精度相对较高,但缺点是求解速度相对较慢.本文提出一种启发式快速近似算法,其计算时间复杂度为 $O(n^2\log n^2)$,因而求解速度较快.经大量上机试算,获得满意的效果,特别在求解欧氏平面上的TSP问题上,该算法具有较好的性能.

1 TSP问题和度约束最小生成树问题

度约束最小生成树问题(Degree-Constrained Mininum Spanning Tree,简称DCMST问题)[3],其组合含义是从图中所有的生成树中(总数最多可达 n^{n-2}),找出顶点度符合约束且总权数最小的生成树.其求解难度,随各顶点的度约束的不同而不同.

设 $G=(V,E,W)$ 为赋权的无向图,$V=\{1,2,\cdots,n\}$ 为顶点集,E 为边集,$W=(w_{ij})_{n\times n}$ 为权矩阵,且

$$w_{ij}=w_{ji},\qquad w_{ii}=+\infty,$$

并设各顶点的度约束为 $b_i(i=1,\cdots,n)$.那么,DCMST问题的数学模型[4]为

$$\min z=\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_{ij}. \tag{1}$$

它受约束于

$$\left.\begin{aligned}&\sum_{i=1}^{n}\sum_{j=1}^{n}x_{ij}=n-1,\\&\sum_{i,j\in S}x_{ij}\leqslant|S|-1,\qquad\forall S\subset V,\quad S\neq\varnothing,\end{aligned}\right\} \tag{2}$$

$$\sum_{j=1}^{n}x_{ij}\leq b_i,\qquad i\in V. \tag{3}$$

这里 $x_{ij}\in\{0,1\}$,$i,j\in V$,变量 $x_{ij}=\begin{cases}1, & \text{边}(i,j)\text{在最优树中}\\0, & \text{其他}\end{cases}$.约束(2)保证所求得的为一棵生成树,约束

收稿日期 2004-09-17

作者简介 宋海洲(1971-),男,副教授,主要从事数学模型的研究.E-mail:hzsong@hqu.edu.cn

(3)为度约束. 于是当 $b_i = 2$,度约束最小生成树问题即转化为 TSP 问题. 因此求解 TSP 问题,可以转化为求解度约束最小生成树问题.

2 求解 DCMST 问题的一种快速近似算法

2.1 实现步骤

下面我们先给出求解模型(1)的一个快速近似算法,其核心是在不违反度约束和不形成圈的前提下,每次加入权最小的边,直至加入 $n-1$ 条边为止. 具体算法步骤有 6 点. (1) 生成图 G的边权矩阵 B (B 的第 i 列表示图 G 的第 i 条边,B 的第 i 列的前两行储存第 i 条边关联的顶点编号,第 3 行储存第 i 条边的权). 对边权矩阵 B 按照边的权进行从小到大排序,设排序后的矩阵为 BW. (2) 由各顶点的度约束 $b_i(i=1,\cdots,n)$,生成各顶点的初始度检查值 d_i, $d_i=b_i(i=1,2,\cdots,n)$. (3) 将 BW中具有最小权的边(为 BW的第一列)及该边所关联的顶点(不妨设该边所关联的顶点 i_1 为 j_1 和)加入到图 T中,修改这两个顶点的度检查值. 将 i_1 和 j_1 对应的度检查值减去 1(即 $d_{i_1}=d_{i_1}-1$, $d_{j_1}=d_{j_1}-1$),并从 BW中删除该边更新 BW. 即可以令原 BW的第 1 列为空列 BW(: ,1) = [ ],BW的原第 2 列就变为新 BW的第 1 列了. (4) 检查 BW的第 1 列对应的边是否可以用. 如果该边所关联的顶点的度检查值都大于 0,而且将该边加入到图 T后,图不会形成圈,则 BW的第 1 列对应的边可以用;否则,该边不可以用,从 BW中删除该边,更新 BW,直到 BW的第 1 列对应的边可以用为止. 转入(5). (5) 检查图 T 中边的数目是否小于 $n-1$,如果图 T中边的数目不小于 $n-1$,则已经找到一棵近似度约束最小生成树,转(6);否则,将 BW中具有最小权的边(为 BW的第一列)及该边所关联的顶点加入到图 T中,并将该边所关联的顶点(不妨设该边所关联的顶点为 i_k 和 j_k),所对应的度检查值减去 1(即 $d_{i_k}=d_{i_k}-1$, $d_{i_k}=d_{i_k}-1$). 同时,从 BW中删除该边更新 BW, (即可以令原 BW的第 1 列为空列 BW(: ,1) = [ ],BW的原第 2 列就变为新 BW的第 1 列,转(4). (6) 计算图 T的总权重,输出图 T及总权重. 从上述算法可知,如果将上述算法用于求解无约束情况下一般的最小生成树问题(MST),则上述快速近似算法就是 Kruskal 算法. 因此上述算法是 Kruskal 算法的推广,此时一定可以得到理论最优解.

