Maintaining Densest Subsets Efficiently in Evolving Hypergraphs

Shuguang Hu
The University of Hong Kong
Pokfulam Road, Hong Kong
sghu@cs.hku.hk

Xiaowei Wu
The University of Hong Kong
Pokfulam Road, Hong Kong
wxw0711@gmail.com

T-H. Hubert Chan
The University of Hong Kong
Pokfulam Road, Hong Kong
hubert@cs.hku.hk

Abstract

Problems: The densest subgraph problem in hypergraphs

Methods:

1)We present two exact algorithms and a near-linear time r-approximation algorithm for the problem.

2)We also consider the dynamic version of the problem. We present two dynamic approximation algorithms in this paper with amortized $poly(\frac{r}{\varepsilon}\log n)$ update time, for any $\epsilon > 0$.

Introduction

- H(V, E) find a subset of nodes S $\subseteq V$ $\ such\ that\ \rho(S) = |E[s]|/\ |S|$ is maximized.
  $E[s] = \{e \in E : e \subseteq S\}$
- The Densest Subgraph Problem:
- 1) Goldberg provided a O(log n) max-flow computations, where n = |V|.
- 2) Charikar provided a O(m) 2-approximation algorithm.
- 3) A LP was proposed with O(m+n) variables.

Introduction

- Densest Subset in Hypergraphs:
- 1. In many applications, nodes under consideration are connected by hyperedges that involve more than 2 objects.
- 2. Given the hypergraph, the goal of the Densest Subgraph Problem is to identify a group of researchers S such that the average number of collaborations within S is maximized.

Introduction

- **Dynamic Setting**:
  - The dynamic Densest Subgraph Problem aims at maintaining an (approximate) densest subgraph under edge insertions and deletions.
- **The Densest Subgraph Problem**:
  - **Bahmani**: $O(\frac{1}{\epsilon}\log n)$ passes, within a factor $(2+\epsilon)$ of the optimum fixes a threshold β and removes nodes with degree smaller than β in each iteration.
  - **Epasto** considered the problem where insertions are adversarial and deletions are random. $(2+\epsilon)$-approximation algorithm poly $O(\frac{1}{\epsilon}\log n)$ time, using O(m+n) space.

Introduction

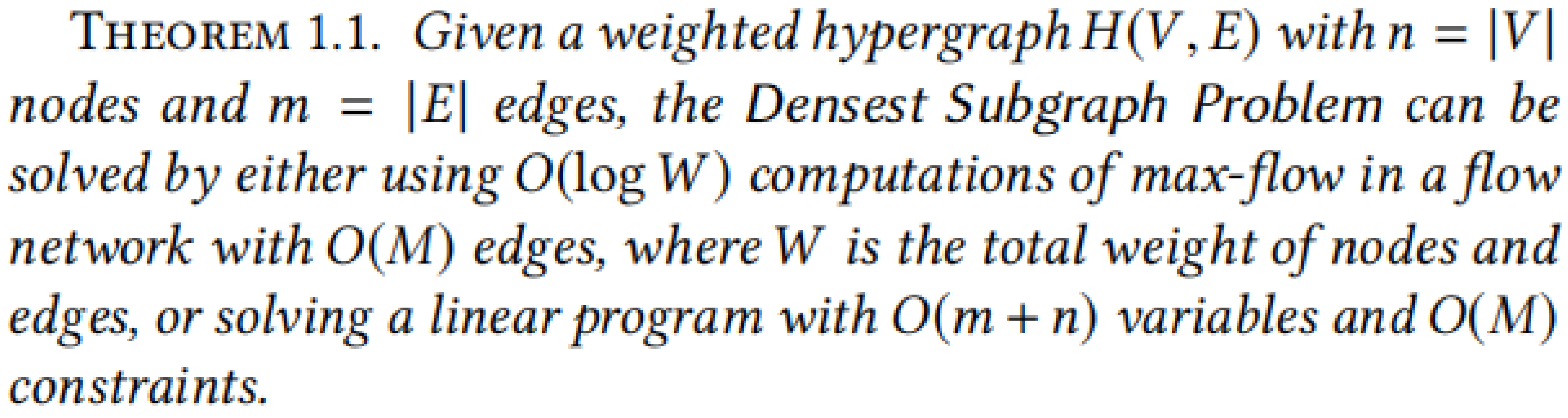
- **Bhattachary**: consider the deletions are also adversarial. $(4+\epsilon)$ –approximation $O(\frac{1}{\epsilon}\log n)$ update time and $O\left(n.poly\left(\frac{1}{\epsilon}logn\right)\right)$ space.
- **Esfandiari et al. and Mitzenmacher** presented semi-streaming algorithms for the problem that maintain a $(1+\epsilon)$-approximation using $O(n.poly\left(\frac{1}{\epsilon}logn\right))$ space. Their algorithms process each update also in $poly\left(\frac{1}{\epsilon}logn\right)$ time, but the query-time can be as large as $\Omega(n.poly\left(\frac{1}{\epsilon}logn\right))$.

Introduction

- ***Our Methods:***

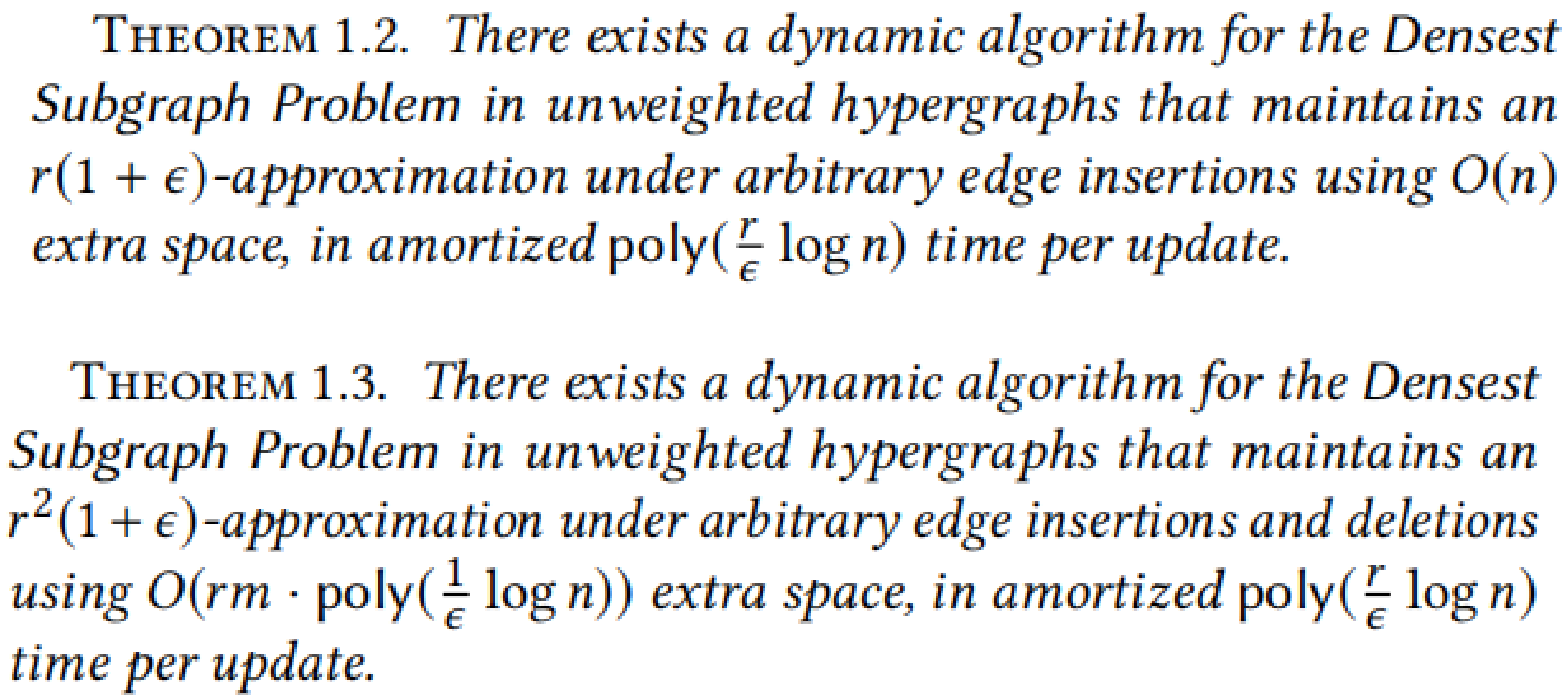
- $r = \max_{e \in E}\{|e|\}$ to denote the maximum cardinality of a hyperedge.

- $M := \sum_{e \in E} |e| \leq rm$



THEOREM 1.1. *Given a weighted hypergraph $H(V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, the Densest Subgraph Problem can be solved by either using $O(\log W)$ computations of max-flow in a flow network with $O(M)$ edges, where W is the total weight of nodes and edges, or solving a linear program with $O(m + n)$ variables and $O(M)$ constraints.*

Introduction



THEOREM 1.2. *There exists a dynamic algorithm for the Densest Subgraph Problem in unweighted hypergraphs that maintains an $r(1+\epsilon)$-approximation under arbitrary edge insertions using $O(n)$ extra space, in amortized* $\mathrm{poly}(\frac{r}{\epsilon}\log n)$ *time per update.*

THEOREM 1.3. *There exists a dynamic algorithm for the Densest Subgraph Problem in unweighted hypergraphs that maintains an $r^2(1+\epsilon)$-approximation under arbitrary edge insertions and deletions using $O(rm \cdot \mathrm{poly}(\frac{1}{\epsilon}\log n))$ extra space, in amortized* $\mathrm{poly}(\frac{r}{\epsilon}\log n)$ *time per update.*

Introduction

- ***Experimental Evaluation:***
- Moreover, our approximation algorithm runs several times faster than the exact algorithm, and returns a solution with density very close to the optimum.
- Moreover, as the first to implement the fully-dynamic maintenance algorithm for densest subgraph on hypergraphs, compared to, our maintained solution has a higher density, and is more stable.

Static Algorithms

- Notations:
- n=|V|, m=|E|, and $r = \max_{e \in E} |e|$ $\mathrm{M} = \sum_{e \in E} |e|$
- $E_u = \{e \in E : u \in e\}$ $E_u[S] = \{e \in E : u \in e \subseteq S\}$ $where\ S \subseteq V.$
- $F \subseteq E\ (resp. S \subseteq V)\ w(F) = \sum_{e \in F} w_e (resp. w(S) = \sum_{u \in S} w_u)$
- $\rho(S^*) = \max_{S \subseteq V} \rho(S)$ $\rho(S^*) \geq \frac{w(E)}{w(V)}$
- For an integer k>=1, we use [k] to denote {1,2,..k}.

Max-Flow-Based Exact Algorithm

- $\frac{w(E)}{w(V)} \leq \beta \leq w(E)\ G_\beta = \{s,t\} \cup V \cup E$
- $c(s,u) = \delta_u = \sum_{e \in E_u} \frac{w_e}{|e|}\quad c(u,t) = \beta w_u\quad c(u,e) = \frac{w_e}{|e|}\quad c(e,u) = \infty$
- *G_β has n+m+2 nodes*

LEMMA 2.1. *The maximum flow from s to t in G_β is less than $w(E)$ if and only if $\rho(S^*) > \beta$.*

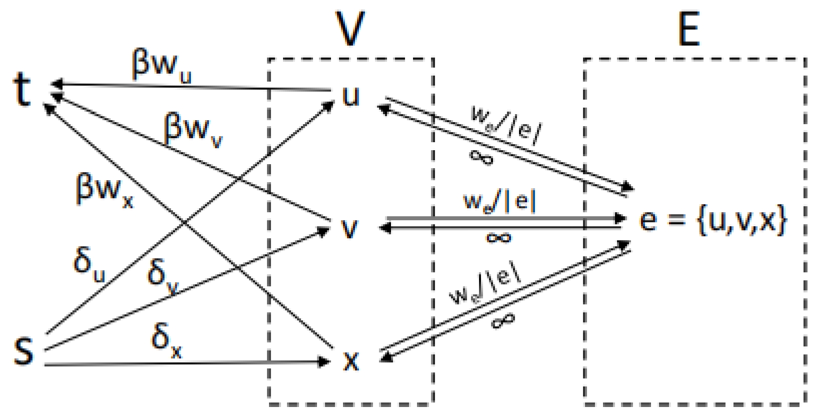


Figure 1: Auxiliary Graph G_β

Max-Flow-Based Exact Algorithm

Lemma 2.1. *The maximum flow from s to t in G_β is less than $w(E)$ if and only if $\rho(S^*) > \beta$.*

