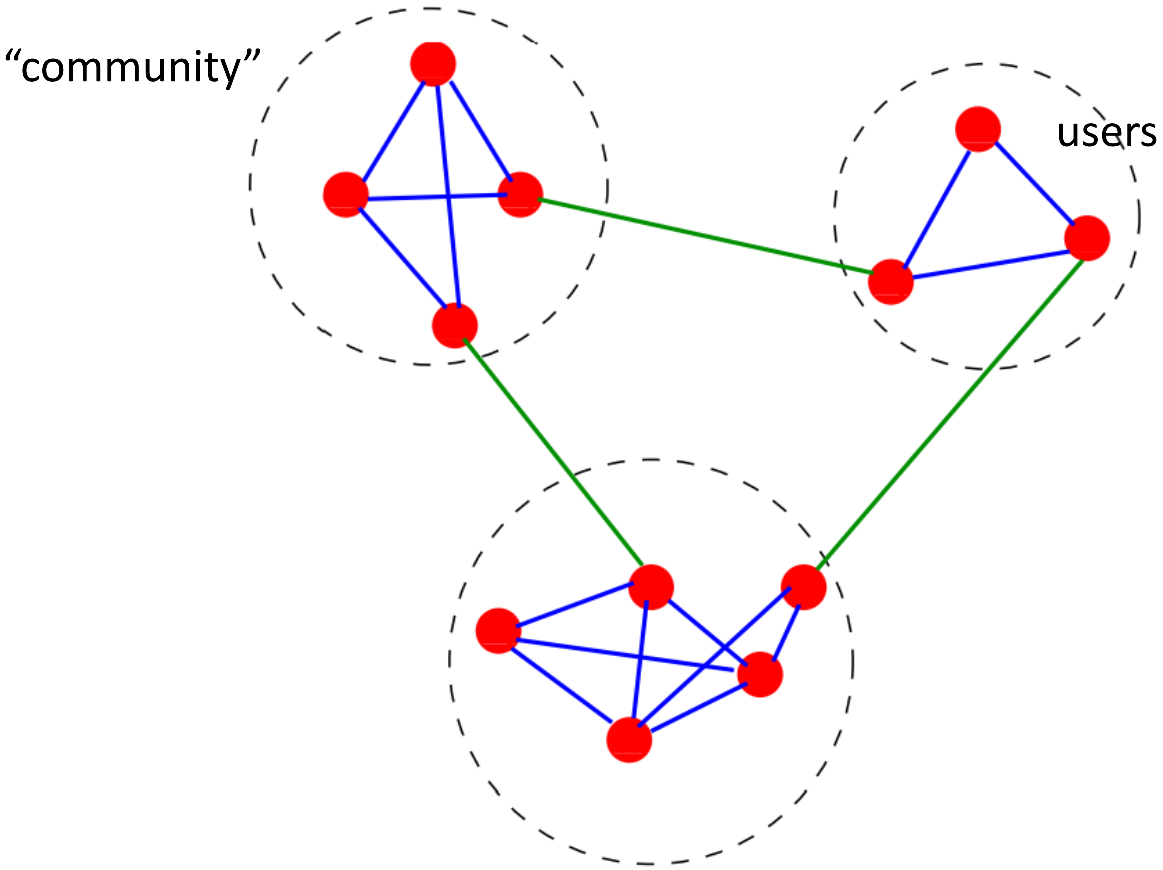
Detecting Communities under Differential Privacy

Hiep H. Nguyen, Abdessamad Imine, and Michael Rusinowitch

2016

Presenter: Zifeng Wang (zifengw2)

Community detection (CD)



Modularity:

