A THE CDS ALGORITHM FOR ALL k VALUES

Algorithm 4: CCAS-A

input : A graph G, two integers T, and ω.
output: Approximate CDS for all k values.

1 $\mathcal{S}_\omega \leftarrow$ an approximate ω-clique CDS of G;
2 $\gamma \leftarrow \max\{x \mid \binom{x}{\omega} \le \rho_\omega(\mathcal{S}_\omega) \cdot |\mathcal{S}_\omega|\}$;
3 $G_3 \leftarrow G$;
4 **foreach** $k \leftarrow 3, 4, \cdots, \omega - 1$ **do**
5 SCT $\leftarrow$ build_SCT(G_k); // build the SCT for G_k
6 $\mathcal{S}_k(G) \leftarrow$ run lines 3-19 in Algorithm 3 ;
7 $\mu_{k+1} \leftarrow \max\{\mu \mid \binom{\mu}{k} \le \frac{\binom{\gamma}{k+1}}{|\mathcal{S}_\omega|}\}$;
8 $V_i \leftarrow$ the first i-th vertices being removed in the first iteration;
9 $\lambda \leftarrow$ the largest i such that $\max_{v \in V_i} l^1(v) \le \binom{\mu_{k+1}}{k}$;
10 $G_{k+1} \leftarrow$ remove all vertices in V_λ from G_k;
11 **return** $\mathcal{S}_3(G), \mathcal{S}_4(G), \cdots, \mathcal{S}_\omega(G)$

Algorithm 4 illustrates the CCAS-A algorithm for finding CDS over all k values. It first computes an approximate CDS of ω-cliques, denoted as S_ω, where ω represents the maximum clique size (line 1). Next, based on Lemma 6.1, it calculates the γ value, which is used for the subsequent graph reduction process (line 2). CCAS-A process the k value from 3 to ω one by one (lines 4-10). Specifically, for each k, it invokes the CCAS to compute the approximate k-CDS (lines 5-6). Afterwards, it updates the μ value and further reduces the search space based on Theorem 6.2 (lines 7-10). Finally, it returns the approximate CDS for all k values (line 11).

B ADDITIONAL PROOFS OF LEMMAS AND THEOREMS

Lemma 4.2. Given a graph $G = (V, E)$, if $|\Psi_k(G)| \ge \binom{\mu}{k}$ where μ is an arbitrary integer no less than k, then $|\Psi_{k-1}(G)| \ge \binom{\mu}{k-1}$.

Proof. We prove Lemma 4.2 holds by proving its contrapositive, that is, given a graph $G = (V, E)$, if $|\Psi_{k-1}(G)| < \binom{\mu}{k-1}$, then $|\Psi_k(G)| < \binom{\mu}{k}$. Since $|\Psi_{k-1}(G)| < \binom{\mu}{k-1}$, the maximum clique size in G is less than μ. Suppose $\mathcal{R}(G)$ is the set of all maximal cliques in G. For each k-clique in G, it must be included in at least one maximal clique of G, so we have:

$$|\Psi_k(G)| = \left| \bigcup_{R \in \mathcal{R}(G)} \Psi_k(R) \right| \tag{19}$$

That is, $\forall \Phi \subseteq \mathcal{R}(G)$, we denote $g(\Phi) = \left| \bigcap_{R \in \Phi} \Psi_k(R) \right|$, then:

$$g(\Phi) = \left| \bigcap_{R \in \Phi} \Psi_k(R) \right| \tag{20}$$

$$= \left| \Psi_k\left(\bigcap_{R \in \Phi} R \right) \right| \tag{21}$$

$$= \binom{|\bigcap_{R \in \Phi} R|}{k} \tag{22}$$

This is because a k-clique is contained in all maximal cliques in $\mathcal{S}$, if and only if it is included in the subgraph formed by the intersection of those maximal cliques. Besides, the intersection of some cliques must either be a clique or an empty set, and the number of k-cliques in this subgraph is $\binom{|\bigcap_{R \in \mathcal{S}} R|}{k}$.

By applying the inclusion-exclusion principle on equation (19),

$$|\Psi_k(G)| = \left| \bigcup_{R \in \mathcal{R}(G)} \Psi_k(R) \right| \tag{23}$$

$$= \sum_{\mathcal{S} \subseteq \mathcal{R}(G)} (-1)^{|\mathcal{S}|} g(\mathcal{S}) \tag{24}$$

$$= \sum_{\mathcal{S} \subseteq \mathcal{R}(G)} (-1)^{|\mathcal{S}|} \binom{|\bigcap_{R \in \mathcal{S}} R|}{k} \tag{25}$$

Similarly,

$$|\Psi_{k-1}(G)| = \sum_{\mathcal{S} \subseteq \mathcal{R}(G)} (-1)^{|\mathcal{S}|} \binom{|\bigcap_{R \in \mathcal{S}} R|}{k-1} \tag{26}$$

Therefore, there exist a vector $\beta = \{\beta_{k-1}, \beta_k, \cdots, \beta_{\mu-1}\}$, such that,

$$|\Psi_{k-1}(G)| = \sum_{i=k-1}^{\mu-1} \beta_i \binom{i}{k-1} \tag{27}$$

$$|\Psi_k(G)| = \sum_{i=k-1}^{\mu-1} \beta_i \binom{i}{k} \tag{28}$$

Since, $\forall i \in [k-1, \mu-1]$, we have $\frac{\binom{i}{k}}{\binom{i}{k-1}} = \frac{i-k+1}{k} < \frac{\mu-k+1}{k}$, that is:

$$\binom{i}{k} < \frac{\mu - k + 1}{k} \cdot \binom{i}{k-1} \tag{29}$$

Therefore, we can conclude that:

$$\frac{|\Psi_k(G)|}{|\Psi_{k-1}(G)|} = \frac{\sum_{i=k-1}^{\mu-1} \beta_i \binom{i}{k}}{\sum_{i=k-1}^{\mu-1} \beta_i \binom{i}{k-1}} \tag{30}$$

$$< \frac{\sum_{i=k-1}^{\mu-1} \beta_i \frac{\mu-k+1}{k} \binom{i}{k-1}}{\sum_{i=k-1}^{\mu-1} \beta_i \binom{i}{k-1}} \tag{31}$$

$$< \frac{\mu - k + 1}{k} \tag{32}$$

Hence, we have $\frac{|\Psi_k(G)|}{|\Psi_{k-1}(G)|} < \frac{\mu-k+1}{k}$, therefore:

$$|\Psi_k(G)| < \frac{\mu - k + 1}{k} |\Psi_{k-1}(G)| \tag{33}$$

$$< \frac{\mu - k + 1}{k} \binom{\mu}{k-1} = \binom{\mu}{k} \tag{34}$$

□

Lemma 4.3. Given a graph $G = (V, E)$, $\forall v \in V$, if $|\Psi_k(v, G)| \ge \rho$, then for any $r \in \mathbb{N}$, with $3 \le r \le k$, we have $|\Psi_r(v, G)| \ge \binom{\mu}{r-1}$, where μ denotes the maximum integer such that $\binom{\mu}{k-1} \le \rho$.

Proof. We use mathematical induction to prove this lemma. (1) we first prove it holds when $r = k$:

$$\Psi_r(v, G) = \Psi_k(v, G) \ge \rho \ge \binom{\mu}{k-1} \ge \binom{\mu}{r-1} \tag{35}$$

(2) Next, we show it holds for arbitrary $r \le k$, and we suppose that the lemma holds for $r = s$, so $|\Psi_s(v, G)| \ge \binom{\mu}{s-1}$. Let $\mathcal{Y}_s =$

$\{u \mid u \in V \wedge \exists C \in \Psi_s(v,G), u \in C\}$, Let $G[\mathcal{Y}_s]$ denotes the induced subgraph of $\mathcal{Y}_s$. Since $|\Psi_s(v)| \geq \binom{\mu}{s-1}$, we have $|\Psi_{s-1}(G[\mathcal{Y}_s])| \geq \binom{\mu}{s-1}$. By Lemma 4.2, we have $|\Psi_{s-2}(G[\mathcal{Y}_s])| \geq \binom{\mu}{s-2}$.

