Enumerating k-Vertex Connected Components in Large Graphs

2019 IEEE 35th International Conference on Data Engineering (ICDE)

Abstract

研究问题：structural cohesion
给定一个无向图，和一个对应的k，求出所有的k-VCC
K-VCC特性：高耦合，高鲁棒性，子图重叠性
方法：
1.证明了分割的上界——可以用多项式的时间解决该问题
2.为了减少局部连接性测试的时间，我们提出了两个优化策略
（1）neighbor sweep （2）group sweep
结果：
（1）在大型数据集上证明了高效性
（2）实验结果证明我们的方法有高达两个数量级的提升

Introduction

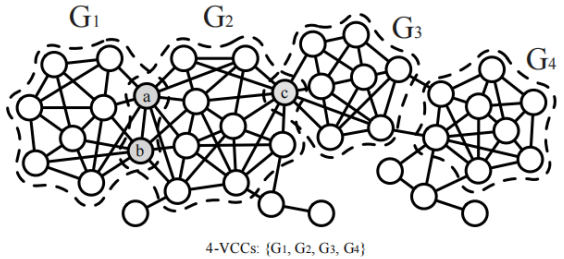


Fig. 1: An example for k-VCC in graph G.

- 适用场景：网络、pagerank、中心性、模块化
- Structural cohesion 的概念与k-VGG的概念在图论中相同
- K-VGG特性：
- （1）Whitney Theorem 证明G的顶点连通性不大于其边连通性和最小度——K-ECC有和k-ECC和K-core相同的特性
- （2）直径较小：
- 具体证明方法未知
- （3）子图的重叠虽然存在，但不超过n/2 其中n为顶点数

community in [12]. The diameter of a k-VCC $G'(V', E')$ is bounded by $\lceil \frac{|V'|-1}{\kappa(G')} \rceil$ where $\kappa(G')$ is the vertex connectivity of G' [13]. Thirdly, community overlap is regarded as an

Introduction

- 问题：给定G和k，求出所有的k-VGG
- 现有方法：
- （1）递归的找到一个顶点小于k个的顶点切割，并对图进行分区
- （2）门格尔定理证明：一个顶点对的最小割是他们之间的顶点独立路径发的数量——使用最大流的多项式算法
- 动机：
- （1）local connectivity testing：testing two vertices u and v can be disconnected in two components by removing at most k-1 vertices from G——不适用于大图，虽然有近似工作，或者是可以设计一个只适用于k=2或k=3的方法，但仍有改进

Introduction

- Our Approaches:
- （1）证明k-VCC的数量不超过n/2——多项式时间完成算法
- （2）Neighbor Sweep：对一些顶点进行属性划分；一旦其被测试或swept，则对其所有邻居sweep；为顶点存一个值，结束测试为其周围邻居增加值，超过阈值则sweep掉
- （3）Group Sweep：将顶点划分为不相交的组，以组为单位进行上面的测试
- 贡献:
- （1）一个多项式时间的算法
- （2）两个有效的剪枝策略
- （3）在实际问题的有效性

PreLiminary

- Problem Statement
- G=（V,E） n为顶点数量，m为边数
- $N(u)=\{v \in V, (u, v) \in E\}$ u的邻居
- d(u)=|N(u)|为u的度
- g’为g的子图
- G[Vs]为G的导出子图
- $G \cup Gs$ 为子图合并

PRELIMINARY

- Definition 1：Vertex Connectivity k(G) 定义为移除最小数量的顶点导致原图成为不相连或平凡图
- Definition 2: k-Vertex Connected (1)|V(G)|>k (2)删除任意k-1个顶点后原图仍联通
- Definition 3：k-Vertex Connected Component 子图为k联通组件的条件是 （1）g是k-vertex connected；(2) g is maximal，感觉最后的属于号有点奇怪

$$\nexists g' \subseteq G,\ s.t.\ \kappa(g') \geq k,\ g \subsetneq g'.$$

Idea：G和G’的k-VCC条件是无法比较的，可能子图大也可能子图小

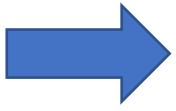
PARTITION-BASED FRAMEWORK

DEFINITION 4. (VERTEX CUT) *Given a connected graph G, a vertex subset $\mathcal{S} \subset V$ is a vertex cut if the removal of $\mathcal{S}$ from G results in a disconnected graph.*

Algorithm 1 KVCC-ENUM(G, k)

Input: a graph G and an integer k;
Output: all k-vertex connected components;

