**AG-Skyline 实验报告**

丁霄汉 2017312365 吴超月 2017213865 罗  瑶  2017213866

[1. 实验目的 2](#_Toc22724)

[2. AG-Skyline 2](#_Toc28767)

[2.1AG-Skyline定义 2](#_Toc19484)

[2.2判断一个组是否属于AG-Skyline 2](#_Toc17067)

[2.3计算数据集P的AG-Skyline 3](#_Toc11610)

[3. 实验结果与分析 5](#_Toc32557)

[3.1 12组生成数据实验结果与分析 5](#_Toc7489)

[3.2 NBA真实数据实验结果与分析 7](#_Toc27850)

# 实验目的

在论文《Finding Pareto Optimal Groups: Group-based Skyline》中的G-Skyline算法基础之上进行问题重定义，给出重新定义后的AG-Skyline，分析相关性质与定理并给出使用若干剪枝策略的AG-Skyline算法。最后实现该算法，并在合成数据和真实数据集上进行实验。

# AG-Skyline

## 2.1AG-Skyline定义

给定一个包含n个点的数据集P，所有点均属于一个维度为d的空间。令，是两个具有k个点的不同组，我们说 ag-domiante 如果对于所有的对，，并且对于至少一对，。AG-Skyline包含所有不被任何其他具有相同大小的组ag-dominate的组。

对比原论文中的g-dominate和我们定义的ag-dominate。g-dominate只要求在两个组各自点集的某一个组合次序上具有一一dominate关系，而ag-dominate要求每个点对另一个组中的所有点均具有dominate关系，换句话说，ag-dominate要求在所有的组合次序上均具有一一dominate关系。显然，ag-dominate关系比g-dominate更严格，所以在相同的数据集和相同的k值上，AG-Skyline是G-Skyline的超集。

## 2.2判断一个组是否属于AG-Skyline

对于一个大小为k的组：。我们判断其是否属于AG-Skyline，就是看是否找得到一个组，组里面任意一个点对于G里面的每一个点，均具有dominate或者等于关系，当然必须存在至少一个dominate关系。

**定义1**：定义一个点的Unit group，Unit group包含和的所有父亲，记为。

如果存在一个点对于G里面的一个点，具有dominate或者等于关系，显然：。而如果存在一个点对于G里面的任何一个点均具有dominate或者等于关系，对应的：。令，即。

**定理1：**如果， ag-dominate G，则对于中的每一个点，，显然。

**引理1：**判断一个组G是否属于AG-Skyline，即判断是否存在一个组，其ag-dominate G，。可以求。如果，证明找得到一个大小为k的组，组里面的点均来自，显然这个组ag-dominate G。

**定理2：**对于，如果存在某个点，，一定属于AG-Skyline。因为，所以找不到一个大小为k的组，组里面的点均来自。

**定理3：**对于，如果存在某个点，。那么只要存在，不是的孩子，那么一定属于AG-Skyline。因为，，并且只有是的孩子时，。

对于，如果所有点，。我们可以通过求来判断其是否属于AG-Skyline。

## 2.3计算数据集P的AG-Skyline

由于判断一个组是否属于AG-Skyline，我们需要利用一个点的unit group和孩子集合，所以，类似G-skyline的做法，我们首先计算P的所有层Skyline，然后生成DSG，这个DSG与G-skyline中的DSG的区别是每个DSGNode不仅包含该点的父亲和孩子集合parents, children，还包含该点的unit group: unit，unit就是自身加上parents构成的集合。

**预处理** 在输出DSG之前，我们可以对DSG中的点进行删减，删减策略为：

1. 当某个点自身的unit的大小小于k时，可以直接构造包含该点的AG-Group。即在剩下的点中任选k-1个点，与该点就能构成AG-Group。
2. 当某个点自身的unit的大小等于k时，也可以直接构造包含该点的AG-Group。即在剩下的点中选k-1个点，这k-1个点不能全是该点的孩子就能构成AG-Group。

令Group中所有点的unit的交集为AG-Group的commonUnit，令AG-Group中所有点的children的交集为AG-Group的commonChildren。我们可以构造一棵Group枚举树。由Group初始大小为0，每次往里面添加一个点直到group大小为k。

**剪枝策略1:** 每次计算新的group的commonUnit和commonChildren，如果commonUnit大小小于k，则包含此group的大小为k的group一定为AG-Group。如果commonUnit大小等于k，则包含此group的大小为k的group剩余的点只要不全属于commonChildren集合，就一定是AG-Group。这两种情况可以直接输出AG-Group集合，结束往group增加新元素。