On the other hand, since all vertices in $\mathcal{Y}_s$ are connected with v, so we have $|\Psi_{s-1}(v,G)| = |\Psi_{s-2}(G[\mathcal{Y}_s])| \geq \binom{\mu}{s-2}$. Hence, the lemma holds. □

Lemma 5.2. Given a graph G with n vertices and degeneracy of δ, CCAS costs $O(n \cdot 3^{\delta/3})$ space, and $O(n \cdot 3^{\delta/3} \cdot \delta^2)$ time for each iteration.

Proof. Consider iteratively removing all vertices in the graph. For each root-to-leaf path Γ, we need to process it each time a node it contains is removed. Therefore, each path is processed for $O(\delta)$ node, with each cost $O(\delta)$ time. Since there are $O(n \cdot 3^{\delta/3})$ nodes in the SCT. Therefore the time complexity for CCAS for one iteration is $O(n \cdot 3^{\delta/3})$. □

Lemma B.1. *Given any $t \geq 1$, denote $\bar{\alpha}$ by the vector that has a minimum inner product with the gradient of $Q_G(\alpha^{(t)})$*

$$\bar{\alpha} = \left[\bar{\alpha}^{C_1}, \quad \bar{\alpha}^{C_2}, \quad \cdots \quad, \bar{\alpha}^{C_{|\Psi_k(G)|}}\right] = \arg\min_{\beta \in D(G,\,k)} \langle \beta, \nabla Q_G(\alpha^{(t)}) \rangle.$$

We have $\widehat{\alpha}$ is an approximate linear minimizer, i.e.,

$$\langle \widehat{\alpha}, \nabla Q_G(\alpha^{(t)}) \rangle \leq \langle \bar{\alpha}, \nabla Q_G(\alpha^{(t)}) \rangle + \frac{1}{2}\gamma_t \theta \xi(Q_G),$$

where $\gamma_t = \frac{1}{t+1}$, $\theta = 2$,and $\xi(Q_G) = 2\Delta|\Psi_k(G)|$.

Proof. We first use $\nabla_C Q_G(\alpha)$ to denote the projection of $\nabla Q_G(\alpha)$ onto $\mathbb{R}^C$. By a straightforward calculation from proof of Lemma 4.5 [23], the (C,u)-coordinate of $\nabla Q_G(\alpha)$ is

$$2 \cdot \frac{l(u)}{t} = 2 \cdot \sum_{\bar{C} \in \Psi_k(G): u \in \bar{C}} \alpha_u^{\bar{C}} \tag{36}$$

It has been noted in the proof of Lemma 4.5 in [23] that one can consider each k-clique $C \in \Psi_k(G)$ independently and Therefore, for $t \geq 1$, we have:

$$< \widehat{\alpha}, \nabla Q_G(\alpha^{(t)}) > - < \bar{\alpha}, \nabla Q_G(\alpha^{(t)}) > \tag{37}$$

$$= \sum_{i=1}^{|\Psi_k(G)|} < \widehat{\alpha}^i, \nabla_{C^i} Q_G(\alpha^{(t)}) > - < \bar{\alpha}^i, \nabla_{C^i} Q_G(\alpha^{(t)}) > \tag{38}$$

$$= \sum_{i=1}^{|\Psi_k(G)|} < \widehat{\alpha}^i, \nabla_{C^i} Q_G(\alpha^{(t)}) > -\frac{2}{t} \min_{v \in C_i} l^{(t)}(v) \tag{39}$$

Consider the value of $\langle \widehat{\alpha}^i, \nabla_{C^i} Q_G(\alpha^{(t)}) \rangle$, it should be equal to $\frac{2l^{(t)}(v)}{t}$, where v is the first vertex removed from C_i in the t-th iteration. Therefore, we have $l^{(t)}(v) \leq \min_{v' \in C_i} l^{(t)}(v') + |\Psi_k(v', G)| \leq \min_{v' \in C_i} l^{(t)}(v') + \Delta$.

$$< \widehat{\alpha}, \nabla Q_G(\frac{\alpha^{(t)}}{t}) > - < \bar{\alpha}, \nabla Q_G(\frac{\alpha^{(t)}}{t}) > \tag{40}$$

$$= \sum_{i=1}^{|\Psi_k(G)|} < \widehat{\alpha}^i, \nabla_{C^i} Q_G(\alpha^{(t)}) > -\frac{2}{t} \min_{v \in C_i} l^{(t)}(v) \tag{41}$$

$$\leq \frac{2}{t} \sum_{i=1}^{|\Psi_k(G)|} \min_{v \in C_i} l^{(t)}(v) + \Delta - \min_{v \in C_i} l^{(t)}(v) \tag{42}$$

$$= \frac{2}{t}|\Psi_k(G)|\Delta \leq \frac{1}{2}\gamma_t \theta \xi(Q_G), \tag{43}$$

□

Theorem 5.4. CCAS reduces the overall running time at least by $O(\rho_k^*(G)\sqrt{k})$ over KCCA and by $O(\frac{(\frac{\delta}{2})^{k-2}}{\xi})$ over SuperGreedy++.

Proof. To obtain a $(1-\epsilon)$-approximation ratio solution, theoretically, KCCA, SuperGreedy++, and CCAS take $O\left(\frac{1}{\epsilon^2} \cdot \sqrt{k}\Delta|\Psi_k(G)| \cdot \xi \cdot \delta^2 \log\delta\right)$, $O\left(\frac{\Delta \log|\Psi_k(G)|}{\rho_k^*(G)\cdot\epsilon^2} \cdot km \cdot (\frac{\delta}{2})^{k-1}\right)$, and $O\left(\frac{\Delta \log|\Psi_k(G)|}{\rho_k^*(G)\cdot\epsilon^2} \cdot \xi \cdot \delta^3\right)$ time, respectively. Compared to KCCA, our method archives an improvement of $O(\frac{\sqrt{k}\Psi_k(G)\rho_k^*(G)\log\delta}{\log\Psi_k(G)\delta})$, which can be simplified to $O(\sqrt{k}\rho_k^*(G))$. This is because, $\frac{\Psi_k(G)\log\delta}{\log\Psi_k(G)\delta} > 1$, as shown by the function $f(x) = \frac{x}{\log x}$. Since $f'(x) = \frac{\log x - 1}{(\log x)^2}$ is positive for $x > 2$, $f(x)$ is an increasing function, and thus $\frac{\Psi_k(G)}{\log\Psi_k(G)} > \frac{\delta}{\log\delta}$.

Besides, compared to SuperGreedy++, our method achieves an improvement of $O(\frac{km\cdot(\frac{\delta}{2})^{k-1}}{\xi\delta^3})$, which simplifies to $O(\frac{k\cdot(\frac{\delta}{2})^{k-1}}{\xi\cdot\delta})$, since $\delta^2 < m$. This equation can be further simplified to $O(\frac{(\frac{\delta}{2})^{k-2}}{\xi})$. □

Lemma 6.1. Given a graph G, and an approximate k-clique CDS, $\mathcal{S}_k(G)$, for any $k' < k$, we have $\rho_{k'}^*(G) \geq \frac{\binom{\gamma}{k'}}{|\mathcal{S}_k(G)|}$, where γ is the maximum integer such that $\binom{\gamma}{k} \leq \rho_k(\mathcal{S}_k(G)) \cdot |\mathcal{S}_k(G)|$.

