HIT CS&E

第十章 On-line Algorithms

胎吉洲
计算机科学与技术学院

HIT CS&E

提要

10.1 *Introduction to On-line Algorithms*
10.2 *On-line Euclidean Spanning Tree Problem*
10.3 *Randomized On-line Algorithm for MST*
10.4 *On-line Algorithm for Convex hull problems*

HIT CS&E

10.1 *Introduction to On-line Algorithms*

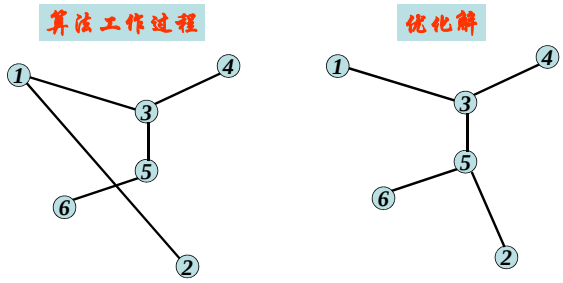
HIT CS&E

- 前九章介绍的算法设计的条件
  - 在算法执行之前整个输入数据的细节都很清楚
  - 问题是在完全了解输入数据信息条件下解决的
- 实际应用存在不满足上述条件的情况
  - 磁盘调度问题
  - 操作系统的页面调度问题
  - Data streams

HIT CS&E

- On-line算法
  - 在算法设计阶段或执行之前无完全信息可用.
  - 输入数据往往是实时到达的.
  - 算法时而正确时而错误.
  - 实时算法难以给出正确解,一般是近似算法.
- 实时最小生成树问题
  - 难以给出优化解,“最小”需要放弃或改为“小”.
  - On-line算法可能的下工作:
    - 当每次一个数据到达时,将其连接到最近的邻居节点.
    - 下边是一个六个实时到达节点的例子.

HIT CS&E



HIT CS&E

- On-line算法的性能
  - 令C_{on}表示On-line算法解的代价
  - 令C_{off}表示Off-line算法解的代价
  - 若$C_{on} \le f(n)C_{off}+c$，c是常数，则称On-line算法的竞争度为 $f(n)$. 这个算法称为$f(n)$-competitive

HIT CS&E

10.2 *On-line Euclidean Spanning Tree Problem*

- 问题的定义
- 实时算法设计
- 算法性能分析

HIT CS&E

问题的定义

- 输入：
  - $S=\{v_1, v_2, \ldots, v_n\}$是平面上的$n$个点的集合
  - 这n个点并非一次给定，是逐个出现的
- 输出
  - 一个"最小"生成树

求解最小生成树问题的On-line算法只能是近似算法

HIT CS&E

On-Line算法

Greedy-On-line-ST(*S*)

1. $n=|S|$;
2. $T=0$;
3. While $n\neq 0$ Do
4. Input(v);
5. 把v与$V(T)$之间最短边加入T; /*$V(T)$是T节点集合*/
6. Endwhile

HIT CS&E

算法的性能分析

- 算法的时间复杂性
  - 3-6步需要的比较次数=1+2+…+(n-1)
  - $T(n)=O(n^2)$
- 解的精确度
  - Greedy-On-line-ST算法的近似比是$O(\log n)$

设T_{onl}是算法产生的生成树，
l是优化生成树的代价或长度，
下边我们来证明这个结论

引理1. T_{onl}中第k长的边的长度最多为$2l/k$, $1\le k\le n-1$.

证.

令$S_k=\{v \mid v$加入T_{onl}后，T_{onl}中出现长度大于$2l/k$的边$\}$.

如果$|S_k|<k$, 则T_{onl}中至多k-1条边的长度大于$2l/k$, 即T_{onl}中第k最长边的长度最多为$2l/k$.

往证$|S_k|<k$.

由算法的Greedy方法可知，S_k中各个点对之间的距离必大于$2l/k$.

若不然，设u和v之间距离小于$2l/k$, 则加入u(或v)以后，再加入v(或u)时，不会产生大于$2l/k$的边.

于是，S_k上TSP的优化解的长度大于$|S_k|2l/k$.

由于任意点集上的TSP优化解的长度是其最小生成树长度的2倍，S_k的最小生成树的长度大于$|S_k|l/k$.

因为S_k上最小生成树长度l'不大于S上最小生成树长度，我们有，$|S_k|l/k < l' \le l$，即$|S_k|l/k < l$，或 $|S_k| < k$.

于是，T_{onl}中第k最长边的长度最多为$2l/k$.

定理1. Greedy-On-line-ST算法的近似比是$O(\log n)$.

证. 由引理1，T_{onl}的长度$L \le \sum_{1\le k\le n-1} 2l/k$

$=2l \sum_{1\le k\le n-1} 1/k$

$=2l \times H(n-1)=O(\log n)$.

