



超图上 (k, t) -超核维护问题的快速算法

献给刘桂真教授 80 寿辰

张飞腾, 刘彬*, 方奇志

中国海洋大学数学科学学院, 青岛 266000

E-mail: zhangfeiteng@stu.ouc.edu.cn, binliu@ouc.edu.cn, qfang@ouc.edu.cn

收稿日期: 2023-11-22; 接受日期: 2024-05-07; 网络出版日期: 2024-09-09; * 通信作者

国家自然科学基金 (批准号: 11971447) 资助项目

摘要 超图 H 上的 (k, t) -超核是最小度不小于 k 且每条超边满足关于 t 的比例约束的极大子超图, 其中比例约束是指超边在子超图中包含的顶点数与它在 H 中包含顶点数之比不小于 t . 在参数 t 为常数的前提下, 每个顶点的 t -超核数为其所在所有 (k, t) -超核中的最大的 k . 本文研究在动态超图上更新每个顶点的 t -超核数这一超核维护问题. 首先, 本文刻画单超边插入或删除时 t -超核数发生变化的顶点所满足的必要条件, 据此设计出该情形下的维护算法, 并给出其并行算法. 然后, 本文研究批量插入或删除超边的情形. 定义主导顶点不交超边集 DDHS (dominant vertices disjoint hyperedge set) 并证明在超图上一次插入或删除一个 DDHS 后每个顶点 t -超核数至多变化 1. 最后, 本文通过 DDHS 设计出并行批量 (k, t) -超核维护算法. 相比于单超边改变的维护算法, 该算法效率更高.

关键词 超图 (k, t) -超核 超核维护 并行算法 批量处理

MSC (2020) 主题分类 05C65, 05C85, 68R10

1 引言

图和网络在描述个体之间关系上发挥了重要作用, 并在社交网络、交通和电信等领域有着广泛应用. 由于实际场景往往是复杂的, 且个体以及个体之间的关系拥有各种属性, 不同类型的图随之出现, 如赋权图、有向图、概率图和超图等. 超图作为近些年的研究热点, 不仅其自身的性质受到了学者们的广泛关注, 而且它包含的比一般图更加丰富的内容使其可以应用到更多的场景中. 在真实世界中, 一个团体通常包括不止两个个体. 传统意义上的图并不能很好地描述这些团体内个体之间的关系, 而超图的出现完美地解决了这个问题. 例如, 一个人可能包含在多个团队中, 若用普通图刻画团队成员之间的合作关系, 则该图呈现出的只有两人是否在同一个团队合作过, 却不能得到每个团队的全部成员. 而用超边代表团队, 超边内的顶点代表团队成员的超图来表示团队合作信息, 不仅可以得到普通图上所呈现的所有信息, 而且能得到每个团队的全部成员.

英文引用格式: Zhang F T, Liu B, Fang Q Z. Fast algorithms for the (k, t) -hypercore maintenance problem on hypergraphs (in Chinese). Sci Sin Math, 2024, 54: 1865–1888, doi: 10.1360/SSM-2023-0319

在普通图中, 密集子图的挖掘与关键节点的探找是分析大规模图网络性质的一种重要手段. 如何高效地挖掘密集子图与探找关键节点是具有挑战性的一个问题, 该问题的相关理论成果被应用在社交网络^[1,10]和生物信息学^[4]等领域的实际应用中. 常见的密集子图有团 (clique)、 k -核 (k -core) 和 k -桁架 (k -truss) 等, 其中 k -核与 k -桁架分别有线性时间^[2]和多项式时间^[5]的分解算法, 因此这两种子图结构受到了较多的关注. 本文主要研究“核”这一类子图结构. 图的一个 k -核^[14]是指每个顶点的度均大于等于 k 的极大子图, 其中 k 为非负整数. 任给一个顶点 u , 如果 u 存在于一个 k -核中, 而不存在于任何一个 $(k+1)$ -核中, 则定义顶点 u 的核数 (core number) 为 k . 对于每个 k , 求出一个图上所有的 k -核即为核分解问题 (the core decomposition problem), 这等价于求出图上每个顶点的核数. 在实际场景中, 个体之间的关系是在不断变化的, 这导致了它们对应的图也在动态变化. 因此, 这需要对动态图上顶点的核数进行更新, 这就是核维护问题 (the core maintenance problem). 虽然对整个图进行处理的分解算法能够解决这一问题, 但因为图上只有一个连通子图的顶点属性发生了变化^[13], 处理整个图会提高计算成本. 因此, 出现了解决动态图核数更新的高效核维护算法, 并且它在子图探找问题中发挥着重要的作用. 接下来, 学者们先后设计了一次插入或删除一条边的核维护算法^[13,16], 一次插入或删除一组边的并行核维护算法^[6,8,15]等多种场景的核维护算法. 另外, 其他图结构同样也有对应的分解与维护问题.

对于超图, Ramadan 等^[12]在研究酵母蛋白问题时, 首次提出了复杂网络上的 k -超核的概念, 即超图上每个顶点的度均大于等于 k 的极大子超图, 其中 k 为非负整数. 如果顶点 u 存在于一个 k -超核中, 而不存在于任何一个 $(k+1)$ -超核中, 则定义顶点 u 的超核数 (hypercore number) 为 k . 后来, Leng 等^[9]提出了一种关于超边数的线性时间的超图核分解算法. 他们考虑的是从一条超边中移除一个顶点之后, 若该超边非空则该超边仍然存在的情形. Luo 等^[11]设计了一种超图的超核分解算法, 他们考虑的超核是一种顶点导出子超图, 即在子超图中删除一个顶点后, 该顶点所关联的所有超边随即不再存在. 然而, 这两篇文献中的超核分解算法的想法和理论与文献 [2] 中首次提出的普通图上核分解算法的思想几乎一致: 如果度小于 k 的顶点被不断地从原图或超图中删除, 直到剩下的顶点的最小度大于等于 k , 则剩下的图或超图就是一个 k -核或 k -超核. 这两种分解算法只在处理删除超边的过程上有所不同. 随后, 文献 [11] 给出了一次插入或删除一条超边的顶点导出超核维护算法, 文献 [7] 给出了在多条超边中一次改变一组顶点的并行超核维护算法.

给定超图 H , 文献 [3] 在上述 k -超核的基础上增加了超边需满足的比例约束, 定义出 H 的 (k, t) -超核这一子超图结构, 其中 $t \in (0, 1]$. 比例约束是指 (k, t) -超核中任意一条超边在核中包含的顶点的个数与在 H 中包含的顶点的个数的比值不小于 t . 接下来, 文献 [3] 分别给出了 k 和 t 均为常数、只有 t 为常数和只有 k 为常数 3 种不同条件下的 (k, t) -超核分解算法, 这 3 种条件分别对应不同的场景.

本文研究动态超图 H 上 t 为常数的 (k, t) -超核维护算法. 在 t 为常数的基础上, 对于超图 H 的任意一个顶点 u , 若 u 存在于一个 (k, t) -超核中而不存在于任何一个 $(k+1, t)$ -超核中, 则定义顶点 u 的 t -超核数 (t -hypercore number) 为 k . 作为一种密集子超图结构, (k, t) -超核可为密集子图挖掘问题、关键节点探测等实际领域提供解决方案. 在密集子图挖掘问题中, 对于 t 为常数的 (k, t) -超核, 其内部满足比例约束的超边比 k -超核中的超边有更强的内部连结性, 结构更紧密. 同时, 对参数 t 取不同的值可以调节这种内部的紧密性, t 越大, 其内部超边的参与程度越高. 当 $t = 1$ 时, (k, t) -超核即为超图上的顶点导出 k -超核. 在关键节点选取问题中, 利用 (k, t) -超核可选取到比利用 k -超核更有代表性、重要程度更高的节点. 例如, 在团队合作网络中, 超边代表团队, 超边中顶点代表该团队的成员, 顶点超核数或 t -超核数越大代表该顶点在超图中越重要. 利用 k -超核选取的关键节点仅考虑人物参与的团队数目, 却未考虑关联团队项目的完成质量. 若项目团队的成员参与比例越高, 项目完成质量越高, 则使

用 t -超核数探找的关键节点是能同时保证项目完成质量达到一定要求的关键节点. 另外, 动态超图的 (k, t) -超核维护算法可快速更新顶点 t -超核数, 以此快速为 (k, t) -超核相关应用问题提供方案.

接下来研究超图动态变化后, 超图上每个顶点的 t -超核数更新这一 (k, t) -超核维护问题. 该问题的难点主要有两个: 一是如何确定 t -超核数发生变化的顶点, 二是它们的变化量是多少. 以上两个难点是核维护问题均需要考虑的问题, 但不同的核结构的性质差异会导致核数变化顶点所满足的性质有所不同. 探找下述两种情形动态超图 t -超核数变化顶点所满足的性质是本文解决上述困难的创新之处. 本文主要成果如下.

- 本文首先考虑在超图上一次插入或删除一条超边 e_0 的情形. 发现在该情形下, 超图上每个顶点的 t -超核数变化量不超过 1. 但是, t -超核数发生变化的顶点的 t -超核数会有多个值, 这由超边 e_0 有效值集合 $VVS_t(e_0)$ 决定. 同时, 本文刻画出 t -超核数发生变化的顶点 u 所满足的必要条件: u 必通过一条满足顶点 t -超核数和超边临界有效值均与顶点 u 的 t -超核数相等、且顶点 t -超核数均发生变化的路 P 与 e_0 主导顶点集 $DVS_t(e_0)$ 中的顶点相连接 (传递性). 与直接使用分解算法相比, 以上述性质设计的局部更新 t -超核数的 (k, t) -超核维护算法, 可以极大降低时间消耗. 根据上述必要条件, 本文利用广度优先搜索分别设计出一插入一条超边和一次删除一条超边的多项式时间的维护算法. 最后, 本文证明这两个算法对不同 t -超核数的处理可以做到互不影响, 因此设计出并行算法来提高 (k, t) -超核维护的效率.

- 本文接下来研究在超图上批量处理一组超边 E_0 的情形. 一次插入或删除一组任意的超边, 首先, 顶点 t -超核数的变化量难以确定. 其次, t -超核数变化的顶点的传递性也会变得十分复杂. 于是, 本文提出主导顶点不交超边集 DDHS (dominant vertices disjoint hyperedge set) 这一超边集合的定义. 一次插入或删除一个 DDHS 后, 每个顶点的 t -超核数变化量仍不超过 1. 那么, 就可以将 E_0 分解成多个 DDHS 来依次处理. 同时, 一次插入或删除一个 DDHS 后, t -超核数发生变化的顶点 u 也满足单超边改变情形的必要条件. 接下来, 本文依据上述结论设计出插入或删除一组超边 E_0 的多项式时间的并行 (k, t) -超核维护算法. 该算法首先利用贪心思想从 E_0 中剥离出一个 DDHS, 然后将该 DDHS 在超图上插入或删除, 接下来并行地去探找 t -超核数变化的顶点并更新所有顶点 t -超核数. 重复上述操作直到 E_0 成为空集, 算法终止. 另外, 与一条条地处理超边相比, 一次插入或删除一个 DDHS 来进行 (k, t) -超核维护会减少重复运算, 效率更高.

本文余下内容安排如下. 第 2 节介绍超图上 (k, t) -超核的相关定义与现有成果, 其中包括 (k, t) -超核的分解算法. 第 3 节介绍并证明在超图上插入或者删除超边 e_0 后, 超图上 t -超核数变化的顶点的性质, 用以支撑第 4 节设计单超边插入或删除的 (k, t) -超核维护算法. 第 5 节研究在超图上一次插入或删除一组超边的 (k, t) -超核的维护问题. 首先提出一种特殊的超边集 DDHS, 一次插入或删除一个 DDHS, 顶点 t -超核数变化量仍至多为 1, 接下来设计一次插入或删除一组超边的并行批量 (k, t) -超核维护算法. 第 6 节给出本文结论和展望.

2 相关知识准备

2.1 基础定义

$H = (V, E)$ 是定义在具有有限个元素的顶点集 V 上的一个无向超图. 超边集 $E = \{e_i \mid i \in I_E, e_i \subseteq V\}$ 是顶点集 V 的一个子集族, 其中 I_E 是关于 E 的一个有限的指标集. 分别用 m 和 n 来表示超边集 E 和顶点集 V 的基数, 因此 $n = |V|$, $m = |E| = |I_E|$. 本文研究的超图不考虑只包含一个顶点

的超边 (环).

设 u 是 V 中的一个顶点. 顶点 u 所关联的所有超边构成集合 $E_H(u)$. 顶点 u 的度

$$\deg_H(u) = |E_H(u)|.$$

进而

$$\delta(H) = \min\{\deg_H(u) \mid u \in H\}, \quad \Delta(H) = \max\{\deg_H(u) \mid u \in H\}.$$

区别于普通图, 本文定义超图 H 中的每一条超边 e 的度为该超边的基数, 用 $\text{edeg}_H(e)$ 表示, $\text{edeg}_H(e) = |e|$.

