AG-Skyline 实验报告

丁霄汉 2017312365 吴超月 2017213865 罗 瑶 2017213866

1. 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。

2. 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 $G=\{p_1,p_2,\ldots,p_k\}$，$G^{'}=\{p^{'}_1,p^{'}_2,\ldots,p^{'}_k\}$是两个具有 k 个点的不同组，我们说 G ag-domiante $G^{'}$如果对于所有的(i,j)对，$p_i \le p^{'}_j$，并且对于至少一对(i,j)，$p_i < p^{'}_j$。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 $G=\{p_1,p_2,\ldots,p_k\}$：。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p_j，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p_j的 Unit group，Unit group 包含p_j和p_j的所有父亲，记为$U(p_j)$。

如果存在一个点$p^{'}_i$对于 G 里面的一个点p_j，具有 dominate 或者等于关系，显然：$p^{'}_i \in U(p_j)$。而如果存在一个点$p^{'}_i$对于 G 里面的任何一个点均具有 dominate 或者

等于关系，对应的：$p_i^{'} \in U(p_1) \cap U(p_2) \ldots \cap U(p_k)$。令 $U(G) = U(p_1) \cap U(p_2) \ldots \cap U(p_k)$，即$p_i^{'} \in U(G)$。

定理 1：如果$G^{'} = \{p_1^{'},p_2^{'},\ldots,p_k^{'}\}$，$G^{'}$ ag-dominate G，则对于$G^{'}$中的每一个点$p_i^{'}$，$p_i^{'} \in U(G)$，显然$G^{'} \subseteq U(G)$。

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，$G = \{p_1,p_2,\ldots,p_k\}$。可以求 $U(G) = U(p_1) \cap U(p_2) \ldots \cap U(p_k)$。如果$|U(G)| \geq k$，证明找得到一个大小为 k 的组，组里面的点均来自 $U(G)$，显然这个组 ag-dominate G。

定理 2：对于 $G = \{p_1,p_2,\ldots,p_k\}$，如果存在某个点$p_j \in G$，$\left|U(p_j)\right| < k$，$G$ 一定属于 AG-Skyline。因为$|U(G)| < |U(p_i)| < k$，所以找不到一个大小为 k 的组，组里面的点均来自 $U(G)$。

定理 3：对于 $G = \{p_1,p_2,\ldots,p_k\}$，如果存在某个点$p_j \in G$，$\left|U(p_j)\right| = k$。那么只要存在$p_i \in G,p_i \neq p_j$，$p_i$不是$p_j$的孩子，那么 G 一定属于 AG-Skyline。因为，$\left|U(p_j) \cap U(p_i)\right| \leq k$，并且只有$p_i$是$p_j$的孩子时，$\left|U(p_j) \cap U(p_i)\right| = \left|U(p_j)\right| = k$。

对于 $G = \{p_1,p_2,\ldots,p_k\}$，如果所有点$p_j \in G$，$\left|U(p_j)\right| \geq k$。我们可以通过求 $U(G)$ 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 p_i in G's commonUnit do
            add p_i's children to Children Set
        for each point p_i in Tail Set do
            if p_i is not in Children Set then
                construct AG-Group concluding G and p_i
                delete p_i
        for each remaining point p_i in Tail Set do
            add p_i 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
```

3. 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：

处理器:	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz 3.60 GHz
已安装的内存(RAM):	16.0 GB (15.9 GB 可用)
系统类型:	64 位操作系统，基于 x64 的处理器

```
C:\Users\Administrator>java -version
java version "9.0.1"
Java(TM) SE Runtime Environment (build 9.0.1+11)
Java HotSpot(TM) 64-Bit Server VM (build 9.0.1+11, mixed mode)
```

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