# of clusters

edges joining the community c

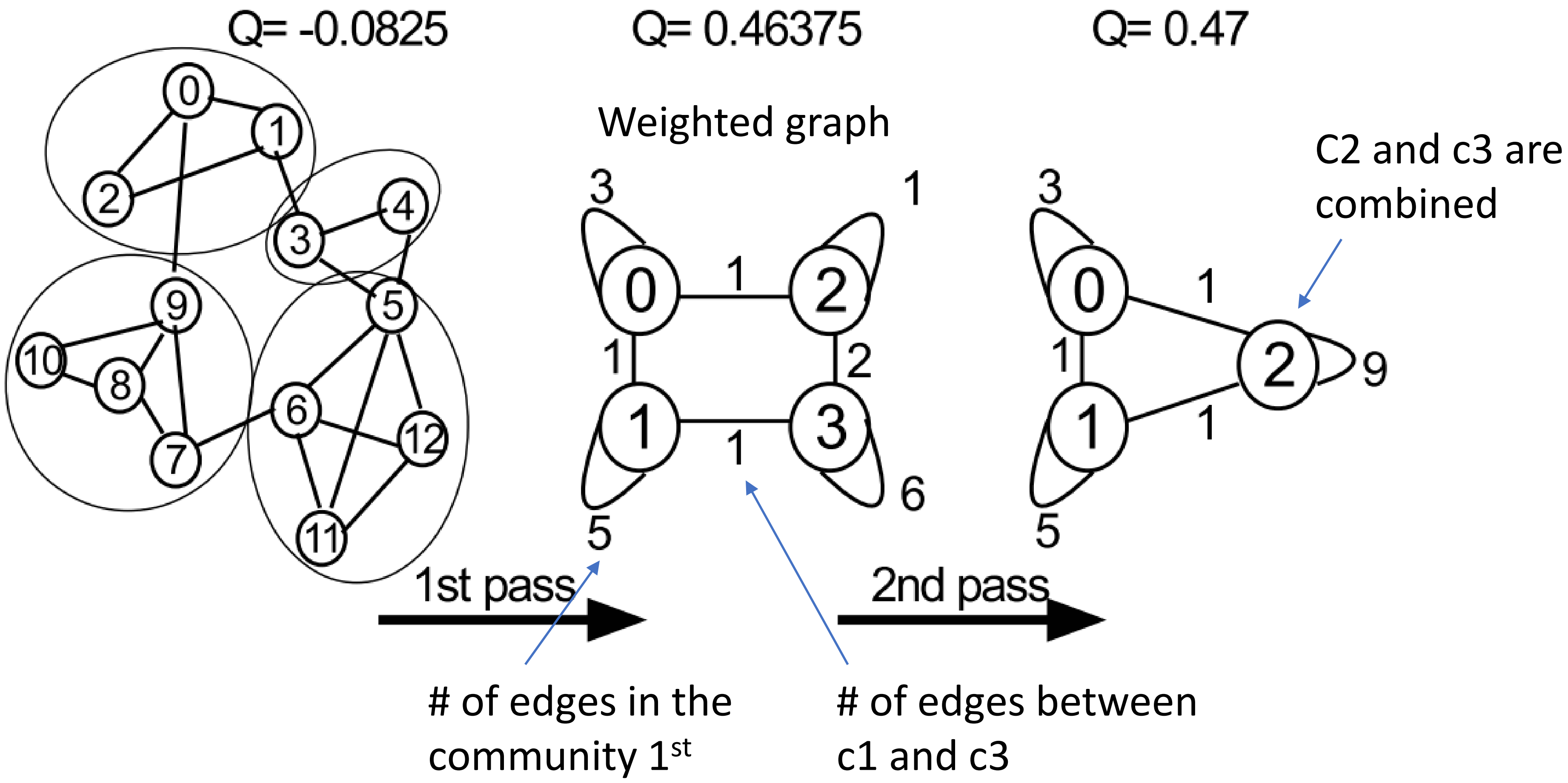
$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right]$$

# of edges in the whole graph

Sum of degree of nodes in c

Fortunato, S. (2010). Community detection in graphs. Physics reports, 486(3-5), 75-174.

Louvain method for CD



V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008

Differential privacy

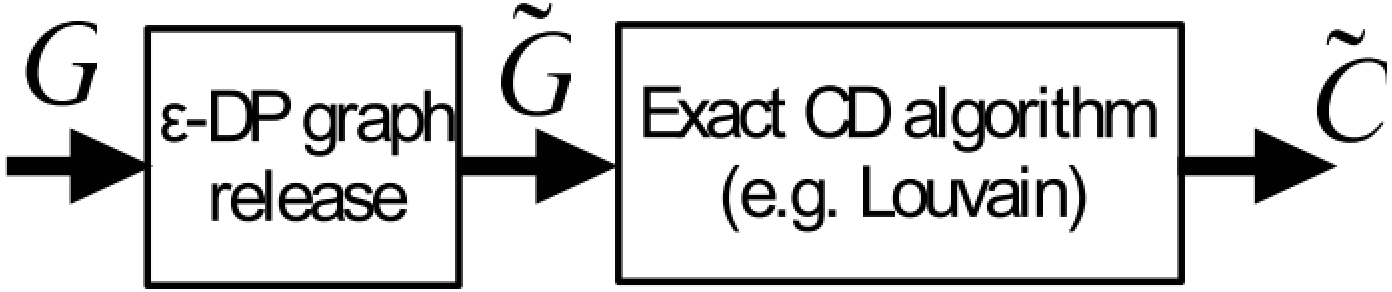
Definition 2.4 (Differential Privacy). A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \mathrm{Range}(\mathcal{M})$ and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta,$$

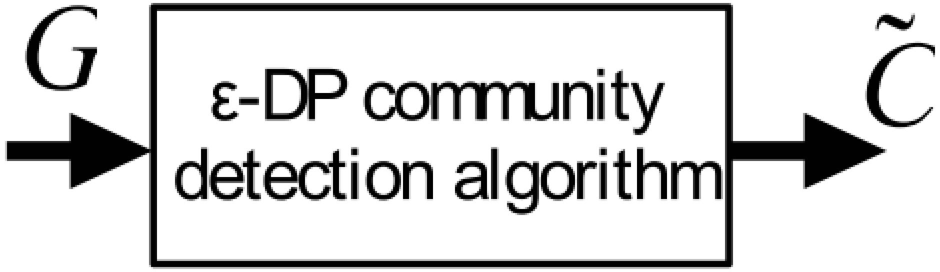
"privacy budget"

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci., 9(3-4), 211-407.