Proof. The number of k-cliques contained in approximate k-clique CDS $\mathcal{S}_k(G)$ should be $|\Psi_k(\mathcal{S}_k(G))| = \rho_k(\mathcal{S}_k(G)) \cdot |\mathcal{S}_k(G)|$. Then, we compute the largest integer γ such that $|\Psi_k(\mathcal{S}_k(G))| \geq \binom{\gamma}{k}$. By Lemma 4.2 and mathematical induction, we can conclude that $|\Psi_{k'}(\mathcal{S}_k(G))| \geq \binom{\mu}{k'}$ and $\rho_{k'}^*(G) \geq \rho_{k'}(\mathcal{S}_k(G)) \geq \frac{\binom{\gamma}{k'}}{|\mathcal{S}_k(G)|}$. □

Lemma 6.2 Given a graph G, an approximate k-clique CDS, $\mathcal{S}_k(G)$, for any integers $3 \leq x < k$, we use $\underline{\rho_x}(G)$ and $\underline{\rho_{x+1}}(G)$ to denote the lower bound of optimal densities obtained by the lemma 6.1, then we have $\mu_x \leq \mu_{x+1}$, where μ_x and μ_{x+1} are derived by Lemma 4.3, using $\underline{\rho_x}(G)$ and $\underline{\rho_{x+1}}(G)$.

Proof. Recall that μ_x is the maximum integer such that:

$$\binom{\mu_x}{x-1} \leq \frac{\binom{\gamma}{x}}{|\mathcal{S}_\omega(G)|}, \tag{44}$$

where γ is the maximum integer s.t. $\binom{\gamma}{\omega} \leq \rho_\omega(\mathcal{S}_\omega(G)) \cdot |\mathcal{S}_\omega(G)|$. Based on the formulation (44) we have:

$$\frac{\mu_x!}{(x-1)!(\mu_x - x + 1)!} \leq \frac{\gamma!}{x!(\gamma - x)!|\mathcal{S}_\omega(G)|} \tag{45}$$

$$\frac{\mu_x!}{(\mu_x - x + 1)!} \leq \frac{\gamma!}{x(\gamma - x)!|\mathcal{S}_\omega(G)|} \tag{46}$$

Similarly, μ_{x+1} is the maximum integer satisfy,

$$\frac{\mu_{x+1}!}{(\mu_{x+1} - x)!} \leq \frac{\gamma!}{(x+1)(\gamma - x - 1)!|\mathcal{S}_\omega(G)|} \tag{47}$$

Table 9: Additional two graphs.

Dataset	Category	$\|V\|$	$\|E\|$	δ
web-Google (WG)	Web	916,428	4,322,051	44
Wikipedia (WP)	Hyperlink	3,033,374	43,845,958	175

Therefore, to show $\mu_x \leq \mu_{x+1}$, we only need to prove the following formulation:

$$\frac{\mu_x!}{(\mu_x - x)!} \leq \frac{\gamma!}{(x+1)(\gamma - x - 1)!|\mathcal{S}_\omega(G)|} \tag{48}$$

Here, we can transfer the formulation (45) to (47) by multiplying $(\mu_x - x + 1)$ and $\frac{x(\gamma-x)}{x+1}$ for the left and right hand side of it, respectively. That is to say , if we can prove:

$$\mu_x - x + 1 \leq \frac{x(\gamma - x)}{x+1} \tag{49}$$

$$\mu_x(x+1) + 1 - x^2 \leq x\gamma - x^2 \tag{50}$$

$$\mu_x \leq \frac{x\gamma - 1}{x+1} \tag{51}$$

$$\mu_x \leq \gamma - \frac{\gamma + 1}{x+1} \tag{52}$$

then the formulation (48) can also be proved.

Since $\rho_\omega(\mathcal{S}_\omega(G)) \cdot |\mathcal{S}_\omega(G)| \geq 1$, we have $\gamma \geq \omega > x$. Therefore, we can rewrite γ in terms of $p \cdot (x+1) + b$, where $p \geq 1$ and $0 \leq b \leq k$.

$$\mu_x \leq p(x+1) + b - \frac{px + p + b + 1}{x+1} \tag{53}$$

$$\mu_x \leq px + b - \frac{b+1}{x+1} \tag{54}$$

Since μ_x is an integer, we only need to prove:

$$\mu_x \leq px + b - 1 \tag{55}$$

We prove this by contradiction, where we aim to show that if $\mu_x > px + b - 1$, it must violate formulation (46). Since the left-hand side of (46) is monotonically non-decreasing with μ_x, we only need to prove the case for $\mu_x = px + b$, i.e.,:

$$\frac{(px+b)!}{(px+b-x+1)!} > \frac{(p\cdot(x+1)+b)!}{x(p\cdot(x+1)+b-x)!|\mathcal{S}_\omega(G)|} \tag{56}$$

Recall that $\rho_\omega(\mathcal{S}_\omega(G)) \cdot |\mathcal{S}_\omega(G)| = |\Psi_\omega(\mathcal{S}_\omega(G))| \geq \binom{\gamma}{\omega}$, so we have $|\mathcal{S}_\omega(G)| \geq \gamma = px + b$. Hence a stricter inequality can be built:

$$\frac{(px+b)!}{(px+b-x+1)!} > \frac{(p\cdot(x+1)+b-1)!}{x(p\cdot(x+1)+b-x)!} \tag{57}$$

$$x \cdot \frac{(px+b)!}{(px+b-x+1)!} > \frac{(p\cdot(x+1)+b-1)!}{(p\cdot(x+1)+b-x)!} \tag{58}$$

$$x \cdot \frac{(p\cdot(x+1)+b-x)!}{(px+b-x+1)!} > \frac{(p\cdot(x+1)+b-1)!}{(px+b)!} \tag{59}$$

$$x \cdot \prod_{i=px+b-x+2}^{px+b-x+p} i > \prod_{i=px+b+1}^{px+b+p-1} i \tag{60}$$

$$x \cdot \prod_{i=px+b-x+2}^{px+b-x+p} i > \prod_{i=px+b-x+2}^{px+b-x+p} (i+x-1) \tag{61}$$

$$x > \prod_{i=px+b-x+2}^{px+b-x+p} \frac{(i+x-1)}{i} \tag{62}$$

If $a = 1$, the right-hand side of formulation (62) is 1, which is clearly smaller than x. For the case of $p \geq 2$, we complete our proof by bounding on the right-hand side of formulation (62).

$$\prod_{i=px+b-x+2}^{px+b-x+p} \frac{(i+x-1)}{i} \leq \prod_{i=px+b-x+2}^{px+b-x+p} \frac{px+b+1}{px+b-x+2} \tag{63}$$

$$= \left(\frac{px+b+1}{px+b-x+2}\right)^{p-1} \tag{64}$$

$$\leq \left(\frac{px}{(p-1)\cdot x}\right)^{p-1} \tag{65}$$

$$= \left(1 + \frac{1}{p-1}\right)^{p-1} \tag{66}$$

$$\leq \lim_{p\to\infty} \left(1 + \frac{1}{p-1}\right)^{p-1} = e \tag{67}$$

Hence, we finish our proof since $x \geq 3 > e$. □

C ADDITIONAL EXPERIMENTS

C.1 Additional Datasets.

We present the statistics of the additional two graphs on Table 9 from the different domains. They are available on the Stanford Network Analysis Platform, and Laboratory of Web Algorithmics.