给定超图 $S = (V', E')$, 其中 $V' \subseteq V$, 如果对于每个 $i \in I_{E'}$, 超边 $e_i \in E$ 并且 $e'_i \subseteq e_i$, 则称 S 为超图 H 的一个子超图 (subhypergraph). 称 e'_i 为 e_i 在子超图 S 中的导出超边 (induced hyperedge). 接下来, 给出一些定义.

定义 2.1 (k -超核) 设 $H = (V, E)$ 是一个超图, $S = (V', E')$ 是 H 的一个子超图. 如果 S 满足如下条件:

- (1) (度约束) 对于每一个顶点 $u \in V'$, 有 $\deg_S(u) \geq k$, 其中 k 是一个非负整数;
- (2) (极大性约束) S 是一个满足条件 (1) 的极大子超图,

则称 S 是一个 k -超核, 用 HC_k 表示.

一个密集子图 (子超图) 结构通常拥有强的内部连结性, 同时它与其外部的连接相对稀疏. 基于该特点, 为了寻找比 k -超核内部连结性更强的子超图结构, Bu 等^[3] 通过在子超图的超边上添加比例约束, 提出了名为 (k, t) -超核的子超图结构.

定义 2.2 ((k, t) -超核) 设 $H = (V, E)$ 是一个超图, $S = (V', E')$ 是 H 的一个子超图. 如果 S 满足如下条件:

- (1) (度约束) 对于每一个顶点 $u \in V'$, 有 $\deg_S(u) \geq k$, 其中 k 是一个非负整数;
- (2) (比例约束) 对于每一个 $i \in I_{E'}$, 有

$$\frac{\text{edeg}_S(e'_i)}{\text{edeg}_H(e_i)} \geq t,$$

其中 $t \in (0, 1]$;

- (3) (极大性约束) S 是一个满足条件 (1) 和 (2) 的极大子超图,

则称 S 是一个 (k, t) -超核, 用 $HC_{(k, t)}$ 表示.

在一个超图 H 中, 每个顶点 u 的超核数 (hypercore number) 用 $\text{core}(u)$ 表示, 定义为

$$\text{core}(u) = \max\{k \mid u \in V(HC_k), HC_k \subseteq H\}. \quad (2.1)$$

对于顶点 u , 它是唯一确定的, 并且该属性能够反映出 u 所处的超图上区域的密集程度以及该顶点在超图上的重要性. 然而, 由于 (k, t) -超核中参数 t 的存在, 利用该子超图结构直接为每个顶点简单地定义超核数是存在困难的. 因此, 针对该问题, 本文引入 t -超核数^[3] 这一概念来描述与顶点的超核数相对应的属性.

定义 2.3 (t -超核数) 设 t 是一个实数且 $t \in (0, 1]$. 假设 $S = (V', E')$ 是超图 $H = (V, E)$ 包含顶点 u 的一个 (k, t) -超核, 其中 $u \in V'$. 如果 H 中没有其他包含顶点 u 的 $(k+1, t)$ -超核, 则顶点 u 的 t -超核数定义为 k , 用 $\text{core}_t(u)$ 表示, 即

$$\text{core}_t(u) = \max\{k \mid u \in V(HC_{(k, t)}), HC_{(k, t)} \subseteq H\}. \quad (2.2)$$

当文中没有特别的解释时, (k, t) -超核中的参数 t 是 $(0, 1]$ 上的实数. 给定超图 H 和顶点 u , 设 $t_1 < t_2$, 则在 H 上, $\text{core}_{t_1}(u)$ 与 $\text{core}_{t_2}(u)$ 可能不相同. 从另外一个角度解释, H 中 $(\text{core}_{t_1}(u), t_1)$ -超核与 $(\text{core}_{t_2}(u), t_2)$ -超核可能是不同的子超图. 因此, 对于任意一个常数 $t^* \in (0, 1]$, 称 H 中包含顶点 u 的 $(\text{core}_{t^*}(u), t^*)$ -超核为顶点 u 的极大- (k, t^*) -超核 (max- (k, t^*) -hypercore), 用 $HC_{t^*}^u$ 表示.

命题 2.1 设 k_1 和 k_2 是两个非负整数, 其中 $k_1 < k_2, t_1 < t_2$. 在超图 H 上, 有如下结论成立:

- (1) 对于任意一个非负整数 k , 有 $HC_{(k, t_2)} \subseteq HC_{(k, t_1)}$;
- (2) 对于任意一个 $t \in (0, 1]$, 有 $HC_{(k_2, t)} \subseteq HC_{(k_1, t)}$.

证明 (1) 设 k 是一个非负整数. 由定义 2.2 可知, 对于任意 $u \in HC_{(k, t_2)}$, 有 $\deg_{HC_{(k, t_2)}}(u) \geq k$, 并且对于任意 $e'_i \in HC_{(k, t_2)}$, 有

$$\frac{\text{edeg}_{HC_{(k, t_2)}}(e'_i)}{\text{edeg}_H(e_i)} \geq t_2.$$

因为 $t_1 < t_2$, 所以对于任意 $e'_i \in HC_{(k, t_2)}$, 有

$$\frac{\text{edeg}_{HC_{(k, t_2)}}(e'_i)}{\text{edeg}_H(e_i)} > t_1.$$

因此, $HC_{(k, t_2)} \subseteq HC_{(k, t_1)}$.

(2) 结论 (2) 的证明与结论 (1) 类似, 此处不再详述. $\square$

例 2.1 在图 1(a) 中, 超图 $H = (V, E)$ 的顶点集为 $V = \{u_1, u_2, \dots, u_{13}\}$, 超边集为 $E = \{e_1 = \{u_1, u_2, u_3, u_4, u_5\}, e_2 = \{u_5, u_6, u_7, u_8\}, e_3 = \{u_6, u_7, u_8, u_9, u_{10}\}, e_4 = \{u_8, u_{11}\}, e_5 = \{u_1, u_{12}, u_{13}\}, e_6 = \{u_3, u_{13}\}\}$. 设 $t = \frac{1}{2}$, 根据 (k, t) -超核的定义可知, 整个超图 H 是一个 $(1, \frac{1}{2})$ -超核. 接下来, 删除超图所有度小于 2 的顶点, 并且不断移除所有环或不满足比例约束的超边, 得到图 1(b). 它是一个 $(2, \frac{1}{2})$ -超核. 同时, 超图 H 上没有其他 $k > 2$ 的 $(k, \frac{1}{2})$ -超核. 因此, 图 1(b) 上所有顶点的 $\frac{1}{2}$ -超核数是 2, 而其他顶点的 $\frac{1}{2}$ -超核数是 1.

2.2 现有结果

本节介绍超图上与 (k, t) -超核相关的一些已有的基础性的结果, 主要分为超核的分解问题与维护问题, 这也是本文所关注的问题.

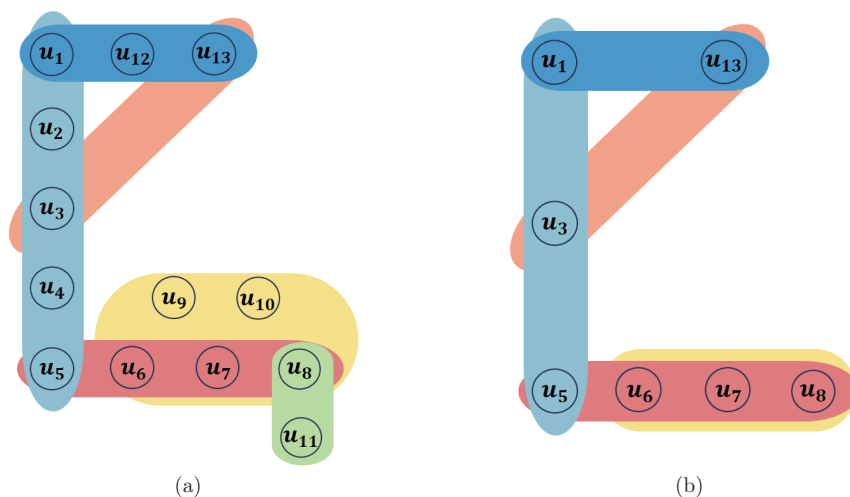


图 1 (网络版彩图) (a) 超图 H ; (b) 超图 H 上的一个 $(2, \frac{1}{2})$ -超核

2.2.1 超核的分解问题

引言中提到文献 [9] 和 [11] 先后给出了超图的超核分解算法, 这两篇文献的超核分解算法的大致过程如下. 给定一个超图 H . 首先, 所有的顶点需要根据它们的度进行递增排序. 接下来, 移除 H 中度最小的顶点并赋这些顶点的超核数为 $\delta(H)$. 同时, 更新 H 中剩下子超图顶点的度. 在子超图上重复上述过程直到子超图成为一个零超图, 算法终止. 整体来说, 这两个分解算法的时间复杂度都为 $O(m)$.

Bu 等^[3] 为超图 H 设计了 3 种形式的 (k, t) -超核分解算法. 第一个是在参数对 (k, t) 均为常数的前提下, 搜索 H 中的 (k, t) -超核. 该算法可以输出不同参数对约束下的目标子超图. 第二个是在参数 t 为常数的前提下, 计算超图 H 上所有顶点的 t -超核数. 第三个是在参数 k 为常数的前提下, 计算超图 H 上所有顶点的 k -比例 (k -fraction), 其中一个顶点 u 的 k -比例是指满足包含顶点 u 的所有 (k, t) -超核中的最大的 t . 后两种算法输出每个顶点 u 的属性值, 这两种属性值都能够表示 u 所在的子超图中密集程度较高的子超图的密集程度的大小以及顶点 u 在 H 中的重要程度.

为了在超图 H 上计算每个顶点的 t -超核数并展示文献 [3] 中第二个超核分解算法的计算过程的细节, 本文用算法 1 将其算法的伪代码进行展示.

算法 1 在超图 H 上计算 $\text{core}_t(\cdot)$ ^[3]

输入: 超图 $H = (V, E)$, V 中每个顶点 u 的度 $\deg_H(u)$, E 中每条超边 e 的度 $\text{edeg}_H(e)$, $t \in (0, 1]$

输出: V 中每个顶点 u 的 $\text{core}_t(u)$

```

1:  $S = (V, E) \leftarrow H$ ;  $R \leftarrow \emptyset$ ;
2: while  $V \neq \emptyset$  do
3:    $c = \min\{\deg_S(u) \mid u \in V\}$ ;
4:    $R \leftarrow \{v \in V \mid \deg_S(v) = c\}$ ;
5:   for  $R$  中的每个顶点  $u$  do
6:      $V \leftarrow V \setminus u$ ;  $\text{core}_t(u) \leftarrow c$ ;
7:     for  $E_S(u)$  中的每条超边  $e$  do
8:        $e \leftarrow e \setminus u$ ;  $\text{edeg}_S(e) \leftarrow \text{edeg}_S(e) - 1$ ;
9:       if  $\text{edeg}_S(e) < \max\{2, t * \text{edeg}_H(e)\}$  then
10:         $E \leftarrow E \setminus e$ ;
11:        for  $e$  中的每个顶点  $v$  do
12:           $\deg_S(v) \leftarrow \max\{\deg_S(v) - 1, c\}$ ;
13:          if  $\deg_S(v) = c$  then
14:             $R \leftarrow R + v$ ;
15:          end if
16:        end for
17:      end if
18:    end for
19:  end for
20: end while
21: 返回:  $V$  中每个顶点  $u$  的  $\text{core}_t(u)$ 

```

类似于文献 [9, 11] 中的分解算法, 算法 1 按照度从小到大的顺序不断地删除顶点, 并在删除顶点时不断地给顶点赋予 t -超核数, 更新邻点的度, 移除不满足条件的超边. 特别地, 由于参数 t 的存在, 在移除超边时算法需要考虑更严格的删边条件 (算法 1 第 9 行).

设 $H = (V, E)$ 是一个超图, 并定义

$$\Delta_E(H) = \max\{\text{edeg}_H(e) \mid e \in H\}, \quad c_{\max}(H) = \max\{\text{core}_t(u) \mid u \in H\}.$$

然后, 算法 1 第 3 和 4 行运行至多 $c_{\max}(H)$ 次. 接下来, 对于每一条超边 $e \in E$, 在第 5–19 行它至多被访问 $|e| - \max\{2, \lceil t|e| \rceil\} + 1$ 次. 对于每一个顶点 $u \in V$, 它只被添加至 R 一次并且从 R 中被移除一次. 因此, 算法 1 的时间复杂度是 $O(n \cdot c_{\max}(H) + m + (1-t) \cdot m \cdot \Delta_E(H))$ [3].

2.2.2 超核维护问题

文献 [6–8, 13, 15, 16] 研究了动态图上的核维护问题, 并给出了高效的核维护算法. 受这些工作的启发, Luo 等 [11] 在动态超图上设计了遍历超核维护算法, 该算法只考虑一次插入或删除一条超边的情形.

