UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RODRIGO NOGUEIRA LIMA DAVID

HARD INSTANCES FOR THE MAXIMUM CLIQUE PROBLEM

FORTALEZA

2022

RODRIGO NOGUEIRA LIMA DAVID

HARD INSTANCES FOR THE MAXIMUM CLIQUE PROBLEM

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Victor Almeida Campos

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

D275h David, Rodrigo Nogueira Lima.
Hard instances for the Maximum Clique problem / Rodrigo Nogueira Lima David. – 2022.
25 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Computação, Fortaleza, 2022.
Orientação: Prof. Dr. Victor Almeida Campos.

1. Teoria dos Grafos. 2. Clique Máxima. 3. Branch and Bound. 4. Coloração de Grafos. I. Título.
CDD 005

RODRIGO NOGUEIRA LIMA DAVID

HARD INSTANCES FOR THE MAXIMUM CLIQUE PROBLEM

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Victor Almeida Campos (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Julio Cesar Silva Araujo
Universidade Federal do Ceará (UFC)

Prof. Dr. Fabricio Siqueira Benevides
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Agradeço ao Prof. Victor Campos, por me orientar em minha pesquisa que levou à escrita desta monografia. Agradeço também ao Prof. Manoel Campêlo por me iniciar no mundo da pesquisa em algoritmos e no ParGO. Agradeço também a todos os professores da UFC pela qualidade no ensino que foi crucial para minha formação acadêmica.

Agradeço à minha família pelo suporte que sempre me deram e pelo investimento em mim. Agradeço a todos os amigos que fiz na faculdade e especialmente à Amanda, minha namorada e melhor amiga, por ter me ajudado nos momentos mais difíceis e por celebrar comigo os momentos de alegria.

Agradeço a Deus pelo dom da vida e por tudo que me foi concedido durante minha caminhada.

RESUMO

Uma clique em um grafo é um conjunto de vértices no qual todos são dois a dois adjacentes. O problema da CLIQUE MÁXIMA é um problema clássico de otimização em Teoria dos Grafos no qual deseja-se encontrar uma clique máxima em um grafo de entrada. Embora existam diversos resultados de dificuldade teórica sobre o problema, experimentos computacionais mostram que ele parece ser mais fácil do que o esperado. Neste trabalho, nós abordamos uma classe de algoritmos de *Branch and Bound* para o problema e como eles tornam CLIQUE MÁXIMA mais fácil na prática. Também, analisamos a correlação entre cliques e colorações e a diferença de dificuldade na enumeração dessas estruturas. Além disso, propomos instâncias cuja solução via enumeração é mais complexa em comparação com a média. Por fim, analisamos uma família de instâncias de pior caso para uma classe mais específica de algoritmos e propomos um pré-processamento que torna o tempo de solução dessas instâncias polinomial, além de uma construção não-determinística de grafos imunes a este pré-processamento.

Palavras-chave: Teoria dos Grafos. Clique Máxima. Branch and Bound. Coloração de Grafos.

ABSTRACT

A clique in a graph is a set of vertices in which all of its elements are pairwise adjacent. The MAXIMUM CLIQUE problem is a classic optimization problem in Graph Theory in which the objective is to find a maximum clique in a input graph G. Despite the existence of several theoretical hardness results for this problem, several experiments paint it as easier than one would expect. In this work, we approach a class of *Branch and Bound* algorithms for this problem, and how they make MAXIMUM CLIQUE often easier in practice. Furthermore, we analyze the correlation between cliques and colorings and the hardness difference in enumerating these structures. Besides that, we present instances whose solution via enumeration is more complex when compared to the average. Finally, we analyze a family of worst case instances for a more specific class of algorithms and propose a pre-processing that makes it possible for these instances to be solved in polynomial time, as well as a randomized construction that builds instances that are immune to this pre-processing.

Keywords: Graph Theory. Maximum Clique. Branch and Bound. Graph Coloring.

LIST OF FIGURES

CONTENTS

1 INTRODUCTION

1.1 Hardness results

A clique in a graph is a subset of vertices in which any two elements are adjacent. In the MAXIMUM CLIQUE problem, the objective is to find the largest clique in a given graph G. Its decision version, denoted by CLIQUE, consists in deciding, given an input graph G and a integer k, if G has a clique of size at least k; it is a fundamental NP-complete problem (KARP, 1972) and also W[1]-complete under the natural parametrization over k (DOWNEY; FELLOWS, 1995). Furthermore, besides being NP-hard, MAXIMUM CLIQUE is also $n^{1-\varepsilon}$-inapproximable in polynomial time (unless P = NP) for any $\varepsilon > 0$, where n denotes the number of vertices in the input graph (ZUCKERMAN, 2006).

Although MAXIMUM CLIQUE is drawn intractable by these theoretical hardness results, several authors report exact algorithms that are able to tackle large instances of practical interest for several application domains in reasonable time (CARRAGHAN; PARDALOS, 1990; KONC; JANEŽIČ, 2007; JR. *et al.*, 2010; SAN SEGUNDO *et al.*, 2016). This interesting contrast has been studied in (CARMO; ZÜGE, 2018), where the authors focus on a widely used class of Branch and Bound (B&B) algorithms and prove that their time complexity is highly concentrated around the sub-exponential (quasi-polynomial, in fact) $n^{\mathcal{O}(\lg n)}$ growth rate.

1.2 Relation with Vertex Colorings

A related problem is the MINIMUM VERTEX COLORING, in which the objective is to partition the vertices of a graph in the least amount of parts in such a way that no adjacent vertices are on the same part. This is also a classic NP-hard problem, its decision version also being one of the 21 original Karp's NP-complete problems (KARP, 1972). Moreover, it is also $n^{1-\varepsilon}$-inapproximable in polynomial time for any $\varepsilon > 0$, unless P = NP (ZUCKERMAN, 2006).