2.2 快速算法的计算时间复杂度及有效性

计算时间复杂度分析. 由于计算量主要集中在对边权矩阵 B 按照边的权进行排序上,对 k 个数进行排序的计算量为 $O(k\log k)$,而边权矩阵 B 的最大边数为 $n(n-1)/2$. 于是,排序的计算时间复杂度为 $O(n^2\log n^2)$,则近似算法的计算时间复杂度为 $O(n^2\log n^2)$. 下面,我们对算法进行有效性分析,首先有如下的引理.

引理 1 设一个图有 n 个顶点和 m 条边,如果该图是连通的,则 $n\leq m+1$.

定理 此即算法的有效性定理. 如果模型(1)中各顶点的度约束为 $b_i\geq 2(i=1,\cdots,n)$,则快速近似算法一定有效(即最后一定产生满足度约束条件的生成树).

证明 要证明快速近似算法有效,必须确认 6 件事情:(1) 不能因为某一步无法完成而“搁置”(卡住或阻塞);(2) 能够终止(能结束而不是永远走下去);(3) 输出一个图;(4) 输出一个树(一个连通图,不允许有圈);(5) 输出一个生成树(包含图 G所有的顶点);(6) 输出图的顶点满足度约束. (2)～(5)是容易验证的,下面证明本算法对(1)也是成立的.

假设某一步,会导致 BW是空矩阵,那么这一算法就会搁置,这种情况不会发生. 因为导致 BW是空矩阵,原因有如下两种情形. (1) 至多一个顶点的度检查值大于 0,此时图 T一定至少含 $n-1$ 条边,那么我们的算法已经完成. (2) 至少有两个顶点的度检查值大于 0,但是这些顶点中任意两个,将它们关联的边加入到图 T后形成的新图 T_1,都会形成圈. 那么,此时图 T必具有以下两个性质. (1) 图 T必是连通的. 否则,可以在图 T的两个不相连通子树中,各选一个叶顶点,将他们相关联的边,加入到图 T中,形成的新图 T_1,则 T_1 不会形成圈. (2) 此时图含的顶点数为 n,即图 T含有图 G的所有顶点. 否则,可以将不在图 T的顶点,与另一个度检查值大于 0 的顶点,它们相关联的边,加入到图 T中,形成的新图 T_1, T_1 不会形成圈. 因此,利用引理 1 可知,此时图 T所包含的边的数目大于或等于 $n-1$. 当发生这种情况时,我们的算法必定已经完成. 因此,我们的算法不会因为某一步无法完成而“搁置”. 因此,我们输出图 T是模型(1)的满足约束条件(2),(3)的有效解. 从而,我们的快速近似算法总是有效.

3　求解 TSP 问题的一种快速近似算法

首先置各顶点的度限制为 2. 这时,我们就可用前述算法求得一线性树. 最后,将该线性树的两个叶顶点相连,就得原 TSP 问题的近似最优解. 该算法的计算时间复杂度为 $O(n^2\log n^2)$. 以上算法,均用 Matlab 编程实现.