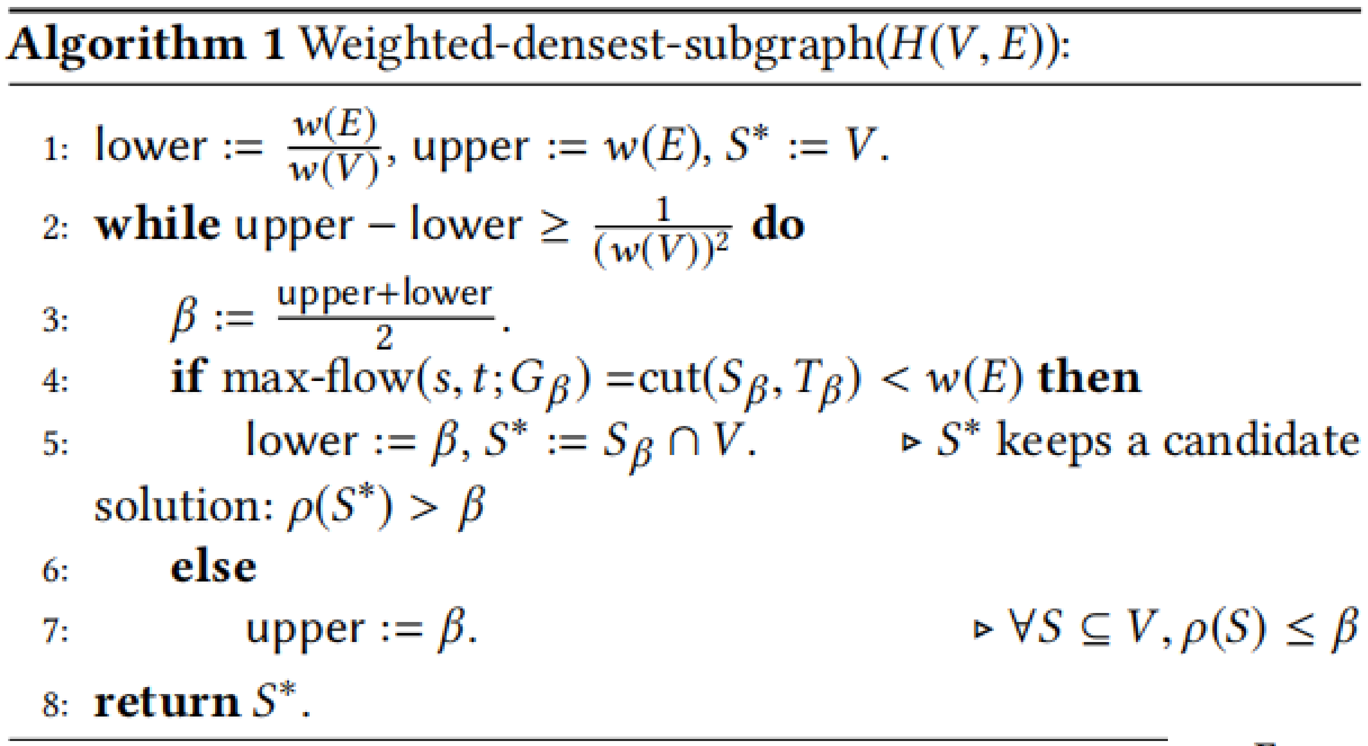
Proof. Note that we always have max-flow$(s,t) \le w(E)$ since there is an st-cut $(\{s\}, \{t\} \cup V \cup E)$ of capacity $\sum_{u \in V} \delta_u = w(E)$. Now suppose we compute the max-flow from s to t in G_β and find a minimum st-cut as $(\{s\} \cup V_1 \cup E_1, \{t\} \cup V_2 \cup E_2)$, where $V_2 = V \backslash V_1, E_2 = E \backslash E_1$, then we have (where cut(A, B) is the total capacities of edges from A to B):

max-flow$(s,t;G_\beta)$ = cut$(\{s\} \cup V_1 \cup E_1, \{t\} \cup V_2 \cup E_2)$
$= \sum_{u \in V_2} \delta_u + \sum_{u \in V_1} \beta w_u + \text{cut}(V_1, E_2) + \text{cut}(E_1, V_2)$.

First, observe that cut$(E_1, V_2) = 0$, since otherwise cut$(E_1, V_2) = \infty$; this implies that any edge e intersecting V_2 cannot be in E_1. On the other hand, since $(\{s\} \cup V_1 \cup E_1, \{t\} \cup V_2 \cup E_2)$ is a minimum st-cut, if there is an edge $e \subseteq V_1$ such that $e \in E_2$, then we can strictly reduce the cut by moving e from E_2 to E_1. Hence, we have shown that $E_1 = E[V_1]$ and $E_2 = E \backslash E[V_1]$ and have the following:

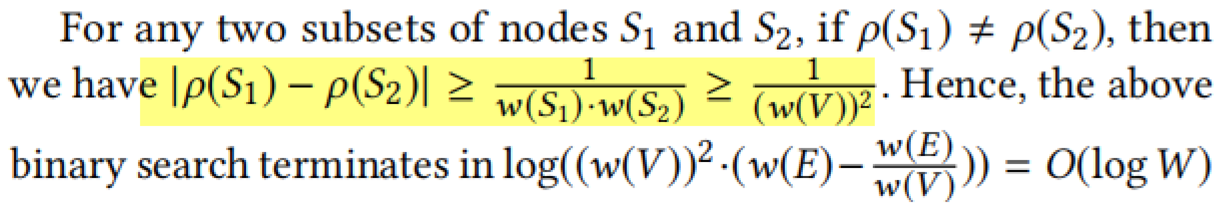
$$
\begin{aligned}
&\text{max-flow}(s,t;G_\beta)\\
&= \sum_{u \in V} \delta_u - \sum_{u \in V_1} \delta_u + \beta w(V_1) + \text{cut}(V_1, E \backslash E[V_1])\\
&= w(E) - (\text{cut}(V_1, E) - \beta w(V_1) - \text{cut}(V_1, E \backslash E[V_1]))\\
&= w(E) - (\text{cut}(V_1, E[V_1]) - \beta w(V_1))\\
&= w(E) - w(V_1)(\rho(V_1) - \beta).
\end{aligned}
$$

Max-Flow-Based Exact Algorithm



Algorithm 1 Weighted-densest-subgraph($H(V,E)$):

```
1: lower := $\frac{w(E)}{w(V)}$, upper := $w(E)$, $S^* := V$.
2: while upper − lower $\geq \frac{1}{(w(V))^2}$ do
3:     $\beta := \frac{\text{upper}+\text{lower}}{2}$.
4:     if max-flow$(s,t;G_\beta)$ =cut$(S_\beta, T_\beta) < w(E)$ then
5:         lower := $\beta$, $S^* := S_\beta \cap V$.        ▷ $S^*$ keeps a candidate solution: $\rho(S^*) > \beta$
6:     else
7:         upper := $\beta$.                              ▷ $\forall S \subseteq V, \rho(S) \leq \beta$
8: return $S^*$.
```



For any two subsets of nodes S_1 and S_2, if $\rho(S_1) \neq \rho(S_2)$, then we have $|\rho(S_1) - \rho(S_2)| \geq \frac{1}{w(S_1)\cdot w(S_2)} \geq \frac{1}{(w(V))^2}$. Hence, the above binary search terminates in $\log((w(V))^2 \cdot (w(E) - \frac{w(E)}{w(V)})) = O(\log W)$

LP-Based Exact Algorithm

$$
\begin{array}{lll}
\max & \sum_{e\in E} w_e x_e & \\
\text{s.t.} & x_e \le y_u, & \forall u \in e \\
& \sum_{u\in V} w_u y_u = 1, & \\
& x_e, y_u \ge 0, & \forall e \in E, u \in V.
\end{array}
$$

LEMMA 2.2. *Given any optimal solution $z^* = (y^*, x^*)$ for the above LP, $P = \{u \in V : y_u^* > 0\}$ induces a graph with maximum density.*

LP-Based Exact Algorithm

LEMMA 2.2. *Given any optimal solution $z^* = (y^*, x^*)$ for the above LP, $P = \{u \in V : y_u^* > 0\}$ induces a graph with maximum density.*

PROOF. First notice that given variables y_u, the objective is maximized when $x_e = \min_{u \in e} y_u$ for all $e \in E$ since $w_e \geq 0$. As noted above, for any $S \subseteq V$, we can derive a feasible solution $z^S = (y^S, x^S)$, whose objective value is $\rho(S)$. Let $\mathrm{LP}^* = \mathrm{LP}(z^*)$ be the optimal value of the LP, then for all $S \subseteq V$ we have

$$\mathrm{LP}^* \geq \mathrm{LP}(z^S) = \sum_{e \in E[S]} w_e \frac{1}{w(S)} = \rho(S). \tag{1}$$

Let $P \subseteq V$ be the nodes v such that $y_v^* > 0$. Let $a = w(P)$ and $b = \min_{u \in P} y_u^*$. Note that $ab \leq \sum_{u \in P} w_u y_u^* = 1$. Then we have $z^* = abz^P + (1 - ab)\widehat{z}$, where

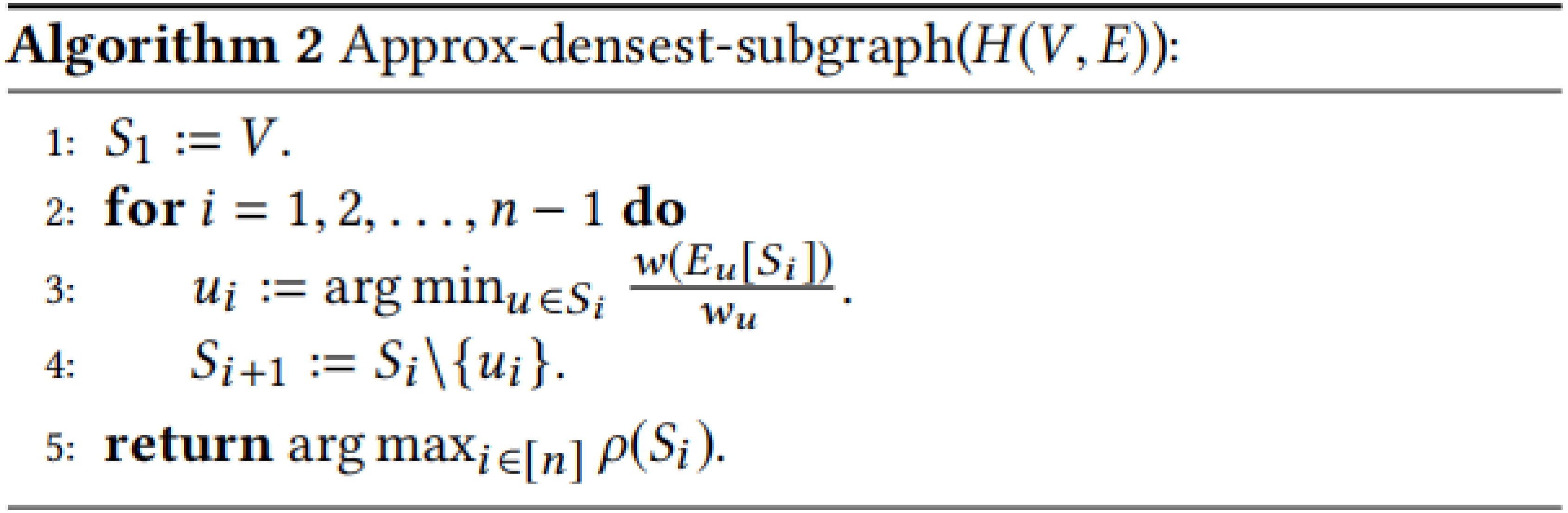
$$\widehat{z} = (\widehat{x}, \widehat{y}), \quad \widehat{y}_u = \max\{0, \frac{y_u^* - b}{1 - ab}\}, \quad \widehat{x}_e = \max\{0, \frac{x_e^* - b}{1 - ab}\}.$$

Note that $\widehat{z}$ is feasible since $\widehat{x}_e = \min_{u \in e} \widehat{y}_u$ and $\sum_{u \in V} w_u \widehat{y}_u = \frac{\sum_{u \in P} w_u y_u^* - ab}{1 - ab} = 1$.