Two categories of DP CD



(a) Input perturbation



(b) Algorithm perturbation

ε-DP graph

Definition 3.1: A mechanism $\mathcal{A}$ is ϵ-differentially private if for any two neighboring graphs G_1 and G_2, and for any output $O \in Range(\mathcal{A})$,

$$Pr[\mathcal{A}(G_1) \in O] \leq e^{\epsilon} Pr[\mathcal{A}(G_2) \in O]$$

Neighboring graph: $G_1 = (V_1, E_1)$ $\quad$ $V_1 = V_2$

$G_2 = (V_2, E_2)$ $\quad$ $E_1 \subset E_2$ and $|E_2| = |E_1| + 1$

Input perturbations

- Laplace mechanism

Global sensitivity: $\Delta f = \max_{G_1,G_2} \|f(G_1) - f(G_2)\|_1$

Theorem 3.1: (Laplace mechanism [11]) For any function $f : G \rightarrow \mathbb{R}^d$, the mechanism $\mathcal{A}$

$$\mathcal{A}(G) = f(G) + \langle Lap_1(\frac{\Delta f}{\epsilon}), ..., Lap_d(\frac{\Delta f}{\epsilon})\rangle \qquad (2)$$

satisfies ϵ-differential privacy, where $Lap_i(\frac{\Delta f}{\epsilon})$ are i.i.d Laplace variables with scale parameter $\frac{\Delta f}{\epsilon}$. □

Laplace distribution: $Lap(\lambda) : p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. TCC, pages 265–284, 2006.

Input perturbations

- Geometric mechanism $Geom(\alpha) : Pr(\Delta = \delta|\alpha) = \frac{1-\alpha}{1+\alpha}\alpha^{|\delta|}$

For a function $f : G \rightarrow \mathbb{Z}^d$ The mechanism $\mathcal{A}(G) = f(G) + \{\Delta_i\}_{i=1^d}$

where $\Delta \sim Geom(\alpha)$ and $\alpha = \exp(-\varepsilon)$ satisfies ε-DP.

Δ is guaranteed to be integral

A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. SIAM Journal on Computing, 2012.

Algorithm1: LouvainDP

Basic idea: create a noisy weighted supergraph G_1 from G by grouping nodes with equal size k

Algorithm 1 LouvainDP(G, s)

Input: undirected graph G, group size k, privacy budget ϵ
Output: noisy partition $\tilde{C}$

Randomly initialize G_1 by node permutation and grouping

$G_1 \leftarrow \varnothing$, $n_1 = \lfloor \frac{|V|}{k} \rfloor - 1$, $V_1 \leftarrow \{0, 1, .., n_1\}$
$\epsilon_2 = 0.1$, $\epsilon_1 = \epsilon - \epsilon_2$, $\alpha = \exp(-\epsilon_1)$
get a random permutation V_p of V
compute the mapping $M : V_p \rightarrow V_1$
compute superedges of G_1: $E_1 = \{e_1(i, j)\}$ where $i, j \in V_1$

Noisy # of superedges

$m_1 = |E_1| + Lap(1/\epsilon_2)$, $m_0 = \frac{n_1(n_1+1)}{2}$
$\theta = \lceil \log_\alpha \frac{(1+\alpha)m_1}{m_0 - m_1} \rceil$
$s = (m_0 - m_1) \frac{\alpha^\theta}{1+\alpha}$

# of all possible edges in G_1

Algorithm1: LouvainDP

Basic idea: create a noisy weighted supergraph G_1 from G by grouping nodes with equal size k

$\theta = \lceil \log_\alpha \frac{(1+\alpha)m_1}{m_0 - m_1} \rceil$

$s = (m_0 - m_1)\frac{\alpha^\theta}{1+\alpha}$

Go through non-zero superedges, add geometric noise, decide if adding edge by threshold θ

for $e_1(i,j)$ in E_1 **do**
  $e_1(i,j) = e_1(i,j) + Geom(\alpha)$
  if $e_1(i,j) \geq \theta$ **then**
    add $e_1(i,j)$ to G_1

Go through zero superedges, draw integral weight and add edge if w is larger than zero

for s edges sampled uniformly at random $e_1(i,j) \notin E_1$ **do**
  draw w from the distribution $Pr[X \leq x] = 1 - \alpha^{x-\theta+1}$
  if $w > 0$ **then**
    add edge $e_1(i,j)$ with weight w to G_1

run Louvain method on G_1 to get $\tilde{C}_1$

compute $\tilde{C}$ from $\tilde{C}_1$ using the mapping M

return $\tilde{C}$

G. Cormode, C. Procopiuc, D. Srivastava, and T. T. Tran. Differentially private summaries for sparse data. In ICDT, pages 299–311. ACM, 2012.