接下来, 以在超图 H 上插入或者删除一条超边 e_0 为例来介绍文献 [11] 中的超核维护算法. 首先, 定义一条超边 e 的超核数 $\text{core}E(e)$ [11] 为

$$\text{core}E(e) = \min\{\text{core}(u) \mid u \in e\}. \quad (2.3)$$

整体上, 超核维护算法在超图的一个局部子超图上更新每个顶点和每条超边的超核数, 这比超核分解算法的时间成本更低. 当 e_0 被插入或者删除时, Luo 等 [11] 发现每个顶点和每条超边的超核数变化量至多为 1, 并且只有超核数为 $\text{core}E(e)$ 的顶点或超边可能会变化. 然后, 所有超核数可能会发生变化的顶点可以从插入的超边 e_0 中超核数为 $\text{core}E(e)$ 的顶点通过深度优先搜索找到. 同时, 所有被寻找到的顶点的超核数是否会发生变化需要通过每个顶点的支持度 (support degree) [11] 来判断. 定义顶点 u 的支持度 $\text{sup}(u)$ 为

$$\text{sup}(u) = |\{e \mid u \in e, \text{core}(u) \leq \text{core}E(e)\}|. \quad (2.4)$$

接下来, 对于插入超边 e_0 的情形, 在搜索到的所有顶点中, 删除所有支持度小于 $\text{core}E(e_0) + 1$ 的顶点并更新剩下顶点的支持度, 直到剩下所有顶点的支持度都不小于 $\text{core}E(e_0) + 1$. 剩下的顶点即为插入 e_0 后超核数增加 1 的顶点. 对于删除超边 e_0 的情形, 在搜索到的所有顶点中, 删除所有支持度小于 $\text{core}E(e_0)$ 的顶点并更新剩下顶点的支持度, 直到剩下所有顶点的支持度都不小于 $\text{core}E(e_0)$. 所有删除的顶点即为删除 e_0 后超核数减小 1 的顶点. 最后, 更新所有包含超核数发生变化的顶点的超边的超核数.

2.3 本文研究的问题

给定一个超图 H , 设在删除一个顶点后, 倘若该顶点关联的超边不是环, 则该超边仍存在于子超图上. 这是本文研究内容的前提条件. 上文中, (k, t) -超核的分解算法已经给出. 接下来, 本文进一步探讨动态超图上的 (k, t) -超核维护问题. 具体而言, 假设 t 是 $(0, 1]$ 上的一个常数, 并且超图 H 上所有顶点的 t -超核数是已知的. 然后, 设计出在超图 H 上插入或删除超边后更新每个顶点 t -超核数的高效超核维护算法.

3 理论结果

超核维护问题的目标是在动态超图上设计出节省时间的更新 t -超核数的算法. 为了解决该问题, 本节对以下两个问题进行刻画: (1) 超图插入或删除一条超边后, 具有什么性质的顶点 t -超核数将会变化; (2) 每个顶点 t -超核数的变化量是多少. 基于上述理论结果, 本文设计出 (k, t) -超核维护算法.

设 $H = (V, E)$ 是一个超图. 本文用 H' 来表示 H 改变后得到的新超图, 用 $\text{core}'_t(u)$ 表示顶点 u 在 H' 上的 t -超核数. 需要注意的是, 在本文中, e 出现在 (存在于) 一个子超图中是指它的导出超边出现在这个子超图中, 插入的超边是原超图中不存在的超边.

首先, 本节给出关于在超图 H 上插入或删除一条超边后, t -超核数变化的顶点的 t -超核数变化量是多少的结论.

定理 3.1 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 在 H 中插入或删除一条超边 e_0 . 然后, 对于任意一个顶点 $u \in V$, 都有

$$|\text{core}_t(u) - \text{core}'_t(u)| \leq 1.$$

证明 对于将一条超边 e_0 插入 H 的情形, 首先, 插入一条超边后显然有 $\text{core}_t(u) \leq \text{core}'_t(u)$. 接下来, 假设顶点 u 的 t -超核数变化情形为 $\text{core}'_t(u) - \text{core}_t(u) \geq 2$. 用 $HC_t'^u$ 表示 H' 中顶点 u 的极大 (k, t) -超核, 其中 $\delta(HC_t'^u) = \text{core}'_t(u)$. 超边 e_0 的导出超边 e'_0 一定存在于 $HC_t'^u$ 中. 否则, $HC_t'^u$ 是 H 中的 $(\text{core}'_t(u), t)$ -超核, 这与 $\text{core}'_t(u) - \text{core}_t(u) \geq 2$ 矛盾. 设 $S = HC_t'^u - e'_0$. S 是 H 包含顶点 u 的子超图, 并且 $\delta(S) \geq \text{core}'_t(u) - 1 > \text{core}_t(u)$. 同时, 对于任意一条超边 $e'_i \in S$, 有

$$\frac{\text{edeg}_S(e'_i)}{\text{edeg}_H(e_i)} \geq t.$$

然后, $\text{core}_t(u) \geq \text{core}'_t(u) - 1$, 矛盾. 因此, 插入超边 e_0 后 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$.

对于删除一条超边的情形, 它可以看作是插入一条超边的逆过程. 因此, 删除超边 e_0 后, $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$. $\square$

定义顶点集 V^* , 它由插入或删除超边后 t -超核数会发生变化的所有顶点构成. 接下来, 本节给出 V^* 中顶点的一些性质.

定义 3.1 (临界有效值、有效值集和主导顶点集) 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 顶点集 $U \subseteq V$. 顶点集 U 在 t 约束下的临界有效值用 $\text{Cval}_t(U)$ 表示, 它满足

$$\begin{cases} |\{u \mid u \in U, \text{core}_t(u) > \text{Cval}_t(U)\}| < \max\{2, t|U|\}, \\ |\{u \mid u \in U, \text{core}_t(u) \geq \text{Cval}_t(U)\}| \geq \max\{2, t|U|\}. \end{cases} \quad (3.1)$$

顶点集 U 在 t 约束下的有效值集用 $\text{VVS}_t(U)$ 表示, 定义为

$$\text{VVS}_t(U) = \{\text{core}_t(u) \mid u \in U, \text{core}_t(u) \leq \text{Cval}_t(U)\}. \quad (3.2)$$

顶点集 U 在 t 约束下的主导点集用 $\text{DVS}_t(U)$ 表示, 定义为

$$\text{DVS}_t(U) = \{u \mid u \in U, \text{core}_t(u) \in \text{VVS}_t(U)\}. \quad (3.3)$$

设 $HC_{(k,t)}$ 是超图 $H = (V, E)$ 中的一个 (k, t) -超核, $t \in (0, 1]$. 对于一条超边 e , 由超边的定义可知它是一个顶点集. 从而有 $\text{Cval}_t(e) = \max \text{VVS}_t(e)$. 如果 $\text{Cval}_t(e) \geq k$, 则 e 出现在 $HC_{(k,t)}$ 中. 这是因为 e 满足 (k, t) -超核的比例约束. 另外, 对于超图的任意一条超边 e , 其临界有效值可以通过算法 1 计算得到: 在第 10 行插入

$$\text{Cval}_t(e) = c$$

即可. 如果 e_0 作为一条被插入的超边出现, 则用 H 中顶点的 t -超核数来定义 e_0 的临界有效值.

引理 3.1 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 超边 e_0 被插入到 H 中 (或从 H 中删除). 如果顶点 u 的 t -超核数满足 $\text{core}_t(u) \notin \text{VVS}_t(e_0)$, 则 $\text{core}'_t(u) = \text{core}_t(u)$.

证明 首先证明在超图 $H = (V, E)$ 中插入一条超边 e_0 的情形, 其中 $H' = H + e_0$, $e_0 \notin E$. 使用反证法证明该结论. 插入超边后, 假设顶点 u 满足 $\text{core}_t(u) \notin \text{VVS}_t(e_0)$ 并且 $\text{core}'_t(u) - \text{core}_t(u) = 1$, 即顶点 u 的 t -超核数增加. 子超图 HC_t^u 与 $HC_t'^u$ 分别是顶点 u 在超图 H 与 H' 中的极大- (k, t) -超核. 通过 (k, t) -超核的定义可知, $\delta(HC_t^u) = \text{core}_t(u)$ 且 $\delta(HC_t'^u) = \text{core}'_t(u)$. 然后有 $e'_0 \in HC_t'^u$, 其中超边 e'_0 是 e_0 在 $HC_t'^u$ 的导出超边. 否则, 插入超边后 $\text{core}_t(u) = \text{core}'_t(u)$, 矛盾. 接下来分两种情形讨论. (1) 如果 $\text{core}_t(u) > \max \text{VVS}_t(e_0)$, 则由定理 3.1 可知 $\text{DVS}_t(e_0)$ 中所有顶点不存在于 $HC_t'^u$ 中. 由定义 3.1 可知 $e'_0 \notin HC_t'^u$, 这与超边 $e'_0 \in HC_t'^u$ 矛盾. (2) 如果 $\text{core}_t(u) < \max \text{VVS}_t(e_0)$, 设 $S = HC_t'^u - e'_0$, 则 $u \in S$. 然后, 超图 H 的 $(\text{core}_t(u) + 1, t)$ -超核一定是 S 的子超图. 对于任意一个 e'_0 中的点 v , 因为 $\text{core}_t(v) > \text{core}_t(u)$, 所以有 $\deg_S(v) \geq \text{core}_t(u) + 1$. 因为 $\delta(HC_t'^u) \geq \text{core}_t(u) + 1$, 所以 $\delta(S) \geq \text{core}_t(u) + 1$. 因此, S 是 H 中 $(\text{core}_t(u) + 1, t)$ -超核的一个子超图. 由此可得 $\text{core}_t(u) \geq \text{core}_t(u) + 1$, 矛盾. 综上, 满足 $\text{core}_t(u) \notin \text{VVS}_t(e_0)$ 顶点 u 的 t -超核数在插入 e_0 后不会变化.

下面证明从超图 $H = (V, E)$ 中删除一条超边 e_0 的情形, 其中 $e_0 \in E$. 令 $H' = H - e_0$, $u \in V$ 且 $\text{core}_t(u) \notin \text{VVS}_t(e_0)$. 子超图 HC_t^u 与 $HC_t'^u$ 分别是顶点 u 在超图 H 与 H' 中的极大- (k, t) -超核. 当顶点 u 满足 $\text{core}_t(u) > \max \text{VVS}_t(e_0)$ 时, e_0 不会出现在 HC_t^u 中. 因此, 删除 e_0 后 $\text{core}_t(u)$ 不会减小. 接下来使用反证法证明. 如果 $\text{core}_t(u) < \max \text{VVS}_t(e_0)$, 则 $\text{core}_t(u)$ 不会减小. 假设顶点 u 满足 $\text{core}_t(u) < \max \text{VVS}_t(e_0)$ 且删除超边后 $\text{core}_t(u) - \text{core}'_t(u) = 1$. 故 $e'_0 \in HC_t^u$, 其中 e'_0 是 e_0 在 HC_t^u 中的导出超边. 设 $S = HC_t^u - e'_0$, 然后有 $u \in S$. 因为对于任意一个顶点 $v \in e'_0$ 都有 $\text{core}_t(v) > \text{core}_t(u)$, 则 $\deg_{HC_t^u}(v) > \text{core}_t(u)$. 因此, 对于任意一个顶点 $v \in e'_0$ 都有 $\deg_S(v) \geq \text{core}_t(u)$. 对于任意一个顶点 $w \in S$, 都有 $\deg_S(w) \geq \text{core}_t(u)$. 因此, S 在 H' 中是 $(\text{core}_t(u), t)$ -超核的一个子超图, 且有 $\text{core}'_t(u) \geq \text{core}_t(u)$, 矛盾. 所以, 满足 $\text{core}_t(u) \notin \text{VVS}_t(e_0)$ 的顶点 u 的 t -超核数在删除 e_0 后不会变化. $\square$

引理 3.2 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 超边 e_0 被插入到 H 中 (或从 H 中删除). 如果 V 中存在一个顶点 u 满足 $|\text{core}_t(u) - \text{core}'_t(u)| = 1$, 则 e_0 中至少存在一个顶点 v 满足 $\text{core}_t(v) = \text{core}_t(u)$, 且在插入 (删除) e_0 后 $|\text{core}_t(v) - \text{core}'_t(v)| = 1$.

证明 通过引理 3.1 可知, 如果插入 (删除) e_0 后 $|\text{core}_t(u) - \text{core}'_t(u)| = 1$, 则 $\text{core}_t(u) \in \text{VVS}_t(e_0)$. 接下来证明 e_0 中至少存在一个满足 $\text{core}_t(v) = \text{core}_t(u)$ 的顶点 v , 在插入 (删除) e_0 后 $|\text{core}_t(v) - \text{core}'_t(v)| = 1$.

首先证明将 e_0 插入 H 的情形. 令 $\text{core}'_t(u) = \text{core}_t(u) + 1$. 假设插入超边后 e_0 中满足 $\text{core}_t(\cdot) = \text{core}_t(u)$ 的所有顶点的 t -超核数都不增加. 子超图 HC_t^u 与 $HC_t'^u$ 分别是点 u 在超图 H 与 H' 中的极大- (k, t) -超核. 然后, $S = HC_t'^u - e'_0$ 在 H 中是 $(\text{core}'_t(u), t)$ -超核的一个子超图, 其中 e'_0 是超边 e_0 在 $HC_t'^u$ 中的导出超边. (该结论已在引理 3.1 的插入超边情形中证明.) 这与 $\text{core}'_t(u) = \text{core}_t(u) + 1$ 矛盾. 因此, e_0 中至少存在一个顶点 v 满足 $\text{core}_t(v) = \text{core}_t(u)$, 且在插入 e_0 后 $|\text{core}_t(v) - \text{core}'_t(v)| = 1$.

