Presentation: SpecGreedy Unified Dense Subgraph Detection

苏森阳

华中农业大学

2025 年 6 月 25 日

概述

复现代码仓库：
https://github.com/Sy-SU/SpecGreedy-Repro

问题定义

最密子图

设无向图 $\mathcal{G}=(V,E)$, 设点集 $S\in V$, 定义 $g(S)=\dfrac{|E(S)|}{|S|}$, 最大化 $g(S)$.

- 约定 A 表示 S 的诱导子图的邻接矩阵, 我们有 $g(S)=\dfrac{1}{2}\cdot\dfrac{x^TAx}{x^Tx}$,
- 其中:

$$x_i=\begin{cases}1, & \text{若} i\in S\\ 0, & \text{若} i\notin S\end{cases}$$

相关工作

传统解决方法

- 考虑形式化的问题: $\max_{S\in V, S\neq\emptyset} \frac{|E(S)|}{|S|}$.
- 对于给定的 λ, 考虑判断是否存在 S, 满足 $\frac{|E(S)|}{|S|} \geq \lambda$.

最小割建图

- 我们假设存在超级源点 s 与超级汇点 t, 将所有无向边 (u, v) 看成两条容量为 1 的有向边 (u, v) 与 (v, u).
- 将 s 连向所有节点且容量为 m, 即原图 $\mathcal{G}$ 中的边数; 将所有节点连向 t 且容量为 $m + 2\lambda - d_i$, 其中 d_i 为节点 i 在原图中的度数。
- 记这个图为 $\mathcal{G}'$. 设 $S, T \in V'$, 且 $S \cap T = \emptyset$, $S \cup T = V'$ 以及 $s \in S$, $t \in T$.

公式推导

- 记 $V_1 = S \setminus \{s\}$, $V_2 = T \setminus \{t\}$, 于是我们有:

$$
\begin{aligned}
\mathrm{Cut}(S,T) &= \sum_{i\in S, j\in T} c_{ij} = \sum_{j\in V_2} c_{sj} + \sum_{i\in V_1} c_{it} + \sum_{i\in V_1, j\in V_2} c_{ij} \\
&= m\left|V_2\right| + (m+2\lambda)\left|V_1\right| - \sum_{i\in V_1} d_i + \sum_{i\in V_1, j\in V_2} c_{ij} \\
&= m\left|V\right| + 2\left|V_1\right| \left(\lambda - \frac{\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} c_{ij}}{2\left|V_1\right|}\right)
\end{aligned}
$$

公式推导

- 存在一种划分 S, T, 使得 $\mathrm{Cut}(S,T)$ 最小.
- 另一方面, $\mathrm{Cut}(\{s\}, V \cup \{t\}) = m|V|$.
- 由于 $\mathrm{Cut}(S,T) = m|V| + 2|V_1|(\lambda - \frac{\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} c_{ij}}{2|V_1|})$ 为最小割, 我们容易得出:

$$m|V| \leq m|V| + 2|V_1|(\lambda - \frac{\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} c_{ij}}{2|V_1|})$$

- 也就是说, $|V_1|(\lambda - \frac{\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} c_{ij}}{2|V_1|}) \leq 0$.

公式推导

- 注意到 $\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} c_{ij}$ 即为 V_1 内边数量的 2 倍.
- 那么相应的, $\dfrac{\sum_{i\in V_1} d_i - \sum_{i\in V_1, j\in V_2} c_{ij}}{2\,|V_1|} = g(V_1)$.
- 所以我们有 $|V_1|\,(\lambda - g(V_1)) \leq 0$. 这也就是说，当 $V_1 \neq \emptyset$ 时, 我们立即有 $g(V_1) \geq \lambda$, 也就是存在 $S = V_1$, 满足 $\dfrac{|E(S)|}{|S|} \geq \lambda$; 否则, $V_1 = \emptyset$, 也就不存在满足约束的 S.

二分答案求最密子图

- 我们可以通过二分 λ 的方式来确定最大的 $g(S)$.

伪代码

二分答案求最密子图

初始化 l, r
while $r - l > \varepsilon$ **do**
    $\lambda \leftarrow \frac{l+r}{2}$
    构建流图
    计算最小割 C
    if $|V_1| > 0$ **then**
        $l \leftarrow \lambda$
    else
        $r \leftarrow \lambda$
    end if
end while
return l

时间复杂度分析

- 时间复杂度为 $O(\log m\ T(n,m))$, 其中 $O(T(n,m))$ 为求最小割集的复杂度，取决于实现方式。常见的实现方式包括 Edmonds-Karp 算法 ($O(nm^2)$), Dinic 算法 ($O(n^2m)$) 等.
- 实际上, 在应用中, 我们并不一定要理论上界的解; 与此同时, 达到理论上界的时间复杂度过高, 这是不可接受的. 所以在最密子图的研究中, 更多被提及的是快速求近似解的算法.

