APPENDIX A
COMPLEXITY ANALYSIS

Given a directed graph $G = (V, E)$, we evaluate the worst-case running time of Algorithm 4. Let $T(m, n)$ be the worst-case running time of function BK2$(G, k_1, k_2, q, P, C, X)$ where $m = |C|$ and $n = |P \cup C|$, and we define $k = max(k_1, k_2)$. Next, we analyze $T(m, n)$ by considering four different cases:

I. When $C = \emptyset$ (Lines 4–6), then

$$T(m, n) = T(0, n) \leq c_1, \tag{4}$$

where c_1 is a constant.

II. When $P \cup C$ is not a (k_1, k_2)-plex and the pivot vertex is in P (Lines 8–17). Here, $p+1$ branches are created each with a smaller candidate set C'. Before each call the sets C' and X' are further shrunk in Lines 15–16 which takes time $O(|V|^2)$. The worst-case running time of this case is, therefore

$$\begin{aligned} T(m, n) \leq (p+1)c_2|V|^2 + \sum_{i=1}^{p} T(m-i, n-1) \\ + T(m-d, n-d+p), \end{aligned} \tag{5}$$

where c_2 is a constant.

III. When $P \cup C$ is a (k_1, k_2)-plex (Lines 19–24), then no further branches are created, and the cost in this case is only incurred by updating X' in Line 22, which is

$$T(m, n) \leq c_2|V|^2 \tag{6}$$

IV. When $P \cup C$ is not a (k_1, k_2)-plex and the pivot vertex is in C (Lines 25–30). The cost in this case is

$$T(m, n) \leq c_2|V|^2 + T(m-1, n) + T(m-1, n-1).$$

In the worst case, no candidate vertex is reduced from $C - \{v_p\}$ in Line 28 so v_p remains the minimum degree vertex in the subsequent recursive call at Line 30 (within which $v_p \in P' = P \cup \{v_p\}$). Hence, by expanding $T(m-1, n)$ using Equation (5) we have:

$$\begin{aligned} T(m, n) \leq (p+2)c_2|V|^2 + \sum_{i=1}^{p} T(m-1-i, n-1) \\ + T(m-1-d, n-d+p) + T(m-1, n-1). \end{aligned} \tag{7}$$

Next, we compute a closed-form upper-bound of $T(m, n)$ which satisfies the Equations (4)–(7), and we can drop the second argument n from $T(m, n)$ since the base case expressions in Equations (4) and (6) are not a function of n. Then, the largest $T(m)$ is upper-bounded by Equations (4) and (7), i.e.

$$T(m) \leq \begin{cases} c_1 & \text{if } m = 0 \\ (p+2)c_2|V|^2 + \sum_{i=0}^{p} T(m-1-i) & \\ + T(m-1-d) & \text{otherwise} \end{cases} \tag{8}$$

Since $p \leq d-1$ and $p < k$, the worst-case is reached when $p = k-1$ and $d = k$. Then, Equation (8) can be rewritten as:

$$T(m) \leq \begin{cases} c_1 & \text{if } m = 0 \\ (k+1)c_2|V|^2 + \sum_{i=0}^{k} T(m-1-i) & \text{otherwise} \end{cases} \tag{9}$$

We show that $T(m) = \mathcal{O}(\alpha_k \gamma_k^m - \beta_k)$ where γ_k is largest real root of $x^{k+2} - 2x^{k+1} + 1 = 0, \beta_k = \frac{(k+1)c_2|V|^2}{k}$ and $\alpha_k = \frac{(2^k k+1)\beta_k + 2^k c_1}{\gamma_k^{k+1}}$.

Lemma 1. *Let $H(m) = \alpha_k \gamma_k^m - \beta_k$ where γ_k is largest real root of $x^{k+2} - 2x^{k+1} + 1 = 0, \beta_k = \frac{(k+1)c_2|V|^2}{k}$ and $\alpha_k = \frac{(2^k k+1)\beta_k + 2^k c_1}{\gamma_k^{k+1}}$. If $T(m)$ satisfies Equation (9), then $T(m) \leq H(m)$ for $m \geq 0$.*

Proof. We first show $T(m) \leq 2^{m-1}k\beta_k + 2^{m-1}c_1$ by induction on m.

