Project 2 G-Skyline 实验文档

何沐　张嘉诚　张飞鸿

相关工作介绍

Skyline 查询作为多属性决策支持的经典方法，自 Börzsönyi 等人在 [1] 中首次提出以来，已经衍生出诸多变体，广泛应用于推荐系统、决策分析、数据挖掘、组合优化等领域。在 Skyline 查询的基本模型中，一个数据点若在所有维度上都不比其他任何点差，并在至少一个维度上更优，则该点被视为 Skyline 点，即“不可支配”的 Pareto 最优点。

然而，传统 Skyline 模型仅支持对单个点的最优选择，无法满足日益增长的“组选择”类决策需求。为了解决该问题，研究者提出了 **群体 Skyline（Group-based Skyline）** 的概念，旨在从数据集中挖掘出由多个点组成、整体上不被其他组支配的最优点集组合。该问题在多个实际应用场景中具有重要意义，例如：旅游公司需推荐多家互补酒店、广告公司需选定多个广告位、NBA 球探需组建一支高效球队等。

一代模型：基于聚合点的群体 Skyline 方法

最初的群体 Skyline 方法主要基于“虚拟聚合点（Virtual Aggregate Point）”的思想。Li 等人 [2]、Zhang 等人 [3] 和 Im等人 [4] 等早期研究通过对组内点的属性值进行均值或加权平均，将每个组映射为一个聚合点，再比较聚合点之间的支配关系来判断组间优劣。这类方法实现较为直接，但存在显著局限：聚合操作会造成信息损失，无法全面捕捉组内点的差异性，进而导致部分“潜在优组”被遗漏。

G-Skyline 概念的提出与标准化模型

为克服聚合点方法的不足，Liu 等人 [5] 在 2012 年首次提出了 **G-dominance** 的定义，并在此基础上引入了 **G-Skyline group** 的概念。该定义要求在两个等大小的点组之间存在一一映射，使得目标组中每个点都不劣于参照组中对应点，且至少一个点严格优于对应点。该模型更贴合真实多维场景下“整体最优但不要求每个点最优”的选择逻辑。

基于 G-dominance，Liu 等人进一步设计了 **Directed Skyline Graph (DSG)** 数据结构，用于刻画前 l 层 Skyline 点及其支配关系，并提出了两种典型的组搜索算法：

- **PWise**：基于“单点扩展”方式，逐个添加满足条件的点构造组；
- **UWise+**：基于“单位组扩展”方式，直接添加候选点的 Parent Group。

DSG 及其算法在二维数据上表现尚可，但随着维度增加，其弊端也逐渐暴露：

1. 图中包含大量冗余点和边，增加了计算与存储负担；
2. Skyline 层构建在高维数据中计算复杂度高；
3. 剪枝策略（如 Tail Set、Subset Pruning）效果有限，搜索效率低。

MDS 框架与 MDG 支持结构的优化突破

为进一步提升 G-Skyline 枚举效率，Wang 等人 [6] 在 2018 年提出了新一代 G-Skyline 计算框架 —— **Minimum Dominance Search (MDS)**。其核心思想是通过结构剪裁和剪枝优化，减少无效计算，从而在高维场景下保持优异性能。

MDS 框架的关键贡献包括：

- 提出 **Minimum Dominance Graph (MDG)**，仅保留那些被少于 l 个点支配的节点及必要边，显著精简图结构；
- 基于 MDG，提出两类搜索算法：

    - **P-MDS**（点扩展）：逐个添加满足 Parent Pruning 条件的点；
    - **G-MDS**（组扩展）：以候选点及其父节点组作为单位进行扩展；
- 引入 **Skyline-combination 优化策略**，提前枚举所有由 Skyline 点组成的组合，避免递归搜索，降低时间复杂度；
- 利用 **改进的 R-tree 索引结构**，通过 Manhattan-norm 启发式划分节点，加速支配关系查找，提高 MDG 构建效率。