1: $VCC_k(G) \leftarrow \emptyset$;
2: **while** $\exists u : d(u) < k$ **do** remove u and incident edges;
3: identify connected components $\mathcal{G} = \{G_1, G_2, ..., G_t\}$ in G;
4: **for all** connected component $G_i \in \mathcal{G}$ **do**
5: $\quad \mathcal{S} \leftarrow$ GLOBAL-CUT(G_i, k);
6: $\quad$ **if** $\mathcal{S} = \emptyset$ **then**
7: $\quad\quad VCC_k(G) \leftarrow VCC_k(G) \cup \{G_i\}$;
8: $\quad$ **else**
9: $\quad\quad \mathcal{G}_i \leftarrow$ OVERLAP-PARTITION$(G_i, \mathcal{S})$;
10: $\quad\quad$ **for all** $G_i^j \in \mathcal{G}_i$ **do**
11: $\quad\quad\quad VCC_k(G) \leftarrow VCC_k(G) \cup$ KVCC-ENUM(G_i^j, k);
12: **return** $VCC_k(G)$;
13: **Procedure** OVERLAP-PARTITION(Graph G, Vertex Cut $\mathcal{S}$)
14: $\mathcal{G} \leftarrow \emptyset$;
15: $G' \leftarrow G[V(G) \setminus S]$;
16: **for all** connected component G'_i in G' **do**
17: $\quad \mathcal{G} \leftarrow \mathcal{G} \cup \{G[V(G'_i) \cup \mathcal{S}]\}$;
18: **return** $\mathcal{G}$;



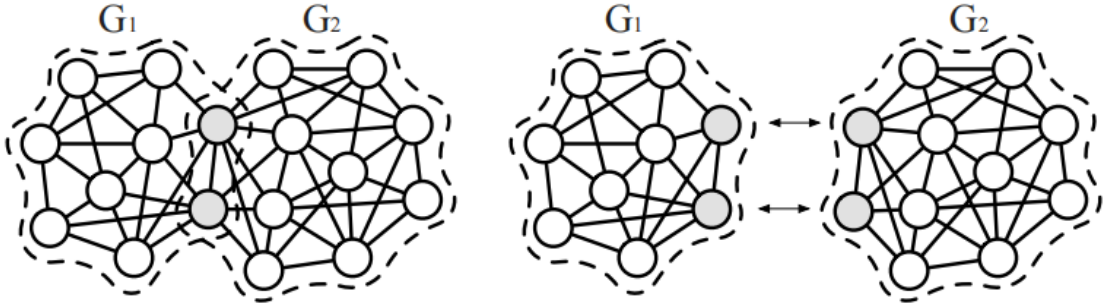


Fig. 2: An example of overlapped graph partition.

Basic Solution

- 1. Vertex Cut Computation

DEFINITION 5. (MINIMUM u-v CUT) *A vertex cut $\mathcal{S}$ is a u-v cut if u and v are in disjoint subsets after removing $\mathcal{S}$, and it is a minimum u-v cut if its size is no larger than that of other u-v cuts.*

DEFINITION 6. (LOCAL CONNECTIVITY) *Given a graph G, the local connectivity of two vertices u and v, denoted by $\kappa(u, v, G)$, is defined as the size of the minimum u-v cut. $\kappa(u, v, G) = +\infty$ if no such cut exists.*

- $u \equiv_G^k v$: The local connectivity between u and v is no less than k in graph G, i.e., $\kappa(u, v, G) \geq k$.
- $u \not\equiv_G^k v$: The local connectivity between u and v is less than k in graph G, i.e., $\kappa(u, v, G) < k$.

LEMMA 1. $u \equiv^k v$ *if* $(u, v) \in E$.

Lemma1：有边相连，那么k（u，v，G）就是正无穷了

Basic Solution

LEMMA 2. *Given a non-k-vertex connected graph G and a vertex $u \in \mathcal{S}$ where $\mathcal{S}$ is a minimal vertex cut and $|\mathcal{S}| < k$, there exist $v, v' \in N(u)$ such that $v \not\equiv^k v'$.*

（1）除v外其余全在割集中，则G为非k联通，则v与割集中点满足上式

（2）否则，v必定与不在割集中的除自身外任意一点与割集中相连的点满足上式

最大流的生成：
将原有的点拆点成为n+2m条边，2n个点的图，然后求u到v的最大流，即为其local connectivity——感觉应该额外加个源点和汇点？

Algorithm 2 GLOBAL-CUT(G, k)

