Instances for the Maximum Clique Problem with Hardness Guarantees

Victor Campos[1], **Renato Carmo**[2], **Rodrigo Nogueira**[1]

[1]ParGO Group, Universidade Federal do Ceará (UFC)

[2]Departamento de Informática da Universidade Federal do Paraná (UFPR)

victoitor@ufc.br, renato@inf.ufpr.br, nogrrodrigo@alu.ufc.br

Abstract. *Branch and bound algorithms which use (an estimate on) the chromatic number as the bounding function are reported among the best known exact algorithms for the maximum clique problem. Their average running time is known to be quasi-polynomial and describing families of instances which induce exponential running time for these algorithms is a non-trivial issue. We describe two such families of graphs. We prove that graphs in the first family induce asymptotically higher running time than the average for algorithms whose running time is* $\Omega(c(G))$ *where* $c(G)$ *is the number of cliques in* G*. We also prove that graphs in the second family induce exponential running time for branch and bound algorithms which use the chromatic number as the bounding function.*

1. Introduction

For a graph G, let $\omega(G)$ denote the maximum size of a clique in G. The Maximum Clique problem (MC) is the problem of finding a maximum clique in a given graph G. The decision version of MC, denoted by CLIQUE, aims to decide whether $\omega(G) \geq k$ for a given graph G and an integer k. CLIQUE is a fundamental NP-complete problem [Karp, 1972] which is also $W[1]$-complete under the natural parameterization over k [Downey and Fellows, 1995]. Besides being NP-hard, MC is also inapproximable to within a factor of $n^{1-\varepsilon}$, for any $\varepsilon > 0$ (unless $P = NP$) [Zuckerman, 2006].

Despite these rather disheartening theoretical facts, several exact algorithms for MC are reported as solving instances of practical interest and considerable size in various application domains quite well (for more details, we refer the reader to a survey [Bomze et al., 1999] and a newer result [San Segundo et al., 2016].) Typically, the performance of these algorithms is evaluated experimentally: standard sets of instances (such as those from the DIMACS Second Implementation Challenge [Johnson and Trick, 1996]) as well as random graphs are used, the running time and other execution parameters are reported and then compared to other published data. The experimental results reported in all these cases seem surprising in the light of the theoretical results mentioned above. This apparent contradiction was investigated in [Züge and Carmo, 2018], which shows that branch and bound algorithms for MC have quasi-polynomial average running time (under $\mathcal{G}(n, p)$for fixed p.) This leaves open the question of whether these (and other) algorithms do have subexponential running time in the worst case or not. We approach this question by exhibiting instances candidate to inducing exponential running-time for these algorithms.

A *clique subinstance* for a graph G, which we call simply a *subinstance* for G, is a pair (Q, K) of disjoint subsets of $V(G)$ where Q is a clique and every vertex of K

is adjacent to every vertex of Q. Each subinstance (Q, K) asks for the maximum clique C in G such that $Q \subseteq C \subseteq Q \cup K$. Note that the instance G of MC corresponds to the subinstance $(\emptyset, V(G))$. When a subinstance (Q, K) for G is not solved, a vertex v in K, called the *pivot*, is chosen and this subinstance is broken into two: $(Q \cup \{v\}, K \cap N(v))$ which considers all cliques containing v and $(Q, K - v)$ which considers all cliques which do not contain v. We call a branch and bound algorithm for MC *standard* if it breaks subinstances using pivot vertices in this way. A *clique search tree* T for a graph G is a binary tree whose nodes are subinstances of G such that: the root of T is the subinstance $(\emptyset, V(G))$; a subinstance (Q, K) is a leaf of T if, and only if, $K = \emptyset$; and, for every non-leaf subinstance (Q, K), there is a vertex $v \in K$ such that its children are $(Q \cup v, K \cap N(v))$ and $(Q, K - v)$. We say that an execution of a standard branch and bound algorithm for MC on G is *contained* in a clique search tree T of G if it considers a set of clique subinstances which induces a connected subgraph of T containing the root and every choice of pivot in the algorithm is the same as in T. Using this notation, the following can be obtained from [Züge and Carmo, 2018].