接下来证明从 H 中删除 e_0 的情形, 该情形可以看作是插入超边的逆过程. 如果删除 e_0 后 $\text{core}'_t(u) = \text{core}_t(u) - 1$, 反过来, 将 e_0 插入 H' 中后, u 的 t -超核数会增加. 因此 e_0 中至少存在一个顶点 v 满足 $\text{core}'_t(v) = \text{core}'_t(u)$, 且在插入 e_0 后 $\text{core}_t(v) = \text{core}'_t(v) + 1$. 所以 e_0 在 H 中至少存在一个顶点 v 满足 $\text{core}_t(v) = \text{core}_t(u)$, 且在删除 e_0 后 $|\text{core}_t(v) - \text{core}'_t(v)| = 1$. $\square$

引理 3.1 和 3.2 呈现了超图发生改变后 V^* 中顶点的一些性质. 基于这两条引理, 本节引出接下

来的定理, 这些定理给出了更加丰富详细的 V^* 中顶点的性质. 这些性质是一个顶点的 t -超核数会发生改变的必要条件.

定理 3.2 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 超边 e_0 插入到 H 中. 对于顶点 $u \in V$, 如果插入 e_0 后 $\text{core}'_t(u) - \text{core}_t(u) = 1$, 则一定存在至少一个 $\text{DVS}_t(e_0)$ 中满足 $\text{core}_t(v) = \text{core}_t(u)$ 的顶点 v , v 在超图 $H + e_0$ 中与顶点 u 通过路 $P = ue_{P_1}u_1e_{P_2}u_2 \cdots u_{l-1}e_{P_l}v$ 相连, 并且路 P 满足:

- (1) 顶点 $u, u_1, u_2, \dots, u_{l-1}$ 和 v 的 t -超核数等于 $\text{core}_t(v)$;
- (2) 对于任意一条超边 $e_{P_j} \in P, 1 \leq j \leq l$, 有 $\text{Cval}_t(e_{P_j}) = \text{core}_t(v)$.

证明 设顶点 u 的 t -超核数在插入超边 e_0 后增加. 然后, 一定存在至少一条 u 的关联超边帮助 u 获得一个更大的 t -超核数. 具体而言, 会发生以下两种情形之一: (i) 顶点 u 的度增加, 即 $u \in e_0$; (ii) 顶点 u 的度不增加, 但存在至少一条 u 的关联超边出现在 $H' = H + e_0$ 的 $(\text{core}_t(u) + 1, t)$ -超核中, 而这些超边不出现在 H 的 $(\text{core}_t(u) + 1, t)$ -超核中. 如果顶点 u 不满足这两种情形, 则 u 的所有不在 H 的 $(\text{core}_t(u) + 1, t)$ -超核中的关联超边都不会出现在 H' 的 $(\text{core}_t(u) + 1, t)$ -超核中. 因此, 顶点 u 在 H' 中不存在足够的出现在 $(\text{core}_t(u) + 1, t)$ -超核的关联超边. 这与假设矛盾, 因此顶点 u 必满足这两种情形之一.

如果顶点 u 满足情形 (i), 超边 e_0 被插入 H 中并且 u 的度增加, 则 $u \in e_0$. 路 $P = u$, 这满足条件 (1) 和 (2).

如果顶点 u 满足情形 (ii), 则存在至少一条 u 的关联超边出现在 H' 的 $(\text{core}_t(u) + 1, t)$ -超核中, 而这些超边不出现在 H 的 $(\text{core}_t(u) + 1, t)$ -超核中. 设超边 e 是这样的一条超边. 则在 H 中有 $\text{Cval}_t(e) < \text{core}_t(u) + 1$, 在 H' 中有 $\text{Cval}_t(e) \geq \text{core}_t(u) + 1$. 因为插入 e_0 后 H 中的每一个顶点 w 都满足 $\text{core}'_t(w) - \text{core}_t(w) \leq 1$, 所以在 H 中有 $\text{Cval}_t(e) = \text{core}_t(u)$, 在 H' 中有 $\text{Cval}'_t(e) = \text{core}_t(u) + 1$. 另外, 与超边 e 具有相同上述特点的超边数量可能不止一条, 但是一定存在包含顶点 u 和 u_1 的超边 e_{P_1} , 满足

$$\text{Cval}'_t(e_{P_1}) = \text{Cval}_t(e_{P_1}) + 1 = \text{core}_t(u) + 1 \quad \text{和} \quad \text{core}'_t(u_1) = \text{core}_t(u_1) + 1 = \text{core}_t(u) + 1,$$

且 $\text{Cval}_t(e_{P_1})$ 的增加是由 $\text{core}_t(u_1)$ 的增加导致的. 这说明顶点 u_1 帮助 u 获得了一个更大的 t -超核数. 按照上述过程找下去, 一定可以找到顶点 u_{l-1} 的 $\text{core}_t(u_{l-1})$ 的增加是由情形 (i) 中的顶点 v 的 $\text{core}_t(v)$ 的增加所导致. 综上, 可以找到从顶点 u 到顶点 v 的路 P , 其中 $v \in \text{DVS}_t(e_0)$, $\text{core}_t(v) = \text{core}_t(u)$ 且路 P 满足条件 (1) 和 (2). $\square$

定理 3.3 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 将超边 e_0 从 H 中删除. 对于顶点 $u \in V$, 如果删除 e_0 后 $\text{core}_t(u) - \text{core}'_t(u) = 1$, 则一定存在至少一个 $\text{DVS}_t(e_0)$ 中满足 $\text{core}_t(v) = \text{core}_t(u)$ 的顶点 v , v 在超图 H 中与顶点 u 通过路 $P = ue_{P_1}u_1e_{P_2}u_2 \cdots u_{l-1}e_{P_l}v$ 相连, 并且路 P 满足:

- (1) 顶点 $u, u_1, u_2, \dots, u_{l-1}$ 和 v 的 t -超核数等于 $\text{core}_t(v)$;
- (2) 对于任意一条超边 $e_{P_j} \in P, 1 \leq j \leq l$, 有 $\text{Cval}_t(e_{P_j}) = \text{core}_t(v)$.

证明 设顶点 u 的 t -超核数在删除超边 e_0 后减小. 从而一定存在至少一条 u 的关联超边帮助 u 减小它的 t -超核数. 具体而言, 会发生以下两种情形之一: (i) 顶点 u 的度减小, 即 $u \in e_0$; (ii) 顶点 u 的度不减小, 但存在至少一条 u 的关联超边出现在 H 的 $(\text{core}_t(u), t)$ -超核中, 而这些超边不出现在 $H' = H - e_0$ 的 $(\text{core}_t(u), t)$ -超核中.

删除超边 e_0 后 t -超核数减小的顶点 u 必满足上述两种情形之一. 满足条件 (1) 和 (2) 的路 P 可以通过与插入超边情形的定理 3.2 的证明中方法类似的方式找到. 证明细节不再详述. $\square$

实际上, 定理 3.2 和 3.3 解释了在超图发生变化后 V^* 中顶点的连通性, 即 t -超核数变化顶点影响的传递性. 依据此连通性, 本文采用广度优先搜索来探找 t -超核数可能会发生变化的顶点, 并设计出更新 t -超核数的维护算法.

4 单超边改变的 (k, t) -超核维护算法

本节根据前一节给出的相关理论结果, 设计出 (k, t) -超核维护问题的算法. 这些算法只考虑一次插入或者删除一条超边的情形.

4.1 单超边插入 t -超核数的更新

给定超图 $H = (V, E)$ 和一条超边 e_0 , 其中 $e_0 \notin H$. 本小节的目标是将超边 e_0 插入 H 后, 找到顶点集 V^* , 并更新 V^* 中顶点的 t -超核数. 为了描述 t -超核数可能会增加的顶点的增长潜力, 接下来为每个探找到的顶点定义增长潜力度:

定义 4.1 (增长潜力度) 给定超图 $H = (V, E)$ 、实数 $t \in (0, 1]$ 和超边 $e_0 \notin H$. 超边 e_0 被插入到 H 中. 顶点 u 的 t -超核数增长潜力度 $\text{GPD}_t(u)$, 定义为

$$\text{GPD}_t(u) = |\{e \mid e \in E_{H+e_0}(u), \text{Cval}_t(e) \geq \text{core}_t(u)\}|. \quad (4.1)$$

顶点 u 的 GPD 值的计算考虑了 u 的所有可能出现在超图 $H' = H + e_0$ 的 $(\text{core}_t(u) + 1, t)$ -超核中的关联超边. 然而, GPD 值计算时考虑到的关联超边并非都会出现在 H' 的 $(\text{core}_t(u) + 1, t)$ -超核中. 这些超边将会在 (k, t) -超核的维护过程中被筛除.

接下来介绍插入超边 e_0 后的 (k, t) -超核维护算法, 其伪代码如算法 2 所示, 其中算法 3 是算法 2 的子算法. 首先, 对于每一个 $\alpha \in \text{VVS}_t(e_0)$, 算法 2 运用广度优先搜索来探找 t -超核数可能会增加的所有顶点. 假设 $\alpha \in \text{VVS}_t(e_0)$ 正在被处理. 则算法从超边 e_0 开始探找, 将满足定理 3.2 中必要条件的所有顶点放入点集 Q 中. 然后, 对任意一个 $v \in Q$ 进行“访问 (visited)”. 算法根据顶点 v 的关联超边的临界有效值, 计算它的 GPD 值. 如果 $\text{GPD}_t(v) \leq \alpha$, 则 $\text{core}_t(v)$ 不会增加, 那么将顶点 v 从点集 Q 中“删除 (evicted)”. 检查顶点 v 的所有满足 $\text{Cval}_t(\cdot) = \alpha$ 的关联超边是否拥有足够的满足 $\text{core}_t(\cdot) \geq \alpha$ 且未被“删除”的顶点来使它们满足比例约束 (算法 3 第 3 行). 如果存在一条关联超边 e' 不满足该条件, 则 e' 不会出现在 $(\alpha + 1, t)$ -超核中, “删除”超边 e' 并更新 e' 中满足 $\text{core}_t(\cdot) = \alpha$ 且未被“删除”顶点的 GPD 值. 继续检查 e' 中被“访问”过的顶点的 GPD 值是否满足 $\text{GPD}_t(\cdot) \leq \alpha$, 以决定是否继续“删除”这些顶点. 通过上述操作, 探找并“删除”顶点和超边直到 Q 中的顶点都被“访问”过. 最后, 点集 Q 中剩余顶点的 t -超核数将会增加 1. 在所有顶点 t -超核数更新结束后, 算法将所有包含 t -超核数改变顶点的超边进行临界有效值的更新. 显然, 这些超边的临界有效值至多增加 1.

算法 2 的时间复杂度分析. 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 首先, 需要计算出超边 e_0 的临界有效值. 为了计算 $\text{Cval}_t(e_0)$, 将 e_0 内所有顶点的 t -超核数进行排序, 且选择第 $\lceil t * \text{edeg}_{H+e_0}(e_0) \rceil$ 大的数作为 e_0 的临界有效值 $\text{Cval}_t(e_0)$. 这个过程时间复杂度至多为 $O(c_{\max}(H) + \text{edeg}_{H+e_0}(e_0))$. 其次, 在算法 2 第 2 行的 for 循环内, 超图上所有的顶点和超边都可能被“访问”或“删除”. 当超边 e 被“访问”时, 算法判断 e 中每个顶点是否需要被放入点集 Q 中, 另外每个顶点的 GPD 值的计算也需要访问 e 一次, 这共需要消耗时间 $O(\text{edeg}_{H+e_0}(e))$. 同时, 当超边 e 中某个点被“删除”时, 判断 e 是否

算法 2 单超边插入的 (k, t) -超核维护算法

输入: 超图 $H = (V, E)$, V 中每个顶点 u 的 t -超核数 $\text{core}_t(u)$, E 中每条超边 e 的度 $\text{edeg}_H(e)$ 和临界有效值 $\text{Cval}_t(e)$, $t \in (0, 1]$, 插入的超边 e_0