实验结果表明，MDS 框架在构建效率、内存占用和搜索性能方面均优于 DSG 及其算法，尤其在维度 $d > 3$、数据支配关系密集时展现出数量级优势。

G-Skyline 的典型应用研究

随着 G-Skyline 理论的发展，其在实际中的应用也不断拓展，已成功应用于以下领域：

- **多用户推荐系统**：为不同偏好用户群体推荐一组互补的商品组合（如旅游、图书、电影等）；
- **广告资源规划**：在价格与人群覆盖率之间平衡，选出最具性价比的广告位组合；
- **设施选址与布局**：如物流中心、连锁商铺等选址问题中，筛选出空间互补、成本控制最优的多点组合；
- **项目组构建**：在技能、经验、成本等多维属性下构建最佳项目团队；
- **复杂事件检测与响应**：在城市安全管理中，结合多源指标选出最值得关注的事件组合。

随着研究的深入，未来 G-Skyline 还可能扩展至以下前沿方向：

- **增量式与在线 G-Skyline 计算**，支持动态数据流；
- **分布式或图计算框架下的高效并行处理**；
- **多约束下的可解释性组优化。**

参考文献

[1] S. Börzsöny, D. Kossmann, and K. Stocker, “The skyline operator,” in Proc. IEEE Int. Conf. Data Eng., 2001, pp. 421–430.

[2] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das, “On sky line groups,” in Proc. ACM Int. Conf. Inf. Knowl. Manag., 2012, pp. 2119–2123.

[3] N. Zhang, C. Li, N. Hassan, S. Rajasekaran, and G. Das, “On sky line groups,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 4, pp. 942 956, Apr. 2014.

[4] H. Im and S.Park, “Group skyline computation,” Inf. Sci., vol. 188, pp. 151–169, 2012.

[5] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang, “Finding pareto opti mal groups: Group-based skyline,” Proc. VLDB Endowment, vol. 8, no. 13, pp. 2086–2097, Sep. 2015.

[6] C. Wang, C. Wang, G. Guo, X. Ye, and P. S. Yu, “Efficient Computation of G-Skyline Groups,” IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 4, pp. 673–686, Apr. 2018.

算法分析

MDG构建算法

算法流程

- 输入：数据点集 P 和 G-Skyline 组的大小 l
- 输出：最小支配图 G
- 步骤：
    - 初始化空图 G

- 对 P 中所有的点按所有维度之和升序排序，以保证后处理点不会被前面的点支配
- 创建 l 个桶 $B[0]$，$B[1]$，……，$B[l-1]$，分别存储具有 0 ~ $l-1$ 个父节点的点
- 遍历 P 中的每个点 p：
  - 初始化 p 的 parents 集合为空
  - 从 $B[l-1]$ 到 $B[0]$ 逆序遍历每个桶中的所有点 p'：
    - 若 p' 支配 p，则将 p' 及其 parents 加入 p 的 parents 集
    - 若 p 的 parents 个数大于等于 l，终止对 p 点的处理
  - 若 p 的 parents 个数小于 l：
    - 将 p 加入 G 的点集中
    - 将 p 的所有 parents 到 p 的边加入 G 的边集中
    - 将 p 加入桶 $B[|\text{parents}|]$

时间复杂度

- 假设 n 为点集大小，m 为最终 MDG 中点的个数（通常远小于 n）
- 对所有点排序的时间复杂度为 $O(n \log n)$
- 遍历 P 时每个点最多比较 m 个已有点，最坏情况的时间复杂度为 $O(nm)$
- 所以整体时间复杂度为$O(n \log n + nm)$

P_MDS

算法流程

- 输入：数据点集 P、 G-Skyline 组的大小 l、最小支配图 G
- 输出：所有 l-点 G-skyline 组组成的集合 R
- 步骤：
  - 主程序：
    - 初始化结果集 $R = \emptyset$
    - 对 G 中所有的点按父节点数升序排序，父节点少的点优先
    - 对每个 Skyline 点 p （没有父节点）：
      - 初始化当前 group $gr = \{p\}$
      - 调用递归函数 $\text{SearchSinglePoint}(gr, p, R)$
  - 递归函数 $\text{SearchSinglePoint}(gr, p, R)$
    - 若当前 gr 大小为 l，将其加入结果集 R，结束当前搜索
    - 否则遍历排序后的 G 中所有在 p 之后的点 p'：
      - 若 p' 的所有 parents 均在当前 gr 中，则将 p' 加入 gr，再递归调用自己