In 1992, the second DIMACS Implementation Challenge was held to encourage the development of algorithmic results on three NP-hard problems, namely SATISFIABILITY, MINIMUM VERTEX COLORING and MAXIMUM CLIQUE. Comparing the selected papers, the conclusion was that the coloring problem was much harder than the clique one, although both are similar difficulty-wise (JOHNSON; TRICK, 1996); in this work, we offer an explanation to the this phenomenon.

In (CARMO; ZÜGE, 2018), the authors prove an upper bound on the expected number of cliques in a random graph, which ties the average complexity of the best known algorithms. Here, we present the asymptotic behavior of the expected number of colorings in random graphs and compare it to the number of cliques to explain the hardness disparity between the coloring and the clique problems. Moreover, we use a polynomial reduction from MINIMUM VERTEX COLORING to MAXIMUM CLIQUE to obtain a family of graphs with $n^{\Theta(n^{3/5-\varepsilon})}$ cliques on average, for any $\varepsilon \in (0, 1/10]$, and these instances should be harder to solve than ordinary random graphs given the higher expected number of cliques.

1.3 Algorithms with an upper bound based on the chromatic number

Even though the worst case for MAXIMUM CLIQUE is not expected to be solved in polynomial time, presenting instances that attend to this complexity is a non-trivial issue. In (LAVNIKEVICH, 2013), the author focus on an even more restricted class of algorithms, B&B algorithms with an upper bound based on the chromatic number. This particularization is justified, since among the best algorithms for MAXIMUM CLIQUE the vast majority uses (directly or indirectly) an estimate of this upper bound. The author, then, introduces a family of graphs that require $\Omega(2^{n/5})$ steps to be solved by any such algorithm, where n is the size of the graph. These instances, however, are artificial, in a sense that it would be very unlikely to find one of those graphs in a real problem and, besides that, its recognition is straightforward polynomial.

We go a little further and present a pre-processing heuristic that reduces solution time for graphs that are disconnected or whose complement is and prove that it enables the mentioned algorithms to solve Lavnikevich's instances in polynomial time. Finally, we describe a randomized construction based on Lavnikevich's family such that the final graph still exhibits worst case behavior, but is also unaffected by the proposed pre-processing.

1.4 Structure of this work

In Chapter 2, we give some basic definitions that are essential to the understanding of this work. In Chapter 3, we introduce the B&B algorithms that we use throughout the text. Moreover, we present some of their basic properties and what is known about their average performance. In Chapter 4, we present correlations between the MAXIMUM CLIQUE and the MINIMUM VERTEX COLORING problems. Moreover, we prove a theorem on the expected

number of colorings of random graphs and show that it is possible to build instances with more cliques than average using this result together with a previous construction in the literature.

In Chapter 5, we shift our attention to a specific type of B&B algorithms that are among the best options for handling MAXIMUM CLIQUE in practice. We give some basic definitions and properties of these algorithms and present some important known results. Furthermore, we analyze a family of graphs that are reported as exponential running time inducing instances for the these algorithms. We also propose a pre-processing heuristic that is able to break the said instances into smaller problems in a way that they can all be solved in polynomial time. Furthermore, we present a non deterministic construction based on the family analyzed that also induces exponential running time for the fore mentioned algorithms and prove that it is sufficient to ensure that the final graph will survive our heuristic.

Finally, in Chapter 6 we present a few concluding remarks about the problem and future research topics.

2 NOTATION

A *graph* G is defined by a pair (V, E) of vertices and edges, where an edge is a set of two vertices. Two vertices u, v are said *adjacent* (or neighbors) if $uv \in E(G)$. The set of all neighbors of a vertex v in a graph is denoted by $N(v)$. A *clique* in G is a set of vertices in which any two of them are adjacent. The size of the largest clique in G is denoted by $\omega(G)$. An *independent set* in G is a set of vertices in which no two of them are adjacent. The size of the largest independent set in G is denoted by $\alpha(G)$. A (proper) *vertex coloring* in G is a partition of $V(G)$ in which every part is an independent set in G. The size of the smallest coloring in G is the chromatic number of G and is denoted by $\chi(G)$. The *chromatic gap* of a graph G is the difference $\chi(G) - \omega(G)$. A *subgraph* H of G is a graph where $V(H) \subseteq V(G)$ and $E(H) \subseteq V(G)$, and it is denoted by $H \subseteq G$. H is said to *span* G if $V(H) = V(G)$, and we call H a *spanning* subgraph. H is said to be induced by $S \subseteq V(G)$ if $V(H) = S$ and $E(H) = E(G) \cap \binom{S}{2}$, and we call H a *induced* subgraph.

A *path* on n vertices, denoted by P_n, is a graph in which its vertices can be ordered as $(v_1, \ldots, v_n)$ and $v_i v_{i+1} \in E(P_n)$ for $1 \leqslant i \leqslant n-1$, but no other edge exists. A *cycle* on n vertices, denoted by C_n, is a graph in which its vertices can be ordered as $(v_1, \ldots, v_n)$ and $v_i v_{i+1} \in E(C_n)$ for $1 \leqslant i \leqslant n-1$, but also $v_n v_1 \in E(C_n)$ and no other edge exists. A graph G is said to be *connected* if between any two vertices u and v there is an induced path in G starting in u and ending in v. A *component* G_i of G is a subgraph that is both connected and maximal, i.e., for any connected $H \subseteq G$ ($H \neq G_i$), we have $G_i \not\subseteq H$.