C.2 The k-clique densities

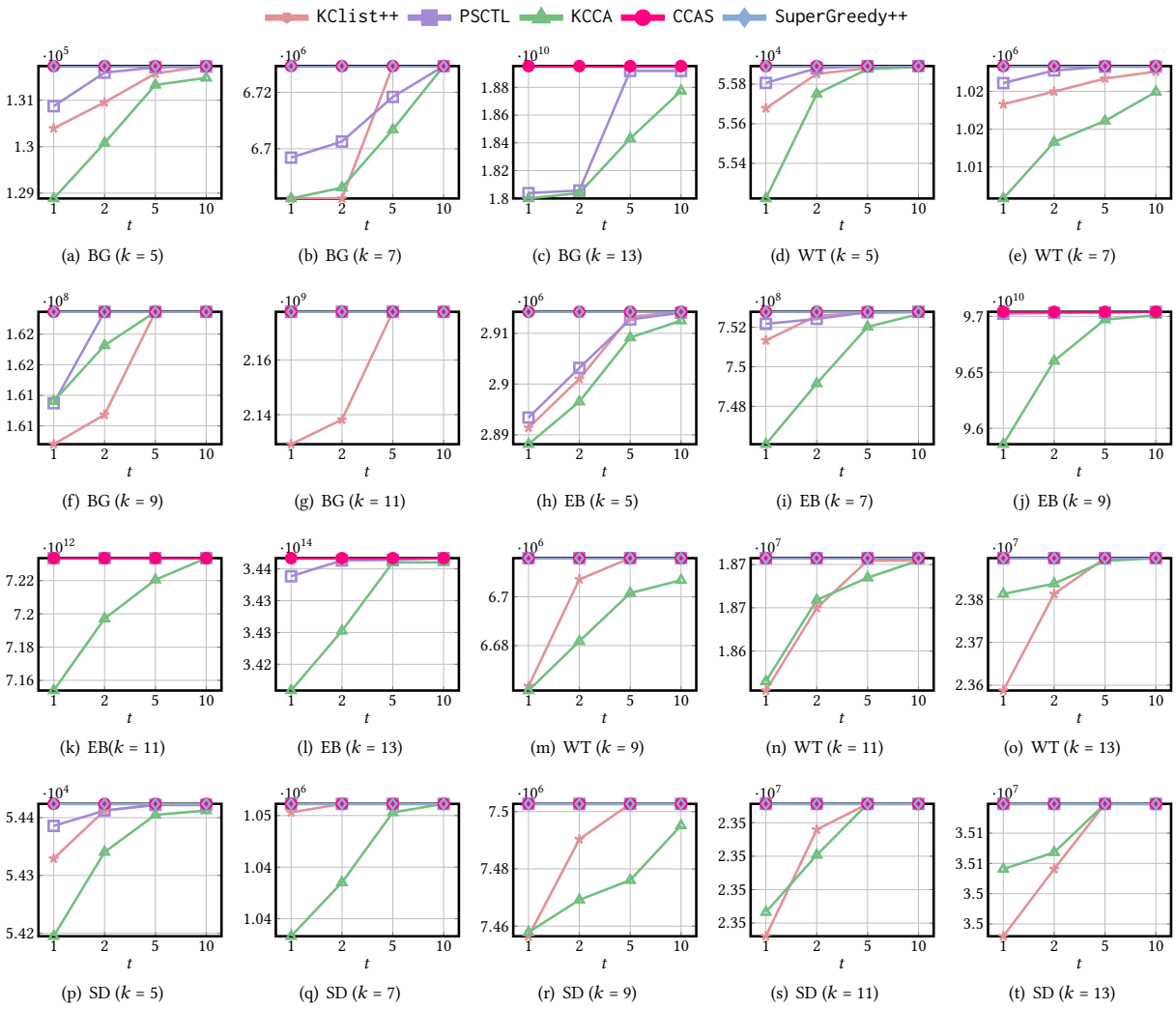
In Figure 12, we report the actual k-clique-densities of the approximation solutions returned by all the CDS algorithms on BG, EB, WT, and SD datasets. We observe that `CCAS` and `SuperGreedy++` always perform better than the `Frank-Wolfe`-based algorithms, and typically, `CCAS` and `SuperGreedy++` only require one iteration to achieve a solution with the same density that other algorithms obtain after ten iterations. In addition, both `KClist++` and `PSCTL` yield comparable performance, and they slightly outperform `KCCA`, since they can enable a more balanced weight distribution among vertices, making them converge faster.

C.3 Effectiveness of the graph reduction

In this experiment, we evaluate the effectiveness of our graph reduction technique, by reporting the number of vertices in the $(k-1)$-core, CDS, and `CCAS` for $k = 7$. We present results only for $k = 7$, as other k values yield similar results. In addition, more results for different k values can be found in our supplementary material. As shown in Table 10, we observe that: (1) the number of vertices in the CDS is remarkably smaller, usually no more than a few thousand; (2) across nearly all datasets, the number of vertices after `HCGR` closely matches the number of vertices in the CDS; (3) the number of vertices in the 6-core remains very high, being up to $100,000\times$ larger than the number of vertices remaining after `HCGR`.

C.4 Case studies

Here, we would conduct the case studies on two real datasets, namely S-DBLP and Yeast. S-DBLP ($|V|$=478, $|E|$=1,086) is a subgraph of the DBLP dataset, which is about the co-authorship network of authors who published at least two DB/DM papers between 2013 and 2015. The triangle densest subgraph is depicted in Figure 13(a). In a triangle (3-clique), every pair of vertices is connected, so the CDS tends to be a near-clique [28, 33, 74]. The researchers



(a) BG (k = 5) (b) BG (k = 7) (c) BG (k = 13) (d) WT (k = 5) (e) WT (k = 7)

(f) BG (k = 9) (g) BG (k = 11) (h) EB (k = 5) (i) EB (k = 7) (j) EB (k = 9)

(k) EB(k = 11) (l) EB (k = 13) (m) WT (k = 9) (n) WT (k = 11) (o) WT (k = 13)

(p) SD (k = 5) (q) SD (k = 7) (r) SD (k = 9) (s) SD (k = 11) (t) SD (k = 13)

Figure 12: The k-clique densities of CDS obtained by SuperGreedy++, KClist++, KCCA, PSCTL, and CCAS.

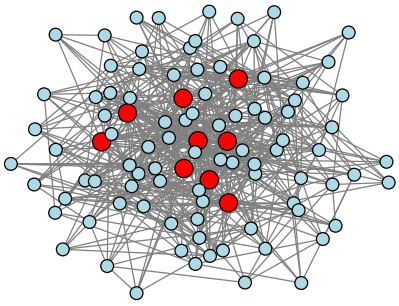
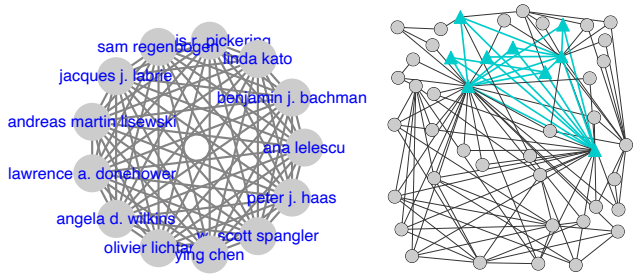


Figure 14: A case study to illustrate the HCGR.



(a) triangle (3-clique) (b) 4-clique (cyan)

Figure 13: The densest subgraphs found in small DBLP and Yeast networks.

involved in this CDS possess a close collaboration relationship: any two researchers have published papers together. In addition, Figure 13(b) shows the 4-clique densest subgraph in the yeast PPI network ($|V|$=1,116, $|E|$=2,148). This CDS shows a protein-subnetwork with the "subcellular localization", "cellular transport", and "protein synthesis" functional classes, which are all 4-cliques. Hence, tasks such as analyzing the conservation and evolution of cellular components [28] can then be performed on this subnetwork.

Table 10: # vertices in the ($k-1$)-core, CDS, and CCAS (k=7).

Datasets	# vertices in 6-core	$\lvert V(\mathcal{D}_k(G))\rvert$	CCAS
BG	1,306	116	268 (43.4 %)
EB	487	231	444 (52.0 %)
WT	2,619	127	551 (23.0 %)
SD	21,686	117	575 (20.3 %)
DP	68,070	114	216 (52.8 %)
HT	21,239	563	563 (100 %)
WG	367,361	116	256 (45.3 %)
WP	1,466,413	178	1,668 (10.7 %)
HW	1,069,126	2209	2209 (100 %)
ZB	3,391,200	268	325 (82.4 %)
UK	11,404,146	944	1,665 (56.7 %)
AC	16,063,729	3,250	3,250 (100 %)
IT	27,201,499	4,279	4,279 (100 %)
FS	36,821,624	141	141 (100 %)

Besides, we also present a case study to illustrate our HCGR algorithm, as shown in Figure 14. The k-core-based method does not remove any vertices, whereas HCGR eliminates the blue vertices while retaining the red ones. This demonstrates that HCGR effectively removes nearly all vertices that are not part of the CDS.

