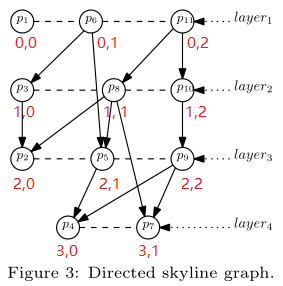
肖翱 2016213650

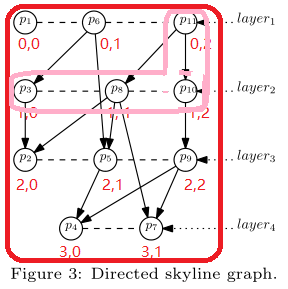
**Pwise** 主要就是基于扩增点的方式在每一层动态生成候选的排列树，同时要将非GSkyline的候选树排除掉。在动态生成的过程中进行剪枝操作，尽可能去掉冗余部分。初始化树的第0层有一个为空的根节点，tail set是预处理后得到的Directed Skyline Graph前k层的所有数据，然后从·tail set中选取合适的点加入G中，第一层是加入1个点后所有满足条件的Gs，第二层继续增加一个合适点，第三、四层等以此类推，层数对应当层Gs中每个G的元素个数。

**PWise+** 在了解论文中整个的PWise算法核心思想之后，不难发现其实该算法还有进一步改进优化的空间。Tail set的规模还可以进一步缩小，每次初始化tail set的时候可以将Directed Skyline Graph的前K层的所有点缩小至G中最后一个元素所处layer之后的点到紧邻的下一个layer的所有点。而这种做法主要是利用到Directed Skyline Graph每层点的有序性，以及候选树节点G中元素的有序性。由于tail set的规模缩小，在动态生成候选树的剪枝过程变得更加精炼快速，而且也能节省空间，减少内存消耗。

因为需要用到有序性，所以除了在论文预处理阶段对点进行一次index标记之后，得到的Directed Skyline Graph后，需要对其中的每个点增添2个属性——标记出所在层数layernum以及在该层中的排序下标layeridx，这在后边缩小tail set的规模和动态剪枝过程大有裨益。



然后初始化tail set时，将Directed Skyline Graph的前k层所有点收缩到两层即可，可以有效避免在后面剪枝过程中一些不必要的重复筛选。由于G是逐元素扩张的，而扩张方向是从Directed Skyline Graph的顶层往下，所以存储G的时候用保留顺序的List数据类型会很方便，于是G中元素有着层数非递减，同层元素index单调递增的性质。那么最后一个元素一定是G中最新加入的layernum和layeridx相对最大的。不防令K=4，此时新构建的候选G={1,6}则论文里的初始化tail set规模用红线标出，而优化后的tail set初始化规模用粉线标出：



这样的tail set规模收缩处理，可以显著减少内存开销，也能减少后面动态构建候选树的筛选操作，也能够一定程度上减少时间开销，提高整个算法的效率。

并且每次在给G从其tail set中挑选合适的元素设为P加入G时，当P通过剪枝过程“幸存”下来，还需要检验新的G’ = G∪{p}是否是G-Skyline，即检验G’中所有元素的parents集合是G’的子集。关于这一点，可以利用有序性，不需要对G’中所有元素求parents集合，只需要对G’最后一个元素（新加入的）P的parents集合验证为G‘子集即可。

**推论 1：按上述算法（通过标记得到的有序性）构造的G’中最后一个元素（最新加入的）P的parents集合Pps****G’，则一定有G’中所有元素的parents集合G’psG’。**

次推论可结合论文中的Theorem 2，通过归纳演绎法得证，这里就不赘述了。