Maximal Clique Enumeration with Hybrid Branching and Early Termination (Technical Report)

Kaixin Wang, Kaiqiang Yu, Cheng Long
College of Computing and Data Science, Nanyang Technological University, Singapore
{kaixin001, kaiqiang002}@e.ntu.edu.sg, c.long@ntu.edu.sg

TABLE I
THE WORST-CASE TIME COMPLEXITIES OF EXISTING VBBMC ALGORITHMS, WHERE n, h AND δ ARE THE NUMBER OF THE VERTICES, THE h-INDEX AND THE DEGENERACY OF THE GRAPH, RESPECTIVELY.

Algorithm name	Time complexity
BK [1]	$O(n \cdot (3.14)^{n/3})$
BK_Pivot [2], BK_Ref [3]	$O(n \cdot 3^{n/3})$
BK_Degree [4]	$O(hn \cdot 3^{h/3})$
BK_Degen [5], [6]	$O(\delta n \cdot 3^{\delta/3})$
BK_Rcd [7]	$O(\delta n \cdot 2^{\delta})$
BK_Fac [8]	$O(\delta n \cdot (3.14)^{\delta/3})$

I. TIME COMPLEXITIES OF VBBMC ALGORITHMS

Different variants of VBBMC have different time complexities, which we summarize in Table I. We note that BK_Rcd [7] and BK_Fac [8] do not provide the worst-case time complexity analysis in their paper. We provide the analysis as follows.

A. Time Complexity of BK_Rcd

We present the algorithm details of BK_Rcd [7] in Algorithm 1. Each branch is represented by three vertex sets S, C and X (correspondingly, $g_C = G[C]$ and $g_X = G[X]$ are the candidate and exclusion subgraph as introduced in Section III).

Algorithm 1: BK_Rcd [7]

```
   Input: A graph G = (V, E)
   Output: All maximal cliques within G
1  BK_Rcd_Rec(∅, V, ∅);
2  Procedure BK_Rcd_Rec(S, C, X)
       /* Termination if C, X are empty */
3      if C ∪ X = ∅ then
4          Output a maximal clique S; return;
       /* Branching when G[C] is not a clique */
5      while G[C] is not a clique do
6          Choose v ∈ C that minimizes |N(u, G[C])|;
7          BK_Rcd_Rec(S∪{v}, C∩N(v, G), X∩N(v, G));
8          C ← C \ {v};
9          X ← X ∪ {v};
       /* Check maximality */
10     if C ≠ ∅ and N(C, G) ∩ X = ∅ then
11         Output a maximal clique S ∪ C;
```

Theorem 1. *Given a graph $G = (V, E)$, the worst-case time complexity of BK_Rcd is $O(n\delta \cdot 2^{\delta})$, where δ is the degeneracy of the graph.*

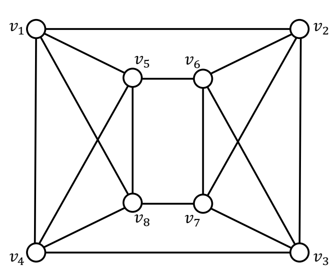

Fig. 1. A worst case for BK_Rcd algorithm.

Proof. From line 6 and lines 8-9, we observe that, given a branch $B = (S, C, X)$, the algorithm iteratively removes the vertex with the minimum degree in the remaining candidate subgraph $G[C]$. This corresponds to using degeneracy ordering of $G[C]$ for branching. Let $T(c, x)$ be the total time cost to enumerate all maximal cliques within a branch $B = (S, C, X)$, where $|C| = c$ and $|X| = x$. Consider the initial branch. Based on the degeneracy ordering, the candidate subgraph of each produced sub-branch have at most δ vertices. Consider the branches other than the initial branch. While removing the vertex with the minimum degree is intuitive enough such that the remaining graph $G[C]$ is possibe to be a non-trivial clique (other than a vertex or an edge) at line 10, it is not always the case. Consider an example as illustrated in Figure 1. The number in each vertex represents the index of the vertex. It is easy to check that the sequence $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ is a valid degeneracy ordering. However, only when C has $\{v_7, v_8\}$ inside, it is a clique (i.e., an edge). All above examples demonstrate that given a branch B, the BK_Rcd would produce $|C| - 2$ branches at the worst case. Thus, we have the following recurrence.