Because the objective value is linear and the optimal solution is a convex combination of feasible solutions z^S and $\widehat{z}$, it follows that $\mathrm{LP}^* = \mathrm{LP}(z^*) = \mathrm{LP}(\widehat{z}) = \mathrm{LP}(z^P) = \rho(P)$, which combined with (1) implies that $\rho(P) \geq \max_{S \subseteq V} \rho(S)$. □

Near Linear-Time r-Approximation

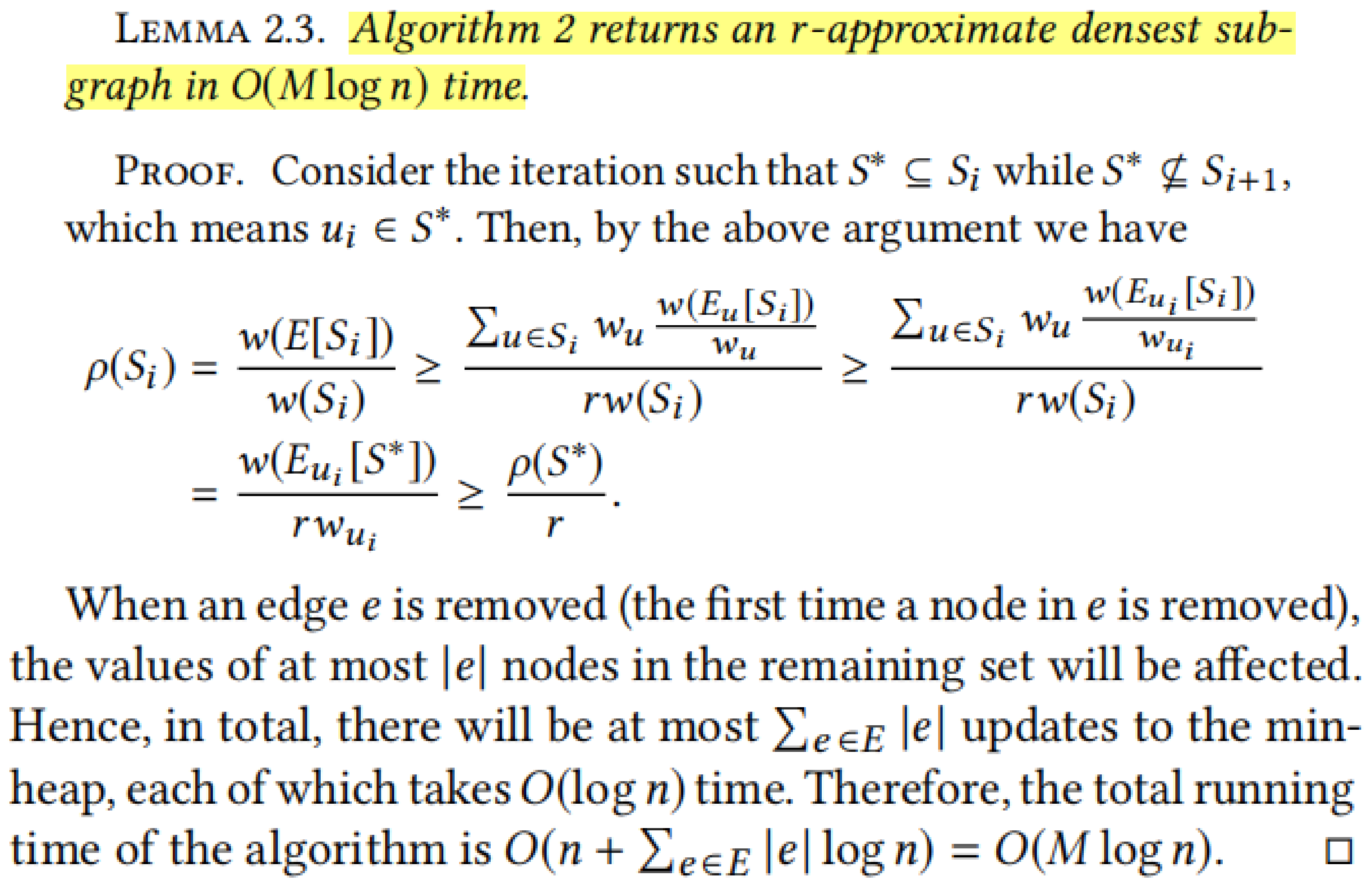
- $\frac{w(E_u[s^*])}{w_u} \geq \rho(S^*)$ as otherwise $\rho(S^*\{u\}) = \frac{w(E[s^*])-w(E_u[S^*])}{w(S^*)-w_u} > \rho(S^*)$



Algorithm 2 Approx-densest-subgraph($H(V,E)$):

```
1: $S_1 := V$.
2: for $i = 1, 2, \ldots, n-1$ do
3:     $u_i := \arg\min_{u \in S_i} \frac{w(E_u[S_i])}{w_u}$.
4:     $S_{i+1} := S_i \setminus \{u_i\}$.
5: return $\arg\max_{i \in [n]} \rho(S_i)$.
```

Near Linear-Time r-Approximation



LEMMA 2.3. *Algorithm 2 returns an r-approximate densest subgraph in $O(M \log n)$ time.*

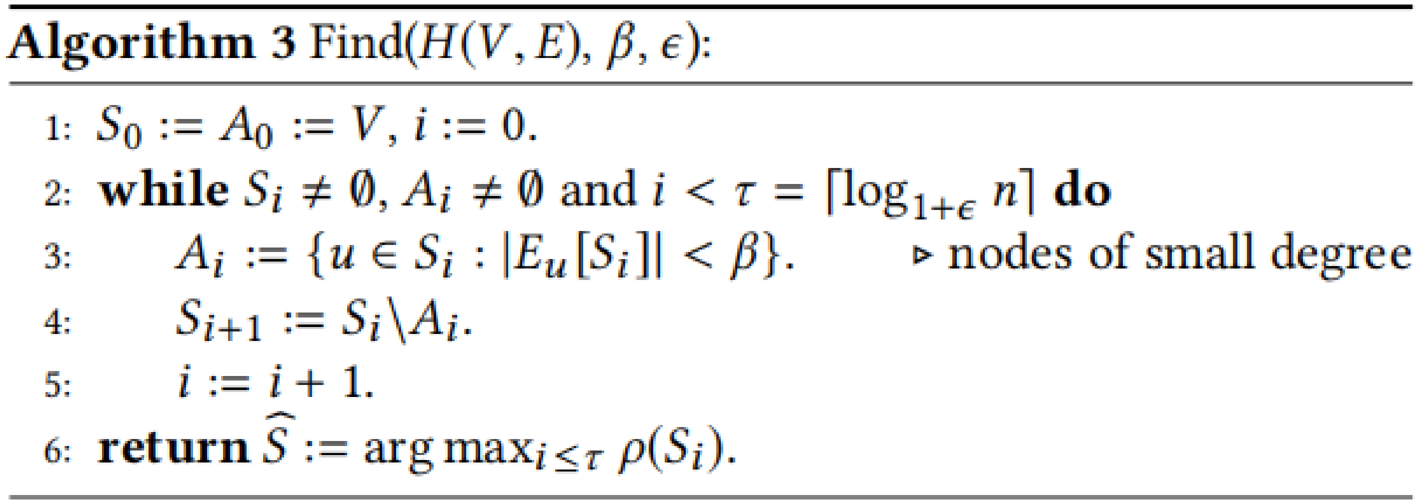
PROOF. Consider the iteration such that $S^* \subseteq S_i$ while $S^* \nsubseteq S_{i+1}$, which means $u_i \in S^*$. Then, by the above argument we have

$$\rho(S_i) = \frac{w(E[S_i])}{w(S_i)} \geq \frac{\sum_{u \in S_i} w_u \frac{w(E_u[S_i])}{w_u}}{rw(S_i)} \geq \frac{\sum_{u \in S_i} w_u \frac{w(E_{u_i}[S_i])}{w_{u_i}}}{rw(S_i)}$$
$$= \frac{w(E_{u_i}[S^*])}{rw_{u_i}} \geq \frac{\rho(S^*)}{r}.$$

When an edge e is removed (the first time a node in e is removed), the values of at most $|e|$ nodes in the remaining set will be affected. Hence, in total, there will be at most $\sum_{e \in E} |e|$ updates to the min-heap, each of which takes $O(\log n)$ time. Therefore, the total running time of the algorithm is $O(n + \sum_{e \in E} |e| \log n) = O(M \log n)$. □

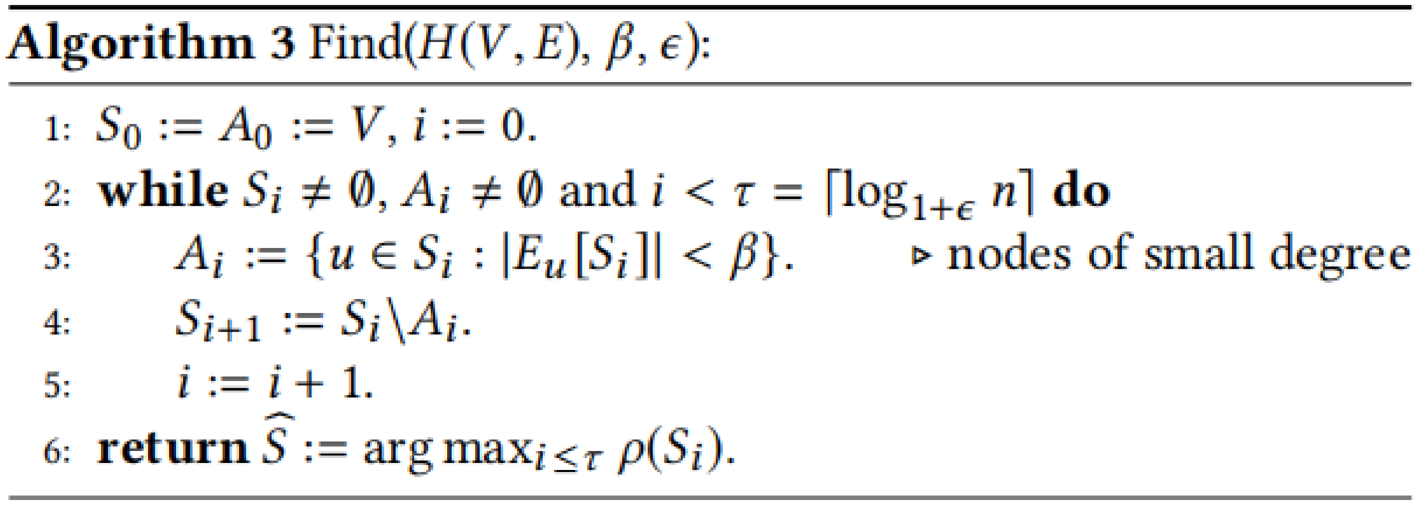
Incremental Algorithm

- $r(1+\epsilon)-approximate$, $\mathrm{poly}(\frac{r}{\epsilon}\log n)$ *time per edge insertion*
- Unweighted hypergraphs Define $\tau = \lceil \log_{1+\epsilon} n \rceil$



```
Algorithm 3 Find($H(V,E)$, $\beta$, $\epsilon$):
1: $S_0 := A_0 := V$, $i := 0$.
2: while $S_i \neq \emptyset$, $A_i \neq \emptyset$ and $i < \tau = \lceil \log_{1+\epsilon} n \rceil$ do
3:     $A_i := \{u \in S_i : |E_u[S_i]| < \beta\}$.        ▷ nodes of small degree
4:     $S_{i+1} := S_i \backslash A_i$.
5:     $i := i + 1$.
6: return $\widehat{S} := \arg\max_{i \leq \tau} \rho(S_i)$.
```

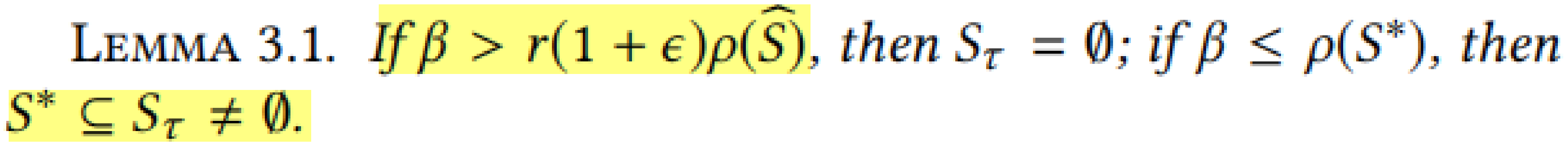
Static $r(1+\epsilon)$ Algorithm



Algorithm 3 Find($H(V,E)$, β, ϵ):

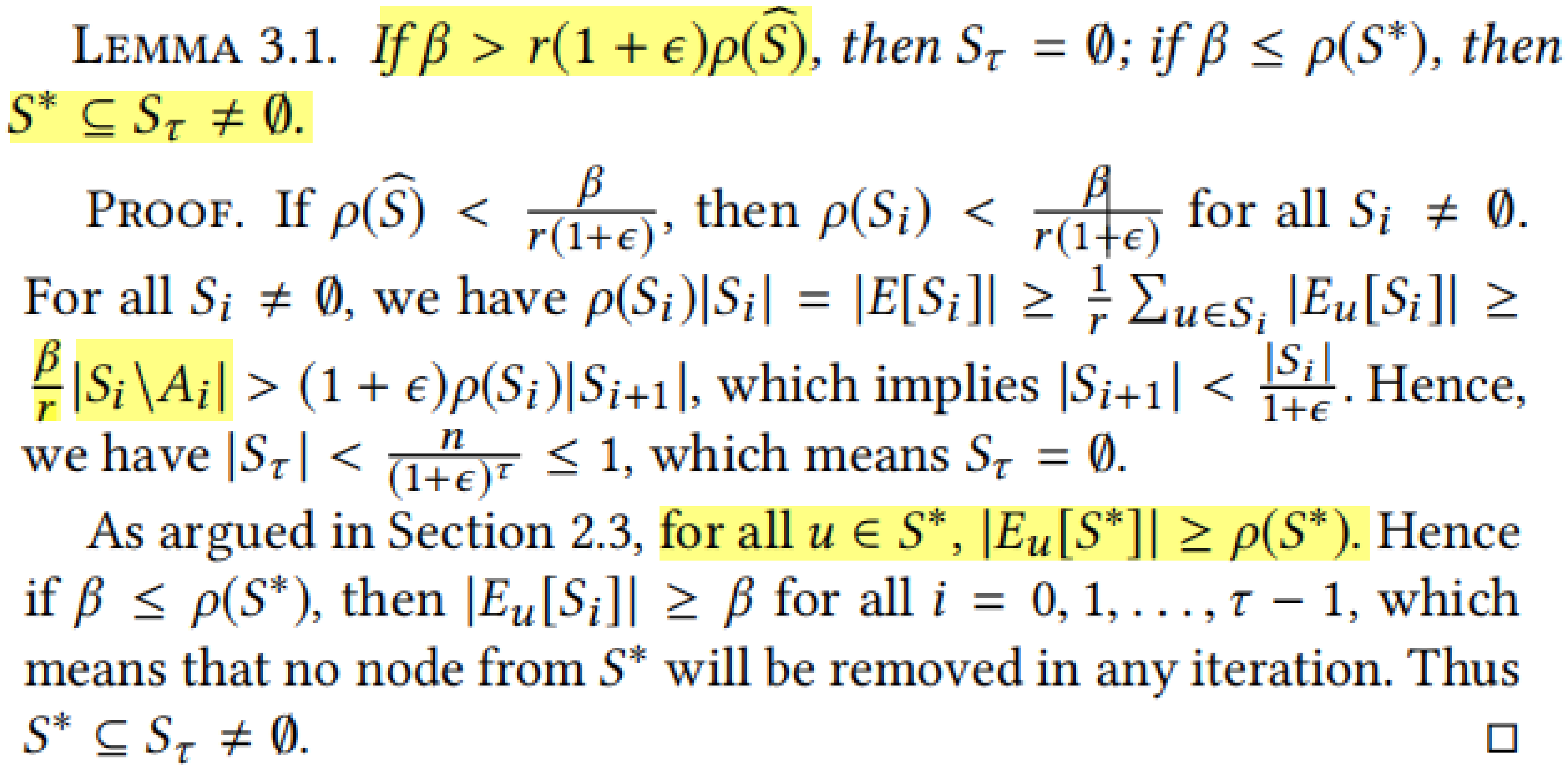
```
1: $S_0 := A_0 := V$, $i := 0$.
2: while $S_i \neq \emptyset$, $A_i \neq \emptyset$ and $i < \tau = \lceil \log_{1+\epsilon} n \rceil$ do
3:     $A_i := \{u \in S_i : |E_u[S_i]| < \beta\}$.        ▷ nodes of small degree
4:     $S_{i+1} := S_i \backslash A_i$.
5:     $i := i + 1$.
6: return $\widehat{S} := \arg\max_{i \le \tau} \rho(S_i)$.
```