Input: a graph G and an integer k;
Output: a vertex cut with fewer than k vertices;

```
1: compute a sparse certification SC of G;
2: select a source vertex u with the minimum degree;
3: construct the directed flow graph D(SC) of SC;
4: for all v ∈ V do
5:     S ← LOC-CUT(u, v, D(SC), SC);
6:     if S ≠ ∅ then return S;
7: for all v_a ∈ N(u) do
8:     for all v_b ∈ N(u) do
9:         S ← LOC-CUT(v_a, v_b, D(SC), SC);
10:        if S ≠ ∅ then return S;
11: return ∅;
12: Procedure LOC-CUT(u, v, D(G), G)
13: if v ∈ N(u) or v = u then return ∅;
14: λ ← calculate the maximum flow from u to v in D(G);
15: if λ ≥ k then return ∅;
16: compute the minimum edge cut in D(G);
17: return the corresponding vertex cut in G;
```

Basic Solution

DEFINITION 7. (CERTIFICATE) *A certificate for the k-vertex connectivity of G is a subset E' of E such that the subgraph (V, E') is k-vertex connected if and only if G is k-vertex connected.*

DEFINITION 8. (SPARSE CERTIFICATE) *A certificate for k-vertex connectivity of G is called sparse if it has $O(k \cdot n)$ edges.*

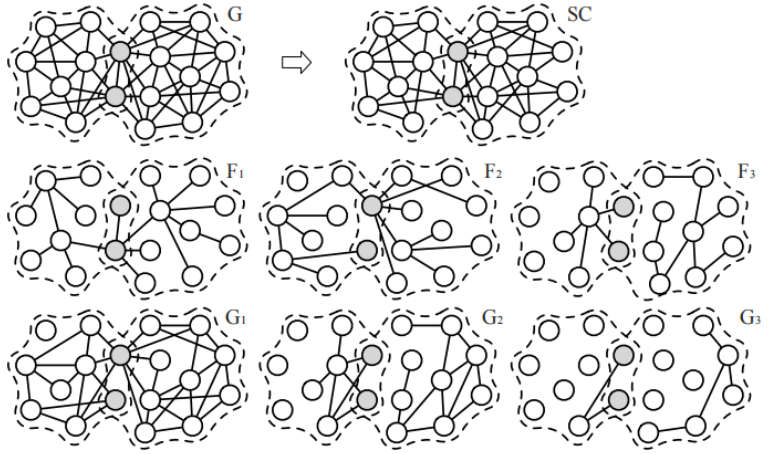


Fig. 3: The sparse certificate of G with $k = 3$

Algorithm 2 GLOBAL-CUT(G, k)

Input: a graph G and an integer k;
Output: a vertex cut with fewer than k vertices;

```
1: compute a sparse certification SC of G;
2: select a source vertex u with the minimum degree;
3: construct the directed flow graph SC̄ of SC;
4: for all v ∈ V do
5:     S ← LOC-CUT(u, v, SC̄, SC);
6:     if S ≠ ∅ then return S;
7: for all v_a ∈ N(u) do
8:     for all v_b ∈ N(u) do
9:         S ← LOC-CUT(v_a, v_b, SC̄, SC);
10:        if S ≠ ∅ then return S;
11: return ∅;
12: Procedure LOC-CUT(u, v, Ḡ, G)
13: if v ∈ N(u) or v = u then return ∅;
14: λ ← calculate the maximum flow from u to v in Ḡ;
15: if λ ≥ k then return ∅;
16: compute the minimum edge cut in Ḡ;
17: return the corresponding vertex cut in G;
```

THEOREM 1. *Let $G(V, E)$ be an undirected graph and let n denote the number of vertices. Let k be a positive integer. For $i = 1, 2, ..., k$, let E_i be the edge set of a scan first search forest F_i in the graph $G_{i-1} = (V, E - (E_1 \cup E_2 \cup ... \cup E_{i-1}))$, where $G_0 = G$. Then $E_1 \cup E_2 \cup ... \cup E_k$ is a certificate for the k-vertex connectivity of G, and this certificate has at most $k \times (n - 1)$ edges.*

Basic Solution

- 时间复杂度计算
- 上述式子可以保证一共只有n/2 个连通块

LEMMA 5. *For each connected component C computed by overlapped partition in Algorithm 1, $|V(C)| \geq k+1$.*

LEMMA 6. *The total number of overlapped partitions during the algorithm* KVCC-ENUM *is no greater than* $\frac{n-k-1}{2}$.

