Computers and Operations Research 92 (2018) 37–46

Contents lists available at ScienceDirect

Computers and Operations Research

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

Efficiently enumerating all maximal cliques with bit-parallelism

Pablo San Segundo [a,*], Jorge Artieda [a], Darren Strash [b]

[a] Centre for Automation and Robotics (UPM-CSIC), Jose Gutiérrez Abascal 2, Madrid 28006, Spain
[b] Department of Computer Science, Colgate University, Hamilton NY, USA

ARTICLE INFO

Article history:
Received 10 March 2017
Revised 5 December 2017
Accepted 6 December 2017
Available online 9 December 2017

Keywords:
Maximal clique
Bitstring
Branch-and-bound
Subgraph enumeration
Combinatorial optimization

ABSTRACT

The maximal clique enumeration (MCE) problem has numerous applications in biology, chemistry, sociology, and graph modeling. Though this problem is well studied, most current research focuses on finding solutions in large sparse graphs or very dense graphs, while sacrificing efficiency on the most difficult medium-density benchmark instances that are representative of data sets often encountered in practice. We show that techniques that have been successfully applied to the maximum clique problem give significant speed gains over the state-of-the-art MCE algorithms on these instances. Specifically, we show that a simple greedy pivot selection based on a fixed maximum-degree first ordering of vertices, when combined with bit-parallelism, performs consistently better than the theoretical worst-case optimal pivoting of the state-of-the-art algorithms of Tomita et al. [Theoretical Computer Science, 2006] and Naudé [Theoretical Computer Science, 2016].

Experiments show that our algorithm is faster than the worst-case optimal algorithm of Tomita et al. on 60 out of 74 standard structured and random benchmark instances: we solve 48 instances 1.2 to 2.2 times faster, and solve the remaining 12 instances 3.6 to 47.6 times faster. We also see consistent speed improvements over the algorithm of Naudé: solving 61 instances 1.2 to 2.4 times faster. To the best of our knowledge, we are the first to achieve such speed-ups compared to these state-of-the-art algorithms on these standard benchmarks.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

The *maximal clique enumeration* (MCE) problem—the problem of enumerating all maximal cliques of a given graph—has numerous applications spanning many disciplines (Augustson and Minker, 1970; Gardiner et al., 2000; Horaud and Skordas, 1989). Unlike the NP-hard maximum clique problem (MCP) (Garey and Johnson, 1990; Karp, 1972), the MCE problem is known to require exponential time in the worst case, since there may be an exponential number of maximal cliques to enumerate. For an n-vertex graph, there may be $\Theta(3^{n/3})$ maximal cliques, known as the Moon–Moser bound (Moon and Moser, 1965), and therefore any algorithm that enumerates all maximal cliques must use at least this amount of time in the worst case. Interestingly, not only does there exist an algorithm that runs in worst-case optimal $\Theta(3^{n/3})$ time, that of Tomita et al. (2006), but it is also among the fastest algorithms in practice. Eppstein et al. (2013) further tightened these bounds for the case of graphs with low *degeneracy* (Lick and White, 1970), the smallest value d such that every induced subgraph of G has a vertex of degree at most d. They showed that graphs with degeneracy d have $\Omega(d(n-d)3^{d/3})$ maximal cliques, and further give an algorithm to enumerate all maximal cliques in time $O(d(n-d)3^{d/3})$, which matches the worst-case output size. Moreover, they showed that their method is efficient in practice on real-world complex networks, which typically have low degeneracy.