输出: V 中每个顶点 u 的 t -超核数 $\text{core}_t(u)$

```

1:  $H' \leftarrow H + e_0$ ,  $\text{visited}(e_0) \leftarrow \text{true}$ ;
2: for  $\alpha \in \text{VVS}_t(e_0)$  do
3:    $Q \leftarrow \{u \mid u \in \text{DVS}_t(e_0), \text{core}_t(u) = \alpha\}$ ;
4:   对每个  $u \in Q$  赋值:  $\text{GPD}_t(u) \leftarrow 0$ ;
5:   for  $v \in Q$  do
6:      $\text{visited}(v) \leftarrow \text{true}$ ;
7:     for  $e \in E_{H'}(v)$  do
8:       if  $\text{Cval}_t(e) > \alpha$  或  $(\text{Cval}_t(e) = \alpha$  且  $\text{evicted}(e) = \text{null})$  then
9:          $\text{GPD}_t(v) \leftarrow \text{GPD}_t(v) + 1$ ;
10:      end if
11:    end for
12:    if  $\text{GPD}_t(v) \leq \alpha$  then
13:       $\text{EvictedIns}(v)$ ;
14:    else
15:      for  $E_{H'}(v)$  中每条满足  $\text{Cval}_t(e) = \alpha$ ,  $\text{visited}(e) = \text{null}$  且  $\text{evicted}(e) = \text{null}$  的超边  $e$  do
16:        for  $\text{DVS}_t(e)$  中每个满足  $\text{core}_t(w) = \alpha$  且  $\text{GPD}_t(w) = \text{null}$  的顶点  $w$  do
17:           $Q \leftarrow Q + w$ ,  $\text{GPD}_t(w) \leftarrow 0$ ;
18:        end for
19:         $\text{visited}(e) \leftarrow \text{true}$ ;
20:      end for
21:    end if
22:  end for
23:  for  $v \in Q$  do
24:     $\text{core}'_t(v) \leftarrow \text{core}_t(v) + 1$ ;
25:  end for
26:   $Q \leftarrow \emptyset$ ;
27: end for
28: for 每个满足  $\text{core}'_t(u) \neq \text{null}$  的顶点  $u$  do
29:    $\text{core}_t(u) \leftarrow \text{core}'_t(u)$ ;
30: end for
31: for 每条满足包含  $\text{core}'_t(\cdot) \neq \text{null}$  的顶点的超边  $e$  do
32:   if  $|\{u \in e \mid \text{core}_t(u) > \text{Cval}_t(e)\}| \geq \max\{2, t * \text{edeg}_{H'}(e)\}$  then
33:      $\text{Cval}_t(e) \leftarrow \text{Cval}_t(e) + 1$ ;
34:   end if
35: end for
36: 返回:  $V$  中每个顶点  $u$  的  $t$ -超核数  $\text{core}_t(u)$ 
    
```

需要被“删除”需要 $O(\text{edeg}_{H+e_0}(e) * \lceil (1-t) * \text{edeg}_{H+e_0}(e) \rceil)$ 量级的时间, 并至多需要 $O(\text{edeg}_{H+e_0}(e))$ 量级的时间来更新 e 中“未被删除”的顶点的 GPD 值. 在算法 2 的最后, H 中所有顶点被赋予新的 t -超核数, 该过程时间复杂度为 $O(n)$. 同时, 最坏情形下需要将所有超边的临界有效值进行更新, 每条超边 e 需要消耗时间 $O(\text{edeg}_{H'}(e))$. 综上, 算法 2 的时间复杂度是

$$O\left(\sum_{e \in H'} (\text{edeg}_{H'}(e) + \text{edeg}_{H'}(e) * \lceil (1-t) * \text{edeg}_{H'}(e) \rceil) + n + c_{\max}(H) + \text{edeg}_{H'}(e_0)\right),$$

化简后为

$$O(m * [(1 - t) * \Delta_E^2(H') + \Delta_E(H')] + n).$$

4.2 单超边删除 t -超核数的更新

本小节针对从超图 H 中删除超边 e_0 的情形, 设计删除一条超边的 (k, t) -超核维护算法, 其中 $e_0 \in H$. 为了描述删除超边后 t -超核数可能会减小的顶点的保持潜力, 并找到点集 V^* , 接下来定义顶点的保持潜力度如下:

定义 4.2 (保持潜力度) 给定超图 $H = (V, E)$ 、实数 $t \in (0, 1]$ 和超边 $e_0 \in H$. 将超边 e_0 从 H 中删除. 顶点 u 的 t -超核数保持潜力度, 记为 $\text{MPD}_t(u)$, 定义为

$$\text{MPD}_t(u) = |\{e \mid e \in E_{H-e_0}(u), \text{Cval}_t(e) \geq \text{core}_t(u)\}|. \quad (4.2)$$

不同于插入超边情形的 GPD 值, MPD 值并不是衡量一个顶点 t -超核数改变潜力的指标, 而是衡量它保持不变的潜力的指标. 对于顶点 u , 它的满足 $\text{Cval}_t(\cdot) > \text{core}_t(u)$ 的关联超边一定会为 $\text{core}_t(u)$ 保持不变作贡献, 将该类超边计算在内. 同时, 对于 u 的一条满足 $\text{Cval}_t(e) = \text{core}_t(u)$ 的关联超边 e , 它可能会为 $\text{core}_t(u)$ 保持不变作贡献, 所以在计算 $\text{MPD}_t(u)$ 时也将它计算在内. 删除超边 e_0 后, 如果判断出 $\text{Cval}_t(e)$ 会减小, 则超边 e 将不会帮助 u 保持 $\text{core}_t(u)$ 不变, 并且 $\text{MPD}_t(u)$ 需要被更新 (减 1).

通过定理 3.3 给出的必要条件, 本小节设计删除一条超边情形的 (k, t) -超核维护算法, 即算法 4, 其中算法 5 是算法 4 的子算法. 算法 4 的搜索过程与插入超边情形的算法 2 相似. 不同于算法 2 的是, 算法 4 在探找过程中使用 MPD 值直接找到 V^* 中的顶点, 这些顶点也就是探找过程中被“删除”的顶点. 另外, 超边的临界有效值也会在超边被“删除”时更新, 而不需要在更新顶点 t -超核数之后重新计算.

算法 4 的时间复杂度分析. 基于上述介绍分析, 算法 4 的维护过程与插入超边的情形类似, 并且在最坏情形下也需要搜索整个超图的顶点和超边. 因此, 算法 4 的时间复杂度为

$$O(m * [(1 - t) * \Delta_E^2(H) + \Delta_E(H)] + n).$$

4.3 单超边改变 t -超核数的并行更新算法

第 3 节中的结论说明, 对于顶点 u , 如果插入超边 e_0 后 u 的 t -超核数 $\text{core}_t(u)$ 增加, 则这一定是由 u 的某些关联超边的临界有效值增加而导致的. 而只有满足临界有效值 $\text{Cval}_t(\cdot) = \text{core}_t(u)$ 的关联

算法 3 EvictedIns(v)

```

1:  $Q \leftarrow Q - v$ ,  $\text{evicted}(v) \leftarrow \text{true}$ ;
2: for  $E_{H'}(v)$  中的每条满足  $\text{Cval}_t(e') = \alpha$  且  $\text{evicted}(e') = \text{null}$  的超边  $e'$  do
3:   if  $|\{u \mid u \in e', (\text{core}_t(u) = \alpha \text{ 且 } \text{evicted}(u) = \text{null}) \text{ 或 } \text{core}_t(u) > \alpha\}| < \max\{2, t * \text{edeg}_{H'}(e')\}$  then
4:      $\text{evicted}(e') \leftarrow \text{true}$ ;
5:     for  $e'$  中的每个满足  $\text{core}_t(x) = \alpha$ ,  $\text{visited}(x) = \text{true}$  且  $\text{evicted}(x) = \text{null}$  的顶点  $x$  do
6:        $\text{GPD}_t(x) \leftarrow \text{GPD}_t(x) - 1$ ;
7:       if  $\text{GPD}_t(x) \leq \alpha$  then
8:          $\text{EvictedIns}(x)$ ;
9:       end if
10:    end for
11:  end if
12: end for
```

算法 4 单超边删除的 (k, t) -超核维护算法

输入: 超图 $H = (V, E)$, V 中每个顶点 u 的 t -超核数 $\text{core}_t(u)$, E 中每条超边 e 的度 $\text{edeg}_H(e)$ 和临界有效值 $\text{Cval}_t(e)$, $t \in (0, 1]$, 删除的超边 e_0

输出: V 中每个顶点 u 的 t -超核数 $\text{core}_t(u)$

```

1:  $H' \leftarrow H - e_0$ ;
2: for  $\alpha \in \text{VVS}_t(e_0)$  do
3:    $Q \leftarrow \{u \mid u \in \text{DVS}_t(e_0), \text{core}_t(u) = \alpha\}$ ;
4:   对每个  $u \in Q$  赋值:  $\text{MPD}_t(u) \leftarrow 0$ ;
5:   for  $v \in Q$  do
6:      $\text{visited}(v) \leftarrow \text{true}$ ;
7:     for  $e \in E_{H'}(v)$  do
8:       if  $\text{Cval}_t(e) > \alpha$  或  $(\text{Cval}_t(e) = \alpha \text{ 且 } \text{evicted}(e) = \text{null})$  then
9:          $\text{MPD}_t(v) \leftarrow \text{MPD}_t(v) + 1$ ;
10:      end if
11:    end for
12:    if  $\text{MPD}_t(v) \leq \alpha - 1$  then
13:       $\text{EvictedDel}(v)$ ;
14:    end if
15:  end for
16:  for 每个  $\text{evicted}(v) = \text{true}$  的顶点  $v$  do
17:     $\text{core}'_t(v) \leftarrow \text{core}_t(v) - 1$ ;
18:  end for
19:   $Q \leftarrow \emptyset$ ;
20: end for
21: for 每个满足  $\text{core}'_t(u) \neq \text{null}$  的顶点  $u$  do
22:    $\text{core}_t(u) \leftarrow \text{core}'_t(u)$ ;
23: end for
24: for 每条满足  $\text{Cval}'_t(e) \neq \text{null}$  的超边  $e$  do
25:    $\text{Cval}_t(e) \leftarrow \text{Cval}'_t(e)$ ;
26: end for
27: 返回:  $V$  中每个顶点  $u$  的  $t$ -超核数  $\text{core}_t(u)$ 
    
```

超边能够对 $\text{core}_t(u)$ 的增加造成影响. 插入 e_0 后, 其他临界有效值改变的关联超边可分为以下两种情形: (i) 满足 $\text{Cval}_t(\cdot) < \text{core}_t(u)$ 的关联超边, 这些超边不能够帮助顶点 u 获得一个更大的 t -超核数; (ii) 满足 $\text{Cval}_t(\cdot) > \text{core}_t(u)$ 的关联超边, 这些超边在插入 e_0 前后均都存在于超图的 $(\text{core}_t(u) + 1, t)$ -超核中. 因此, 在算法 2 中, 对于每一个 $\alpha \in \text{VVS}_t(e_0)$, t -超核数变化的顶点的探找和计算不会相互影响. 这个结论可以运用到删除超边 e_0 的情形中. 通过上述分析可知, 可以将算法 2 和 4 中处理不同 α 的操作并行运行. 接下来, 算法 6 给出插入或删除一条超边后 (k, t) -超核维护的并行算法的伪代码.

算法 6 的时间复杂度分析. 该算法将 t -超核数 (即算法中 α) 不同的顶点分在不同的线程上并行更新, 消耗时间最多的 t -超核数对应的线程消耗的时间决定了本算法的最终时间复杂度. 设 $\alpha \in \text{VVS}_t(e_0)$, 令

$$V_\alpha = \{u \mid \text{core}_t(u) = \alpha\}, \quad E_\alpha^+ = \{e \mid e \in E_{H+e_0}(u), u \in V_\alpha\}, \quad E_\alpha^- = \{e \mid e \in E_H(u), u \in V_\alpha\},$$

$$n_\alpha = |V_\alpha|, \quad m_\alpha^+ = |E_\alpha^+|, \quad m_\alpha^- = |E_\alpha^-|.$$

而实际上 $m_\alpha^+ = m_\alpha^- + 1$. 以插入超边为例, 在 t -超核数为 α 的线程上, 算法至多访问 V_α 与 E_α^+ 中的顶点和超边来探找 t -超核数由 α 增加到 $\alpha + 1$ 的顶点. 算法最后更新超图上所有顶点的 t -超核数. 综

算法 5 EvictedDel(v)

```

1:  $Q \leftarrow Q - v$ ,  $\text{evicted}(v) \leftarrow \text{true}$ ;
2: for  $E_{H'}(v)$  中的每条满足  $\text{Cval}_t(e') = \alpha$  且  $\text{evicted}(e') = \text{null}$  的超边  $e'$  do
3:   if  $|\{u \mid u \in e', (\text{core}_t(u) = \alpha \text{ 且 } \text{evicted}(u) = \text{null}) \text{ 或 } \text{core}_t(u) > \alpha\}| < \max\{2, t * \text{edeg}_{H'}(e')\}$  then
4:      $\text{evicted}(e') \leftarrow \text{true}$ ;  $\text{Cval}'_t(e') = \text{Cval}_t(e') - 1$ ;
5:     for  $e'$  中的每个满足  $\text{core}_t(x) = \alpha$  的顶点  $x$  do
6:       if  $\text{MPD}(x) = \text{null}$  then
7:          $Q \leftarrow Q + x$ ,  $\text{MPD}_t(x) = 0$ ;
8:       end if
9:       if  $\text{visited}(x) = \text{true}$  且  $\text{evicted}(x) = \text{null}$  then
10:         $\text{MPD}_t(x) \leftarrow \text{MPD}_t(x) - 1$ ;
11:        if  $\text{MPD}_t(x) \leq \alpha$  then
12:          EvictedDel( $x$ );
13:        end if
14:      end if
15:    end for
16:  end if
17: end for

```

上, 结合算法 2 与 4 时间复杂度的分析可得, 算法 6 插入超边情形的时间复杂度为

$$O\left(\max_{\alpha \in \text{VVS}_t(e_0)} \{m_{\alpha}^{+} * [(1-t) * \Delta_E^2(H') + \Delta_E(H')]\} + n\right),$$

删除超边情形的时间复杂度为

$$O\left(\max_{\alpha \in \text{VVS}_t(e_0)} \{m_{\alpha}^{-} * [(1-t) * \Delta_E^2(H) + \Delta_E(H)]\} + n\right).$$

5 批量插入或删除超边的 (k, t) -超核维护算法

在普通图上, Jin 等^[8]、Yu 等^[15] 和 Hua 等^[6] 分别发现了在图上插入或删除一组匹配 (matching)、上边集 (superior edge set) 和联合边集 (joint edge set) 后, 图上每个顶点的核数至多增加或减小 1. 接下来, 他们利用相应的结论来对核维护算法进行优化. 同时, 他们证明了在图上一次性插入或删除相应的图结构时, 算法对插入或删除的每条边的处理互不影响, 据此设计出并行算法来进一步降低算法运行的总时间消耗. 基于上述成果, 本节研究在超图上插入或删除一组超边的 (k, t) -超核维护问题. 与一次只插入或删除一条超边的情形相比, 批量处理超边会减少不必要的运算, 提高算法的运行效率, 尤其是当算法可以并行运行时, 可以进一步降低时间消耗.