**剪枝策略2：**每个group都有一个tail set，我们每次都向group中添加tail set中的一个candidate point。仅当candidate point属于group的commonUnit中某个点的孩子集合时，新的group的commonUnit才有可能不为空，当candidate point不满足这种条件时，新的group的commonUnit必为空，可以直接利用新group构造AG-Group输出。所以我们将遍历tail set删减为仅检查满足这种条件的candidate。

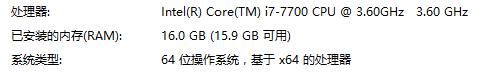
**算法伪代码：**

|  |
| --- |
| input: DSG and group size k  output: AG-Skyline(k) groups  initialize the Group (0) at root node as an empty set and its tail set as all points from DSG after preprocessing  for i = 1 to k:  for each Group (i-1) G do  for each point in G’s commonUnit do  add ’s children to Children Set  for each point in Tail Set do  if is not in Children Set then  construct AG-Group concluding G and  delete  for each remaining point in Tail Set do  add to G to form a Group (i)  if the size of new candidate group’s commonUnit < k do  construct AG-Group concluding new candidate group  delete  if the size of new candidate group’s commonUnit = k do  construct AG-Group concluding new candidate group and the rest points don’t all belong to common children set of new candidate group  delete |

# 实验结果与分析

  实验首先在给定的三种数据分布（inde、corr、anti）的4种维度（2、4、6、8）下共12组数据上进行，每组数据集大小均为50。实验的group size k取值分别为2、4、6、8。此外，实验还收集了50位NBA球员的五个维度的属性（篮板、助攻、抢断、盖帽、得分）数据作为实验数据，取group size k为4、5、6进行测试。NBA实验数据来自http://www.stat-nba.com/。

实验环境：





## 3.1 12组生成数据实验结果与分析

表1、表2、表3、表4分别展示了k取2、4、6、8时算法在12组生成数据上的运行结果。其中每张表的第一列展示被测试的数据集性质（名称），第二列展示预处理之后的点的个数，第三列展示预处理之后得到的group个数，第四列展示最终获得的结果集大小，第五列展示算法的运行时间（单位为微秒），最后一列为表中的第四列减去第三列得到的结果。

从表中的实验结果数据我们可以看出实验数据集的特点。首先观察第二列，即预处理之后的点数，如果预处理之后的点数较少，比如anti系列数据集，则说明该数据集中的元素普遍实力相当，即没有明显的优劣之分；如果预处理之后的点数较多，比如corr和inde数据集，则说明该数据集中的元素分层较为明显。

观察最后一列，即结果集与预处理结果之差，可以看出在预处理之后的点数差不多的情况下，不同类型的数据集的结果集与预处理结果之差的差异明显。比如表1中的corr\_2和inde\_2两个数据集，它们的预处理之后的点数相差不大，但是观察结果集与预处理结果之差可以看到，corr\_2为0，而inde\_2为131，说明corr\_2这个数据集中实力较强的点在各个方面的属性值都是排在前列的，而inde\_2数据集中的数据虽然分层明显，但是各个元素的优劣是有互补的，因此可以结合成为次优组合，这也是我们算法的目的——在数据分层明显的情况下找到一些虽然每个元素并不是最好的那一类，但是他们相互组合可以成为一个较为优秀的组合，即找到这些次优组合。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **预处理之后的点数** | **预处理得到的group数目** | **结果集大小** | **时间（微秒）** | **结果集与预处理结果之差** |
| **anti\_2** | 11 | 1135 | 1213 | 1820 | 78 |
| **anti\_4** | 0 | 1225 | 1225 | 1097 | 0 |
| **anti\_6** | 0 | 1225 | 1225 | 1223 | 0 |
| **anti\_8** | 0 | 1225 | 1225 | 1134 | 0 |
| **corr\_2** | 47 | 50 | 50 | 25129 | 0 |
| **corr\_4** | 33 | 639 | 819 | 5801 | 180 |
| **corr\_6** | 16 | 1061 | 1157 | 2297 | 96 |
| **corr\_8** | 47 | 50 | 50 | 24073 | 0 |
| **inde\_2** | 39 | 363 | 494 | 8956 | 131 |
| **inde\_4** | 13 | 224514 | 230296 | 3008 | 5782 |
| **inde\_6** | 9 | 1152 | 1206 | 1820 | 54 |
| **inde\_8** | 2 | 1216 | 1225 | 1536 | 9 |

