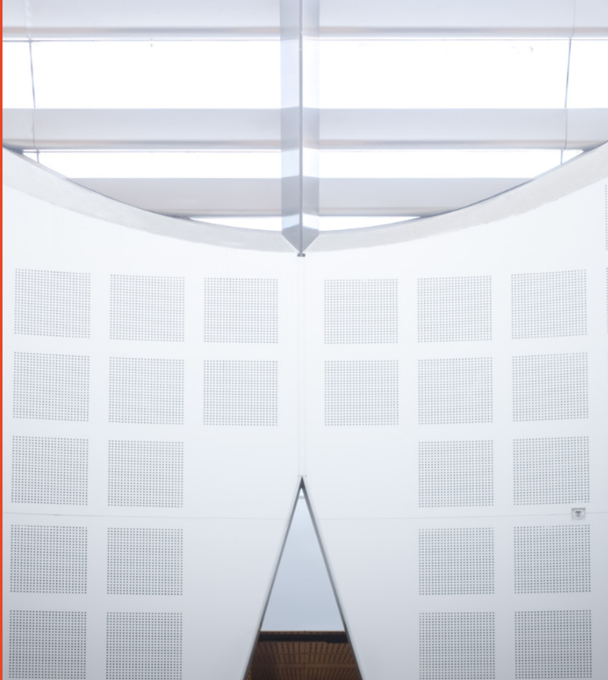
Maximum k-Plex Computation: Theory and Practice

Lijun Chang, Kai Yao

School of Computer Science
The University of Sydney

June 11, 2024

THE UNIVERSITY OF SYDNEY



Graphs are Everywhere

- A graph $G = (V, E)$ consists of a set V of vertices and a set E of edges

Figure: Social networks

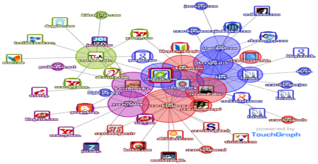


Figure: Web graphs

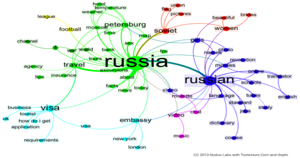


Figure: Graph of texts

Figure: Internet of things

Real Graphs are usually Globally Sparse but Locally Dense

- The entire graph is sparse, but there are groups of vertices with high concentration of edges within them.

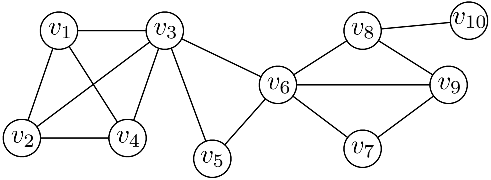
Graphs	n	m	$d_{avg}(G)$	$d_{max}(G)$	$\omega(G)$
as-Skitter	$1,694,616$	$11,094,209$	13.09	$35,455$	67
soc-LiveJournal1	$4,843,953$	$42,845,684$	17.69	$20,333$	321
uk-2005	$39,252,879$	$781,439,892$	39.82	$1,776,858$	589
it-2004	$41,290,577$	$1,027,474,895$	49.77	$1,326,744$	$3,222$

Table: Statistics of some real graphs ($\omega(G)$ is the clique number of G)

- Finding dense subgraphs is a fundamental problem with many applications.
  - community detection in social networks
  - anomaly detection in financial networks
  - protein complexes detection in biological networks
  - ...

k-Plex

- The clique model, requiring all vertices to be connected to each other, represents the most dense subgraph model.
  - Clique-related problems have been extensively studied.
  - E.g., enumerate all maximal cliques, find a maximum clique.
- However, the clique model is often too restrictive for applications
  - Various clique relaxations have been formulated in the literature, such as quasi-clique, k-plex, k-club, and k-defective clique.
- k-plex allows each vertex in the subgraph to miss up-to $k-1$ neighbors (excluding the vertex itself)
  - $\{v_1, v_2, v_3, v_4\}$ and $\{v_6, v_7, v_8, v_9\}$ are two maximum 2-plexes.



Maximum k-Plex Computation

- The maximum k-plex computation problem aims to find the k-plex with the largest number of vertices
  - It is an NP-hard problem.
- Existing exact algorithms
  - BS[1], BnB[2], Maplex[3], KpLeX[4], and kPlexS[5]
  - kPlexS only considers k-plexes of size at least $2k-1$
    - All such k-plexes are of diameter at most 2.
  - KpLeX is general, but performs much worse than kPlexS when the maximum k-plex size is at least $2k-1$.
  - None of these algorithms, except BS, beat the trivial time complexity of $\mathcal{O}^*(2^n)$.
    - The $\mathcal{O}^*(\cdot)$ notation hides polynomial factors.

[1] Mingyu Xiao et al. "A Fast Algorithm to Compute Maximum *k*-Plexes in Social Network Analysis". In: *Proc. of AAAI'17*. 2017.

[2] Jian Gao et al. "An Exact Algorithm for Maximum k-Plexes in Massive Graphs". In: *Proc. IJCAI'18*. 2018.