该问题的难点在于: (1) 找到一种超图结构, 该超图结构的插入与删除会导致整个超图顶点 t -超核数的变化有一般的、便于设计 (k, t) -超核维护算法的规律; (2) 在一次插入或删除一组超边的前提下, 利用已有结论和找到的超图结构, 设计并行 (k, t) -超核维护算法.

5.1 插入或删除一个 DDHS 后顶点 t -超核数的变化

接下来给出一种超图结构: DDHS. 它由一些满足特定条件的超边构成. 一次插入或删除一个 DDHS, 超图上每个顶点的 t -超核数至多变化 1, 其定义如下.

算法 6 单超边插入或删除的并行 (k, t) -超核维护算法

输入: 超图 $H = (V, E)$, V 中每个顶点 u 的 t -超核数 $\text{core}_t(u)$, E 中每条超边 e 的度 $\text{edeg}_H(e)$ 和临界有效值 $\text{Cval}_t(e)$, $t \in (0, 1]$, 插入或删除的超边 e_0

输出: V 中每个顶点 u 的 t -超核数 $\text{core}_t(u)$

```

1: if  $e_0$  被插入  $H$  then
2:    $H' \leftarrow H + e_0$ ,  $\text{visited}(e_0) \leftarrow \text{true}$ ;
3:   for  $\text{VVS}_t(e_0)$  中每一个  $\alpha$  并行进行下述操作 do
4:     算法 2: 3-26 行;
5:   end for
6:   for 每个满足  $\text{core}'_t(u) \neq \text{null}$  的顶点  $u$  do
7:      $\text{core}_t(u) \leftarrow \text{core}'_t(u)$ ;
8:   end for
9:   for 每条满足包含  $\text{core}'_t(\cdot) \neq \text{null}$  的顶点的超边  $e$ , 并行地 do
10:    if  $|\{u \in e \mid \text{core}_t(u) > \text{Cval}_t(e)\}| \geq \max\{2, t * \text{edeg}_{H'}(e)\}$  then
11:       $\text{Cval}_t(e) \leftarrow \text{Cval}_t(e) + 1$ ;
12:    end if
13:  end for
14: else
15:    $H' \leftarrow H - e_0$ ;
16:   for  $\text{VVS}_t(e_0)$  中每一个  $\alpha$  并行进行下述操作 do
17:     算法 4: 3-19 行;
18:   end for
19:   for 每个满足  $\text{core}'_t(u) \neq \text{null}$  的顶点  $u$  do
20:      $\text{core}_t(u) \leftarrow \text{core}'_t(u)$ ;
21:   end for
22:   for 每条满足  $\text{Cval}'_t(e) \neq \text{null}$  的超边  $e$  do
23:      $\text{Cval}_t(e) \leftarrow \text{Cval}'_t(e)$ ;
24:   end for
25: end if
26: 返回:  $V$  中每个顶点  $u$  的  $t$ -超核数  $\text{core}_t(u)$ 
    
```

定义 5.1 (主导顶点不交超边集) 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 如果超边集 E_D 中任意两条超边 e_1 和 e_2 都满足

$$\text{DVS}_t(e_1) \cap \text{DVS}_t(e_2) = \emptyset, \quad (5.1)$$

则称 E_D 是一个主导顶点不交超边集 (DDHS).

下面从另外一个角度来解释 DDHS. 设 E_D 是一个 DDHS, 任给一个顶点 u , 若存在 $e \in E_D$ 且 $u \in \text{DVS}_t(e)$, 则 e 是 E_D 中唯一主导顶点集包含顶点 u 的超边. 而对于存在于至少两条 E_D 中超边的顶点 v , 它们必存在于 E_D 中某些超边的非主导顶点集中, 直观上来看这些超边的插入或删除不会对 v 的 t -超核数的改变造成影响. 下面定理 5.1 给出插入或删除一个 DDHS 后, 每个顶点 t -超核数至多改变 1 这一结论及其证明.

定理 5.1 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 一个主导顶点不交超边集 E_D 被插入到 H 中 (或从 H 中删除). 则对于任意一个 $u \in V$, 有 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$.

证明 由主导顶点不交超边集的定义可知, E_D 中的超边可能会相交. 若存在超边 $e_1, e_2 \in E_D$ 相交, 不妨设顶点 $u \in e_1 \cap e_2$, 则 $u \in e_1 \setminus \text{DVS}_t(e_1)$ 或 $u \in e_2 \setminus \text{DVS}_t(e_2)$. 定义

$$V_{E_D} = \{u \mid \deg_{H[E_D]}(u) \geq 2\}. \quad (5.2)$$

由定义可知, 点集 V_{E_D} 是存在于 E_D 中至少两条超边的顶点的集合. 接下来, 定义 C_{E_D} 为 V_{E_D} 中顶点的 t -超核数构成的集合, 即

$$C_{E_D} = \{\text{core}_t(u) \mid u \in V_{E_D}\}. \quad (5.3)$$

对集合 C_{E_D} 中的数值从小到大进行排序: $k_1 < k_2 < \cdots < k_s$, 其中 $s = |C_{E_D}|$.

下面证明插入主导顶点不交超边集 E_D 的情形, 对 $k_i, 1 \leq i \leq s$ 使用数学归纳法. 当 $i = 1$ 时, 假设存在满足 $\text{core}_t(u) < k_1$ 的顶点 u 在插入 E_D 后有

$$|\text{core}_t(u) - \text{core}'_t(u)| > 1.$$

设 u 在超图 $H' = H + E_D$ 中的极大- (k, t) -超核为 $HC_t'^u$, 那么必存在部分 E_D 中超边的导出超边出现在 $HC_t'^u$ 上, 并设这些导出超边构成超边集 E'_D . 令子超图 $S = HC_t'^u - E'_D$, 则 $S \subseteq H$ 且 $u \in S$. 将超图 H 中所有满足 $\text{Cval}_t(\cdot) \geq k_1$ 的超边以及这些超边中满足 $\text{core}_t(\cdot) \geq k_1$ 的顶点加入 S , 扩充得到超图 S' . 下面证明 S 是 H 上的一个 $(\text{core}_t(u) + 1, t)$ -超核的子超图. 首先, 对于 S 中任意一条超边 e' , 必有

$$\frac{\text{edeg}_{S'}(e')}{\text{edeg}_H(e)} \geq \frac{\text{edeg}_S(e')}{\text{edeg}_H(e)} = \frac{\text{edeg}_{HC_t'^u}(e')}{\text{edeg}_{H'}(e)} \geq t.$$

其次, 对于任意一个 S 中满足 $\text{core}_t(v) < k_1$ 的顶点 v , 因为顶点 v 至多与一条 E'_D 中超边关联, 所以

$$\deg_{S'}(v) \geq \deg_S(v) \geq \deg_{HC_t'^u}(v) - 1 \geq \text{core}_t(u) + 1.$$

对于任意一个 S 中满足 $\text{core}_t(w) \geq k_1$ 的顶点 w , w 在超图 H 中拥有至少 k_1 个满足 $\text{Cval}_t(\cdot) \geq k_1$ 的关联超边, 且这些关联超边都在子超图 S' 中. 那么 S' 中顶点 w 满足 $\deg_{S'}(w) \geq \text{core}_t(u) + 1$. 同时, S 扩充为 S' , 新加的超边在 S' 中满足比例约束, 新加的顶点在 S' 必满足 $\deg_{S'}(\cdot) \geq \text{core}_t(u) + 1$. 综上可知, S' 为超图 H 上的 $(\text{core}_t(u) + 1, t)$ -超核的子超图, 即 S 是 H 上的一个 $(\text{core}_t(u) + 1, t)$ -超核的子超图. 因此 $\text{core}_t(u) = \text{core}_t(u) + 1$, 矛盾. 故满足 $\text{core}_t(u) < k_1$ 的顶点 u 在插入 E_D 后有 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$.

接下来证明, 在插入 E_D 后, 如果满足 $\text{core}_t(u) < k_{i-1}$ 的顶点 u 有 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$, $1 < i \leq s$, 则满足 $k_{i-1} \leq \text{core}_t(u_1) < k_i$ 的顶点 u_1 有 $|\text{core}_t(u_1) - \text{core}'_t(u_1)| \leq 1$. 反证该结论, 假设存在满足 $k_{i-1} \leq \text{core}_t(u_1) < k_i$ 的顶点 u_1 在插入 E_D 后有 $|\text{core}_t(u_1) - \text{core}'_t(u_1)| > 1$. 因为所有满足 $\text{core}_t(u) < k_{i-1}$ 的顶点 u 在插入 E_D 后有 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$, 所以每一条满足 $\text{Cval}_t(e) < k_{i-1}$ 的超边 e 都有 $\text{Cval}'_t(e) \leq k_{i-1}$, 因此超边 e 必不出现在顶点 u_1 在 H' 中的极大- (k, t) -超核 $HC_t'^{u_1}$ 中. 故顶点 u_1 在 $HC_t'^{u_1}$ 中至多与一条 E_D 中超边的导出超边相关联. 接下来利用 $i = 1$ 情形的证明方法可证明顶点 u_1 在插入 E_D 后有 $|\text{core}_t(u_1) - \text{core}'_t(u_1)| \leq 1$. 因此可得, 满足 $\text{core}_t(u) < k_s$ 的顶点 u 在插入 E_D 后有 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$. 故满足 $k_s \leq \text{core}_t(u_2)$ 的顶点 u_2 在插入 E_D 后的极大- (k, t) -超核 $HC_t'^{u_2}$ 中至多与 1 条 E_D 中超边的导出超边相关联. 若 $|\text{core}_t(u_2) - \text{core}'_t(u_2)| > 1$, 则 $HC_t'^{u_2}$ 删除 E_D 中超边的导出超边后的超图是一个 H 中 $(\text{core}_t(u_2) + 1, t)$ -超核的子超图, 矛盾, 则

$$|\text{core}_t(u_2) - \text{core}'_t(u_2)| \leq 1.$$

综上所述, 对于插入主导顶点不交超边集 E_D 的情形, 超图上任意一个 $u \in V$, 都满足

$$|\text{core}_t(u) - \text{core}'_t(u)| \leq 1.$$

下面证明删除主导顶点不交超边集 E_D 的情形. 仍然使用数学归纳法证明该情形. 首先, 设顶点 u 满足 $\text{core}_t(u) \geq k_s$ 且它在超图 H 上的极大- (k, t) -超核为 HC_t^u . 若删除 E_D 后有 $\text{core}_t(u) - \text{core}'_t(u) > 0$, 则必存在部分 E_D 中超边的导出超边出现在 HC_t^u 中, 并设这些导出超边构成超边集 E'_D . 因为 HC_t^u 中顶点至多与一条 E_D 中超边关联, 因此

$$\delta(HC_t^u - E'_D) \geq \text{core}_t(u) - 1.$$

又因为 $HC_t^u - E'_D$ 中超边必满足比例约束, 故有 $\text{core}_t(u) - \text{core}'_t(u) \leq 1$. 接下来, 假设满足 $\text{core}_t(u) \geq k_{i+1}$ 的顶点 u 在删除 E_D 后有

$$|\text{core}_t(u) - \text{core}'_t(u)| \leq 1, \quad 1 \leq i < s.$$

设顶点 u_1 满足 $k_i \leq \text{core}_t(u_1) < k_{i+1}$, 它的极大- (k, t) -超核为 $HC_t^{u_1}$. 若删除 E_D 后有 $\text{core}_t(u_1) - \text{core}'_t(u_1) > 0$, 则同样存在部分 E_D 中超边的导出超边出现在 $HC_t^{u_1}$ 中, 并设这些导出超边构成超边集 E''_D . 则对于 $HC_t^{u_1} - E''_D$ 中任意一个顶点 v , 有

$$\begin{cases} \deg_{HC_t^{u_1} - E''_D}(v) \geq \text{core}_t(u_1) - 1, & k_i \leq \text{core}_t(v) < k_{i+1}, \\ \deg_{HC_t^{u_1} - E''_D}(v) \geq \text{core}'_t(v) \geq \text{core}_t(u_1) - 1, & \text{core}_t(v) \geq k_{i+1}. \end{cases} \quad (5.4)$$

因为 $HC_t^{u_1} - E''_D$ 中超边必满足比例约束, 故有 $\text{core}_t(u_1) - \text{core}'_t(u_1) \leq 1$. 对于满足 $\text{core}_t(u_1) < k_1$ 的顶点 u_1 , 同样可以证明 $\text{core}_t(u_1) - \text{core}'_t(u_1) \leq 1$. 综上可知, 对于删除主导顶点不交超边集 E_D 的情形, 超图中任意一个顶点 $u \in V$, 都满足 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$. 从而有 $|\text{core}_t(u) - \text{core}'_t(u)| \leq 1$. $\square$

定理 5.1 给出插入或者删除一个 DDHS 后每个顶点 t -超核数的变化范围, 而更新顶点的 t -超核数还需要得到哪些顶点的 t -超核数发生了变化. 接下来, 引理 5.1 和定理 5.2 给出顶点的 t -超核数发生变化的必要条件, 并给出满足必要条件的顶点的搜索方案.

