LKH算法的理论梳理

目录

背景

旅行推销员问题（Travelling salesman problem, TSP）是这样一个问题：**给定一系列城市和每对城市之间的距离，求解访问每一座城市一次并回到起始城市的最短回路**。 它是组合优化中的一个NP难问题，在运筹学和理论计算机科学中非常重要。

Lin-Kernighan-Helsgaun（*LKH*）*算法*是解决TSP问题的最先进的搜索算法之一。

本文初步介绍LKH算法的理论框架，试图基于一些关键技术点梳理并帮助阅读者了解掌握LKH算法。

原论文链接：https://www.sciencedirect.com/science/article/abs/pii/S0377221799002842

Lin-Kernighan 启发式算法

一． TSP问题描述

在平面已知位置的n个点中，从某点出发寻找一条最短的路径满足：

1. 必须经过所有n个点。

2. 路径上的点有且仅出现一次。

3. 路线是一条环路，从某点出发最终必须回到起点。即该路径是回路。

二． 朴素的求解思想

1.贪婪求解：从起点开始，下一个点为与当前点距离最短的点(不包括已选中的点)，依次类推，直至包含所有的点，再回到起点。缺点很显然。

2.改进思路一：基于边交换的路径优化

三． 边交换

1.给定一条路径，尝试交换一些边，若交换后的路径的长度比原路径短，则说明通过此次边交换找到更优的路径。

2.（2-交换）：

（1）找出路径中任意两条不相邻的边（n1,n2）,(n3,n4)，构造与之前不同的路径:(n1,n3),(n2,n4)或（n1,n4）(n2,n3)。

（2）若满足Cost_ori - Cost_new > 0 即 dist(n1,n2)+dist(n3,n4)-(dist(n1,n3)+dist(n2,n4))>0 或者dist(n1,n2)+dist(n3,n4)-(dist(n1,n4)+dist(n2,n3))>0 ，则删除(n1,n2),(n3,n4)，使用(n1,n3),(n2,n4)或（n1,n4）(n2,n3)构建新的路径。

（3）时间复杂度O(n^2), 即在n条边中任选两条不相邻的边所花费的时间。

3.（3-交换）：与2-交换类似。共有7种交换类型，包含3种二边交换和4种三边交换。时间复杂度（O(n^3)）

4.以此类推，k-交换，直至n-交换。但k越大，对应的时间复杂度越高，从O（n^k）到O（n^n)。

四． 找出如何获取合适的k—LK算法。

基本思路：

- a. 维护两个边的集合：remove集（待删除边集合，简称RSet）和add集（带加入的集合, 简称A_Set）。
- b. 算法简要流程：
  - i. 选择起点n1.
  - ii. 选n1的前驱或者后继节点为n2，组成第一条待删除的边(n1,n2)进入R_Set.
  - iii. 选n2的邻近节点n3（要求(n2,n3):

*不属于原路径上的边。

*不在R*Set,*与A_Set中。

则，(n2,n3)进入A_Set。

3.从n3的前驱或者后继中选择节点n4，(n3,n4)进入R_Set.

4.**若n4和n1连接，即(n1,n2) (n3,n4)-> (n1,n4)(n2,n3) ，能形成一条路径，且使得dist(n1,n2)+ dist(n3,n4)> dist(n1,n4)+dist(n2,n3) ，则得到一条新的路径。**

5.否则，(n1,n4)进入A_Set。从n4出发，按照2，3，4的步骤重新找待删除的边和待添加边。这里，由于寻找越多，计算复杂度越大，通常当R_Set的大小超过5，退出搜索，表明从n1出发找不到更合适的路径。

（这种方法保证了所有点的度保持为2，即在初始解是单环的情况下，保证了单环的存在）

假设x1,x2,...,xi 为需要删除的边，其集合为X; y1,y2,...,yi为需新增加的边，其集合为Y.

(1) xi与yi要有共同的端点，yi与xi+1也要有共同的端点，即xi与yi交替出现，最终形成：

> *(x1, y1, x2, y2,... , xi, yi)*

(2)X与Y没有公共边

(3)X的总长度小于Y的总长度,

(4)经过交换后的tour仍然为一个有效的tour

令**修正边链路S**=[v1,v2,v3,v4,v5...,vx]，使得

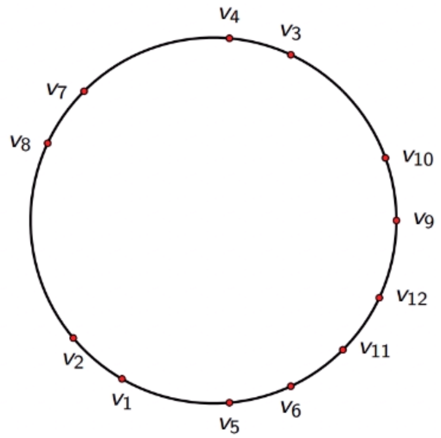
((V1 , V2),(V2 , V3),(V3 , V4),(V4 , V5)...(Vx,V0)) , x = 2*i , i∈N

