The Point-Wise Algorithm算法文档

肖翱 2016213650

在枚举树法进行G-Skyline groups的求解过程中，会产生庞大的时间与空间开销，于是论文中提出基于扩增点的方法进行树构建以及动态剪枝，最大程度地去掉冗余的部分，提高算法的效率，节省了计算过程的时间和空间。初始化树的第0层有一个根节点为空，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group中，第一层是加入1个点后所有满足条件的groups集合，第二层继续增加一个合适点，第3层，第4层以此类推，层数对应当层groups列表中每个group的元素个数。其中的剪枝优化主要有以下两个方面：

**子树修剪** 应用超集单调性这一性质，如果一个包含i个元素的集合Gi不是G-Skyline group，即使从其tail set中新增加一个点，组成的新的包含i+1个元素的集合Gi+1也不可能是G-Skyline group。所以在算法中，每从一个tail set中添加完一个合适的元素到原group得到的新group都要进行G-Skyline的判定，若新生成的group不是G-Skyline group，直接删除该节点，相当于直接去掉了一个以该节点为根节点的子树。

**Tail Set修剪** 根据文中引理G-Skyline group里面的点不能被除此group里面的元素以外的点所控制，即G-skyline group里每个元素只能是无父节点，或所有父节点在此group内。但这一步骤并不充分，若成功组成新group之后，得再次验证确保准确性。还有一个就是新加入的点必须保证在Directed Skyline Graph的前i+1层，若加入了大于等于i+2层以上的点，那么该点会被Gi以外的点所控制，不满足G-Skyline group定义。

其实，实现这个算法之后总结起来就是，每新加入的一个点首先是得在Gi的上一个加入的点对应的Directed Skyline Graph同一层找skyline点（即该层下标较大点），若已没有skyline点则向下一层i+1层寻找被Gi中元素控制的孩子即可。然后再循环建树的下一层，直到建完第k层截止，便可得到所有的满足条件的G-Skyline groups。

这个算法函数是**public** List<PDataNode> getGSkylineP(DirectedSkylineGraph graph, **int** k)，是按照论文中的伪码实现的，在实现过程中丰富了其中的细节，代码较长这里就不贴图了，助教可自行查看源码，它继承自GSkylineService.java类。同时也在项目中定义了树节点PDataNode，方便计算：

