Appendix

A. Proof of Theorem 4

Proof. This can be seen from Figure 9. Let us first define $\overline{N_P^*}(v) = \overline{N_P}(v) - \{v\}$, so $|\overline{N_P^*}(v)| \leq k-1$. In Case (i) where $(u, v) \notin E$, any vertex $w \in P$ can only fall in the following 3 scenarios: (1) $w \in \overline{N_P^*}(u)$ (which contains v), (2) $w \in \overline{N_P^*}(v)$ (which contains u), and (3) $w \in N_P(u) \cap N_P(v)$. Note that w may be in both (1) and (2). So we have:

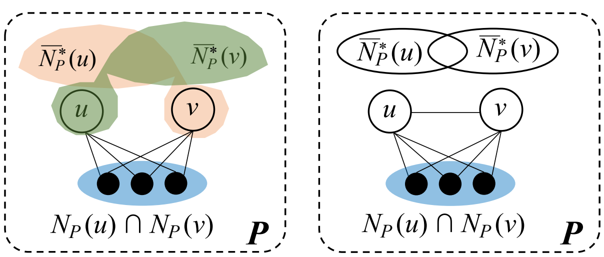
$$\begin{aligned} |P| &= |N_P(u) \cap N_P(v)| + |\overline{N_P^*}(u) \cup \overline{N_P^*}(v)| \\ &\leq |N_P(u) \cap N_P(v)| + |\overline{N_P^*}(u)| + |\overline{N_P^*}(v)| \\ &\leq |N_P(u) \cap N_P(v)| + 2(k-1) \\ &= |N_P(u) \cap N_P(v)| + 2k - 2, \end{aligned}$$

so $|N_P(u) \cap N_P(v)| \geq |P| - 2k + 2 \geq q - 2k + 2$.

In Case (ii) where $(u, v) \in E$, any vertex $w \in P$ can only be in one of the following 4 scenarios: (1) $w = u$, (2) $w = v$, (3) $w \in N_P(u) \cap N_P(v)$, and (4) $w \in \overline{N_P^*}(u) \cup \overline{N_P^*}(v)$, so:

$$\begin{aligned} |P| &= 2 + |N_P(u) \cap N_P(v)| + |\overline{N_P^*}(u) \cup \overline{N_P^*}(v)| \\ &\leq 2 + |N_P(u) \cap N_P(v)| + |\overline{N_P^*}(u)| + |\overline{N_P^*}(v)| \\ &\leq 2 + |N_P(u) \cap N_P(v)| + 2(k-1) \\ &= |N_P(u) \cap N_P(v)| + 2k, \end{aligned}$$

so $|N_P(u) \cap N_P(v)| \geq |P| - 2k \geq q - 2k$. $\square$



(a) $(u, v) \notin E$ (b) $(u, v) \in E$

Fig. 9. Second-Order Pruning

B. Proof of Theorem 6

Proof. We prove by contradiction. Assume that there exists $K' = P_m \cap N_C(v_p)$ with $|K'| > |K|$. Also, let us denote by $\psi = \{w_1, w_2, \ldots, w_\ell\}$ the vertex ordering of $w \in N_C(v_p)$ in Line 5 to create K, as illustrated in Figure 10.

Specifically, Figure 10 top illustrates the execution flow of Lines 4–8 in Algorithm 3, where w_1, w_3 and w_4 select their non-neighbor u_1 as u_m in Line 5, w_2 and w_5 select u_2 as u_m, and w_6 selects u_3 as u_m. We define $\{w_1, w_3, w_4\}$ as u_1-group, $\{w_2, w_5\}$ as u_2-group, and $\{w_6\}$ as u_3-group.

Let us consider the update of $\sup_P(u_1)$, whose initial value computed by Line 2 is assumed to be 2. When processing w_1, Line 7 decrements $\sup_P(u_1)$ as 1. Then, w_3 decrements it as 0. When processing w_4, since Line 4 already finds that $\sup_P(u_1) = 0$, w_4 is excluded from K (i.e., Line 8 does not add it to ub). In a similar spirit, $w_5 \notin K$ since $\sup_P(u_2) = 0$, and $w_6 \notin K$ since $\sup_P(u_3) = 0$.