4　数值实验

实例 1[1]　对于 CHN144 实例($n = 144$),求其 TSP 近似最短回路. 对于 CHN144 实例,该实例对应的最小生成树长为 26 126.129 2,最近两个城市的距离为 37. 因此,CHN144 实例的最优 TSP 回路长度的一个下界为 26 163.129 2. 如果随机地产生一个 TSP 回路,则该随机的 TSP 回路长度一般在 210 000 左右,而该实例目前用遗传算法得到的最佳近似最优长度为 30 353.86. 利用本文的快速近似算法,得到近似最优回路长度为 37 671.483 3. 快速近似算法得到近似最优回路长度,与目前用遗传算法得到的最佳近似最优长度比为 1.160 2,但快速近似算法仅费时 19.656 0 s. 该时间是用 Matlab 编程运行所需时间,如果用 C 语言编程则费时(t)会更少.

为了检验算法的性能,下面的数值实验是先产生 n 个点的坐标,每个点的坐标分量都是[0,100]内的伪随机数,再由此生成权矩阵 $W = (w_{ij})_{n\times n}$. 算法性能指标 r 定义为 $r = (TSP_{kuaishu}/W_{MST} + \min w_{ij})$,其中 $TSP_{kuaishu}$ 为近似算法求得的最优 TSP 回路长,W_{MST} 为这 n 个顶点的最小生成树长度,$W_{MST} + \min w_{ij}$ 为这 n 个顶点的 TSP 回路长度一个下界. 这里,r_{min},r_{mean} 和 r_{max} 分别为各算例的算法性能指标 r 的最小值、均值和最大值(样本数 = 10). 实验结果如表 1 所示.

表 1　实验结果

n	r_{min}	r_{mean}	r_{max}	t/s	n	r_{min}	r_{mean}	r_{max}	t/s
20	1.257 8	1.421 6	1.605 2	0.03	100	1.237 6	1.363 9	1.503 1	3.80
40	1.236 0	1.333 1	1.447 0	0.18	120	1.290 7	1.366 7	1.470 8	7.50
60	1.259 9	1.405 4	1.539 9	0.70	140	1.275 6	1.331 4	1.401 2	13.60
70	1.269 5	1.388 5	1.473 4	1.20	160	1.255 1	1.324 1	1.410 0	21.70
80	1.264 0	1.340 3	1.508 9	1.80	180	1.309 3	1.348 1	1.425 9	35.00

从上述结果可看出,快速近似算法求得的 n 个顶点的近似最优 TSP 回路长 $TSP_{kuaishu}$ 与这 n 个顶点的 TSP 回路长度一个下界之比的均值 r_{mean} 在 1.4 左右. 这说明该算法的性能不错.

5　快速近似算法用于遗传算法的初始种群生成

用遗传算法求解 TSP 可得相对较好的结果,并已有许多有效的编码方案及相应的遗传算子. 但无论哪一种方案,只要其在原始解空间中搜索、寻优,就必会很耗时间. 我们将求解 TSP 问题的近似快速算法作改进,应用于遗传算法的初始种群生成. 这样可使遗传算法的寻优速度显著加快.

改进的近似快速算法的思想,是在不违反度约束和不形成圈的前提下,在剩余可用的边中选取具有权最小的前 d 条边,随机地挑选一条,进行度约束值检查和是否会形成圈的检查. 如果检查通不过,则将选中的边从可用的边中删除;否则,将选中的边加入 TSP 回路中,直至加入 $n-1$ 条边为止,这样就可得到一线性树. 最后,将树的两个叶子相连,就得到一 TSP 回路.

采用每次加权较小的边,可以使得回路质量比较好. 而从权最小的前 d 条边中随机地挑选一条边加入,形成回路的方法,则可以保证初始种群的多样性.