时间复杂度

- 该算法的时间复杂度取决于：
  - m：MDG 中点的个数
  - 每个点的支配关系稀疏程度（ Parent Pruning 剪枝效率）
- 整体时间复杂度应为 $O(km)$，其中 k 为实际生成的 G-skyline 组的数量
- 最坏情况是从 MDG 的所有点中枚举所有可能的 l-组合：复杂度为 $O(C(m,l))$，但实际情况中应有 $km \ll C(m,l)$

实验结果展示

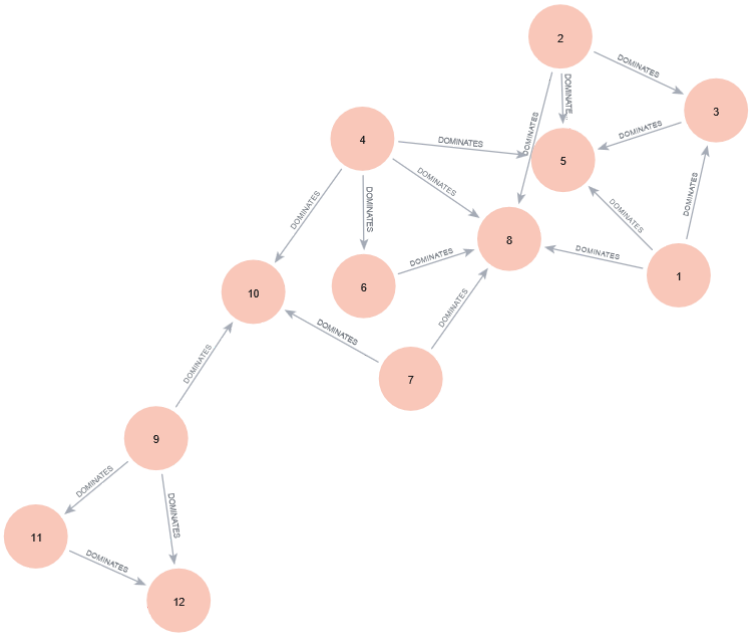
我们使用 python 对上述算法进行复现，`main.py` 为主程序入口：

- 定义了 node 类用于存储点，graph 类用于存储图（包括 DSG 和 MDG 等），相关类和类方法定义见 `nodegraph.py`
- 在 `dsg.py` 中实现了 DSG 构建算法
- 在 `mdg.py` 中实现了 MDG 构建算法
- 在 `readdata.py` 中实现了从文件读入数据的方法
- 在 `search.py` 中实现了 P_MDS 算法
- 在 `visual.py` 中实现了将构建结果通过图数据库 neo4j 可视化展示和查询的操作，我们将 DSG 和 MDG 的节点统一存储到 neo4j 数据库中，并以不同关系标签进行区分，同时将 G-Skyline 组的相关节点单独存入 neo4j 数据库中。