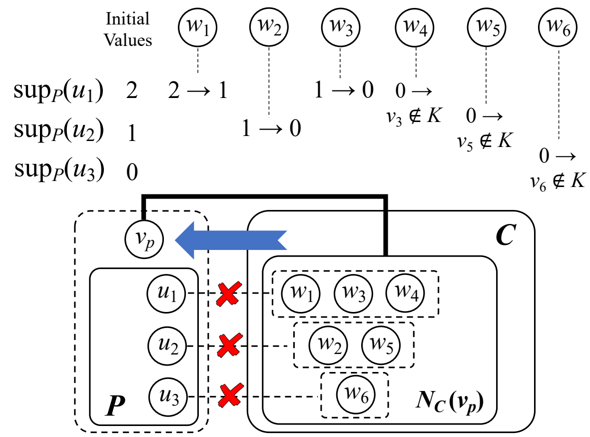


Fig. 10. Illustration of the Proof of Theorem 6

Note that since w_1, w_3 and w_4 cannot co-exist in a k-plex containing $P \cup \{v_p\}$, they cannot all belong to K'. In other words, if $w_4 \notin K$ belongs to K', then at least one of w_1 and w_3 is not in K'. In a similar spirit, if $w_5 \notin K$ belongs to K', then $w_2 \notin K'$. As for those u_i whose initial value of $\sup_P(u_i)$ is 0, we can show that any element in u_i-group can belong to neither K nor K'. See u_3-group $= \{w_6\}$ in Figure 10 for example. This is because if w is added to P, then $\overline{d_P}(u_i) > k$ so P cannot be a k-plex.

In general, in each u_i-group where $\sup_P(u_i) \neq 0$, if a vertex $w \in K' - K$ exists (e.g., w_4 in Figure 10), then there must exist a different vertex $w' \in K - K'$ in the u_i-group (e.g., w_1 or w_3). This implies that $|K - K'| \geq |K' - K|$.

Therefore, we have

$$\begin{aligned} |K| &= |K - K'| + |K \cap K'| \\ &\geq |K' - K| + |K \cap K'| \ = \ |K'|, \end{aligned}$$

which contradicts with our assumption $|K'| > |K|$. $\square$

C. Proof of Lemma 1

Proof. To implement Algorithm 4, given a seed graph $G_i = (V_i, E_i)$, we maintain $d_P(v)$ for every $v \in V_i$. These degrees $d_P(.)$ are incrementally updated; for example, when a vertex v_p is moved into P, we will increment $d_P(v)$ for every $v \in N_{G_i}(v_p)$. As a result, we can compute $\overline{d_P}(v) = |P| - d_P(v)$ and $\sup_P(v) = k - \overline{d_P}(u)$ in $O(1)$ time.

Moreover, we materialize $\sup_P(u)$ for each vertex $u \in P$ in Line 2, so that in Line 5 we can access them directly to compute u_m, and Line 7 can be updated in $O(1)$ time.

Since P is a k-plex of G_i, $|P|$ is bounded by $O(D + k)$ by Theorem 5. Thus, Line 2 takes $O(D + k)$ time.

Also, $|N_C(v_p)|$ is bounded by $O(D)$ since $N_C(v_p) \subseteq C_S = N_{G_i}(v_i)$, so the for-loop in Line 4 is executed for $O(D)$ iterations. In each iteration, Line 5 takes $O(k)$ time since $|\overline{N_P}(w)| \leq k$, so the entire for-loop in Line 4–8 takes $O(kD)$.

Putting them together, the time complexity of Algorithm 4 is $O(D + k) + O(kD) = O(k + (k + 1)D) \approx O(D)$ as k is usually very small constant. $\square$

D. *Proof of Lemma 2*

Proof. Since $\eta = \{v_1, \ldots, v_n\}$ is the degeneracy ordering of V and $V_i \subseteq \{v_i, v_{i+1}, \ldots, v_n\}$, we have $d_{G_i}(v_i) = |N_{G_i}(v_i)| \leq D$.

To show that $|N^2_{G_i}(v_i)| = O\left(\frac{D\Delta}{q-2k+2}\right)$, consider $E^* = \{(v,u) \,|\, v \in N_{G_i}(v_i) \wedge u \in N^2_{G_i}(v_i)\}$, which are those edges between $N_{G_i}(v_i)$ and $N^2_{G_i}(v_i)$ in Figure 2. Since $|N_{G_i}(v_i)| \leq D$, and each $v \in N_{G_i}(v_i)$ has at most Δ neighbors in $N^2_{G_i}(v_i)$, we have $|E^*| \leq D\Delta$. Also, let us denote by E' all those edges $(v,u) \in E^*$ that are valid (i.e., v and u can appear in a k-plex P in G_i with $|P| \geq q$), then $|E'| \leq |E^*| \leq D\Delta$.