Algorithm1: LouvainDP

Basic idea: create a noisy weighted supergraph G_1 from G by grouping nodes with equal size k

High-pass filtering (Cormode 2012) for private data sampling

$\theta = \lceil \log_\alpha \frac{(1+\alpha)m_1}{m_0 - m_1} \rceil$

$s = (m_0 - m_1)\frac{\alpha^\theta}{1+\alpha}$

for $e_1(i,j)$ in E_1 **do**

$\quad e_1(i,j) = e_1(i,j) + Geom(\alpha)$

1

formly at random $e_1(i,j) \notin E_1$ **do**

tribution $Pr[X \leq x] = 1 - \alpha^{x-\theta+1}$

with weight w to G_1

G_1 to get $\tilde{C}_1$

ng the mapping M

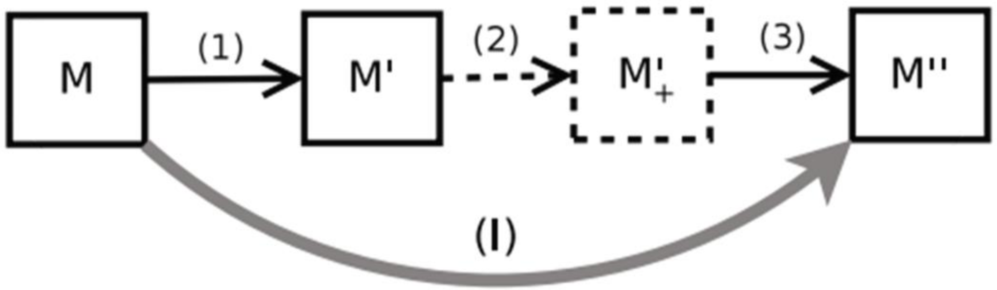
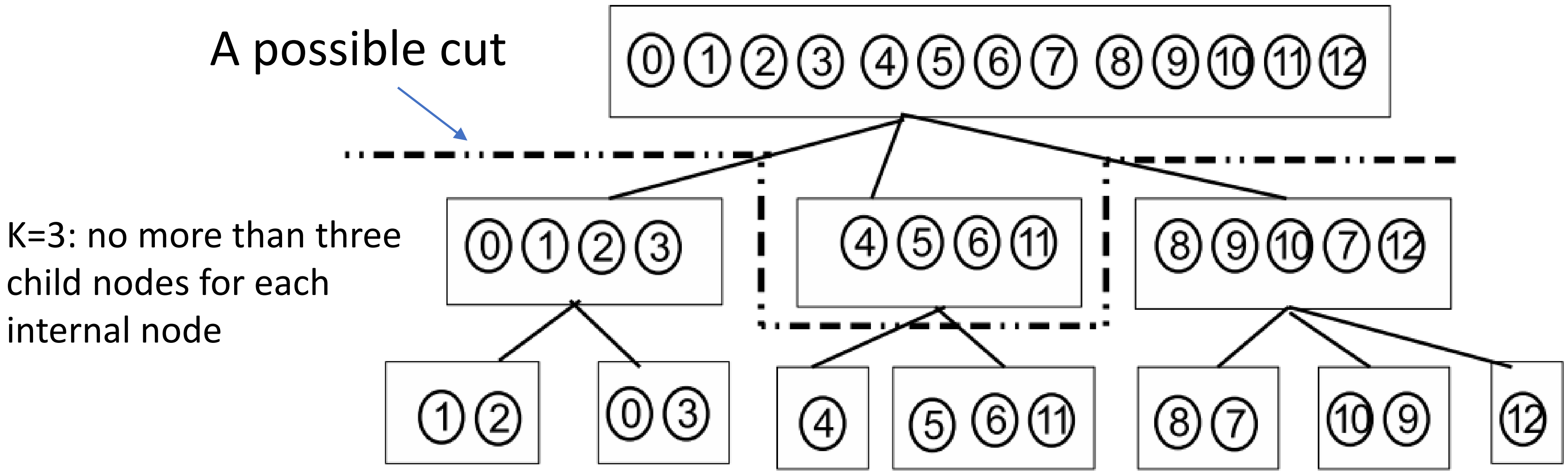


Figure 2: Anonymization process: (1) adding Geometric/Laplace noise, (2) filtering negative entries, (3) sampling, (I) shortcut to generate M'' directly from M.

G. Cormode, C. Procopiuc, D. Srivastava, and T. T. Tran. Differentially private summaries for sparse data. In ICDT, pages 299–311. ACM, 2012.

Algorithm perturbations

Basic idea: heuristically detect cohesive groups of nodes privately



Challenges:

- Efficiently find good split of nodes with high modularity under DP restrictions
- Merge small groups

Preliminary: Exponential mechanism

Theorem 3.2: (Exponential mechanism [21]) Given a score function $u : (G \times \mathcal{O}) \rightarrow \mathbb{R}$ for a graph G, the mechanism $\mathcal{A}$ that samples an output O with probability proportional to $\exp(\frac{\epsilon . u(G,O)}{2\Delta u})$ satisfies ϵ-differential privacy. □

Global sensitive $\Delta u = \max_{O,G_1,G_2} |u(G_1, O) - u(G_2, O)|$

Sampling output $O_i \sim \exp\left(\frac{\epsilon * u(G,O_i)}{2\Delta u}\right) / \sum_j \exp\left(\frac{\epsilon * u(G,O_j)}{2\Delta u}\right)$

F. McSherry and K. Talwar. Mechanism design via differential privacy. In FOCS, pages 94–103. IEEE, 2007.

Preliminary: Composability of DP

Theorem 3.3: (Sequential and parallel compositions [22]) Let each A_i provide ϵ_i-differential privacy. A sequence of $A_i(D)$ over the dataset D provides $\Sigma_{i=1}^{n} \epsilon_i$-differential privacy.