- $A_\tau = S_\tau \quad (A_0, \ldots A_\tau) \quad S_i = A_{\geq i} = \cup_{j=1}^{\tau} A_j$



LEMMA 3.1. *If $\beta > r(1+\epsilon)\rho(\widehat{S})$, then $S_\tau = \emptyset$; if $\beta \leq \rho(S^*)$, then $S^* \subseteq S_\tau \neq \emptyset$.*

Static $r(1+\epsilon)$ Algorithm

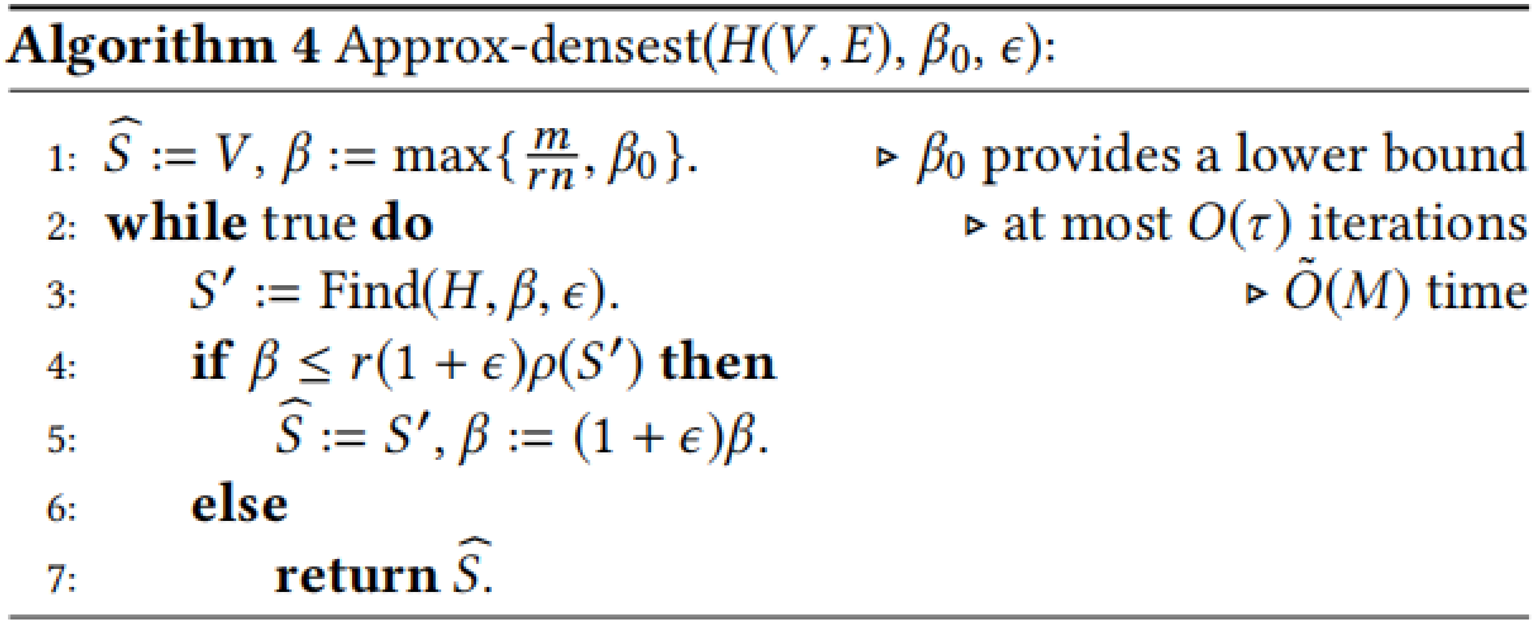


LEMMA 3.1. *If $\beta > r(1+\epsilon)\rho(\widehat{S})$, then $S_\tau = \emptyset$; if $\beta \leq \rho(S^*)$, then $S^* \subseteq S_\tau \neq \emptyset$.*

PROOF. If $\rho(\widehat{S}) < \frac{\beta}{r(1+\epsilon)}$, then $\rho(S_i) < \frac{\beta}{r(1+\epsilon)}$ for all $S_i \neq \emptyset$. For all $S_i \neq \emptyset$, we have $\rho(S_i)|S_i| = |E[S_i]| \geq \frac{1}{r}\sum_{u \in S_i} |E_u[S_i]| \geq \frac{\beta}{r}|S_i \backslash A_i| > (1+\epsilon)\rho(S_i)|S_{i+1}|$, which implies $|S_{i+1}| < \frac{|S_i|}{1+\epsilon}$. Hence, we have $|S_\tau| < \frac{n}{(1+\epsilon)^\tau} \leq 1$, which means $S_\tau = \emptyset$.

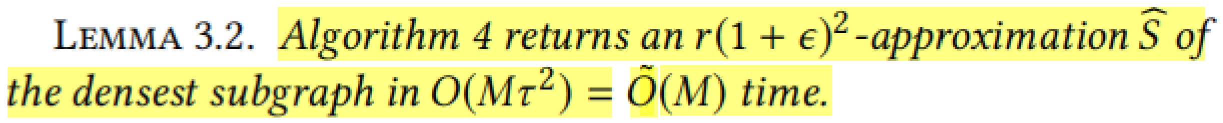
As argued in Section 2.3, for all $u \in S^*$, $|E_u[S^*]| \geq \rho(S^*)$. Hence if $\beta \leq \rho(S^*)$, then $|E_u[S_i]| \geq \beta$ for all $i = 0, 1, \ldots, \tau - 1$, which means that no node from S^* will be removed in any iteration. Thus $S^* \subseteq S_\tau \neq \emptyset$. $\square$

Static $r(1+\epsilon)$ Algorithm



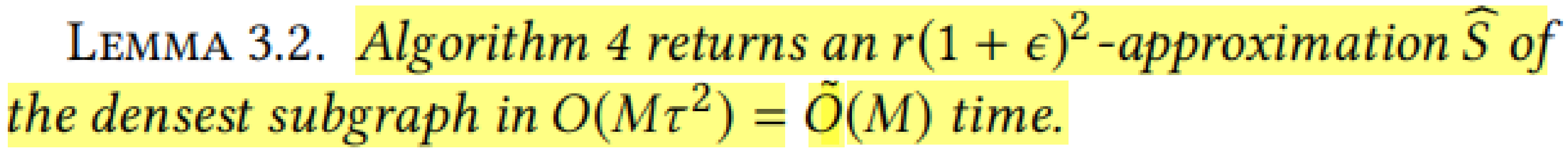
Algorithm 4 Approx-densest($H(V,E)$, β_0, ϵ):

```
1: $\widehat{S} := V$, $\beta := \max\{\frac{m}{rn}, \beta_0\}$.        ▷ $\beta_0$ provides a lower bound
2: while true do                                  ▷ at most $O(\tau)$ iterations
3:     $S' :=$ Find($H, \beta, \epsilon$).                      ▷ $\tilde{O}(M)$ time
4:     if $\beta \le r(1+\epsilon)\rho(S')$ then
5:         $\widehat{S} := S'$, $\beta := (1+\epsilon)\beta$.
6:     else
7:         return $\widehat{S}$.
```

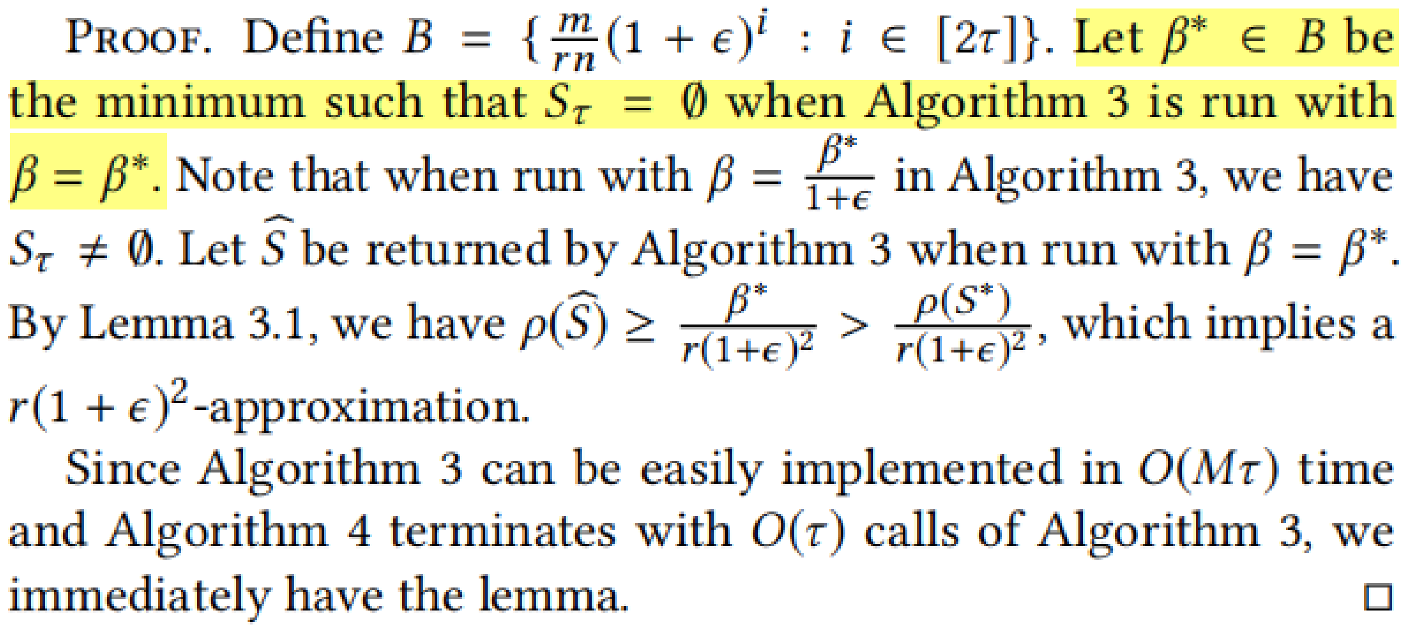


LEMMA 3.2. *Algorithm 4 returns an $r(1+\epsilon)^2$-approximation $\widehat{S}$ of the densest subgraph in $O(M\tau^2) = \tilde{O}(M)$ time.*

Static $r(1+\epsilon)$ Algorithm



LEMMA 3.2. *Algorithm 4 returns an $r(1+\epsilon)^2$-approximation $\widehat{S}$ of the densest subgraph in $O(M\tau^2) = \tilde{O}(M)$ time.*



PROOF. Define $B = \{\frac{m}{rn}(1+\epsilon)^i : i \in [2\tau]\}$. Let $\beta^* \in B$ be the minimum such that $S_\tau = \emptyset$ when Algorithm 3 is run with $\beta = \beta^*$. Note that when run with $\beta = \frac{\beta^*}{1+\epsilon}$ in Algorithm 3, we have $S_\tau \neq \emptyset$. Let $\widehat{S}$ be returned by Algorithm 3 when run with $\beta = \beta^*$. By Lemma 3.1, we have $\rho(\widehat{S}) \geq \frac{\beta^*}{r(1+\epsilon)^2} > \frac{\rho(S^*)}{r(1+\epsilon)^2}$, which implies a $r(1+\epsilon)^2$-approximation.