The *complement* of a graph G, denoted by $\overline{G}$, is the graph with the same vertices as G such that $E(\overline{G}) = \{uv \in \binom{V}{2} \mid uv \notin E(G)\}$. The *join* of two graphs G and H, denoted by $G \vee H$, is a copy of G together with a copy of H in which every vertex in G is adjacent to every vertex in H.

A *random graph* is a graph whose structure is not deterministic. In the *binomial model*, denoted by $\mathcal{G}(n, p)$, a graph has n vertices and every possible edge occurs independently with probability p. If a graph G belongs to the $\mathcal{G}(n, p)$ model, we say $G \sim \mathcal{G}(n, p)$. In the *uniform model*, detoned by $\mathcal{G}(n, m)$, a graph has n vertices and m edges chosen uniformly at random from the collection of all possible edges. Similarly, if a graph G belongs to the $\mathcal{G}(n, m)$ model, we say $G \sim \mathcal{G}(n, m)$. Finally, we say an event X_n occurs *with high probability* (w.h.p.) if $\mathbb{P}(X_n) \to 1$ when $n \to \infty$.

3 A BRANCH AND BOUND FRAMEWORK

We will study here a specific family of B&B algorithms. First, we define a *clique subinstance* for a graph G, which we call simply a *subinstance* for G, as a pair (Q, K) of disjoints subsets of $V(G)$ where Q is a clique and every vertex in K is adjacent to every vertex in Q. In each subinstance (Q, K), the objective is to find the largest clique C of G such that $Q \subseteq C \subseteq Q \cup K$. Note that the instance G for MAXIMUM CLIQUE corresponds to the subinstance $(\emptyset, V(G))$.

Intuitively, any vertex is either in the largest clique containing Q or not, we will enumerate the possibilities as follows. If a subinstance (Q, K) of G is not already solved, then $K \neq \emptyset$, a *pivot* vertex $v \in K$ is chosen and this subinstance branches into two others:

a) $(Q \cup \{v\}, K \cap N(v))$, which considers all cliques that contain v (and do not contain any vertices that are not adjacent to v);

b) $(Q, K \setminus \{v\})$, which considers all cliques that *do not* contain v (and may or may not contain some of its neighbors).

Note that no bounding rule is defined yet. We call any algorithm that implements this scheme (and possibly a bounding rule) a *standard algorithm*. A standard algorithm that does not have a bounding rule simply enumerates all feasible solutions. Figure 1 illustrates the branching of a node in a standard algorithm.

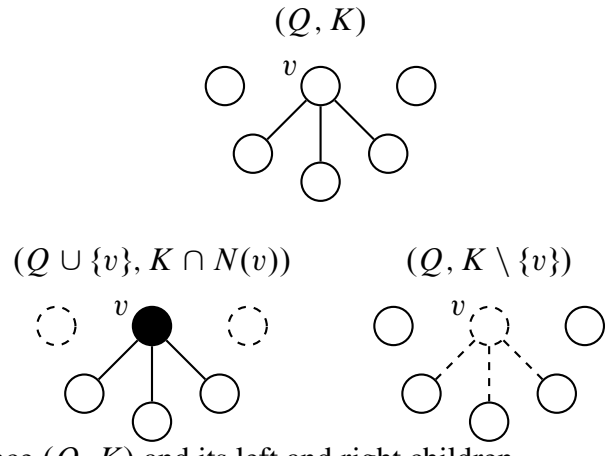


Figure 1 – A subinstance (Q, K) and its left and right children.

Definition 1. Given a graph G, a *clique search* tree T of G is a binary tree such that:

a) The root of T is the subinstance $(\emptyset, V(G))$;

b) A subinstance (Q, K) is a leaf of T only if $K = \emptyset$;

c) If a subinstance (Q, K) is not a leaf, there is a vertex $v \in K$ such that the left and right children of the subinstance are $(Q \cup \{v\}, K \cap N(v))$ and $(Q, K \setminus \{v\})$ respectively.

A clique search tree of a graph G can be seen as the result of an execution of a standard algorithm with no bounding rule. We say an execution $\mathcal{E}$ of a standard algorithm is *contained* in a clique search tree T of a graph G if the instances analyzed in $\mathcal{E}$ induce a connected subgraph in T, this subgraph contains the root of T and every pivot choice is the same in any subinstance of $\mathcal{E}$ and its equivalent node in T. Using this notation, Carmo and Züge prove the following.

Proposition 1 (Carmo and Züge (2018))**.** *Let G be a graph and $\mathcal{C}$ be the set of all its cliques. If T is a clique search tree of G, then T has $2|\mathcal{C}| - 1$ nodes. Furthermore, each execution of a standard algorithm for the Maximum Clique Problem on G is contained in some clique search tree of G.*

Note that, by definition, in every leaf $(Q, \emptyset)$ of a clique search tree T of G, Q is a (not necessarily maximal) clique in G. Moreover, each clique Q of G is represented in exactly one leaf $(Q, \emptyset)$ of T, because each vertex $v \in V(G)$ will be chosen as pivot at some point (just consider the rightmost path from the root to a leaf), so when the first vertex $u \in Q$ is chosen as pivot, follow the node's left child and keep following left whenever the chosen pivot v is in Q, but follow the right child if $v \notin Q$. When the last vertex of the clique is selected as pivot, the rightmost leaf in the node's subtree is $(Q, \emptyset)$.

The authors approach the gap between theoretical and empirical hardness results regarding Maximum Clique by analyzing the average behavior of B&B algorithms through clique search trees. Indeed, by Proposition 1, the size of a clique search tree is an upper bound on the number of instances considered by any standard algorithm. With that in mind, they prove the following lemma.