Recall from Corollary 1 that if $u \in N^2_{G_i}(v_i)$ belongs to a valid k-plex in G_i, then $|N_{G_i}(u) \cap N_{G_i}(v_i)| \geq q - 2k + 2$. This means that each valid $u \in N^2_{G_i}(v_i)$ share with v_i at least $(q-2k+2)$ common neighbors that are in $N_{G_i}(v_i)$ (c.f., Figure 2), or equivalently, u is adjacent to (or, uses) at least $(q-2k+2)$ edges (v,u) in E'. Therefore, the number of valid $u \in N^2_{G_i}(v_i)$ is bounded by $\frac{|E'|}{q-2k+2} \leq \frac{D\Delta}{q-2k+2}$.

It may occur that $\frac{D\Delta}{q-2k+2} > n$, in which case we use $|N^2_{G_i}(v_i)| = O(n)$ instead. Combining the above two cases, we have $|N^2_{G_i}(v_i)| = O(r_1)$ where $r_1 = \min\left\{\frac{D\Delta}{q-2k+2}, n\right\}$.

Finally, the number of subsets $S \subseteq N^2_{G_i}(v_i)$ ($|S| \leq k-1$) (c.f., Line 7 of Algorithm 2) is bounded by $C^0_{r_1} + C^1_{r_1} + \cdots + C^{k-1}_{r_1} \approx O\left(r_1^k\right)$, since k is a small constant. □

E. *Proof of Theorem 8*

Proof. We just showed that the recursive body of Branch(.) takes time $O(|P|(|C|+|X|))$. Let us first bound $|X|$. Recall Algorithm 2, where by Line 9, vertices of $X \subseteq X_S$ are from either V'_i or $(N^2_{G_i}(v_i) - S)$. Moreover, by Line 5, vertices of V'_i are from either $N_G(v_i)$ or $N^2_G(v_i)$. Since $(N^2_{G_i}(v_i) - S) \subseteq N^2_G(v_i)$, vertices of X are from either $N_G(v_i)$ or $N^2_G(v_i)$. Let us denote $X_1 = X \cap N_G(v_i)$ and $X_2 = X \cap N^2_G(v_i)$.

We first bound X_2. Consider $E^* = \{(u,v) \,|\, u \in N_G(v_i) \wedge v \in X_2\}$. Since $|N_G(v_i)| \leq \Delta$, and each $v \in N_G(v_i)$ has at most Δ neighbors in $N^2_G(v_i)$, we have $|E^*| \leq \Delta^2$. Recall from Theorem 4 that if $v \in X_2$ belongs to k-plex P with $|P| \geq q$, then $|N_G(v) \cap N_G(v_i)| \geq q - 2k + 2$. This means that each $v \in X_2$ share with v_i at least $(q-2k+2)$ common neighbors that are in $N_G(v_i)$, or equivalently, v is adjacent to (or, uses) at least $(q-2k+2)$ edges (u,v) in E^*. Therefore, the number of $v \in X_2$ is bounded by $\frac{|E^*|}{q-2k+2} \leq \frac{\Delta^2}{q-2k+2}$.

As for $X_1 \subseteq N_G(v_i)$, we have $|X_1| \leq \Delta$. In general, we do not set q to be too large in reality, or there would be no results, so $(q-2k+2)$ is often much smaller than Δ. Therefore, $|X| = |X_1| + |X_2| = O(\frac{\Delta^2}{q-2k+2} + \Delta) \approx O(\frac{\Delta^2}{q-2k+2})$.

Since $|P|$ is bounded by $O(D+k) \approx O(D)$ by Theorem 5, and $C \subseteq C_S$ so $|C| \leq D$, the recursive body of Branch(.) takes $O(|P|(|C|+|X|)) \approx O\left(D(D + \frac{\Delta^2}{q-2k+2})\right) \approx O\left(\frac{D\Delta^2}{q-2k+2}\right)$ time. This is because $\frac{\Delta^2}{q-2k+2} > \Delta \geq D$.

It may occur that $\frac{\Delta^2}{q-2k+2} > n$, in which case we use $|X| = O(n)$ instead, so the recursive body of Branch(.) takes $O(|P|(|C|+|X|)) \approx O\left(D(D+n)\right) = O(nD)$ time.

Combining the above two cases, the recursive body of Branch(.) takes $O(r_2)$ time where $r_2 = \min\left\{\frac{D\Delta^2}{q-2k+2}, nD\right\}$.

By Lemma 3, Branch$(G_i, k, q, P_S, C_S, X_S)$ in Line 10 of Algorithm 2 recursively calls the body of Algorithm 4 for $O(\gamma_k^D)$ times, so the total time is $O(r_2\gamma_k^D)$.

