三角形计数算法调研

从精确算法到近似算法

刘丁玮

202218013229010

目录

目录

介绍

图

- 建模实体关系
- 社交网络、通信网络、生物网络等

三角形

- 重要拓扑结构
- 集聚性、同质性、传递性等

三角形计数

任务

- 统计三角形数量
- 枚举三角形并统计数量

应用

- 网络垃圾邮件检测 [1]
- 数据库查询优化 [2]
- · · ·

挑战

- 计算复杂度高，大规模场景开销巨大
- 图的偏斜分布 [3] 导致相同复杂度的算法性能差异大

目录

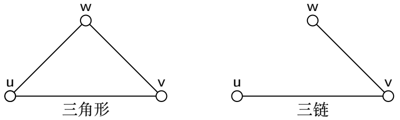
基本定义

符号

G, V, E, n, m	图、节点集、边集、节点数 $\|V\|$、边数 $\|E\|$
$u, e = (u, v)$	节点，边
$d(u), d_{\max}, adj(u)$	u 的度，图中最大的度、u 的邻居节点集合
$\Pi, \Pi^{\angle}, \Pi^{\triangle}$	三元组集合、三链集合、三角形集合

三元组、三角形与三链

- 三元组 $< u, v, w >$ 是以 v 为中心的长度为 2 的路径。
- 如果 uw 也有边相连，构成闭三元组，称为三角形。
- 如果 uw 没有边相连，构成开三元组，称为三链。



基本定义

$\Pi^{\triangle}$ 中三角形的不同节点贡献不同的三元组。Λ 表示图 G 中不同三角形的集合。三角形个数

$$t(G) = |\Lambda| = \frac{1}{3}|\Pi^{\triangle}| = \frac{1}{3}\sum_{v\in V}|\Pi_v^{\triangle}| \tag{1}$$

- 当且仅当图为全连接时，图中三角形数量最多，为 $\binom{n}{3}$。
- 从点的角度考虑，$t(G) = O(n^3)$。
- 从边的角度考虑，$t(G) = O(m^{\frac{3}{2}})$。

指标

传递率

- 三角形与三元组总数的比值，记为 $\gamma(G)$

$$\gamma(G) = \frac{|\Pi^{\Delta}|}{|\Pi|} = \frac{\Pi^{\Delta}}{\Pi^{\Delta} + \Pi^{\angle}} \tag{2}$$

- $t(G)$ 可以从其传递率计算，

$$t(G) = \frac{1}{3} \cdot \gamma(G) \cdot |\Pi| \tag{3}$$

目录

精确算法

最简单的算法

- 枚举三节点集，检测是否构成三角形。
- $O(n^3)$

计数算法（不枚举）

- 基于邻接矩阵乘
- $t(G) = \frac{1}{6}\mathrm{Tr}(A^3)$
- $O(n^\omega)$，ω 最低 2.373[4]。

AYZ 算法 [5]

节点集划分为 $V_{low} = \{v \in V : \mathrm{d}(v) \leq \beta\}$ 和 $V_{high} = V \setminus V_{low}$。
$\beta = m^{\omega-1/\omega+1}$。

- 对于 V_{low}，至多 $m \cdot \beta$ 条路径能构成三角形，复杂度为 $O(m\beta)$。
- 对于 V_{high}，讨论至多 $2m/\beta$ 个节点，复杂度为 $O((m/\beta)^\omega)$。

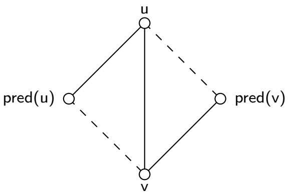
时间复杂度为 $O(m^{\frac{2\omega}{\omega+1}})$。$\omega = 3$ 时，为 $O(m^{\frac{3}{2}})$。

枚举算法

枚举算法

Itai 等 (1978) 提出的早期算法 [6]

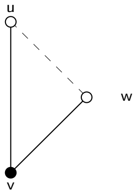
- 流程
  - 构造图 $G(V,E)$ 的生成树 $T(V,E_T)$
  - E_T 的每条边 (u,v)，检查 $(\text{pred}(u),v)\in E$? $(\text{pred}(v),u)\in E$?
  - 删除 T 的所有边更新 G
  - 迭代上述三步直到图中没有边
- $O(m^{\frac{3}{2}})$
- 需要修改图数据结构，实际开销很高



枚举算法

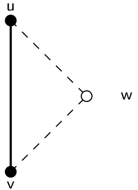
节点迭代

- 每个节点，检测任意两个不同邻居是否有边
- $\sum_{v \in V} \binom{d(v)}{2}$，$O(nd_{\max}^2)$



边迭代

- 每条边，检测两个端点的邻居的交集
- $O(md_{max})$



精确算法

计数算法	
$O(n^3)$	矩阵乘
$O(n^\omega)$	快速矩阵乘
$O(m^{\frac{2\omega}{\omega+1}})$	AYZ 使用快速矩阵乘

枚举算法	
$O(n^3)$	简单枚举
$O(nd_{\max}^2)$	节点迭代
$O(md_{\max})$	边迭代

- 最优的时间复杂度为 $O(m^{\frac{2\omega}{\omega+1}})$，$\omega$ 最低 2.373[4]
- 枚举算法中避免冗余统计可以有效减少运行时间 [7, 8]
- 对十亿级别边的图仍非常昂贵

