one-sided crossing minimization

丛宇

April 22, 2024

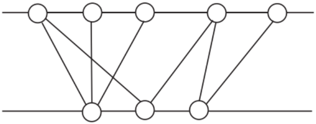
Overview

1. problem
2. heuristics
3. exact

Some definition

1、bipartite graph（二部图）:

A graph G = (V, E) with vertex set V and edge set E ⊆ V × V is called bipartite if there is a partition of V into two disjoint non-empty sets L_1 and L_2 such that $V = L_1 \cup L_2$ and $E \subseteq L_1 \times L_2$.



Some definition

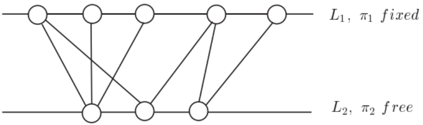
2、 **some properties** of the bipartite graph:

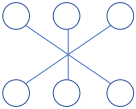
①In these drawings the vertices are arranged on two “layers”.

②Edges are drawn straight between vertices on adjacent layers.

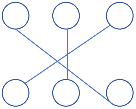
③Edges between vertices on the same layer are not permitted.

④**No point between layers may lie on more than two edges**.





Wrong

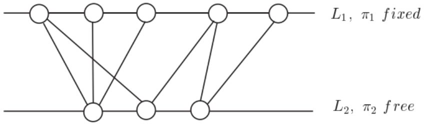


Right

1-Sided Crossing Minimization problem

Instance: **a bipartite graph G = (L_1, L_2; E)**, an **integer k**, and a **fixed ordering π_1 for the vertex set L_1** on the top layer.

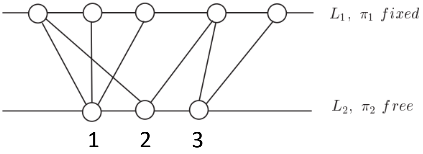
Question:Is there a 2-layer drawing of G that respects π_1 and that has **at most** k crossings?(At first,we don't consider multiple edges.We will handle it in the end.)



Some Facts and definitions

Definition 1:Consider a problem instance $<G, \pi_1, k>$, and let v and w be vertices in L_2. The crossing number $\mathbf{c_{vw}}$ is the number of crossings that edges incident with v make with edges incident with w in drawings having v < w; the crossing number $\mathbf{c_{wv}}$ is for w < v.（c_{vw}的定义）

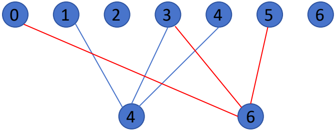
c_{12}=2
c_{13}=0



An Efficient FPT Algorithm

Input: $|L_2| \times |L_1|$ adjacency matrix A of G
Output: $|L_2| \times |L_2|$ matrix C

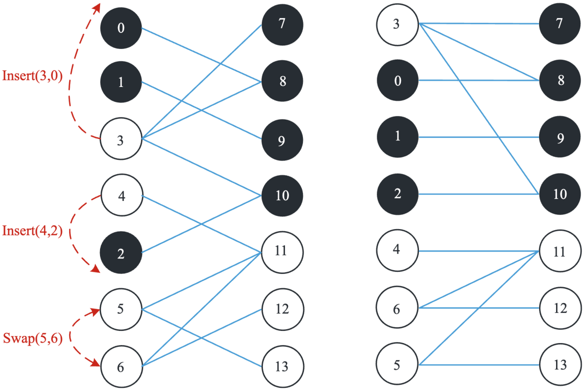
1. Augment adjacency matrix A as described above.
2. **for** $v = 1$ to $|L_2|$
3. **for** $w = 1$ to $|L_2|$
4. **if** $v \neq w$
5. $c_{v,w} = 0$ /* initialize $c_{v,w}$ */
6. $w' = l_w$ /* start examining the neighbors w' of w starting with the leftmost neighbor, l_w */
7. **while** $w' \leq r_w$ and $w' < r_v$ and $c_{v,w} \leq k$ **do**
8. $c_{v,w} = c_{v,w} + r_{v,w'}$ /* increment $c_{v,w}$ by the number of crossing points on edge (w, w') created by edges incident to v */
9. $w' = p_{w,w'}$ /* advance to the next neighbor of w */



如果计算过程中出现大于k的值，则停止继续计算，因为在后续的搜索中，如果有一个c_{vw}大于k，那么就剪枝，所以大于k的数，只需要知道他比k大即可，不需要知道具体的数字。

dynamic programming

直接用 dp 求最优解, $O(\text{poly}(n)2^n)$



想要一个很快的 dp 但是不用得到最优解.
限制我们能做的操作. 求最优解的 dp 中我们可以把顶点插入任何地方, 如果我们插入的区间不能重叠, 操作之间就没有影响.