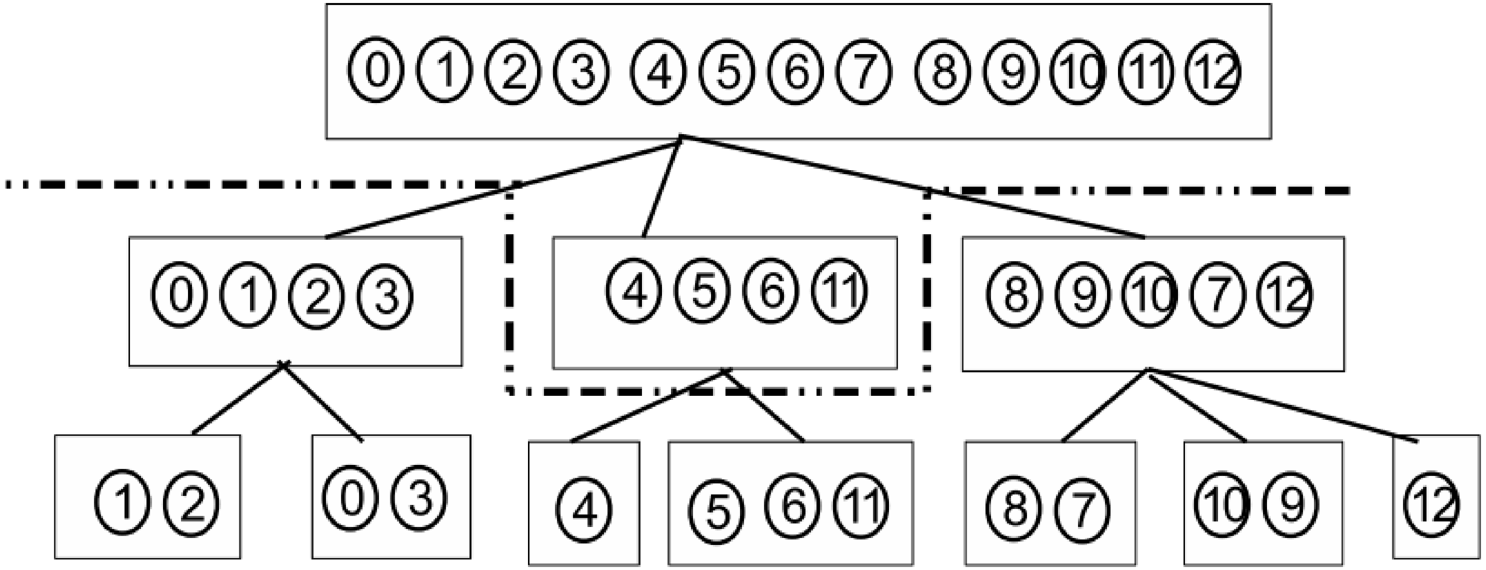
Let each A_i provide ϵ_i-differential privacy. Let D_i be arbitrary disjoint subsets of the dataset D. The sequence of $A_i(D_i)$ provides $\max_{i=1}^{n} \epsilon_i$-differential privacy. □

Sequential: same D, different A

Parallel: different D, different A

F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In SIGMOD, pages 19–30. ACM, 2009.

Algorithm: ModDivisive



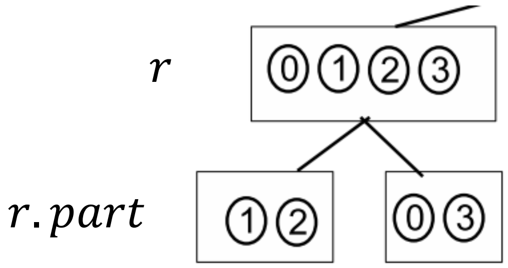
Stage 1st : DP sampling a tree of depth $maxL$, use budget ϵ_1

Stage 2nd : find best cut for all levels, use budget $\epsilon_m * maxL$

Algorithm: ModDivisive

Allocate the privacy budget for all levels in the tree: $\sum_i eA[i] = \epsilon_1$ cuz the sequential composability of DP

Go across each node in Q, try to make a node partition by **ModMCMC** algorithm. Then attach each partition as the child nodes



Algorithm 3 ModDivisive

Input: graph G, group size k, privacy budget ϵ, max level $maxL$, ratio λ, BestCut privacy at each level ϵ_m