Finally, we have at most $O(n)$ initial task groups (c.f., Line 3 of Algorithm 2), and by Lemma 2, each initial task group with seed vertex v_i generates $O\left(r_1^k\right)$ sub-tasks that call Branch$(G_i, k, q, P_S, C_S, X_S)$. So, the total time cost of Algorithm 2 is $O\left(nr_1^k r_2 \gamma_k^D\right)$. □

F. *Proof of Theorem 10*

Proof. Assume that $u_1 \in N^2_{G_i}(v_i)$ and $u_2 \in N_{G_i}(v_i)$ co-occur in a k-plex P with $|P| \geq q$. Let us assume $u_1 \in S$ and $P^+ = P_S \cup \{u_2\}$, then $P^+ \subseteq P$. As the proof of Theorem 9 has shown, we have $|P_S| \leq k$, so $|P^+| \leq k+1$.

Applying Eq (4) in Lemma 4 with $P = P^+$, $u = u_1$ and $v = u_2$, we require

$$|P^+| + \sup_{P^+}(u_1) + \sup_{P^+}(u_2) + |N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| \geq q,$$

or equivalently (recall that $|P^+| \leq k+1$),

$$|N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| \geq q - (k+1) - \sup_{P^+}(u_1) - \sup_{P^+}(u_2).$$

If $(u_1, u_2) \in E_i$, then $\sup_{P^+}(u_1) \leq k-2$ since $v_i \in P_S$ is a non-neighbor of u_1 besides u_1 itself in P_S, and $\sup_{P^+}(u_2) \leq k-1$ since u_2 is a non-neighbor of itself in P_S. Thus,

$$\begin{aligned} |N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| &\geq q - (k+1) - \max\{k-2, 0\} \\ &\quad - (k-1) \\ &= q - 2k - \max\{k-2, 0\} \end{aligned}$$

While if $(u_1, u_2) \notin E_i$, then $\sup_{P^+}(u_1) \leq k-3$ since $v_i \in P_S$ is a non-neighbor of u_1 besides $u_1, u_2 \in P^+$, and $\sup_{P^+}(u_2) \leq k-2$ since $u_1, u_2 \subseteq P^+$ are the non-neighbors of u_2. Thus, we have

$$\begin{aligned} |N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| &\geq q - (k+1) - \max\{k-2, 0\} \\ &\quad - \max\{k-3, 0\} \\ &= q - k - \max\{k-2, 0\} \\ &\quad - \max\{k-2, 1\} \end{aligned}$$

This completes our proof of Theorem 10. □

G. *Proof of Theorem 11*

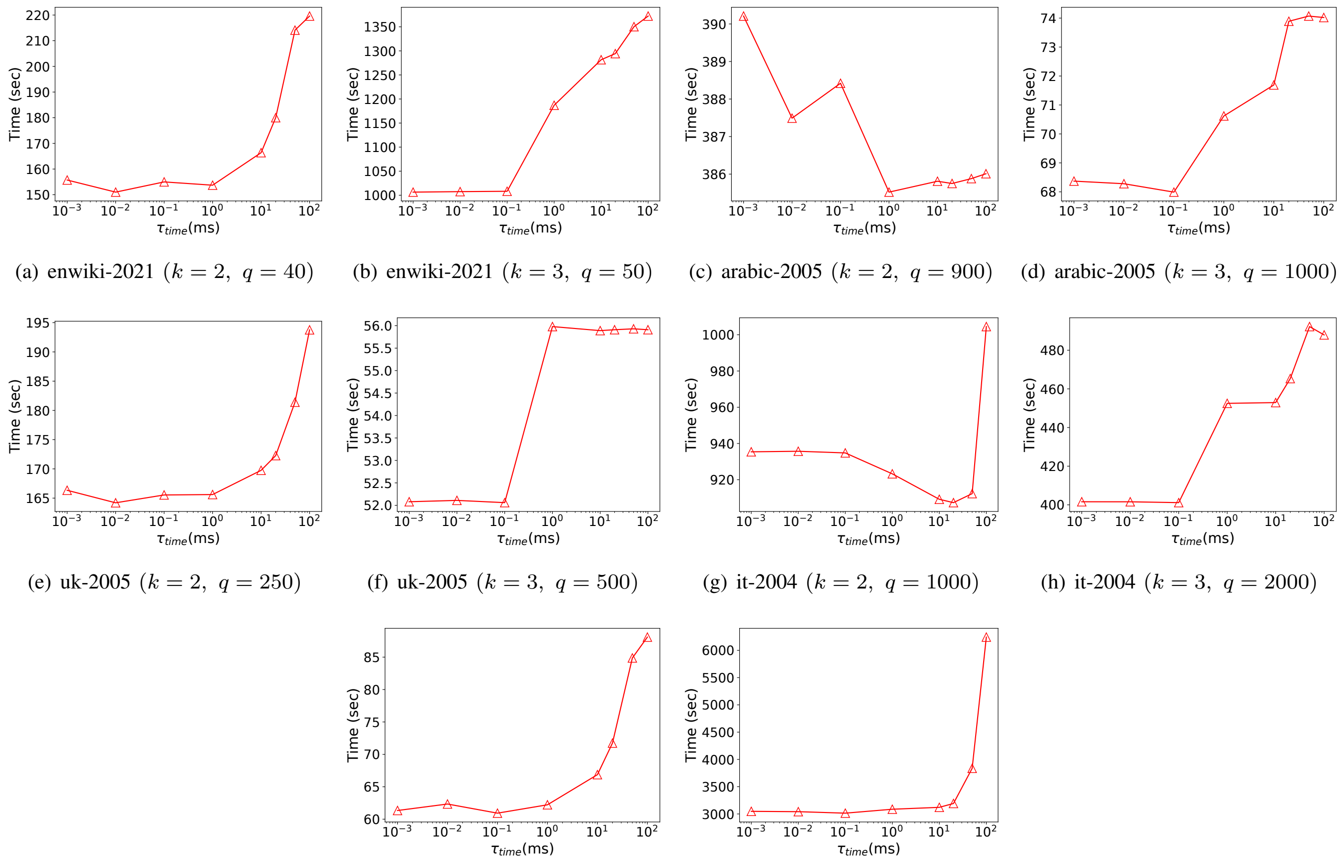
Proof. Assume that u_1, $u_2 \in N_{G_i}(v_i)$ co-occur in a k-plex P with $|P| \geq q$. Let us define $P^+ = P_S \cup \{u_1, u_2\}$, then $P^+ \subseteq P$. As the proof of Theorem 9 has shown, we have $|P_S| \leq k$, so $|P^+| \leq k+2$.