Discussion. Theorem 3 shows that all k-VCCs can be enumerated in polynomial time. Although the time complexity is still high, it performs much better in practice. Note that the time complexity is the product of three parts:

- The first part $O(\min(n^{1/2}, k) \cdot m)$ is the time complexity for LOC-CUT to test whether there exists a vertex cut of size smaller than k. In practice, the graph to be tested is much smaller than the original graph G since (1) The graph to be tested has been pruned using the k-core technique and sparse certification technique. (2) Due to the graph partition scheme, the input graph is partitioned into many smaller graphs.
- The second part $O(n+\delta^2)$ is the number of times such that LOC-CUT (local connectivity testing) is invoked by the algorithm GLOBAL-CUT. We will discuss how to significantly reduce the number of local connectivity testings in Section V.
- The third part $O(n)$ is the number of times GLOBAL-CUT is invoked. In practice, the number can be significantly reduced since the number of k-VCCs is usually much smaller than $\frac{n}{2}$.

THEOREM 2. *Given a graph G and an integer k, the number of k-VCCs is bounded by $\frac{n}{2}$, i.e., $|VCC_k(G)| < \frac{|V(G)|}{2}$.*

THEOREM 3. *The total time complexity of* KVCC-ENUM *is* $O(\min(n^{0.5}, k) \cdot m \cdot (n+\delta^2) \cdot n)$.

Search Reduction

DEFINITION 9. (SIDE-VERTEX) *Given a graph G and an integer k, a vertex u is called a side-vertex if there does not exist a vertex cut $\mathcal{S}$ such that $|\mathcal{S}| < k$ and $u \in \mathcal{S}$.*

- Neighbor Sweep—using Side-vertex
- 直观的想法就是，两个在连通块的点是可以传递这种连通性的，因此有了side-vertex的定义，其实不在over-lap也就是割集的点都是

LEMMA 7. *Given a graph G and an integer k, suppose $a \equiv^k b$ and $b \equiv^k c$, we have $a \equiv^k c$ if b is a side-vertex.*

LEMMA 8. *Given a graph G, a vertex u is a side-vertex if and only if $\forall v, v' \in N(u)$, $v \equiv^k v'$.*

LEMMA 9. *Given two vertices u and v, $u \equiv^k v$ if $|N(u) \cap N(v)| \geq k$.*

THEOREM 4. *A vertex u is a side-vertex if $\forall v, v' \in N(u)$, either $(v, v') \in E$ or $|N(v) \cap N(v')| \geq k$.*

DEFINITION 10. (STRONG SIDE-VERTEX) *A vertex u is called a strong side-vertex if it satisfies the conditions in Theorem 4.*

Neighbor Sweep

- 如何维护strong side-vertex
- 上面的复杂度很好证明
- 因为对于每个点的邻居，
- 都需要遍历其原来点的所有邻居

LEMMA 10. *The time complexity of computing all strong side-vertices in graph G is $O(\sum_{u \in V(G)} d(u)^2)$.*

LEMMA 11. *Let G be a graph and G_i be one of the graphs derived by partitioning G using* OVERLAP-PARTITION *in Algorithm 1, a vertex is a strong side-vertex in G if it is a strong side-vertex in G_i.*

From Lemma 11, we know that a vertex is not a strong side-vertex in G_i if it is not a strong side-vertex in G. This property allows us checking limited number of vertices in G_i, which is the set of strong side-vertices in G.

LEMMA 12. *Let G be a graph, G_i be one of the graphs derived by partitioning G using* OVERLAP-PARTITION *in Algorithm 1, and $\mathcal{S}$ is a vertex cut of G, for any vertex $v \in V(G_i)$, if v is a strong side-vertex in G and $N(v) \cap \mathcal{S} = \emptyset$, then v is also a strong side-vertex in G_i.*

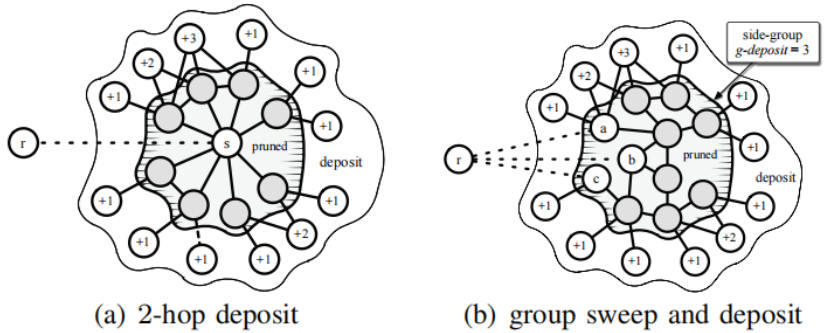
Vertex Deposit

LEMMA 13. *Given a source vertex u, for any vertex $v \in V(G)$, we have $u \equiv^k v$ if there exist k vertices $w_1, w_2, \ldots, w_k$ such that $u \equiv^k w_i$ and $w_i \in N(v)$ for any $1 \leq i \leq k$.*