C.5 Effect of ϵ

We evaluate the effect of ϵ using five datasets from different domains, where each domain has a dataset, and the values of ϵ are set to 1, 0.1, 0.05, and 0.01, respectively. The experimental results are reported in Table 11, which clearly shows that CCAS outperforms the other algorithms on all datasets. For around half of the datasets, CCAS is over three orders of magnitude faster than its competitors, those results are similar to the results for 0.1 and 0.01 cases, as shown in Table 6.

C.6 Efficiency of finding the CDS's for all k.

We compare the efficiency of our algorithm CCAS-A and others for finding the CDS's for all k values. Specifically, for each graph, we record the running time of the three algorithms PSCTL, KCCA, and CCAS-A as they process the k values from 3 to 25% $\cdot\omega$, 50% $\cdot\omega$, 75% $\cdot\omega$, and 100% $\cdot\omega$ across six datasets in Table 12. We present the results for ϵ from 1 to 0.01. The conclusions are similar to Experiment 3 in Section 7.2. Besides, to our best knowledge, CCAS is the first algorithm that can produce solutions of all k values with an approximation ratio of 0.99 for graphs with billions of edges in one hour.

C.7 Scalability test.

We conduct additional experiments to evaluate the effect of graph size on the performance of our CDS algorithm, using k=7 and T=10. To achieve this, we control the graph size from two perspectives:

- For each graph, we create five induced subgraphs by randomly selecting 20%, 40%, 60%, 80%, and 100% of the vertices. We run CCAS on these subgraphs and report the average runtime and the resulting degeneracy value δ in Figure 15.
- To isolate the impact of the number of edges while fixing the degeneracy value of the graph, we construct another set of five subgraphs by selecting the top 20% to 100% of edges based on their core numbers, where the core number of an edge is defined as the smaller core number of its two endpoints. This ensures that all subgraphs are sampled from the high-density region of the graph, preserving the core structure while gradually increasing the number of edges. We evaluate CCAS on these subgraphs and report the average runtime and resulting edge counts in Figure 16.

Based on the above results, we make the following observations and analysis: (1) The running time of CCAS on all datasets is proportional to the value of δ, which is aligned with our time complexity analysis of CCAS. (2) When the value of δ is fixed, the number of edges does not highly affect the efficiency of CCAS.

C.8 Comprehensive Comparison of the CDS algorithms

We provide a comprehensive comparison of the CDS algorithms in terms of accuracy, running time across 10 datasets in Table 16, where $k = 5 \sim 30$ and $T = 10$. If an algorithm cannot finish within 100 hours, we mark its running time as "INF" and its memory usage as "—". If an algorithm requires more than 1TB of memory, we mark its running time as "—" and its memory usage as "OOM". In addition, if an algorithm encounters either "INF" or "OOM", we mark its accuracy as "—". We can make the following observations and analysis: (1) All methods achieve comparable performance in terms of accuracy, which implies that T=10 is enough for those approximation algorithms to obtain the near-optimal solutions. (2) Our method, CCAS, is significantly faster than all competitors on all datasets and different k values. For graphs with high degeneracy values (e.g., AC and IT), CCAS is the only method that successfully obtains a near-optimal solution, whereas other algorithms either incur "INF" or "OOM" issues. In addition, all clique-counting-based algorithms are significantly faster than the clique enumeration-based method, SuperGreedy++, since it needs to enumerate all cliques, which is extremely time-consuming. (3) SuperGreedy++ always consumes less memory than other methods due to its lightweight space complexity (i.e., $O(m)$), where m denotes the number of edges in the graph. Among the three clique-counting-based algorithms, although they share the same theoretical space complexity, CCAS typically uses less memory in practice, thanks to our graph reduction technique, which effectively reduces memory usage. In a few small datasets, CCAS may consume slightly more memory than PSCTL and KCCA, which is mainly due to implementation differences. Specifically, CCAS maintains additional auxiliary structures for iterative vertex weight updates.

C.9 Effect of r in graph reduction algorithm.

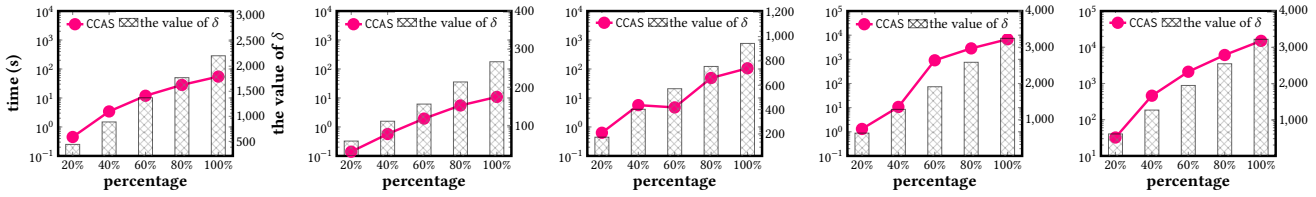
In HCGR, a larger r intuitively increases the time cost of clique counting, while enabling more vertices to be pruned. As shown in Figure 17, we evaluate the effect of r by varying its value from 3 to 6 across seven datasets, including four with more than one billion edges, where $r = 0$ corresponds to locating the graph into its $(k-1)$-core without using HCGR. We can see that: (1) Using HCGR significantly reduces the running time compared to not using it, achieving up to a 1,000× speedup on the UK dataset, while also pruning more vertices from the graph. (2) In HCGR, as the r increases, there is no huge decrease in total running time and the number of remaining vertices. In contrast, a larger r may increase the overall running time of the CDS algorithm, as r-clique counting in HCGR becomes more expensive to compute. Based on these results, we observe that setting r=3 achieves a good trade-off: it effectively removes most vertices that are not part of the CDS while remaining computationally efficient, as 3-clique counting can be done in $O(m^{1.5})$ time. This suggests that setting r=3 is recommended for the CDS problem.

Table 11: Effect of ϵ and k. (Processing time (in seconds); Best performers are highlighted in bold.)

Dataset	Method	$k=7$				$k=11$				$k=15$				$k=19$			
		1	0.1	0.05	0.01	1	0.1	0.05	0.01	1	0.1	0.05	0.01	1	0.1	0.05	0.01
HT	PSCTL	79,752	79,752	155,885	268,701	–	–	–	–	–	–	–	–	–	–	–	–
	KCCA	76,191	76,191	150,316	261,231	–	–	–	–	–	–	–	–	–	–	–	–
	CCAS	**8.6**	**8.6**	**8.6**	**8.6**	**8.5**	**8.5**	**8.5**	**8.5**	**8.5**	**8.5**	**8.5**	**8.5**	**8.4**	**8.4**	**8.4**	**8.4**
WG	PSCTL	2.8	2.8	6.9	18.3	1.4	3.2	3.2	4.8	1.3	1.3	1.3	2.5	0.9	0.9	0.9	2.0
	KCCA	7.0	8.2	11.5	16.0	8.3	8.3	8.3	8.6	6.9	6.9	6.9	6.9	6.1	6.1	6.1	6.1
	CCAS	0.2	0.2	0.3	0.3	0.2	0.2	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
HW	PSCTL	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	KCCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CCAS	**52.3**	**52.3**	**52.3**	**52.3**	**52.2**	**52.2**	**52.2**	**52.2**	**53.9**	**53.9**	**53.9**	**53.9**	**51.1**	**51.1**	**51.1**	**51.1**
ZB	PSCTL	89.4	123.5	182.1	507.6	100.4	143.9	143.9	634.7	79.4	112.6	112.6	466.0	65.1	94.9	94.9	404.9
	KCCA	185.5	261.5	261.5	339.2	150.2	176.9	176.9	405.3	105.7	127.4	127.4	323.2	122.6	147.9	147.9	370.6
	CCAS	**11.5**	**11.5**	**11.5**	**12.1**	**9.3**	**9.3**	**9.3**	**10.0**	**7.0**	**7.0**	**7.2**	**8.4**	**5.4**	**5.4**	**5.5**	**6.7**
UK	PSCTL	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	KCCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CCAS	**78.6**	**78.6**	**78.6**	**78.6**	**20.0**	**20.0**	**20.0**	**20.0**	**18.9**	**18.9**	**18.9**	**18.9**	**17.8**	**17.8**	**17.8**	**17.8**
AC	PSCTL	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	KCCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CCAS	**5,853**	**5,853**	**5,853**	**5,853**	**6,703**	**6,703**	**6,703**	**6,703**	**6,967**	**6,967**	**6,967**	**6,967**	**7,056**	**7,056**	**7,056**	**7,056**