Proposition 1. *Let G be a graph and C be the set of all its cliques. If T is a clique search tree of G, then T has precisely $2|C| - 1$ nodes. Furthermore, each execution of any standard branch and bound algorithm for MC on G is contained in some clique search tree of G.*

The part of the analysis in [Züge and Carmo, 2018] which concerns us stems from the fact that almost every graph (under $\mathcal{G}(n, p)$for fixed p) has a quasi-polynomial number of cliques. Consequently, any standard branch and bound algorithm for MC will have a quasi-polynomial running time for almost any instance (as long as it takes no more than quasi-polynomial time for each subinstance.) Our first family of hard instances is one that provides graphs with an exponential number of cliques. This family may be useful in discerning whether the running time of a particular algorithm is proportional to the number of cliques in the graph or if it really reduces substantially the search space through the heuristics incorporated in it.

2. Instances with Exponential Number of Cliques

The following construction is based on a reduction from the Vertex Coloring problem to MC. It can be derived directly from [Campêlo et al., 2008] and a similar result has been observed in [Cornaz and Jost, 2008]. Here a *coloring* means a partition of the vertices into (unlabeled) stable sets.

Lemma 2. *Let G be a graph with n vertices and $\overline{m}$ non-edges. We can build a graph G^* with $\overline{m}$ vertices together with a bijection f from cliques of G^* to colorings of G such that, if C is a clique in G^*, then $f(C)$ is a coloring of G with $n - |C|$ colors.*

We observe that an interesting aspect of this reduction is that the number of cliques in G^* is equal to the number of colorings of G.

Theorem 3. *For any $0 \leq \alpha < 1/3$ and $n \in \mathbb{N}$, if $m \leq \binom{n}{2}(1 - n^{-\alpha})$, then the expected number of colorings of $G \sim \mathcal{G}(n, m)$ is $n^{\Theta(n)}$.*

Using the construction from Lemma 2 on $\mathcal{G}(n, m)$, for carefully chosen values of n and m, we get the following.

Corollary 4. *For any* $0 \leq \varepsilon < 1/10$ *and* $n \in \mathbb{N}$*, there is a random process to build a graph with n vertices and whose expected number of cliques is* $n^{\Theta\left(n^{3/5-\varepsilon}\right)}$.

Let p be a constant, $G \sim \mathcal{G}(n, p)$ and G^* be obtained from G using the construction of Lemma 2 with $N = |V(G^*)|$. Based on a much stronger result from [Pittel, 1982], [Züge and Carmo, 2018] point out that the number of cliques in G is $n^{\Theta(\log n)}$ with high probability. In contrast, Theorem 3 tells us that the expected number of cliques in G^* is $N^{\Theta(\sqrt{N})}$. The problems of finding a maximum clique and a minimum vertex coloring were considered in the second DIMACS Implementation Challenge [Johnson and Trick, 1996] and coloring was considered to be much harder to solve than CLIQUE. Since solving MC in G^* is equivalent to solving a coloring problem on G, this transformation gives a clue on why $\mathcal{G}(n, p)$ graphs are harder instances for minimum coloring than for maximum clique.

3. Instances with Persistent Chromatic Gap

Let $\chi(G)$ denote the chromatic number of a graph G. We say that a standard branch and bound algorithm is *χ-bounded* if the only way to discard a subinstance (Q, K) of G is by comparing a known clique in G to an upper bound on $\chi(G[Q \cup K])$. Some of the best algorithms for MC are indeed standard and χ-bounded. Note that if $K = \emptyset$ in a subinstance (Q, K), then we have $\chi(G[Q \cup K]) = |Q| \leq \omega(G)$. For a clique search tree T of G, the *χ-pruned* subtree of T, denoted by T_χ, is the unique minimal subtree of T that contains the root and whose leaves are subinstances (Q, K) such that $\chi(G[Q \cup K]) \leq \omega(G)$.