DEFINITION 11. *(Vertex Deposit) Given a source vertex u, the deposit for each vertex v, denoted by $deposit_u(v)$, is the number of neighbors w of v such that $u \equiv^k w$.*

THEOREM 5. *Given a source vertex u, for any vertex v, we have $u \equiv^k v$ if $deposit_u(v) \geq k$.*

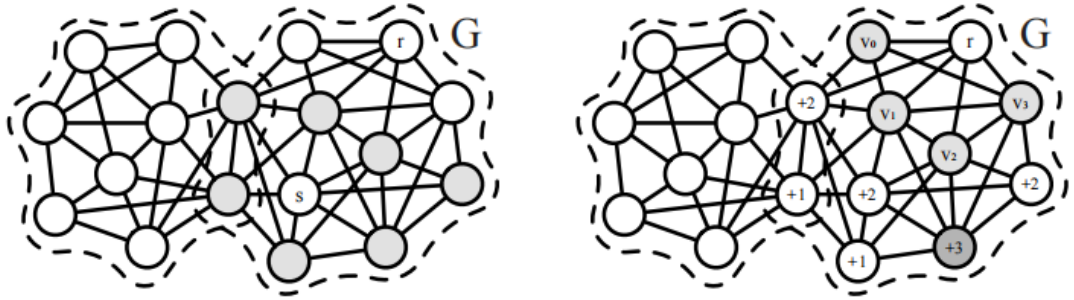
- 问题转化成了对于deposit的动态维护了，并可借助strong source vertex进行更好地动态维护



(a) 2-hop deposit (b) group sweep and deposit

Fig. 5: Increasing deposit with neighbor and group sweep

hop neighbors of s increase their deposits accordingly. The increased deposit for each vertex is shown in the figure.



(a) A strong side-vertex s (b) Vertex deposit

Fig. 4: Strong side-vertex and vertex deposit when $k = 3$

Group Sweep

Next we show that the side-groups can also be used to prune the local connectivity testings in the second phase of GLOBAL-CUT. Recall that in the second phase of GLOBAL-CUT, given a source vertex u, we need to test the local connectivity of every pair (v_a, v_b) of the neighbors of u. With side-groups, we can easily infer the following group sweep rule.

(Group Sweep Rule 3) *Let u be the source vertex, and v_a and v_b be two neighbors of u. If v_a and v_b belong to the same side-group, we have $v_a \equiv^k v_b$ and thus we do not need to test the local connectivity of (v_a, v_b) in the second phase of* GLOBAL-CUT.

DEFINITION 12. (SIDE-GROUP) *Given a graph G and an integer k, a vertex set $\mathcal{CC}$ is a side-group if $\forall u, v \in \mathcal{CC}, u \equiv^k v$.*

Side-Group Construction. Section IV-A introduces sparse certificate to bound the graph size. Let F_i and G_i be the notations defined in Theorem 1. Assume G is not k-connected and there exists a vertex cut $\mathcal{S}$ with $|\mathcal{S}| < k$. We introduce the following lemma [22].

LEMMA 14. *F_k does not contain a tree path whose two end points are in different connected components of $G - \mathcal{S}$.*

THEOREM 6. *Let $\mathcal{CC}$ denote the vertex set of any connected component in F_k. $\mathcal{CC}$ is a side-group.*

Group Sweep可与前面的Neighbor Sweep结合，
大幅度的降低两个顶点之间的连接程度

Group Sweep

- 和之前类似，这里需要通过合理赋值来进行权值更新而减少local connectivity test

(Group Sweep Rule 1) *Let u be the source vertex in the algorithm* GLOBAL-CUT*, given a side-group $\mathcal{CC}$, if there exists a strong side-vertex $v \in \mathcal{CC}$ such that $u \equiv^k v$, we can skip the local connectivity testings of vertex pairs (u, w) for all $w \in \mathcal{CC} - \{v\}$.*

LEMMA 15. *Given a source vertex u, an integer k, and a side-group $\mathcal{CC}$, we have $u \equiv^k \mathcal{CC}$ if $|\{v | v \in \mathcal{CC}, u \equiv^k v\}| \geq k$.*

DEFINITION 13. *(Group Deposit) Given a source vertex u, the group deposit for each side-group $\mathcal{CC}$, denoted by $g\text{-}deposit_u(\mathcal{CC})$, is the number of vertices $v \in \mathcal{CC}$ such that $u \equiv^k v$.*