Basis: for $m = 1, T(1) \leq (k+1)c_2|V|^2 + T(0) \leq k\beta_k + c_1$.

Induction: We assume that $T(i) \leq 2^{i-1}k\beta_k + 2^{i-1}c_1$ for all $i = 1, \ldots, m-1$ and prove that it also holds for $i = m$. By Equation (9) we have:

$$\begin{aligned} T(m) &\leq (k+1)c_2|V|^2 + T(m-1) + \cdots + T(0) \\ &= (k+1)c_2|V|^2 + c_1 + \sum_{i=1}^{m-1} T(i) \\ &\leq (k+1)c_2|V|^2 + c_1 + \sum_{i=1}^{m-1} (2^{i-1}k\beta_k + 2^{i-1}c_1) \\ &= k\beta_k(1 + \sum_{i=1}^{m-1} 2^{i-1}) + c_1(\sum_{i=1}^{m-1} 2^{i-1} + 1) \\ &= 2^{m-1}k\beta_k + 2^{m-1}c_1 \end{aligned} \tag{10}$$

Then, we prove that $T(m) \leq H(m)$ for all $m \geq 0$ again by induction on m.

Basis: for $m = 1, T(1) \leq (k+1)c_2|V|^2 + T(0) \leq k\beta_k + c_1$, and $H(1) = \alpha_k\gamma_k - \beta_k = \frac{(2^k k+1)\beta_k + 2^k c_1}{\gamma_k^k} \geq k\beta_k + c_1$ (since $\gamma_k < 2$ and hence $\gamma_k^k < 2^k$).

Induction: We assume that $T(m) \leq H(m)$ for $m = 1, \ldots, i$ and prove that it also holds for $m = i+1$. First, let us factorize $x^{k+1} - 2x^{k+1} + 1$ as $(x-1)(x^{k+1} - x^k - \cdots - 1)$. Denote $F_k(x) = (x^{k+1} - x^k - \cdots - 1)$. We have $F_k(1) < 0$ and $F_k(2) > 0$, indicating that $F_k(x) = 0$ has a root between $(1, 2)$. Combining the fact that γ_k is also the root of $F_k(x) = 0$, thus $\gamma_k^k + \cdots + 1 = \gamma_k^{k+1}$. Multiply both side with γ_k^{i-k}, we have $\gamma_k^i + \cdots + \gamma_k^{i-k} = \gamma_k^{i+1}$. Then the following inequalities hold.

$$\begin{aligned} T(i+1) &\leq T(i) + \cdots + T(i-k) + (k+1)c_2|V|^2 \\ &\leq \alpha_k(\gamma_k^i + \cdots + \gamma_k^{i-k}) - (k+1)\beta_k + k\beta_k \\ &\leq \alpha_k(\gamma_k^i + \cdots + \gamma_k^{i-k}) - \beta_k \\ &= \alpha_k\gamma_k^{i+1} - \beta_k \\ &= H(i+1) \end{aligned} \tag{11}$$