其中(V2*i+1 , V2*i+2) 为摧毁边集合R_set，**i∈N**

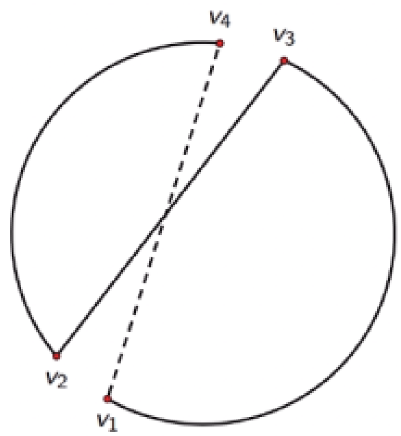
(V2*i+2 , V2*i+3) 为新增边集合A_set，**i∈N**

另，A_set中还有一个元素**(Vx,V0)**

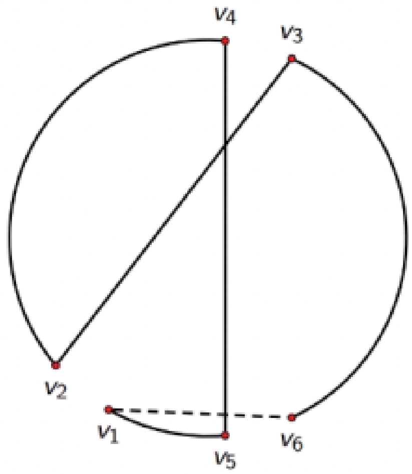
示意图



初始状态如下

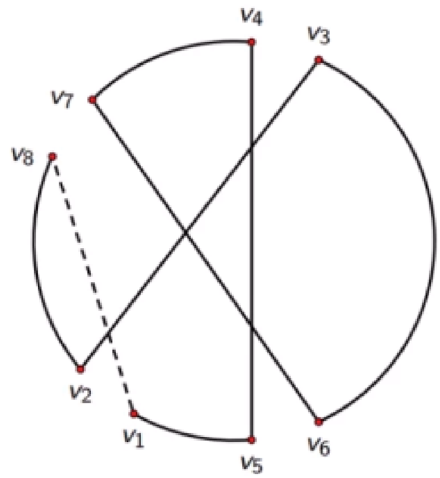


令修正边链路S = [v1,v2,v3,v4]，则有

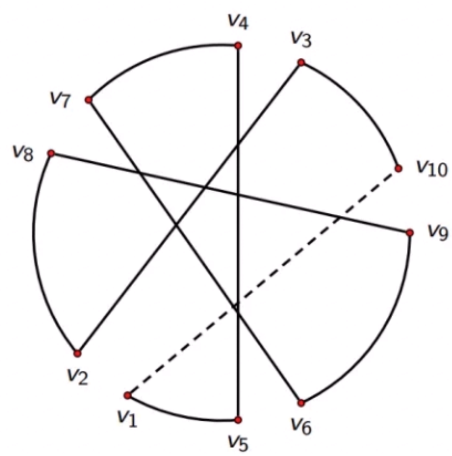


令修正边链路S = [v1,v2,v3,v4,v5,v6]，则有

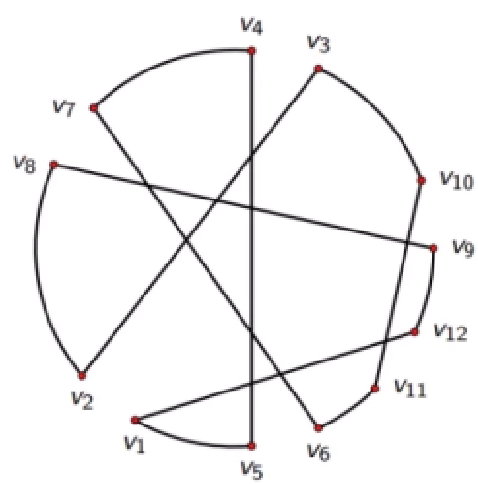
数学推导



令修正边链路S = [v1,v2,v3,v4,v5,v6,v7,v8]，则有



令修正边链路S = [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10]，则有



令修正边链路S = [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12]，则有

由此搜索解空间交换边

Keld Helsgaun 改进

数学推导

注意到，在选择边加入D_set时很容易，因为对于一个确定的点v来说，最多只有两条边符合条件（加入D_set的边必须是原存在的实边）。但是选择加入A_set的边就比较困难，因为理论上可形成的虚拟边非常多。

那么，在选择加入A_set的虚拟边时，就会有一个选择的问题，受限于计算时间，不能遍历所有的可能性，所以每个点都会维护一个candidate集合，在该点被选为虚拟边的起点的时候，终点从candidate集合中顺序选取。

