2015-11

现代数据库系统

——GSkyline算法实现与评估

杭天梦 2015213527 hangtianmenglisa@163.com

王需 2015311975 wangxu.93@icloud.com

康荣 2015311965 kr11@mails.tsinghua.edu.cn

**GSkyline算法实现与评估**

在本次作业中，我们小组实现了***Finding Pareto Optimal Groups: Group-based Skyline***中提出的PointWise Algorithm 和 UnitWisePlus Algorithm，并对算法实现的正确性和效率进行了评估，通过比较两种算法在不同数据集的表现，初步提出了进一步改进的思路，具体实现将在第二次作业中完成。

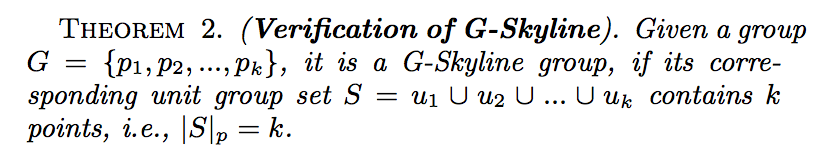
# 背景

Skyline的关键是它包含了所能支配其他点的“最好”的点。但是有时候我们想要搜索一组备选数据而不是唯一一个数据，这时就需要引入GSkyline概念。给定组的长度k，它能选出点集中最优的组，且该组不会被其他组支配。

## PointWise算法

该算法的主要思想是每次在每一层动态生成候选的排列树，同时也要将非G-Skyline的候选树尽可能多的删除。对于每一个节点，我们会为它存储一个尾集(tail set)，尾集用来存储比此节点序号大的所有节点。这样，每次从节点的尾集中挑选一个合适的节点加入到当前的集合中，便可以生成新的集合。在预处理后，根节点包含的集合是一个空集，并且它的尾集是前k层所有节点的集合。

## UnitWisePlus算法



根据定理2，我们可以知道G-Skyline组可以由单元组（unit group）的集合组成。

前面的PWise算法是一次添加一个点，而UWise+算法一次添加一个组，接下来的思想和PWise基本相似。值得一提是，UWise+是UWise算法的加强版，它对所有的单元集做了一个逆序，因为节点的序号越大，就意味着其所在的层数越大，那么它就会有很多父亲节点，因此，它的单元组的元素个数就更大。这样的好处是：不符合要求的单元组会在算法执行前期被删掉，因此增加了算法的执行效率。

# 实验环境

|  |  |
| --- | --- |
| 操作系统 | Windows8.1 |
| 处理器 | Intel(R) Core(TM) i7-4790K CPU 4.00GHz |
| 内存 | 8．00GB |
| 编译环境 | Visual Studio 2013 |

# 使用指南

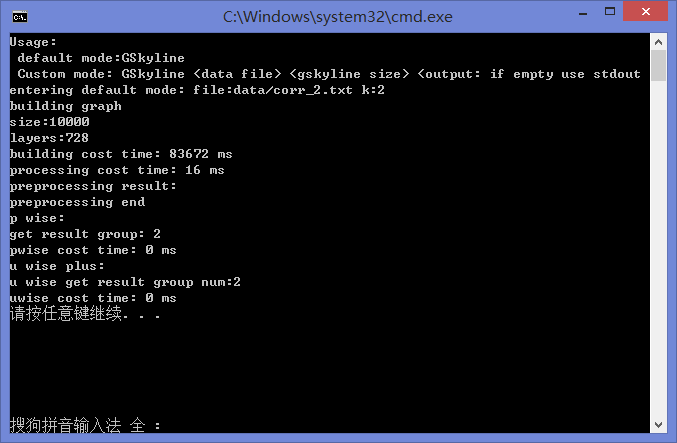
算法使用方法介绍：

1. 在控制台中，输入

“GSkyline.exe 数据文件路径+文件名 k的值”