Table 12: Ruining time of all k values. (Processing time (in seconds); Best performers are highlighted in bold.)

Dataset	Method	$\epsilon=1$				$\epsilon=0.1$				$\epsilon=0.05$				$\epsilon=0.01$			
		25%	50%	75%	100%	25%	50%	75%	100%	25%	50%	75%	100%	25%	50%	75%	100%
DP	PSCTL	5.3	9.3	12.9	16.0	6.2	10.2	13.8	16.9	6.4	10.4	14.0	17.1	9.1	13.2	16.7	19.8
	KCCA	5.0	5.7	6.1	6.3	6.4	7.1	7.5	7.7	7.3	8.0	8.4	9.6	8.1	8.8	9.3	9.5
	CCAS-A	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**	**0.7**
HT	PSCTL	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	KCCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CCAS-A	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**	**322.4**
WP	PSCTL	3,219	3,855	4,127	4,211	4,798	5,434	5,707	5,797	7,080	7,719	7,995	8,085	21,103	21,789	22,080	22,172
	KCCA	4,557	6,071	6,556	6,702	5,713	7,338	7,869	8,056	6,597	8,254	8,840	9,050	13,970	15,767	16,622	16,853
	CCAS-A	**28.5**	**30.8**	**32.7**	**33.9**	**39.0**	**44.2**	**46.8**	**48.0**	**48.6**	**55.1**	**60.7**	**62.2**	**243.1**	**255.7**	**268.0**	**274.5**
ZB	PSCTL	2,438	2,983	3,441	3,707	3,509	4,053	4,511	4,777	3,788	4,390	4,884	5,150	12,181	12,968	13,569	13,835
	KCCA	3,960	4,206	4,334	4,396	4,781	5,033	5,163	5,230	4,996	5,258	5,393	5,468	9,949	10,273	10,453	10,581
	CCAS-A	**66.0**	**103.0**	**136.9**	**161.7**	**66.0**	**103.0**	**136.7**	**161.7**	**72.8**	**114.1**	**151.1**	**176.7**	**106.0**	**166.6**	**217.3**	**247.5**
UK	PSCTL	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	KCCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CCAS-A	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**	**456.6**
FS	PSCTL	119,740	185,679	239,801	280,842	125,544	191,483	245,604	286,646	135,619	201,558	255,679	296,721	201,621	267,560	321,681	–
	KCCA	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	CCAS-A	**2,167**	**2,168**	**2,169**	**2,169**	**2,167**	**2,168**	**2,169**	**2,169**	**2,273**	**2,274**	**2,275**	**2,276**	**3,090**	**3,091**	**3,092**	**3,092**

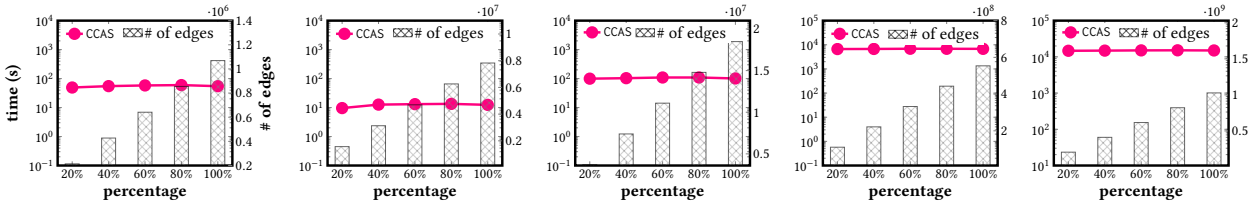


(a) HW (b) ZB (c) UK (d) AC (e) IT

Figure 15: Scalability test for CCAS algorithm (increasing δ).

C.10 Density lower bound for HCGR

In HCGR, we use the maximum clique size ω to compute a lower bound for the optimal density, i.e., $\underline{\rho_k}(G) = \binom{\omega}{k}$. While computing ω is NP-hard, the state-of-the-art algorithm [12] can efficiently find it in practice. As shown in Table 13, the time cost of finding ω is negligible (typically $< 10\%$ of total running time) across all datasets with $T = 10$ and $k = 7$. When computing ω is infeasible, we adopt a heuristic way to find a large clique of size ω' in $O(\delta \cdot m)$ time [12], yielding a lower bound $\underline{\rho_k}'(G) = \binom{\omega'}{k}$, where $\omega' \leq \omega$, $\underline{\rho_k}'(G) \leq \underline{\rho_k}(G)$, δ and m denote the degeneracy value and the number of edges in G respectively. We further develop CCAS-Heu by using ω' instead of ω as the lower bound used in HCGR. As reported



(a) HW (b) ZB (c) UK (d) AC (e) IT

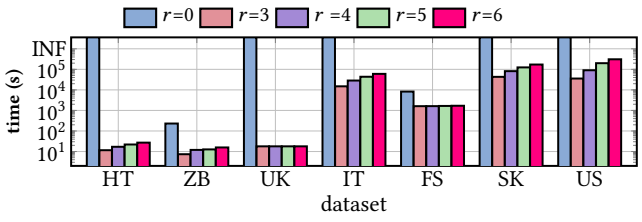
Figure 16: Scalability test for CCAS algorithm (increasing m).

Table 14: Comparison CCAS with CCAS-Heu on all datasets.

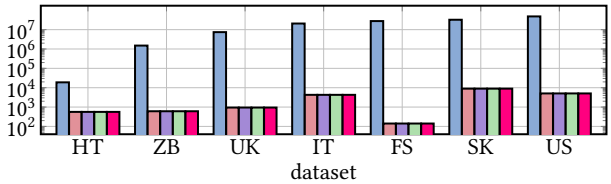
Dataset	$k = 7$		$k = 15$	
	CCAS (s)	CCAS-Heu (s)	CCAS (s)	CCAS-Heu (s)
BG	0.32	0.32	0.51	0.50
EB	28.92	28.92	308.86	308.86
WT	1.34	1.33	3.04	3.03
SD	1.13	1.12	2.09	2.08
DP	1.45	1.44	1.12	1.11
HT	11.38	11.37	11.59	11.58
HW	52.69	52.70	53.92	53.93
ZB	9.86	10.01	7.39	7.53
UK	102.85	102.90	17.86	17.90
AC	6,317.72	6,317.36	6,762.32	6,762.01
IT	14,316.31	14,295.20	14,921.61	14,900.50
FS	1,539.34	1,530.77	1,436.45	1,425.89

Table 13: Total running time vs. maximum clique time.