Lemma 2 (Carmo and Züge (2018))**.** *For any $n \in \mathbb{N}$ and constant $p \in (0, 1)$, if $G \sim \mathcal{G}(n, p)$, then the average number of cliques in G is $n^{\mathcal{O}(\lg n)}$.*

Now, as the size of any clique tree grows as fast as the number of cliques in the graph, their size is, on average, quasi-polynomial. This explains why standard algorithms seem to be much faster than they should, for if the time to process a node it at most quasi-polynomial, the final execution time will still be quasi-polynomial and far away from the worst case.

In (CHVÁTAL, 1977), the author defined a structure that is very similar to a clique search tree. The *f-driven* tree is a binary tree whose nodes are subinstances for the MAXIMUM INDEPENDENT SET and the children of a node is defined in an analogous way to nodes in the clique search tree. With this notion, Pittel proved that the size of a f-driven tree is quasi-polynomial with high probability (PITTEL, 1982). Adapting this result to clique search trees, Carmo and Züge proved the following.

Theorem 3 (Carmo and Züge (2018))**.** *For any $n \in \mathbb{N}$ and constant $p \in (0, 1)$, if $G \sim \mathcal{G}(n, p)$, then any clique search tree of G has size $n^{\mathcal{O}(\lg n)}$ w.h.p.*

Theorem 3 strengthens the explanation of the easiness in practice of MAXIMUM CLIQUE, in the sense that not only in average the time complexity is quasi-polynomial, but almost always.

These results raises the following question: "If almost always a maximum clique is found in quasi-polynomial time, when does it take more time to find it?" The natural approach is to find instances that induce worse scenarios than average to algorithms. Our first result concerns a family of instances that have asymptotically more cliques and, thus, are harder to enumerate.

4 USING VERTEX COLORINGS TO BUILD CLIQUES

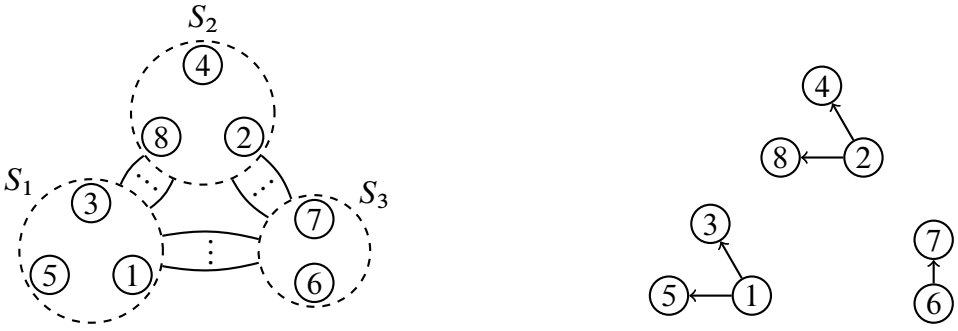
In (CAMPÊLO *et al.*, 2008), the authors introduce a linear integer programming formulation for MINIMUM VERTEX COLORING. This model, called the "Asymmetric Representatives Model" establishes a connection between colorings and independent sets (and, consequently, between cliques as well.)

Given a graph G and a linear order $\prec$ over its vertices, a coloring $\mathcal{S}$ can be expressed by representatives, one for each color class. The representative v_i of the color class S_i is the minimum vertex with respect to $\prec$ in the class, i.e., $v_i \prec u, \forall u \in S_i \setminus \{v_i\}$. Every vertex is either a representative or is represented by exactly one other vertex. This model was explored by Cornaz and Jost, who prove the following theorem.

Theorem 4 (Cornaz and Jost (2008))**.** *For any graph G with n vertices and $\overline{m}$ non-edges, we can build a graph G^* together with a bijection f from cliques of G^* to colorings of G such that for each clique C of G^* $f(C)$ is a coloring of G with $n - |C|$ colors.*

G^* has a vertex for each non-edge in G. If $uv \notin E(G)$, then u and v could be in the same color class, and if $u \prec v$, then u could represent v. Given $uv, uw \notin E(G)$ such that $u \prec v$ and $u \prec w$, if $vw \in E(G)$, then u cannot represent both v and w and the vertices uv and uw in G^* are not adjacent. Moreover, if $uv, wv \notin E(G)$, $u \prec v$ and $w \prec v$, then u and w cannot both represent v and the vertices uv and wv in G^* are not adjacent.

Now, note that a coloring of G defines uniquely a sets of its representatives and who they represent, and the converse is also true. Figure 2 shows the relation between a coloring and its representatives.



(a) Coloring $\mathcal{S}$ of a graph G. (b) Representatives of $\mathcal{S}$.

Figure 2 – Modeling a coloring through its representatives, where the arc from a vertex u to a vertex v indicates that the former represents the latter.

In other words, a set of representative vertices is defined and so are the non-edges connecting a representative and one of its represented vertices. By the construction of G^*, the set of vertices $\{uv \in V(G^*) \mid u \text{ represents } v\}$ is a clique in G^*.

With that in mind, we search for graphs with a high enough number of colorings in order to apply the bijection and obtain a family with a high number of cliques. It turns out that random graphs have, on average, enough colorings for this purpose.

4.1 Counting colorings in random graphs

Following the observations given in (JOHNSON; TRICK, 1996) about the hardness disparity between MAXIMUM CLIQUE and MINIMUM VERTEX COLORING, we study the hardness to enumerate colorings on average, in a similar fashion to what was done to cliques. In order to obtain this average, we shall consider random graphs, namely graphs in the $\mathcal{G}(n, p)$ model. Our first main result is the following.