如“GSkyline.exe E:/input.txt 2”,结果会输出到控制台。

1. 输出值解释



Size:数据量大小

Layer：生成的层数

Building cost time:建立有向图所花的时间

Processing cost time:预处理所花的时间

Get result group: pwise算法所得的最优组的组数

Pwise cost time：pwise所花费的总时间

U wise get result group num: Uwise+算法所得最优组的组数

Uwise cost time： Uwise+算法所花的时间

# 性能比较

本次实验所展示的算法为Pwise+算法。我们对原生Pwise进行了一些优化，从图表可以看出，在k=1,2时，Uwise+所花费的时间比Pwise+要多（具体优化方案在下次实验会统一说明）。由于电脑配置有限，下表仅展示k取1,2的时候程序运行情况。测试k=3的时候，程序也可跑出结果，但花费很长的时间。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文件 | k | 建图时间(ms) | 预处理时间(ms) | | Pwise+组数 | Pwise+运行时间(ms) | Uwise+组数 | Uwise+运行时间(ms) |
| corr\_2.txt | 1 | 1326 | 0 | 1 | | 0 | 1 | 0 |
| inde\_2.txt | 1 | 920 | 0 | 10 | | 0 | 10 | 0 |
| anti\_2.txt | 1 | 468 | 0 | 139 | | 0 | 139 | 0 |
| corr\_4.txt | 1 | 156 | 0 | 9968 | | 0 | 9968 | 0 |
| inde\_4.txt | 1 | 312 | 0 | 8087 | | 0 | 8087 | 0 |
| anti\_4.txt | 1 | 32 | 15 | 9988 | | 0 | 9988 | 0 |
| corr\_6.txt | 1 | 63 | 15 | 9995 | | 0 | 9995 | 0 |
| inde\_6.txt | 1 | 125 | 0 | 9647 | | 0 | 9647 | 0 |
| anti\_6.txt | 1 | 47 | 15 | 9997 | | 0 | 9997 | 0 |
| corr\_8.txt | 1 | 47 | 15 | 9996 | | 0 | 9996 | 0 |
| inde\_8.txt | 1 | 62 | 16 | 9924 | | 0 | 9924 | 0 |
| anti\_8.txt | 1 | 78 | 16 | 9928 | | 0 | 9928 | 0 |
| anti\_2.txt | 2 | 515 | 0 | 9717 | | 0 | 9717 | 0 |
| corr\_2.txt | 2 | 1435 | 0 | 2 | | 0 | 2 | 0 |
| inde\_2.txt | 2 | 999 | 0 | 47 | | 0 | 47 | 0 |
| corr\_4.txt | 2 | 47 | 0 | 49675528 | | 16786 | 49675528 | 38922 |
| inde\_4.txt | 2 | 250 | 0 | 32695745 | | 10031 | 32695745 | 25646 |
| anti\_4.txt | 2 | 32 | 0 | 49875087 | | 14477 | 49875087 | 37362 |
| corr\_6.txt | 2 | 31 | 0 | 49945015 | | 15210 | 49945015 | 39796 |
| inde\_6.txt | 2 | 94 | 0 | 46527487 | | 15225 | 46527487 | 38065 |
| anti\_6.txt | 2 | 31 | 0 | 49965008 | | 15164 | 49965008 | 38688 |
| corr\_8.txt | 2 | 31 | 0 | 49955010 | | 15522 | 49955010 | 38143 |
| inde\_8.txt | 2 | 62 | 0 | 49237927 | | 15257 | 49237927 | 38142 |
| anti\_8.txt | 2 | 47 | 0 | 49277630 | | 14851 | 49277630 | 38377 |

由表可以看出，当k=1时，除去建图和预处理所花的时间，Uwise+正式运行所花的时间可以忽略不计。而当k=2时，当数据的维数大于等于4的时候，Uwise+正式运行时所花的时间开始增多。

# 未来的工作

改进UnitWise算法或者提出新的算法解决在Anti系列数据集合的问题。

# 结论

Uwise+算法对于不同种类的数据集(corr,inde,anti)体现出的性能也不同，下一步需要改进UnitWise算法或者提出新的算法解决在Anti系列数据集合的问题。