Dataset	Total time (s)	Maximum clique time (s)	Ratio
BG	0.32	0.001	0.003
EB	28.92	0.001	<0.001
WT	1.33	0.02	0.017
SD	1.12	0.03	0.029
DP	1.45	0.04	0.028
HT	11.38	0.18	0.016
HW	52.70	0.79	0.015
ZB	9.86	6.23	0.631
UK	102.85	2.63	0.026
AC	6,317	9.25	0.001
IT	14,316	39.05	0.003
FS	1,539	387.22	0.252



(a) running time



(b) # of vertices

Figure 17: Effectiveness of the graph reduction technique.

in Table 14, CCAS and CCAS-Heu exhibit comparable performance, validating the efficiency and applicability of our graph reduction technique.

C.11 SCT cardinality and memory usage.

We report the SCT cardinality and memory usage for all datasets at $k = 7$ and $k = 15$ in Table 15, along with the corresponding dataset statistics. In practice, the actual memory usage remains low despite the high theoretical complexity, thanks to the inherent sparsity and structural properties of real-world graphs (e.g., the power-law degree distribution), as well as our graph reduction technique, which significantly reduces the memory costs. As a result, CCAS is capable of efficiently handling large-scale real-world graphs while maintaining moderate memory usage (e.g., less than 80 GB for graphs with over 2 billion edges). In terms of scalability, storing the SCT typically consumes around six times the memory required to store the original graph. This suggests that CCAS is capable of scaling to extremely large graphs on modern servers (e.g., with 512 GB of memory). Based on this observation, we estimate that CCAS can handle input graphs with up to 10 billion edges on such a machine.

C.12 Experiment on very dense dataset

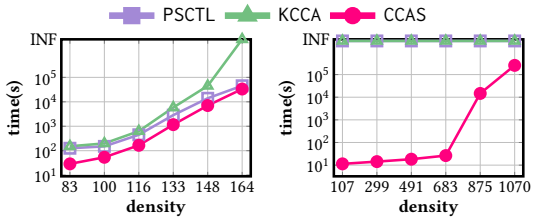
To further investigate the performance of CDS algorithms on extremely dense graphs with a high number of overlapping cliques, we select BG and HT as seed graphs and randomly append edges to them, gradually increasing their densities and the number of overlapping cliques. Specifically, for each graph, we generate five new graphs with progressively higher density. We then run PSCTL, KCCA, and CCAS on these newly generated denser graphs with k=7 and T=10, and report their results in Figure 18. We observe the following: (1) CCAS still always outperforms PSCTL and KCCA on all datasets. (2) As the graph becomes denser, all algorithms incur higher runtimes due to the increased number of cliques and the cost of updating vertex weights. Meanwhile, the performance gap between CCAS and the two competitors narrows, since our graph reduction technique becomes less effective on extremely dense graphs where fewer vertices can be pruned.

C.13 Efficiency of without HCGR

We compare the efficiency of our algorithm CCAS without graph reduction (denoted by CCAS-NR) and the two SOTA CDS algorithms. As shown in Figure 8, we can see that: (1) the runtime of PSCTL and KCCA increases proportionally with ϵ decrease, while CCAS-NR maintains a nearly constant runtime. This stable performance is because our method achieves a better convergence rate compared with PSCTL and KCCA; (2) even without the graph reduction

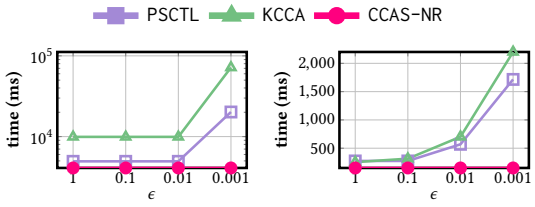
Table 15: Dataset statistics and properties of SCTs for $k = 7$ and $k = 15$.

Dataset	Graph Statistics					$k = 7$		$k = 15$	
	n	m	Density ($\frac{m}{n}$)	δ	Size (MB)	# of Nodes	Memory (MB)	# of Nodes	Memory (MB)
BG	1,716	31,564	18.4	60	5	201,201	246	160,152	242
EB	507	42,176	83.2	118	5	12,612,082	890	16,555,507	1,085
WT	120,834	237,551	2.0	54	9	1,580,404	319	1,517,500	317
SD	77,360	469,180	6.1	54	15	1,119,307	303	968,608	292
DP	317,080	1,049,866	3.3	113	30	11,778	244	11,626	234
HT	22,908	2,444,798	106.7	561	68	158,714	303	158,638	296
HW	1,069,126	56,306,652	52.7	2,208	433	2,440,917	1,529	2,440,825	1,303
ZB	7,827,193	62,246,014	7.9	267	479	91,174	1,566	91,006	1,412
UK	18,483,190	261,787,260	14.2	943	1,999	446,012	6,924	445,920	6,046
AC	22,743,892	553,903,073	24.4	3,247	4,226	5,282,730	13,110	5,282,654	13,903
IT	41,290,648	1,027,474,947	24.9	3,224	7,840	8,597,288	27,471	8,597,212	23,835
FS	124,836,180	1,806,067,135	14.5	304	13,782	15,849	42,437	15,882	41,832
SK	50,636,059	1,810,063,330	35.8	4,510	13,812	20,005,124	48,803	20,004,972	47,159
US	77,449,748	2,635,849,931	34.0	5,014	20,113	37,640,917	71,250	12,812,958	67,963



(a) EB (b) HT

Figure 18: Efficiency results on very dense datasets.



(a) EB ($k = 5$) (b) BG ($k = 13$)

Figure 19: Time cost to obtain (1 - ϵ)- approximation CDS.

technique, CCAS still achieves the best performance in real-world datasets.

D MORE DISCUSSIONS

D.1 Applications of CDS

The CDS solution has been used in many fundamental graph data management tasks, such as supporting graph visualization [85, 86], community detection (or search) [4, 15, 28, 74, 75], identifying near-cliques [27, 46, 49, 55, 74], and path/reachability queries [17, 42]. In addition, the CDS problem also as a key task in graph data management, has gained notable attention at top-tier data management conferences, including VLDB 2019 [28], 2020 [69], 2022 [27], SIGMOD 2023 [36], and 2024 [87].

Finding CDS efficiently is very useful in many graph data mining applications. Specifically, it can help identify research communities in the DBLP network [23,59,60], detect subnetworks with a specific function in the biology network [28, 74, 75] and clusters in senators' networks on US bill voting [74], and discover compact dense subgraphs from e-commerce and social networks [28] when k is relatively small. In addition, identifying CDS with large k also has many applications. As shown in [27, 46, 49, 55, 74], a CDS becomes more akin to a large near-clique as k grows, useful for tasks like finding biologically relevant groups in protein interactions [19, 41, 74, 75], community detection [4, 15, 75], and anomaly detection [31, 70, 83], etc. In many of these applications, finding a "near-clique" is very important since a "near-clique" can be considered a clique in the forming stage or one with missing edges due to data corruption.

D.2 Comparison of different density measures

While the definitions of the edge-density densest subgraph (EDS), triangle-density densest subgraph (TDS), and CDS differ, the CDS problem can be viewed as a generalized extension of EDS and TDS, since an edge and a triangle can be regarded as a 2-clique and a 3-clique, respectively. In real-world applications, EDS, TDS, and CDS are suited to different scenarios. We summarize their differences in terms of definition, cohesiveness, scalability, structural complexity, and typical application scenarios in Table 17. As we shall see, according to their definitions, moving from EDS to TDS and then to CDS, the cohesiveness and structural complexity of the subgraphs gradually increase, while the scalability gradually decreases. Similarly, in terms of computational complexity, both the space and time complexities also gradually increase from EDS to TDS and then to CDS.

While they share common key applications, such as community detection, fraud detection, and the discovery of biologically relevant groups, each is more suitable for different types of scenarios. For networks where relationships can be naturally represented as simple links, such as communication networks and transportation systems, EDS is more appropriate due to its focus on edge connectivity. In contrast, TDS is preferable when relationships are better characterized by local clustering, such as identifying tightly-knit research communities in the DBLP network or revealing coherent voting blocs in U.S. Senate bill voting networks. When modeling higher-order and more complex group interactions is required, CDS offers significant advantages. For instance, in biological networks, CDS can effectively capture functional protein subnetworks, such as those involved in "subcellular localization", "cellular transport",

Table 16: Results across different CDS algorithms in terms of accuracy, running time (s), and memory usage (MB).