Output: noisy partition $\tilde{C}$

compute the array $eA[0..maxL-1]$ s.t. $\sum_i eA[i] = \epsilon_1$, $eA[i] = eA[i+1] * \lambda$ where $\epsilon_1 = \epsilon - maxL.\epsilon_m$

initialize the root node with nodeset V

$root = \text{NodeSet}(G, V, k)$

$root.level = 0$

queue $Q \leftarrow root$

while Q is not empty **do**

    $r \leftarrow Q.dequeue()$

    if $r.level < maxL$ **then**

        $r.part = \text{ModMCMC}(G, r.S, k, eA[r.level])$

        for subset S_i in $r.part$ **do**

            $P_i = \text{NodeSet}(G, S_i, k)$

            $P_i.level = r.level + 1$

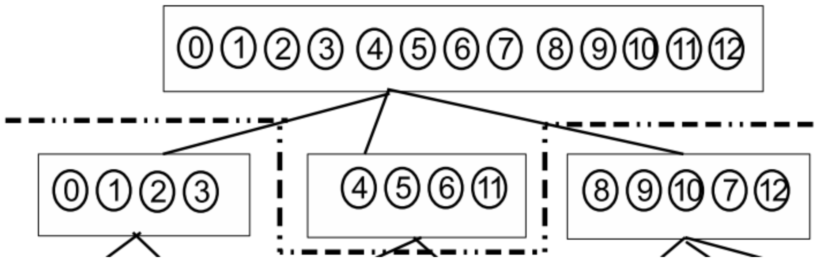
            $r.child_i \leftarrow P_i$

            $Q.enqueue(P_i)$

$\tilde{C} \leftarrow \text{BestCut}(root, \epsilon_m)$

return $\tilde{C}$

Algorithm: ModMCMC



$eA[r.level = 1] = \max_{i \in \{1,2,3\}} \epsilon_i$

if $r.level < maxL$ **then**
 $r.part = \text{ModMCMC}(G, r.S, k, \boxed{eA[r.level]})$
 for subset S_i in $r.part$ **do**
  $P_i = \text{NodeSet}(G, S_i, k)$
  $P_i.level = r.level + 1$
  $r.child_i \leftarrow P_i$
  $Q.enqueue(P_i)$

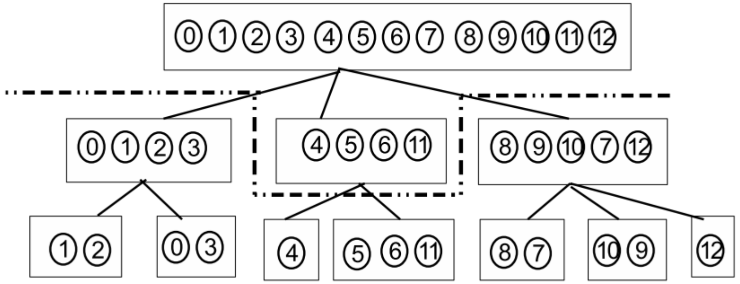
Algorithm 4 ModMCMC

Input: graph G, nodeset $r.S$, group size k, privacy budget ϵ_p
Output: sampled partition $r.part$

1: initialize $r.part$ with a random partition P_0 of k groups
2: **for** each step i in the Markov chain **do**
3: pick a neighboring partition P' of P_{i-1} by randomly selecting node $u \in r.S$ and moving u to another group.
4: accept the transition and set $P_i = P'$ with probability $min(1, \frac{\exp(\frac{\epsilon_p}{2\Delta Q} Q(P',G))}{\exp(\frac{\epsilon_p}{2\Delta Q} Q(P_{i-1},G))})$
5: // until equilibrium is reached
6: **return** a sampled partition $r.part = P_i$

Transform partition P_{i-1} to P_i with the specified probability by exponential mechanism

Algorithm: ModDivisive



Algorithm 3 ModDivisive

Input: graph G, group size k, privacy budget ϵ, max level $maxL$, ratio λ, BestCut privacy at each level ϵ_m
Output: noisy partition $\tilde{C}$

```
compute the array eA[0..maxL−1] s.t. ∑_i eA[i] = ε_1, eA[i] = eA[i+1] ∗ λ where ε_1 = ε − maxL.ε_m
initialize the root node with nodeset V
root = NodeSet(G, V, k)
root.level = 0
queue Q ← root
while Q is not empty do
    r ← Q.dequeue()
    if r.level < maxL then
        r.part = ModMCMC(G, r.S, k, eA[r.level])
        for subset S_i in r.part do
            P_i = NodeSet(G, S_i, k)
            P_i.level = r.level + 1
            r.child_i ← P_i
            Q.enqueue(P_i)
C̃ ← BestCut(root, ε_m)
return C̃
```