Since Algorithm 3 can be easily implemented in $O(M\tau)$ time and Algorithm 4 terminates with $O(\tau)$ calls of Algorithm 3, we immediately have the lemma. □

Edge Insertion-Only Setting

- $maintains\ (A_0, \dots A_\tau) \quad A_\tau = \emptyset$
- $u \in V\ l(u)\ be\ the\ level\ of\ u : u \in A_{l(u)} \quad b(u) = \left|E_u\left[S_{l(u)}\right]\right| < \beta$
- $l(e) = \min_{u \in e} l(u)\ \ for\ all\ e \in E$
- **Idea**:
- under edge insertions, the degrees of nodes could only increase and to maintain the partition, we increase the level of node u if $b(u) = \left|E_u\left[S_{l(u)}\right]\right| \geq \beta$after edge insertions. To guarantee the approximation ratio, we rebuild the partition if $A_\tau \neq \emptyset$.

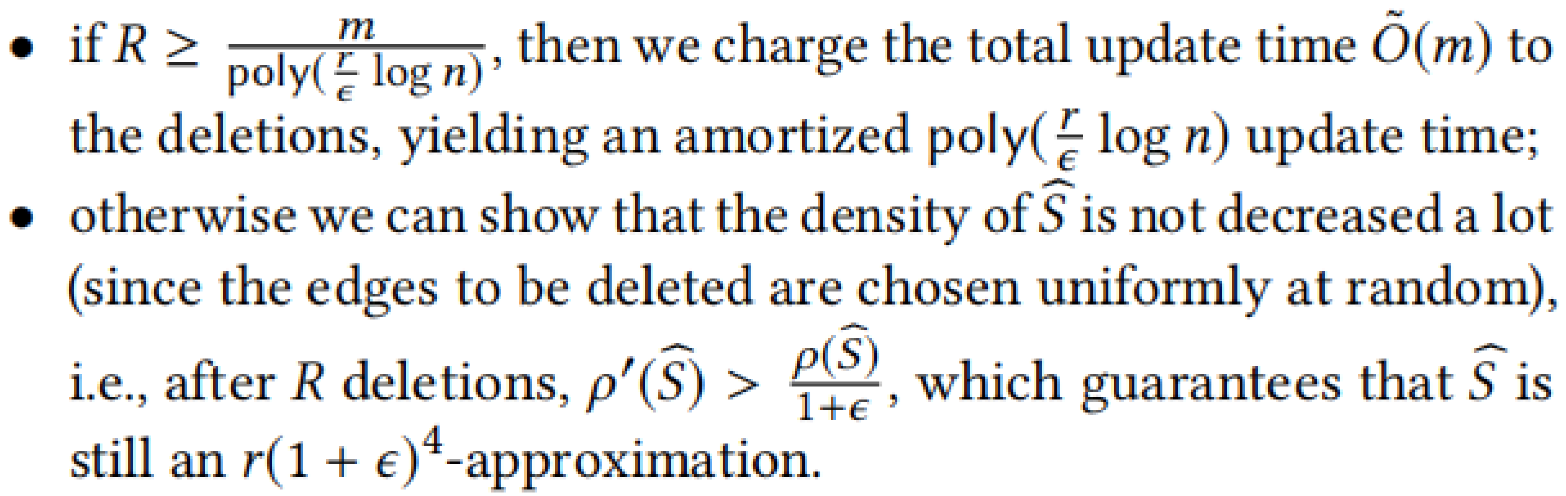
Edge Insertion-Only Setting

Algorithm 5 Insertion-only-approx-densest($H(V,E)$, ϵ):

```
$\widehat{S}$ := Approx-densest($H, 0, \epsilon$),
let $A_i$, $S_i$ and $\beta$ be as in the last call of Find().        ▹ $S_\tau = \emptyset$
**for** each newly inserted edge $e$ **do**
    $E := E \cup \{e\}$ and update $b(u)$ for all $u \in e$.        ▹ $O(|e|)$ time
    label all nodes in $e$ "bad".
    **while** exists a bad node **do**
        pick a bad node $u$, label $u$ "good" and let $l'(u) := l(u)$.
        **while** $b(u) \geq \beta$ and $l'(u) < \tau$ **do**
            $l'(u) := l'(u) + 1$,
            $b(u) := |\{e \in E_u : \min_{v \in e \setminus \{u\}} l(v) \geq l'(u)\}|$.
        **if** $l'(u) > l(u)$ **then**
            **for** each $v \in N(u)$ s.t. $l(u) < l(v) \leq l'(u)$ **do**
                update $b(v)$, label $v$ "bad".
            $l(u) := l'(u)$.
        **if** $l(u) = \tau$ **then**
            **Rebuild**: $\widehat{S}$ := approx-densest($H, \beta, \epsilon$),
            update $A_i$, $S_i$, $\beta$, $l()$ and $b()$.
            label all nodes "good".
```

Edge Insertion-Only Setting

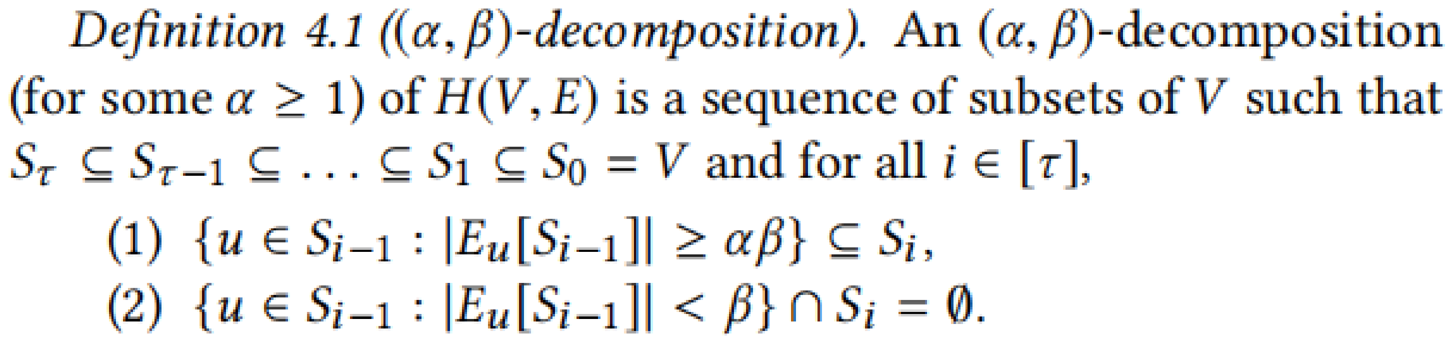
- **Approximation Ratio**: $r(1+\epsilon) - approximation$
- **Update Time**: $\max(O(M + \tau r^2 m), O(\frac{r}{\epsilon}\log n))$
- **Space Complexity**: O(n)
- Remark:



- if $R \geq \frac{m}{\text{poly}(\frac{r}{\epsilon}\log n)}$, then we charge the total update time $\tilde{O}(m)$ to the deletions, yielding an amortized $\text{poly}(\frac{r}{\epsilon}\log n)$ update time;
- otherwise we can show that the density of $\widehat{S}$ is not decreased a lot (since the edges to be deleted are chosen uniformly at random), i.e., after R deletions, $\rho'(\widehat{S}) > \frac{\rho(\widehat{S})}{1+\epsilon}$, which guarantees that $\widehat{S}$ is still an $r(1+\epsilon)^4$-approximation.

Fully Dynamic Approximation

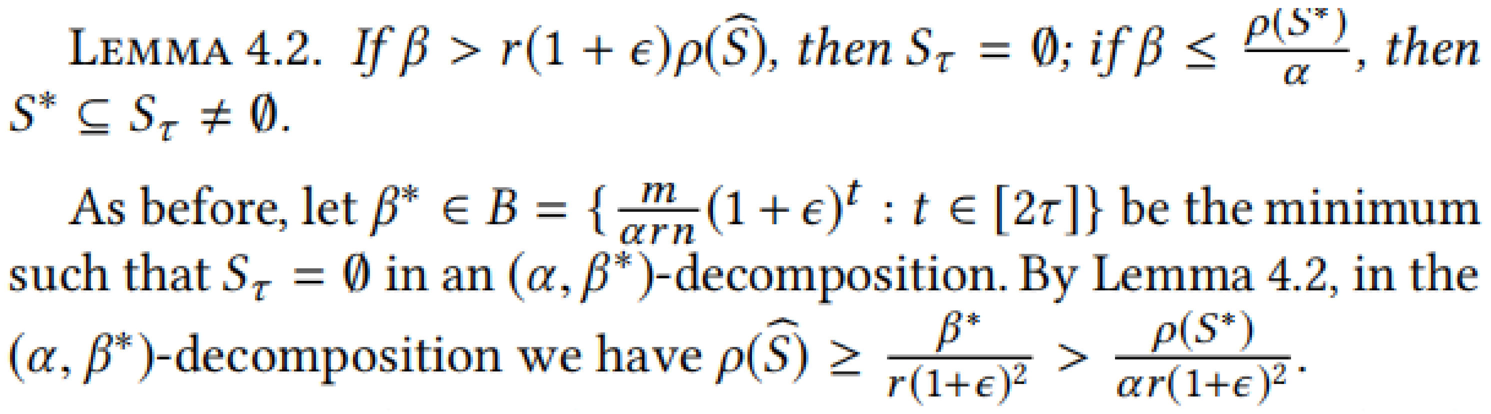
- **Lazy update**: For a fixed threshold β, we remove nodes with degree less than β while keeping nodes with degree at least $\alpha\beta$, for some $\alpha > 1$.



Definition 4.1 ((α, β)-decomposition). An (α, β)-decomposition (for some $\alpha \geq 1$) of $H(V, E)$ is a sequence of subsets of V such that $S_\tau \subseteq S_{\tau-1} \subseteq \ldots \subseteq S_1 \subseteq S_0 = V$ and for all $i \in [\tau]$,

(1) $\{u \in S_{i-1} : |E_u[S_{i-1}]| \geq \alpha\beta\} \subseteq S_i$,
(2) $\{u \in S_{i-1} : |E_u[S_{i-1}]| < \beta\} \cap S_i = \emptyset$.

- $A_i = S_i \backslash S_{i+1}, A_\tau = S_\tau$
- $\hat{S} = argmax_{i \leq \tau} \rho(S_i)$



Lemma 4.2. *If $\beta > r(1+\epsilon)\rho(\widehat{S})$, then $S_\tau = \emptyset$; if $\beta \leq \frac{\rho(S^*)}{\alpha}$, then $S^* \subseteq S_\tau \neq \emptyset$.*

As before, let $\beta^* \in B = \{\frac{m}{\alpha r n}(1+\epsilon)^t : t \in [2\tau]\}$ be the minimum such that $S_\tau = \emptyset$ in an (α, β^*)-decomposition. By Lemma 4.2, in the (α, β^*)-decomposition we have $\rho(\widehat{S}) \geq \frac{\beta^*}{r(1+\epsilon)^2} > \frac{\rho(S^*)}{\alpha r(1+\epsilon)^2}$.

Maintaining an (α, β) - Decomposition

- l(u) , l(e) as the levels of nodes and edges in partitioning $(\mathrm{A}_0, \ldots \mathrm{A}_\tau)$
- $For\ all\ i \leq l(u),\ E_u^{(i)} = E_u[S_i] - E_u[S_{i+1}]$ be the hyperedges adjacent to u that are removed at level i.
- $\left(E_u^{(0)}, E_u^{(1)}, \ldots E_u^{(l(u))}\right)$define s a partition of E_u
- For all $i \leq l(u), b_i(u) = |E_u[S_i]|$
- $b_{l(u)}(u) = \left|E_u\left[S_{l(u)}\right]\right| < \alpha\beta\ for\ all\ u \notin S_\tau\ and b_{l(u)-1}(u) \geq \beta\ for\ all\ u \notin A_0$