[3] Yi Zhou et al. "Improving Maximum k-plex Solver via Second-Order Reduction and Graph Color Bounding". In: *Proc. of AAAI'21*. 2021.

[4] Hua Jiang et al. "A New Upper Bound Based on Vertex Partitioning for the Maximum K-plex Problem". In: *Proc. of IJCAI'21*. 2021.

[5] Lijun Chang, Mouyi Xu, and Darren Strash. "Efficient Maximum k-Plex Computation over Large Sparse Graphs". In: *PVLDB* 16.2 (2022).

Summary of Time Complexities

Algorithm	Time complexity	Problem	Limitation
BS[6]	$\mathcal{O}^*(\beta_k^n)$	Maximum	None
FaPlexen[7]	$\mathcal{O}^*(\beta_k^n)$	Enumeration	None
ListPlex[8]	$\mathcal{O}^*((\alpha\Delta)^{k+1}\beta_k^\alpha)$	Enumeration	k-plex size $\geq 2k-1$
FP[9]	$\mathcal{O}^*(\beta_k^{\alpha\Delta})$	Enumeration	k-plex size $\geq 2k-1$
kPlexT	$\mathcal{O}^*((\alpha\Delta)^{k+1}\gamma_k^\alpha)$	Both problems	k-plex size $\geq 2k-1$
kPlexT	$\mathcal{O}^*\left((\alpha\Delta)^{k+1}\gamma_k^\alpha + \min\{\gamma_k^n, n^{2k-2}\}\right)$	Both problems	None

Table: A summary of the time complexities (β_k and γ_k are constants smaller than 2 that only depend on k; $\gamma_k < \beta_k$; α is the degeneracy and Δ is the maximum degree of G; kPlexT is our algorithm)

[6] Mingyu Xiao et al. "A Fast Algorithm to Compute Maximum *k*-Plexes in Social Network Analysis". In: *Proc. of AAAI'17*. 2017.

[7] Yi Zhou et al. "Enumerating Maximal *k*-Plexes with Worst-Case Time Guarantee". In: *Proc. of AAAI'20*. 2020, pp. 2442–2449.

[8] Zhengren Wang et al. "Listing Maximal k-Plexes in Large Real-World Graphs". In: *Proc. of WWW'22*. 2022, pp. 1517–1527.

[9] Qiangqiang Dai et al. "Scaling Up Maximal *k*-plex Enumeration". In: *Proc. of CIKM'22*. 2022, pp. 345–354.

Our (Branch and Bound) Algorithm

Algorithm 1: kPlexBB(G, k)

Input: A graph G and an integer $k \geq 2$

Output: A maximum k-plex in G

$P \leftarrow \emptyset$;

Branch&Bound$(G, k, \emptyset, P)$;

return P;

Procedure Branch&Bound(g, k, S, P)

/* g is the working subgraph, S is the partial solution */

$(g', S') \leftarrow$ apply reduction rules to (g, S); /* *e.g.*, RR1--RR3 */;

if g' *is a k-plex* **then**

  if $|V(g')| > |P|$ **then** $P \leftarrow V(g')$;

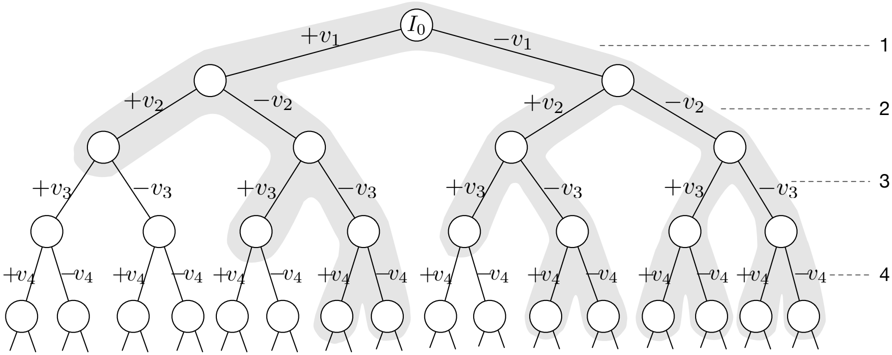
else

  $b \leftarrow$ ChooseBranchingVertex(g', k, S');

  Branch&Bound$(g', k, S' \cup \{b\}, P)$; /* Add b into S' */;

  Branch&Bound$(g' \setminus \{b\}, k, S', P)$; /* Remove b from g' */;

Recursion Tree of Our Algorithm



- Each node $I = (g, S)$ is a backtracking instance
  - S must be included, $V(g) \setminus S$ are the candidate vertices
- Prove the time complexity by induction on the recursion tree

Time Complexity Proof (General Idea)