Get a cut on the final partition to reach the best modularity

Algorithm 5 BestCut

Input: undirected graph G, root node $root$, privacy budget at each level ϵ_m
Output: best cut C

```
stack S ← ∅, queue Q ← root
while Q is not empty do
    r ← Q.dequeue()
    S.push(r)
    for child node r_i in r.children do
        Q.enqueue(r_i)
dictionary sol ← ∅
while S is not empty do
    r ← S.pop(), r.mod_n = r.mod + Laplace(ΔQ/ε_m)
    if r is a leaf node then
        sol.put(r.id, (val = r.mod_n, self = True))
    else
        s_m = ∑_{r_i ∈ r.children} sol[r_i.id].mod_n
        if r.mod_n < s_m then
            sol.put(r.id, (val = s_m, self = False))
        else
            sol.put(r.id, (val = r.mod_n, self = True))
list C ← ∅, queue Q ← root
while Q is not empty do
    r ← Q.dequeue()
    if sol[r.id].self == True then
        C = C ∪ {r}
    else
        for child node r_i in r.children do
            Q.enqueue(r_i)
return C
```

Experiments

- Aims: verify if the DP CD gets good clustering quality in good efficiency

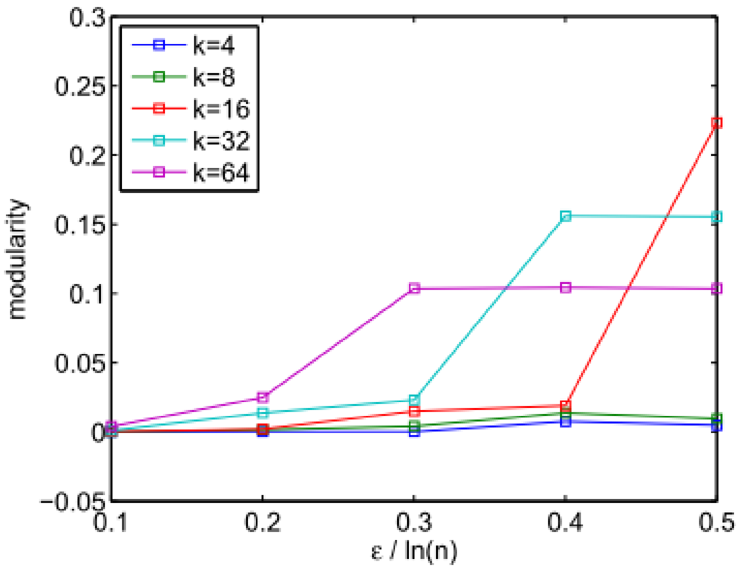
TABLE II: Characteristics of the test graphs

	Nodes	Edges	Com	Mod
as20graph	6,474	12,572	30	0.623
ca-AstroPh	17,903	196,972	37	0.624
amazon	334,863	925,872	257	0.926
dblp	317,080	1,049,866	375	0.818
youtube	1,134,890	2,987,624	13,485	0.710

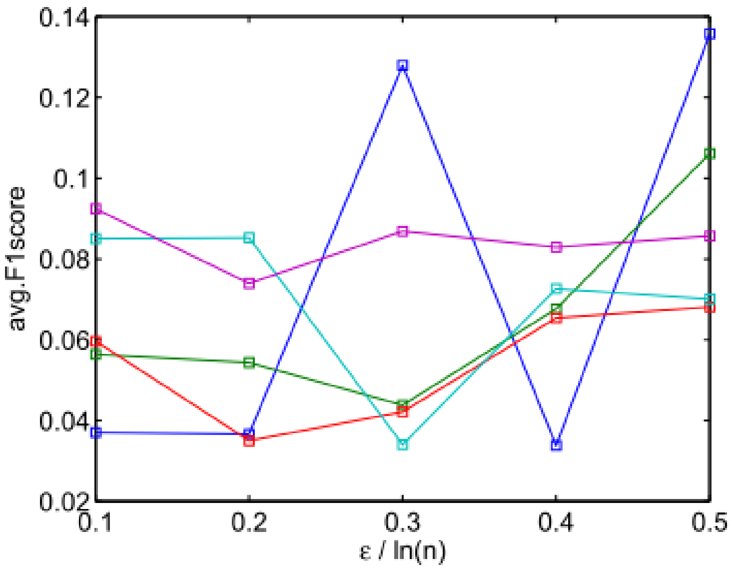
Obtained by Louvain method

Performance of LouvainDP

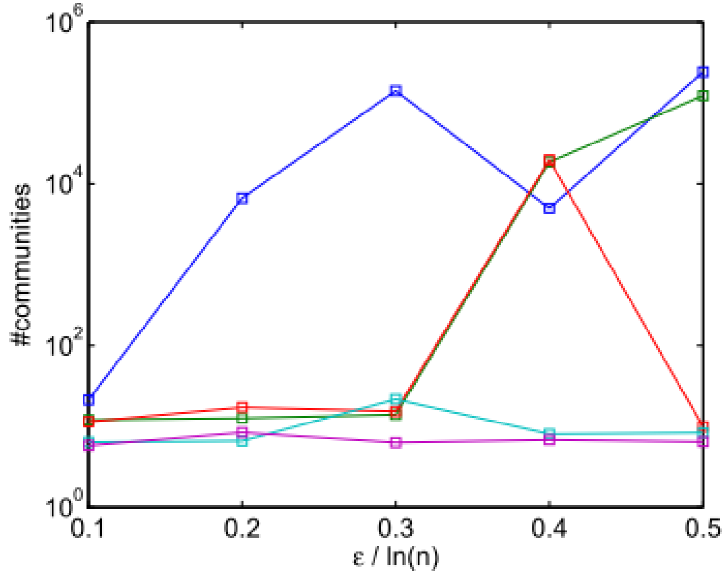
Group size k



(a)