Theorem 5. *For any $\alpha \in [0, 1/3)$ and $n \in \mathbb{N}$, if $p \leqslant 1 - n^{-\alpha}$, then the expected number of colorings of a graph $G \sim \mathcal{G}(n, p)$ is $n^{\Theta(n)}$.*

Proof. We note that the number C of proper colorings is bounded from above by the total number of partitions of $V(G)$, which is equal to the Bell number B_n. We prove the upper bound using the following bound of B_n from (BEREND; TASSA, 2010):

$$B_n < \left(\frac{0.792n}{\ln(n+1)}\right)^n = n^{\mathcal{O}(n)}.$$

Now, let C_k be the number of different k-colorings of G. To prove the lower bound, we note that $C \geqslant C_k$, for any k. We consider the case when

$$k = \left\lfloor \frac{1}{2\alpha} \frac{\ln(1/q)}{\ln n} n \right\rfloor, \quad \text{where } q = 1 - p$$

and we note that $k \leqslant n/2$ when $q \geqslant n^{-\alpha}$.

Following a similar approach of that on (GRIMMETT; MCDIARMID, 1975), let $d = \lfloor n/k \rfloor$ and note that $kd \leqslant n < k(d+1)$. We say that a partition of $V(G)$ into k parts is *balanced* if each part of the partition has size d or $d+1$. Let $N(r)$ be the number of balanced partitions with k parts of a set of size r and note that $N(n) \geqslant N(kd)$. We remark that C_k is at least the number of k-colorings of G with k parts which are balanced. Hence,

$$\mathbb{E}[C_k] \geqslant N(n) q^{k\binom{d+1}{2}} \geqslant N(kd) q^{\frac{1}{2}kd(d+1)} \geqslant \frac{(kd)!}{(d!)^k\, k!} q^{\frac{1}{2}(n^2/k+n)}.$$

Using Stirling's approximation for factorials (ROBBINS, 1955), we have

$$\begin{aligned}
\frac{(kd)!}{(d!)^k\, k!} &> \frac{\sqrt{2\pi kd}}{\left(\sqrt{2\pi d}\right)^k \sqrt{2\pi k}} \frac{\left(\frac{kd}{e}\right)^{kd}}{\left(\frac{k}{e}\right)^k \left(\frac{d}{e}\right)^{kd}} \frac{\exp\left(\frac{1}{12kd+1}\right)}{\exp\left(\frac{1}{12k}+\frac{k}{12d}\right)} \\
&= \frac{1}{(2\pi)^{k/2}\, d^{(k-1)/2}} k^{k(d-1)} \exp\left(k + \frac{1}{12kd+1} - \frac{1}{12k} - \frac{k}{12d}\right) \\
&\geqslant \frac{k^{k(d-1)+(k-1)/2}}{n^{(k-1)/2}} \exp\left(\frac{12kd-k}{12d} - \frac{k}{2}\ln(2\pi) + \frac{1}{12kd+1} - \frac{1}{12k}\right) \\
&= \exp\left((n-k)\ln n + \mathcal{O}(n)\right) \\
&= \exp\left(\tfrac{1}{2} n \ln n + \mathcal{O}(n)\right).
\end{aligned}$$

Finally, we get

$$\begin{aligned}
\mathbb{E}[C_k] &\geqslant \exp\left(\tfrac{1}{2} n \ln n + \mathcal{O}(n)\right) q^{n(n/k+1)/2} \\
&= \exp\left(\tfrac{1}{2} n \ln n + \mathcal{O}(n) - \left(\frac{n^2}{2k} + \frac{n}{2}\right)\ln(1/q)\right) \\
&= \exp\left(\tfrac{1}{2} n \ln n - \frac{n^2}{2k}\ln(1/q) - \frac{n}{2}\ln(1/q) + \mathcal{O}(n)\right) \\
&= \exp\left(\tfrac{1}{2} n \ln n - \alpha n \ln n - \tfrac{1}{2}\alpha n \ln n + \mathcal{O}(n)\right) \\
&= \exp\left(\tfrac{1}{2} n \ln n\,(1-3\alpha) + \mathcal{O}(n)\right) \\
&= \exp\left(\Omega\left(n \ln n\right)\right) \\
&= n^{\Omega(n)}.
\end{aligned}$$

which finishes the proof of the lower bound. □

Theorem 5 gives an intuition on the hardness difference between enumerating colorings and enumerating cliques, as there are $n^{\mathcal{O}(\lg n)}$ cliques versus $n^{\Theta(n)}$ colorings on graphs in the $\mathcal{G}(n, p)$ model with constant p and most state-of-the-art algorithms implement some kind of enumeration. Note that if $G \sim \mathcal{G}(n, m)$ instead and $m \leqslant \binom{n}{2}(1 - n^{-\alpha})$ the conclusion is still valid for any $\alpha \in [0, 1/3)$.

Proving that there are more colorings than cliques is the first step towards applying the bijection given by Theorem 4. In fact, G^* need not have the same number of vertices as G, so the number of cliques in G^* as a function of $|V(G^*)|$ has to be translated into a function of $|V(G)|$ and this could potentially harm the desired number of cliques too badly. Fortunately, this is not the case and the number of cliques in G^* is still asymptotically higher than that of G.

4.2 Instances with more cliques than average

We will apply the given bijection between colorings of a random graph G into cliques of G^* and analyze the expected number of cliques in the latter.