于是，算法的近似比为$L/l \le O(\log n)$.

10.3 Randomized On-line Algorithm for MST

- 问题的定义
- 随机On-line算法
- 算法的性能分析

问题的定义

- 输入
  - Euclidean空间上的n个点：$v_1, v_2, \ldots, v_n$
  - n个逐个出现
- 输出
  - 由$v_1, v_2, \ldots, v_n$构成的最小生成树

随机On-line算法

Algoritm $R(m)$

T=0; $input(m)$; $input(v_1)$; $input(v_2)$;

把边(v_1, v_2)添加到T;

For k=3 **To** n **Do**

  $input(v_k)$;

  If $k \le m+1$

  Then 从$(v_k, v_1), \ldots, (v_k, v_{k-1})$中选择最小边加入$T$;

  Else 从$(v_k, v_1), \ldots, (v_k, v_{k-1})$中随机地选择$m$条边，把这$m$条边中最小边加入$T$;

Return T.

$R(n-1)$是一个确定的贪心算法

算法的时间复杂性

- 时间复杂性
  - 每次考察需要$O(m)$时间
  - 需要$O(n)$次考察
  - $T(n)=O(nm)$

近似解的精确度

定义1. 随机On-line算法A称为c-competitive，如果存在一个常数b，使得

$$E(C_A(\sigma)) \le c \times C_{opt}(\sigma)+b$$

其中，$E(C_A(\sigma))$是算法A在问题实例σ上产生的近似解的代价，$C_{opt}(\sigma)$是σ的优化解的代价.

定理1. 如果m是固定常数，算法$R(m)$的competitive，则比是$\Theta(n)$.

证.

我们首先定义一组记号：

$\sigma=\{1, 2, \ldots, n\}$ 输入点集合，输入顺序为1、2、…、n.

$D(a, b)$ = 点a和b之间的距离.

$N(a, k)$ = 在已经出现的点中点a的第k最近点.

$E(L_{R(m)}(\sigma))$ = 算法$R(m)$产生的树的期望长度.

假定算法收到第i个输入点i.

如果$2\le i\le m+1$，则i与前i-1个点中最近点连接，加入树中.

如果$m+1\le i\le n$，则i与前i-1个点中第j最近点连接的概率为

$$\binom{i-1-j}{m-1}\Big/\binom{i-1}{m} \qquad 1\le j\le i-m$$

于是，我们有

$$E(L_{R(m)}(\sigma))=\sum_{i=2}^{m+1}D(i,N(i,1))+\sum_{i=m+2}^{n}\sum_{j=1}^{i-m}\binom{i-1-j}{m-1}\Big/\binom{i-1}{m}\times D(i,N(i,j)) \qquad (1)$$

我们来计算$E(L_{R(m)}(\sigma))/L(\sigma)$的下界.

$L(\alpha)$是输入点集合为α时的最小生成树的长度.

构造一个特殊输入序列σ^*，使得

$$E(L_{R(m)}(\sigma^*))/L(\sigma^*)=\Omega(n)$$

于是，$E(LR(m)(\sigma))/L(\sigma)=\Omega(n)$

我们来计算$E(L_{R(m)}(\sigma))/L(\sigma)$的上界.

令D是n个输入点的直径.

最小生成树的长度$\ge D$.

因每条边的长度$\le D$，任何On-line算法产生的树的长度$\le(n-1)D$.

我们有，$E(L_{R(m)}(\sigma))/L(\sigma)\le(n-1)D/D=n-1$.

于是，$R(m)$的competitive比是$\Theta(n)$.