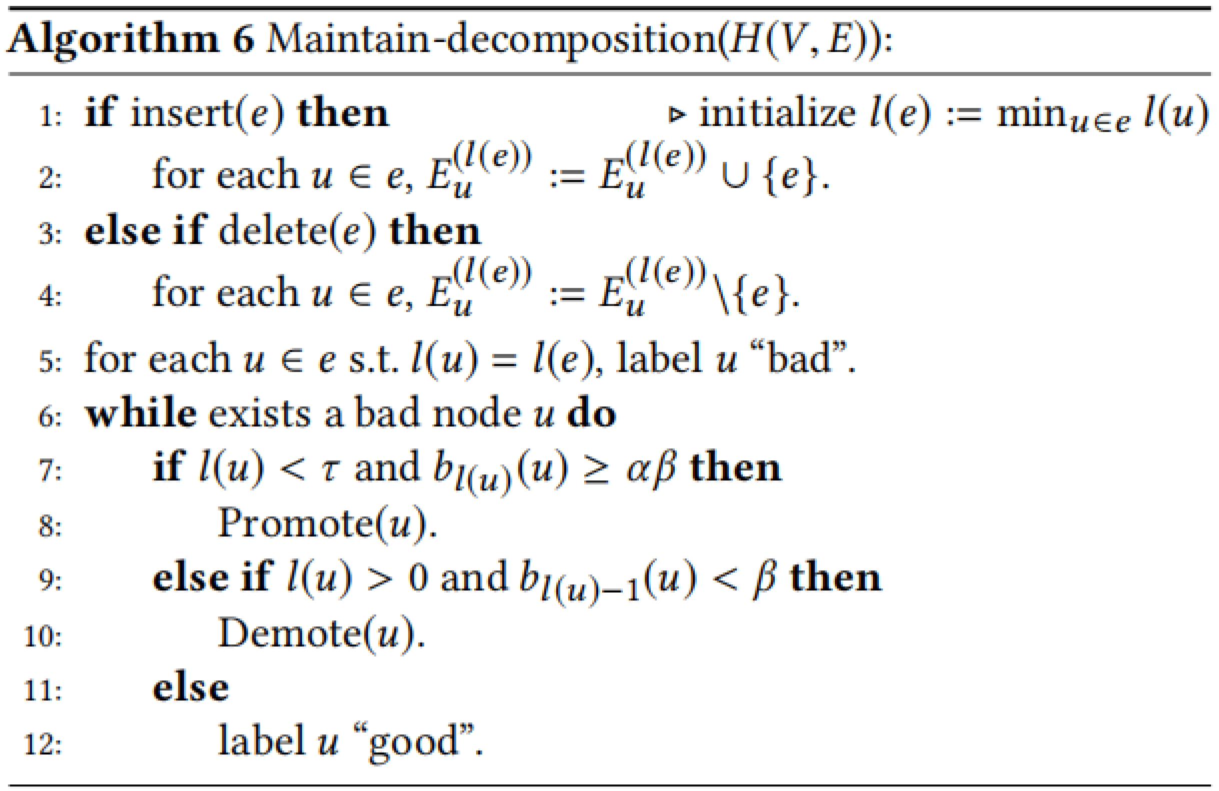
Maintaining an (α, β) - Decomposition

- l(u) , l(e) as the levels of nodes and edges in partitioning $(A_0, \ldots A_\tau)$
- $For\ all\ i \leq l(u),\ E_u^{(i)} = E_u[S_i] - E_u[S_{i+1}]$ be the hyperedges adjacent to u that are removed at level i.
- $\left(E_u^{(0)}, E_u^{(1)}, \ldots E_u^{(l(u))}\right)$define s a partition of E_u
- For all $i \leq l(u), b_i(u) = |E_u[S_i]|$
- $b_{l(u)}(u) = \left|E_u\left[S_{l(u)}\right]\right| < \alpha\beta\ for\ all\ u \notin S_\tau\ and b_{l(u)-1}(u) \geq \beta\ for\ all\ u \notin A_0$

Maintaining an (α, β) - Decomposition

- We maintain for each $u \in V$ its level l(u), the partitioning $\left(E_u^{(0)}, E_u^{(1)}, \ldots E_u^{(l(u))}\right)$ of E_u and the degree of u at each level $b_0(u) \ldots b_{l(u)}(u)$.
- We further maintain l(e) for every $e \in E \; and \; \rho(S_i)$
- The other can be updated by l(u) and $E_u^{(j)}$

Maintaining an (α, β) - Decomposition

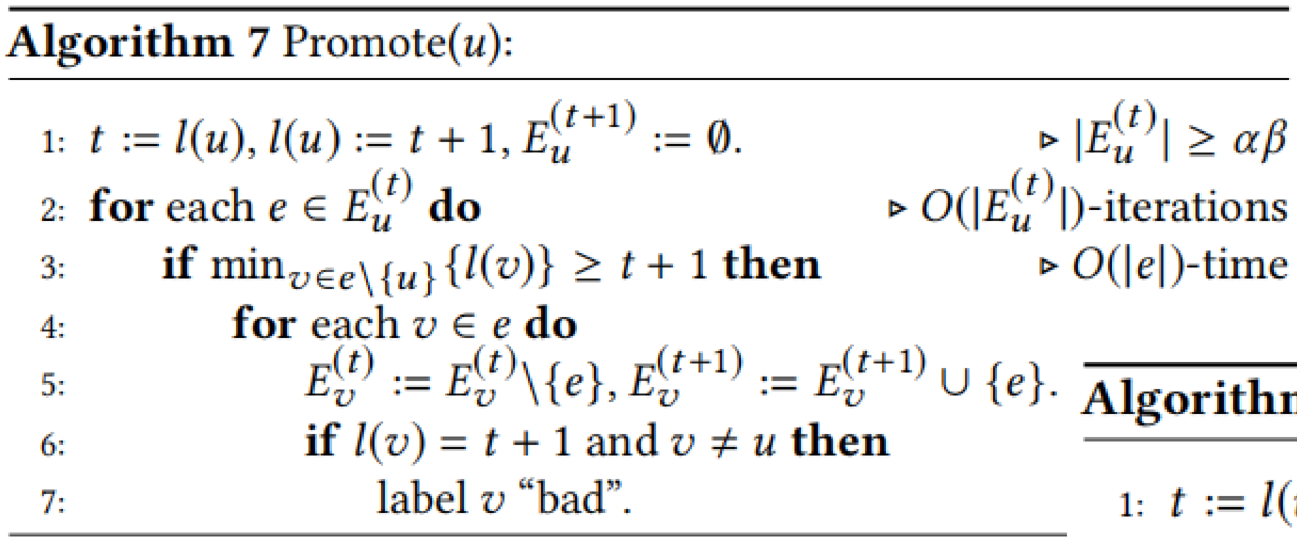


Algorithm 6 Maintain-decomposition($H(V,E)$):

```
1:  if insert($e$) then                              ▷ initialize $l(e) := \min_{u \in e} l(u)$
2:      for each $u \in e$, $E_u^{(l(e))} := E_u^{(l(e))} \cup \{e\}$.
3:  else if delete($e$) then
4:      for each $u \in e$, $E_u^{(l(e))} := E_u^{(l(e))} \setminus \{e\}$.
5:  for each $u \in e$ s.t. $l(u) = l(e)$, label $u$ "bad".
6:  while exists a bad node $u$ do
7:      if $l(u) < \tau$ and $b_{l(u)}(u) \geq \alpha\beta$ then
8:          Promote($u$).
9:      else if $l(u) > 0$ and $b_{l(u)-1}(u) < \beta$ then
10:         Demote($u$).
11:     else
12:         label $u$ "good".
```

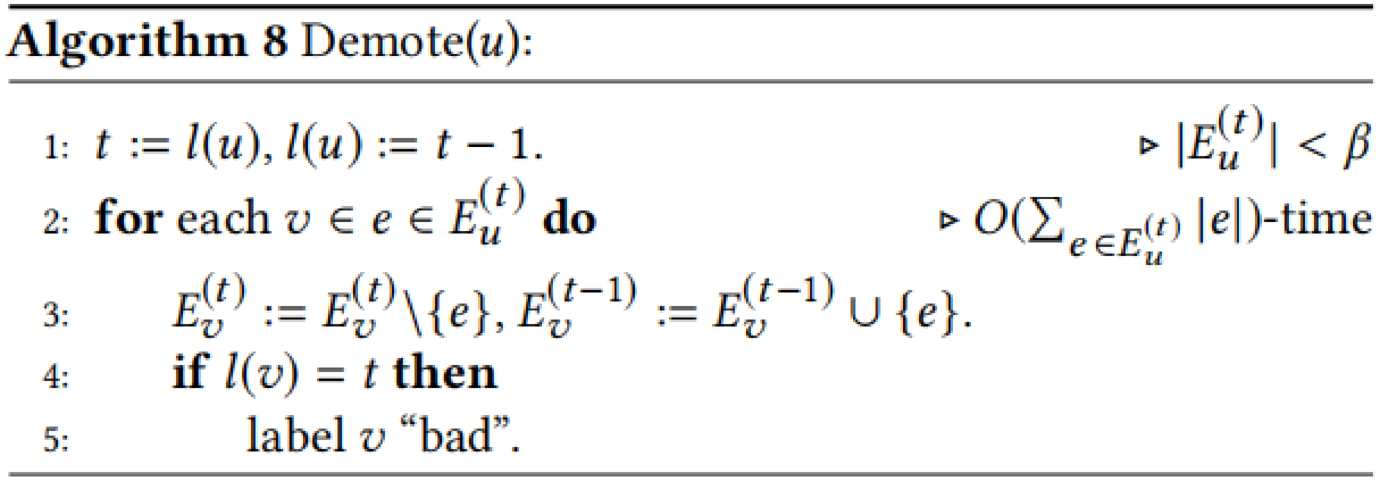
- Update the partitioning of each E_u and guarantees $b_{l(u)}(u) < \alpha\beta$ *and* $b_{l(u)-1}(u) > \beta$

Maintaining an (α, β) - Decomposition



Algorithm 7 Promote(u):

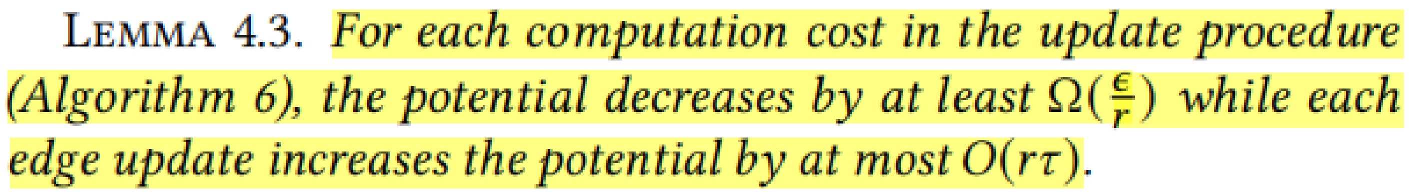
```
1: $t := l(u)$, $l(u) := t + 1$, $E_u^{(t+1)} := \emptyset$.        ▷ $|E_u^{(t)}| \geq \alpha\beta$
2: for each $e \in E_u^{(t)}$ do        ▷ $O(|E_u^{(t)}|)$-iterations
3:     if $\min_{v \in e \setminus \{u\}} \{l(v)\} \geq t + 1$ then        ▷ $O(|e|)$-time
4:         for each $v \in e$ do
5:             $E_v^{(t)} := E_v^{(t)} \setminus \{e\}$, $E_v^{(t+1)} := E_v^{(t+1)} \cup \{e\}$.
6:             if $l(v) = t + 1$ and $v \neq u$ then
7:                 label $v$ "bad".
```



Algorithm 8 Demote(u):

```
1: $t := l(u)$, $l(u) := t - 1$.        ▷ $|E_u^{(t)}| < \beta$
2: for each $v \in e \in E_u^{(t)}$ do        ▷ $O(\sum_{e \in E_u^{(t)}} |e|)$-time
3:     $E_v^{(t)} := E_v^{(t)} \setminus \{e\}$, $E_v^{(t-1)} := E_v^{(t-1)} \cup \{e\}$.
4:     if $l(v) = t$ then
5:         label $v$ "bad".
```

Maintaining an (α,β) - Decomposition


LEMMA 4.3. *For each computation cost in the update procedure (Algorithm 6), the potential decreases by at least $\Omega(\frac{\epsilon}{r})$ while each edge update increases the potential by at most $O(r\tau)$.*

- **Insert**(e): $P' - P \le P'(e) \le r\tau$.
- **Delete**(e): $P' - P \le \sum_{u\in e}(P'(u) - P(u)) \le \epsilon|e|\tau \le \epsilon r\tau$.

Maintaining an (α, β) - Decomposition

Promote(u): assume $l(u) = t$, then $b_t(u) \geq \alpha\beta$, $l'(u) = t + 1$ and $S'_{t+1} = S_{t+1} \cup \{u\}$. The potential of nodes and edges are changed as follows.

- Since $S'_i = S_i$ for all $i \leq t$, we have

$$P(u) - P'(u) = -\max\{0, \alpha\beta - \epsilon b_t(u)\} \geq \epsilon b_t(u) - \alpha\beta.$$

- For all $v \in e \in E_u[S_t]$ s.t. $l(v) \geq t + 2$,

$$P(v) - P'(v) = \max\{0, \alpha\beta - \epsilon b_{t+1}(v)\} - \max\{0, \alpha\beta - \epsilon b'_{t+1}(v)\} \geq 0.$$

- For all other nodes v, $P(v) - P'(v) = 0$.

- For all $e \in E_u[S_t]$ s.t. $\min_{v \in e \setminus \{u\}}\{l(v)\} \geq t + 1$,

$$P(e) - P'(e) \geq r(l'(e) - l(e) + \frac{1}{|e|} - 1) = \frac{r}{|e|} \geq 1.$$

- For all $e \in E_u[S_t]$ s.t. $\min_{v \in e \setminus \{u\}}\{l(v)\} = t$,

$$P(e) - P'(e) \geq \frac{r}{|e|} \geq 1.$$

- For all other edges e, $P(e) - P'(e) = 0$.

Hence, overall the total potential is decreased by at least $\mathrm{P} - \mathrm{P}' \geq \epsilon b_t(u) - \alpha\beta + |E_u[S_t]| \geq \epsilon|E_u[S_t]|$. Since each promotion executes in $O(r|E_u^{(t)}|) = O(r|E_u[S_t]|)$ time, for each computation cost, the potential is decreased by $\Omega(\frac{\epsilon}{r})$.