TABLE IV
THE RUNNING TIME (SEC) OF MINING LARGE MAXIMAL (k_1, k_2)-PLEXES IN PARALLEL EXECUTION

Dataset	k_1	k_2	q	#(k_1, k_2)-plexes	Serial	2 Threads	4 Threads	8 Threads	16 Threads	32 Threads
bitcoin	2	5	10	144.00	0.06	0.04	0.02	0.01	0.01	0.02
			12	0	0.01	0.00	0.00	0.00	0.00	0.00
	3	5	10	24,261	8.35	4.78	2.46	1.29	0.66	0.36
			12	286	1.05	0.62	0.33	0.17	0.09	0.05
wiki-vote	3	5	10	474	1.85	1.08	0.55	0.29	0.15	0.08
			12	0	0.08	0.05	0.02	0.01	0.01	0.01
	4	5	10	30,222	349.32	197.94	102.45	52.57	26.57	16.44
			12	94	16.63	9.76	4.99	2.61	1.32	0.91
mathoverflow	2	2	10	523,633	28.55	18.48	9.48	4.93	2.50	1.34
			12	150,888	13.35	9.01	4.56	2.38	1.22	0.69
	2	3	10	1,762,917	63.16	47.81	24.41	12.71	6.45	3.52
			12	602,862	32.44	24.92	12.61	6.63	3.36	1.86
as-caida	2	3	10	23,314	7.99	0.34	0.19	0.10	0.08	0.15
			15	185	0.93	0.02	0.01	0.01	0.01	0.02
	3	4	15	17,303	14.87	0.63	0.33	0.18	0.10	0.10
			20	0	0.19	0.01	0.00	0.00	0.00	0.00
epinions	2	3	10	773,095	23.85	18.34	9.48	4.94	2.56	1.41
			12	252,511	10.33	8.06	4.18	2.23	1.19	0.80
	3	4	12	24,599,029	2,053.04	1,174.05	605.69	314.25	158.78	85.62
			15	2,382,084	381.99	229.27	118.32	61.53	31.62	18.13
email-euall	2	3	10	17,438	1.69	1.26	0.65	0.34	0.18	0.19
			12	1442	0.58	0.39	0.21	0.11	0.07	0.12
	3	4	12	196,878	80.58	46.44	23.49	12.18	6.14	3.38
			15	1351	9.19	5.20	2.68	1.40	0.72	0.44
amazon0505	3	5	10	4,696	0.27	0.08	0.04	0.02	0.02	0.01
			12	24	0.14	0.01	0.00	0.00	0.00	0.00
	5	3	10	1,532	0.20	0.04	0.02	0.01	0.01	0.00
			12	1	0.13	0.00	0.00	0.00	0.00	0.00
web-google	2	3	15	997	0.29	0.05	0.02	0.01	0.01	0.00
			20	57	0.21	0.01	0.01	0.00	0.00	0.00
	3	4	15	1,993	0.32	0.06	0.03	0.02	0.01	0.01
			20	61	0.22	0.02	0.01	0.00	0.00	0.00
soc-pokec	2	3	15	17,980	2.11	0.92	0.49	0.25	0.21	0.39
			20	17	0.95	0.03	0.01	0.01	0.01	0.02
	3	3	15	508,884	54.91	32.24	16.52	8.65	4.41	2.41
			20	648	1.66	0.49	0.25	0.13	0.07	0.06
wiki-talk	2	2	15	22,219	287.28	173.50	87.73	44.75	22.50	11.48
			20	0	69.70	38.84	19.89	10.27	5.17	2.67
	2	3	15	165,156	525.13	376.12	191.71	97.82	48.99	25.01
			20	0	129.02	77.30	39.23	20.22	10.16	5.21
arabic-2005	2	2	1000	106,895	3,906.70	1,897.35	1,017.31	527.40	269.04	140.58
	3	3	3000	3,500	111.60	60.20	29.56	15.10	8.69	4.64
uk-2005	2	2	200	117,255	301.09	162.70	84.11	43.34	21.82	11.38
	3	3	400	66,639	168.98	91.51	46.34	23.95	12.11	6.34
it-2004	2	2	1000	15,087	1,170.05	621.80	321.12	166.53	85.40	52.04
	3	3	3000	1,034	77.90	41.35	21.72	10.21	6.40	3.94
webbase-2001	2	2	300	327,882	646.41	320.84	164.46	90.02	53.56	30.75
	3	3	500	88,777	184.44	95.16	52.07	30	18.15	8.75
clue-web	2	2	10	1,215,977	26.05	14.15	7.66	3.82	1.75	1.69
	3	3	20	95,383	52.92	31.76	16.46	8.56	4.42	2.37

□

Finally, as we call BK2(.) with $m = |V|$ and k is a constant, we get the worst-case running time of our algorithm as given in Theorem 8.

APPENDIX B
COMPLETE RESULTS IN PARALLEL EXECUTIONS

We implemented the parallel version of Ours based on the description in Section VIII. Table IV reports the (k_1, k_2)-plex enumeration time of parallel Ours on all our datasets in Table I when running with 2, 4, 8, 16 and 32 threads, respectively, where we use the default timeout threshold $\tau_{time} = 0.1$ ms is adopted, which is tested to work consistently well on various datasets.

