Theoretical Computer Science 407 (2008) 564–568

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Note

A note on the problem of reporting maximal cliques

F. Cazals [a,*], C. Karande [b]

[a] INRIA Sophia-Antipolis, France

[b] Georgia Institute of Technology, USA

ARTICLE INFO

Article history:
Received 29 August 2007
Received in revised form 30 January 2008
Accepted 11 May 2008
Communicated by W. Szpankowski

Keywords:
Maximal cliques
Output sensitive algorithms
Maximal common subgraphs
Enumeration problems

ABSTRACT

Reporting the maximal cliques of a graph is a fundamental problem arising in many areas. This note bridges the gap between three papers addressing this problem: the original paper of Bron–Kerbosh [C. Bron, J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, Communication of ACM 16 (9) (1973) 575–577], and two papers recently published in TCS, namely that of Tomita et al. [Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science 363 (1) (2006) 28–42], and that of Koch [I. Koch, Fundamental study: Enumerating all connected maximal common subgraphs in two graphs, Theoretical Computer Science 250 (1–2) (2001) 1–30]. In particular, we show that the strategy of Tomita et al. is a simple modification of the Bron–Kerbosch algorithm, based on an (un-exploited) observation raised in Koch's paper.

1. Reporting all maximal cliques of a graph

1.1. Maximal cliques and applications

Reporting the maximal cliques of a graph is a fundamental problem arising wherever graphs are the natural way to model objects and their interactions. Example applications can be found in network design and analysis, social sciences, or mathematical biology. Our interest in this problem comes from this latter vein, and in particular structural biology. More precisely, the detection of maximal cliques is fundamental for the question of *combinatorial partial shape matching* which consists, given two graphs, of reporting either all *Maximal Common Induced Subgraphs* —MCIS, or all *Maximal Common Edge Subgraphs* —MCES. See [9] for a general discussion of these problems. Example applications of clique detection algorithms for the identification of MCIS and MCES in computational structural biology are as follows: In [13], MCIS involving four atoms are used to define rigid motions for docking a ligand into a protein; also for docking, MCIS involving pairs of compatible atoms (each pair consisting of (donor, acceptor) of electrons) are sought in [8]; tertiary structure similarity is investigated from Maximum Common Induced Subgraph in [7]; strategies to report connected common sub-structures rather than general common sub-structures are investigated in [10] [1]; the detection of cliques for structure prediction is carried out in [14].

1.2. Maximum weight clique, maximal cliques, enumeration problems

Reporting *all* maximal cliques of a graph should be not confused with the problem of finding the *maximum weight clique*, a well studied problem of combinatorial optimization for which a vast literature exists [2]. As opposed to the maximum

* Corresponding address: INRIA Sophia-Antipolis, Geometrica group, 2004 route des Lucioles, 06902 Sophia-Antipolis, France. Tel.: +33 4 92 38 71 88.
E-mail addresses: Frederic.Cazals@inria.fr (F. Cazals), ckarande@cc.gatech.edu (C. Karande).

[1] In [10] and [9], the algorithm reporting the so-called maximal c-cliques is incorrect though: see [5] for the corrected version.

weight clique problem, the output of maximal clique enumeration algorithms may be exponentially sized [11], so that an algorithm with provably good running time w.r.t. the input size is not possible. However, even with this consideration, any algorithm reporting all maximal cliques should, in some sense, be output sensitive.

Up to 1999, a comprehensive bibliography of clique enumeration algorithms can be found in [2, Section 5.1]. General algorithms – i.e. algorithms not geared towards specific graphs [11] – fall into two veins. First, one finds the Bron–Kerbosch (BK) algorithms [3] and its sibling [1], a list to which one should prepend the recent variant of BK by I. Koch [9], motivated by the insights of [1]. We call this family of algorithms *greedy* algorithms. Alternatively, one may resort to output-sensitive algorithms, either that of Tsukiyama et al. [15], or the more recent alternative strategy based on matrix multiplication [12].