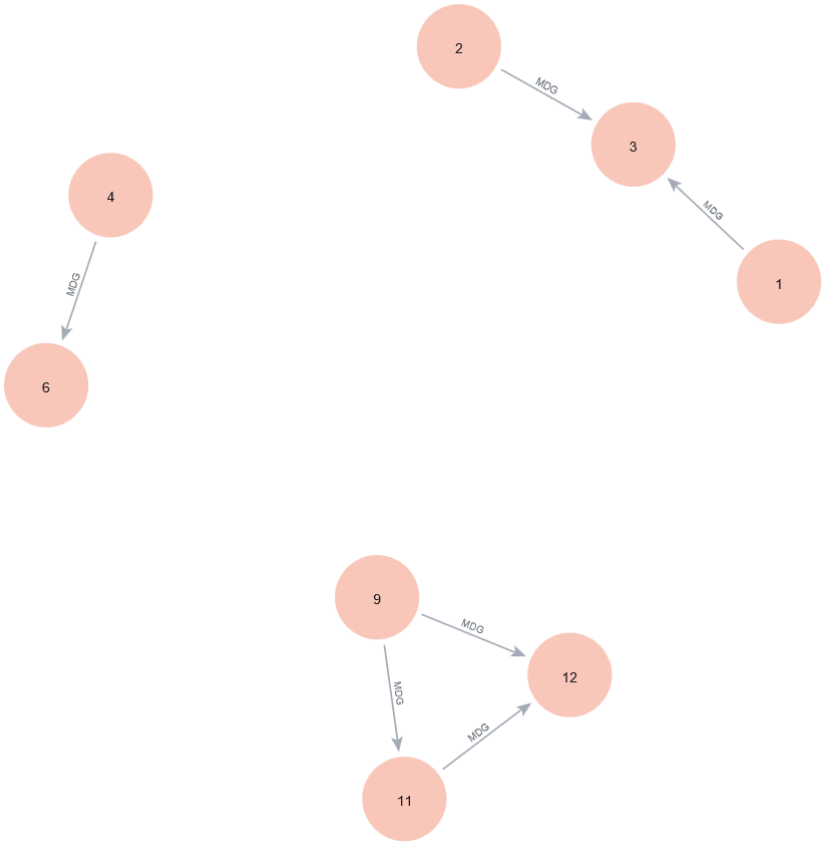
注意在运行程序之前务必先按照 `README.md` 中的说明做好相关配置

以论文中的实例进行验证，结果如下：

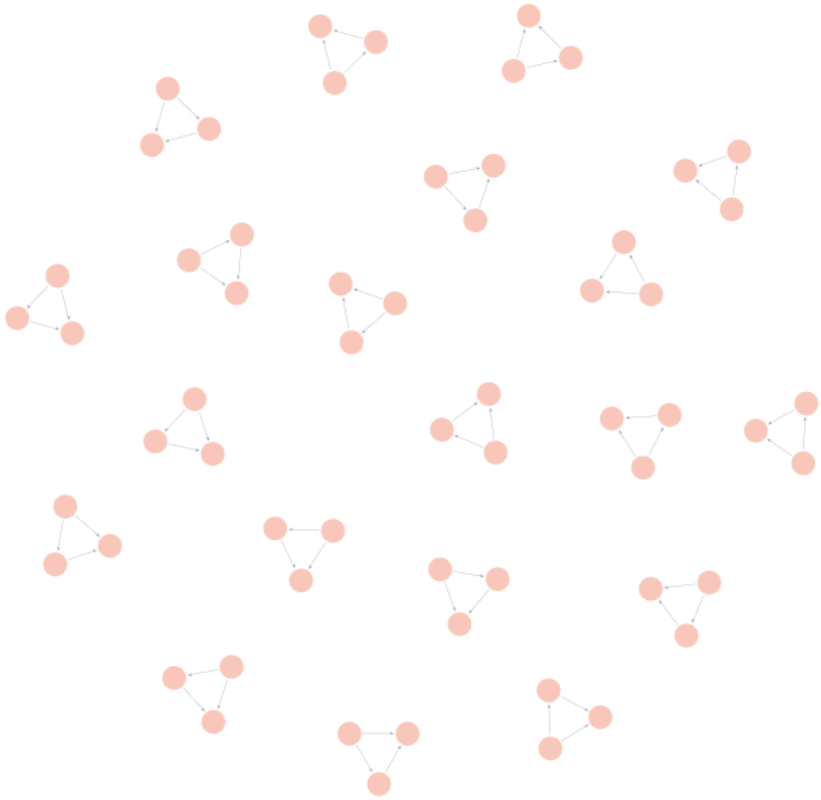
- 查询 DSG（使用语句 `MATCH p=()-[:DOMINATES]->() RETURN p;`）：