THEOREM 7. *Given a source vertex u, for any side-group $\mathcal{CC} \in \mathcal{CS}$, we have $u \equiv^k \mathcal{CC}$ if $g\text{-}deposit_u(\mathcal{CC}) \geq k$.*

(Group Sweep Rule 2) *Given a selected source vertex u, we can skip the local connectivity testings between u and vertices in $\mathcal{CC}$ if $g\text{-}deposit_u(\mathcal{CC}) \geq k$.*

Overall Algorithm

Algorithm 4 SWEEP$(v, pru, deposit_u, g\text{-}deposit_u, \mathcal{CS})$

```
1: pru(v) ← true;
2: for all w ∈ N(v) s.t. pru(w) = false do
3:     deposit_u(w)++;
4:     if v is a strong side-vertex or deposit_u(w) ≥ k then
5:         SWEEP(w, pru, deposit_u, g-deposit_u, CS);
6: if v is contained in a CC_i and CC_i has not been processed then
7:     g-deposit_u(CC_i)++;
8:     if v is a strong side-vertex or g-deposit_u(CC_i) ≥ k then
9:         mark CC_i as processed;
10:        for all w ∈ CC_i s.t. pru(w) = false do
11:            SWEEP(w, pru, deposit_u, g-deposit_u, CS)
```

Algorithm 3 GLOBAL-CUT$^*(G, k)$

Input: a graph G and an integer k;
Output: a vertex cut with size smaller than k;

```
1: compute a sparse certification SC of G and collect all side-groups
   as CS = {CC_1, ..., CC_t};
2: construct the directed flow graph ¬SC of SC;
3: SV ← compute all strong side vertices in SC;
4: if SV = ∅ then
5:     select a vertex u with the minimum degree;
6: else
7:     randomly select a vertex u from SV;
8: for all CC_i in CS: g-deposit_u(CC_i) ← 0;
9: for all v in V: deposit_u(v) ← 0, pru(v) ← false;
0: SWEEP(u, pru, deposit_u, g-deposit_u, CS);
1: for all v ∈ V in non-ascending order of dist(u, v, G) do
2:     if pru(v) = true then continue;
3:     S ← LOC-CUT(u, v, ¬SC, SC);
4:     if S ≠ ∅ then return S;
5:     SWEEP(v, pru, deposit_u, g-deposit_u, CS);
6: if u is not a strong side-vertex then
7:     for all v_a ∈ N(u) do
8:         for all v_b ∈ N(u) do
9:             if v_a and v_b are in the same CC_i then continue;
0:             S ← LOC-CUT(u, v, ¬SC, SC);
1:             if S ≠ ∅ then return S;
2: return ∅;
```

Experiments

- Datasets如右图所示，按照E来排序
- 参数设置不选最小的Kmax?
- 时间上如Fig.6 毫无质疑都有很大的提升，数量级单位的变化

Datasets	$\|V\|$	$\|E\|$	$\bar{d}$
ca-CondMat	23,133	93,497	8.08
ca-AstroPh	18,772	198,110	21.11
Stanford	281,903	2,312,497	16.41
cnr	325,557	3,216,152	19.76
DBLP	986,324	6,707,236	13.60
Web-BerkStan	685,230	7,600,595	22.18
as-Skitter	1,696,415	11,095,298	13.08
cit-Patents	3,774,768	16,518,948	8.75
LiveJournal	4,847,571	68,993,773	28.47
Webbase	118,142,155	1,019,903,190	17.27

TABLE I: Network Statistics

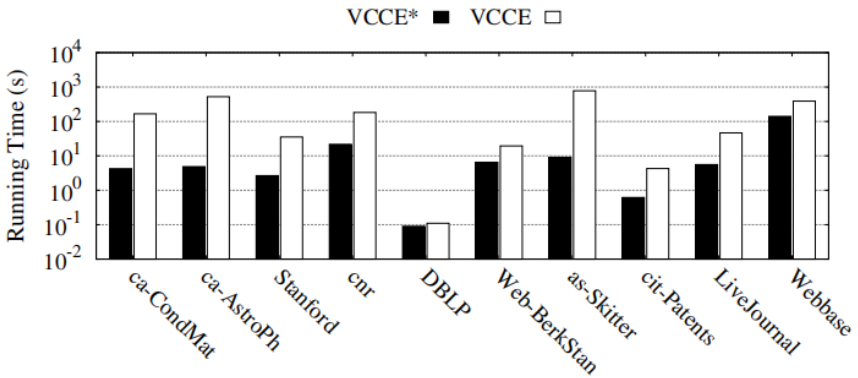


Fig. 6: Performance on different datasets

Parameter Setting. Given a graph G, let $k_{max}(G)$ be the maximum k such that a k-VCC exists. For the input parameter k, we choose 20%, 40%, 60%, and 80% of the $k_{max}(G)$ of each tested graph G, with $k = 40\% \cdot k_{max}(G)$ as default.