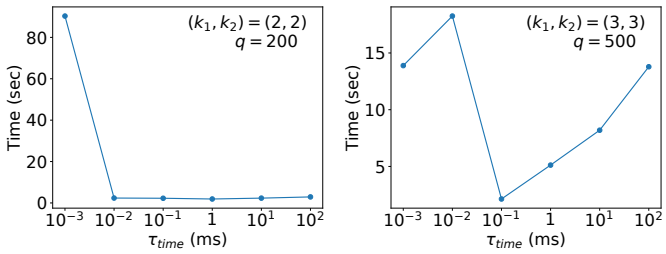
As Table IV shows, our parallel algorithm is efficient and scales up well with the number of CPU cores on all the datasets.

APPENDIX C
ABLATION STUDY

To verify the effectiveness of the various techniques we proposed, we conduct ablation study with Ours by disabling one of the technique at a time. This gives four variants

TABLE V
ABLATION STUDY.

Network	k_1	k_2	q	Ours\SOP	Ours\ITP	Ours\BRA	Ours\LAP	Ours
bitcoin	2	5	10	92.57	0.07	0.08	0.07	**0.06**
			12	48.56	0.02	0.02	0.02	**0.01**
	3	5	10	98.05	10.37	11.49	8.82	**8.35**
			12	79.78	1.13	1.49	1.09	**1.05**
wiki-vote	3	5	10	1685.38	7.73	2.55	1.98	**1.85**
			12	1423.06	0.17	0.14	0.11	**0.08**
	4	5	10	2363.89	6192.11	447.41	369.35	**349.32**
			12	1859.31	81.95	20.68	17.53	**16.63**
arabic-2005	2	2	1000	8436.63	>6 hrs	>6 hrs	>6 hrs	**3906.70**
	3	3	3000	133.37	>6 hrs	215.30	>6 hrs	**111.60**
uk-2005	2	2	200	>6 hrs	386.27	>6 hrs	>6 hrs	**301.09**
	3	3	400	412.13	189.50	>6 hrs	>6 hrs	**168.98**



(a) uk-2005 (b) webbase-2001

Fig. 11. The Running Time (sec) of Large Datasets When τ_{time} is Varied.

Ours\SOP, Ours\ITP, Ours\BRA, Ours\LAP. For example, Ours\SOP is the variant when second-order pruning is turned off in Ours. Table V shows the results of our ablation study with serial execution for the two small datasets and the first two large datasets. The other results are similar but omitted. It can be observed that Ours is the clear winner, and all of the techniques are beneficial. In particular, SOP is essential on the two small graphs to achieve orders of magnitude speedup, while ITP is also essential in a number of cases such as on wiki-vote with $k_1 = 4$ and $k_2 = 5$. The four techniques are even more important on large graphs since the program frequently runs beyond 6 hours even when missing one technique. In particular, LAP is essential to allow our four large-graph experiments to finish within 6 hours, followed by BRA for 3 experiments to be within 6 hours, then by ITP for 2 experiments and finally by SOP for 1 experiment. We can see that BK2 with BRA and LAP becomes more important than the graph pruning techniques SOP and ITP on large graphs, though SOP and ITP are more important on small graphs.

APPENDIX D
LOAD BALANCING

We also conducted experiments to study the impact of τ_{time}. Recall from Section VIII that if a task runs for a period of more than τ_{time}, it will decompose itself so that the decomposed tasks can be processed in parallel. Fig. 11 shows the impact of τ_{time} on the job running time, when using 32 threads and varying τ_{time} as 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100 for two representative settings on our large graphs uk-2005 and webbase-2001. We can see that our default setting of $\tau_{time} = 0.1$ ms does work the best in general. Also, Fig. 11(a) shows that if we set τ_{time} too small, too many tasks will be created so the task creation time dominates and the running time can be significantly increased, and Fig. 11(b) shows that if we set τ_{time} too large, load balancing suffers so the running time can be many times longer.