- 查询 MDG（使用语句 `MATCH p=()-[:MDG]->() RETURN p;`）：



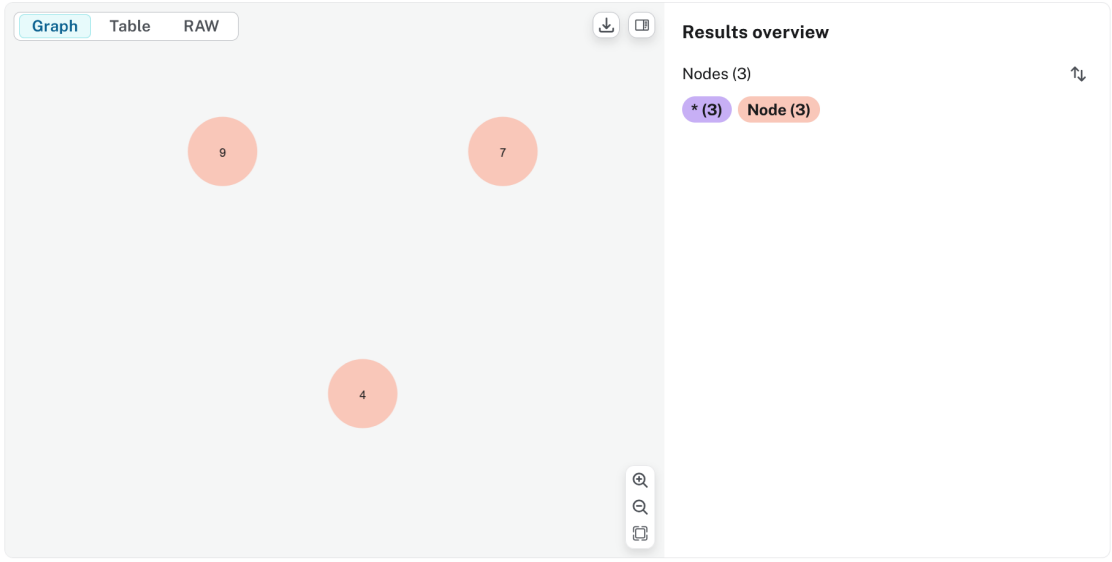
- 查询所有 G-Skyline 组（使用语句 `MATCH p=()-[:GSky]->() RETURN p;`）：



截图视觉效果不太好，可运行程序后自行在 neo4j 中放大查看，注意必须先按 `README.md` 所说将 `config.py` 中的 `importgskyline` 配置为 `True`

- 查询具备特定属性的 Skyline 节点，可通过修改查询语句中 AND 操作符后的相关语句改变查询条件：

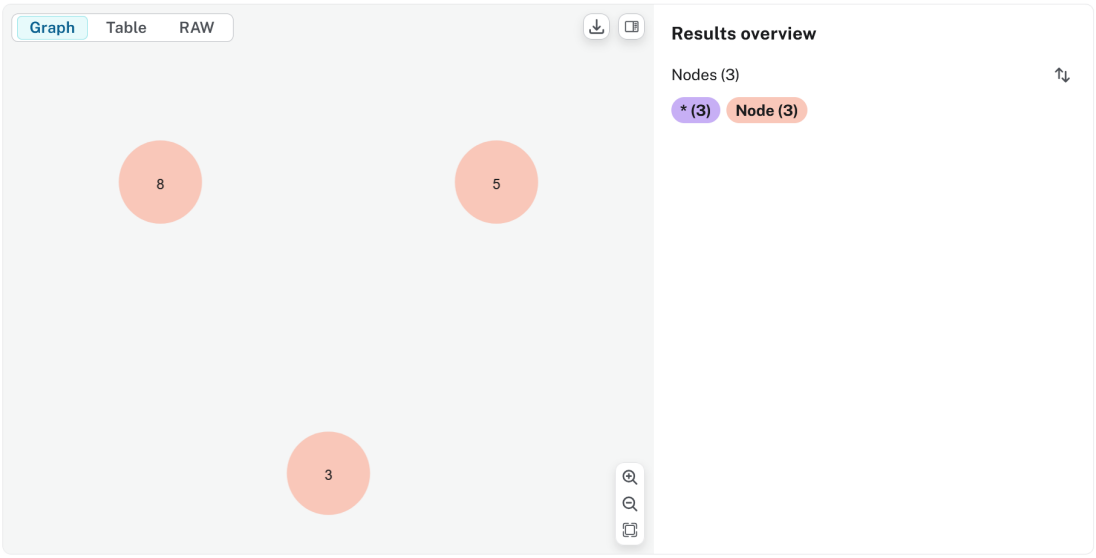
```
MATCH (n)
WHERE NOT EXISTS(()-[:DOMINATES]->(n)) AND n.value[0] > 2
RETURN n
```