$$T(c, x) \leq \begin{cases} O(\delta) & c = 0 \\ O(|E|) + \sum_{v \in V} T(\delta, \Delta) & c = |V| \\ T(c-1, x) + T(c-2, x) + \cdots + T(2, x) & c \neq 0 \end{cases} \quad (1)$$

By solving the recurrence, we have the time complexity of BK_Rcd, which is $O(\delta n \cdot 2^{\delta})$. □

Algorithm 2: BK_Fac [8]

```
   Input: A graph G = (V, E)
   Output: All maximal cliques within G
1  Let v_1, ··· , v_n be in the degeneracy ordering of G;
2  for each v_i ∈ V do
3      C_i ← N(v_i, G) ∩ {v_1, ··· , v_{i−1}};
4      X_i ← N(v_i, G) ∩ {v_{i+1}, ··· , v_n};
5      BK_Fac_Rec({v_i}, C_i, X_i);

6  Procedure BK_Fac_Rec(S, C, X)
       /* Termination if C, X are empty  */
7      if C ∪ X = ∅ then
8          Output a maximal clique S; return;
       /* Initialize a pivot v, create
          branches on P                   */
9      v ← an arbitrary vertex in C;
10     P ← C \ N(v, G);
11     for each u ∈ P do
12         BK_Fac_Rec(S ∪ {u}, C ∩ N(u, G), X ∩
             N(u, G));
13         C ← C \ {u};
14         X ← X ∪ {u};
           /* Update P if possible        */
15         P ← P \ {u};
16         P′ ← C \ N(u, G);
17         if |P′| < |P| then P ← P′;
```

B. Time Complexity of BK_Fac

We present the algorithm details of BK_Fac [8] in Algorithm 2. Similarly, each branch is represented by three sets S, C and X (correspondingly, $g_C = G[C]$ and $g_X = G[X]$ are the candidate and exclusion subgraph as introduced in Section III).

Theorem 2. *Given a graph $G = (V, E)$, the worst-case time complexity of BK_Fac is $O(\delta n \cdot (3.14)^{\delta/3})$, where δ is the degeneracy of the graph.*

Proof. Based on the degeneracy ordering, each produced sub-branch from line 5 will have at most δ vertices in C_i, where δ is the degeneracy of the graph. Then consider the time complexity of the BK_Fac_Rec. We have the following observations. First, the initial pivot is chosen from C (line 9) and P is the vertex set that includes those vertices at which the branching steps are conducted (line 10). Second, although lines 15-17 would update P to reduce the number of produced branches, there exists the worst case in which P is not changed. This worst case is the same as the second algorithm proposed in [1], whose time complexity is $O(|C| \cdot (3.14)^{|C|/3})$ given a branch $B = (S, C, X)$. In summary, the time complexity of BK_Fac is $O(\delta n \cdot (3.14)^{\delta/3})$. □

REFERENCES

[1] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[2] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical computer science*, vol. 363, no. 1, pp. 28–42, 2006.

[3] K. A. Naudé, "Refined pivot selection for maximal clique enumeration in graphs," *Theoretical Computer Science*, vol. 613, pp. 28–37, 2016.

[4] Y. Xu, J. Cheng, A. W.-C. Fu, and Y. Bu, "Distributed maximal clique computation," in *2014 IEEE International Congress on Big Data*. IEEE, 2014, pp. 160–167.

[5] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," in *Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I 21*. Springer, 2010, pp. 403–414.

[6] ——, "Listing all maximal cliques in large sparse real-world graphs," *Journal of Experimental Algorithmics (JEA)*, vol. 18, pp. 3–1, 2013.

[7] Y. Li, Z. Shao, D. Yu, X. Liao, and H. Jin, "Fast maximal clique enumeration for real-world graphs," in *International Conference on Database Systems for Advanced Applications*. Springer, 2019, pp. 641–658.

[8] Y. Jin, B. Xiong, K. He, Y. Zhou, and Y. Zhou, "On fast enumeration of maximal cliques in large graphs," *Expert Systems with Applications*, vol. 187, p. 115915, 2022.