These algorithms, as well as many other efficient algorithms, are derived from the Bron-Kerbosch algorithm—which maintains both a currently growing clique and a set of already examined vertices throughout recursive backtracking search, only reporting a clique when it is found to be maximal (Bron and Kerbosch, 1973). However, a separate class of theoretically-efficient algorithms, those with bounded *time delay* (the time between reported cliques), exist for the MCE problem, which use the reverse search technique of Avis and Fukuda (1996). Tsukiyama et al. (1977) were the first to give a bounded time delay algorithm for this problem, giving an algorithm with delay $O(nm)$ for graphs with m edges. Chiba and Nishizeki (1985) improved this result for graphs with arboricity a, a sparsity measure, giving a $O(am)$-delay algorithm. Makino and Uno (2004) removed the linear dependence on m, reducing the delay to $O(\Delta^4)$, where Δ is the maximum degree of G; however, their technique uses quadratic space. Chang et al. (2013) further showed how to reduce the preprocessing time of Makino and Uno from quadratic to linear, while giving a tighter delay of $O(\Delta h^3)$, where h is the h-index of the graph.

* Corresponding author.
E-mail addresses: pablo.sansegundo@upm.es (P. San Segundo), dstrash@cs.colgate.edu (D. Strash).

https://doi.org/10.1016/j.cor.2017.12.006
0305-0548/© 2017 Elsevier Ltd. All rights reserved.

Finally, Conte et al. (2016) gave the first bounded delay maximal clique enumeration algorithm with sublinear extra space, with delay $O(qd(\Delta + qd)\text{polylog}(n+m))$, where q is the size of a maximum clique.

Though these bounded time delay algorithms are theoretically efficient, Bron–Kerbosch-derived algorithms are much faster in practice. As noted by Conte et al. (2016), their algorithm (which is at present the fastest bounded time delay algorithm) is 3.7 times slower than the Bron–Kerbosch-derived algorithm by Eppstein et al. (2013) on sparse graphs, which is itself slower than the algorithm by Tomita et al. (2006) on dense and medium-density instances that we consider here. Even though the algorithm by Tomita et al. (2006) has worst-case exponential time, repeated experiments show that it is fast on a variety of benchmark instances (Cazals and Karande, 2006; Eppstein et al., 2013; Koch, 2001; Naudé, 2016; Tomita et al., 2006). At the time of writing, we are unaware of any algorithms that achieve significant speedups over this algorithm, though moderate speedups are possible on graphs that are either very sparse (Eppstein et al., 2013) or very dense (Naudé, 2016).

For sparse graphs, the algorithm by Eppstein et al. (2013) rivals that of Tomita et al. (2006) while only consuming space linear in the size of the graph, whereas the algorithm of Tomita et al. requires quadratic space to store an adjacency matrix. Dasari et al. (2014) further improved this result by factors of 2-4x using bit-parallelism, though the main algorithm remains unchanged. For larger instances, external memory algorithms have been developed which take advantage of the property that real-world sparse graphs typically have small induced subgraphs that can fit into memory (Cheng et al., 2012). For even larger instances, algorithms have been implemented in the MapReduce framework (Wu et al., 2009).

In the case of dense graphs, researchers have looked at different strategies for pruning search. In particular, Cazals and Karande (2006) showed that detecting and removing dominated vertices is much faster than the traditional pivoting method commonly used in the fastest algorithms, such as that of Tomita et al. (2006), when graphs are dense. This is because the time to pick a pivot can be very expensive when there are many edges. Naudé (2016) investigated the pivot computation set, and showed that it is possible to break out of pivot computation early under certain conditions, and still maintain a worst-case optimal running time of $O(3^{n/3})$. In Naudé's experiments on small random graphs (with 180 vertices or less), his algorithm is at most 1.56 times faster than that of Tomita et al. (Tomita et al., 2006) on graphs with a high edge density of 0.8; on the other hand, on graphs with a lower edge density of 0.4, the method gives a modest speed up of at most 13%.

Surprisingly, recent algorithms have focused only on sparse and high density graphs, and have not considered performance on medium density graphs, which are representative of many real-world instances, and where pruning techniques based on structure, different from the theoretical-optimal pivoting, are much less effective. In particular, the benchmarks from the second DIMACS challenge (Johnson and Trick, 1996) and BHOSLIB (Xu, 2004), have medium density and are among the most difficult sets in practice. As far as we are aware, no algorithm gives significant improvement over the algorithm of Tomita et al. (2006) for these instances.