Proposition 5. *For any execution $\mathcal{E}$ of a standard χ-bounded branch and bound algorithm for MC on G, there exists a clique search tree T of G such that $\mathcal{E}$ is contained in T and the subinstances considered in $\mathcal{E}$ contain $V(T_\chi)$.*

Let n be a multiple of 5. We define L_n to be the n-vertex graph obtained from the join of $n/5$ copies of C_5. Although not using the notation introduced here, [Lavnikevich, 2013] essentially proves that the χ-pruned subtree of every clique search tree of L_n has at least $2^{n/5}$ nodes. By using Proposition 5, we get that L_n induces exponential running-time for any χ-bounded standard branch and bound algorithm. These instances, however, can be efficiently solved through a rather simple pre-processing algorithm (discussed bellow.) Our second family of hard instances also induces exponential running-time for these algorithms and is immune to this pre-processing.

We know that $\omega(G)$ is the maximum clique number over the components of G. Moreover, if G is the join of two other graphs, then the clique number of G is the sum of the clique number of these graphs. These two observations can be used to pre-process a graph G by constructing $\overline{G}$ and checking the connectivity of both. If the graph is separated into smaller subgraphs, this procedure can be iterated until every subgraph is both connected and has a connected complement and this procedure can be done in polynomial time. We note that finding $\omega(L_n)$ becomes easy by using this pre-processing technique as it reduces to the problem of finding the maximum clique in the $n/5$ copies of C_5. This observation leads us to look for instances which are connected and whose complements are also connected. Our second construction is based on the following result.

Theorem 6. *If G is a spanning subgraph of L_n with no stable set of size three and $\delta(G) \geq n - 1 - d$, then the χ-pruned subtree of every clique search tree of G has at least $2^{n/(5d)+1} - 1$ nodes.*

We note that L_n is $(n-3)$-regular and, if G is a spanning subgraph of L_n with $\delta(G) \geq n-3$, then G is equal to L_n. Since the complement of L_n is disconnected, we are interested in Theorem 6 when $d \geq 3$. We also note that when $d = 3$, there are many strategies to make the random instance.

Corollary 7. *For any $n \in \mathbb{N}$, there is a random process to build a connected graph G with n vertices whose complement is connected and such that the χ-pruned subtree of every clique search tree of G has $\Omega(2^{n/15})$ nodes.*

Finally, we thank the anonymous reviewers for the suggestions that helped improve and clarify this text.

References

Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. (1999). The maximum clique problem. In Du, D.-Z. and Pardalos, P. M., editors, *Handbook of Combinatorial Optimization: Supplement Volume A*, volume 4, pages 1–74. Springer, Boston, MA.

Campêlo, M., Campos, V., and Corrêa, R. (2008). On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097–1111.

Cornaz, D. and Jost, V. (2008). A one-to-one correspondence between colorings and stable sets. *Operations Research Letters*, 36(6):673–676.

Downey, R. and Fellows, M. (1995). Parameterized computational feasibility. In Clote, P. and Remmel, J. B., editors, *Feasible Mathematics II*, pages 219–244, Boston, MA. Birkhäuser Boston.

Johnson, D. and Trick, M., editors (1996). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, October 11–13, 1993*, volume 26. American Mathematical Society, Boston, MA, USA.

Karp, R. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer.

Lavnikevich, N. (2013). On the complexity of maximum clique algorithms: usage of coloring heuristics leads to the $\Omega(2^{n/5})$ algorithm running time lower bound.

Pittel, B. (1982). On the probable behaviour of some algorithms for finding the stability number of a graph. *Mathematical Proceedings of the Cambridge Philosophical Society*, 92:511–526.

San Segundo, P., Lopez, A., and Pardalos, P. M. (2016). A new exact maximum clique algorithm for large and massive sparse graphs. *Computers & Operations Research*, 66:81–94.

Zuckerman, D. (2006). Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 681–690, New York, NY, USA. Association for Computing Machinery.

Züge, A. and Carmo, R. (2018). On comparing algorithms for the maximum clique problem. *Discrete Applied Mathematics*, 247.