贪心算法

- 直观地，一个点的度越大，那么它的密度也相对较大. Charikar 提出了一种贪心算法，方法是每次删去图中度最小的点，记录中间图中密度的最大值.

时间复杂度

- 考虑使用优先队列维护每个点的度数，删去优先队列队首点后更新相邻点的度数并重新插入优先队列. 由于每条边最多被遍历 2 次，时间复杂度为 $O((n+m)\log m)$，可以近似认为是线性的.

已有的对最密子图的研究

MinQuotientCut

- 给定 $\mathcal{G} = (V, E)$, 设 $S \in V$, 记 $\overline{S} = V \setminus S$, 最小化 $\dfrac{|\mathrm{Cut}(S, \overline{S})|}{|S|}$.
- 目标: 找到割边相对最少的子图.
- 意义: 在社区检测中, 找到分割清晰的子群体.

已有的对最密子图的研究

Charikar

- 定义 $g(S) = \dfrac{|E(S)|}{|S|}$, 最大化 $g(S)$.
- 目标：找到内部连接最紧密的子图.

已有的对最密子图的研究

Fraudar

- 求 $\max_S \dfrac{|E(S)| + 2 \cdot D_w(S)}{|S|}$. 其中 $D_w(S)$ 表示 S 内点权之和.
- 目标: D_w 可以用于表示在社区中的活跃度. 即求在社交网络中活跃度过高的团伙.

已有的对最密子图的研究

SparseCutDS

- 求 $\max_{S} \dfrac{|E(S)| - \lambda \cdot \left|\mathrm{Cut}(S, \overline{S})\right|}{|S|}$
- 目标：希望找到一个子图，它的密度尽可能高，且割边尽可能疏，可以用于社区检测.

已有的对最密子图的研究

TempDS

- 找到在 t 时 $\dfrac{|E_t(S)|}{|S|}$ 较高而 $\dfrac{|E_{t-1}(S)|}{|S|}$ 较低的 S.
- 目标: 发现突发密集团体.

已有的对最密子图的研究

Risk-averse DS

- 在 $\mathcal{G}=(V,E)$ 中, 定义正常边 E^+ 与惩罚边 E^-, 最大化 $$\frac{|E^+(S)|-\lambda\cdot|E^-(S)|}{|S|}.$$
- 目标: 在金融反欺诈中发现“密集且风险低”的群体.

已有的对最密子图的研究

- 可以发现，在不同的应用场景中，我们需要求解的“最密子图”问题实际上是不同的．所以，每种问题都需要新的数学模型，给研究的推进带来了不便．

GenDS

- 给定一个图 $\mathcal{G}=(V,E)$ 及其对比图 $\mathcal{G}'=(V,E')$, $|V|=n$;
- 寻找一个最优子集 $S^*\subseteq V$, 且 $|S^*|\geq 1$, 使得

$$S^*=\underset{S\subseteq V,\,|S|\geq 1}{\operatorname{argmax}}\; g(S;\mathbf{P},\mathbf{Q})=\underset{x\in\{0,1\}^n,|x|\geq 1}{\operatorname{argmax}}\;\frac{x^\top\mathbf{P}\,x}{x^\top\mathbf{Q}\,x}$$

- 其中矩阵 $\mathbf{P}$ 和 $\mathbf{Q}$ 与图 $\mathcal{G}$ 和 $\mathcal{G}'$ 相关.

GenDS

- 在对比最密子图中 (即原图中尽可能密, 在对比图中尽可能疏), $\mathbf{P}$ 和 $\mathbf{Q}$ 的设置如下:

$$\mathbf{P} = \mathbf{A} + 2\,\mathbf{D}_c, \quad \mathbf{Q} = \mathbf{A}' + \gamma\,\mathbf{I}.$$

其中 D_c 表示点权重向量 $(c_1, c_2, \cdots, c_n)^\top$ 的对角矩阵.

GenDS

- GenDS 是一种框架, 现有的几种有关最密子图检测的任务都可以归约到这个框架上.
- 下表提供了前面提到的不同的最密子图问题对应的 **P** 与 **Q** 的设置.