Experiments

- K越大，时间越小
- （1）会去除大量的顶点
- （2）切割更有可能找到，对应图将会被分割的更小

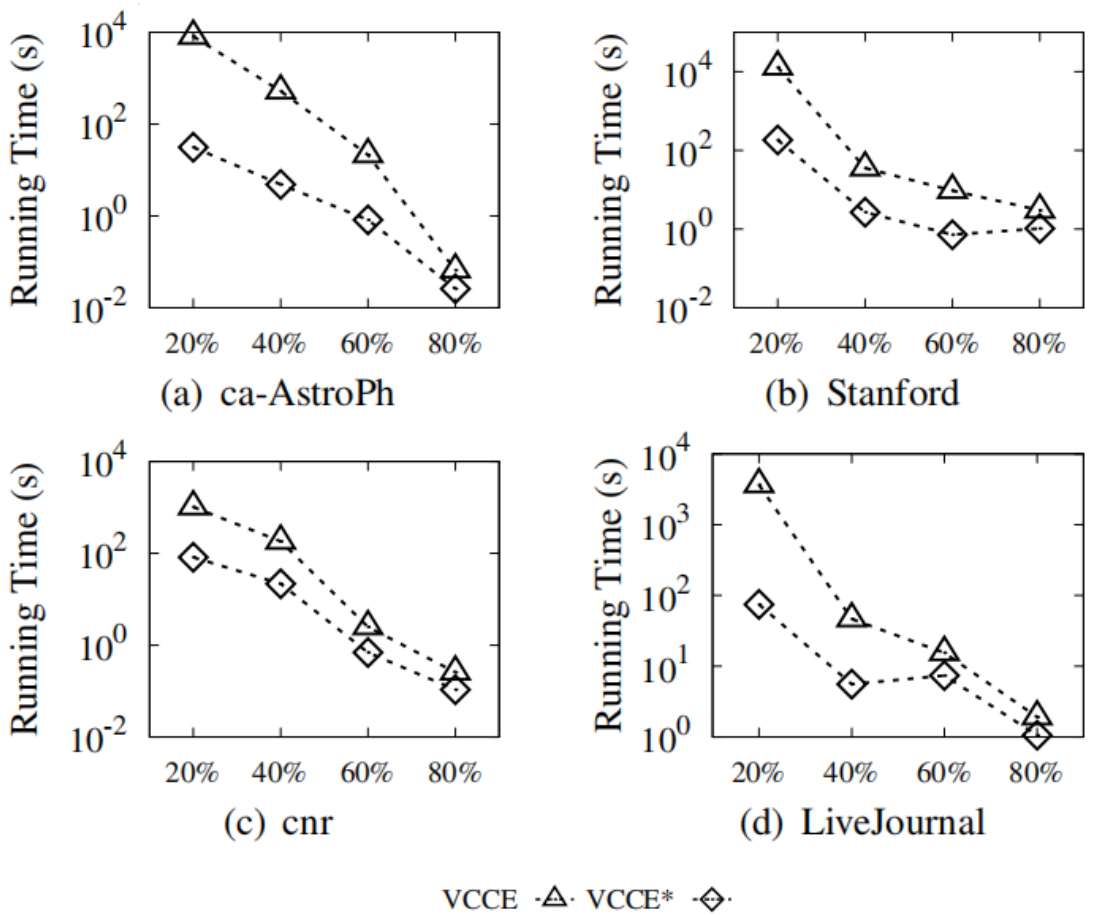


Fig. 7: Against basic algorithm (Vary k)