引理 5.1 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 一个主导顶点不交超边集 E_D 被插入到 H 中 (或从 H 中删除). 如果顶点 u 满足 $\text{core}_t(u) \notin \bigcup_{e \in E_D} \text{VVS}_t(e)$, 则 $\text{core}'_t(u) = \text{core}_t(u)$.

证明 首先证明插入主导顶点不交超边集 E_D 的情形. 由引理 3.1 可知, 插入一条超边 e_0 后, 若顶点 u 的 t -超核数满足 $\text{core}_t(u) \notin \text{VVS}_t(e_0)$, 则 $\text{core}'_t(u) = \text{core}_t(u)$. 同时, 根据定理 5.1, 将 E_D 中的超边按顺序 $e_1, e_2, \dots, e_i, \dots, e_s$ 一条条地插入 H 中, 每次插入时所有满足 $\text{core}_t(u) < \min \bigcup_{e \in E_0} \text{VVS}_t(e)$ 或 $\text{core}_t(u) > \max \bigcup_{e \in E_0} \text{VVS}_t(e) + 1$ 的顶点 u 都有 $\text{core}'_t(u) = \text{core}_t(u)$, 即插入 E_D 后有 $\text{core}'_t(u) = \text{core}_t(u)$. 若顶点 u 满足 $\text{core}_t(u) \notin \bigcup_{e \in E_D} \text{VVS}_t(e)$ 且

$$\min \bigcup_{e \in E_D} \text{VVS}_t(e) < \text{core}_t(u) \leq \max \bigcup_{e \in E_D} \text{VVS}_t(e) + 1,$$

假设 $\text{core}'_t(u) = \text{core}_t(u) + 1$, 则将 E_D 中超边一条条地插入时, 必存在超边 e_i , 它的插入使得 $\text{core}_t(u)$ 增加 1. 不妨设插入 e_i 前的超图为 H_{i-1} , 插入后的超图为 H_i . 由引理 3.2 可知, 在超图 H_{i-1} 上, 超边 e_i 中必存在 t -超核数为 $\text{core}_t(u)$ 且在插入 e_i 后 t -超核数增加的顶点 v . 而 $\text{core}_t(u) \notin \bigcup_{e \in E_D} \text{VVS}_t(e)$, 所以顶点 v 的 t -超核数在插入 e_i 前已增加, 矛盾. 因此 $\text{core}'_t(u) = \text{core}_t(u)$. 综上可知, 对于插入主导顶点不交超边集的情形, 结论成立.

接下来证明删除主导顶点不交超边集 E_D 的情形. 同样地, 由引理 3.1 和定理 5.1 可知, 满足 $\text{core}_t(u) < \min \bigcup_{e \in E_D} \text{VVS}_t(e) - 1$ 或 $\text{core}_t(u) > \max \bigcup_{e \in E_D} \text{VVS}_t(e)$ 的顶点 u 在删除 E_D 后有

$\text{core}'_t(u) = \text{core}_t(u)$. 由引理 3.2 可知, 对于满足

$$\text{core}_t(u) \notin \bigcup_{e \in E_D} \text{VVS}_t(e)$$

且

$$\min_{e \in E_D} \bigcup \text{VVS}_t(e) - 1 \leq \text{core}_t(u) < \max_{e \in E_D} \bigcup \text{VVS}_t(e)$$

的顶点 u , 用插入 E_D 情形的证明方法也可以得到 $\text{core}'_t(u) = \text{core}_t(u)$. 因此对于删除主导顶点不交超边集的情形, 结论也成立. $\square$

定理 3.2 和 3.3 说明, 一个顶点的 t -超核数变化, 一定是由它的某些临界有效值等于其 t -超核数的关联超边临界有效值的变化所导致的. 而一个超边临界有效值的变化必是因为其内部某些 t -超核数等于其临界有效值的顶点 t -超核数变化所导致. 这就是顶点 t -超核数与超边临界有效值变化影响的传递性. 对于一次插入或删除一个 DDHS 的情形, t -超核数变化的顶点也遵循该传递性规律. 在引理 5.1 给出的 t -超核数可能会发生变化的顶点的范围的基础上, 基于传递性规律以及定理 3.2 和 3.3 的证明方法, 本小节可直接得到插入或删除一个 DDHS 后 t -超核数变化的顶点所满足的必要条件, 该条件也是本节设计 (k, t) -超核维护算法的理论基础.

定理 5.2 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 主导顶点不交超边集 E_D 被插入到 H 中 (或从 H 中删除). 对于顶点 $u \in V$, 如果插入 (删除) E_D 后 $\text{core}_t(u)$ 增加 (减小), 则一定存在至少一个 $\bigcup_{e \in E_D} \text{DVS}_t(e)$ 中满足 $\text{core}_t(v) = \text{core}_t(u)$ 的顶点 v , v 在超图 $H + E_D$ (H) 中与顶点 u 通过路 $P = ue_{P_1}u_1e_{P_2}u_2 \cdots u_{l-1}e_{P_l}v$ 相连. 并且路 P 满足:

- (1) 顶点 $u, u_1, u_2, \dots, u_{l-1}$ 和 v 的 t -超核数等于 $\text{core}_t(v)$;
- (2) 对于任意一条超边 $e_{P_j} \in P$, $1 \leq j \leq l$, 有 $\text{Cval}_t(e_{P_j}) = \text{core}_t(v)$.

5.2 批量并行 (k, t) -超核维护算法

本小节给出在超图 $H = (V, E)$ 上一次插入或删除一组超边 E_0 的 (k, t) -超核的维护算法, 其中超边集合 E_0 是一组任意的超边, 其内部的超边的顶点相交情形是随机的. 在此, 基于前一节得到的理论结果来设计 (k, t) -超核的维护算法. 本小节需要解决以下两个问题:

(1) 将超边集 E_0 划分为多个 DDHS. 超边集合 E_0 中超边的顶点相交情形是随机的, 在此需要利用主导顶点不交超边集 DDHS 的定义, 从 E_0 中不断分离出 DDHS. 本小节选取 DDHS 的方法如下. 首先构建以 E_0 中所有超边主导顶点为键, 是否存在于已选取的某条超边的主导顶点集中为值的 Hash 表. 所有顶点的初始值均为 “不存在 ($\text{exist}(\cdot) = \text{false}$)”. 接下来对 E_0 中每条超边依次进行访问, 设被处理的超边为 e , 若所有 $u \in \text{DVS}_t(e)$ 的属性均为 “不存在”, 则选取 e 将其加入 E_D . 直到 E_0 中所有超边均被访问, 输出主导顶点不交超边集 E_D . 该过程的伪代码如算法 7 所示.

(2) 插入或删除一个 DDHS 的并行 (k, t) -超核维护算法. 由前文的定理可知, 一次插入或者删除一条超边与一次插入或删除一个 DDHS, 不仅 t -超核数变化的顶点的变化量相同, 而且 t -超核数变化的顶点所满足的必要条件也类似. 因此, 插入或删除一个 DDHS 的 (k, t) -超核维护算法探找 t -超核数可能会变化的顶点的方法过程也是类似的. 同时, 根据前文的分析, 对不同的 α 作并行计算也适用于此. 区别在于, 一次插入或删除一个 DDHS, 需要作为进行广度优先搜索的根节点的顶点数量更多, 以及这些根节点所对应的不同的 t -超核数 (算法中的 α) 更多. 而一次性处理更多的根节点可以减少重

算法 7 提取 DDHS E_D

输入: 超图 $H = (V, E)$, V 中每个点 u 的 t -超核数 $\text{core}_t(u)$, E 中每条超边 e 的度 $\text{edeg}_H(e)$ 和临界有效值 $\text{Cval}_t(e)$, $t \in (0, 1]$, 插入或删除的超边集 E_0

输出: E_0 中的一个 DDHS E_D

```

1: for  $E_0$  中每一条超边  $e$  do
2:   通过 (3.3) 计算  $e$  主导顶点集  $\text{DVS}_t(e)$ ;
3:   for  $\text{DVS}_t(e)$  中的每一个顶点  $u$  do
4:      $\text{exist}(u) \leftarrow \text{false}$ ;
5:   end for
6: end for
7:  $E_D \leftarrow \{\}$ ;
8: for  $e \in E_0$  do
9:   if  $\text{DVS}_t(e)$  中的每一个顶点  $u$  满足  $\text{exist}(u) == \text{false}$  then
10:     $E_D \leftarrow E_D + e$ ;
11:    for  $u \in \text{DVS}_t(e)$  do
12:       $\text{DVS}_t(e)$  中的每一个顶点  $u$  赋值  $\text{exist}(u) \leftarrow \text{true}$ ;
13:    end for
14:   end if
15: end for
16: 返回:  $E_0$  中的一个 DDHS  $E_D$ 
    
```

复计算, 提高计算速度. 对于根节点所对应的不同的 t -超核数更多这一问题, 并行算法能够很好地解决它.

算法 8 给出批量并行 (k, t) -超核维护算法的伪代码. 以插入超边集 E_0 为例, 算法 8 首先调用算法 7 选取一个 DDHS E_D . 然后将 E_D 插入超图 H , 并按照不同 t -超核数划分的不同线程, 并行地从 E_D 中超边的主导顶点开始, 按广度优先搜索的方式探找所有 t -超核数可能会变化的顶点. 同时, 对找到的所有 t -超核数可能会增加的顶点, 利用其增长潜力度 GPD 进行筛选, 其 GPD 达不到增加标准的顶点即为 t -超核数不会增加的顶点, 以此确定最终 t -超核数变化的顶点. 接下来在新的超图 $H' = H + E_D$ 上更新顶点的 t -超核数, 以及清空顶点与超边的“访问”和“删除”信息, 为下一轮插入新的 DDHS 作准备. 本轮更新结束后, 在新的超图 H' 上进行下一轮更新, 直到超边集 E_0 为空集, 输出最终顶点的 t -超核数, 算法终止. 对于删除超边集 E_0 的情形, 整体过程类似, 但在筛选 t -超核数变化的顶点时, 使用的是顶点的保留潜力度 MPD. 其 MPD 达不到保持不变标准的顶点即为 t -超核数减小的顶点.