GenDS

方法	P	Q
MinQuotientCut	$\mathbf{A} - \mathbf{D} = -\mathbf{L}$	$\mathbf{I}$
Charikar	$\mathbf{A}$	$\mathbf{I}$
Fraudar	$\mathbf{A} + 2\,\mathbf{D}_w$	$\mathbf{I}$
SparseCutDS	$\mathbf{A} - \frac{2\alpha}{2\alpha+1}\mathbf{D}$	$\mathbf{I}$
TempDS	$\mathbf{A}_t$	$\mathbf{A}_{t-1} + 2\,\mathbf{I} = \widetilde{\mathbf{A}}_{t-1}$
Risk-averse DS	$\mathbf{A}^+ + \lambda_1\,\mathbf{I} = \widetilde{\mathbf{A}}^+$	$\mathbf{A}^- + \lambda_2\,\mathbf{I} = \widetilde{\mathbf{A}}^-$
GenDS	$\mathbf{A} + 2\,\mathbf{D}_c$	$\mathbf{A}' + \gamma\,\mathbf{I} = \widetilde{\mathbf{A}}'$

奇异值

- 奇异值是递减的, 即 $\sigma_1 \geq sigma_2 \geq \cdots \geq \sigma_n$. 前几个特征值代表着图中最重要的密集方向, 对应着最密集的子结构.
- 因此, 我们只需要选取前 k 个奇异值和奇异向量.

SpecGreedy

- 根据不同的问题类型确定对应的 $\mathbf{P}$ 与 $\mathbf{Q}$. 求出 $\mathbf{A}_r = (\mathbf{P} - \mathbf{Q})^+$.
- 对 $\mathbf{A}_r$ 奇异值分解, 取前 k 个奇异值与奇异向量, 分别记为 Σ, $\mathbf{U}$, $\mathbf{V}$.
- 我们至多进行 k 次迭代. 在第 r 迭代中, 我们尝试基于 $\mathbf{U}$ 与第 $\mathbf{V}$ 的第 r 个分量 $\mathbf{u}_r$ 与 $\mathbf{v}_r$ 选取一个候选子集.

SpecGreedy

- 具体的筛选方法是: 选取所有满足 $\mathbf{u}_{ri} \geq \frac{1}{|L|}, i \in L$ 的点 i 或 $\mathbf{v}_{rj} \geq \frac{1}{|R|}, j \in R$ 的点 j. 特别地, 在普通的无向图中, $L = R = V$, 而在二分图中, L 和 R 分别表示二分图的左右两部分节点.
- 形式化地, 在这一步, 我们实际上挑选了如下点集 S_r 作为候选点集:

$$S_r = \{i : \mathbf{u}_{ri} \geq \frac{1}{|L|}, i \in L\} \cup \{j : \mathbf{v}_{rj} \geq \frac{1}{|R|}, j \in R\}$$

SpecGreedy

- 接下来，我们只需要在点集 S_r 的诱导子图 $\mathcal{G}(S_r)$ 上进行贪心，每次删去度数最小的点，记录平均密度.
- 记贪心过程中的平均密度最大值为 $g(S_r^*)$，并更新迭代过程中诱导子图的密度最大的点集 S.
- 如果 $g(S)$ 比下一轮的奇异值 σ_{r+1} 大，则提前结束循环.

SpecGreedy

- 经过至多 k 次迭代之后，点集 S 的诱导子图 $\mathcal{G}(S)$ 即为 SpecGreedy 算法找到的最密子图.
- 时间复杂度大致是 $O(k(n+m)\log m)$.

运行时间

- 在高密度图 (先生成稠密子图, 然后嵌入边) 中, 我们进行了 10 次实验, 用于对比不同方法的运行效率.

方法	n = 50	n = 200	n = 1000
Charikar	0.005 s	0.019 s	0.109 s
Flow	0.340 s	4.242 s	88.148 s
SpecGreedy	0.043 s	0.075 s	0.317 s

注: 实验中边数 $m = 5n$.

准确程度

- 由于使用二分网络流的方法得到的是准确答案，所以我们采用比较找到最密子图的密度大小与二分网络流得到的密度的方法，定性分析不同模型在寻找最密子图中的准确程度.

方法	n = 10	n = 200	n = 1000
Charikar	1.000	1.000	1.000
Flow	1.000	1.000	1.000
SpecGreedy	0.725	0.788	0.774

注：实验中边数 $m = 5n$.

Idea

- 在我开展实验的时候，我主要尝试了几种方法在有稠密子图和随机图上的表现．比较奇怪的是，贪心的方法都能达到和二分网络流一样的精度，而 SpecGreedy 的精度表现不如贪心方法，这个是不符合我们预测的．
- 这可能是实现中存在一些 bug．排除掉这些 bug 后，我希望探索不同性质的图对这些检测方法的精度影响与效率影响．
- 例如，最密子图的结构或者是图中是否存在一些会让 SpecGreedy 陷入局部最优的子图，以及我们如何去规避它．