实例 2[1]　对于 CHN144 实例($n = 144$),比较用随机产生的初始种群与本文算法产生的初始种群的质量. 下面让我们来求解这个问题. 用随机产生 TSP 回路的方法,可以得到初始种群的最小 TSP 回路长度 l_1、平均 TSP 回路长度 l_2 和最大 TSP 回路长度 l_3,如表 2 所示. 运行本文的近似快速算法,得到初始种群的最小 TSP 回路长度 l_1、平均 TSP 回路长度 l_2 和最大 TSP 回路长度 l_3,如表 3 所示. 表中 m 表示种群数.

从实验结果可以看出,我们采用本文算法所生成的初始种群,要比用一般方法产生的初始种群,在

性能上有很大改进.

表 2 用随机产生 TSP 回路的方法得到初始种群的最小 TSP 回路长度和
平均 TSP 回路长度以及最大 TSP 回路长度

m	l_1	l_2	l_3	m	l_1	l_2	l_3
100	197 420.76	217 228.53	234 328.93	500	193 895.49	216 573.61	237 282.42
200	196 872.52	216 101.62	235 851.59	1 000	192 639.96	216 982.49	236 205.60

表 3 用快速近似算法产生的初始种群的最小 TSP 回路长度和
平均 TSP 回路长度和最大 TSP 回路长度

m	d	l_1	l_2	l_3	m	d	l_1	l_2	l_3
10	8	36 019.86	36 903.60	37 634.76	10	30	33 960.14	36 607.49	40 659.55
10	12	35 352.53	36 831.67	38 279.80	100	10	34 968.32	36 899.94	38 829.75

实例 3[1] 对于 CHN144 实例($n = 144$) ,用本文算法产生的初始种群,进行单个个体的单亲演化. 我们现来解此问题. 先用快速近似算法产生包含 10 个回路的初始种群,得到的 TSP 回路平均长度为 36 903.60. 然后再将该初始种群中的每个个体,各进行一次单亲遗传算法,得到的 TSP 回路平均长度为 31 597.92. 其中一个较佳的 TSP 回路长度为 30 631.18,而目前 CHN144 实例的最佳的 TSP 回路长度为 30 353.86.

6 结束语

通过大量的数值试验表明,本文的快速近似算法是求解 TSP 问题是一种颇为有效的算法. 尤其是当问题的规模较大时,它更是为实际提供了一种简便、快速的求解手段. 而在遗传算法中,利用该快速近似算法产生的初始种群,其性能也有很大的改进.

参 考 文 献

1 康立山,谢 云,尤矢勇等. 非数值并行算法 ——模拟退火法[M]. 北京:科学出版社,2000. 149～153
2 宋海洲. 生产函数中参数方法的进一步改进[J]. 华侨大学学报(自然科学版),2005,26(1):23～26
3 顾立尧. 带有度约束的最小耗费生成树的分支限界算法[J]. 计算机应用与软件,1989,6(6):49～54
4 马 良,蒋 馥. 度约束最小生成树的快速算法[J]. 运筹与管理,1998,7(1):1～5

A Fast and Approximate Algorithm For Solving Travelling Salesman Problem and Its Application

Song Haizhou
(Department of Mathematics, Huaqiao University, 362021, Quanzhou, China)

Abstract A fast and approximate algorithm is given at first for solving the problem of minimum spanning tree with constraint of degree. On this basis, a fast and approximate algorithm is then given for solving travelling salesman problem (TSP); and is implemented on a micro-computer, with a good effect as shown by enormous numerical tests. This fast and approximate algorithm for solving TSP is applied finally to generating initial population of genetic algorith after a little modification; and the improved algorithm is tested numerically. As indicated by experimental results, the initial population generated by this algorithm inproved greatly as compared with the performance of the initial population generated by the conventional method; and furthermore, this algorithm will accelerate the optimum search of genetic algorithm.

Keywords travelling salesman problem, approximate algorithm, generate algorithm, initial population