(b)

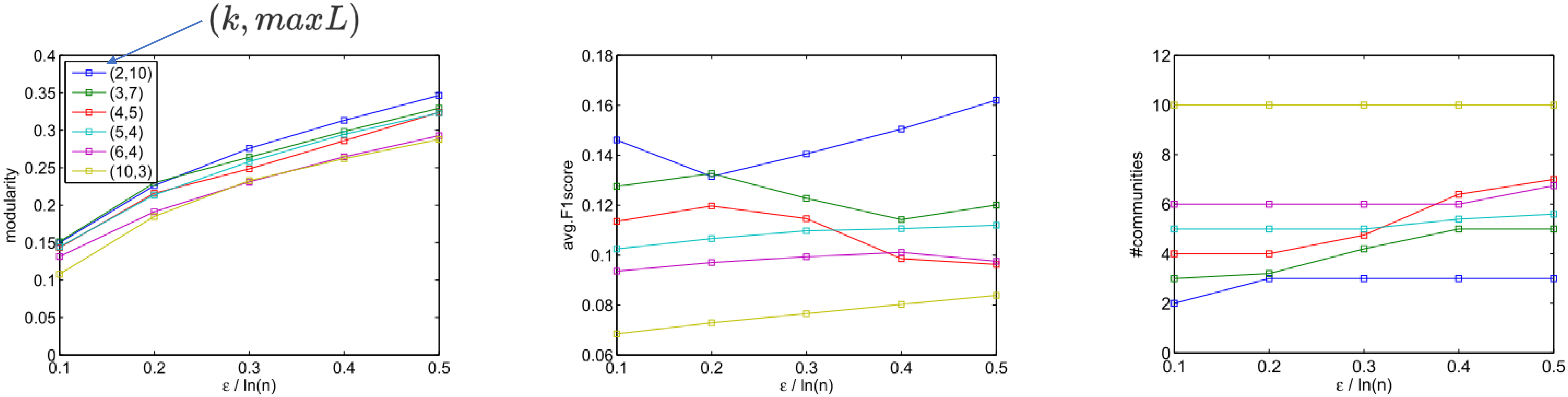


(c)

$$\varepsilon = \{0.1 \ln n, 0.2 \ln n, \ldots, 0.5 \ln n\}$$

# of nodes in the graph

Performance of ModDivisive



(a) (b) (c)

Fig. 6: ModDivisive on youtube with $\lambda = 2.0$, $K = 50$ $(0.5 \ln n = 7.0)$

Performance comparison

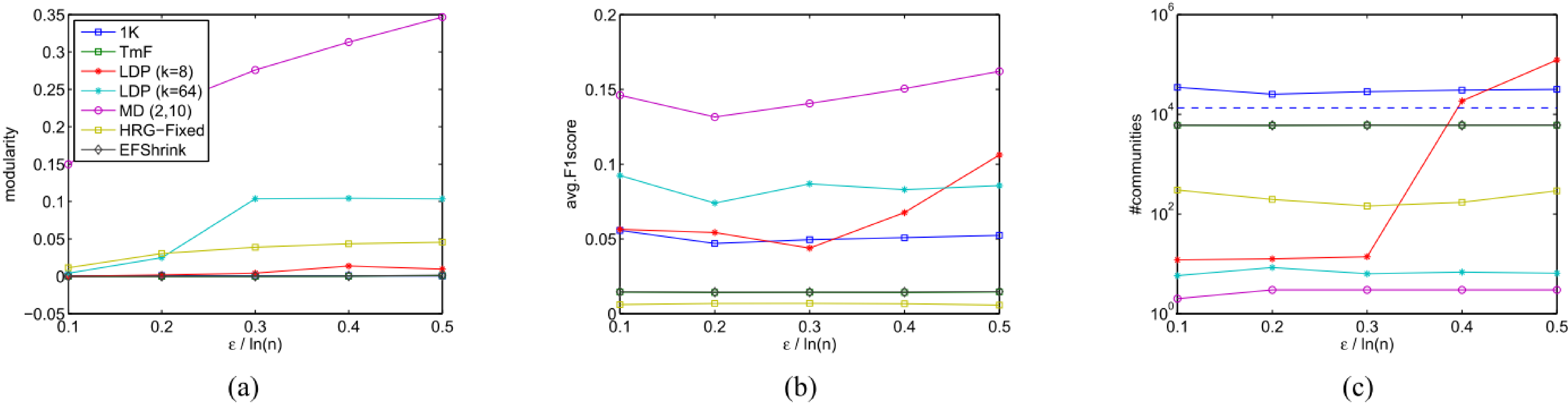


Fig. 13: Quality metrics and the number of communities (`youtube`) ($0.5 \ln n = 7.0$)