Dataset	k	Accuracy				Running time (s)				Memory (MB)			
		PSCTL	KCCA	SuperGreedy++	CCAS	PSCTL	KCCA	SuperGreedy++	CCAS	PSCTL	KCCA	SuperGreedy++	CCAS
BG	5	0.99	0.99	1.00	1.00	1.3	5.5	3.0	0.3	27	7,677	3	245
	7	1.00	1.00	1.00	1.00	1.7	5.5	97.0	0.3	29	7,686	2	246
	9	1.00	1.00	1.00	1.00	1.6	5.4	2,253.0	0.3	28	7,679	3	244
	11	1.00	1.00	1.00	1.00	1.4	5.3	32,550.0	0.3	28	7,675	2	241
	13	0.99	0.99	—	1.00	1.3	5.2	INF	0.3	27	7,673	—	242
EB	5	0.99	0.99	1.00	1.00	24.0	59.0	86.0	8.2	185	8,426	3	493
	7	1.00	0.99	1.00	1.00	126.0	151.0	19,076.0	28.9	1,291	9,157	4	890
	9	1.00	0.99	—	1.00	190.2	197.3	INF	44.4	2,540	9,378	—	1,068
	11	1.00	0.99	—	1.00	205.0	200.0	INF	44.2	2,541	9,396	—	1,086
	13	1.00	0.99	—	1.00	207.1	200.0	INF	46.0	2,540	9,396	—	1,086
WT	5	1.00	0.99	1.00	1.00	3.2	6.8	2.5	0.9	61	7,775	11	286
	7	0.99	0.99	1.00	1.00	7.9	7.8	26.4	1.3	89	7,805	11	323
	9	1.00	0.99	1.00	1.00	10.3	8.0	194.7	1.4	120	7,804	11	321
	11	1.00	1.00	1.00	1.00	11.1	9.7	763.3	1.5	120	7,805	11	319
	13	1.00	1.00	1.00	1.00	9.5	7.4	1,546.3	1.4	120	7,797	12	320
SD	5	0.99	0.99	1.00	1.00	2.7	8.8	2.8	0.8	57	7,811	19	287
	7	1.00	0.99	1.00	1.00	5.6	10.2	21.4	1.1	92	7,827	19	303
	9	1.00	0.99	1.00	1.00	5.7	7.7	164.5	1.1	92	7,821	19	303
	11	1.00	1.00	1.00	1.00	5.9	7.4	728.1	1.0	91	7,810	20	301
	13	1.00	1.00	1.00	1.00	5.1	7.2	1,715.3	0.9	91	7,799	20	294
HT	7	—	—	—	1.00	INF	INF	INF	11.4	OOM	OOM	—	303
	11	—	—	—	1.00	INF	INF	INF	14.5	OOM	OOM	—	296
	15	—	—	—	1.00	INF	INF	INF	11.6	OOM	OOM	—	296
	19	—	—	—	1.00	INF	INF	INF	13.1	OOM	OOM	—	292
	23	—	—	—	1.00	INF	INF	INF	12.8	OOM	OOM	—	292
HW	10	—	—	—	1.00	INF	INF	INF	54.2	OOM	OOM	—	1,310
	15	—	—	—	1.00	INF	INF	INF	52.0	OOM	OOM	—	1,303
	20	—	—	—	1.00	INF	INF	INF	52.0	OOM	OOM	—	1,293
	25	—	—	—	1.00	INF	INF	INF	52.0	OOM	OOM	—	1,282
	30	—	—	—	1.00	INF	INF	INF	51.0	OOM	OOM	—	1,270
ZB	10	0.99	0.99	—	0.99	322.0	276.0	INF	9.2	5,798	46,747	—	1,524
	15	0.99	0.99	—	0.99	288.0	230.2	INF	7.4	5,786	21,283	—	1,412
	20	0.99	0.99	—	0.99	242.0	192.0	INF	6.1	5,770	18,064	—	1,326
	25	0.99	0.99	—	0.99	154.0	138.3	INF	4.5	4,766	15,060	—	1,283
	30	0.99	0.99	—	0.99	85.0	89.0	INF	4.1	4,337	12,381	—	1,256
UK	10	—	—	—	1.00	INF	INF	INF	18.0	OOM	OOM	—	6,232
	15	—	—	—	1.00	INF	INF	INF	17.0	OOM	OOM	—	6,046
	20	—	—	—	1.00	INF	INF	INF	16.0	OOM	OOM	—	5,834
	25	—	—	—	1.00	INF	INF	INF	14.0	OOM	OOM	—	5,612
	30	—	—	—	1.00	INF	INF	INF	13.0	OOM	OOM	—	5,426
AC	15	—	—	—	1.00	INF	INF	INF	6,762	OOM	OOM	—	13,903
	20	—	—	—	1.00	INF	INF	INF	6,684	OOM	OOM	—	12,896
	25	—	—	—	1.00	INF	INF	INF	6,690	OOM	OOM	—	12,294
	30	—	—	—	1.00	INF	INF	INF	6,755	OOM	OOM	—	12,021
	35	—	—	—	1.00	INF	INF	INF	6,687	OOM	OOM	—	11,752
IT	15	—	—	—	1.00	INF	INF	INF	14,921	OOM	OOM	—	13,903
	20	—	—	—	1.00	INF	INF	INF	14,856	OOM	OOM	—	12,896
	25	—	—	—	1.00	INF	INF	INF	14,697	OOM	OOM	—	12,294
	30	—	—	—	1.00	INF	INF	INF	14,747	OOM	OOM	—	12,021
	35	—	—	—	1.00	INF	INF	INF	14,627	OOM	OOM	—	11,752

and "protein synthesis", which typically correspond to densely connected structures like 4-cliques and 5-cliques.

D.3 Future works

In this subsection, we outline three promising future directions for the CDS problem:

- All existing CDS algorithms assume the setting of a single machine. What if the graph is too large to fit on a single machine? For example, the Facebook social network contains 1.32 billion nodes and 140 billion edges (http://newsroom.fb.com/companyinfo). Can we design I/O-efficient, distributed, or sublinear time algorithms for the CDS problem?

Table 17: Comparison of different density measures.

Model	Definition	Cohesiveness	Scalability	Structural Complexity	Typical Scenarios
EDS	$\frac{\lvert\Psi_2(G)\rvert}{\lvert V\rvert}$	Low	High	Low	Communication networks, Transportation networks, Financial network
TDS	$\frac{\lvert\Psi_3(G)\rvert}{\lvert V\rvert}$	Medium	Medium	Medium	Social networks, Citation networks
CDS	$\frac{\lvert\Psi_k(G)\rvert}{\lvert V\rvert}$	High	Low	High	Protein interaction networks, Collaboration networks, Biology networks

⋆ Note: Given a graph G=(E, V); $\Psi_k(G)$ denotes the set of k-cliques in G, clearly $|\Psi_2(G)| = |E|$;

- In practical scenarios, graphs often evolve over time, with vertices and edges being frequently inserted or deleted. However, most existing CDS algorithms are designed for static graphs and do not account for such changes. This highlights an important and challenging direction: designing efficient CDS algorithms on dynamic graphs.
- Given the advantages of GPUs in executing multiple tasks concurrently due to their high bandwidth parallel processing capabilities, designing GPU-friendly CDS algorithms is another interesting opportunity.
- While CCAS has achieved remarkable performance on the CDS problem, it still faces limitations when handling extremely dense graphs due to increased memory costs. A promising direction for future work is to design more memory-efficient algorithms for CDS problem in such scenarios.