Theoretically, an important measure of any enumeration algorithm is its output sensitivity, and in this regard, the first result was published by Tsukiyama et al. [15]. Results on output sensitivity are proved by bounding (i) the time to report the first clique (ii) the time required to report any additional new clique. To the best of our knowledge, state of-the-art results in this direction have been established in [12], where it is proved that all maximal cliques of a n vertices graph can be enumerated using n^2 space, and with time delay $O(M(n))$ with $M(n)$ the cost of multiplying two $n \times n$ matrices.

The practical efficiency of algorithms is a different topic, and until recently, all papers acknowledged BK algorithm as the best one in practice [4], especially when I. Koch's pivot selection strategy is used [9]. Of particular interest is the comparison of greedy versus output sensitive algorithm. As shown in [16], greedy algorithms outperform provably good output sensitive ones [12].

1.3. Contribution and paper overview

The goal of this note is to bridge the gap between [3], [9] and [16]:
– in [3], Bron and Kerbosh report two algorithms, `Version 1` and `Version 2`, the second one being an optimization of the first one based on *fixed points* or *pivots*;
– in [9], a number of pivot selection strategies are investigated, together with an interesting observation which is left aside, while this observation actually provides a work-around to tricky cases reported by Koch herself;
– in [16], Tomita et al. propose an algorithm which is exactly Koch's algorithm, with a subtle variation which is the observation left aside by Koch. In passing, notice we re-discovered this algorithm independently from [16], as testified by the 2005 version of the technical report [6].

To bridge this gap while keeping the paper self-contained, Section 2 is organized as follows: first, `Version 1` of the Bron–Kerbosch algorithm is recalled; second, the generalizations of `Version 2` by Koch [9] are recalled; third, the connection with [16] is presented.

1.4. Notations

The size of a finite set S is denoted $| S |$. We shall denote the graph $G = (V[G], E[G])$, with $| V[G] | = n$. Given a node $u \in G$, $N[u]$ denotes the neighbors of u, i.e. $N[u] = \{v \mid (u, v) \in E[G]\}$.

2. Pivot selection algorithms in the lineage of the Bron–Kerbosch algorithm

2.1. The Bron–Kerbosch algorithm

To begin with, we recall the standard Bron–Kerbosch (BK) Algorithm [3]— Fig. 1. The algorithm maintains three disjoint sets of nodes R, P, X. Set R stands for the currently growing clique; set P stands for prospective nodes which are connected to all nodes in R and using which R can be expanded; set X contains nodes already processed i.e. nodes which were previously in P and hence all maximal cliques containing them have already been reported. An important invariant is that all nodes which are connected to every node of R are either in P or X. Purpose of X is to avoid reporting the same maximal clique multiple times, through the update $P = P - \{u_i\}$. To avoid reporting cliques which are not maximal, the algorithm checks whether set X is empty : if X is non-empty, the nodes in X may be added to R, but this would yield previously found maximal cliques.

Naturally, the standard BK algorithm exhibits very poor output sensitivity in the worst case. Since the algorithm knows that a clique is non-maximal only when the clique has been fully formed in R, all cliques are enumerated – possibly exponential number of them in the size of output – the number of maximal cliques. To get around this problem, a variant of the algorithm, called `Version 2`, is presented in [3]. We now present this variant and put it in perspective as in [9].

2.2. Pivot selection heuristics

Using a pivot. Based on the observations raised in [3] and [1], which aim at reducing the size of the recursion tree of Bron–Kerbosch, a general (i.e. template) heuristic for BK-like algorithms is analyzed in [9].[2] The heuristic is based on the

[2] The node sets manipulated by the BK algorithm in Koch's paper are denoted C, P, S. Because additional node sets are required to report special cliques [5], we changed the notations to R, P, X.

```
Algorithm call: BK($\emptyset, V[G], \emptyset$).

BK($R, P, X$)
if $P = \emptyset$ and $X = \emptyset$ then
    Report $R$ as a maximal clique
else
    Assume $P = \{u_1, u_2, \dots, u_k\}$
    for $i \leftarrow 1$ to $k$ do
        $P = P - \{u_i\}$
        $R_{new} = R \cup \{u_i\}$
        $P_{new} = P \cap N[u_i]$
        $X_{new} = X \cap N[u_i]$
        BK($R_{new}, P_{new}, X_{new}$)
        $X = X \cup \{u_i\}$
```