- Consider a backtracking instance $I = (g, S)$
  - S must be included, $V(g) \setminus S$ are the candidate vertices
  - The instance size is $|I| = |V(g) \setminus S$.
- Worst-case scenario of the existing algorithms
  - Let $u \in V(g) \setminus S$ be a vertex that has exactly k non-neighbors $\{v_1, v_2, \ldots, v_k\}$
  - It generates $k+1$ branches
    1. $-u$ (remove u from the graph); the instance size is reduced by 1
    2. $+u$, $-v_1$ (add u to the solution and remove v_1); the instance size is reduced by 2
    3. $+\{u, v_1, \ldots, v_{i-1}\}$, $-v_i$, for $2 \leq i \leq k$; the instance size is reduced by $i+1$
  - The time complexity is $\mathcal{O}^*(\beta_k^n)$ where β_k is the largest real root of $x^{|I|} = x^{|I|-1} + \cdots + x^{|I|-k} + x^{|I|-(k+1)}$, equivalent to $x^{k+2} - 2x^{k+1} + 1 = 0$.
- Our algorithm
  - We generate $k+1$ branches
    1. $+u$; the instance size is reduced by 1
    2. $-\{u, v_1, \ldots, v_{i-1}\}$, $+v_i$, for $1 \leq i \leq k-1$; the instance size is reduced by $i+1$
    3. $-\{u, v_1, \ldots, v_{k-1}\}$, $+\{v_k$, all of v_k's non-neighbors$\}$; reduced by at least $k+2$
  - The time complexity is $\mathcal{O}^*(\gamma_k^n)$ where γ_k is the largest real root of $x^{|I|} = x^{|I|-1} + \cdots + x^{|I|-k} + x^{|I|-(k+2)}$.

Our Two-Stage Approach to Reduce the Exponent

Algorithm 2: kPlexT(G, k)

$P \leftarrow \emptyset$;

`/* Stage-I */`

Let $(v_1, \ldots, v_n)$ be a degeneracy ordering of the vertices of G;

for each $v_i \in V(G)$ **do**

  Let A be v_i's neighbors that are in $\{v_{i+1}, \ldots, v_n\}$, *i.e.*, $A \leftarrow N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$;

  Let g be the subgraph of G induced by $N[A] \cap \{v_i, \ldots, v_n\}$;

  Branch&Bound$(g, k, \{v_i\}, P)$;

`/* Stage-II */`

if $|P| < 2k - 2$ **then** Branch&Bound$(G, k, \emptyset, P)$;

return P;

- kPlexT runs in $\mathcal{O}(n \times (\alpha\Delta)^{k+1} \times \gamma_k^{\alpha})$ time if the maximum k-plex size $\geq 2k - 1$.
  - Any two non-adjacent vertices in k-plex $\geq 2k - 1$ must have common neighbors.
- kPlexT runs in $\mathcal{O}\big(n \times (\alpha\Delta)^{k+1} \times \gamma_k^{\alpha} + m \times \min\{\gamma_k^{n}, n^{2k-2}\}\big)$ time otherwise.

Other Contributions (in the paper)

- With slight modification, kPlexT runs in $\mathcal{O}^*((\alpha\Delta)^{k+1} \times (k+1)^{\alpha+k-\omega_k(G)})$ time when $\omega_k(G) \geq 2k-1$.
  - $\omega_k(G)$ is the maximum k-plex size.
  - $\alpha + k$ is an upper bound of $\omega_k(G)$.
- We also propose a new reduction rule and a better initialization method for improving the practical performance
- Our improved time complexities also hold for enumerating all maximal k-plexes, and maximal k-biplexes.

Performance Study

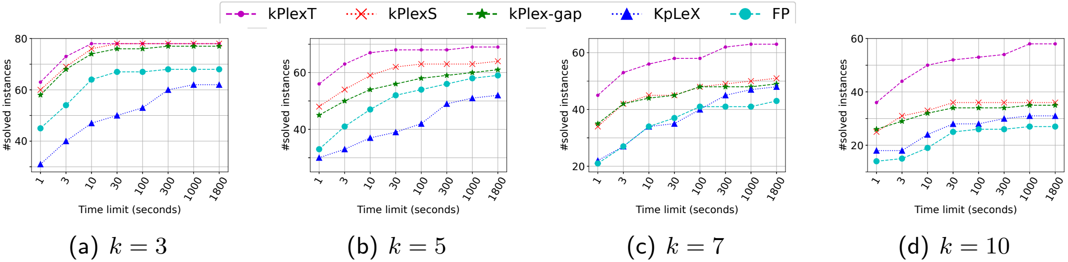


Figure: Against existing algorithms on 10th DIMACS graphs (vary time limit)

- The 10th DIMACS graphs collection contains 84 graphs with up to 5.09 $\times$ 107 vertices from the 10th DIMACS implementation challenge.

Conclusion

- We improved the time complexity for maximum k-plex computation, maximal k-plex enumeration, and maximal k-biplex enumeration.
- Our algorithm also runs faster than the existing algorithms in practice for maximum k-plex computation.
- The source code is available at `https://lijunchang.github.io/Maximum-kPlex-v2/`