Maintaining an (α, β) - Decomposition

Demote(u): assume $l(u) = t$, then $b_{t-1}(u) < \beta$, $l'(u) = t - 1$ and $S'_t = S_t \backslash \{u\}$. The potential of nodes and edges are changed as follows.

- Since $S'_i = S_i$ for all $i \le t$, we have $P(u) - P'(u) = \max\{0, \alpha\beta - \epsilon b_{t-1}(u)\} = \alpha\beta - \epsilon b_{t-1}(u)$.
- For all $v \in e \in E_u[S_t]$ s.t. $l(v) \ge t+1$, $P(v) - P'(v) = \max\{0, \alpha\beta - \epsilon b_t(v)\} - \max\{0, \alpha\beta - \epsilon b'_t(v)\} \ge -\epsilon(b_t(v) - b'_t(v))$, which means that the increase in potential of each such node v is at most ϵ fraction of the number of hyperedges adjacent to v at level t that are removed due to the demotion of u. Hence, the total decrease of potential of those nodes is $\sum_{v \in e \in E_u[S_t] \text{ s.t. } l(v) \ge t+1} P(v) - P'(v) \ge -\epsilon \sum_{e \in E_u[S_t]} |e| \ge -\epsilon r |E_u[S_t]|$.
- For all other nodes v, $P(v) - P'(v) = 0$.
- For all $e \in E_u[S_t]$, $P(e) - P'(e) \ge r(l'(e) - l(e) + \frac{1}{|e|} - \frac{1}{|e|}) = -r$.
- For all $e \in E_u^{(t-1)}$, $P(e) - P'(e) \ge -\frac{r}{|e|} \ge -r$.
- For all other edges e, $P(e) - P'(e) = 0$.

  Hence, the total potential decrease by (when $\alpha = r(1 + 3\epsilon)$)

$$\begin{aligned} \mathrm{P} - \mathrm{P}' &\ge \alpha\beta - \epsilon b_{t-1}(u) - \epsilon r b_t(u) - r|E_u(S_t)| - r|E_u^{(t-1)}| \\ &\ge \alpha\beta - (\epsilon + \epsilon r + r) b_{t-1}(u) \ge \epsilon |E_u[S_{t-1}]|. \end{aligned}$$

Since each demotion executes in $O(r|E_u^{(t)}|) = O(r|E_u[S_{t-1}]|)$ time, for each computation cost, the potential is decreased by $\Omega(\frac{\epsilon}{r})$, which completes the analysis. □

Maintaining an (α, β) - Decomposition

Demote(u): assume $l(u) = t$, then $b_{t-1}(u) < \beta$, $l'(u) = t - 1$ and $S'_t = S_t \backslash \{u\}$. The potential of nodes and edges are changed as follows.

- Since $S'_i = S_i$ for all $i \leq t$, we have $P(u) - P'(u) = \max\{0, \alpha\beta - \epsilon b_{t-1}(u)\} = \alpha\beta - \epsilon b_{t-1}(u)$.
- For all $v \in e \in E_u[S_t]$ s.t. $l(v) \geq t+1$, $P(v) - P'(v) = \max\{0, \alpha\beta - \epsilon b_t(v)\} - \max\{0, \alpha\beta - \epsilon b'_t(v)\} \geq -\epsilon(b_t(v) - b'_t(v))$, which means that the increase in potential of each such node v is at most ϵ fraction of the number of hyperedges adjacent to v at level t that are removed due to the demotion of u. Hence, the total decrease of potential of those nodes is $\sum_{v \in e \in E_u[S_t] \text{ s.t. } l(v) \geq t+1} P(v) - P'(v) \geq -\epsilon \sum_{e \in E_u[S_t]} |e| \geq -\epsilon r |E_u[S_t]|$.
- For all other nodes v, $P(v) - P'(v) = 0$.
- For all $e \in E_u[S_t]$, $P(e) - P'(e) \geq r(l'(e) - l(e) + \frac{1}{|e|} - \frac{1}{|e|}) = -r$.
- For all $e \in E_u^{(t-1)}$, $P(e) - P'(e) \geq -\frac{r}{|e|} \geq -r$.
- For all other edges e, $P(e) - P'(e) = 0$.

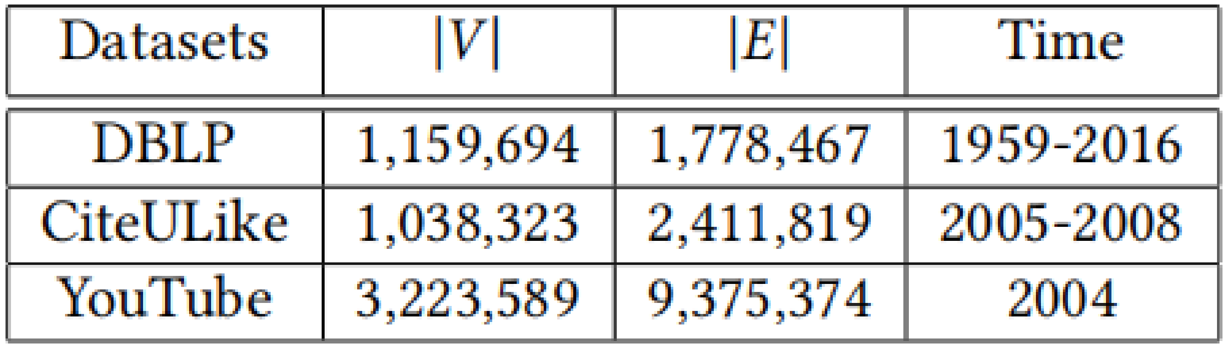
  Hence, the total potential decrease by (when $\alpha = r(1 + 3\epsilon)$)

$$\mathrm{P} - \mathrm{P}' \geq \alpha\beta - \epsilon b_{t-1}(u) - \epsilon r b_t(u) - r|E_u(S_t)| - r|E_u^{(t-1)}|$$
$$\geq \alpha\beta - (\epsilon + \epsilon r + r) b_{t-1}(u) \geq \epsilon |E_u[S_{t-1}]|.$$

Since each demotion executes in $O(r|E_u^{(t)}|) = O(r|E_u[S_{t-1}]|)$ time, for each computation cost, the potential is decreased by $\Omega(\frac{\epsilon}{r})$, which completes the analysis. $\square$

Experiments

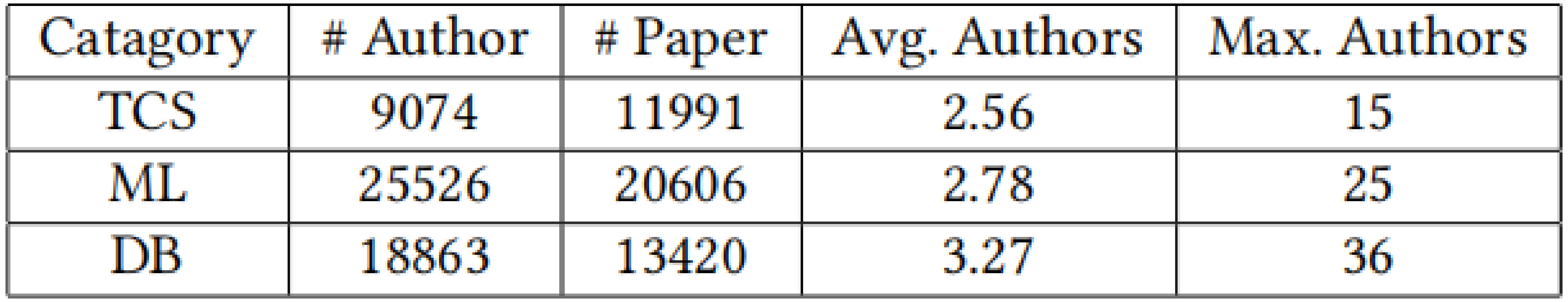
Datasets:



Datasets	$\|V\|$	$\|E\|$	Time
DBLP	1,159,694	1,778,467	1959-2016
CiteULike	1,038,323	2,411,819	2005-2008
YouTube	3,223,589	9,375,374	2004

Experiments

Exact vs Approximation

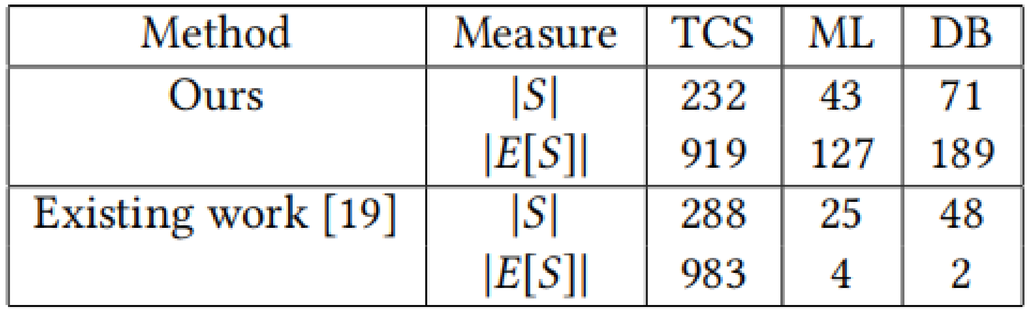


Catagory	# Author	# Paper	Avg. Authors	Max. Authors
TCS	9074	11991	2.56	15
ML	25526	20606	2.78	25
DB	18863	13420	3.27	36

Table 2: Properties of publications, where Avg. Authors denotes the average number of authors per paper and Max. Author denotes the maximum number of authors in a paper.

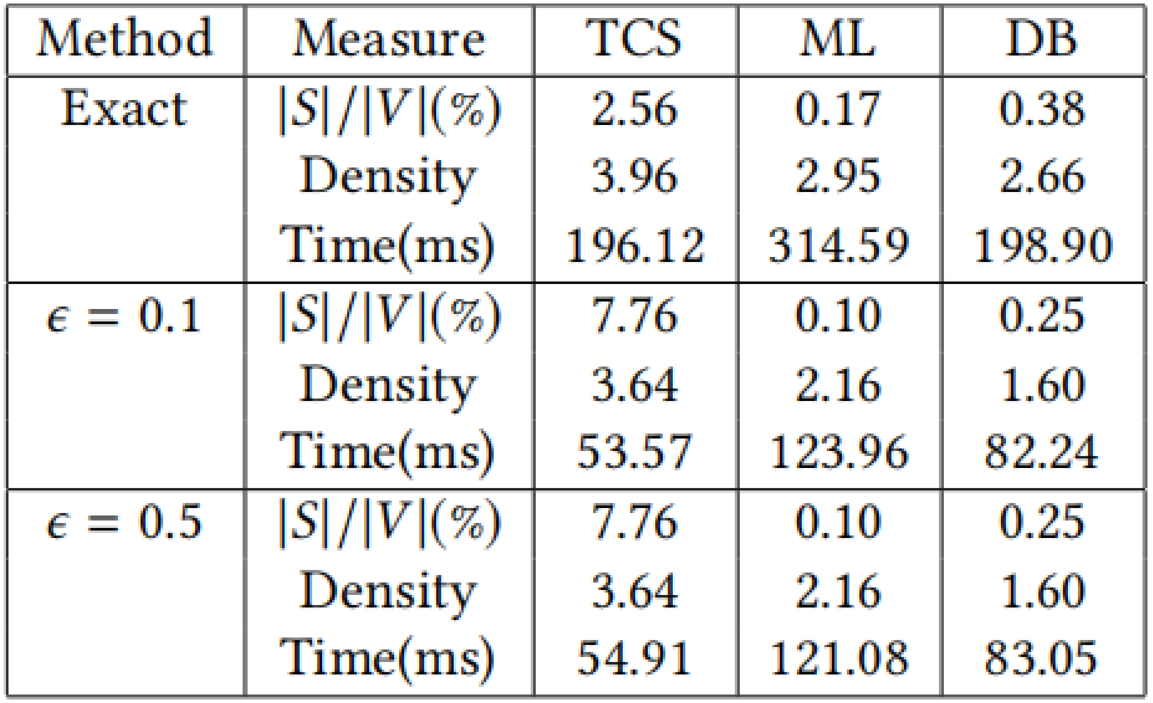
Experiments

Exact vs Approximation



Method	Measure	TCS	ML	DB
Ours	$\|S\|$	232	43	71
	$\|E[S]\|$	919	127	189
Existing work [19]	$\|S\|$	288	25	48
	$\|E[S]\|$	983	4	2