表1 k=2时各数据集实验结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **预处理之后的点数** | **预处理得到的group数目** | **结果集大小** | **时间（微秒）** | **结果集与预处理结果之差** |
| **anti\_2** | 3 | 230299 | 230300 | 1744 | 1 |
| **anti\_4** | 0 | 230300 | 230300 | 1142 | 0 |
| **anti\_6** | 0 | 230300 | 230300 | 1184 | 0 |
| **anti\_8** | 0 | 230300 | 230300 | 1131 | 0 |
| **corr\_2** | 46 | 51935 | 51935 | 617751 | 0 |
| **corr\_4** | 22 | 218723 | 228718 | 13343 | 9995 |
| **corr\_6** | 7 | 230265 | 230300 | 2164 | 35 |
| **corr\_8** | 1 | 230300 | 230300 | 1605 | 0 |
| **inde\_2** | 32 | 185065 | 219295 | 53553 | 34230 |
| **inde\_4** | 32 | 185065 | 219295 | 52407 | 34230 |
| **inde\_6** | 6 | 229275 | 230300 | 1922 | 1025 |
| **inde\_8** | 1 | 230296 | 230300 | 1703 | 4 |

表2 k=4时各数据集实验结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **预处理之后的点数** | **预处理得到的group数目** | **结果集大小** | **时间（微秒）** | **结果集与预处理结果之差** |
| **anti\_2** | 1 | 15890700 | 15890700 | 1699 | 0 |
| **anti\_4** | 1 | 15890700 | 15890700 | 1654 | 0 |
| **anti\_6** | 0 | 15890700 | 15890700 | 1318 | 0 |
| **anti\_8** | 0 | 15890700 | 15890700 | 1368 | 0 |
| **corr\_2** | 44 | 8831648 | 8831648 | 48112515 | 0 |
| **corr\_4** | 17 | 15870938 | 15890611 | 11362 | 19673 |
| **corr\_6** | 4 | 15890700 | 15890700 | 1887 | 0 |
| **corr\_8** | 0 | 15890700 | 15890700 | 1332 | 0 |
| **inde\_2** | 28 | 14690963 | 15874094 | 132307 | 1183131 |
| **inde\_4** | 10 | 15688092 | 15890700 | 2562 | 202608 |
| **inde\_6** | 2 | 15889413 | 15890700 | 1694 | 1287 |
| **inde\_8** | 0 | 15890700 | 15890700 | 1335 | 0 |

表3 k=6时各数据集实验结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **预处理之后的点数** | **预处理得到的group数目** | **结果集大小** | **时间（微秒）** | **结果集与预处理结果之差** |
| **anti\_2** | 1 | 536878650 | 536878650 | 1682 | 0 |
| **anti\_4** | 0 | 536878650 | 536878650 | 1210 | 0 |
| **anti\_6** | 0 | 536878650 | 536878650 | 1290 | 0 |
| **anti\_8** | 0 | 536878650 | 536878650 | 1285 | 0 |
| **corr\_2** | 44 | 8831648 | 8831648 | 28096736 | 0 |
| **corr\_4** | 13 | 536876439 | 536878650 | 4502 | 2211 |
| **corr\_6** | 4 | 536878650 | 536878650 | 2009 | 0 |
| **corr\_8** | 0 | 536878650 | 536878650 | 1305 | 0 |
| **inde\_2** | 25 | 529004894 | 536873515 | 199233 | 7868621 |
| **inde\_4** | 6 | 536878606 | 536878650 | 2131 | 44 |
| **inde\_6** | 1 | 536878650 | 536878650 | 1745 | 0 |
| **inde\_8** | 0 | 536878650 | 536878650 | 1281 | 0 |

表4 k=8时各数据集实验结果

## 3.2 NBA真实数据实验结果与分析

表5展示了k分别取4、5、6时在NBA数据集上的实验结果，从预处理之后的点数和结果集与预处理结果之差两列数据可以看出，在真实数据集中，存在数据分层明显的情况，但是各个元素之间的优劣是有互补的，因此可以结合成为次优组合，也就是说在需要考虑次优组合的情况下，本实验的AG-Skyline算法是有意义的。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **k** | **预处理之后的点数** | **预处理得到的group数目** | **结果集大小** | **时间（微秒）** | **结果集与预处理结果之差** |
| **4** | 38 | 155116 | 163274 | 216656 | 8158 |
| **5** | 37 | 1682533 | 1745573 | 1178148 | 63040 |
| **6** | 37 | 13565916 | 14573615 | 4288352 | 1007699 |

表5 k分别取4、5、6时在NBA数据集上的实验结果