一般来说，为使虚拟边尽可能短（因为最后要删D_set并添加A_set，要使得总里程缩短才能产生一个有效解），因此一般candidate集合中存储的点，是按远近距离升序存储的。

但是，Keld Helsgaun 指出，这样生成candidate是有问题的，他做了大量统计，发现在最优解中，大部分边的终点并不是起点的较为邻近的点，在一个587个city的tsp问题中，最优解中有一条边的终点是起点的第22邻近点。 也就是说，局部贪心在tsp问题中效果并不好。

Keld Helsgaun 同时发现，最优解75%以上的边，都满足一个特征：它们都是最大1-tree上的边。

1-tree

1-tree是Keld Helsgaun 提出的新概念，1-tree是图的最小生成树，加上一条任意两个叶子节点（度为1）产生的新边生成的树。而最大1-tree，也即所有边总和最大的1-tree。

事实上，Keld Helsgaun的目标是找到最大1-tree恰好是一个单环，因此，他要对所有的边都添加一个修正量bias（这意味着可能有边长会变成负数，但暂时没有关系），使得其产生的最大1-tree恰好是一个单环。

这就是拉格朗日算法梯度求解bias的问题了。

$$f(x) = min \sum_{i,j} C_{ij} * X_{ij}$$

目标函数

s.t. $$\sum_{i} X_{ij} = 1$$ ∀ j

s.t. $$\sum_{j} X_{ij} = 1$$ ∀i xi,j ∈{0,1}

可得拉格朗日函数：F = $$\sum_{i,j} C_{ij} * X_{ij} + \sum_{i} \pi_i * (\sum_{j} X_{ij} - 1) + \sum_{j} \pi_j * (\sum_{x} X_{ij} - 1)$$

$$\sum_{i,j} (C_{ij} - \pi_i - \pi_j) X_{ij} + \sum_{i} \pi_i + \sum_{j} \pi_j$$

也即

prim最小生成树、最小1-tree 和tsp最佳回路

最小生成树提供了一个含义，即为，将点连通成一个簇所需的最小代价。从这个含义出发，开始逐步逼近tsp最佳回路解。

	以总实边长度最小为目标	所有点连通为一个簇	有且仅有n条边	所有点度均保证为2
最小生成树	true	true	false	false
最小1-tree	true	true	true	false
tsp最佳回路	true	true	true	true

最小1-tree是作为tsp最优解的下界使用的，求得1-tree的极大值，便可以逼近最优解。这个结论是直观的。因为 最小1-tree点的度并不一定均为2。

可见，最小1-tree相对于tsp的解，有一个明显的区别就是，1-tree的所有的点度并不一定为2。换句话说，所有的点度都为2度最小1-tree，即为tsp的最优解。

但是如何改变1-tree的度呢？

对每个节点设置pi值，使得每条连接它的边长度计算增加pi(pi可以为负数)。由于连接该点的所有边同时改变了相同的数值，因此tsp最优解将仍旧保证是最优解。但这样会改变1-tree的生成结构，从而改变1-tree点的度。

从最小生成树出发，先是增加一条边构建最小1-tree，满足有且仅有n条边的条件，然后再用梯度下降法(以及次梯度下降法)逼近所有点度均为2。

梯度下降法：

node的pi值根据上一步的V值调整，V值为每一步结束后最小1-tree的度距2的绝对值。比2大，pi值增加，增加1-tree总值成为负担，不再偏向经过这个node，反之亦然。

迭代朝着1-tree各个点度为2 的方向前进。

最小1-tree产生点的candidate集合

论文指出，最小1-tree的边包含许多(80%)与最优路径相同的边，它非常适合作为“接近度”的启发式度量。上文可知，最小1-tree是最优解的下界。可知点j在点i 的candidate集合中的顺序需的使得边(i,j)在最小1-tree中，若不在，需构建。

令T为长度为L(T)的最小1-tree，令T+(i,j)表示包含边(i,j)所需的最小1-tree。将边(i,j)的接近度定义为

$a(i,j) = L(T+(i,j)) - L(T)$

也就是说，给定任何1-tree的长度，一条边的 a-nearness 是当最小1-tree需要包含该边时长度的增加。

因此求得目标1-tree后，对选定from点，candidate可由以下规则产生：

1、增加(from,to)边长。(这样会导致1-tree上出现实环，需要破坏环内一条最长的边)

2.1、如果from或to是原先虚边的一个点，则删该虚边。

2.2、对每个其他的点to，如果to在from的父系路径上，删to到from节点(不经过根结点)最大的实边；如果to不在from的父系路径上，删from到to(经过根结点)的route的最大实边。得到一个新的1-tree。

新的1-tree包含了(i,j)边，且是一个合法的1-tree

3、对每个to得到的新1-tree计算结果（也即 L(T+(i,j)) ），升序排序，取前k个加入from的candidate集合。

至此，求得每个点的candidate集合，用于在Lk算法中替代最近点搜索规则。