Fig. 1. Bron–Kerbosch Algorithm: `Version 1` in [3].

```
Algorithm call: IK_∗($\emptyset, V[G], \emptyset$).

IK_∗($R, P, X$)
if $P = \emptyset$ and $X = \emptyset$ then
    Report $R$ as a maximal clique
else
    Let $u_p$ be the pivot vertex //see text
    Assume $P = \{u_1, u_2, \dots, u_k\}$
    for $i \leftarrow 1$ to $k$ do
        if $u_i$ is not a neighbor of $u_p$ then
            $P = P - \{u_i\}$
            $R_{new} = R \cup \{u_i\}$
            $P_{new} = P \cap N[u_i]$
            $X_{new} = X \cap N[u_i]$
            IK_∗($R_{new}, P_{new}, X_{new}$)
            $X = X \cup \{u_i\}$
```

Fig. 2. Bron–Kerbosch Algorithm with pivot selection. The * in IK_* stands for the multiple options in choosing the pivot.

identification and elimination of equal sub-trees appearing in different branches of the recursion tree which lead to the formation of non-maximal cliques. As indicated in the statement of [9, Thm **3.4**], this identification is performed via the choice of a pivoting node $u_p \in P$. Given the pivot u_p, we shall partition P as $P = P^+ \cup P^-$ with: $P^- = P \cap N[u_p]$, the neighbors of the pivot in P and $P^+ = P - P^-$, the remaining nodes which are not neighbors of u_p. (Note that by definition, a node cannot have an edge to itself, hence u_p belongs to P^+.) Observe that no subset of P^- can be appended to R by itself to form a maximal clique, because such a clique can always be extended by u_p. In other words, any maximal clique containing R must also contain at least one node $u^+ \in P^+$, and any such clique will be discovered in the sub-tree of recursive call in which u^+ is appended to R. Hence, nodes from P^- can be skipped from the main `for` loop of the BK algorithm. The pseudo-code of this algorithm (denoted IK) is presented in Fig. 2. This code is a template in the sense that the precise way in which the pivot is chosen is not described, as conveyed by IK_*.

Alternatives investigated in [9]. In Koch's paper, two classes of selection strategies for the pivot are reported:
— In Thm **3.4**, the pivot is picked from set P. Practically, this leads to two options which are experimentally investigated: the first one (IK_RP) consists of choosing the pivot at random in P; the second one (IK_GP) consists of choosing as pivot the vertex with largest degree in P. For random graphs, it is shown experimentally that IK_GP outperforms IK_RP. However, for a particular class of graphs discussed below, the performance of greedy selection can be as poor or even poorer than randomized selection.
— Thm **3.6** observes that the pivot can also be picked from set $P \cup X$. But this observation is not used algorithmically.

2.3. Improved pivot selection and [16]

Fixing the case of $K_n \cup K_{1,n}$. The problem reported in [9, Thm **3.5**, Fig. 7] is concerned with a graph of $2n + 1$ nodes consisting of two connected components, namely K_n and $K_{1,n}$ —reproduced on Fig. 3. As depicted on that figure, if the pivot is selected

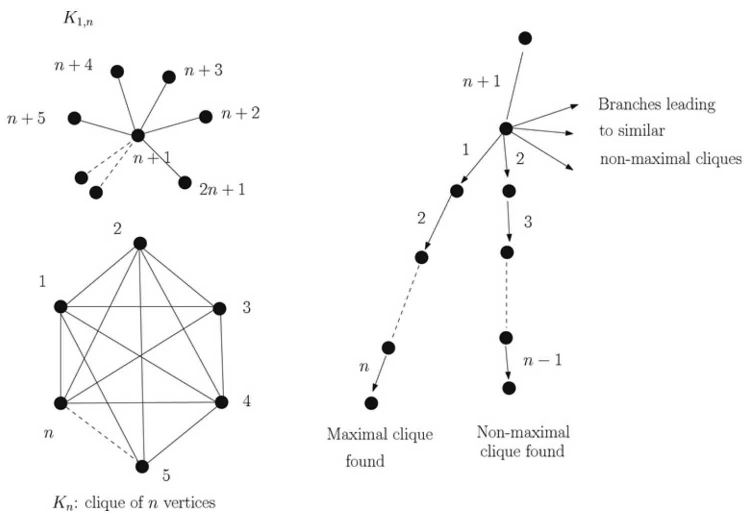


Fig. 3. Calling IK_GP on $K_{1,n} \cup K_n$ yields a recursion tree of size $\Theta(n^2)$.