目录

近似算法

基于图稀疏化的三角形计数方法

- 三角形以均匀概率采样

基于三元组采样的三角形计数方法

- 三元组以均匀概率采样

基于节点或边迭代法的三角形计数估计方法

- 节点、边以均匀概率采样

基于图稀疏化的三角形计数方法

思路

- 随机删除图中的一个边子集，得到稀疏图
- 从稀疏图的精确三角形计数推算原始图三角形计数

DOULION 算法变体 [9]

对给定图 G 的边以 p 的概率均匀采样，得到稀疏图 G_s。统计 G_s 中三角形和 G 中对应的三元组是三角形的三链

- 三角形的采样概率为 p^2
- $\hat{t}(G) = \frac{1}{p^2} \cdot t\left(G_{\mathrm{s}}\right)$

基于三元组采样的三角形计数方法

思路

- 均匀采样三元组，得到三元组集合 T，计算传递率的无偏估计 $\hat{\gamma}$

$$\hat{\gamma} = \frac{\sum_{t \in T} \mathbb{I}_{t \text{ is closed}}}{|T|} \tag{4}$$

- 使用 $\hat{\gamma}$ 估算原始图三角形计数

Schank 等 (2005a) 提出的算法 [10]

先以概率 $\frac{|\Pi_v|}{|\Pi|}$ 采样节点，再从得到的每个节点的三元组中随机返回一个，以 v 为中心的三元组随机返回概率为 $\frac{1}{|\Pi_v|}$

- 三元组的采样概率为 $\frac{1}{|\Pi|}$
- $\hat{t}(G) = \frac{1}{3} \cdot \hat{\gamma} \cdot |\Pi|$

基于节点或边迭代的三角形计数估计方法 [11]

- 采样部分节点或边，并统计它们所在三角形数量。
- 根据采样比例估算原始图三角形计数。

Graph	Sample Factor p	accuracy (%)	Variance of accuracy	APPROXEI speedup	PARAPPROXEI Speedups threads 4	8	16	32
Wiki-1	0.1	99.21	0.40	4.49	24.23	42.84	68.57	91.68
	0.01	98.2	2.36	33.54	239.94	418.75	664.37	837.74
Wiki-2	0.1	99.56	0.10	4.14	20.1	35.6	55.43	63.42
	0.01	98.95	0.49	32.42	199.56	351.02	548.01	614.27
Wiki-3	0.1	99.61	0.12	4.21	19.87	35.54	54.22	60.96
	0.01	98.72	1.93	32.85	198.44	341.15	515.91	592.26
Wiki-4	0.1	99.60	0.09	4.33	19.54	34.95	52.55	56.84
	0.01	98.28	1.61	33.71	197.13	346.92	504.8	547.29
Zewail	0.1	98.45	0.08	4.29	11.35	12.46	10.0	6.19
	0.01	92.26	12.28	9.92	40.14	30.01	15.14	10.03
Flikr	0.1	99.74	0.07	5.18	28.92	51.46	77.67	96.67
	0.01	99.43	0.19	33.26	277.44	501.83	730.21	796.85
EN	0.1	99.03	0.62	4.93	22.69	33.1	40.6	38.71
	0.01	97.03	5.82	18.83	147.4	164.77	143.46	97.96
EAT RS	0.1	98.21	1.66	4.15	17.74	24.79	27.92	23.62
	0.01	96.64	3.88	13.62	101.7	111.67	86.82	56.3

目录

总结

- 三角形在图分析领域有着重要作用
- 三角形计数和枚举是大图数据挖掘的重要任务、也可以作为更复杂分析任务的前置处理步骤
- 通过算法设计和实现优化，三角形计数精确算法的实际运行开销降到了可接受的范围
- 无需精确值的超大图场景下，近似算法有远超精确算法优越的性能

参考文献 I

[1] Luca Becchetti et al. "Efficient semi-streaming algorithms for local triangle counting in massive graphs". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 16–24.

[2] Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. "Reductions in streaming algorithms, with an application to counting triangles in graphs". In: *SODA*. Vol. 2. 2002, pp. 623–632.

[3] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". In: *science* 286.5439 (1999), pp. 509–512.

[4] François Le Gall. "Powers of tensors and fast matrix multiplication". In: *Proceedings of the 39th international symposium on symbolic and algebraic computation*. 2014, pp. 296–303.

[5] Noga Alon, Raphael Yuster, and Uri Zwick. "Finding and counting given length cycles". In: *Algorithmica* 17.3 (1997), pp. 209–223.

参考文献 II

[6] Alon Itai and Michael Rodeh. "Finding a minimum circuit in a graph". In: *SIAM Journal on Computing* 7.4 (1978), pp. 413–423.

[7] Thomas Schank and Dorothea Wagner. "Finding, counting and listing all triangles in large graphs, an experimental study". In: *International workshop on experimental and efficient algorithms*. Springer. 2005, pp. 606–609.

[8] Matthieu Latapy. "Main-memory triangle computations for very large (sparse (power-law)) graphs". In: *Theoretical computer science* 407.1-3 (2008), pp. 458–473.

[9] Roohollah Etemadi, Jianguo Lu, and Yung H Tsin. "Efficient estimation of triangles in very large graphs". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 1251–1260.

[10] Thomas Schank and Dorothea Wagner. "Approximating clustering coefficient and transitivity.". In: *Journal of Graph Algorithms and Applications* 9.2 (2005), pp. 265–275.

[11] Mahmudur Rahman and Mohammad Al Hasan. "Approximate triangle counting algorithms on multi-cores". In: *2013 IEEE International Conference on Big Data*. IEEE. 2013, pp. 127–133.