Corollary 6. *For any $\varepsilon \in (0, 1/10]$ and $n \in \mathbb{N}$, there is a random process to build a graph with n vertices and whose expected number of cliques is $n^{\Theta(n^{3/5-\varepsilon})}$.*

Proof. For any $0 < \varepsilon \leqslant 1/10$, let $\varepsilon' = 25\varepsilon/(9 - 15\varepsilon)$ so that $3/(5 + 3\varepsilon') = 3/5 - \varepsilon$ and note that $0 \leqslant \varepsilon' < 1/3$. Let $q = n^{-(1/3-\varepsilon')}$, now we can build our random graph $G \sim \mathcal{G}(n, m)$ where

$$\overline{m} = \binom{n}{2} q = \Theta(n^{5/3+\varepsilon'})$$

and expected number of colorings equal to

$$n^{\Theta(n)} = \overline{m}^{\Theta(\overline{m}^{3/5-\varepsilon})},$$

according to Theorem 5. Now, as there is a bijection between colorings of G and cliques of G^*, the expected number of cliques of the latter is equal to the expected number of colorings of the former. □

So, when p is constant, a graph $G \sim \mathcal{G}(n, p)$ on n vertices has $n^{\mathcal{O}(\lg n)} = 2^{\mathcal{O}(\lg^2 n)}$ cliques w.h.p, but a graph G^* on n vertices built by the described process on a random graph (with constant p) has $n^{\Theta(\sqrt{n})} = 2^{\Theta(\sqrt{n} \lg n)}$ expected number of cliques, because in this scenario $\overline{m} = \Theta(n^2)$. This means that our non deterministic instances should be harder to enumerate than ordinary random graphs given the expected higher number of cliques.

5 STANDARD ALGORITHMS WITH A CHROMATIC UPPER BOUND

We now turn our attention to a more specific type of standard algorithms.

5.1 Introducing a bounding rule

Recall that standard algorithms are enumerative but need not implement any kind of upper bound in order to avoid branching when it is not necessary. Introducing such bounds can be very beneficial to the algorithm's efficiency, provided the bound does not take up much time to be evaluated. A very common strategy to stop a node from branching in our scenario is comparing the size of the largest clique already found to the fewest number of colors needed to color the graph induced by a node, where a graph G induced by the node (Q, K) is $G[Q \cup K]$.

It is a well known fact that if a graph G has a clique Q, then any proper coloring of G uses at least $|Q|$ colors, as each vertex on Q must have its own color. Furthermore, if a node (Q, K) is such that $\chi(G[Q \cup K]) \leqslant |Q'|$ where Q' is an already found clique, then $\omega(G[Q \cup K]) \leqslant |Q'|$ and there is no real reason to keep branching after this node.

Note that computing the chromatic number of the graph induced by some instance is not trivial in general, so, in order to keep the upper bound feasible time-wise, a possibly non optimal number of colors is computed by some heuristic instead. When a standard algorithm applies this strategy to avoid unnecessary branching, we call it a *χ-bounded* algorithm. We remark that some of the best algorithms for MAXIMUM CLIQUE are χ-bounded. We now define a substructure of clique search trees.

Definition 2. Given a graph G and a clique search tree T of G, the *χ-pruned* subtree of T, denoted by T_χ, is the (unique) subtree of T such that

a) T_χ is minimal in size;

b) The node $(\emptyset, V(G)) \in T_\chi$;

c) In every leaf (Q, K) of T_χ, we have $\chi(G[Q \cup K]) \leqslant \omega(G)$.

The following result establishes a relation between χ-bounded algorithms and χ-pruned subtrees.

Proposition 7. *For any execution $\mathcal{E}$ of a χ-bounded algorithm on a graph G, there is a clique search tree T of G such that $\mathcal{E}$ is contained in T and the subinstances considered in $\mathcal{E}$ contain $V(T_\chi)$.*

Proof. Let $\Gamma(G)$ be the upper bound on the chromatic number used by the χ-bounded algorithm, k be the number of pivot choices made in $\mathcal{E}$ and T be the clique search tree of G in which its first k pivot choices are equal to the choices made in $\mathcal{E}$. It is clear that all nodes in $\mathcal{E}$ must be in T, or there would have been a subinstance generated by a pivot choice not made in T. Moreover, note that no subinstance in T generated after the kth choice of pivot will appear on $\mathcal{E}$, as all of its choices have already been used up. Therefore, $\mathcal{E}$ induces a subgraph of T that is, indeed, connected and, thus, it is contained in T. Now, note that if a T_χ node (Q, K) is not a leaf, then $\chi(G[Q \cup K]) > \omega(G) \geqslant |Q'|$ where Q' is the largest clique found so far and, because $\Gamma(G)$ is an upper bound on $\chi(G)$, $\Gamma(G[Q \cup K]) > |Q'|$ and (Q, K) is a node on $\mathcal{E}$ and is not a leaf, thus, $\mathcal{E}$ contains T_χ. □

So, essentially, the number of subinstances considered in an execution of a χ-bounded algorithm is bounded from above by the size of a clique search tree T and bounded from below by the size of the χ-pruned subtree T_χ.

5.2 Exponential running time inducing graphs

We now define the class of Lavnikevich graphs. This notion was introduced in (LAVNIKEVICH, 2013) and Figure 3 provides an example.

Definition 3. For any $n \equiv 0 \pmod 5$, the Lavnikevich graph on n vertices, denoted by L_n, is obtained by the graph join between $n/5$ C_5's.

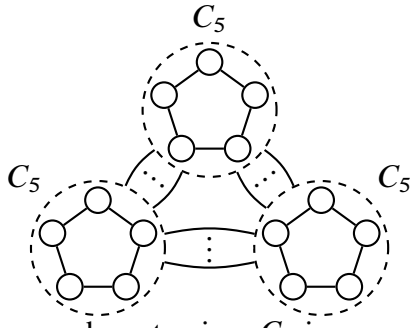


Figure 3 – The L_{15} graph, where each vertex in a C_5 is connected to all other vertices in the other 2 C_5's.