only from P, at every recursive call with non-empty P, the main loop in the algorithm is run at least once (exactly once in the situation in Fig. 3) since the pivot is a non-neighbor of itself. Being able to select the pivot from X removes this anomaly, since all the vertices in P can be skipped if they are neighbors to the pivot. More precisely, the pivot is the node u_p from $P \cup X$ maximizing the cardinality of the set $P \cap N[u_p]$. This strategy consists of integrating Thm **3.6** from [9] as pivot selection in algorithm IK_*. We denote the corresponding algorithm IK_GPX, and make the following observation:

Observation 1. *Consider a recursive call of the IK_GPX algorithm, such that the set P consists of the graph $K_n \cup K_{1,n}$. The corresponding recursion tree has size $O(n)$.*

The reason why this simple trick works is also quite intuitive — although the disconnected large degree pivot is selected at the topmost level even in this case, at least one node from the n-node clique appears in the set X for all the branches leading to non-maximal cliques (see Fig. 3), in subsequent recursive calls. The pivot then is selected to be such a node from X, which is a neighbor to all the nodes from the n-clique. All the branches leading to non-maximal cliques are thus cut down at the next recursion level, because the neighbors of a pivot are not appended to R.

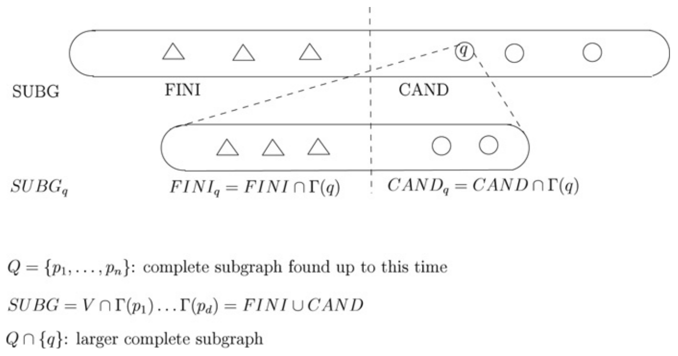
Bridging the gap with the `Version 2` *algorithm of [3], and the* `Cliques` *algorithm [16].* We now explain why this pivoting strategy is identical to the algorithm developed in [16]. As explained on Fig. 4 —adapted from [16, Fig. 1], the set *SUBG* consists of the nodes of the graph which exhibit full connectivity to the clique Q found so far. (Set Q is a global variable used to store the maximal clique under construction, and corresponds to parameter R of algorithm IK_*.) With our notations, set *SUBG* is exactly $P \cup X$. Then, the vertex u of algorithm `Cliques` —[16, Fig. 2], is chosen in *SUBG*, that is $P \cup X$. Finally, the `while` loop consists of iterating on the non neighbors of the pivot u.

It should be noticed that the pivoting strategy from $P \cup X$ was already essentially present in [3], where the pivot is termed *fixed point*, while the sets P and X are called *candidates* and *not*. One indeed reads ≪ *the fixed point is taken either from not or from the original candidates, whichever point yields the lowest counter value after the first addition to not*. ≫ This strategy is that just described, excepted that the pivot selection from $P \cup X$ is deferred after the first insertion into X.

Incidence on global performances. The difficulty faced by Koch's algorithm for the graph $K_n \cup K_{1,n}$ may be faced in a more general setting in the course of a recursion, namely whenever any node in P has a number of neighbors larger than the size of a clique to be discovered. In such a case, the pivot selection from $P \cup X$ avoids the (local) quadratic trap. This situation may occur a number of times in the course of a recursion, so that one expects IK_GPX to outperform IK_GP and IK_RP. This observation is indeed borne out by the experiments reported in [6], where a strategy based on so-called dominated nodes is also explored. Interestingly, it is also reported in [16] that the same algorithm outperforms provably good output sensitive ones [12].