Experiments

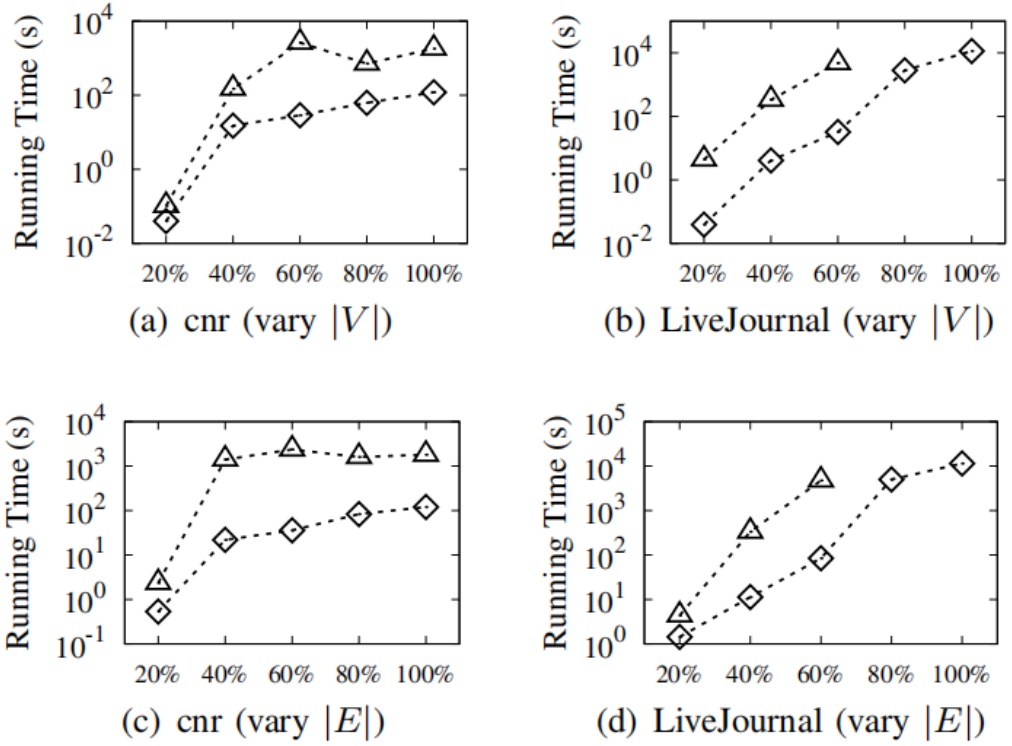
- 证明各个阶段的剪枝确实有用
- K增大的时候效果会下降
- 不同剪枝策略是否奏效取决于图结构

Input Parameter	*Non-Pru*	*NS_1*	*NS_2*	*GS*
ca-AstroPh				
$20\% \cdot k_{max}$	3.5%	20.4%	30.2%	45.9%
$40\% \cdot k_{max}$	3.8%	40.0%	23.8%	32.3%
$60\% \cdot k_{max}$	13.8%	42.10%	38.40%	5.70%
$80\% \cdot k_{max}$	100%	0%	0%	0%
avg	4.1%	27.3%	28.7%	39.9%
Stanford				
$20\% \cdot k_{max}$	2.9%	32.3%	14.0%	50.9%
$40\% \cdot k_{max}$	10.1%	12.0%	18.7%	59.2%
$60\% \cdot k_{max}$	7.1%	3.6%	25.6%	63.6%
$80\% \cdot k_{max}$	12.0%	0%	14.7%	73.3%
avg	3.3%	30.7%	14.4%	51.6%
cnr				
$20\% \cdot k_{max}$	3.0%	12.4%	16.0%	68.6%
$40\% \cdot k_{max}$	30.7%	8.9%	47.3%	13.2%
$60\% \cdot k_{max}$	15.8%	27.1%	16.3%	40.7%
$80\% \cdot k_{max}$	61.1%	38.9%	0%	0%
avg	7.0%	12.1%	20.4%	60.5%
LiveJournal				
$20\% \cdot k_{max}$	1.5%	3.6%	39.2%	55.7%
$40\% \cdot k_{max}$	8.5%	6.9%	61.1%	23.4%
$60\% \cdot k_{max}$	19.3%	2.6%	69.7%	8.4%
$80\% \cdot k_{max}$	84.4%	15.6%	0%	0%
avg	3.4%	4.0%	43.2%	49.4%

TABLE II: Evaluating pruning rules

Experiments——Scalability Testing

- 对图中顶点和边随机抽样——20%到100%
- 分别取导出子图，边和点的
- Kmax=10%，相对于抽样前的图
- 部分实验时间太长没有跑完
- 这里时间比之前增长是因为k选得变小了更多



Related Work

- Cohesive Subgraph
- （1）最大团枚举的算法——过于苛刻的定义
- （2）Global Cohesiveness
- S-clique，s-club，k-plex，Quasi-clique
- （3）Local Degree and Triangulation
- K-core，k-truss，k-mutual-friend，DN-Graph
- （4）Connectivity Cohesiveness
- （5）Vertex Connectivity：
- O（n0.5m）可以在无向图中求出最大流，若每个点要么流入要么流出
- 测试顶点连通性在给定k下为O（n0.5m2） 可以优化到O（k3m+knm）

Conclusions

- 多项式时间可解决k-VCCs的问题
- 提出了两种优化策略
- 通过实验证明了算法的有效性