With a different notation, the author proves the following.

Proposition 8 (Lavnikevich (2013))**.** *The χ-pruned subtree of any clique search tree of L_n has size $\Omega(2^{n/5})$.*

But the number of instances considered in a χ-bounded algorithm is at least the size of a χ-pruned subtree that contains it by Proposition 7.

Corollary 9 (Lavnikevich (2013))**.** *Any χ-bounded algorithm on L_n has $\Omega(2^{n/5})$ time complexity.*

Indeed, even if the algorithm takes $\mathcal{O}(1)$ time to process each node, it still needs to process an exponential number of nodes. This essentially establishes Lavnikevich graphs as worst case instances for χ-bounded algorithms.

5.3 A pre-processing heuristic

A Lavnikevich graph is indeed hard to solve by χ-bounded algorithms, but its structure allows us to implement a simple pre-processing step that speeds up the solving process.

Proposition 10. *Given a graph G, we have*

$$\omega(G) = \max\{\omega(G_i) \mid G_i \text{ is a component of } G\}.$$

Proof. As there are no edges between components, any clique in G is contained in a single component, so its largest clique is also the largest clique in some component G_i. □

So, if a graph is given as input to a χ-bounded algorithm, instead of solving the problem in the graph as a whole, it can simply solve for each of its components and return the largest value. Note that if the input graph is connected, this heuristic is not useful.

Proposition 11. *Given a graph G that is the join between $G_1, \ldots, G_k$, we have*

$$\omega(G) = \sum_{i=1}^{k} \omega(G_i).$$

Proof. For any clique Q of G_i, each vertex of Q is adjacent to every vertex in a clique Q' of G_j. That means that $Q_1 \cup \cdots \cup Q_k$, where Q_i is a clique in G_i, is a clique of G and if we chose each Q_i to be a maximum clique in G_i, we have a maximum clique in G, as if there was a largest clique Q' in G, we would find a clique $Q' \cap V(G_i)$ that is largest than Q_i for some $i \in \{1, \ldots, k\}$. □

Now, if a graph G is such that $\overline{G}$ is disconnected and has $\overline{G}_1, \ldots, \overline{G}_k$ as its components, we can write $G = G_1 \vee \cdots \vee G_k$. This means that if a connected graph G is given

as input to a χ-bounded algorithm and the first heuristic fails but $\overline{G}$ is disconnected, one can split the problem in $\overline{G}$'s components and return the sum of the results.

These heuristics can be computed in polynomial time, as one could naively implement depth-first searches in G and $\overline{G}$ in time $\mathcal{O}(n^2)$. Besides that, if the pre-processing is applied before a χ-bounded algorithm with any L_n as the input graph, the problem would be greatly reduced. This is because $L_n = C_5 \vee \cdots \vee C_5$ where the join is done $n/5$ times and, thus, the algorithm has to solve MAXIMUM CLIQUE in the C_5 a linear number of times. The process would take polynomial time, which is a great improvement from the original exponential time needed.

As the final step of this work, we focus our attention in graphs that maintain the exponential solving time requirement for any χ-bounded algorithm *and* resist the pre-processing described above.

5.4 Worst case instances with connected complement

Essentially, we search for graphs that are connected and whose complement is also connected, so that none of the heuristics proposed can be applied. This can be achieved by a random perturbation on a L_n.

Theorem 12. *For any $n, d \in \mathbb{N}$, if G is a spanning subgraph of L_n where $\alpha(G) \leqslant 2$ and $\delta(G) \geqslant n - 1 - d$, then the χ-pruned subtree of every clique search tree of G has $\Omega(2^{n/(5d)})$ nodes.*

Proof. Note that $\omega(G) \leqslant 2n/5$ and $\alpha(G) \leqslant 2$, so if at most $n/5$ vertices have been discarded due to branch operations, then $\chi(G) \geqslant \frac{n-n/5}{2} = 2n/5$ and the chromatic gap is still linear. Now, each branching operation may discard 1 vertex if it excludes the pivot from the current clique and at most d if the pivot is included, as there are at most d vertices non adjacent to it. Therefore, more than $2n/(5d)$ branch operations are needed before the current search tree node becomes a leaf and, thus, the height of the search tree is at least $2n/(5d)$ and there are $\Omega(2^{n/(5d)})$ nodes to be analyzed. □

We want to remove edges from a L_n in order to ensure its complement is connected, of course this process should not disconnect L_n itself. We remark that L_n is $n-3$-regular and, thus, if G is a spanning subgraph of L_n with $\delta(G) \geqslant n-3$, then $G = L_n$. As $\overline{L_n}$ is disconnected,

we need $d \geqslant 3$ in order to ensure that the pre-processing step fails. When $d = 3$, we have the following.