Applying Eq (4) in Lemma 4 with $P = P^+$, $u = u_1$ and $v = u_2$, we require

$$|P^+| + \sup_{P^+}(u_1) + \sup_{P^+}(u_2) + |N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| \geq q,$$

or equivalently (recall that $|P^+| \leq k+2$),

$$|N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| \geq q - (k+2) - \sup_{P^+}(u_1) - \sup_{P^+}(u_2).$$



(a) enwiki-2021 $(k = 2,\ q = 40)$ (b) enwiki-2021 $(k = 3,\ q = 50)$ (c) arabic-2005 $(k = 2,\ q = 900)$ (d) arabic-2005 $(k = 3,\ q = 1000)$

(e) uk-2005 $(k = 2,\ q = 250)$ (f) uk-2005 $(k = 3,\ q = 500)$ (g) it-2004 $(k = 2,\ q = 1000)$ (h) it-2004 $(k = 3,\ q = 2000)$

(i) webbase-2001 $(k = 2,\ q = 400)$ (j) webbase-2001 $(k = 3,\ q = 800)$

Fig. 11. The Running Time (sec) of Parallel Ours with Different τ_{time} on Five Large Datasets

If $(u_1, u_2) \in E_i$, then $\sup_{P^+}(u_1) \leq k - 1$ (resp. $\sup_{P^+}(u_2) \leq k - 1$), since u_1 (resp. u_2) is a non-neighbor of itself in P^+. Thus,

$$|N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| \geq q - (k+2) - 2 \cdot (k-1) = q - 3k.$$

While if $(v_1, v_2) \notin E_i$, then $\sup_{P^+}(u_1) \leq k - 2$ (resp. $\sup_{P^+}(u_2) \leq k - 2$), since u_1 (resp. u_2) is a non-neighbor of $u_1, u_2 \in P^+$. Thus,

$$\begin{aligned} |N_{C_S^-}(u_1) \cap N_{C_S^-}(u_2)| &\geq q - (k+2) - 2 \cdot \max\{k-2, 0\} \\ &= q - k - 2 \cdot \max\{k-1, 1\}. \end{aligned}$$

This completes our proof of Theorem 11. $\square$

H. Effect of τ_{time}

We vary τ_{time} from 10^{-3} to 100 and evaluate the running time of our parallel algorithm on five large datasets with the same parameters as Table III. The results are shown in Figure 11, where we can see that an inappropriate parameter τ_{time} (e.g., one that is very long) can lead to a very slow performance. Note that without the timeout mechanism (as is the case in ListPlex and Ours), we are basically setting $\tau_{time} = \infty$ so the running time is expected to be longer (e.g., than when $\tau_{time} = 100$) due to poor load balancing.