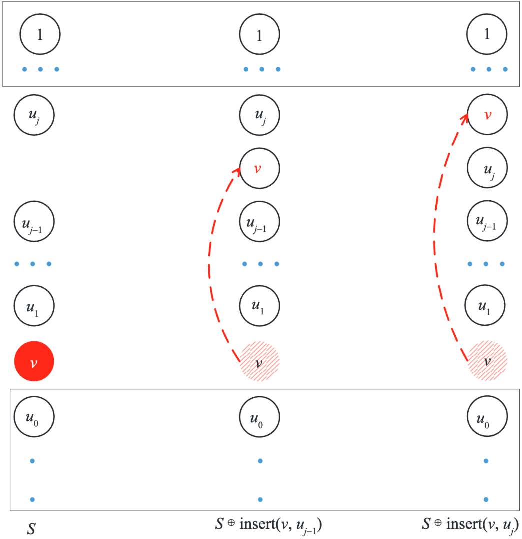
dp

$\mathrm{dp}(i)$ 表示前 i 个顶点交叉数最少是多少个. $\delta(i,j)$ 表示在当前排列下把第 i 个点插入到第 j 个点前面的交叉数减少量.

$$\mathrm{dp}_k(i) = \begin{cases} 0, & \text{if } \ i = 1, \\ \delta_k(1,2), & \text{if } \ i = 2, \\ \max\limits_{1 \leqslant q \leqslant i-1} \{\mathrm{dp}_k(q) + \delta_k(q+1,i)\}, & \text{otherwise.} \end{cases}$$

预处理和 dp 都是 $O(n^2)$

preprocessing



FTP

后面考虑的都是 OSMC 的判定版本, 交叉点数是否小于等于 k.
有几个结果:

- $\tilde{O}(1.618^k)$
- $\tilde{O}(1.46^k)$
- $\tilde{O}(2^{O(\sqrt{k}\log k)})$ subexponential – weighted feedback arc set.

1.618 and 1.46

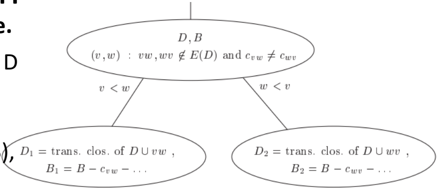
An Efficient FPT Algorithm

Step 3. Building and exploring the search tree.

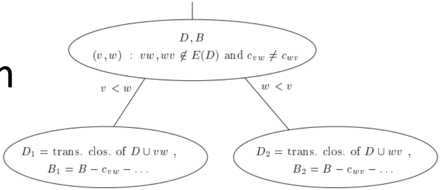
①Label the root of the tree with (D, B) where D = D_0 and B = B_0

②In general, for a non-leaf node labeled (D, B), choose a pair (v, w) such that **D contains no edge** joining v and w and such that $\mathbf{c_{vw} \neq c_{wv}}$（**说明并不在集合C中，也不在D中，是一个$c_{vw} \neq c_{wv}$的unsuited pair**）

③由于选中的v和w只有两种可能的排列，v<w和w<v，所以搜索树上的每个节点都有两个子节点



An Efficient FPT Algorithm



④D_1 = transitive closure of D∪vw

$B_1 = B - c_{vw} - \sum_{pq} c_{pq}$，其中$c_{pq} \neq c_{qp}$

D_2 = transitive closure of D∪vw

$B_2 = B - c_{wv} - \sum_{pq} c_{pq}$，其中$c_{pq} \neq c_{qp}$

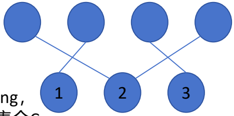
⑤搜索结束条件：

(Ⅰ)不存在unsuited pair (v, w)∉ C

(Ⅱ) $B_1 < 0$ 且$B_2 < 0$

⑥有解意味着存在某个叶子节点(solution leaf)，使得∀(v,w) 如果 vw ∉ D and wv ∉ D, 那么 (v, w) ∈ C，即所有的点对的序列要么在D中，要么在C中，除了集合C不存在未排序的点对。

⑦如果有一个solution leaf，那么输出D的拓扑排序，如果有些点并不在D中，说明这些点的任意排列对交叉数并无影响，可以任意排列。



1<3是natural ordering，(1,2)以及(2,3)会在集合C中，2的位置可以是任意的。

An Efficient FPT Algorithm

⑧update the minimum number of crossings found so far to $k - B$, where B is the budget of the leaf, and update the best ordering so far to π_2.通过这个方法，可以找到一个最优排列。

weighted feedback arc set