Table 4: Comparison of hyperedge density

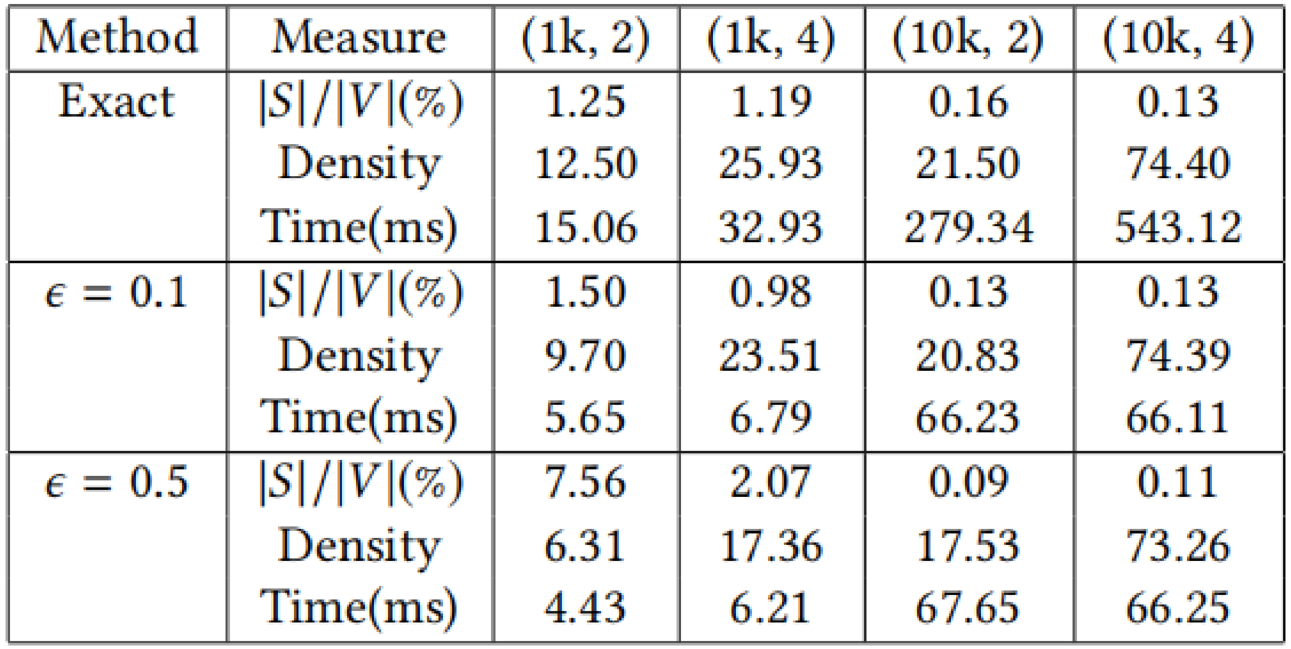


Method	Measure	TCS	ML	DB
Exact	$\|S\|/\|V\|(\%)$	2.56	0.17	0.38
	Density	3.96	2.95	2.66
	Time(ms)	196.12	314.59	198.90
$\epsilon = 0.1$	$\|S\|/\|V\|(\%)$	7.76	0.10	0.25
	Density	3.64	2.16	1.60
	Time(ms)	53.57	123.96	82.24
$\epsilon = 0.5$	$\|S\|/\|V\|(\%)$	7.76	0.10	0.25
	Density	3.64	2.16	1.60
	Time(ms)	54.91	121.08	83.05

Table 3: Performance on real datasets

Experiments

Synthetic Datasets

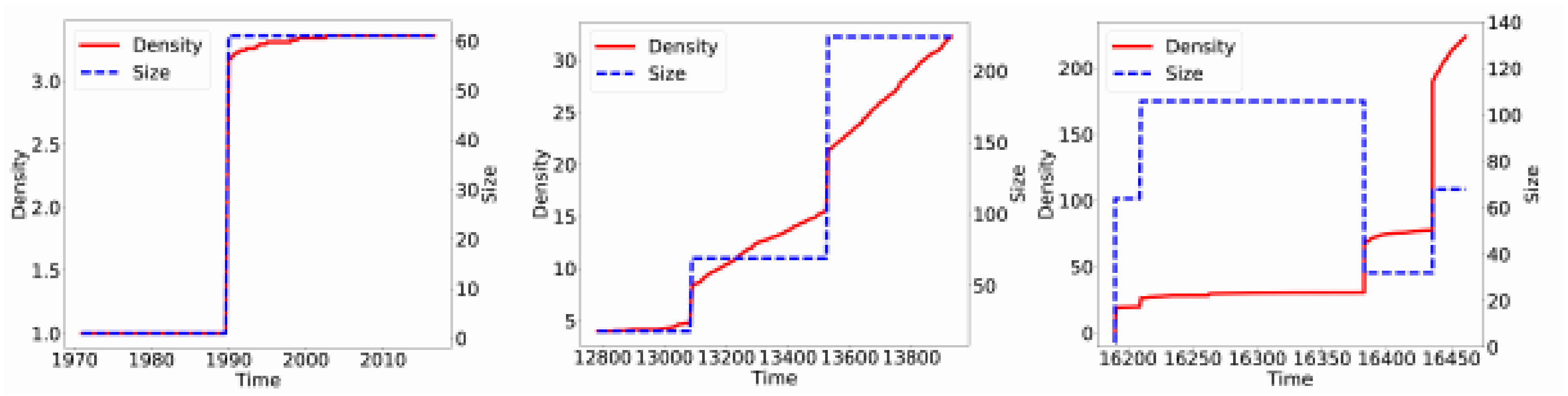


Method	Measure	(1k, 2)	(1k, 4)	(10k, 2)	(10k, 4)
Exact	$\lvert S\rvert/\lvert V\rvert(\%)$	1.25	1.19	0.16	0.13
	Density	12.50	25.93	21.50	74.40
	Time(ms)	15.06	32.93	279.34	543.12
$\epsilon = 0.1$	$\lvert S\rvert/\lvert V\rvert(\%)$	1.50	0.98	0.13	0.13
	Density	9.70	23.51	20.83	74.39
	Time(ms)	5.65	6.79	66.23	66.11
$\epsilon = 0.5$	$\lvert S\rvert/\lvert V\rvert(\%)$	7.56	2.07	0.09	0.11
	Density	6.31	17.36	17.53	73.26
	Time(ms)	4.43	6.21	67.65	66.25

Table 5: Performance on synthetic datasets

Incremental Case

Evolution of the Densest Subgraph

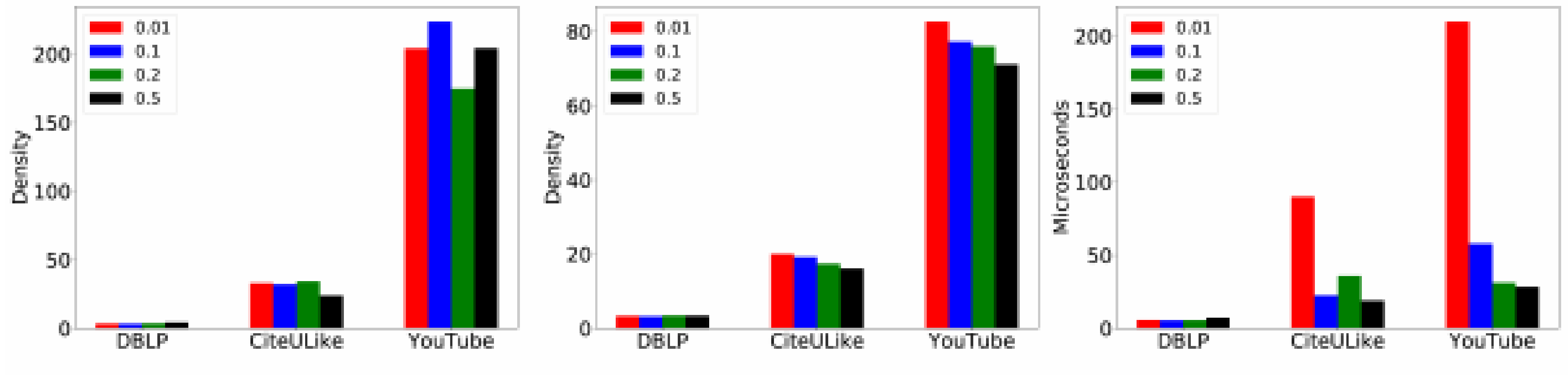


(a) DBLP (b) CiteULike (c) YouTube

Figure 2: Evolution of densest subgraph: insertion only.

Incremental Case

Efficiency Accuracy Trade-offs



(a) max. density (b) avg. density (c) avg. update time

Figure 3: Effect of ϵ in the incremental case.

Fully Dynamic Case

Improved Maintenance on Normal Graphs

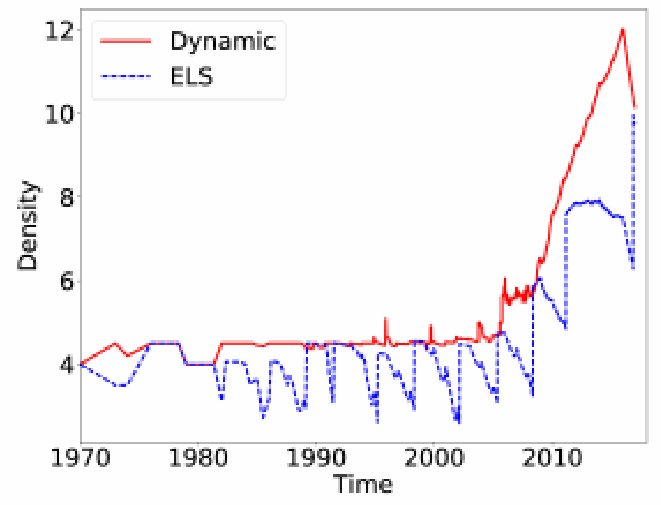
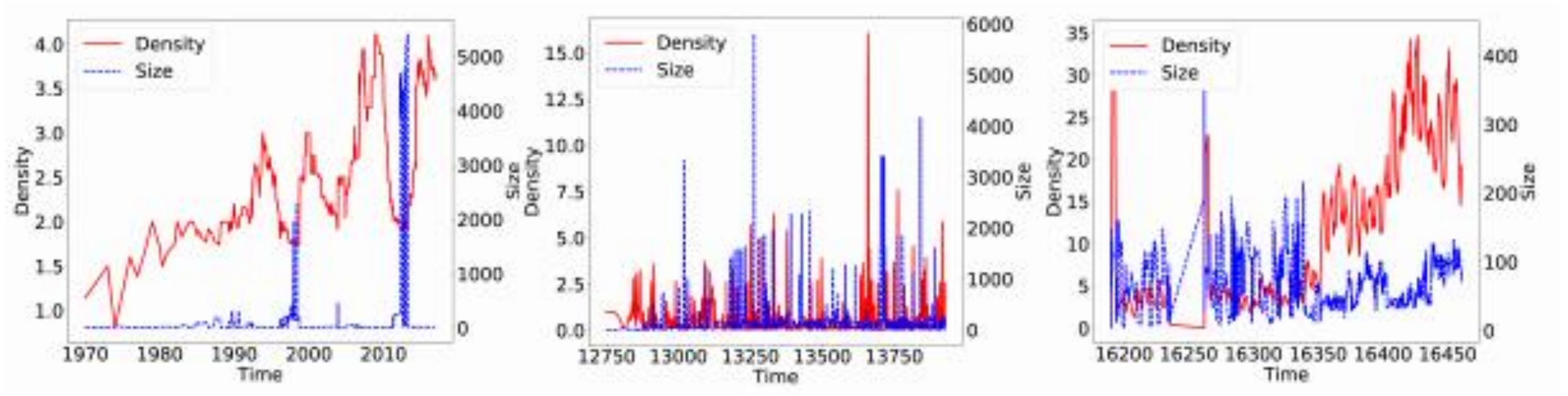


Figure 4: Evolution of the densest subgraph: ours vs ELS

Fully Dynamic Case

Evolution of the Densest Sub-hypergraph

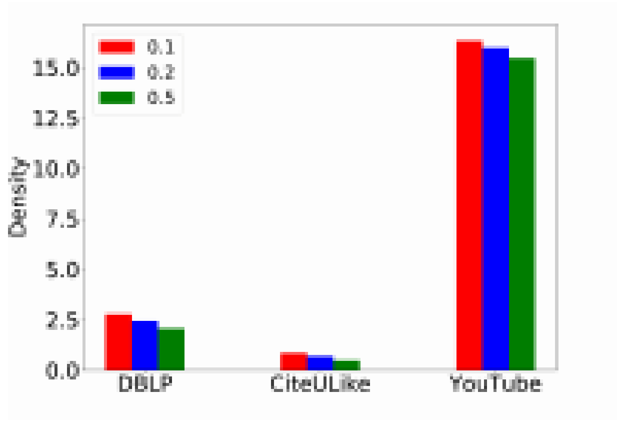


(a) DBLP (b) CiteULike (c) YouTube

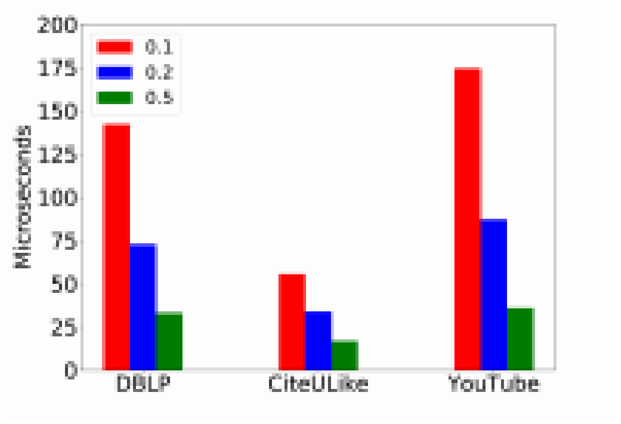
Figure 5: Evolution of densest subgraph: fully Dynamic.

Fully Dynamic Case

Efficiency Accuracy Trade-offs



(a) avg. density



(b) avg. update time

Figure 6: Trade-off between the average update time (in microseconds) and the density of the subgraph.

谢 谢 大 家 ！