Corollary 13. *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.*

Proof. We want to build a spanning subgraph G of L_n such that $\alpha(G) \leqslant 2$ and $\delta(G) \geqslant n - 4$ ($d = 3$ in this case), we initially set $G = L_n$ and proceed to remove some of its edges. Let H be a connected graph with $n/5$ vertices where each of its vertices represents a C_5 from $\overline{L_n}$. There are many possible choices for H, but we are only interested in those where $\Delta(H) \leqslant 5$. For each $uv \in E(H)$, let C_5^u and C_5^v be the C_5's in $\overline{L_n}$ associated with u and v, respectively, and $xy \in E(G)$ be some edge where $x \in C_5^u$, $y \in C_5^v$; we will add xy to $E(\overline{G})$ (and remove it from G) if this addition does not create a triangle in $\overline{G}$ nor increases $\Delta(\overline{G})$ above 3, this can be achieved if neither x nor y have already been chosen in another iteration. Note that no isolated C_5 in G will be created, as $\Delta(H) \leqslant 5$ and no vertex $v \in V(G)$ has to be chosen twice, so G's connectivity is kept. Moreover, $\overline{G}$ will be connected at the end of the process, because if there was no path from some C_5 to all others, no edge from H was added there, but such an addition between any two isolated C_5's cannot create a triangle. Also, no triangle occurs in $\overline{G}$, as no edge between vertices of the same C_5 is inserted and a vertex gains at most one neighbor (in a C_5 different from his) in the process, but a triangle demands a vertex with two neighbors in a different C_5 (or in two other different C_5's.) Therefore, as $\alpha(G) \leqslant 2$ and each vertex of G loses at most one edge (i.e., $\delta(G) \geqslant n - 4$), from Theorem 12 we conclude that the χ-pruned subtree of any clique search tree of G has size $\Omega(2^{n/15})$ for any B&B algorithm with a coloring upper bound and both G and $\overline{G}$ are connected. □

Finally, we remark that if one chooses H to be a tree such that $\Delta(H) \leqslant 5$, the resulting graph $\overline{G}$ satisfies the conditions we need, but if H has as many edges as possible (and each vertex has degree at most 5), then $\overline{G}$ will have more edges and it will be harder to exploit its strucutre. Thus, one could even allow H to have parallel edges in order to keep its vertices' degrees as close as possible to 5 and disturb both G's and $\overline{G}$'s structure as much as possible.

6 CONCLUSIONS

In this work, we presented results concerning both cliques and colorings in graphs. This topic of research is still very active and we believe our contribuitions provide some ground for future works. It is worth mentioning some points that could still be explored. For example, testing the graphs described in Corollary 6 with state-of-the-art solvers to verify the impact of the higher number of cliques; there are asymptotically more cliques than average, but their structure could somehow be explored by some heuristic applied in those solvers. Another point is testing the graphs constructed in Corollary 13 in solvers that implement χ-bounded algorithms and the described pre-processing in order to compare the results to those due to Lavnikevich. Finally, a theoretical aspect that can be pursued is improving Theorem 5 from the average case to a high probability scenario.

BIBLIOGRAPHY

BEREND, D.; TASSA, T. Improved bounds on bell numbers and on moments of sums of random variables. **Probability and Mathematical Statistics**, v. 30, n. 2, p. 185–205, 2010.

CAMPÊLO, M.; CAMPOS, V.; CORRÊA, R. On the asymmetric representatives formulation for the vertex coloring problem. **Discrete Applied Mathematics**, Elsevier, v. 156, n. 7, p. 1097–1111, 2008.

CARMO, R.; ZÜGE, A. On comparing algorithms for the maximum clique problem. **Discrete Applied Mathematics**, v. 247, 02 2018.

CARRAGHAN, R.; PARDALOS, P. An exact algorithm for the maximum clique problem. **Operations Research Letters**, v. 9, n. 6, p. 375–382, 1990. ISSN 0167-6377.

CHVÁTAL, V. Determining the stability number of a graph. **SIAM Journal on Computing**, v. 6, n. 4, p. 643–662, 1977.

CORNAZ, D.; JOST, V. A one-to-one correspondence between colorings and stable sets. **Operations Research Letters**, Elsevier, v. 36, n. 6, p. 673–676, 2008.

DOWNEY, R.; FELLOWS, M. Parameterized computational feasibility. In: CLOTE, P.; REMMEL, J. (Ed.). **Feasible Mathematics II**. Boston, MA: Birkhäuser Boston, 1995. p. 219–244.

GRIMMETT, G.; MCDIARMID, C. On colouring random graphs. **Mathematical Proceedings of the Cambridge Philosophical Society**, Cambridge University Press, v. 77, n. 2, p. 313–324, 1975.

JOHNSON, D.; TRICK, M. (Ed.). **Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, October 11–13, 1993**. Boston, MA, USA: American Mathematical Society, 1996. v. 26. ISBN 0821866095.

JR., E. D.; GARRETT, T.; BONA, L.; CARMO, R.; ZÜGE, A. Finding stable cliques of planetlab nodes. In: **2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)**. [S.l.]: IEEE, 2010. p. 317–322. ISBN 978-1-4244-7500-1.

KARP, R. Reducibility among combinatorial problems. In: **Complexity of computer computations**. [S.l.]: Springer, 1972. p. 85–103.

KONC, J.; JANEŽIČ, D. An improved branch and bound algorithm for the maximum clique problem. **MATCH Communications in Mathematical and in Computer Chemistry**, Jun. 2007.

LAVNIKEVICH, N. **On the Complexity of Maximum Clique Algorithms: usage of coloring heuristics leads to the $\Omega(2^{n/5})$ algorithm running time lower bound**. [S.l.]: arXiv, 2013.

PITTEL, B. On the probable behaviour of some algorithms for finding the stability number of a graph. **Mathematical Proceedings of the Cambridge Philosophical Society**, v. 92, p. 511–526, Nov. 1982. ISSN 1469-8064.

ROBBINS, H. A remark on stirling's formula. **The American mathematical monthly**, JSTOR, v. 62, n. 1, p. 26–29, 1955.

SAN SEGUNDO, P.; LOPEZ, A.; PARDALOS, P. A new exact maximum clique algorithm for large and massive sparse graphs. **Computers & Operations Research**, v. 66, p. 81–94, 2016. ISSN 0305-0548.

ZUCKERMAN, D. 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**. New York, NY, USA: Association for Computing Machinery, 2006. (STOC '06), p. 681–690. ISBN 1595931341.