3. Conclusion

This note bridges the gap between three papers addressing the problem of reporting maximal cliques, namely Bron–Kerbosh [3], Tomita et al. [16], and Koch [9]. More precisely, we show that these algorithms resort to the same strategy which consists of pruning the search tree, resorting to well chosen pivoting nodes. As the resulting algorithm has been shown to be worst-case optimal by Tomita et al. for graphs yielding $O(3^{n/3})$ cliques, two types of question remain. On the one hand, the question of proving output-sensitive properties of this algorithm for non worst-case inputs remains open. On the other hand, one may develop algorithms with enhanced output-sensitive properties.


SUBG
FINI
CAND
q
$SUBG_q$
$FINI_q = FINI \cap \Gamma(q)$
$CAND_q = CAND \cap \Gamma(q)$

$Q = \{p_1, \ldots, p_n\}$: complete subgraph found up to this time

$SUBG = V \cap \Gamma(p_1) \ldots \Gamma(p_d) = FINI \cup CAND$

$Q \cap \{q\}$: larger complete subgraph

Fig. 4. The expansion procedure of Tomita et al.: Fig. 1 in [16]. The pivot is selected from set *SUBG* (set $P \cup X$ with our notations). This set exhibits full connectivity with the clique Q (set R with our notations) grown by adding nodes from *CAND* (set P with our notations).

Acknowledgment

C. Karande acknowledges the support of the INRIA internship program.

References

[1] E.A. Akkoyunlu, The enumeration of maximal cliques of large graphs, SIAM Journal on Computing 2 (1) (1973).
[2] I. Bomze, M. Budinich, P. Pardalos, M. Pelillo, The maximum clique problem, in: D.-Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, vol. 4, Kluwer, 1999.
[3] C. Bron, J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, Communication of ACM 16 (9) (1973) 575–577.
[4] D. Braum, Finding all maximal cliques of a family of induced subgraphs, Technical Report 03-53, Konrad-Zuse-Zentrum, 2003.
[5] F. Cazals, C. Karande, An algorithm for reporting maximal c-cliques, Theoretical Computer Science 349 (3) (2005) 484–490.
[6] F. Cazals, C. Karande, Reporting maximal cliques: New insights, Rapport de recherche 5615, INRIA, 2007. revision of the 2005 version.
[7] H.M. Grindley, P.J. Artymiuk, D.W. Rice, P. Willett, Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm, J. Mol. Biol. 229 (1993).
[8] E.J. Gardiner, P. Willett, P.J. Artymiuk, Graph-theoretic techniques for macromolecular docking, Journal of Chemical Information and Computer Science 40 (2000).
[9] I. Koch, Fundamental study: Enumerating all connected maximal common subgraphs in two graphs, Theoretical Computer Science 250 (1–2) (2001) 1–30.
[10] I. Koch, E. Wanke, T. Lengauer, An algorithm for finding maximal common subtopologies in a set of protein structures, Journal of Computational Biology 3 (2) (1996).
[11] J.W. Moon, L. Moser, On cliques in graphs, Israel Journal of Mathematics 3 (1965).
[12] K. Makino, T. Uno, New Algorithms for Enumerating All Maximal Cliques, in: SWAT—LNCS, vol. 3111, Springer, 2004.
[13] B.K. Stoichet, D.L. Bodian, I.D. Kuntz, Molecular docking using shape descriptors, Journal of Computational Chemistry 13 (3) (1992).
[14] R. Samudrala, J. Moult, A graph-theoretic algorithm for comparative modelling of protein structure, Journal of Molecular Biology 279 (1998).
[15] S. Tsukiyama, M. Ide, H. Ariyoshi, I. Shirakawa, A new algorithm for generating all the maximal independent sets, SIAM Journal on Computing 6 (1977) 505–517.
[16] E. Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science 363 (1) (2006) 28–42.