Started streaming 3 records after 21 ms and completed after 22 ms.

- 查询被特定节点支配或支配特定结点的结点，如下图为查询所有被 id为1的节点支配的节点：

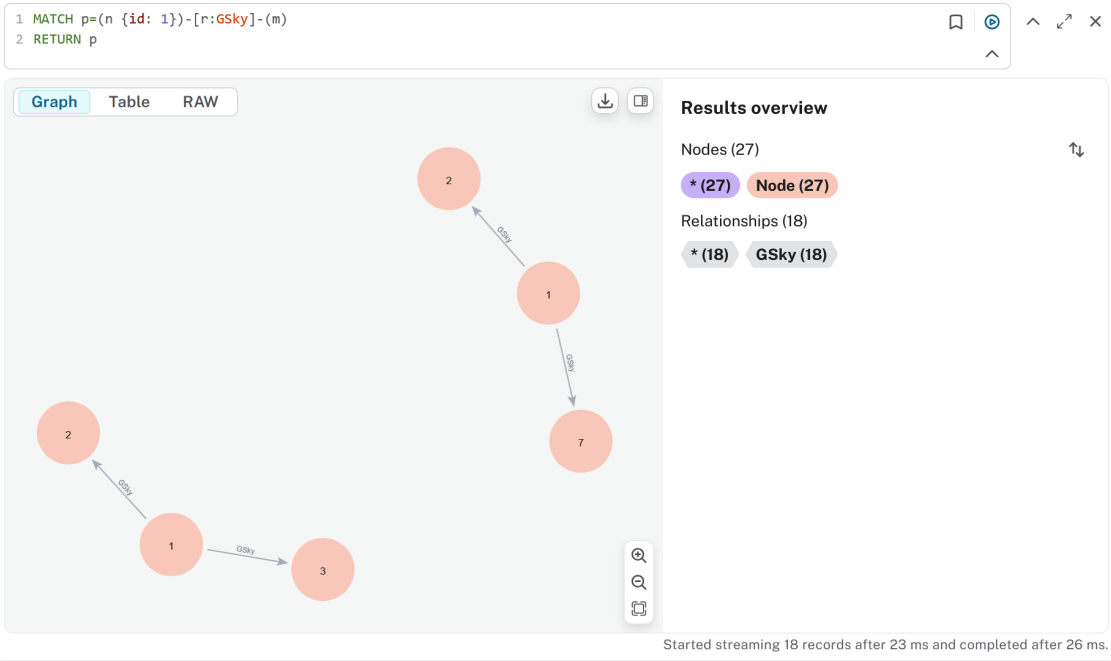
```
MATCH (m),(n)
WHERE EXISTS((m)-[:DOMINATES]->(n)) AND m.id = 1
RETURN n
```



Started streaming 3 records after 53 ms and completed after 55 ms.

This query builds a cartesian product between disconnected patterns.

- 查询包含特定结点的 G-Skyline 组，如下图为查询所有包含 id 为 1 的节点的 G-Skyline 组：



与论文中的相应结果进行对比可知，上述查询和可视化都是正确的，即我们实现的相关算法都是正确的

分工

- 何沐：DSG 和 MDG 构建算法实现，报告撰写（对应算法分析和结果展示部分）
- 张嘉诚：P_MDS 算法实现，报告撰写（对应算法分析和结果展示部分）
- 张飞鸿：数据结构和程序框架设计，可视化实现，报告撰写（相关工作部分）