1.1. Our contribution

We provide a new algorithm for the MCE problem, and show that it consistently outperforms the state-of-the-art algorithms of Tomita et al. (2006) and Naudé (2016) on benchmark graphs that are representative of difficult instances. This work is inspired by a number of techniques that have been described for branch and bound maximum clique solvers, such as employing an initial ordering of nodes (Carraghan and Pardalos, 1990; San Segundo et al., 2013, 2011; Tomita and Kameda, 2006; Tomita and Seki, 2003; Tomita et al., 2010), the use of bit-parallelism (San Segundo et al., 2013, 2011) and keeping a fixed vertex ordering throughout recursion (San Segundo et al., 2013, 2011). Note that these solvers differ from any MCE solver in the fact that they also prune enumerable cliques in the search tree when they cannot possibly improve the incumbent solution.

While leading MCP solvers (as well as the MCE solver by Eppstein et al. (2013), and the improved bitstring encoding proposed by Dasari et al. (2014)) attempt to quickly reduce subproblem size by branching on vertices initially (at the root) with low degree (following a *degeneracy ordering*), we evaluate vertices with high degree according to a *maximum-degree-first ordering*. This ordering helps us address one of the main challenges when applying bit-parallelism to state-of-the-art MCE solvers: efficient *pivot selection*, for which most algorithms must enumerate vertices to find a high degree vertex in the current subproblem (Cazals and Karande, 2006, 2008; Eppstein et al., 2013; Koch, 2001; Naudé, 2016; Tomita et al., 2006). Moreover, enumeration of items is a well-known bottleneck of bitstrings.

With our proposed initial ordering, we efficiently perform a simple *greedy* pivot selection strategy, which allows us to pivot without enumeration. We likewise branch on vertices according to the initial ordering, since they are likely to have high degree in the subproblem being evaluated. Although this strategy increases the size of the search space on average when compared to the theoretical algorithm of Tomita et al. (2006), our algorithm outperforms state-of-the-art solvers on 60 out of 74 of instances tested, which we attribute to the speed of greedy pivot selection, when combined with a bitstring representation. In contrast, a direct bit-parallel implementation of the state-of-the-art solver by Tomita et al. (2006) is slower than the original implementation on most instances.

1.2. Organization

The remainder of our paper is organized as follows: in Section 2 we cover useful definitions and other preliminaries and in Section 3 we describe variations of the Bron-Kerbosch algorithm. Our contributions appear in Section 4, where we describe our new enumeration algorithm. We then present our experimental results in Section 6, and conclude and give ideas for future work in Section 7.

2. Preliminaries

We work with a simple undirected graph $G = (V, E)$, which consists of a finite set of vertices $V = \{1, 2, \ldots, n\}$ and a finite set of edges $E \subseteq V \times V$ made up of pairs of distinct vertices. Two vertices u and v are said to be adjacent (or neighbors) if $(u, v) \in E$. The neighborhood of a vertex v, denoted $N(v)$ (or $N_G(v)$ when the graph needs to be mentioned explicitly), is defined as $N(v) = \{u \in V \mid (u, v) \in E\}$. We denote the *degree* of a vertex v by $\deg(v)$, and denote the *maximum degree* of G by $\Delta(G) = \max_{v \in V} \deg(v)$.

A *clique*, also called a complete subgraph, is a set $K \subseteq V$ of pairwise adjacent vertices. A clique K is said to be *maximal* if there is no vertex in $V \setminus K$ that is adjacent to all vertices in K. Note that this definition is different from a *maximum clique*, which is a clique of maximum cardinality $\omega(G)$.

Finally, we also consider the following terminology and definitions for vertex orderings. We define a vertex ordering to be a sequence $v_1, v_2, \ldots, v_n$ of the vertices in V, where v_i is said to be in position i, and further define a permutation $\phi : V \to \{1, 2, \ldots, n\}$ that maps each vertex $v_i \in V$ to its position i; that is $\phi(v_i) = i$. The

Download English Version:

https://daneshyari.com/en/article/6892683

Download Persian Version:

https://daneshyari.com/article/6892683

Daneshyari.com