算法 8 的时间复杂度分析. 给定超图 $H = (V, E)$ 和实数 $t \in (0, 1]$. 在超图上插入或删除超边集 E_0 , 设 $a = |E_0|$. 以在超图 H_S 上选取并插入 DDHS E_D 为例, 其中 $H_S = (V, E \cup S)$, $S \subseteq E_0$, DDHS 在超边集 $E_0 \setminus S$ 中选取. 对于超图 H_S 上的超边 e , 算法计算 e 的主导顶点集 $\text{DVS}_t(e)$ 首先需要求出 $\text{Cval}_t(e)$, 然后找出 $\text{DVS}_t(e)$ 中的顶点, 所需要的总时间为

$$O(c_{\max}(H_S) + \text{edeg}_{H_S+E_D}(e)).$$

在最坏情形下, 超边 e 的所有顶点都在 $\text{DVS}_t(e)$ 中. 同时对 $\bigcup_{e \in E_0 \setminus S} \text{DVS}_t(e)$ 中顶点构建 Hash 表并从 $E_0 \setminus S$ 中选取一个 DDHS 需要的时间为

$$O\left(\left|\bigcup_{e \in E_0 \setminus S} \text{DVS}_t(e)\right| + \sum_{e \in E_0 \setminus S} |\text{DVS}_t(e)|\right).$$

算法 8 批量插入或删除超边的并行 (k, t) -超核维护算法

输入: 超图 $H = (V, E)$, V 中每个顶点 u 的度 $\deg_H(u)$ 和 $\text{core}_t(u)$, E 中每条超边 e 的度 $\text{edeg}_H(e)$ 和临界有效值 $\text{Cval}_t(e)$, $t \in (0, 1]$, 插入或删除的超边集 E_0

输出: V 中每个顶点 u 的 t -超核数 $\text{core}_t(u)$

```

1: while  $E_0$  非空 do
2:    $E_D \leftarrow$  算法 7;
3:   if  $E_D$  被插入  $H$  then
4:      $H' \leftarrow H + E_D$ , 对  $E_D$  中每一条  $e$  赋值  $\text{visited}(e) \leftarrow \text{true}$ ;
5:     for  $\bigcup_{e \in E_D} \text{VVS}_t(e)$  中每一个  $\alpha$  并行进行下述操作 do
6:       算法 2: 3–26 行;
7:     end for
8:     for 每个满足  $\text{core}'_t(u) \neq \text{null}$  的顶点  $u$  do
9:        $\text{core}_t(u) \leftarrow \text{core}'_t(u)$ ;
10:    end for
11:    for 每条满足包含  $\text{core}'_t(\cdot) \neq \text{null}$  的顶点的超边  $e$ , 并行地 do
12:      if  $|\{u \in e \mid \text{core}_t(u) > \text{Cval}_t(e)\}| \geq \max\{2, t * \text{edeg}_{H'}(e)\}$  then
13:         $\text{Cval}_t(e) \leftarrow \text{Cval}_t(e) + 1$ ;
14:      end if
15:    end for
16:  else
17:     $H' \leftarrow H - E_D$ ;
18:    for  $\bigcup_{e \in E_D} \text{VVS}_t(e)$  中每一个  $\alpha$  并行进行下述操作 do
19:      算法 4: 3–19 行;
20:    end for
21:    for 每个满足  $\text{core}'_t(u) \neq \text{null}$  的顶点  $u$  do
22:       $\text{core}_t(u) \leftarrow \text{core}'_t(u)$ ;
23:    end for
24:    for 每条满足  $\text{Cval}'_t(e) \neq \text{null}$  的超边  $e$  do
25:       $\text{Cval}_t(e) \leftarrow \text{Cval}'_t(e)$ ;
26:    end for
27:  end if
28:  清空顶点与超边的  $\text{visited}$  和  $\text{evicted}$  信息,  $H \leftarrow H'$ ,  $E_0 \leftarrow E_0 - E_D$ ;
29: end while
30: 返回:  $V$  中每个顶点  $u$  的  $t$ -超核数  $\text{core}_t(u)$ 

```

因此, 整体上从 $E_0 \setminus S$ 中选取一个 DDHS 所需要的时间为

$$O\left(\sum_{e \in E_0 \setminus S} (c_{\max}(H_S) + \text{edeg}_{H_S + E_D}(e)) + \sum_{e \in E_0 \setminus S} |\text{DVS}_t(e)|\right),$$

化简为

$$O(|E_0 \setminus S| * (c_{\max}(H_S) + \Delta_E(H_S + E_D))).$$

则在 E_0 成为空集合前, 算法共需要

$$O(a^2 * (c_{\max}(H_{E_0}) + \Delta_E(H_{E_0})))$$

时间来分离 DDHS.

同样以在超图 H_S 上选取并插入 DDHS E_D 为例来分析插入超边集情形的时间复杂度. 该算法同样是将 t -超核数 (即算法中 α) 不同的顶点分在不同的线程上并行更新, 消耗时间最多的 t -超核数对

应的线程消耗的时间决定了本次插入更新 t -超核数的最终时间复杂度. 设

$$\alpha \in \bigcup_{e \in E_D} \text{VVS}_t(e), \quad K(H_S) = \{\text{core}_t(u) \mid u \in H_S\},$$

则令

$$V_\alpha = \{u \mid \text{core}_t(u) = \alpha\}, \quad E_\alpha^+ = \{e \mid e \in E_{H_S + E_D}(u), u \in V_\alpha\}, \quad n_\alpha = |V_\alpha|, \quad m_\alpha^+ = |E_\alpha^+|.$$

在 t -超核数为 α 的线程上, 算法至多访问 V_α 与 E_α^+ 中的顶点和超边来探找 t -超核数由 α 增加到 $\alpha + 1$ 的顶点. 算法最后更新超图上所有顶点的 t -超核数, 并清空“访问”和“删除”信息. 该过程时间复杂度为

$$O\left(\max_{\alpha \in \bigcup_{e \in E_D} \text{VVS}_t(e)} \{m_\alpha^+ * [(1-t) * \Delta_E^2(H_S) + \Delta_E(H_S)]\} + n + m\right).$$

因此, 插入 E_0 情形下算法 8 的时间复杂度为

$$\begin{aligned} & O(a^2 * (c_{\max}(H_{E_0}) + \Delta_E(H_{E_0})) \\ & + a * \max_{S \subseteq E_0} \max_{\alpha \in \bigcup_{e \in E_D} \text{VVS}_t(e)} \{m_\alpha^+ * [(1-t) * \Delta_E^2(H_S) + \Delta_E(H_S)]\} + a * (n + m)), \end{aligned}$$

整理后为

$$O\left(a * (m + n) + a^2 * (c_{\max}(H_{E_0}) + \Delta_E(H_{E_0})) + a * [(1-t) * \Delta_E^2(H_{E_0}) + \Delta_E(H_{E_0})] * \max_{S \subseteq E_0} \max_{\alpha \in K(H_S)} m_\alpha^+\right).$$

同理, 删除超边集 E_0 情形下算法 8 的时间复杂度为

$$O\left(a * (m + n) + a^2 * (c_{\max}(H) + \Delta_E(H)) + a * [(1-t) * \Delta_E^2(H) + \Delta_E(H)] * \max_{S \subseteq E_0} \max_{\alpha \in K(H_S)} m_\alpha^-\right),$$

其中

$$m_\alpha^- = |\{e \mid e \in E_{H_S}(u), u \in V(H_S), \text{core}_t(u) = \alpha\}|, \quad H_S = (V, E \setminus S).$$

6 结论与展望

图上密集子图的挖掘是应用在社交网络等众多应用领域中的一个重要问题. 超图是一种能够包含更多信息的图种类, 在超图上探找 k -超核等图结构受到了学者们的广泛关注. 本文研究的 (k, t) -超核是一种超图上的子超图, 它不仅对顶点的度有要求, 同时对超边的顶点参与比例 (比例约束) 有要求. 通过对控制比例约束的参数 t 取不同的值, 可以调节传统的 k -超核内部超边的连结性与整个子超图的内部紧密程度. 首先, 本文给出了 (k, t) -超核的相关定义以及已有的 3 种分解算法. 然后, 本文研究了动态超图上的 (k, t) -超核维护问题, 即在超图变化后更新顶点 t -超核数, 包括单超边插入或删除以及超边批量插入或删除两种动态变化情形. 对于单超边 e_0 改变的情形, 本文发现超图变化后每个顶点的 t -超核数至多变化 1. 同时, 本文证明了 t -超核数发生变化的顶点 u 必满足 $\text{core}_t(u) \in \text{VVS}_t(e_0)$ 且通过一条顶点 t -超核数与超边临界有效值均为 $\text{core}_t(u)$ 的路与 $\text{DVS}_t(e_0)$ 中顶点相连这一性质. 然后根据上述必要条件设计出单超边改变的 (k, t) -超核维护算法, 并给出其并行算法. 对于超边集 E_0 批量插入或删除的情形, 本文定义了一种主导顶点不交超边集 DDHS, 证明得到一次插入或删除一

个 DDHS, 每个顶点的 t -超核数至多变化 1 的结论. 同时 t -超核数发生变化的顶点 u 也必满足单超边改变情形下的必要条件. 通过上述结论, 本文设计了并行批量 (k, t) -超核维护算法. 该算法从插入或删除的超边集合 E_0 中一次剥离一个 DDHS, 将其在原超图中插入或删除并更新顶点 t -超核数, 重复上述操作直到 E_0 成为空集合. 并行批量维护算法能减少单超边维护算法中的重复运算, 提高效率.

在未来, 可继续对超图密集子图的挖掘问题展开研究. 第一, 可继续研究固定参数 k 的 (k, t) -超核维护问题, 更新求解此时每条超边所处的 t 最大的 (k, t) -超核或顶点的 k -比例. 第二, 根据实际应用与超图的性质, 定义出多种超图的密集子超图结构, 研究新的子超图结构的分解与维护问题, 例如考虑顶点的邻点数、关联超边数以及超边基数等信息的密集子超图结构. 第三, 研究利用 h -索引等工具来解决超图上子超图结构的分解与维护问题, 以达到降低时间成本的目的.

参考文献

- 1 Bae J, Kim S. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. *Phys A*, 2014, 395: 549–559
- 2 Batagelj V, Zaveršnik M. An $O(m)$ algorithm for cores decomposition of networks. *arXiv:cs/0310049*, 2003
- 3 Bu F, Lee G, Shin K. Hypercore decomposition for non-fragile hyperedges: Concepts, algorithms, observations, and applications. *Data Min Knowl Discov*, 2023, 37: 2389–2437
- 4 Cheng Y, Lu C, Wang N. Local k -core clustering for gene networks. In: *Proceedings of the 2013 IEEE International Conference on Bioinformatics and Biomedicine*. Shanghai: IEEE, 2013, 9–15
- 5 Cohen J. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report*, 2008, 16: 1–29
- 6 Hua Q S, Shi Y, Yu D, et al. Faster parallel core maintenance algorithms in dynamic graphs. *IEEE Trans Parallel Distrib Syst*, 2019, 31: 1287–1300
- 7 Hua Q S, Zhang X, Jin H, et al. Revisiting core maintenance for dynamic hypergraphs. *IEEE Trans Parallel Distrib Syst*, 2023, 34: 981–994
- 8 Jin H, Wang N, Yu D, et al. Core maintenance in dynamic graphs: A parallel approach based on matching. *IEEE Trans Parallel Distrib Syst*, 2018, 29: 2416–2428
- 9 Leng M, Sun L Y, Bian J N, et al. An $O(m)$ algorithm for cores decomposition of undirected hypergraph (in Chinese). *J Chin Comput Syst*, 2013, 34: 2568–2573 [冷明, 孙凌宇, 边计年, 等. 一种时间复杂度为 $O(m)$ 的无向超图核值求解算法. *小型微型计算机系统*, 2013, 34: 2568–2573]
- 10 Li R H, Qin L, Yu J X, et al. Finding influential communities in massive networks. *VLDB J*, 2017, 26: 751–776
- 11 Luo Q, Yu D, Cai Z, et al. Toward maintenance of hypercores in large-scale dynamic hypergraphs. *VLDB J*, 2023, 32: 647–664
- 12 Ramadan E, Tarafdar A, Pothén A. A hypergraph model for the yeast protein complex network. In: *Proceedings of the 18th International Parallel and Distributed Processing Symposium*. Santa Fe: IEEE, 2004, 189
- 13 Sariyüce A E, Gedik B, Jacques-Silva G, et al. Incremental k -core decomposition: Algorithms and evaluation. *VLDB J*, 2016, 25: 425–447
- 14 Seidman S B. Network structure and minimum degree. *Soc Networks*, 1983, 5: 269–287
- 15 Yu D, Wang N, Luo Q, et al. Fast core maintenance in dynamic graphs. *IEEE Trans Comput Soc Syst*, 2021, 9: 710–723
- 16 Zhang Y, Yu J X, Zhang Y, et al. A fast order-based approach for core maintenance. In: *Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. San Diego: IEEE, 2017, 337–348

Fast algorithms for the (k, t) -hypercore maintenance problem on hypergraphs

Feiteng Zhang, Bin Liu & Qizhi Fang

Abstract In the hypergraph H , the (k, t) -hypercore $HC_{(k,t)}$ is the maximal subhypergraph with $\delta(HC_{(k,t)}) \geq k$ and each hyperedge satisfying the fraction constraint about t . The fraction constraint about t is that the ratio of the number of vertices a hyperedge contains in the subhypergraph to the number of vertices it contains in H is not less than t . For each vertex $u \in H$, the t -hypercore number of u is the maximal k among all (k, t) -hypercores containing u . In this paper, we study the hypercore maintenance problem that is updating the t -hypercore numbers of vertices on dynamic hypergraphs. Firstly, we prove the necessary conditions satisfied by vertices whose t -hypercore numbers change when a hyperedge is inserted or deleted, and accordingly design (k, t) -hypercore maintenance algorithms for these cases, and give the parallel algorithm. Then, we study the (k, t) -hypercore maintenance problem for batch insertion or deletion of hyperedges. We define the dominant vertices disjoint hyperedge set (DDHS) and prove that each vertex's t -hypercore number can change by at most 1 after inserting or deleting a DDHS on a hypergraph. Finally, a parallel batch (k, t) -hyperedge maintenance algorithm is designed by DDHS. This algorithm is more efficient than the maintenance algorithms with single hyperedge change.

Keywords hypergraph, (k, t) -hypercore, hypercore maintenance, parallel algorithm, batch process

MSC(2020) 05C65, 05C85, 68R10

doi: 10.1360/SSM-2023-0319