HIT CS&E

10.4 *On-line Algorithm for Convex hull problems*

- 问题的定义
- On-line算法的基本思想
- On-line算法
- 算法的性能分析

HIT CS&E

问题定义

- 输入
  - 平面上的n个点的集合Q
  - n个点逐个到达，并非同时出现
- 输出
  - CH(Q): Q的convex hull

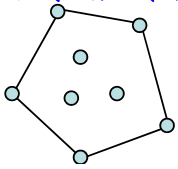
Q的convex hull是一个凸多边形P，Q的点或者在P上或者在P内

凸多边形P是具有如下性质多边形：连接P内任意两点的边都在P内

HIT CS&E

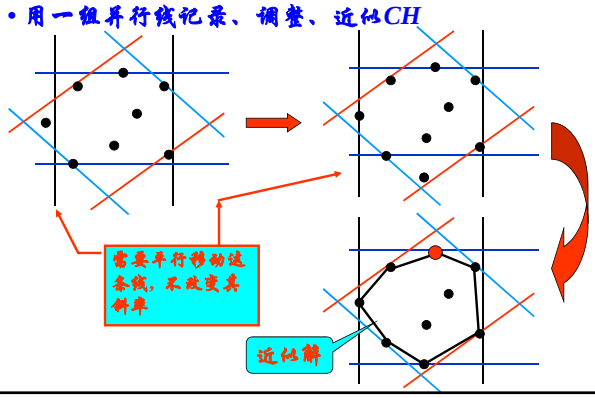
On-line算法的思想

- 基本思想
  - 当k个节点已经到达后，我们得到一个部分CH



  - 当第k+1个节点v到达时
    - 如果v落在CH内部，不需做任何事
    - 如果v落在CH外部，需要调整CH的边界
    - 关键是：记住和调整部分CH边界

- 平行线近似法
  - 用一组平行线记录、调整、近似CH



HIT CS&E

- 平行线的一般形式
  - 取m对平行线，其斜率分别为：
    $0,\ \tan(\pi/m),\ \tan(2\pi/m),\ \ldots,\ \tan((m-1)\pi/m)$.
  - 这m对平行线必须满足：
    - 每个输入节点必须在所有平行线对内；
    - 每对平行线必须尽可能的靠近

HIT CS&E

On-line算法

- 输入
  - 节点序列 $p_1, p_2, \ldots$
  - 满足前面条件的m对平行线
- 输出
  - 近似convex hull序列 $a_1, a_2, \ldots$
  - a_i是覆盖$p_1, p_2, \ldots, p_i$的近似convex hull

HIT CS&E

- 算法A

初始化：构造m对平行线，斜率为$0, \tan(\pi/m), \ldots, \tan((m-1)\pi/m)$；搜索平行线相交在第一个点$p_1$.

第一步：对于任意一个新到达点p_i，如果p_i落在所有平行线对之间，不执行任何操作；否则，平移最接近p_i的线到p_i，不改变其斜率，使p_i成为该线的标记.

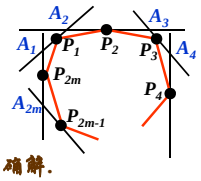
第二步：沿逆时针方向连接每条平行线上的输入点，形成近似convex hull a_i.

第三步：如果不再有点输入，则停止；否则令$i=i+1$，接受下一个点p_i，goto 第一步.

HIT CS&E

算法的性能分析

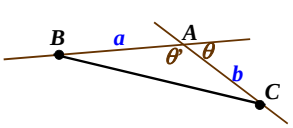
- 时间复杂性
  - $O(mn)=O(n)$ （因为m是常数）
- 近似解精度
  - 三种相关的多边形
    - E: 由$2m$个平行线形成的多边形.
    - C: 输入点集合的Convex hull, 即准确解.
    - A: 算法A给出的近似Convex hull.
  - $L(P)$表示多边形P的总边长，则$L(E)\geq L(C)\geq L(A)$.
  - 近似解A的误差定义为
    $ERR(A)=(L(C)-L(A))/L(C)$

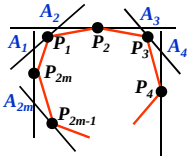


定理1. $ERR(A)\leq sec(\pi/2m)-1$.

*定理1说明m越大(即平行线对越多)，错误越小.

证. 考虑下图：

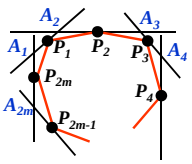




$$\left(\frac{AB+AC}{BC}\right)^2=\frac{(a+b)^2}{a^2+b^2-2ab\cos\theta'}=\frac{a^2+b^2+2ab}{a^2+b^2-2ab\cos\theta'}=\frac{(a^2+b^2)/2ab+1}{(a^2+b^2)/2ab+\cos\theta}$$
$$\leq\frac{1+1}{1+\cos\theta}=\sec^2\frac{\theta}{2}$$

于是，$AB+AC\leq BC\cdot sec(\theta/2)$.

考虑下图，



由$AB+AC\leq BC\ sec(\theta/2)$，我们有

$L(E)= P_{2m}A_1+A_1P_1+P_1A_2+\ldots+A_{2m}P_{2m}$
$\leq sec(\pi/2m)(P_{2m}P_1+P_1P_2+\ldots+P_{2m-1}P_{2m})$
$= sec(\pi/2m)L(A)$.

于是，$Err(A)= (L(C)-L(A))/L(C)$
$\leq (L(E)-L(A))/L(A)$
$\leq sec(\pi/2m)-1$

HIT
CS&E

作业

设Euclidean空间中点集R和B形成完全二分图$\mathcal{B}(R,B)$，且$|R|=|B|=n$，R中的点已知但B中的点online到达。试设计一个online算法计算$\mathcal{B}(R,B)$中的最大匹配，分析其近似比。